AIX версии 7.2

Управление операционной системой и устройствами

AIX версии 7.2

Управление операционной системой и устройствами

Примечание

Перед началом работы с этим изданием и описанным в нем продуктом ознакомьтесь с информацией, приведенной в разделе “Примечания” на стр. 653.

Данное издание относится к AIX версии 7.2, а также ко всем последующим выпускам и модификациям, если в соответствующих изданиях не будет оговорено обратное.

Содержание

Об этом документе

Этот документ содержит исчерпывающую информацию для пользователей и системных администраторов, которая позволит выбрать правильные опции при выполнении таких задач, как резервное копирование и восстановление системы, управление физической и логической памятью, настройка размера пространства подкачки и т.д. Он содержит исчерпывающие сведения о выполнении таких задач, как управление локальными томами, памятью и ресурсами. Из этого раздела пользователи системы могут узнать, как выполнять такие задачи, как запуск команд, обработка процессов, файлов и каталогов, а также работа с печатью.

Кроме того, в этом разделе рассмотрены такие вопросы, как создание пространства подкачки и изменение его размера, управление виртуальной памятью, резервное копирование и восстановление системы, работа с физическими и псевдоустройствами, а также применение Контроллера ресурсов системы (SRC), защита файлов, применение носителей памяти, настройка файлов среды и создание сценариев оболочки. Данный документ можно найти и на компакт-диске документации, который поставляется вместе с операционной системой.

Выделение текста

В данном документе применяются следующие специальные обозначения:

Полужирный шрифт	Полужирным шрифтом выделены команды, процедуры, ключевые слова, имена файлов, структуры, каталоги и прочие объекты системы с предопределенными именами. Кроме того, полужирным шрифтом выделены названия графических объектов, выбираемых пользователем - кнопок, меток, значков и т.д.
Курсив	Курсивом выделены параметры, фактические имена и значения, которые указываются пользователем.
`Непропорциональный шрифт`	Непропорциональным шрифтом выделены примеры данных, текст, появляющийся на экране компьютера, примеры программного кода, системные сообщения и вводимая пользователем информация.

Учет регистра символов в AIX

В операционной системе AIX учитывается регистр символов, т.е. различаются прописные и строчные буквы. Например, с помощью команды **ls** можно просмотреть список файлов. Если ввести `LS`, то будет выдано сообщение о том, что команда `не найдена`. Аналогично, имена файлов **FILEA**, **FiLea** и **filea** считаются разными, даже если эти файлы расположены в одном каталоге. Для того чтобы избежать ошибок при выполнении операций, всегда проверяйте регистр символов.

ISO 9000

При разработке и производстве данного продукта использовались зарегистрированные системы ISO 9000.

Управление операционной системой и устройствами

Из этого раздела администраторы и пользователи системы могут узнать, как выполнять такие задачи, как запуск команд, обработка процессов, файлов и каталогов, создание резервной копии и восстановление системы, а также работа с печатью.

Кроме того, в этом разделе рассмотрены такие вопросы, как создание пространства подкачки и изменение его размера, управление виртуальной памятью, резервное копирование и восстановление системы, работа с физическими и псевдо-устройствами, а также применение Контроллера ресурсов системы (SRC), защита файлов, применение носителей памяти, настройка файлов среды и создание сценариев оболочки. Данный раздел можно найти и на компакт-диске документации, который поставляется вместе с операционной системой.

Новое в управлении операционной системой и устройствами

Ознакомьтесь с дополнениями и существенными изменениями в группе разделов Управление операционной системой и устройствами.

Как узнать об изменениях и добавлениях

В этом файле PDF дополнения и изменения могут обозначаться символами вертикальной черты (|) в левом поле.

Октябрь 2017 года

Ниже приведено краткое описание изменений, внесенных в разделы из этой книги:

- Добавлен раздел “Поддержка восстановления памяти для структуры логических томов” на стр. 389.
- Обновлена информация о USB-устройствах, поддерживаемых в операционной системе AIX, в разделе “Поддержка USB-устройств” на стр. 575.

Июнь 2017 года

Ниже приведено краткое описание изменений, внесенных в разделы из этой книги:

- Добавлена информация о мониторинге статистики в раздел “Мониторинг статистики кэша” на стр. 584.
- Добавлена информация о команде **lsmpio** в раздел “Управление совместимыми с MPIO устройствами” на стр. 540.

Управление операционной системой

С помощью команд можно управлять запуском системы, резервным копированием системы, завершением работы системы, управлять средами и оболочками системы, ресурсами системы и другими различными частями AIX.

Лицо, отвечающее за управление операционной системой, в терминологии UNIX обычно называется системным администратором. Однако, задачи системного администратора не ограничиваются только управлением системой. Это, а также другие упомянутые здесь руководства, призваны помочь системным администраторам в выполнении их многочисленных обязанностей.

Данная операционная система содержит специальные средства управления системой, созданные для упрощения работы системного администратора и улучшения защиты.

Доступные интерфейсы управления системой

Кроме командной строки данная операционная система поддерживает интерфейсы SMIT.

Ниже описаны интерфейсы SMIT:

- Инструмент управления системой (SMIT). Это пользовательский интерфейс на основе меню, автоматически создающий и запускающий команды, соответствующие выбранным вами опциям.

  С помощью SMIT вы можете выполнять следующие задачи:
  - Устанавливать, обновлять и обслуживать программное обеспечение
  - Настраивать устройства
  - Настраивать дисковые накопители, создавая группы томов и логические тома.
  - Создавать и расширять файловые системы и пространство подкачки
  - Управлять пользователями и группами
  - Настраивать сети и средства связи
  - Управлять печатью
  - Устранять неполадки
  - Планировать задания
  - Управлять системными ресурсами и нагрузкой
  - Управлять различными системными средами
  - Управлять системными данными кластера
- Объектно-ориентированный графический пользовательский интерфейс, поддерживает те же задачи по управлению системой, что и программа SMIT, однако упрощает управление системой, обеспечивая:
  - Уменьшение количества ошибок пользователя с помощью проверки ошибок и конструкции окон
  - Пошаговые инструкции по выполнению сложных задач
  - Дополнительные опции для опытных администраторов
  - Визуализацию сложных данных и взаимосвязей между системными объектами
  - Контроль за работой системы и предупреждение системного администратора о наступлении определенных событий
  - Контекстную справку, обзоры, советы и ссылки на электронные документы

Данные программного продукта

В базе данных Реестра программного обеспечения (SWVPD) хранится информация о программных продуктах и их компонентах.

SWVPD представляет собой набор команд и классов Администратора объектных данных (ODM). Команды SWVPD позволяют пользователю запросить (**lslpp**) и проверить (**lppchk**) информацию об установленных программных продуктах. Объектные классы ODM задают диапазон и формат этой информации.

Команда **installp** с помощью ODM добавляет в базу данных SWVPD следующую информацию:

- Имя установленного программного продукта
- Версию программного продукта
- Номер выпуска программного продукта, отражающий изменения его внешнего программного интерфейса.
- Уровень модификации программного продукта, отражающий изменения, не влияющие на его внешний программный интерфейс.
- Уровень исправления программного продукта, отражающий незначительные изменения, которые будут включены в следующую модификацию.
- Поле идентификатора исправления
- Имена, контрольные суммы и размеры файлов, составляющих программный продукт или компонент

- Состояние установки программного продукта: применяется, применен, фиксируется, фиксирован, отклонен или поврежден.

Обновления операционной системы

Пакет с операционной системой разбит на наборы логически связанных файлов. Каждый набор файлов можно устанавливать и обновлять независимо от других.

Уровень набора файлов определяется номером версии, выпуска, модификации и исправления (VRMF). При применении каждого обновления набора файлов изменяется номер исправления. При применении каждого пакета обслуживания AIX изменяется уровень модификации, а уровень исправления становится равен 0. Версия AIX, установленная изначально, (например AIX 6.1) называется базовой. Обновления операционной системы и ее функций существуют в виде пакетов обслуживания, технологических уровней, временных исправлений программ (PTF) или пакета обслуживания (группы PTF).

Пакеты обслуживания или технологические уровни

Пакеты обслуживания и технологические уровни предназначены для обновления базовой версии. При этом в номере VRMF изменяется часть, относящаяся к пакету обслуживания. Например первым пакетом обслуживания для AIX 6.1 может быть 6.1.1.0, вторым - 6.1.2.0 и т.д. Для просмотра списка пакетов обслуживания воспользуйтесь командой **oslevel -r**.

Для определения установленного пакета обслуживания или версии введите:

```
oslevel
```

Для поиска наборов файлов, которые необходимо обновить до заданного уровня пакета обслуживания или модификации (в данном примере - 6.1.1.0), введите следующую команду:

```
oslevel -l 6.1.1.0
```

Для того чтобы определить, установлен ли рекомендуемый пакет обслуживания (в данном примере - 6100-02):

```
oslevel -r 6100-02
```

Для поиска наборов файлов, которые необходимо обновить до уровня пакета обслуживания или модификации 6100-02, введите следующую команду:

```
oslevel -rl 6100-02
```

Для определения уровня пакета обслуживания или конкретного набора файлов (например, `bos.mp`), введите:

```
lslpp -L bos.mp
```

PTF PTF позволяют исправить конкретную ошибку, не дожидаясь выхода новой версии. Для различных программ может потребоваться установить некоторые или все доступные PTF.

Рекомендуемые пакеты обслуживания

Рекомендуемые пакеты обслуживания - это тщательно протестированные наборы PTF, рекомендуемые для установки.

Временные исправления

Временное исправление схоже с PTF, но оно обычно предлагается при отсутствии доступного PTF. Временные исправления выпускаются в случае, когда применение PTF приведет к обновлению системы до следующего уровня обслуживания, а пользователям требуется оставить систему на текущем уровне.

Для определения версии и уровня выпуска, пакета обслуживания, технологического уровня и уровня пакета обслуживания, а также для получения информации о том, какие наборы файлов должны быть обновлены для достижения определенного уровня, обратитесь к описаниям команд oslevel и lslpp в книге в книге *Справочник по командам*.

Загрузка системы

После загрузки Базовой операционной системы система выполняет достаточно сложный набор задач. Обычно эти задачи выполняются автоматически.

Перезагрузка системы выполняется в различных целях. Например, для завершения установки нового программного обеспечения, для сброса периферийных устройств, для выполнения регулярных задач по обслуживанию (таких, как проверка файловых систем) или восстановления после зависания или сбоя системы. Дополнительная информация об этих процедурах приведена в разделах:

Задачи, связанные с данной:

“Восстановление поврежденного загрузочного образа” на стр. 34
Ниже описана процедура идентификации и восстановления поврежденного загрузочного образа.

Управление запуском системы

При начальной загрузке или перезапуске системы выполняемые действия могут проходить по нескольким сценариям. Для завершения работы системы или для перезагрузки можно использовать команду shutdown или команду reboot. Если в системе работает несколько пользователей, следует использовать команду shutdown.

Перезагрузка работающей системы:

Так как в системе могут быть запущены процессы, требующие более мягкого завершения, чем позволяет выполнить метод **reboot**, то для всех систем предпочтительным является метод **shutdown**.

Существует два способа завершения работы и перезагрузки системы, **shutdown** и **reboot**. Если в системе работает несколько пользователей, следует использовать метод **shutdown**.

Процедура	Команда SMIT	Команда или файл
Перезагрузка системы с несколькими активными пользователями	`smit shutdown`	**shutdown -r**
Перезагрузка системы с одним активным пользователем	`smit shutdown`	**shutdown -r** или **reboot**

Удаленная перезагрузка системы, не отвечающей на запросы:

Функция удаленного перезапуска позволяет перезапускать систему через стандартный (интегрированный) системный порт.

Интегрированные *системные порты* POWER5 схожи с последовательными портами за исключением того, что системные порты доступны только особым поддерживаемым функциям.

Система перезапускается при получении через этот порт строки **reboot_string**. Данная функция полезна в тех случаях, когда система не отвечает на другие запросы, но может обслуживать прерывания системного порта. Удаленный перезапуск можно включить только для одного стандартного системного порта. Пользователи должны обеспечить внешнюю безопасность порта собственными средствами. Данная функция использует максимальный класс прерываний устройств и если UART (универсальный асинхронный приемопередатчик) не сможет быстро очистить буфер, то возможна потеря данных другими устройствами при переполнении их буферов. Рекомендуется применять данную функцию только для перезапуска системы, которая не отвечает ни на какие запросы и не позволяет войти в систему. Файловые системы *не будут* синхронизированы, а данные, которые не были записаны на диск, могут быть утеряны. Во избежание случайного перезапуска при включенной функции удаленного перезапуска настоятельно не рекомендуется использовать порт для каких-либо других целей, особенно для передачи данных.

Удаленным перезапуском управляют два атрибута стандартных системных портов.

reboot_enable

Разрешает перезапуск при получении через этот порт строки **reboot_string** с возможностью предварительного создания системного дампа.

```
no        - Удаленный перезапуск запрещен
reboot    - Удаленный перезапуск разрешен
dump      - Удаленный перезапуск разрешен, причем перед перезапуском
            необходимо создать системный дамп на основном устройстве дампа
```

reboot_string

Задает строку **reboot_string**, которую будет ожидать включенная функция удаленного перезапуска. Если удаленный перезапуск разрешен и на порт поступила указанная строка **reboot_string** , то передается символ **>**, указывающий на готовность системы к перезапуску. При получении символа 1 выполняется перезапуск; при получении любого другого символа перезапуск отменяется. Максимальная длина строки **reboot_string** составляет 16 символов, причем эта строка не должна содержать пробелов, двоеточий, символов равенства, пустых символов, символов перевода строки и Ctrl-\.

Удаленный перезапуск можно включить с помощью SMIT или командной строки. При работе с уже настроенным терминалом выберите **Системная среда -> Управление удаленным перезапуском**. При настройке нового терминала включить удаленный перезапуск можно с помощью меню **Создать терминал** или **Показать/изменить параметры терминала**. Для перехода к этим меню выберите **Устройства -> Терминалы**.

Для включения удаленного перезапуска из командной строки введите команду **mkdev** или **chdev**. Например, следующая команда разрешает удаленный перезапуск с созданием дампа и устанавливает строку перезапуска **ReBoOtMe** для **tty1**.

```
chdev -l tty1 -a remreboot=dump -a reboot_string=ReBoOtMe
```

Следующий пример разрешает в базе данных удаленный перезапуск для **tty0** с текущей строкой **reboot_string** (изменение вступит в силу после следующего перезапуска).

```
chdev -P -l tty0 -a remreboot=reboot
```

Если терминал используется как обычный порт, то для перед включением функции удаленного перезапуска необходимо ввести команду **pdisable**. Для возврата порта в нормальный режим работы введите команду **penable**.

Информация, связанная с данной:

Функциональные различия между системными и последовательными портами.

Загрузка с жесткого диска для обслуживания:

Загрузить систему в режиме обслуживания можно с жесткого диска.

Предварительные требования

В устройствах не должны находиться загрузочные съемные носители (магнитные ленты или диски CD-ROM). Кроме того, необходимо включить режим обслуживания, выполнив инструкции, приведенные в документации по аппаратному обеспечению.

Процедура

Загрузка системы с жесткого диска в режиме обслуживания:

1. Выключите и вновь включите систему или нажмите кнопку сброса.
2. Нажмите указанную в документации по аппаратному обеспечению комбинацию клавиш для загрузки в режиме обслуживания.
3. Система будет загружаться до некоторого момента, определяемого настройкой консоли. Если необходимо сохранить системный дамп, на консоли появится соответствующее меню.

Примечание:

a. Если консоль настроена неправильно, и при этом необходимо получить системный дамп, то система зависнет. Для получения дампа ее придется загрузить со съемных носителей.

b. Система автоматически создает дамп на указанном устройстве при нажатии кнопки сброса. Инструкции по изменению основного или дополнительного устройства дампа в работающей системе приведены в описании команды **sysdumpdev**.

4. Если системный дамп отсутствует, или если он уже был скопирован, то появятся инструкции по выполнению диагностики. Нажмите клавишу Enter, чтобы перейти к меню **Выбор функции**.
5. В меню **Выбор функций** можно выбрать режим диагностики или однопользовательский режим:

   `Однопользовательский режим:` Выберите эту опцию (опция 5), чтобы выполнить обслуживание в однопользовательской среде. Система продолжит загрузку и перейдет в однопользовательский режим. В этом режиме можно выполнять задачи обслуживания системы, для которых требуется автономный режим, и запускать, при необходимости, команду **bosboot**.

Информация, связанная с данной:

Создание системного дампа

Загрузка системы после сбоя:

В некоторых случаях необходимо загрузить систему, работа которой была завершена аварийно (например, после сбоя).

Для этой процедуры имеются следующие предварительные требования:

- Произошел аварийный сбой системы, и ее работу не удалось завершить с помощью обычной процедуры.
- Система выключена.

Ниже описаны основные действия по загрузке системы, которую не удалось восстановить после аварийного сбоя. Выполните следующие действия:

1. Убедитесь в том, что все аппаратное обеспечение и внешние устройства подключены должным образом.
2. Включите все периферийные устройства.
3. Следите за появляющимися на экране сообщениями о ходе автоматической диагностики аппаратного обеспечения.

   a. Если какие-либо из диагностических тестов завершаются неудачно, обратитесь к документации по данному устройству.

   b. Если все тесты пройдены успешно, включите системный блок.

Сброс неизвестного пароля root:

В этом разделе описано восстановление доступа к учетной записи root в том случае, когда пароль этой записи неизвестен или недоступен.

Эта процедура требует выключения системы. Во избежание потери данных следует планировать ее выполнение на время, соответствующее минимальной загрузке системы.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Установите в устройство носитель с той же версией и уровнем обслуживания, что и применяемая в данный момент операционная система.
2. Включите систему.
3. При появлении меню со значками или после выдачи двойного звукового сигнала нажимайте клавишу F1 до тех пор, пока не появится меню **Службы управления системой**.
4. Выберите опцию **Выбор операционной системы**.

5. Выберите опцию **Установить из**.
6. Укажите устройство, в котором находится носитель продукта и выберите опцию **Установить**.
7. Выберите версию AIX.
8. Определите системную консоль, нажав F1, а затем - Enter.
9. Укажите номер, соответствующий предпочитаемому языку, и нажмите Enter.
10. Выберите опцию **Запуск в режиме обслуживания для восстановления системы**, введите 3 и нажмите Enter.
11. Выберите **Доступ к корневой группе томов**. Будет показано сообщение с информацией о том, что в случае изменения корневой группы томов вы не сможете вернуться к меню установки без перезагрузки.
12. Введите ответ 0 и нажмите Enter.
13. Укажите номер показанной в списке корневой группы томов и нажмите Enter.
14. Выберите опцию **Получить доступ к группе томов и запустить оболочку**. Для этого укажите 1 и нажмите Enter.
15. Для сброса пароля root введите в приглашении # (символ номера) команду **passwd**. Например:

```
# passwd
Изменение пароля "root"
Новый пароль root:
Введите новый пароль еще раз:
```

16. Для записи всей информации из буфера на жесткий диск и перезагрузки системы введите следующую команду:

```
sync;sync;sync;reboot
```

При появлении окна входа в систему укажите имя пользователя root и пароль, заданный на шаге 15.

Информация, связанная с данной:

passwd, команда

Команда reboot

Начальная загрузка системы с планарным графическим адаптером:

Если в компьютере был установлен только планарный графический адаптер, то при последующем добавлении дополнительного графического адаптера произойдет следующее:

1. В систему будет добавлен новый графический адаптер и установлено программное обеспечение связанного с ним драйвера.
2. Система будет перезагружена, после чего произойдет одно из следующих событий:
   a. Если системная консоль определена как /dev/lft0 (эту информацию можно просмотреть командой **lscons**), то пользователь должен будет выбрать дисплей, который будет считаться системной консолью на время перезагрузки. Если пользователь выберет графический адаптер (а не терминал), то он также станет новым дисплеем по умолчанию. Если пользователь вместо устройства LFT выберет устройство терминала, то вход в систему не будет выполнен. После перезагрузки вновь появится терминальное окно входа в систему. Предполагается, что если пользователь добавляет в систему дополнительный графический адаптер, а системная консоль является устройством LFT, то пользователь не будет выбирать устройство терминала в качестве системной консоли.
   b. Если системная консоль определена как терминал, то при перезагрузке добавленный адаптер дисплея будет считаться дисплеем по умолчанию.

      Примечание: Терминал, применяемый в качестве системной консоли, останется системной консолью.
3. Если системная консоль определена как /def/lft0, то после перезагрузки система DPMS будет отключена, чтобы на экране в течение некоторого времени был показан текст для выбора системной консоли. Чтобы снова включить систему DPMS, еще раз перезагрузите систему.

Развертывание сценариев режима работы:

Сценарии режима работы позволяют пользователям запускать и завершать выбранные приложения, изменяя режим работы.

Поместите сценарии режима работы в каталог /etc/rc.d, соответствующий режиму:

- /etc/rc.d/rc2.d
- /etc/rc.d/rc3.d
- *Автоматическое монтирование*
- /etc/rc.d/rc5.d
- /etc/rc.d/rc6.d
- /etc/rc.d/rc7.d
- /etc/rc.d/rc8.d
- /etc/rc.d/rc9.d

/etc/rc.d/rc запускает сценарий запуска, находящийся в указанном каталоге, и выполняет его при изменении режима работы, сначала запуская сценарий останова приложения, а затем сценарий запуска приложения.

Примечание: Сценарии, имена которых начинаются с символа K, - это сценарии остановки, а сценарии, имена которых начинаются с символа S - это сценарии запуска.

Изменение файла /etc/inittab:

Для изменения файла доступны следующие четыре команды etc/inittab.

Добавление записей - команда mkitab

Для того чтобы добавить запись в файл /etc/inittab, введите в командной строке:

```
mkitab Идентификатор:Режим-работы:Действие:Команда
```

Например, для того чтобы добавить запись для tty2, введите в командной строке:

```
mkitab tty002:2:respawn:/usr/sbin/getty /dev/tty2
```

В предыдущем примере:

Элемент	Описание
tty002	Уникальное имя записи.
2	Уровень выполнения, на котором будет запускаться процесс.
respawn	Действие, которое команда **init** должна выполнять для данного процесса.
/usr/sbin/getty /dev/tty2	Команда оболочки, запускающая процесс.

Изменение записей - команда chitab

Для того чтобы изменить запись в файле **/etc/inittab**, введите в командной строке:

```
chitab Идентификатор:Режим-работы:Действие:Команда
```

Например, для того чтобы изменить запись из предыдущего примера так, чтобы она действовала для уровней 2 и 3, введите следующую команду:

```
chitab tty002:23:respawn:/usr/sbin/getty /dev/tty2
```

В предыдущем примере:

Элемент	Описание
tty002	Уникальное имя записи.
23	Уровни выполнения, на которых будет запускаться процесс.
respawn	Действие, которое команда **init** должна выполнять для данного процесса.
/usr/sbin/getty /dev/tty2	Команда оболочки, запускающая процесс.

Просмотр записей - команда lsitab

Для того чтобы получить список записей в файле /etc/inittab, введите в командной строке:

```
lsitab -a
```

Для того чтобы просмотреть отдельную запись файла /etc/inittab, введите:

```
lsitab Идентификатор
```

Например, для просмотра записи устройства tty2 нужно ввести команду `lsitab tty2`.

Удаление записей - команда rmitab

Для того чтобы удалить запись из файла /etc/inittab, введите в командной строке:

```
rmitab Идентификатор
```

Например, для удаления записи о tty2 нужно ввести команду `rmitab tty2`.

Понятия, связанные с данным:

“Режим работы системы” на стр. 14
Режим работы задает состояние системы и определяет набор активных процессов.

Повторная активация неактивной системы:

Неполадка аппаратного или программного обеспечения, а также их комбинации, могут привести к прерыванию работы системы.

Ниже описана процедура, позволяющая устранить неполадку и повторно запустить систему. Если по завершении процедуры система не заработает, перейдите к информации по определению неполадки в документации по аппаратному обеспечению.

Для возобновления работы системы выполните следующие действия:

Проверка аппаратного обеспечения:

Для проверки аппаратного обеспечения можно использовать несколько процедур.

Проверьте аппаратное обеспечение:

Проверка питания:

Если горит индикатор питания, перейдите к разделу **Проверка индикатора панели управления**, расположенному ниже.

Если индикатор питания не горит, убедитесь, что переключатель включен и вилка шнура питания вставлена в розетку.

Проверка индикатора панели управления:

Если система оснащена индикатором панели управления, проверьте, нет ли на нем каких-либо сообщений.

Если индикатор панели управления системы не показывает сообщений, перейдите к разделу **Активация дисплея или терминала**, расположенному ниже.

Если индикатор панели управления показывает код сообщения, обратитесь к руководству по обслуживанию системы, чтобы узнать значение кода.

Активация дисплея или терминала:

Проверьте следующие компоненты дисплея или терминала:

- Убедитесь, что кабель дисплея надежно подключен к дисплею и системному блоку.
- Убедитесь, что кабель клавиатуры надежно подключен.
- Убедитесь, что кабель мышки подключен надежно.
- Убедитесь, что питание дисплея включено и горит индикатор питания дисплея.
- Настройте яркость дисплея.
- Проверьте правильность параметров связи терминала.

Если система заработала, значит при проверке аппаратного обеспечения неполадка была устранена.

Задачи, связанные с данной:

“Перезапуск системы” на стр. 12
Помимо проверки аппаратного обеспечения и проверки процессов можно также перезапустить систему для повторной активации неактивной системы.

“Проверка процессов”
Остановившийся или зависший процесс может привести к прекращению работы системы.

Проверка процессов:

Остановившийся или зависший процесс может привести к прекращению работы системы.

Проверьте процессы системы:

1. Перезапуск прокрутки строк
2. Применение комбинации клавиш Ctrl+D
3. Применение комбинации клавиш Ctrl+C
4. Вход в систему с удаленного терминала или хоста
5. Удаленное завершение работы устаревшего процесса

Перезапуск прокрутки линий

Для повторного запуска прокрутки строк, остановленного нажатием комбинации клавиш Ctrl-S, выполните следующие действия:

1. Активизируйте окно или оболочку с процессом, в котором возникла неполадка.
2. Нажмите комбинацию клавиш Ctrl-Q для перезапуска прокрутки строк. Комбинация клавиш Ctrl-S позволяет остановить прокрутку строк, а Ctrl-Q - перезапустить прокрутку.

Если после проверки прокрутки неполадка не была устранена, перейдите к следующему разделу **Применение комбинации клавиш Ctrl-D**.

Применение комбинаций клавиш Ctrl-D:

1. Активизируйте окно или оболочку с процессом, в котором возникла неполадка.
2. Нажмите Ctrl-D. Процессу будет отправлен сигнал конец файла (EOF). Нажатие клавиш Ctrl-D может закрыть окно или оболочку и привести к выходу из системы.

Если после нажатия комбинации клавиш Ctrl+D неполадка не была устранена, перейдите к следующему разделу **Применение комбинации клавиш Ctrl-C**.

Применение комбинации клавиш Ctrl-C.

Для завершения остановившегося процесса выполните следующие действия:

1. Активизируйте окно или оболочку с процессом, в котором возникла неполадка.
2. Нажмите Ctrl-C. Комбинация клавиш Ctrl-C позволяет завершить текущий поиск или работу фильтра.

Если после нажатия комбинации клавиш Ctrl+C неполадка не была устранена, перейдите к следующему разделу **Вход в систему из удаленного хоста или терминала:**.

Вход в систему с удаленного терминала или хоста

Выполнить удаленный вход в систему можно двумя способами:

- Если к системе подключено несколько терминалов, войдите в систему с терминала.
- Если система подключена к сети, войдите в систему с внешнего хоста, введя следующую команду **tn**:

  tn *имя-системы*

  После вызова команды **tn** система запросит имя пользователя и пароль.

Если вам удалось войти в систему с удаленного терминала или хоста, перейдите к следующему разделу **Удаленное завершение устаревшего процесса**.

Если войти в систему с удаленного терминала или хоста не удалось, необходимо перезапустить систему.

Кроме того, для определения причины сбоя вы можете создать дамп системы.

Удаленное завершение работы устаревшего процесса

Для завершения зависшего процесса с удаленного терминала выполните следующие действия:

1. Просмотрите список процессов, введя следующую команду **ps**:

   ```
   ps -ef
   ```

   Флаги **-e** и **-f** обозначают все активные и неактивные процессы.
2. Определите ИД зависшего процесса.

   Идентификацию процессов можно упростить, введя команду **grep** со строкой поиска. Например, для определения ИД процесса **xlock** можно ввести следующую команду:

   ```
   ps -ef | grep xlock
   ```

   Команда **grep** позволяет осуществлять поиск в выходных данных команды **ps** для определения ИД конкретного процесса.
3. Для завершения процесса введите следующую команду **kill**:

   Примечание: Для применения команды **kill** к процессам, запущенным другими пользователями, необходимы права доступа root.

   kill -9 *ИД-процесса*

   Если вам не удается определить процесс, в котором возникла неполадка, необходимо проверить процесс, который был запущен в системе последним. Если вы считаете, что этот процесс послужил причиной возникновения неполадки, завершите его работу.

Если в неактивной системе после проверки процессов неполадка не была устранена, необходимо перезапустить систему.

Понятия, связанные с данным:

“Проверка аппаратного обеспечения” на стр. 9
Для проверки аппаратного обеспечения можно использовать несколько процедур.

Задачи, связанные с данной:

“Перезапуск системы”
Помимо проверки аппаратного обеспечения и проверки процессов можно также перезапустить систему для повторной активации неактивной системы.

Информация, связанная с данной:

Средства системного дампа

Перезапуск системы:

Помимо проверки аппаратного обеспечения и проверки процессов можно также перезапустить систему для повторной активации неактивной системы.

Если после выполнения процедур “Проверка аппаратного обеспечения” на стр. 9 и “Проверка процессов” на стр. 10 неполадка, препятствующая работе системы, не была устранена, необходимо перезапустить систему.

Примечание: Перед перезапуском создайте дамп системы.

1. Проверьте состояние загрузочного устройства.

   Загрузка системы выполняется со съемного носителя, внешнего устройства, устройства SCSI, устройства IDE или из локальной сети (LAN). Выберите нужный способ загрузки и выполните для проверки загрузочного устройства следующие действия:

   - В случае съемного носителя, такого как магнитная лента, убедитесь, что носитель правильно установлен.
   - В случае устройства IDE, убедитесь, что параметры ИД устройства IDE уникальны для каждого адаптера. Если к устройству подключено только одно устройство, ИД устройства IDE должно соответствовать основному устройству.
   - В случае внешнего устройства, такого как накопитель на магнитной ленте, проверьте выполнение следующих условий:
     – Питание устройства включено.
     – Кабели устройства правильно подключены к устройству и системному блоку.
     – Горит индикатор готовности (если есть).
   - В случае внешнего устройства SCSI проверьте уникальность заданного адреса SCSI.
   - В случае локальной сети убедитесь, что сеть настроена и работает.

   Если загрузочное устройство работает нормально, перейдите к следующему действию.
2. Для загрузки операционной системы выполните следующие действия:
   a. Отключите питание системы.
   b. Подождите одну минуту.
   c. Включите питание системы.
   d. Подождите, пока система загрузится.

Если операционная система не загрузилась, загрузите систему в режиме обслуживания или диагностики аппаратного обеспечения.

Если вам не удалось перезапустить систему, с помощью SRN сообщите о неполадке в сервисное представительство.

Понятия, связанные с данным:

“Проверка аппаратного обеспечения” на стр. 9
Для проверки аппаратного обеспечения можно использовать несколько процедур.

Задачи, связанные с данной:

“Проверка процессов” на стр. 10
Остановившийся или зависший процесс может привести к прекращению работы системы.

Информация, связанная с данной:

Средства системного дампа

Создание загрузочных образов

Для установки базовой операционной системы или для обращения к системе, которая не может быть загружена с системного жесткого диска, необходим загрузочный образ. В данном разделе описано создание загрузочных образов. Формат загрузочного образа зависит от типа устройства.

Во время начальной установки системы команда **bosboot** создает загрузочный образ из файловой системы на электронном диске и ядра операционной системы. Загрузочный образ располагается на отдельном носителе, например, жестком диске. Во время повторной загрузки системы загрузочный образ копируется с носителя в оперативную память. Дополнительная информация о команде **bosboot** приведена в разделе **bosboot**.

На дисках RAM хранятся процедуры настройки для следующих устройств:

- **Диск**
- **Магнитная лента**
- **CD-ROM**
- **Сетевое устройство Token-Ring, Ethernet или FDDI**
- Для применения команды **bosboot** у вас должны быть права root.
- В файловой системе `/tmp` должно быть не менее 20 МБ свободного пространства.
- Физический диск должен содержать загрузочный логический том. Чтобы определить, какой диск следует указать, введите в командной строке следующую команду:

  ```
  lsvg -l rootvg
  ```

  Команда **lsvg -l** показывает список логических томов корневой группы томов (rootvg). Найдите в этом списке имя загрузочного логического тома.

  Затем введите в командной строке следующую команду:

  ```
  lsvg -M rootvg
  ```

  Команда **lsvg -M** показывает список физических томов, содержащих логические тома.

Создание загрузочного образа в загрузочном логическом томе:

Если выполняется установка базовой операционной системы (первая установка или модернизация), то с помощью команды **bosboot** вы можете разместить загрузочный образ на загрузочном логическом томе. Загрузочный логический том - это физически непрерывная область диска, созданная с помощью Администратора логических томов (LVM) во время установки.

Список предварительных требований для этой процедуры приведен в разделе “Создание загрузочных образов”.

Команда **bosboot** выполняет следующие действия:

1. Проверяет, достаточно ли в файловой системе места для создания загрузочного образа.
2. Создает файловую систему RAM с помощью команды **mkfs** и файла-прототипа.
3. Вызывает команду **mkboot**, которая объединяет ядро и файловую систему RAM в загрузочный образ.
4. Записывает загрузочный образ на загрузочный логический том.

Для создания загрузочного образа на загрузочном логическом томе, находящемся на жестком диске, введите в командной строке:

```
bosboot -a
```

ИЛИ:

```
bosboot -ad /dev/ipldevice
```

Примечание: Если в процессе создания загрузочного образа команда **bosboot** завершилась неудачно, не следует перезагружать компьютер. Устраните неполадку и запустите команду **bosboot**.

Для активизации нового загрузочного образа необходимо перезагрузить систему.

Создание загрузочных образов для сетевых устройств:

Можно создать загрузочные образы для загрузки через Ethernet или Token-Ring.

Список предварительных требований для этой процедуры приведен в разделе “Создание загрузочных образов” на стр. 13.

Для создания загрузочного образа для загрузки Ethernet, введите в командной строке:

```
bosboot -ad /dev/ent
```

Для загрузки по сети Token-Ring:

```
bosboot -ad /dev/tok
```

Режим работы системы

Режим работы задает состояние системы и определяет набор активных процессов.

Например, если режим работы системы равен 3, то запускаются все процессы, которые должны быть активны на этом уровне. Во время загрузки системы режим работы считывается из записи initdefault файла `/etc/inittab`. Система работает на этом уровне до тех пор, пока не будет получен сигнал изменения уровня выполнения. Для изменения режима работы предназначена команда **init**. Записи файлов `/etc/inittab` содержат режим работы каждого процесса. Во время загрузки системы команда **init** просматривает файл `/etc/inittab` и запускает нужные процессы.

Ниже описаны существующие в данный момент режимы работы:

Элемент	Описание
0-9	Когда команда **init** включает один из уровней в диапазоне 0-9, то сначала убиваются все процессы, связанные со старым уровнем выполнения, а затем перезапускаются процессы, определенные для нового уровня.
0-1	Зарезервированы для операционной системы, в настоящее время не используются.
2	Уровень выполнения по умолчанию.
3-9	Могут быть определены пользователем.
a, b, c	При переключении в режимы **a**, **b** и **c** команда **init** запускает процессы, связанные с новым режимом, но не убивает текущие процессы.
Q, q	Служебные значения, указывающие, что команда **init** должна проанализировать файл `/etc/inittab`.

Задачи, связанные с данной:

“Изменение файла /etc/inittab” на стр. 8
Для изменения файла доступны следующие четыре команды `etc/inittab`.

Определение режима работы системы:

Перед настройкой операционной системы или изменением режима работы системы ознакомьтесь с информацией о различных уровнях выполнения.

Ниже описана процедура, с помощью которой можно узнать текущий режим работы и просмотреть хронологию предыдущих режимов. Определить режим работы системы можно с помощью команды **init**.

Определение текущего режима работы

Введите в командной строке: `cat /etc/.init.state`. Появится одна цифра, означающая текущий режим. Дополнительные сведения о режимах работы приведены в описании команды **init** или `/etc/inittab`.

Просмотр хронологии предыдущих режимов работы:

Для просмотра хронологии предыдущих режимов работы служит команда **fwtmp**.

Примечание: Для выполнения этой команды в системе должен быть установлен файл `bosext2.acct.obj`.

1. Войдите в систему как пользователь root.
2. Введите в командной строке:

   ```
   /usr/lib/acct/fwtmp </var/adm/wtmp |grep run-level
   ```

   Появится приблизительно следующая информация:

   ```
   run-level 2  0 1 0062 0123 697081013 Sun Feb  2 19:36:53 CST 1992
   run-level 2  0 1 0062 0123 697092441 Sun Feb  2 22:47:21 CST 1992
   run-level 4  0 1 0062 0123 698180044 Sat Feb 15 12:54:04 CST 1992
   run-level 2  0 1 0062 0123 698959131 Sun Feb 16 10:52:11 CST 1992
   run-level 5  0 1 0062 0123 698967773 Mon Feb 24 15:42:53 CST 1992
   ```

Настройка режима работы в многопользовательских системах:

Можно изменить режим работы в многопользовательских системах.

1. Просмотрите файл `/etc/inittab` и убедитесь, что в режиме, который вы хотите установить, поддерживаются нужные вам процессы. Особенно важен процесс getty, отвечающий за доступ к системной консоли и другим устройствам входа в систему через терминалы. Убедитесь, что процесс getty включен на всех уровнях выполнения.
2. С помощью команды **wall** сообщите всем пользователям системы о предстоящем изменении режима и попросите пользователей завершить работу в системе. Дополнительная информация о команде **wall** приведена в разделе **wall**.
3. Введите команду **smit telinit** для запуска программы SMIT и перехода в меню **Установить режим работы системы**.
4. Укажите нужное значение в поле **Режим работы**.
5. Нажмите Enter для изменения уровня. Система покажет информацию о том, какие процессы будут убиты и запущены в результате изменения уровня, а затем выдаст следующее сообщение:

   ```
   INIT: Новый уровень: n
   ```

   где *n* - новый уровень выполнения.

Настройка режима работы в однопользовательских системах:

Можно изменить режим работы в однопользовательских системах.

1. Проверьте файл `/etc/inittab` и убедитесь, что в режиме, который вы хотите установить, поддерживаются нужные вам процессы. Особенно важен процесс getty, отвечающий за доступ к системной консоли и другим устройствам входа в систему через терминалы. Убедитесь, что процесс getty включен на всех уровнях выполнения. Дополнительная информация о файле `inittab` приведена в разделе `inittab`.
2. Введите команду `smit telinit` для перехода к меню **Установить рабочий режим системы**. Дополнительная информация о команде `telinit` приведена в разделе `telinit`.
3. Укажите новый режим работы в поле **Режим работы системы**.

4. Нажмите Enter для изменения уровня.

   Система покажет информацию о том, какие процессы будут убиты и запущены в результате изменения уровня, а затем выдаст следующее сообщение:

   `INIT: Новый уровень: n`

   где *n* - новый уровень выполнения.

Процесс загрузки

Возможны три типа загрузки системы и два ресурса, необходимые для загрузки операционной системы.

Во время загрузки система проверяет аппаратное обеспечение, загружает и запускает операционную систему и настраивает устройства. Для загрузки операционной системы необходимы следующие ресурсы:

- *Загрузочный образ*, который может быть загружен после включения или перезагрузки компьютера.
- Доступ к корневой файловой системе (/) и файловой системе /usr.

Возможны три типа загрузки системы:

Элемент	Описание
Загрузка с жесткого диска	Компьютер загружается для обычной работы.
Загрузка бездискового клиента по сети	Рабочая станция без диска или без данных загружается удаленно по сети. Компьютер загружается для обычной работы. Один или несколько файловых серверов передают этой рабочей станции файлы и программы для загрузки.
Загрузка в режиме обслуживания	Компьютер загружается с жесткого диска, магнитной ленты, компакт диска или из сети в режиме обслуживания. Системный администратор может выполнять задачи установки нового или обновленного программного обеспечения, а также запуск диагностических тестов.

Для загрузки с жесткого диска применяется загрузочный образ, находящийся на локальном диске и созданный при установке операционной системы. Во время загрузки система настраивает все устройства и инициализирует основные программы, необходимые для работы (например, Администратор логических томов). Этот процесс заканчивается монтированием и подготовкой к работе файловых систем.

Процесс загрузки бездисковых сетевых клиентов в основном совпадает с описанным выше. Кроме того, бездисковым клиентам требуется загрузочный образ и доступ к дереву файлов и каталогов операционной системы. У бездисковых сетевых клиентов нет собственной файловой системы, поэтому они получают всю информацию из удаленных источников.

Понятия, связанные с данным:

“Выполнение начальной загрузки системы”
Большая часть пользователей загружают систему с жесткого диска. На жестком диске находится вся информация, необходимая для загрузки.

“Процесс загрузки в режиме обслуживания” на стр. 18
Иногда требуется загружать систему в режиме обслуживания для выполнения специальных задач, таких как установка и обновление программного обеспечения, запуск диагностических тестов и обслуживание системы. В этом случае система загружается со сменного носителя (компакт-диска, диска DVD магнитной ленты и т.д.), по сети или с жесткого диска.

“файловая система RAM” на стр. 19
Файловая система RAM - это часть загрузочного образа, целиком находящаяся в оперативной памяти и содержащая все программы, необходимые для загрузки. Набор файлов этой системы зависит от типа загрузки.

Выполнение начальной загрузки системы:

Большая часть пользователей загружают систему с жесткого диска. На жестком диске находится вся информация, необходимая для загрузки.

С момента включения питания системы (холодная загрузка) или перезапуска системы командой **reboot** либо **shutdown** (горячая загрузка) и до начала работы системы должны быть выполнены определенные действия. Эти действия можно разделить на следующие этапы:

Понятия, связанные с данным:

“Процесс загрузки” на стр. 16
Возможны три типа загрузки системы и два ресурса, необходимые для загрузки операционной системы.

Этап встроенного программного обеспечения:

Встроенное программное обеспечение подготавливает систему к загрузке и запуску операционной системы.

Этап его инициализации состоит их следующих шагов:

1. Встроенное программное обеспечение выполняет общую проверку системных ресурсов, необходимых для запуска операционной системы.
2. Встроенное программное обеспечение проверяет пользовательский список загрузки, в который должны быть внесены доступные устройства загрузки. Этот список загрузки можно изменить командой **bootlist**. Если пользовательский список загрузки в NVRAM поврежден, либо ни одно из указанных в нем устройств не подходит для загрузки, проверяется список загрузки по умолчанию. В любом случае для загрузки системы применяется первое допустимое устройство из списка загрузки. Если пользовательский список загрузки в NVRAM не поврежден, то проверяются по порядку устройства из этого списка. Если пользовательский список загрузки отсутствует, проверяются все адаптеры и устройства. В любом случае устройства проверяются циклически, пока с одного из них не удастся загрузить систему.

   Примечание: При обычной загрузке система применяет список загрузки по умолчанию, хранящийся в NVRAM). При служебной загрузке система также применяет список, хранящийся в NVRAM, и для ознакомления с информацией по доступу к списку служебной загрузки необходимо обратиться к инструкциям для специального аппаратного обеспечения.
3. После того как найдено допустимое устройство загрузки, проверяется его первая запись или номер сектора программы (PSN). Если загрузочная запись допустима, она загружается в память и добавляется к управляющему блоку загрузки начальной программы (IPL). Загрузочная запись содержит информацию о начальном положении и длине загрузочного образа на устройстве загрузки, а также о смещении точки входа после загрузки образа в память.
4. Загрузочный образ последовательно считывается с загрузочного устройства в память, начиная с положения, указанного в NVRAM. Загрузочный образ, хранящийся на диске, содержит ядро, файловую систему RAM и информацию о настройке основных устройств.
5. Управление передается ядру, которое начинает инициализацию системы.
6. Ядро загружает программу **init**, которая выполняет первый этап сценария `rc.boot`.

После завершения этапа инициализации ядра начинается настройка базовых устройств.

Этап настройки основного устройства:

Процесс **init** запускает сценарий `rc.boot`. На первом этапе сценарий `rc.boot` задает конфигурацию основных устройств.

Первый этап сценария `rc.boot` содержит следующие шаги:

1. Сценарий загрузки вызывает программу **restbase** для создания в файловой системе RAM базы данных настройки Администратора объектных данных (ODM) из сжатых данных настройки.
2. Сценарий загрузки запускает Администратор настройки, который задает конфигурацию основных устройств согласно правилам первого этапа настройки ODM.
3. Администратор настройки запускает процедуры настройки **sys**, **bus**, **disk**, SCSI, Администратора логических томов (LVM) и корневой группы томов (rootvg).

4. Процедуры настройки загружают драйверы устройств, создают специальные файлы и обновляют информацию в базе данных настройки ODM.

Загрузка системы:

Эта процедура выполняет загрузку системы.

1. Процесс **init** запускает второй этап сценария `rc.boot`. Этот этап сценария `rc.boot` содержит следующие шаги:
   a. Вызов программы **ipl_varyon** для активизации группы томов rootvg.
   b. Монтирование файловых систем, хранящихся на жестком диске, в обычных точках монтирования.
   c. Вызов программы **swapon** для запуска подкачки.
   d. Копирование базы данных настройки ODM из RAM на жесткий диск.
   e. Завершение работы сценария `rc.boot`.
2. После выполнения второго этапа сценария `rc.boot` система переключается с файловой системы RAM на файловые системы, расположенные на жестком диске.
3. Затем процесс **init** вызывает команды, перечисленные в файле `/etc/inittab`. В одной из записей файла `/etc/inittab` задана команда выполнения третьего этапа сценария `rc.boot`, включающего следующие шаги:
   a. Монтирование файловой системы `/tmp`, хранящейся на жестком диске.
   b. Запуск этапа 2 Администратора настройки для настройки всех остальных устройств.
   c. Вызов команды **savebase** для сохранения данных настройки в загрузочном логическом томе.
   d. Завершение работы сценария `rc.boot`.

После выполнения всех перечисленных выше действий система готова к работе.

Процесс загрузки в режиме обслуживания:

Иногда требуется загружать систему в режиме обслуживания для выполнения специальных задач, таких как установка и обновление программного обеспечения, запуск диагностических тестов и обслуживание системы. В этом случае система загружается со сменного носителя (компакт-диска, диска DVD магнитной ленты и т.д.), по сети или с жесткого диска.

Последовательность действий при загрузке в режиме обслуживания почти не отличается от обычной загрузки.

1. Встроенное программное обеспечение выполняет общую проверку системных ресурсов, необходимых для запуска операционной системы.
2. Встроенное программное обеспечение проверяет пользовательский список загрузки. Этот список можно изменить с помощью команды **bootlist**. Если пользовательский список загрузки в NVRAM поврежден, либо ни одно из указанных в нем устройств не подходит для загрузки, проверяется список загрузки по умолчанию. В любом случае для загрузки системы применяется первое допустимое устройство из списка загрузки.

   Примечание: При обычной загрузке система применяет список загрузки по умолчанию и пользовательский список загрузки, которые хранятся в NVRAM. Список загрузки по умолчанию и пользовательский список загрузки также применяются для запуска системы в режиме обслуживания.
3. После того как найдено допустимое устройство загрузки, проверяется его первая запись или номер сектора программы (PSN). Если загрузочная запись допустима, она загружается в память и добавляется к управляющему блоку загрузки начальной программы (IPL). Загрузочная запись содержит информацию о начальном положении и длине загрузочного образа на загрузочном устройстве, а также о смещении точки входа, к которой следует перейти после загрузки образа в память.
4. Загрузочный образ последовательно считывается с загрузочного устройства в память, начиная с положения, указанного в NVRAM.

5. Управление передается ядру, которое запускает программы, хранящиеся в файловой системе RAM.
6. Команда **cfgmgr** получает информацию об устройствах системы из базы данных ODM и динамически настраивает найденные устройства, включая все диски, на которых расположена корневая файловая система.
7. Если система загружается с компакт-диска, диска DVD, магнитной ленты или по сети, группа томов rootvg не активизируется, поскольку rootvg может в это время еще не существовать (например, при установке операционной системы на новый компьютер). Вместо этого выполняется настройка сети. При загрузке в режиме обслуживания функция подкачки отключена.

После завершения описанного выше процесса система готова к выполнению процедур установки, обслуживания и диагностики.

Примечание: Если система запускается с жесткого диска, то активизируется rootvg, корневая файловая система и файловая система `/usr` монтируются в файловой системе RAM, и пользователю выдается меню, позволяющее перейти к различным диагностическим функциям или в однопользовательский режим. Если выбран однопользовательский режим, то можно продолжить загрузку, установив уровень выполнения программы **init** равным S. После этого система готова к обслуживанию, обновлению программного обеспечения или запуску команды **bosboot**.

Понятия, связанные с данным:

“Процесс загрузки” на стр. 16
Возможны три типа загрузки системы и два ресурса, необходимые для загрузки операционной системы.

файловая система RAM:

Файловая система RAM - это часть загрузочного образа, целиком находящаяся в оперативной памяти и содержащая все программы, необходимые для загрузки. Набор файлов этой системы зависит от типа загрузки.

Файловая система RAM при загрузке в режиме обслуживания может не содержать функции для работы с логическими томами, так как активизация группы томов rootvg может не потребоваться. Однако при загрузке с жесткого диска желательно, чтобы группа томов rootvg и пространство подкачки были активизированы как можно раньше. Несмотря на указанные различия, в обоих вариантах загрузки общая структура файловой системы RAM будет приблизительно одинаковой.

Команда **init**, расположенная в файловой системе RAM, представляет собой интерпретатор команд загрузки, предназначенный для применения в процессе загрузки. Эта программа интерпретатора команд загрузки управляет процессом загрузки путем вызова сценария `rc.boot`. Сценарий `rc.boot` определяет устройство, с помощью которого была запущена система. Загрузочное устройство определяет список устройств, которые должны быть настроены с помощью файловой системы RAM. Если компьютер запускается из сети, должны быть настроены сетевые устройства, обеспечивающие удаленное монтирование файловых систем клиента. При загрузке с магнитной ленты, компакт-диска или диска DVD необходимо настроить консоль для отображения меню установки базовой операционной системы (BOS). После того как загрузочное устройство будет найдено, сценарий `rc.boot` вызывает соответствующие функции настройки из файловой системы RAM. Сценарий `rc.boot` вызывается программой интерпретатора команд загрузки дважды, во время каждого из двух этапов настройки. При загрузке по сети или с диска сценарий `rc.boot` вызывается в третий раз после запуска обычной команды **init**. Файл `inittab` содержит команду вызова сценария `rc.boot`, который выполняет последний этап настройки компьютера.

Для каждого загрузочного устройства существует своя файловая система RAM, так как в разных режимах загрузки требуется настраивать различные устройства. С каждым типом загрузочного устройства связан файл прототипа. Прототип - это шаблон для выбора файлов, составляющих файловую систему RAM. Для создания файловой системы RAM по одному из файлов прототипов команда **bosboot** вызывает команду **mkfs**. Дополнительная информация приведена в описании команды **bosboot**.

Понятия, связанные с данным:

“Процесс загрузки” на стр. 16
Возможны три типа загрузки системы и два ресурса, необходимые для загрузки операционной системы.

Устранение неполадок при запуске системы

Следующие приемы устранения неполадок предназначены для разрешения некоторых наиболее распространенных неполадок, которые могут возникнуть при запуске системе. Если эта информация не помогла устранить неполадку, обратитесь в службу поддержки.

Незагружающиеся системы:

Если система не загружается в жесткого диска вы можете получить доступ к ней для обнаружения и устранения неполадки.

Если систему не удается загрузить с жесткого диска, обратитесь к разделу Устранение неполадок при установке книги *Установка и миграция*.

Выполнив инструкции из указанного раздела, вы увидите командную строку системы и сможете попытаться восстановить данные или устранить причину, по которой систему не удается загрузить.

Примечание:

- Эта процедура рассчитана только на опытных администраторов, знакомых с принципами загрузки системы и восстановления данных. Остальным пользователям мы рекомендуем обратиться в сервисное представительство.
- Эта процедура предназначена только для восстановления пользовательских данных с жестких дисков незагружающейся системы. Если не загружается новая система или система, в которой была выполнена установка с заменой всех данных, то в ней нет пользовательской информации, и эту процедуру выполнять не нужно. Если после установки новая система не загружается с жесткого диска, обратитесь в сервисное представительство.

Ссылки, связанные с данной:

“Диагностика неполадок загрузки”
Загрузка системы может быть прервана по нескольким причинам.

Диагностика неполадок загрузки:

Загрузка системы может быть прервана по нескольким причинам.

К этим причинам относятся:

- Аппаратные неполадки
- Неисправные загрузочные магнитные ленты или диски CD-ROM
- Неправильно настроенные сетевые серверы загрузки
- Поврежденные файловые системы
- Ошибки в сценариях, например, в `/sbin/rc.boot`

Если в ходе загрузки будет выдан информационный код 2702 и сообщение "Недостаточно выделенной памяти", следует увеличить объем выделенной памяти раздела с помощью HMC.

Понятия, связанные с данным:

“Незагружающиеся системы”
Если система не загружается в жесткого диска вы можете получить доступ к ней для обнаружения и устранения неполадки.

Резервное копирование системы

После завершения настройки системы рекомендуется создать резервную копию всех файловых систем, каталогов и файлов. При резервном копировании файловых систем можно восстановить файлы или файловые системы в случае сбоя жесткого диска. Есть несколько методов резервного копирования данных.

Файлы и каталоги хранят в себе значительный объем вложенных средств, сил и времени. В то же время все файлы на компьютере легко могут быть изменены или удалены, случайно или преднамеренно.

Внимание: Когда происходит аварийный сбой на жестком диске, все данные, хранившиеся на нем, теряются. Единственный источник восстановления данных при этом - резервная копия.

При аккуратном и методичном подходе к резервному копированию файловых систем вы всегда без труда сможете восстановить последние версии файлов.

Существует несколько различных способов резервного копирования данных. Чаще всего применяется *резервное копирование по именам файлов*, *создание архива файлов* или *обычное резервное копирование*. Это копирование, при котором создается копия исходных данных для передачи в другую систему или на случай их случайного изменения или уничтожения. Для применения такого способа резервного копирования укажите флаг **i**, позволяющий создавать резервные копии отдельных файлов и каталогов. С его помощью конкретные пользователи обычно создают резервные копии своих учетных файлов.

Другой распространенный способ - резервное копирование файловой системы, также называемый *i-копированием*, *архивом файловых систем* или *резервным копированием архива*. Этот способ выбирается в случае, если флаг **i** *не* указан. Он используется для будущего использования, в архивных целях или для восстановления исходных данных в случае их повреждения. Его применяют для создания резервной копии всей файловой системы; обычно им пользуются системные администраторы для копирования большого количества файлов, например, всех учетных пользовательских файлов в каталоге `/home`. Данный способ позволяет без труда выполнять дополняющее резервное копирование. При дополняющем резервном копировании создаются резервные копии всех файлов, которые были изменены с момента прошлого копирования.

С помощью команд **compress** и **pack** можно сжать файлы для хранения, а с помощью команд **uncompress** и **unpack** можно распаковать файлы после восстановления. Процесс упаковки и распаковки файлов занимает некоторое время, но позволяет значительно сэкономить место на носителе резервной копии. Дополнительная информация об этих командах приведена в разделах **compress**, **pack**, **uncompress** и **unpack**.

Существует несколько команд создания резервных копий и архивов. Поэтому созданные копии должны иметь метку с указанием применявшейся команды и способа копирования: по имени или файловой системы.

Элемент	Описание
backup	Создает резервные копии файлов по имени или в файловой системе. Дополнительная информация приведена в описании команды **backup**.
mksysb	Создает устанавливаемый образ rootvg. Дополнительная информация приведена в описании команды **mksysb**.
cpio	Копирует файлы в архив и из архива. Дополнительная информация приведена в описании команды **cpio**.
дд	Преобразует и копирует файлы. Обычно эта команда применяется для преобразования и копирования данных из других операционных систем, например, при копировании данных с мейнфреймов. Команда **dd** не позволяет поместить несколько файлов в один архив. Она применяется только для перемещения и обработки данных. Дополнительная информация приведена в описании команды **dd**.
tar	Позволяет создать и работать с архивами `tar`. Дополнительная информация приведена в описании команды **tar**.
rdump	Создает на удаленном компьютере резервную копию файлов из файловой системы. Дополнительная информация приведена в описании команды **rdump**.
pax	(Совместима с POSIX) Читает и записывает архивы **tar** и **cpio**. Дополнительная информация приведена в описании команды **pax**.

Понятия, связанные с данным:

“Резервное копирование - информация для системных администраторов BSD 4.3” на стр. 329
Системные администраторы BSD 4.3 могут выполнять резервное копирование данных.

Задачи, связанные с данной:

“Резервное копирование пользовательских файлов или файловых систем” на стр. 26
Существует два способа резервного копирования файлов и файловых систем: команды быстрого доступа SMIT **smit backfile** и **smit backfilesys**, а также команда **backup**.

Принципы резервного копирования

Перед началом резервного копирования данных необходимо ознакомиться с основными сведениями об используемых типах данных, стратегиях и носителях.

Стратегии резервного копирования:

Единственной стратегии резервного копирования, которая подходила бы всем пользователям, не существует. Стратегия, хорошо работающая в системе с одним пользователем, может оказаться непригодной для системы, обслуживающей сто пользователей. Аналогично, стратегия, разработанная для системы, в которой каждый день меняется множество файлов, окажется неэффективной в системе, в которой данные меняются редко.

Какой бы ни была стратегия резервного копирования для вашей системы, она должна выполняться. В противном случае восстановление после потери данных может потребовать очень больших усилий.

Ниже приведены некоторые общие рекомендации по выбору стратегии резервного копирования:

- **Убедитесь, что возможно восстановление данных даже после серьезных сбоев.**

  Сможет ли система продолжать работу в случае отказа одного из жестких дисков? Можно ли будет восстановить систему, если откажут все жесткие диски? Возможно ли восстановление системы, если дискеты или магнитная лента с резервной копией будут потеряны, украдены или повреждены? Насколько трудно будет вновь создать данные в случае потери? Обязательно разработайте стратегию, позволяющую восстановить систему после потери всех данных, даже если такое событие кажется вам маловероятным.
- **Периодически проверяйте резервные копии.**

  Носители резервных копий и соответствующие устройства могут оказаться ненадежными. Большая библиотека резервных лент и дискет будет совершенно бесполезна, если данные невозможно будет вновь перенести на жесткий диск. Для того чтобы убедиться, что резервные копии работоспособны, периодически просматривайте их оглавление (командой **restore -T** или, для архивов, **tar -t**). Если вы создаете архив на дискетах, и у вас есть несколько дисководов, проверьте этот архив на дисководе, отличном от того, на котором он был записан. Для повышения надежности резервного копирования можно создать дубликат резервной копии нулевого уровня на дополнительном наборе носителей. Если для создания резервной копии применяется потоковый накопитель на магнитной ленте, то элементарную проверку целостности ленты можно выполнить, например, с помощью команды **tapechk**. Дополнительная информация об этих командах приведена в разделах **restore -T**, **tar -t** и **tapechk**.
- **Не выбрасывайте старые резервные копии.**

  Разработайте расписание повторного использования носителей резервных копий, но не перезаписывайте все резервные копии. Иногда проходят месяцы, прежде чем какой-либо пользователь заметит, что один из его файлов поврежден или отсутствует. Именно в такой ситуации вам и пригодятся заботливо сохраненные старые резервные копии. Например, мы можете применять следующие три цикла для магнитных лент или дискет с резервной копией:
  - Создавая резервные копии каждый день, на следующей неделе используйте те же носители повторно, сохраняя только резервную копию, созданную в пятницу.
  - Каждый месяц повторно используйте все носители с созданными в пятницу резервными копиями, сохраняя только копию последней пятницы месяца. При этом резервные копии за последние четыре пятницы всегда будут доступны.
  - Каждый квартал повторно используйте носители с резервными копиями, созданными в последние пятницы каждого месяца, кроме последнего в квартале. Дискеты последних пятниц последнего месяца каждого квартала можно хранить неограниченно долго.
- **Перед выполнением резервного копирования проверяйте файловые системы.**

Резервная копия поврежденной файловой системы бесполезна. Перед созданием резервных копий рекомендуется проверять целостность файловых систем с помощью команды **fsck**. Дополнительная информация приведена в описании команды **fsck**.

- **Убедитесь в том, что копируемые файлы в момент копирования не меняются.**

  Не работайте в системе во время создания резервных копий. Изменение файлов во время резервного копирования приведет к тому, что созданная копия будет неточной.

- **Создавайте резервную копию системы перед внесением в нее значительных изменений.**

  Рекомендуется всегда создавать резервную копию системы перед выполнением аппаратного тестирования или восстановления, перед установкой новых устройств, программ и т.п.

- **Прочие факторы.**

  При разработке и реализации стратегии резервного копирования учтите следующие факторы:

  – Как часто меняются данные? Данные операционной системы изменяются сравнительно редко, поэтому их не нужно часто сохранять. С другой стороны, пользовательские данные обычно меняются часто и требуют соответствующей частоты обновления резервной копии.

  – Сколько пользователей работают в системе? Количество пользователей влияет на емкость носителя и частоту резервного копирования.

  – Насколько трудно будет воссоздать данные? Важно помнить, что некоторые данные нельзя восстановить в отсутствие резервной копии.

Создание стратегии резервного копирования необходимо для сохранения данных. Разработайте такую стратегию, исходя из особенностей своей системы. Информацию пользователей следует сохранять регулярно, через небольшие промежутки времени. В противном случае восстановление после потери данных может потребовать очень больших усилий.

Примечание: При создании резервных копий именованных конвейеров (специальных файлов FIFO) конвейеры могут быть как закрыты, так и открыты. Однако, при попытке восстановления открытого конвейера произойдет ошибка. Для восстановления специального файла FIFO необходим только его i-узел, поскольку он содержит всю необходимую информацию. Содержимое конвейера с именем не восстанавливается. Таким образом, перед резервным копированием размер файла должен быть равен нулю (все FIFO закрыты).

Внимание: Процедуры резервного копирования и восстановления требуют, чтобы восстановление выполнялось в системе с тем же типом платформы, что и сохранение. Особенно важно, чтобы платы CPU и планара ввода-вывода были одного типа.

Носитель резервной копии:

Существует несколько типов носителей резервной копии. Конкретные рекомендации зависят от установленного в вашей системе аппаратного и программного обеспечения.

Для создания резервной копии применяются носители нескольких типов. Какие конкретно носители можно использовать в вашей системе, зависит от ее аппаратного и программного обеспечения. В число таких носителей входят магнитные ленты, (8-мм пленка и 9-дорожечная пленка), дискеты (5,25-дюймовая и 3,5-дюймовая), удаленные архивы и съемные жесткие диски. Если в команде **backup -f** не задано другое устройство, то команда **backup** автоматически записывает данные на устройство `/dev/rfd0`, то есть дисковод.

Внимание: При выполнении команды **backup** удаляются все данные, ранее записанные на резервном носителе.

Дискеты

Дискеты - это стандартный носитель резервных копий. Если в команде **backup -f** не задано другое устройство, то команда **backup** автоматически записывает данные на устройство `/dev/rfd0`, то есть дисковод. Для создания резервной копии на накопителе на магнитной ленте по умолчанию укажите устройство `/dev/rmt0` и нажмите Enter.

Будьте осторожны при обращении с дискетами. Поскольку каждая небольшая область дискеты занята некоторым объемом информации, небольшие царапины, пыль или пепел от табака могут повредить данные. Запомните следующие правила:

- Не касайтесь поверхности дискеты.
- Храните дискеты в отдалении от магнитов и источников магнитных полей, таких как телефоны, динамики, микрофоны и калькуляторы.
- Оберегайте дискеты от жары и холода. Рекомендуемый диапазон температур: от 10 до 60 градусов Цельсия.
- Правильное обращение с дискетами предотвратит потерю информации.
- Регулярно создавайте резервные копии ваших дискет.

Внимание: Для надежного хранения данных типы дисководов и дискет должны соответствовать друг другу. Применение дискеты неправильного типа в 3.5-дюймовом дисководе может привести к разрушению данных на ней.

Дисководы поддерживают 3.5-дюймовые дискеты следующих типов:

- Емкостью 1 МБ (содержит до 720 КБ данных)
- Емкостью 2 МБ (содержит до 1,44 МБ данных)

Накопители на магнитной ленте

Учитывая большую емкость и высокую износостойкость магнитных лент, они часто выбираются для сохранения больших файлов или большого количества файлов, например, для архивного копирования файловых систем. Кроме того, магнитные ленты применяются для передачи больших объемов информации из одной системы в другую. Магнитные ленты редко применяются для хранения часто используемых данных, поскольку носители других типов имеют меньшее время доступа.

Файлы на магнитных лентах создаются командами **backup**, **cpio** и **tar**, открывающими накопитель на магнитной ленте, записывающими данные и закрывающими его.

Стратегия резервного копирования:

Существует два способа резервного копирования больших объемов данных.

- Полное копирование системы
- Дополняющее копирование

Для того чтобы выбрать способ, наиболее подходящий для конкретной системы, необходимо ознакомиться со структурой файловых систем и принципами размещения данных. После этого можно разработать стратегию резервного копирования этих данных.

Задачи, связанные с данной:

“Реализация запланированных резервных копий” на стр. 44
Ниже описана процедура создания и применения сценария еженедельного полного резервного копирования и ежедневного дополняющего копирования.

Системные данные и пользовательские данные:

В качестве данных могут рассматриваться либо программы, либо текст; в данном разделе мы выделяем два их класса:

- Системные данные, входящие в операционную систему и ее дополнительные компоненты. Эти данные всегда должны храниться в системных файловых системах, а именно: / (корневая файловая система), /usr, /tmp, /var и т.д.
- Пользовательские данные. Обычно это локальные данные, необходимые конкретным пользователям для выполнения своих задач. Эти данные должны храниться в файловой системе /home или в других файловых системах, специально созданных для пользовательских данных.

Пользовательские программы и данные не должны размещаться в файловых системах, предназначенных для хранения системных данных. Например, администратор системы может создать новую файловую систему и смонтировать ее в каталоге /local. Исключением является файловая система /tmp, используемая для временного хранения системных и пользовательских данных.

Резервные копии:

Как правило, резервные копии пользовательских и системных данных сохраняют на случай их непреднамеренного удаления или отказа жесткого диска. Работать с резервными копиями гораздо проще, когда системные и пользовательские данные хранятся отдельно.

Ниже перечислены преимущества такого размещения данных:

- Пользовательские данные изменяются намного чаще, чем системные. Объем резервной копии только пользовательских данных значительно меньше. Необходимый объем носителя резервной копии и частота резервного копирования зависят от количества пользователей.
- Восстановить пользовательские данные можно намного быстрее и проще, если они хранятся отдельно. Восстановление операционной системы вместе с пользовательскими данными потребует лишних усилий и времени. Это связано с тем, что процедура восстановления системных данных требует загрузки системы со съемных носителей (магнитной ленты или компакт-дисков) и установки резервной копии системы.

Перед резервным копированием системных данных размонтируйте все пользовательские файловые системы, включая /home, с помощью команды **umount**. Файловые системы нельзя размонтировать, когда они используются. Поэтому резервное копирование нужно запланировать на то время, когда система не используется. Если файловые системы с пользовательскими данными не будут размонтированы, то они будут сохранены вместе с данными операционной системы. Убедитесь в том, что в настоящий момент смонтированы только файловые системы, связанные с операционной системой, с помощью команды **mount**.

Смонтированными должны остаться следующие файловые системы: /, /usr, /var и /tmp. Результат выполнения команды **mount** должен выглядеть примерно следующим образом:

```
узел  смонтир. точка монт.  vfs   дата           опции

      /dev/hd4     /        jfs   11 июня 10:36  rw,log=/dev/hd8

      /dev/hd2     /usr     jfs   11 июня 10:36  rw,log=/dev/hd8

      /dev/hd9var  /var     jfs   11 июня 10:36  rw,log=/dev/hd8

      /dev/hd      /tmp     jfs   11 июня 10:36  rw,log=/dev/hd8
```

После того как вы убедитесь, что все пользовательские файловые системы размонтированы, можно приступить к резервному копированию данных операционной системы.

После создания резервной копии операционной системы смонтируйте пользовательские файловые системы с помощью команды **smit mount**. Теперь вы можете создать резервные копии файлов, файловых систем или иных групп томов, по своему усмотрению.

Понятия, связанные с данным:

“Резервное копирование системного образа и пользовательских групп томов” на стр. 40
Группа томов rootvg представляет собой жесткий диск или группу дисков, содержащих файлы загрузки системы, BOS, данные о конфигурации, а также дополнительные программные продукты. *Пользовательская*

группа томов (не *rootvg*) обычно применяется для хранения файлов данных и программного обеспечения.

Дублирование системы (клонирование):

Дублирование сохраняет не только пользовательские и системные данные, но и данные конфигурации. Например, вам может потребоваться создать точную копию системы или группы томов. Эту процедуру также иногда называют клонированием.

Созданный образ можно затем установить на другой компьютер и работать с ним в точности так же, как и с исходной системой. Команда **mksysb** позволяет создать дубликат группы томов rootvg, содержащей операционную систему. Дубликаты других групп томов можно создать с помощью команды **savevg**.

Обзор команд резервного копирования файлов и работы с носителями:

Имеются определенные команды для резервного копирования файлов и хранения данных.

Элемент	Описание
backup	Создает резервные копии и файловые системы
compress	Сжимает и разворачивает данные
cpio	Копирует файлы на архивные носители и в каталоги; также может выполнять обратное копирование
fdformat	Форматирует дискеты
flcopy	Копирует данные на дискету или с дискеты
format	Форматирует дискеты
fsck	Проверяет целостность файловой системы и при необходимости исправляет ошибки в интерактивном режиме
pack	Сжимает файлы
restore	Копирует с локального устройства файловые системы и файлы, для которых ранее были созданы резервные копии командой **backup**
tapechk	Проверяет целостность данных на магнитной ленте
tar	Команда для работы с архивами
tcopy	Копирует содержимое магнитной ленты
uncompress	Сжимает и разворачивает данные
unpack	Разворачивает файлы

Управление резервным копированием системы

Существует несколько способов резервного копирования системы и ее восстановления.

Резервное копирование пользовательских файлов или файловых систем:

Существует два способа резервного копирования файлов и файловых систем: команды быстрого доступа SMIT **smit backfile** и **smit backfilesys**, а также команда **backup**.

- При резервном копировании файловых систем по i-узлам следует сначала размонтировать те файловые системы, которые могут использоваться в данный момент.

  Внимание: Если вы попытаетесь создать резервную копию смонтированной файловой системы, то будет выдано предупреждающее сообщение. Работа команды **backup** будет продолжена, но полученная в результате резервная копия может не соответствовать действительности. Это ограничение не распространяется на корневую (/) файловую систему.
- Перед созданием резервной копии убедитесь, что накопитель на магнитной ленте был своевременно очищен.

Для резервного копирования файлов и файловых систем можно воспользоваться командой быстрого доступа SMIT **smit backfile** или **smit backfilesys**.

Интерфейс SMIT применяют для резервного копирования небольших файловых систем, например, локальной файловой системы `/home`. Учтите, что SMIT может создавать архивы только в том же формате, что и команда **backup**. Кроме того, интерфейс SMIT не позволяет применять некоторые флаги команды

backup. Если для резервного копирования потребуется несколько магнитных лент или дисков, программа SMIT может зависнуть. Дополнительная информация приведена в описании команды **backup** в книге *Справочник по командам, том 1*.

Команда **backup** применяется для создания резервной копии больших файловых систем или нескольких файловых систем. Можно указать объем копируемых данных, задав уровень резервного копирования (полное 0; дополняющее 1-9). Уровень резервного копирования можно задать только в случае применения команды **backup**.

Команда **backup** может создавать резервные копии в одном из двух форматов:

- Для создания резервной копии отдельных файлов, перечисленных по имени, применяется флаг **-i**.
- >Для создания резервной копии файловых систем с указанными i-узлами применяются параметры **-уровень** и **файловая-система**. Файловая система при восстановлении дефрагментируется.

Внимание: При сохранении файловой системы с указанным i-узлом могут неправильно обрабатываться файлы, у которых ИД пользователя (UID) или группы (GID) превышает 65535. UID или GID таких файлов при резервном копировании усекаются, и следовательно, после восстановления файлы будут иметь неверные значения UID и GID. В таких случаях файлы следует сохранять по имени.

Задачи резервного копирования пользовательских файлов и файловых систем

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Резервное копирование пользовательских файлов	`smit backfile`	1. Войдите в систему. 2. Создайте резервную копию: `find . -print | backup -ivf /dev/rmt0`
Резервное копирование пользовательских файловых систем	`smit backfilesys`	1. Размонтируйте файловые системы, для которых вы планируете создать резервные копии. Например: `umount all` или `umount /home /filesys1` 2. Проверьте целостность файловых систем. Например: `fsck /home /filesys1` 3. Создайте резервные копии по i-узлу. Например: `backup -5 -uf/dev/rmt0 /home/libr` 4. Восстановите файлы с помощью следующей команды: **restore -t**

Примечание: Если будет выдано сообщение об ошибке, то вам придется повторить весь процесс создания резервной копии.

Понятия, связанные с данным:

“Резервное копирование системы” на стр. 21
После завершения настройки системы рекомендуется создать резервную копию всех файловых систем, каталогов и файлов. При резервном копировании файловых систем можно восстановить файлы или файловые системы в случае сбоя жесткого диска. Есть несколько методов резервного копирования данных.

Восстановление резервных копий файлов:

Существует несколько различных способов восстановления данных, зависящих от типа команды, применявшейся для создания резервной копии.

Для правильного восстановления данных необходимо знать, каким образом была создана резервная копия или архив файлов. Каждая из процедур резервного копирования предоставляет информацию о последующем восстановлении. Например, команда **backup** позволяет создать резервную копию как всей файловой

системы, так и отдельных файлов с указанными именами. Соответственно и восстановление этих данных необходимо выполнять таким же способом: либо по имени, либо из файловой системы. Информация о команде **backup** приведена в разделе **backup**.

Вот некоторые команды восстановления резервных данных:

Элемент	Описание
restore	Копирует файлы, созданные командой **backup**. Дополнительная информация о применении этой команды приведена в следующем разделе.
rrestore	Создает резервные копии файловых систем удаленного компьютера в локальной системе. Дополнительная информация приведена в описании команды **rrestore**.
cpio	Копирует файлы в архив и из архива. Дополнительная информация приведена в описании команды **cpio**.
tar	Позволяет создать и работать с архивами **tar**. Дополнительная информация приведена в описании команды **tar**.
pax	(Совместима с POSIX) Читает и записывает архивы **tar** и **cpio**. Дополнительная информация приведена в описании команды **pax**.

В следующем разделе описаны команды **restore** и **smit**.

Примечание:

- Файлы следует восстанавливать тем же способом, которым они сохранялись. Например, если резервная копия файловой системы создавалась по именам, то и восстанавливать ее следует по именам.
- Если информация записана на нескольких дискетах, команда **restore** считывает ту из них, которая смонтирована, запрашивает следующую и ожидает ответа. Вставив новую дискету, нажмите клавишу Enter для продолжения восстановления файлов.

Восстановление файлов с помощью команды restore

С помощью команды **restore** можно восстанавливать файлы из резервных копий, созданных командой **backup**.

Примеры:

- Для просмотра списка имен ранее сохраненных файлов введите:
  ```
  restore -T
  ```
  Информация будет считана из устройства резервного копирования по умолчанию, `/dev/rfd0`. Если были созданы резервные копии отдельных файлов, то будут показаны только имена файлов. Если же была создана копия всей файловой системы, то дополнительно будет показан номер узла.
- Для восстановления файлов в главной файловой системе введите:
  ```
  restore -x -v
  ```
  Файл **-x** означает, что необходимо считать все файлы с резервного носителя и восстановить их в соответствующих каталогах файловой системы. Флаг **-v** выдает отчет о ходе восстановления файлов. Если восстанавливается вся файловая система, то имена файлов будут перечислены вместе с их номерами узлов. Если восстанавливаются отдельные файлы, то будут показаны только их имена.
- Для восстановления файла `/home/mike/manual/chap1` введите:
  ```
  restore -xv /home/mike/manual/chap1
  ```
  Команда восстановит файл `/home/mike/manual/chap1` с резервного носителя. Имя файла `/home/mike/manual/chap1` должно быть допустимым для команды **restore -T**.
- Для восстановления всех файлов каталога `manual` введите:
  ```
  restore -xdv manual
  ```
  Эта команда восстановит каталог `manual` и все файлы в нем. Если каталог `manual` не существует, он будет создан в текущем каталоге для размещения восстановленных файлов.

Сведения о синтаксисе приведены в описании команды **restore** книги *Справочник по командам, том 4*.

Восстановление файлов с помощью команды smit

Команда **smit** вызывает команду **restore**, которая восстанавливает файлы из резервных копий, созданных командой **backup**.

1. Введите в командной строке:

   ```
   smit restore
   ```

2. Введите значение в поле **Целевой каталог**. В этом каталоге будут храниться восстановленные файлы.
3. Введите имя устройства вывода в поле **Устройство резервного копирования** или **Файл**. Ниже приведен пример имени накопителя на магнитной ленте:

   ```
   /dev/rmt0
   ```

   Если устройство недоступно, появится сообщение примерно следующего содержания:

   ```
   Невозможно открыть /dev/rmtX, файл или
   каталог не существует.
   ```

   Это сообщение означает, что система не может обратиться к драйверу устройства, поскольку файл **rmtX** в каталоге /dev не найден. В каталоге /dev находятся только файлы доступных устройств.
4. В поле **Число блоков, считываемых за одну операцию** рекомендуется оставить значение по умолчанию.
5. Для восстановления указанной файловой системы или каталога нажмите Enter.

Создание удаленного архива:

С помощью этой процедуры можно архивировать файлы на удаленном накопителе на магнитной ленте.

Работающая система AIX не может смонтировать удаленный накопитель на магнитной ленте, как это делается в локальной системе, однако данные можно отправить на накопитель в удаленной системе с помощью команды **rsh**. Описанная процедура позволяет записать только одну кассету. Создание многотомных архивов требует установки специализированного программного обеспечения.

В описанной процедуре применяются следующие обозначения:

размер-блока
: Задает размер блока целевого накопителя на магнитной ленте.

удаленный-хост
: Имя целевой системы, в которой находится накопитель на магнитной ленте.

исходный-хост
: Имя исходной системы, в которой находятся архивируемые файлы.

/dev/rmt0
: Имя удаленного накопителя на магнитной ленте.

путь Полный путь к каталогу или файлу.

При применении следующих инструкций предполагается, что и в локальной и в удаленной системе применяется пользователь root.

1. Убедитесь, что у вас есть доступ к удаленной системе. У локальной системы должен быть доступ к системе с накопителем на магнитной ленте. (К целевой системе можно обращаться с помощью любого определенного в ней имени пользователя, но для выполнения большинства описанных шагов у этого пользователя должны быть права доступа root.)
2. С помощью текстового редактора создайте в корневом каталоге (/) целевой системы файл .rhosts, позволяющий исходной системе обращаться к целевой системе. В этом файле необходимо указать имя

хоста и ИД пользователя. Для определения имени исходного хоста, которое следует указать в файле `.rhosts`, вы можете воспользоваться следующей командой:

```
host исходный-IP-адрес
```

В данном примере в файл `.rhosts` добавляется следующая информация:

```
sourcehost.mynet.com   root
```

3. Сохраните файл и измените права доступа к нему с помощью следующей команды:

   ```
   chmod 600 .rhosts
   ```

4. С помощью команды **rsh** проверьте возможность доступа из исходной системы. Пример:

   ```
   rsh удаленный-хост
   ```

   При правильной настройке вы получите доступ к оболочке удаленной системы. Приглашение для ввода имени пользователя показано не будет. Для выхода из оболочки введите команду `exit`.

5. Выберите размер блока накопителя на магнитной ленте. Рекомендуется применять следующие значения:

Элемент	Описание
Размер блока для 9-дорожечных или 0,25-дюймовых носителей:	512
Размер блока для 8-мм и 4-мм носителей:	1024

   Если вы не уверены в выборе и хотите проверить текущий размер блока накопителя на магнитной ленте, то воспользуйтесь командой **tctl**. Например:

   ```
   tctl -f /dev/rmt0 status
   ```

   Для изменения размера блока накопителя на магнитной ленте воспользуйтесь командой **chdev**. Пример:

   ```
   chdev -l rmt0 -a block_size=1024
   ```

6. Создайте архив с помощью одного из следующих способов:

   Резервное копирование по имени

   Для удаленного создания архивной резервной копии по имени введите следующую команду:

   ```
   find путь -print | backup -ivqf- | rsh удаленный-хост \
       "dd of=/dev/rmt0 bs=размер-блока conv=sync"
   ```

   Резервное копирование по i-узлу

   Для удаленного создания архивной резервной копии по i-узлу сначала необходимо размонтировать файловую систему, а затем воспользоваться командой **backup**. Например:

   ```
   umount /myfs
   backup -0 -uf- /myfs | rsh удаленный-хост \
        "dd of=/dev/rmt0 bs=размер-блока conv=sync"
   ```

   Создание архива и копирование его на удаленный накопитель на магнитной ленте

   Для создания архива и копирования его на удаленный накопитель на магнитной ленте воспользуйтесь следующей командой:

   ```
   find путь -print | \
   cpio -ovcB | rsh удаленный-хост \
       "dd ibs=5120 obs=размер-блока of=/dev/rmt0"
   ```

   Создание архива tar

   Для создания удаленного архива **tar** введите следующую команду:

   ```
   tar -cvdf- путь | rsh удаленный-хост \
       "dd of=/dev/rmt0 bs=размер-блока conv=sync"
   ```

   Удаленное создание дампа

   Для удаленного создания дампа файловой системы `/myfs` введите следующую команду:

   ```
   rdump -u -0 -f удаленный хост:/dev/rmt0 /myfs
   ```

   Флаг **-u** означает, что необходимо обновить запись текущего уровня резервной копии в файле `/etc/dumpdates`. **-0** - это значение флага *Уровень*. Уровень резервного копирования 0 указывает,

что необходимо создать резервную копию всех файлов, находящихся в каталоге /myfs. Дополнительная информация приведена в описании команды **rdump** в книге *Справочник по командам, том 4*.

7. Восстановление данных из удаленного архива можно выполнить несколькими способами:

   Восстановление из удаленного архива по имени

   Для восстановления данных из удаленного архива по имени воспользуйтесь следующей командой:

   ```
   rsh удаленный-хост "dd if=/dev/rmt0 \
   bs=размер-блока" | restore \
        -xvqdf- путь
   ```

   Восстановление из удаленного архива по i-узлу

   Для восстановления данных из удаленного архива по i-узлу воспользуйтесь следующей командой:

   ```
   rsh удаленный-хост "dd if=/dev/rmt0 \
   bs=размер-блока" | restore \
       -xvqf- путь
   ```

   Восстановление из удаленного архива cpio

   Для восстановления данных из удаленного архива, созданного командой **cpio**, введите следующую команду:

   ```
   rsh удаленный-хост \
   "dd if=/dev/rmt0 ibs=размер-блока obs=5120" | \
        cpio -icvdumB
   ```

   Восстановление из архива tar

   Для восстановления данных из удаленного архива **tar** введите следующую команду:

   ```
   rsh удаленный-хост "dd if=/dev/rmt0 \
   bs=размер-блока" | tar -xvpf- путь
   ```

   Восстановление удаленного дампа

   Для восстановления удаленного дампа файловой системы /myfs введите следующую команду:

   ```
   cd /myfs
   rrestore -rvf удаленный-хост:/dev/rmt0
   ```

Восстановление пользовательских файлов из резервной копии:

При восстановлении данных из резервной копии наибольшую трудность представляет определение магнитной ленты, на которой записан нужный файл. С помощью команды **restore -T** можно просмотреть список содержимого архива. Рекомендуется восстанавливать файлы в каталог /tmp, чтобы случайно не перезаписать другие пользовательские файлы.

Убедитесь, что устройство подключено и доступно. Введите команду:

```
lsdev -C | pg
```

Если при создании резервной копии применялась схема дополняющего резервного копирования, то следует узнать у пользователя, когда файл был в последний раз изменен. Эта информация поможет вам определить, в какой из дополняющих резервных копий записан файл. Если время последнего изменения определить не возможно, то начните просматривать дополняющие резервные копии в обратном порядке (7, 6, 5,...). При работе с дополняющими резервными копиями файловой системы поиск и восстановление файлов удобно выполнять с помощью команды **restore** с флагом **-i** (интерактивный режим). (Интерактивный режим удобен также при восстановлении файлов отдельных пользователей из резервной копии файловой системы /home.)

В следующей таблице описаны процедуры, позволяющие выполнить полное (уровень 0) восстановление каталога или файловой системы.

Задачи восстановления из резервной копии		
Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Восстановление отдельных пользовательских файлов	**smit restfile**	См. описание команды **restore**.
Восстановление пользовательской файловой системы	**smit restfilesys**	1. **mkfs /dev/hd1** 2. **mount /dev/hd1 /filesys** 3. **cd /filesys** 4. **restore -r**
Восстановление пользовательской группы томов	**smit restvg**	См. описание команды **restvg -q**.

Задачи, связанные с данной:

“Устранение повреждений файловой системы” на стр. 447
Файловая система повреждается, если искажается информация об i-узлах или главных блоках в структуре каталогов.

Восстановление доступа к удаленной или не имеющей связи системной библиотеке:

Если существующая библиотека **libc.a** недоступна, то операционная система не будет распознавать большинство команд.

Чаще всего данная ошибка бывает вызвана следующими причинами:

- Связь в `/usr/lib` больше не существует.
- Удален файл в `/usr/ccs/lib`

Ниже описана процедура восстановления доступа к библиотеке **libc.a**. Эта процедура требует выключения системы. Во избежание потери данных следует планировать ее выполнение на время, соответствующее минимальной загрузке системы.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

Информация, связанная с данной:

Команда mount

Команда unmount

Команда reboot

Восстановление удаленной символьной связи:

Для восстановления символьной связи библиотеки `/usr/lib/libc.a` с `/usr/ccs/lib/libc.a` выполните следующие действия.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. От имени пользователя с правами доступа root укажите в переменной среды **LIBPATH** каталог `/usr/ccs/lib`. Для этого введите следующие команды:

   ```
   # LIBPATH=/usr/ccs/lib:/usr/lib
   # export LIBPATH
   ```

   Теперь вы сможете выполнять системные команды.
2. Для восстановления связей библиотеки `/usr/lib/libc.a` и каталога `/lib` с каталогом `/usr/lib` введите следующие команды:

   ```
   ln -s /usr/ccs/lib/libc.a /usr/lib/libc.a
   ln -s /usr/lib /lib
   ```

Теперь вы сможете выполнять команды операционной системы. Если доступа к оболочке по-прежнему нет, то пропустите остальные инструкции и перейдите к следующему разделу, “Восстановление удаленного файла системной библиотеки”.

3. Для сброса переменной среды LIBPATH введите следующую команду:

   ```
   unset LIBPATH
   ```

Восстановление удаленного файла системной библиотеки:

Описанная в этом разделе процедура восстановления удаленного файла системной библиотеки требует выключения системы. Во время ее выполнения система перезагружается и файл восстанавливается с последней ленты **mksysb**.

1. Перед перезагрузкой убедитесь, что в файле `bosinst.data` параметру **PROMPT** присвоено значение `yes`.
2. Установите в накопитель на магнитной ленте последнюю ленту **mksysb**. Версия операционной системы и уровень обслуживания в образе **mksysb** *обязательно* должны совпадать с аналогичными параметрами установленной системы. Если восстановить библиотеку `libc.a` из образа **mksysb** другого уровня, то вы не сможете выполнять команды операционной системы.
3. Перезагрузите систему.
4. При появлении меню со значками или после выдачи двойного звукового сигнала нажимайте клавишу F1 до тех пор, пока не появится меню Службы управления системой.
5. Выберите опцию **Выбор операционной системы**.
6. Выберите опцию **Установить из**.
7. Укажите устройство, в котором находится лента **mksysb**, и выберите опцию **Установить**. Перед появлением следующего приглашения может пройти несколько минут.
8. Определите системную консоль, нажав F1, а затем - Enter.
9. Укажите номер, соответствующий предпочитаемому языку, и нажмите Enter.
10. Выберите опцию **Запуск в режиме обслуживания для восстановления системы**, введите `3` и нажмите Enter.
11. Выберите **Доступ к корневой группе томов**. Будет показано сообщение с информацией о том, что в случае изменения корневой группы томов вы не сможете вернуться к меню установки без перезагрузки.
12. Введите ответ `0` и нажмите Enter.
13. Укажите номер показанной в списке корневой группы томов и нажмите Enter.
14. Выберите опцию **Доступ к группе томов**. Для этого укажите `2` и нажмите Enter.
15. С помощью следующих команд смонтируйте файловые системы `/` (корневая) и `/usr`:

    ```
    mount /dev/hd4 /mnt
    mount /dev/hd2 /mnt/usr
    cd /mnt
    ```

16. Для восстановления символьной связи библиотеки `libc.a` введите следующую команду:

    ```
    ln -s /usr/ccs/lib/libc.a /mnt/usr/lib/libc.a
    ```

    После выполнения команды выполните одну из следующих операций:

    - Если команда выполнена успешно, перейдите к шагу 20.
    - Если показано сообщение о том, что связь уже существует, то перейдите к шагу 17.

17. Задайте размер блока накопителя на магнитной ленте. Для этого введите следующие команды, в которых *X* обозначает номер накопителя.

    ```
    tctl -f /dev/rmtX rewind
    tctl -f /dev/rmtX.1 fsf 1
    restbyname -xvqf /dev/rmtX.1 ./tapeblksz
    cat tapeblksz
    ```

    Если значение, возвращенное командой **cat tapeblksz**, *не равно* 512, то введите следующие команды, заменив *Y* на значение, показанное командой **cat tapeblksz**:

```
ln -sf /mnt/usr/lib/methods /etc/methods
/etc/methods/chgdevn -l rmtX -a block_size=Y
```

Должно быть показано сообщение о том, что устройство rmt*X* изменено.

18. С помощью следующей команды перемотайте ленту в кассете для восстановления библиотеки (*X* обозначает номер накопителя):

    ```
    tctl -f /dev/rmtX rewind
    tctl -f /dev/rmtX.1 fsf 3
    ```

19. С помощью одной из следующих команд восстановите отсутствующую библиотеку (*X* - номер накопителя:)
    - Для восстановления только библиотеки `libc.a` введите следующую команду:

      ```
      restbyname -xvqf /dev/rmtX.1 ./usr/ccs/lib/libc.a
      ```

    - Для восстановления каталога `/usr/ccs/lib` введите следующую команду:

      ```
      restbyname -xvqf /dev/rmtX.1 ./usr/ccs/lib
      ```

    - Для восстановления каталога `/usr/ccs/bin` введите следующую команду:

      ```
      restbyname -xvqf /dev/rmtX.1 ./usr/ccs/bin
      ```

20. С помощью следующей команды запишите все данные на диск:

    ```
    cd /mnt/usr/sbin
    ./sync;./sync;./sync
    ```

21. Размонтируйте файловые системы `/usr` и `/` (корневая) с помощью следующих команд:

    ```
    cd /
    umount /dev/hd2
    umount /dev/hd4
    ```

    В случае сбоя любой из команд **umount** выключите питание системы, затем снова включите его и повторите процедуру.

22. Перезагрузите систему с помощью следующей команды:

    ```
    reboot
    ```

После перезагрузки вы сможете выполнять команды операционной системы.

Восстановление поврежденного загрузочного образа:

Ниже описана процедура идентификации и восстановления поврежденного загрузочного образа.

Если система работает и вы знаете, что загрузочный образ поврежден или удален, то вы можете создать его повторно запустив от имени пользователя с правами доступа root команду **bosboot**.

Внимание: Ни в коем случае не перезагружайте систему, если вы подозреваете, что загрузочный образ поврежден.

В данной процедуре предполагается, что система не может правильно перезагрузиться из-за повреждения загрузочного образа. Если это возможно, для предотвращения потери данных и функциональности запланируйте восстановление системы на время, соответствующее минимальной загрузке.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Установите носитель продукта в соответствующее устройство.
2. Включите систему, выполнив инструкции, прилагаемые к системе.
3. В меню **Службы управления системой** выберите пункт **Альтернативная загрузка**.
4. На следующем экране выберите **Установить из**.
5. Укажите устройство, в котором находится носитель продукта и выберите опцию **Установить**.

6. Щелкните на значке версии AIX.
7. Следуйте инструкциям по установке вплоть до момента выбора режима установки. Выберите **Запуск в режиме обслуживания для восстановления системы**.
8. Выберите **Доступ к корневой группе томов**.
9. Следуйте инструкциям до выбора опции **Получить доступ к группе томов и запустить оболочку**.
10. Создайте загрузочный образ заново с помощью команды **bosboot**. Пример:

    ```
    bosboot -a -d /dev/hdisk0
    ```

    Если при выполнении команды возникнет ошибка и будет показано сообщение

    ```
    0301-165 bosboot: WARNING! bosboot failed - do not attempt to boot device.
    ```

    Попытайтесь устранить неполадку, воспользовавшись одним из перечисленных ниже способов, а затем снова введите команду **bosboot** для создания загрузочного образа:

    - Удалите загрузочный логический том по умолчанию (hd5) и создайте новый hd5.

    или

    - Выполните диагностику жесткого диска. При необходимости восстановите или замените диск.

    Если команда **bosboot** по-прежнему выдает сообщение об ошибке, обратитесь в сервисное представительство.

    Внимание: Если команда **bosboot** выдает сообщение об ошибке при создании загрузочного образа, то не пытайтесь перезагружать систему.
11. После успешного завершения команды **bosboot** введите команду **reboot** для перезапуска системы.

Понятия, связанные с данным:

“Загрузка системы” на стр. 4
После загрузки Базовой операционной системы система выполняет достаточно сложный набор задач. Обычно эти задачи выполняются автоматически.

Информация, связанная с данной:

Команда bosboot

Создание резервной копии JFS:

При создании резервной копии смонтированной файловой системы JFS или JFS2 фактически создается статический образ логического тома, содержащего файловую систему.

Для создания резервной копии смонтированной JFS необходимо, чтобы для логического тома, содержащего файловую систему, и для логического тома, содержащего протокол файловой системы, применялась зеркальная защита.

Примечание: В связи с тем, что запись файлов выполняется асинхронно, выделенная копия диска может включать не все данные, записанные непосредственно перед выделением этого диска. Все изменения, внесенные после запуска процедуры выделения, могут отсутствовать в полученной резервной копии. В связи с этим, при отделении зеркальной копии рекомендуется свести работу с файловой системой к минимуму.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

Для выделения зеркальной копии файловой системы `/home/xyz` в новой точке монтирования `/jfsstaticcopy` введите следующую команду:

```
chfs -a splitcopy=/jfsstaticcopy /home/xyz
```

Атрибут **copy** позволяет выбрать одну из зеркальных копий. По умолчанию применяется вторая копия. Например:

```
chfs -a splitcopy=/jfsstaticcopy -a copy=1 /home/xyz
```

Теперь копия файловой системы доступна для чтения в каталоге `/jfsstaticcopy`. Изменения, внесенные в исходную файловую систему после копирования, не будут отражены в резервной копии.

Для интеграции выделенного образа JFS с исходной зеркальной копией в точке монтирования `/testcopy` введите следующую команду:

```
rmfs /testcopy
```

Команда **rmfs** отменяет выделение копии файловой системы, позволяя вновь использовать ее в качестве зеркальной копии.

Создание и резервное копирование моментальной копии JFS2:

Можно создавать моментальные копии смонтированных файловых систем JFS2, создавая соответствующий определенному моменту времени образ файловой системы на уровне блоков.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

Образ моментальной копии останется стабильным даже в том случае, если файловая система, с которой создается моментальная копия, продолжает изменяться. В моментальной копии сохраняются права доступа, действовавшие в *копируемой файловой системе* в момент создания копии.

В следующем сценарии создается моментальная копия, которая затем сохраняется на съемном носителе, без размонтирования или стабилизации файловой системы. При этом применяется только одна команда: **backsnap**. Вы также можете использовать моментальную копию для других целей, например, для доступа к файлам и каталогом в том виде, которому они соответствовали в момент создания моментальной копии. С помощью программы SMIT или команд **backsnap** и **snapshot** можно выполнять различные процедуры с моментальными копиями.

Для создания моментальной копии файловой системы `/home/abc/test` и ее сохранения (по имени) на накопителе `/dev/rmt0` введите следующую команду:

```
backsnap -m /tmp/snapshot -s size=16M -i f/dev/rmt0 /home/abc/test
```

Эта команда создаст для моментальной копии файловой системы JFS2 (`/home/abc/test`) логический том размером 16 Мб. Моментальная копия будет смонтирована в каталог `/tmp/snapshot`, а затем сохранена на магнитной ленте. После выполнения резервного копирования моментальная копия остается смонтированной. Для удаления моментальной копии после завершения сохранения укажите в команде **backsnap** флаг **-R**.

Понятия, связанные с данным:

“Файловые системы” на стр. 425
Файловая система - это иерархическая структура (дерево) файлов и каталогов.

Информация, связанная с данной:

команда backsnap

команда chfs

команда rmfs

команда snapshot

Создание и резервное копирование внешней моментальной копии JFS2:

Можно создавать моментальные копии смонтированных файловых систем JFS2, создавая соответствующий определенному моменту времени образ файловой системы на уровне блоков.

Образ моментальной копии останется стабильным даже в том случае, если файловая система, с которой создается моментальная копия, продолжает изменяться. В моментальной копии сохраняются права доступа, действовавшие в *копируемой файловой системе* в момент создания копии.

В следующем сценарии создается внешняя моментальная копия, которая затем сохраняется на съемном носителе, без размонтирования или стабилизации файловой системы. При этом применяется только одна команда: **backsnap**. Вы также можете использовать моментальную копию для других целей, например, для доступа к файлам и каталогом в том виде, которому они соответствовали в момент создания моментальной копии. С помощью SMIT или команд **backsnap** и **snapshot** вы можете создавать различные моментальные копии.

Для создания внешней моментальной копии файловой системы `/home/abc/test` и ее сохранения ,по имени, на накопителе `/dev/rmt0` введите следующую команду:

```
backsnap -m /tmp/snapshot -s size=16M -if/dev/rmt0 /home/abc/test
```

Эта команда создаст для моментальной копии файловой системы JFS2 (`/home/abc/test`) логический том размером 16 Мб. Моментальная копия будет смонтирована в каталог `/tmp/snapshot`, а затем сохранена на магнитной ленте. После завершения сохранения моментальная копия будет размонтирована, но останется при этом доступной. Для удаления моментальной копии после завершения сохранения укажите в команде **backsnap** флаг **-R**.

Понятия, связанные с данным:

“Файловые системы” на стр. 425
Файловая система - это иерархическая структура (дерево) файлов и каталогов.

Создание и резервное копирование внутренней моментальной копии JFS2:

Можно создавать моментальные копии смонтированных файловых систем JFS2, создавая соответствующий определенному моменту времени образ файловой системы на уровне блоков.

Образ моментальной копии останется стабильным даже в том случае, если файловая система, с которой создается моментальная копия, продолжает изменяться. В моментальной копии сохраняются права доступа, действовавшие в *копируемой файловой системе* в момент создания копии.

В следующем сценарии создается внутренняя моментальная копия, которая затем сохраняется на съемном носителе, без размонтирования или стабилизации файловой системы. При этом применяется только одна команда: **backsnap**. Вы также можете использовать моментальную копию для других целей, например, для доступа к файлам и каталогом в том виде, которому они соответствовали в момент создания моментальной копии. С помощью SMIT или команд **backsnap** и **snapshot** вы можете создавать различные моментальные копии.

Для создания внутренней моментальной копии файловой системы `/home/abc/test` и ее сохранения ,по имени, на накопителе `/dev/rmt0` введите следующую команду:

```
backsnap -n mysnapshot -if/dev/rmt0 /home/abc/test
```

С помощью предыдущей команды была создана внутренняя моментальная копия, по имени `mysnapshot`, файловой системы `/home/abc/test`. Доступ к моментальной копии можно получить из каталога `/home/abc/test/.snapshot/mysnapshot`, которая затем будет сохранена на магнитной ленте. Для удаления моментальной копии после завершения сохранения укажите в команде **backsnap** флаг **-R**.

Понятия, связанные с данным:

“Файловые системы” на стр. 425
Файловая система - это иерархическая структура (дерево) файлов и каталогов.

Сжатие файлов (команды compress и pack):

Команды **compress** и **pack** применяются для сжатия файлов в целях экономии памяти.

Команды **uncompress** и **unpack** позволяют восстанавливать упакованные файлы.

Сжатие и развертывание файлов отнимает определенное время, но зато позволяет сэкономить место на носителе.

Для сжатия файловой системы выполните одно из следующих действий:
- Укажите флаг **-p** в команде **backup**.
- Воспользуйтесь командой **compress** или **pack**.

Сжатие файлов применяется в следующих целях:
- Для экономии времени и средств путем сжатия файлов перед их отправкой по сети.
- Для экономии памяти и архивирования системных ресурсов:
  - Если вы сожмете файловую систему перед ее сохранением, то она займет меньше места на магнитной ленте.
  - Сжатие файлов протоколов, которые создаются сценариями оболочки, выполняемыми ночью; сценарий может сжимать файлы перед завершением работы.
  - Сжатие файлов, которые в данный момент не нужны. Например, если файлы принадлежат пользователю, который отсутствует в течение длительного времени, то их можно сжать и поместить в архив **tar** на диске или магнитной ленте, а впоследствии восстановить.

Примечание:
- В файловой системе может не хватить памяти для выполнения команды **compress**. Эта команда сначала создает сжатые файлы, а затем удаляет исходные файлы, поэтому ей требуется примерно в полтора раза больше памяти по сравнению с общим объемом всех файлов.
- Сжатие файла может быть не выполнено из-за того, что он уже сжат. Если команде **compress** не удается сократить размер файла, возникает ошибка.

Подробные сведения о выводе команды **compress** приведены в ее описании, однако все эти ошибки можно разбить на несколько категорий:
- Во время сжатия в файловой системе может возникнуть нехватка памяти. Команда **compress** сначала создает все сжатые файлы, а затем удаляет оригиналы, поэтому перед ее запуском объем свободной памяти в файловой системе должен быть не менее 50-100% объема сжимаемых файлов.
- Сжатие файла может быть не выполнено из-за того, что он уже сжат. Если команде **compress** не удается уменьшить размер файла, она выдает сообщение об ошибке.

Сжатие файлов с помощью команды compress:

Команда **compress** сокращает размер файлов по алгоритму кодирования Лемпела-Зива.

Каждый исходный файл, указанный в параметре *Файл* , будет заменен на сжатый файл, к имени которого будут добавлены символы `.Z`. Сжатый файл наследует принадлежность, режимы доступа и даты обращения и изменения от исходного файла. Если файлы не указаны, то команда преобразует стандартный ввод в стандартный вывод. Если сжатие не сокращает размер файла, то в стандартный протокол ошибок будет занесено сообщение, и исходный файл заменен не будет.

Команда **uncompress** применяется для разворачивания сжатых файлов.

Объем сжатия зависит от размера файла, количества битов на код, указанного в переменной *Биты*, и распределения часто встречающихся строк. Как правило, в результате сжатия исходный код или английский

текст сокращается на 50-60 процентов. Команда **compress** обычно сжимает данные быстрее и эффективнее, чем команда **pack**, применяющая алгоритм кодирования Хафмана.

Например, для сжатия файла `foo` и выдачи значения коэффициента сжатия в процентах в стандартный протокол ошибок введите:

```
compress -v foo
```

Сведения о синтаксисе приведены в описании команды **compress** книги *Справочник по командам, том 1*.

Сжатие файлов с помощью команды pack:

Команда **pack** сжимает файл или файлы, указанные в параметре *Файл*, по алгоритму кодирования Хафмана.

В результате сжатия исходный файл будет заменен на упакованный файл с именем, основанным на имени исходного файла (*Файл*`.z`), и теми же режимами доступа и датами обращения и изменения. Имя исходного файла должно состоять не более чем из 253 байтов, чтобы к нему можно было добавить суффикс `.z`. В случае успешного выполнения команды **pack** исходный файл будет удален.

Для разворачивания файлов применяется команда **unpack**.

Если команда **pack** не может сократить размер файла, то обработка прекращается и выдается сообщение о том, что сократить объем занимаемого пространства невозможно. (Чаще всего такое происходит с небольшими файлами или с файлами с равномерным распределением символов.) Эффективность упаковки зависит от размера исходного файла и распределения символов. Поскольку в начале каждого файла `.z` записана структура расшифровки, при сжатии файлов размером менее трех блоков место на диске не экономится. Текстовые файлы обычно уменьшаются на 25-40 процентов.

Значение возврата для команды **pack** - это количество файлов, которые не удалось упаковать. Упаковка не выполняется в следующих случаях:

- Файл уже сжат.
- Размер имени файла превышает 253 байта.
- Для файла созданы связи.
- Вместо файла указан каталог.
- Не удалось открыть файл.
- Упаковка не приведет к сокращению размера файла.
- Уже существует файл с именем *Файл*`.z`.
- Не удалось создать файл `.z`.
- Во время обработки произошла ошибка ввода-вывода.

Например, для упаковки файлов `chap1` и `chap2` введите:

```
pack chap1 chap2
```

Файлы `chap1` и `chap2` будут заменены на упакованные версии `chap1.z` и `chap2.z`. Команда **pack** укажет, насколько сократился размер каждого файла в процентах.

Сведения о синтаксисе приведены в описании команды **pack** книги *Справочник по командам, том 4*.

Развертывание сжатых файлов (команды uncompress и unpack):

Команды **uncompress** и **unpack** применяются для разворачивания сжатых файлов.

Распаковка файлов с помощью команды uncompress

Команда **uncompress** восстанавливает исходные версии файлов, сжатых командой **compress**. Все файлы, указанные в параметре *Файл*, будут удалены и заменены развернутыми версиями.

Развернутый файл имеет то же имя, что и сжатая версия, но без расширения `.Z`. Развернутый файл сохраняет владельца, режимы доступа и время последнего чтения и записи исходного файла. Если файлы не указаны, стандартный ввод будет развертываться в стандартный вывод.

Схожая с командой **uncompress** команда **zcat** всегда записывает результат развертывания файла в стандартный вывод.

Например, для развертывания файла `foo` введите:

```
uncompress foo
```

Полное описание синтаксиса приведено в описании команды **uncompress** в книге *Справочник по командам, том 5*.

Распаковка файлов с помощью команды unpack

Команда **unpack** развертывает файлы, созданные командой **pack**. Для каждого из указанных имен файлов команда **unpack** ищет файл *Имя-файла*`.z`. Если сжатый файл с таким именем существует, команда **unpack** заменяет его развернутой версией. Затем команда **unpack** переименовывает новый файл, удаляя суффикс `.z`. Развернутый файл сохраняет режимы доступа, дату последнего чтения и записи, а также владельца исходного упакованного файла.

Команда **unpack** работает только с файлами, заканчивающимися на `.z`. Если вы укажете имя файла без суффикса `.z`, команда **unpack** сама добавит этот суффикс при поиске файла в каталоге.

Код завершения команды **unpack** представляет собой число файлов, которые не удалось развернуть. Ошибка при развертывании файла может возникнуть по одной из следующих причин:

- Длина имени файла (без учета суффикса `.z`) превышает 253 байта.
- Не удалось открыть файл.
- Файл не сжат.
- Уже существует файл с именем, которое будет присвоено развернутому файлу.
- Не удалось создать развернутый файл.

Примечание: Команда **unpack** записывает предупреждения в стандартный поток вывода ошибок. При развертывании файла ему присваивается новый номер i-узла. Все остальные файлы, связанные с номером i-узла сжатого файла, останутся сжатыми.

Например, для развертывания сжатых файлов `chap1.z` и `chap2.z` введите:

```
unpack chap1.z chap2
```

Файлы `chap1.z` и `chap2.z` будут развернуты и заменены на файлы с именами `chap1` и `chap2`.

Примечание: В команде **unpack** имена файлов можно указывать как с суффиксом `.z`, так и без него.

Полное описание синтаксиса приведено в описании команды **unpack** в книге *Справочник по командам, том 5*.

Резервное копирование системного образа и пользовательских групп томов

Группа томов rootvg представляет собой жесткий диск или группу дисков, содержащих файлы загрузки системы, BOS, данные о конфигурации, а также дополнительные программные продукты. *Пользовательская группа томов* (не *rootvg*) обычно применяется для хранения файлов данных и программного обеспечения.

Резервную копию системы и групп томов можно создать с помощью SMIT или обычных команд. Резервное копирование системы преследует две цели. Первая - восстановление поврежденной системы с помощью резервной копии. Вторая - перенос установленного и настроенного программного обеспечения из одной системы в другую.

Для создания резервных копий, которые можно сохранить на магнитной ленте или в файле, в SMIT применяется команда **mksysb**. При выборе магнитной ленты программа backup записывает на ленту *загрузочный образ*, с помощью которого можно выполнять установку.

Примечание:

- Загрузочные ленты нельзя создавать на персональных компьютерах PowerPC и применять для их загрузки.
- Для создания резервной копии с помощью программы SMIT необходимо установить набор файлов `sysbr` из пакета `bos.sysmgt`.

Понятия, связанные с данным:

“Резервные копии” на стр. 25
Как правило, резервные копии пользовательских и системных данных сохраняют на случай их непреднамеренного удаления или отказа жесткого диска. Работать с резервными копиями гораздо проще, когда системные и пользовательские данные хранятся отдельно.

Информация, связанная с данной:

Установка дополнительного программного обеспечения и обновлений

Создание резервной копии системного образа и пользовательских групп томов:

Можно создать резервные копии для системных образов и пользовательских групп томов.

Перед созданием резервной копии группы томов rootvg:

- Подключите к системе все аппаратное обеспечение, включая накопители на магнитной ленте и дисководы для компакт-дисков.
- Для выполнения данной процедуры необходим набор файлов `sysbr` из программного пакета Инструменты и приложения для управления системой BOS. С помощью следующей команды определите, установлен ли в системе набор файлов `sysbr`:

  ```
  lslpp -l bos.sysmgt.sysbr
  ```

  Если набор файлов `sysbr` в системе присутствует, можно начать процедуру резервного копирования.

  Если команда **lslpp** не нашла набор файлов `sysbr`, его необходимо установить перед началом резервного копирования.

  ```
  installp -agqXd устройство bos.sysmgt.sysbr
  ```

  где `устройство` - это название устройства, на котором расположено программное обеспечение; например, `/dev/rmt0` в случае накопителя на магнитной ленте.

Перед созданием резервной копии пользовательской группы томов:

- Перед копированием необходимо подключить группу томов и смонтировать все файловые системы.

  Внимание: Команда **savevg** стирает все данные, изначально находившиеся на носителе.
- Во избежание ошибок проверьте, не нуждается ли устройство резервного копирования в чистке.

Ниже описана процедура создания установочного образа системы.

Задачи создания резервной копии системы

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Создание резервной копии группы томов **rootvg**	1. Войдите в систему как пользователь root. 2. Смонтируйте файловые системы, предназначенные для резервного копирования.[1] **smit mountfs** 3. Размонтируйте все локальные каталоги, смонтированные в другом локальном каталоге. **smit umountfs** 4. Освободите по крайней мере 8,8 Мб дискового пространства в каталоге `/tmp`.[2] 5. Создайте резервную копию: **smit mksysb** 6. Защитите носитель резервной копии от записи. 7. Запишите все пользовательские пароли, вошедшие в резервную копию.	1. Войдите в систему как пользователь root. 2. Смонтируйте файловые системы, предназначенные для резервного копирования.[1] См. описание команды **mount**. 3. Размонтируйте все локальные каталоги, смонтированные в другом локальном каталоге. См. описание команды **umount**. 4. Освободите по крайней мере 8,8 Мб дискового пространства в каталоге `/tmp`.[2] 5. Создайте резервную копию. См. описание команды **mksysb**. 6. Защитите носитель резервной копии от записи. 7. Запишите все пользовательские пароли, вошедшие в резервную копию.
Проверка созданной магнитной ленты[3]	**smit lsmksysb**	
Создание резервной копии пользовательской группы томов[4]	**smit savevg**	1. В случае необходимости измените размер файловой системы перед резервным копированием.[5] **mkvgdata** *имя-группы-томов*, а затем измените `/tmp/vgdata/`*имя-группы-томов*`/`*имя-группы-томов*`.data` 2. Сохраните группу томов. См. описание команды **savevg**.

Примечание:

1. Команда **mksysb** не создает резервные копии файловых систем, смонтированных с помощью NFS.
2. Команде **mksysb** во время резервного копирования необходимо дополнительное дисковое пространство. Определите объем свободной памяти в каталоге `/tmp` с помощью команды **df**. При необходимости измените размер файловой системы с помощью команды **chfs**.
3. Ниже описана процедура просмотра содержания магнитной ленты, созданной командой **mksysb**. При этом выполняется проверка целостности большей части данных ленты, но не ее пригодности для загрузки и установки. Проверить загрузочный образ ленты **mksysb** можно только одним способом: непосредственно загрузить с нее систему.
4. Если вы не хотите включать некоторые файлы пользовательской группы томов в резервный образ, создайте файл `/etc/exclude.`*группа-томов*, где *группа-томов* - это название копируемой группы томов. Затем с помощью редактора внесите в файл `/etc/exclude.`*группа-томов* шаблоны имен файлов, которые должны быть исключены из резервного образа. С помощью этих стандартных шаблонов команда **grep** определяет, какие файлы следует исключить из резервной копии.
5. Если вы изменили размер файловой системы, отредактировав файл *группа-томов*`.data`, то не задавайте для команды **-i** флаги **-m** и **savevg**, поскольку в этом случае файл *группа-томов*`.data` будет перезаписан.

Информация, связанная с данной:

Установка дополнительного программного обеспечения и обновлений

Установка резервных копий системы

Настройка перед резервным копированием:

Рекомендуется настроить исходную систему перед созданием резервной копии. Если же вы планируете применять резервную копию для установки системы с другой конфигурацией, то копию следует создать *до* настройки исходной системы.

Исходная система - это система, в которой создается резервная копия. *Целевая* система - система, на которую устанавливается резервная копия.

Система автоматически устанавливает поддержку только тех устройств, которые есть в данном компьютере. Таким образом, если вы планируете применять резервную копию для установки других систем, то перед ее созданием в исходной системе может потребоваться установить дополнительные устройства.

Для установки поддержки дополнительных устройств в исходной системе воспользуйтесь командой SMIT `smit devinst`.

- Если в исходной и целевой системах достаточно дискового пространства, установите поддержку всех устройств.
- Если объем свободного дискового пространства в исходной и целевой системах ограничен, то устанавливайте поддержку только необходимых устройств.

С помощью резервной копии в целевую систему переносятся следующие параметры исходной системы:

- Параметры пространства подкачки
- Параметры логических томов
- Информация rootvg
- Расположение логических разделов (если выбрана соответствующая опция).

Информация, связанная с данной:

Установка дополнительного программного обеспечения и обновлений

Настройка установленного продукта

Монтирование и размонтирование файловой системы:

Перед выполнение резервного монтирования следует смонтировать все файловые системы, подлежащие резервному копированию, и размонтировать файловые системы, создавать резервные копии для которых не требуется.

Способы резервного копирования сохраняет только смонтированные файловые системы в группе томов rootvg. Следовательно, перед выполнением этой процедуры нужно смонтировать все файловые системы, которые нужно сохранить. Кроме того, нужно размонтировать те файловые системы, которые сохранять *не нужно*.

Если какой-либо каталог смонтирован в другом каталоге той же файловой системы, то он будет сохранен дважды. Например, если вы смонтировали каталог `/tmp` в каталоге `/usr/tmp`, то файлы из каталога `/tmp` будут сохранены дважды. Многократное сохранение одних и тех же файлов может привести к превышению максимального допустимого числа файлов в файловой системе и ошибкам при последующем восстановлении резервной копии.

Замечания о безопасности для резервных копий:

При установке резервной копии не рекомендуется копировать в целевую систему пароли и сетевые адреса исходной системы.

Кроме того, наличие в сети нескольких систем с одинаковыми адресами может нарушить ее нормальную работу.

Восстановление резервного образа:

При восстановлении резервной копии система проверяет, достаточно ли дискового пространства для размещения всех логических томов, сохраненных в резервной копии. Если памяти достаточно, резервная копия восстанавливается полностью. В противном случае вам будет предложено увеличить число целевых дисков.

Если при создании резервной копии переменной **SHRINK** в файле `image.data` не было присвоено значение `yes`, то файловые системы, создаваемые в целевой системе, будут иметь тот же размер, что и в исходной системе. Исключение составляет каталог `/tmp`, размер которого может быть увеличен до значения, необходимого для правильной работы команды **bosboot**. Информация о настройке переменных приведена в описании файла `image.data`.

После восстановления резервной копии программа установки изменяет конфигурацию ODM в целевой системе. Если конфигурация аппаратного обеспечения целевой системы не совпадает с конфигурацией исходной системы, то программа может изменить атрибуты устройств в следующих файлах целевой системы:

- Всех файлах каталога `/etc/objrepos`, начинающихся с `Cu`
- Всех файлах каталога `/dev`.

Информация, связанная с данной:

Установка резервных копий системы

Реализация запланированных резервных копий:

Ниже описана процедура создания и применения сценария еженедельного полного резервного копирования и ежедневного дополняющего копирования.

- Объем данных для резервного копирования должен умещаться на одной магнитной ленте.
- Перед тем, как команда **cron** запустит сценарий, необходимо вставить в накопитель ленту для резервного копирования.
- Убедитесь в том, что устройство подключено и доступно, в особенности если копирование планируется на ночное время. Для проверки доступности устройства можно воспользоваться командой **lsdev -C | pg**.
- Во избежание ошибок проверьте, не нуждается ли устройство резервного копирования в чистке.
- При резервном копировании файловых систем, которые могут использоваться, эти файловые системы следует сначала размонтировать.
- Проверьте целостность файловой системы перед тем, как создавать ее резервную копию. Воспользуйтесь процедурой “Проверка файловой системы” на стр. 442 или запустите команду **fsck**.

Приведенный здесь сценарий - это не более чем образец, который необходимо изменить в соответствии с конкретными условиями.

Понятия, связанные с данным:

“Стратегия резервного копирования” на стр. 24
Существует два способа резервного копирования больших объемов данных.

Резервное копирование файловых систем с помощью команды cron:

Ниже приведены инструкции по созданию сценария **crontab**, который затем будет передан на выполнение командой **cron**.

Примером может служить сценарий резервного копирования двух пользовательских файловых систем, `/home/plan` и `/home/run`, каждую ночь с понедельника по субботу. Обе файловые системы записываются на одну ленту, и каждое утро в устройство загружается новая лента для использования следующей ночью. Ночью в понедельник создается полный архив (уровня 0). Со вторника по субботу выполняется дополняющее резервное копирование.

1. Создание сценария **crontab** начинается с запуска команды **crontab-e**. Она открывает пустой файл, в который можно добавлять записи, передаваемые каждую ночь команде **cron** (по умолчанию файл открывается в редакторе **vi**). Введите:

   ```
   crontab -e
   ```

2. В данном примере показаны шесть полей команды **crontab**. В поле 1 нужно ввести минуты, в поле 2 - часы (в 24-часовом формате), в поле 3 - день месяца, а в поле 4 - месяц. В полях 3 и 4 показаны звездочки

(*), означающие, что сценарий будет запускаться каждый месяц в день, указанный полем **день/нед**. В поле 5 указывается день недели; кроме того, можно указать диапазон дней, например, 1-6. В поле 6 указывается команда оболочки.

```
мин час день/мес мес/год день/нед       команда оболочки

0    2     *      *        1             backup -0 -uf /dev/rmt0.1 /home/plan
```

При использовании такой командной строки предполагается, что обслуживающий персонал системы будет присутствовать в это время и при необходимости сможет ответить на приглашения. Флаг **-0** (нуль) команды **backup** означает нулевой уровень копирования, то есть полное резервное копирование. Флаг **-u** обновляет запись в файле `/etc/dumpdates`, а флаг **f** указывает имя устройства, в данном случае - накопитель на магнитной ленте с прямым доступом 0.1.

3. Для каждой файловой системы, резервная копия которой должна быть создана в определенный день, введите строку, аналогичную пункту 2. Ниже показан пример полного сценария резервного копирования двух файловых систем каждый день шесть дней в неделю:

```
0 2 * * 1 backup -0 -uf/dev/rmt0.1 /home/plan
0 3 * * 1 backup -0 -uf/dev/rmt0.1 /home/run
0 2 * * 2 backup -1 -uf/dev/rmt0.1 /home/plan
0 3 * * 2 backup -1 -uf/dev/rmt0.1 /home/run
0 2 * * 3 backup -2 -uf/dev/rmt0.1 /home/plan
0 3 * * 3 backup -2 -uf/dev/rmt0.1 /home/run
0 2 * * 4 backup -3 -uf/dev/rmt0.1 /home/plan
0 3 * * 4 backup -3 -uf/dev/rmt0.1 /home/run
0 2 * * 5 backup -4 -uf/dev/rmt0.1 /home/plan
0 3 * * 5 backup -4 -uf/dev/rmt0.1 /home/run
0 2 * * 6 backup -5 -uf/dev/rmt0.1 /home/plan
0 3 * * 6 backup -5 -uf/dev/rmt0.1 /home/run
```

4. Сохраните созданный файл и закройте редактор. После этого операционная система передаст файл `crontab` сценарию **cron**.

Информация, связанная с данной:

Особый файл rmt

Резервное копирование файлов в управляемой DMAPI файловой системе JFS2:

В командах **tar** и **backbyinode** доступны опции, позволяющие выполнять резервное копирование расширенных атрибутов (EA).

В файловой системе DMAPI команда **backbyinode** позволяет сохранить только резидентные данные файловой системы на момент выполнения команды. Команда **backbyinode** проверяет текущее состояние метаданных. Такой подход может быть предпочтителен в случае применения DMAPI, поскольку создается резервная копия состояния управляемой файловой системы. Автономные данные в резервную копию не добавляются.

Для резервного копирования всех данных файловой системы DMAPI воспользуйтесь командой, считывающей файлы целиком, такой как **tar**. В этом случае приложение с поддержкой DMAPI сможет восстановить данные всех файлов, обработанных с помощью команды **tar**, путем перемещения данных между вторичным и третичным носителями. Обратите внимание, что этот процесс может вызвать снижение производительности.

Форматирование дискет (команда format или fdformat):

Команды **format** и `fdformat` позволяют отформатировать дискету, расположенную в дисководе, который указан в параметре *Устройство* (по умолчанию применяется устройство `/dev/rfd0`).

Внимание: Форматирование дискеты приводит к потере всех записанных на ней данных.

Команда **format** распознает тип устройства. Допустимы следующие типы устройств:

- 5,25-дюймовая дискета с низкой плотностью (360 КБ), содержащая 40x2 дорожек, каждая из которых разделена на 9 секторов
- 5,25-дюймовая дискета с высокой плотностью (1,2 МБ), содержащая 80x2 дорожек, каждая из которых разделена на 15 секторов
- 3,5-дюймовая дискета с низкой плотностью (720 КБ), содержащая 80x2 дорожек, каждая из которых разделена на 9 секторов
- 3,5-дюймовая дискета с высокой емкостью (2,88 МБ), содержащая 80x2 дорожек, каждая из которых поделена на 36 секторов

Для дискет всех типов размер сектора равен 512 байтам.

Команда **format** форматирует дискету с высокой плотностью, если в параметре *Устройство* не задана другая плотность.

Команда **fdformat** форматирует дискету с низкой плотностью, если не задан флаг **-h**. Параметр *Устройство* задает дисковод, содержащий дискету для форматирования (например, устройство `/dev/rfd0` для диска 0).

Перед форматированием дискеты с помощью команды **format** или **fdformat** запрашивается подтверждение. Это позволяет вам завершить операцию без ошибок.

Примеры:

- Для того чтобы отформатировать дискету в устройстве `/dev/rfd0`, введите:
  ```
  format -d /dev/rfd0
  ```
- Для форматирования дискеты без проверки наличия поврежденных дорожек введите:
  ```
  format -f
  ```
- Для форматирования дискеты емкостью 360 КБ, загруженной в дисковод для 5,25-дюймовых дискет 1,2 МБ `/dev/rfd1`, введите:
  ```
  format -l -d /dev/rfd1
  ```
- Для принудительного форматирования дискеты с высокой плотностью с помощью команды **fdformat** введите:
  ```
  fdformat -h
  ```

Сведения о синтаксисе приведены в описании команды **format** книги *Справочник по командам, том 2*.

Проверка целостности файловой системы (команда fsck):

Можно проверять согласованность файловых систем и исправлять нарушения целостности в интерактивном режиме с помощью команды **fsck**.

Во время инициализации системы эту команду необходимо вызвать для каждой файловой системы. Файл устройства, в котором находится файловая система (например, устройство `/dev/hd0`), должен быть доступен для чтения. Обычно файловая система согласована, и команда **fsck** просто сообщает о количестве файлов и занятых и свободных блоков в ней. Если целостность файловой системы нарушена, то команда **fsck** выдает информацию об обнаруженных несоответствиях и запрашивает у вас разрешения исправить их. Команда **fsck** прежде всего пытается всеми возможными способами исправить ситуацию и избежать действий, которые могут привести к утере данных. Тем не менее, в некоторых случаях команда **fsck** вынуждена рекомендовать вам уничтожить поврежденный файл.

Внимание: После сбоя системы всегда выполняйте команду **fsck** в файловых системах. Действия по исправлению могут привести к потере некоторых данных. По умолчанию перед выполнением любого действия по восстановлению согласованности ожидается, что пользователь введет ответ `yes` или `no`. Если у вас нет прав на *запись* в поврежденный файл, то команда **fsck** по умолчанию выберет ответ `no`.

Примеры:

- Для проверки всех файловых систем по умолчанию введите:
  ```
  fsck
  ```

  В таком варианте команда **fsck** запросит у вас разрешения перед тем, как вносить любые изменения в файловую систему.
- Для автоматического исправления мелких неполадок в файловых системах по умолчанию введите:
  ```
  fsck -p
  ```
- Для проверки файловой системы /dev/hd1 введите:
  ```
  fsck /dev/hd1
  ```

  Будет проверена размонтированная файловая система, находящаяся на устройстве /dev/hd1.

Примечание: Команда **fsck** не исправляет ошибки в смонтированной файловой системе.

Сведения о синтаксисе приведены в описании команды **fsck** книги *Справочник по командам, том 2*.

Копирование данных на дискеты и с дискет (команда flcopy):

Команда **flcopy** позволяет скопировать содержимое дискеты (открытой как /dev/rfd0) в файл floppy, созданный в текущем каталоге.

При необходимости появится сообщение Замените дискету, нажмите Enter. Команда **flcopy** позволяет также скопировать файл floppy на дискету.

Примеры:
- Для копирования файла /dev/rfd1 в файл floppy текущего каталога введите:
  ```
  flcopy -f /dev/rfd1 -r
  ```
- Для копирования содержимого первых 100 дорожек дискеты введите:
  ```
  flcopy -f /dev/rfd1 -t 100
  ```

Сведения о синтаксисе приведены в описании команды **flcopy** книги *Справочник по командам, том 2*.

Копирование файлов на магнитную ленту или диск (команда cpio -o):

С помощью команды **cpio -o** можно считать пути к файлам из стандартного ввода и скопировать эти файла в стандартный вывод вместе с информацией о каталогах и состоянии.

Длина пути не должна превышать 128 символов. Не рекомендуется указывать в команде **cpio** пути, состоящие из множества уникальным образом связанных файлов, поскольку на отслеживание всех связей может не хватить памяти.

Примеры:
- Для копирования на дискету из текущего каталога всех файлов, имена которых оканчиваются на .c, введите:
  ```
  ls *.c | cpio -ov >/dev/rfd0
  ```

  Флаг **-v** означает, что необходимо показать имена всех файлов.
- Для копирования на дискету текущего каталога и всех его подкаталогов введите:
  ```
  find . -print | cpio -ov >/dev/rfd0
  ```

  Будет сохранено дерево каталогов, начинающееся с текущего каталога (.) и содержащее все подкаталоги и файлы.
- Эту команду можно записать в более коротком формате:

```
find . -cpio /dev/rfd0 -print
```

Запись `-print` означает, что необходимо показать имя каждого копируемого файла.

Сведения о синтаксисе приведены в описании команды **cpio** книги *Справочник по командам, том 1*.

Копирование файлов с магнитной ленты или диска (команда cpio -i):

Команда **cpio -i** считывает из стандартного ввода архивный файл, созданный командой **cpio -o**, и копирует из него файлы с именами, соответствующими параметру *Шаблон*.

Файлы копируются в текущий каталог. Вы можете указать несколько параметров *Шаблон*, воспользовавшись форматом имен файлов, описанным в справке по команде **ksh**. Значение параметра *Шаблон* по умолчанию - звездочка (`*`), т.е. выбор всех файлов текущего каталога. В выражении типа `[a-z]` дефис (`-`) заменяет *все промежуточные значения* согласно текущей последовательности упорядочения.

Примечание: Шаблоны `"*.c"` и `"*.o"` должны быть заключены в кавычки, чтобы оболочка не рассматривала звездочку (`*`) как символ соответствия шаблону. Это особый случай, в котором команда **cpio** сама расшифровывает символы соответствия шаблону.

Примеры:

- Для просмотра списка файлов, сохраненных на дискете командой **cpio**, введите:
  ```
  cpio -itv </dev/rfd0
  ```
  Будет показан список всех данных, ранее сохраненных в файле `/dev/rfd0` в формате команды **cpio**. Список аналогичен полному списку каталогов, выдаваемому командой **ls -l**.
- Для просмотра только каталогов и файлов укажите только флаги **-it**.
- Для копирования с дискеты файлов, ранее сохраненных командой **cpio**, введите:
  ```
  cpio -idmv </dev/rfd0
  ```
  Файлы, ранее сохраненные в файле `/dev/rfd0` командой **cpio**, будут скопированы обратно в файловую систему (укажите флаг **-i**). Флаг **-d** позволяет команде **cpio** создать необходимые каталоги, если сохраняется дерево каталогов. Флаг **-m** определяет время последнего изменения при сохранении файлов. Флаг **-v** означает, что команда **cpio** должна показать имя каждого копируемого файла.
- Для копирования выбранных файлов с дискеты введите:
  ```
  cpio -i "*.c" "*.o" </dev/rfd0
  ```
  С дискеты будут скопированы файлы, имена которых оканчиваются символами `.c` или `.o`.

Сведения о синтаксисе приведены в описании команды **cpio** книги *Справочник по командам, том 1*.

Копирование данных на магнитную ленту и с магнитной ленты (команда tcopy):

Для копирования данных с магнитной ленты служит команда **tcopy**.

Например, для копирования данных с магнитной ленты потокового накопителя на ленту с 9 дорожками введите:

```
tcopy /dev/rmt0 /dev/rmt8
```

Сведения о синтаксисе приведены в описании команды **tcopy** книги *Справочник по командам, том 5*.

Проверка целостности содержимого ленты (команда tapechk):

Можно выполнить частичную проверку целостности содержимого ленты в подключенном накопителе на магнитной ленте с помощью команды **tapechk**.

Некоторые аппаратные сбои накопителя на магнитной ленте можно обнаружить, просто считав содержимое ленты. Команда **tapechk** позволяет выполнять чтение ленты на уровне файлов.

Например, для проверки первых трех файлов на магнитной ленте введите:

```
tapechk 3
```

Сведения о синтаксисе приведены в описании команды **tapechk** книги *Справочник по командам, том 3*.

Архивация файлов (команда tar):

Архивное копирование является способом, при котором один файл, несколько файлов или целая база данных сохраняются для будущего использования, в архивных целях или для восстановления на случай повреждения исходных данных.

При этом скопированные в архив данные удаляются из системы.

С помощью команды **tar** можно записывать файлы на носитель и восстанавливать их с носителя. Команда **tar** ищет архив на устройстве по умолчанию (обычно на магнитной ленте), если не указано другое устройство.

При записи в архив команда **tar** создает временный файл (`/tmp/tar*`) и хранит в памяти таблицу файлов с несколькими ссылками. Если команде **tar** не удастся создать временный файл или выделить достаточно памяти для хранения таблиц ссылок, выдается сообщение об ошибке.

Примеры:

- Для записи файлов `file1` и `file2` в новый архив на накопителе на магнитной ленте по умолчанию введите:

  ```
  tar -c файл1 файл2
  ```

- Для извлечения всех файлов каталога `/tmp` из архивного файла на устройстве `/dev/rmt2` и установки текущего времени в качестве времени последнего изменения введите следующую команду:

  ```
  tar -xm -f/dev/rmt2 /tmp
  ```

- Для просмотра имен файлов из архива `out.tar` в текущем каталоге введите:

  ```
  tar -vtf out.tar
  ```

Дополнительные сведения и описание синтаксиса приведены в описании команды **tar** в книге *Справочник по командам, том 5*.

Резервное копирование файлов

Резервные копии файлов на носителе, например, магнитной ленте или дискете, можно создать с помощью команды **backup** или **smit**.

Внимание: При попытке создать резервную копию смонтированной файловой системы будет выдано предупреждающее сообщение. Выполнение команды **backup** будет продолжено, но созданная резервная копия может содержать неполные варианты файлов. Это не относится к корневой (`/`) файловой системе.

Созданные с помощью команды **backup** или **smit** копии хранятся в одном из следующих форматов резервного копирования:

- Отдельные файлы, скопированные поименно с помощью флага **-i**.
- Вся файловая система, скопированная по номеру i-узла с помощью параметров *-уровень* и *файловая-система*.

  Примечание:

– Если файл изменяется во время резервного копирования, существует вероятность повреждения данных. В связи с этим во время резервного копирования число других операций, выполняемых в системе, должно быть сведено к минимуму.
– Резервную копию, созданную на 8-мм магнитной ленте с нулевым размером блока, нельзя восстановить напрямую. Для восстановления такой резервной копии необходимо выполнить особую процедуру, приведенную в описании команды **restore**.

Внимание: Убедитесь, что указанные флаги соответствуют носителю резервной копии.

Резервное копирование файлов с помощью команды backup:

Команда **backup** предназначена для создания резервных копий файлов.

Например, для создания резервной копии выбранных по именам файлов из каталога **$HOME** введите:

```
find $HOME -print | backup -i -v
```

Флаг **-i** указывает системе, что имена копируемых файлов нужно прочитать из стандартного ввода. Команда **find** создает список файлов из каталога пользователя. Затем этот список передается по конвейеру команде **backup** в качестве стандартного ввода. Флаг **-v** будет выдавать отчет о состоянии после каждого скопированного файла. Резервная копия создается на устройстве резервного копирования по умолчанию в локальной системе.

Примеры:

- Для создания резервной копии корневой файловой системы введите:

  ```
  backup -0 -u /
  ```

  Уровень 0 и символ / указывают на корневую файловую систему /. Файловая система будет скопирована в файл /dev/rfd0. Флаг **-u** означает, что необходимо обновить запись текущего уровня резервной копии в файле /etc/dumpdates.
- Для сохранения всех файлов из файловой системы / (root), измененных с момента создания последней резервной копии уровня 0, введите:

  ```
  backup -1 -u /
  ```

Сведения о синтаксисе приведены в описании команды **резервное копирование** книги *Справочник по командам, том 4*.

Резервное копирование файлов с помощью команды smit:

Команда **smit** вызывает команду **backup**, которая создает резервные копии файлов на указанном носителе.

1. Введите в командной строке:

   ```
   smit backup
   ```
2. В поле **Полное имя каталога** введите путь к каталогу, в котором обычно монтируется файловая система:

   ```
   /home/bill
   ```
3. В поле **Устройство** или **Файл резервной копии** введите имя устройства вывода, например, накопителя на магнитной ленте:

   ```
   /dev/rmt0
   ```
4. С помощью клавиши Tab измените значение поля **Отчет о каждом этапе**, если вы хотите, чтобы сообщения об ошибках выдавались на экран.
5. В среде управления системами оставьте значение по умолчанию в поле **Максимальное число блоков для записи на носитель резервной копии**, так как это поле не применяется в случае копирования на магнитную ленту.
6. Для сохранения указанного каталога или файловой системы нажмите Enter.

7. Вызовите команду **restore -t**. Если будет выдано сообщение об ошибке, то вам придется повторить весь процесс создания резервной копии.

Завершение работы системы

Команда **shutdown** - это наиболее безопасный способ завершить работу системы.

Вам может потребоваться завершить работу системы:

- После установки нового или изменения конфигурации существующего программного обеспечения
- При обнаружении аппаратных неполадок
- При зависании системы
- При снижении производительности системы
- При возможном повреждении файловой системы.

Флаги этой команды позволяют уведомить пользователей, что система завершает работу, завершить все активные процессы, размонтировать файловые системы и завершить работу. Дополнительная информация приведена в описании команды **shutdown**.

Сведения об отдельных ситуациях завершения работы системы приведены в следующем документе:

Завершение работы системы без перезагрузки

Есть два способа завершить работу системы без перезагрузки.

Существует два средства, позволяющих завершить работу системы без перезагрузки - команда SMIT и команда **shutdown**.

Предварительные требования

Для завершения работы системы необходимы права доступа root.

Для завершения работы системы с помощью SMIT:

1. Войдите в систему как пользователь root.
2. В командной строке введите:
   ```
   smit shutdown
   ```

Для завершения работы системы с помощью команды **shutdown**:

1. Войдите в систему как пользователь root.
2. В командной строке введите:
   ```
   завершение работы
   ```

Закрытие системы и переход в однопользовательский режим

Иногда возникает необходимость закрыть систему и перейти в однопользовательский режим для обслуживания и диагностики программного обеспечения.

1. Для перехода в корневой каталог введите команду `cd /`. Для того чтобы при отключении системы и перехода к однопользовательскому режиму файловая система была размонтирована правильно, вы должны находиться в корневом каталоге.
2. Введите `shutdown -m`. Система перейдет в однопользовательский режим.

Появится системное приглашение и вы сможете перейти к процедурам обслуживания.

Завершение работы системы в чрезвычайно ситуации

Команда **shutdown** предназначена для быстрого завершения работы системы без уведомления пользователей.

При чрезвычайных ситуациях можно завершить работу системы с помощью команды **shutdown**.

Введите `shutdown -F`. Флаг **-F** указывается в команде **shutdown** для того, чтобы система максимально быстро завершила работу без предварительного уведомления пользователей.

Системная среда

Системная среда - это набор переменных, значения которых задают определенные характеристики работы процессов.

Эти переменные устанавливаются или сбрасываются при каждом запуске оболочки. С точки зрения управления системой важно, чтобы после входа пользователя в систему для него были установлены правильные значения переменных среды. Значения большей части переменных задаются при инициализации системы. Определения переменных считываются из файла `/etc/profile`, либо им присваиваются значения по умолчанию.

Профили

При входе в систему оболочка применяет файлы profile двух типов.

Сначала выполняются команды файлов, затем настраивается среда оболочки. Функции этих файлов большей частью совпадают, но файл `/etc/profile` управляет переменными среды всех пользователей системы, а файл `.profile` задает среду конкретного пользователя.

Информация о параметрах профайла и системной среды приведена в следующих разделах:

- Файл `/etc/profile`
- Файл `.profile`
- Настройка переменной системной среды
- Изменение ежедневного приветствия
- “Функции управления временем” на стр. 53.

/etc/profile, файл

Первый файл, который просматривается операционной системой при входе в систему - `/etc/profile`. Это файл управляет системными переменными по умолчанию, такими как:

- Переменные экспорта
- Маску создания файла (umask)
- Типы терминала
- Сообщение о поступлении новой почты

Системный администратор настраивает файл `profile` для всех пользователей системы. Только он может изменять этот файл.

Файл .profile

Второй файл, который просматривается при входе в систему - `.profile`. Файл `.profile` из домашнего каталога пользователя (`$HOME`) позволяет настроить собственную среду. Файл `.profile` переопределяет команды и переменные, заданные в файле `/etc/profile`. Поскольку файл `.profile` скрыт, для его просмотра воспользуйтесь командой **ls -a**. Например, с помощью файла `.profile` можно управлять следующими значениями по умолчанию:

- Открываемые оболочки
- Внешний вид приглашения
- Переменные среды (например, путь поиска)
- Звуковой сигнал клавиатуры

Ниже приведен пример файла `.profile`:

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user:/home/gsc/bin::
epath=/home/gsc/e3:
export PATH epath
csh
```

В этом примере определены и экспортированы два пути (PATH и epath), и была запущена оболочка C (csh).

В файле .profile (или, если его нет, в файле .profile) можно задать и переменные начальной оболочки. Вы можете также настраивать среды других оболочек. Например, с помощью файлов .chsrc и .kshrc вы можете настроить отдельно оболочки C и Korn.

Функции управления временем

Функции управления временем используются для чтения и настройки текущей системной даты и времени.

Для вызова функций, работающих со временем, для компилятора не требуется указывать специальный флаг. Для вызова этих функций включите в программу файл описаний. Для включения файла воспользуйтесь оператором:

```
#include <time.h>
```

Существуют следующие службы времени:

Элемент	Описание
adjtime	Уточняет время, синхронизируя его с системными часами.
ctime, **localtime**, **gmtime**, **mktime**, **difftime**, **asctime**, **tzset**	Преобразует дату и время в строку символов.
getinterval, **incinterval**, **absinterval**, **resinc**, **resabs**, **alarm**, **ualarm**, **getitimer**, **setitimer**	Обрабатывает моменты истечения времени для таймеров.
gettimer, **settimer**, **restimer**, **stime**, **time**	Считывает или устанавливает текущие значения для конкретного таймера на системном уровне.
gettimerid	Создает таймер для процесса.
gettimeofday, **settimeofday**, **ftime**	Считывает и устанавливает дату и время.
nsleep, **usleep**, **sleep**	Приостанавливает выполнение текущего процесса.
reltimerid	Удаляет таймер, который был до этого создан какого-либо процесса.

Наборы файлов и аппаратное обеспечение, необходимые для 64-разрядного режима

64-разрядный режим обеспечивает быстрый доступ к большим объемам данных, а также эффективную обработку 64-разрядных типов данных.

64-разрядная среда выполнения базовой операционной системы поставляется в наборе файлов bos.64bit. При установке bos.64bit дополнительно устанавливается файл /etc/methods/cfg64. Файл /etc/methods/cfg64 - это команда, включающая 64-значную среду выполнения. Эта команда вызывается сценарием rc.boot на третьем этапе процесса загрузки.

Начиная с AIX 6.1, 32-битовое ядро помечается как устаревшее. При установке AIX 6.1 с помощью базовой операционной системы включается 64-битовый режим.

Примечание: Для запуска AIX 6.1 необходимо 64-битовое аппаратное обеспечение. Для моделей RS/6000 используются процессоры 604e, которые не являются 64-битовыми:

- 7025 F50 Series
- 7026 H50 Series
- 9076 H50 Series
- 7043 150 Series
- 7046 B50 Series

Для проверки возможностей процессора выполните следующую команду:

```
/usr/sbin/prtconf -c
```

В выводе команды **prtconf** указывается значение 32 или 64 в зависимости от возможностей процессора. Если в системе отсутствует команда **prtconf** , то можно использовать команду **bootinfo** с флагом **-y**.

Аппаратное обеспечение, необходимое для 64-разрядного режима

64-разрядные приложения можно выполнять только на 64-разрядном аппаратном обеспечении.

Для того чтобы определить тип архитектуры системы, выполните следующие действия:

1. Войдите в систему как пользователь root.
2. Введите в командной строке `bootinfo -y`.

В выводе этой команды указывается значение **32** или **64** в зависимости от типа архитектуры. Кроме того, во всех версиях AIX команда `lsattr -El proc0` позволяет просмотреть тип процессора сервера.

Сравнение 32-разрядной и 64-разрядной производительности

Как правило, 64-разрядное аппаратное обеспечение позволяет выполнять 32-разрядные приложения, поскольку в нем предусмотрена поддержка как 32-разрядного, так и 64-разрядного программного обеспечения. Однако, 64-разрядное программное обеспечение нельзя запускать на 32-разрядном аппаратном обеспечении.

Особенности приложений, связанные с производительностью, а также рекомендации по выбору среды для выполнения можно найти в руководствах пользователей этих приложений.

динамическое отключение процессоров

AIX может обнаруживать и автоматически останавливать вышедший из строя процессор.

В системах типа 7044 модели 270, а также в более новых системах с двумя и более процессорами аппаратное обеспечение обнаруживает устранимые ошибки, сведения о которых собираются встроенным программным обеспечением. Если число таких ошибок невелико, то они не представляют серьезной опасности и их можно игнорировать. Однако если число ошибок какого-либо процессора постоянно возрастает, то это указывает на возможность скорого отказа этого процессора. Такой прогноз делается встроенным программным обеспечением на основании анализа частоты возникновения неполадок и заданных пороговых значений.

В компьютерах указанного типа AIX предоставляет функцию контроля аппаратного обеспечения и постоянно запрашивает у встроенного программного обеспечения сведения об аппаратных ошибках. Когда число ошибок процессора достигает порогового значения, встроенное программное обеспечение делает вывод о возможном сбое аппаратного компонента и возвращает отчет об ошибке. Все ошибки заносятся в системный протокол ошибок. В многопроцессорных системах при возникновении ошибок определенного типа AIX может отказаться от использования ненадежного процессора. Эта функция называется *динамическое отключение процессоров*.

При этом процессор помечается встроенным программным обеспечением как отключенный для всех последующих запусков системы вплоть до замены процессора.

Влияние отключения процессора на приложения:

Описанное отключение процессора незаметно для большинства приложений, включая драйверы и расширения ядра. Тем не менее, с помощью стандартных интерфейсов можно определить, работает ли приложение или расширение ядра в многопроцессорной системе, узнать число процессоров и связать нити с определенными процессорами.

Интерфейс bindprocessor связывания процессов и нитей с процессорами использует номера связывания CPU. Номера связывания CPU находятся в диапазоне от 0 до *N*-1, где *N* - общее число процессоров. Во избежание возникновения неполадок в приложениях и расширениях ядра, не предполагающих наличия пропусков в нумерации CPU, при отключении процессора операционная система AIX динамически изменяет нумерацию остальных процессоров. При этом с точки зрения приложений всегда отключается процессор с максимальным номером. Например, в 8-процессорной системе SMP применяются номера CPU [0..7]. В

случае сбоя одного из процессоров в системе останется 7 доступных CPU с номерами от 0 до 6. С точки зрения приложений это выглядит как отключение CPU с номером 7, независимо от того, какой именно физический процессор оказался неисправным.

Примечание: В дальнейшем словом *CPU* будет обозначаться логический ресурс, а словом *процессор* - физический ресурс.

Нормальная работа приложений или расширений ядра может быть нарушена, если при динамическом отключении процессора AIX без предупреждения прервет работу каких-либо нитей, явно связанных с определенными CPU, или принудительно перенесет эти нити на другой CPU. В связи с этим функция динамическое отключение процессоров поддерживает программный интерфейс, позволяющий уведомлять приложения и расширения ядра об отключении процессоров. При получении такого уведомления приложения и расширения ядра должны самостоятельно отключить нити и другие связанные ресурсы от CPU с максимальным номером и адаптироваться к новой конфигурации CPU.

Если после получения уведомления некоторые нити остаются связанными с последним ИД CPU, то процесс прерывается, в протокол заносится сообщение об ошибке, и AIX продолжает использовать этот процессор. После окончательного выхода процессора из строя происходит общий сбой системы. Таким образом, при разработке приложений и расширений ядра важно предусмотреть процедуру обработки поступающих уведомлений об отключении процессора, включающую в себя перераспределение нитей и других ресурсов.

В тех редких случаях, когда динамическое отключение процессоров не выполняется, системному администратору отправляется уведомление о возможном сбое. Ознакомившись с записью, занесенной в протокол ошибок, системный администратор может запланировать обслуживание системы и заменить ненадежный компонент до того, как система полностью выйдет из строя.

Процесс отключения процессора:

В AIX можно остановить вышедший из строя процессор, отключив его.

Обычно процедура отключения процессора выглядит следующим образом:

1. Встроенное программное обеспечение обнаруживает, что один из процессоров превысил допустимое число устранимых ошибок.
2. Встроенное программное обеспечение записывает отчет об ошибке в системный протокол ошибок и, при работе в системе с поддержкой отключения процессоров, AIX начинает процедуру отключения.
3. AIX отправляет уведомление пользовательским процессам и нитям, связанным с CPU с максимальным логическим номером.
4. AIX дожидается того момента, когда все нити, связанные с этим CPU, будут перенесены на другие процессоры. Если какая-либо нить остается связанной с процессором, то процедура отключения процессора прерывается.
5. Если с процессором больше не связаны никакие нити или процессы, вызывается зарегистрированный ранее обработчик HAEH. Если обработчик HAEH возвращает код ошибки, то процедура отключения прерывается.
6. В противном случае работа неисправного процессора прекращается.

При возникновении ошибки в ходе отключения процессора в протокол заносится сообщение об ошибке. Системный администратор может просмотреть протокол ошибок, выполнить необходимые действия по устранению ошибки и повторить попытку отключения. Например, если отключение было отменено из-за того, что какое-либо приложение не отключило свои нити от CPU, то системный администратор может завершить работу этого приложения, заново инициировать отключение процессора, а затем снова запустить приложение.

Включение динамическое отключение процессоров:

Если ваша система поддерживает динамическое отключение процессоров, то с помощью SMIT или системных команд вы можете **включить** или **выключить** эту функцию.

Функция динамическое отключение процессоров включается по умолчанию во время установки (при условии, что аппаратное и встроенное программное обеспечение системы поддерживают эту функцию).

Процедура быстрого доступа SMIT

1. Войдя в систему под именем root, введите следующую команду `smit system` и нажмите Enter.
2. В окне **Системная среда** выберите **Изменить / Просмотреть характеристики операционной системы**.
3. Выполните задачи с помощью программы SMIT.

Дополнительную информацию по выполнению этих задач можно получить, нажав клавишу справки F1 в окне диалога SMIT.

Процедура команд

Обладая правами доступа root, вы можете применять следующие команды функции динамическое отключение процессоров:

- Команда **chdev** позволяет изменить характеристики указанного устройства. Дополнительная информация приведена в описании команды **chdev** в книге *Справочник по командам, том 1*.
- При сбое функции освобождения процессоров команда **ha_star** позволяет перезапустить эту функцию после устранения неполадок. Дополнительная информация приведена в описании команды **ha_star** в книге *Справочник по командам, том 2*.
- Команда **errpt** позволяет создать отчет о занесенных в протокол ошибках. Дополнительная информация приведена в описании команды **errpt** в книге *Справочник по командам, том 2*.

Способы включения и выключения динамического отключения процессора:

Функция динамическое отключение процессоров может быть включена или выключена путем изменения значения атрибута **cpuguard** объекта ODM `sys0`.

Этот атрибут может принимать значения `enable` и `disable`.

Значение по умолчанию - `enabled` (атрибуту **cpuguard** присвоено значение `enable`). Если системный администратор планирует отключить эту функцию, то он может воспользоваться системными меню, меню SMIT **Системные среды** или командой **chdev**. (В предыдущих версиях AIX по умолчанию применялось значение `disabled`.)

Примечание: Даже если функция отключения процессора выключена (значение "disabled"), сообщения об ошибках заносятся в протокол. Протокол ошибок будет содержать сообщение типа `CPU_FAILURE_PREDICTED`, указывающее, что AIX был уведомлен о сбое CPU.

Перезапуск прерванной процедуры отключения процесса:

Иногда процедура отключения процессора прерывается из-за того, что приложение не перенесло на другой процессор свои нити, связанные с последним CPU.

После устранения неполадки путем отмены связывания нитей (если это можно сделать) или завершения приложения системный администратор может повторить попытку отключения процессора с помощью команды **ha_star**.

Эта команда вызывается в следующем формате:

```
ha_star -C
```

где **-C** обозначает прогнозируемое событие сбоя CPU.

Замечания о состоянии процесса:

Есть несколько связанных с состояниями процессов моментов, о которых следует знать.

Физическим процессорам в базе данных ODM соответствуют объекты с именами **proc***n*, где *n* - это физический номер процессора (n - десятичное число). Как и у любых других устройств в базе данных ODM, у объектов процессоров есть атрибуты и состояние (определено/доступно).

Объект **proc** находится в состоянии Доступен всегда, когда связанный с ним процессор присутствует в системе, независимо от того, исправен ли этот процессор. Информация о том, используется ли процессор, и если нет, то почему, хранится в атрибуте **state** объекта **proc**. Этот атрибут может принимать следующие значения:

Элемент	Описание
enable	Процессор применяется системой.
disable	Процессор динамически отключен.
faulty	Встроенное программное обеспечение пометило процессор как неисправный при загрузке системы.

Если процессор был успешно отключен, то его состояние изменяется с **enable** на **disable**. Независимо от AIX, процессор помечается в микрокоде как неработающий. После следующей загрузки системы он станет недоступным, а его состояние изменится на **faulty**. При этом объект ODM **proc** по-прежнему будет помечен как доступный. Для изменения состояния объекта **proc** на Определен нужно физически удалить процессор или карту процессора из системы.

В следующем примере процессор **proc4** работает правильно и используется операционной системой:

```
# lsattr -EH -l proc4
атрибут   значение описание    user_settable

state  enable  Сост. процессора  Нет
type  PowerPC_RS64-III Тип процессора  Нет
 #
```

Когда для процессора **proc4** поступает сообщение о возможном сбое, этот процессор отключается операционной системой:

```
# lsattr -EH -l proc4
атрибут   значение описание    user_settable

state  disable   Сост. процессора  Нет
type  PowerPC_RS64-III Тип процессора  Нет
 #
```

При следующем запуске системы процессор **proc4** помечается встроенным программным обеспечением как неисправный:

```
# lsattr -EH -l proc4
атрибут   значение описание    user_settable

state  faulty   Сост. процессора Нет
type  PowerPC_RS64-III Тип процессора  Нет
 #
```

Однако состояние процессора **proc4** по-прежнему будет Доступен:

```
# lsdev -CH -l proc4
имя   состояние расположк. описание

proc4  Доступен   00-04   Процессор
#
```

Записи протокола ошибок отключения CPU:

С отключением CPU связаны три сообщения протокола ошибок.

Ниже приведены примеры.

краткие записи errpt - обзор

Ниже приведен пример записей, выводимых командой **errpt** без параметров:

```
# errpt
ИДЕНТИФИКАТОР      ВРЕМЯ             T      C      ИМЯ РЕСУРСА          ОПИСАНИЕ
804E987A           1008161399        I      O      proc4                CPU ОТКЛЮЧЕН
8470267F           1008161299        T      S      proc4
ОТКЛЮЧЕНИЕ CPU ПРЕРВАНО
1B963892           1008160299        P      H      proc4                ВОЗМОЖЕН СБОЙ CPU
#
```

- Если в системе применяется функция отключения процессора, то за сообщением ВОЗМОЖЕН СБОЙ CPU всегда следует сообщение CPU ОТКЛЮЧЕН или ОТКЛЮЧЕНИЕ CPU ПРЕРВАНО.
- Если функция отключения процессора не применяется, то в протокол заносится только сообщение ВОЗМОЖЕН СБОЙ CPU. Если функция отключения процессора будет включена после занесения в протокол одного или нескольких сообщений ВОЗМОЖЕН СБОЙ CPU, то будет начата процедура отключения; в результате в протокол для каждого неисправного процессора будет занесено сообщение об успешном или прерванном отключении.

полные записи errpt - подробное описание

Ниже приведен пример вывода команды **errpt -a**:

- CPU_FAIL_PREDICTED

  Описание ошибки: Возможен сбой процессора

  Данная ошибка означает, что аппаратное обеспечение обнаружило высокую вероятность сбоя процессора в ближайшее время. Это сообщение заносится в протокол всегда, независимо от того, применяется функция отключения процессора или нет.

  ПОДРОБНАЯ ИНФОРМАЦИЯ: *Номер* и *расположение* физического процессора

  Пример: запись протокола ошибок - полный формат

```
METKA:CPU_FAIL_PREDICTED
ИДЕНТИФИКАТОР:  1655419A

Дата/Время: Чтв, 30 Сент, 13:42:11
Порядковый номер:  53
ИД системы:  00002F0E4C00
ИД узла:  auntbea
Класс:  H
Тип:  PEND
Имя ресурса:  proc25
Класс ресурса:  процессор
Тип ресурса:  proc_rspc
Расположение:  00-25

Описание
ВОЗМОЖЕН СБОЙ CPU

Возможные причины
СБОЙ CPU

Причины сбоя
СБОЙ CPU
```

```
 Рекомендуемые действия
 ВКЛЮЧИТЕ РЕЖИМ ЗАЩИТЫ CPU
 ВЫПОЛНИТЕ ПРОЦЕДУРЫ ДИАГНОСТИКИ

Подробные сведения
PROBLEM DATA
0144 1000 0000 003A 8E00 9100 1842 1100 1999 0930 4019
0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 4942 4D00 5531
2E31 2D50 312D 4332 0000
0002 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000
... ... ... ... ...
```

- CPU_DEALLOC_SUCCESS

  Описание ошибки: При получении сообщения о прогнозируемом сбое процессор был успешно отключен. Это сообщение заносится в протокол в том случае, если в системе применяется функция отключения CPU, и процессор был успешно отключен.

  ПОДРОБНАЯ ИНФОРМАЦИЯ: *Логический номер CPU отключенного процессора.*

  Пример: запись протокола ошибок - полный формат:

```
МЕТКА:   CPU_DEALLOC_SUCCESS
ИДЕНТИФИКАТОР:  804E987A

Дата/Время: Чтв, 30 Сент, 13:44:13
Порядковый номер:  63
ИД системы:  00002F0E4C00
ИД узла:  auntbea
Класс:  O
Тип:             INFO
Имя ресурса:  proc24

Описание
CPU ОСВОБОЖДЕН


 Рекомендуемые действия
 ПРОИЗОШЕЛ СБОЙ CPU - ВЫПОЛНИТЕ ОБСЛУЖИВАНИЕ СИСТЕМЫ

Подробные сведения
ЛОГИЧЕСКИЙ НОМЕР ОТКЛЮЧЕННОГО CPU

 0
```

  В этом примере указано, что процессор **proc24** был успешно отключен, и в момент отключения был связан с CPU **0**.

- CPU_DEALLOC_FAIL

  Описание ошибки: Отключение процессора, связанное с сообщением о прогнозируемом сбое, не было выполнено. Это сообщение заносится в протокол в том случае, если в системе применяется функция отключения процессора, и CPU не был отключен.

  ПОДРОБНАЯ ИНФОРМАЦИЯ: *Код причины, логический номер CPU, дополнительная информация, зависящая от типа сбоя.*

  Код причины представляет собой шестнадцатеричное значение. Возможны следующие значения:

Элемент	Описание
2	Некоторые процессы или нити связаны с последним CPU. В этом случае в разделе с подробной информацией указываются PID таких процессов.
3	При получении уведомления зарегистрированный драйвер или расширение ядра отправило сообщение об ошибке. В этом случае в разделе с подробной информацией указывается имя драйвера или расширения ядра (в текстовом формате).
4	После отключения процессора в системе осталось менее двух доступных CPU. Данная операционная система не отключает более *N*-2 процессоров в *N*-процессорных системах; это позволяет обеспечить правильную работу приложений и расширений ядра, которые используют общее число процессоров для определения типа системы, в которой они работают (однопроцессорная или многопроцессорная система SMP).
200 (0xC8)	Процессор запрещено отключать (атрибуту ODM **cpuguard** присвоено значение `disable`). Эта ошибка может возникнуть только в том случае, если команда **ha_star** была запущена вручную.

Примеры: записи протокола ошибок - полный формат

Пример 1:

```
 МЕТКА: CPU_DEALLOC_ABORTED
 ИДЕНТИФИКАТОР:  8470267F
 Дата/Время: Чтв, 30 Сент, 13:41:10
 Порядковый номер:  50
 ИД системы:  00002F0E4C00
 ИД узла:  auntbea
 Класс:  S
 Тип:  TEMP
 Имя ресурса:  proc26

Описание
ОТКЛЮЧЕНИЕ CPU ПРЕРВАНО

Возможные причины
ПРИКЛАДНАЯ ПРОГРАММА

Причины сбоя
ПРИКЛАДНАЯ ПРОГРАММА

 Рекомендуемые действия
 ПРОИЗОШЕЛ СБОЙ CPU - ВЫПОЛНИТЕ ОБСЛУЖИВАНИЕ СИСТЕМЫ
 ОЗНАКОМЬТЕСЬ С ПОЛЬЗОВАТЕЛЬСКОЙ ДОКУМЕНТАЦИЕЙ ПО CPU

Подробные сведения
ПРИЧИНА ОТМЕНЫ ОТКЛЮЧЕНИЯ
0000 0003
ДАННЫЕ О ПРЕРВАННОМ ОТКЛЮЧЕНИИ
6676 6861 6568 3200
```

В этом примере указано, что процессор **proc26** не был отключен. Код причины 3 означает, что расширение ядра вернуло сообщение об ошибке процедуре уведомления ядра. В поле `ДАННЫЕ О ПРЕРВАННОМ ОТКЛЮЧЕНИИ` указано значение **fvhaeh2**, представляющее собой имя зарегистрированного расширения ядра.

Пример 2:

```
 МЕТКА: CPU_DEALLOC_ABORTED
 ИДЕНТИФИКАТОР:  8470267F
 Дата/Время: Чтв, 30 Сент, 14:00:22
 Порядковый номер:  71
 ИД системы:  00002F0E4C00
 ИД узла:  auntbea
 Класс:  S
 Тип:  TEMP
 Имя ресурса:  proc19

Описание
ОТКЛЮЧЕНИЕ CPU ПРЕРВАНО

Возможные причины
```

```
ПРИКЛАДНАЯ ПРОГРАММА

Причины сбоя
ПРИКЛАДНАЯ ПРОГРАММА

 Рекомендуемые действия
 ПРОИЗОШЕЛ СБОЙ CPU - НЕОБХОДИМО ВЫПОЛНИТЬ ОБСЛУЖИВАНИЕ СИСТЕМЫ
 ОЗНАКОМЬТЕСЬ С ПОЛЬЗОВАТЕЛЬСКОЙ ДОКУМЕНТАЦИЕЙ ПО CPU

Подробные сведения
ПРИЧИНА ОТМЕНЫ ОТКЛЮЧЕНИЯ
0000 0002
ДАННЫЕ О ПРЕРВАННОМ ОТКЛЮЧЕНИИ
0000 0000 0000 4F4A
```

В этом примере указано, что процессор **proc19** не был отключен. Код причины 2 означает, что с последним логическим CPU были связаны нити, которые не были отключены от этого CPU при получении сигнала SIGCPUFAIL. В поле ДАННЫЕ О ПРЕРВАННОМ ОТКЛЮЧЕНИИ указано, что эти нити принадлежат процессу **0x4F4A**.

Опции команды **ps** (-o THREAD, -o BND) позволяют просмотреть список нитей процесса, связанных с CPU с указанным номером.

Пример 3:

```
 МЕТКА: CPU_DEALLOC_ABORTED
 ИДЕНТИФИКАТОР:  8470267F

 Дата/Время: Чтв, 30 Сент, 14:37:34
 Порядковый номер:  106
 ИД системы:  00002F0E4C00
 ИД узла:  auntbea
 Класс:  S
 Тип:  TEMP
 Имя ресурса:  proc2

Описание
ОТКЛЮЧЕНИЕ CPU ПРЕРВАНО

Возможные причины
ПРИКЛАДНАЯ ПРОГРАММА

Причины сбоя
ПРИКЛАДНАЯ ПРОГРАММА

 Рекомендуемые действия
 ПРОИЗОШЕЛ СБОЙ CPU - ВЫПОЛНИТЕ ОБСЛУЖИВАНИЕ СИСТЕМЫ
 ОЗНАКОМЬТЕСЬ С ПОЛЬЗОВАТЕЛЬСКОЙ ДОКУМЕНТАЦИЕЙ ПО CPU

Подробные сведения
ПРИЧИНА ОТМЕНЫ ОТКЛЮЧЕНИЯ
0000 0004
ДАННЫЕ О ПРЕРВАННОМ ОТКЛЮЧЕНИИ
0000 0000 0000 0000
```

В этом примере указано, что процессор **proc2** не был отключен, поскольку в момент сбоя в системе оставалось менее двух активных процессоров (код причины 4).

Настройка переменной системной среды

Системная среда - это набор переменных, значения которых задают определенные характеристики работы процессов.

Эти переменные устанавливаются или сбрасываются при каждом запуске оболочки. С точки зрения управления системой важно, чтобы после входа пользователя в систему для него были установлены

правильные значения переменных среды. Значения большей части переменных задаются при инициализации системы. Определения переменных считываются из файла `/etc/profile`, либо им присваиваются значения по умолчанию.

Проверка системной батареи:

Отставание и неверные показания системных часов могут быть вызваны разрядкой или отключением батареи.

1. Для проверки состояния батареи введите следующую команду **diag**.
   ```
   diag -B -c
   ```
2. В появившемся меню Диагностика выберите вариант **Определение неполадок**. Если батарея разрядилась или отсутствует контакт, будет показано меню неполадки с номером служебного запроса (SRN). Запишите SRN в четвертом пункте Формы отчета о неполадке и сообщите о неполадке в сервисный центр.

Если батарея системы исправна, неправильные показания часов могли быть установлены в результате неправильного запуска команд **date** или **setclock**, а также ошибками, возникшими во время обработки этих команд.

Понятия, связанные с данным:

"Настройка системных часов"
Системные часы записывают время событий в системе, позволяют планировать события в системе (например, запуск диагностики аппаратного обеспечения в 03:00)и позволяют определить время создания или последнего изменения файлов.

Настройка системных часов:

Системные часы записывают время событий в системе, позволяют планировать события в системе (например, запуск диагностики аппаратного обеспечения в 03:00)и позволяют определить время создания или последнего изменения файлов.

Команда **date** позволяет настроить системные часы. Команда **setclock** обращается к серверу времени и устанавливает время и дату.

Задачи, связанные с данной:

"Проверка системной батареи"
Отставание и неверные показания системных часов могут быть вызваны разрядкой или отключением батареи.

Команда date:

Команда **date** позволяет просмотреть или задать дату и время.

Для определения системной даты и времени введите следующую команду:
```
/usr/bin/date
```

Внимание: Не изменяйте дату, если в системе работает несколько пользователей.

С помощью параметра *Дата* дата может быть задана в следующих форматах:

- *ммддЧЧММ*[.*ГГгг*] (по умолчанию)
- *ммддЧЧММ*[.*гг*]

Переменные параметра *Дата* определяются следующим образом:

Элемент	Описание
мм	Задает номер месяца.
дд	Задает день месяца.
ЧЧ	Задает часы (в 24-часовом формате).
ММ	Задает минуты.
ГГ	Задает первые две из четырех цифр года.
гг	Задает последние две цифры года.

Войдя в систему под именем пользователя с правами доступа root, для задания текущей даты и времени введите команду **date**. Например:

```
date 021714252002
```

Эта команда установит дату 17 февраля, 2002, и время 14:25. Дополнительные сведения о команде **date** приведены в ее описании в *Справочник по командам, том 2*.

Команда setclock:

Команда **setclock** позволяет просмотреть или задать значения даты и времени, обратившись к серверу времени.

Для просмотра системной даты и времени введите следующую команду:

```
/usr/sbin/setclock
```

Команда **setclock** получает первый ответ от сервера времени, преобразует содержащийся в нем показатель часов-календаря и показывает локальные дату и время. Если ни один сервер времени не отвечает или сеть не работает, команда **setclock** выводит соответствующее сообщение и оставляет прежние значения даты и времени.

Примечание: Выполнять роль сервера времени может любой хост, на котором выполняется демон **inetd**.

Войдя в систему под именем пользователя с правами доступа root, вы можете применять команду **setclock** для отправки серверу времени запроса службы TIME и настройки местного времени и даты в соответствии с полученным ответом. Например:

```
setclock
сервер-времени
```

где *сервер-времени* - это имя или IP-адрес хоста сервера времени.

Информация, связанная с данной:

setclock, команда

Настройка и поддержка часовых поясов Олсона:

Начиная с AIX 6.1, предоставляется поддержка значений часовых поясов согласно базе данных Олсона.

Спецификация часового пояса POSIX, поддерживаемая в предыдущих версиях AIX, неправильно обрабатывает изменения в правилах для часового пояса, например, в летнее время. В базе данных Олсона сохраняется хронологическая запись правил часового пояса, так что если правила изменяются для конкретного адреса, то AIX корректно интерпретирует показатели даты и времени как для настоящего момента времени, так и для прошедшего.

Согласно спецификации POSIX описания часового пояса до сих пор поддерживаются и распознаются AIX. AIX проверяет переменную среды **TZ** для выявления ее соответствия значению часового пояса Олсона. Если переменная среды **TZ** не соответствует значению часового пояса Олсона, тогда AIX следует правилам спецификации POSIX.

Дополнительные сведения о переменной среды TZ приведены в разделе Файл среды.

Для использования часовых поясов, определенных Олсоном, воспользуйтесь следующим путем SMIT: **Системные среды > Изменить/ показать дату, время и часовой пояс > Задать часовой пояс, указанный в системе**.

Настройка ежедневного приветствия:

Ежедневное приветствие появляется на экране при каждом входе пользователя в систему.

Это удобный способ передавать пользователям различную информацию, например номера версий установленного программного обеспечения или текущие сведения о системе. Для изменения приветствия достаточно просто отредактировать файл `/etc/motd`

AIX Runtime Expert

В состав AIX Runtime Expert входит упрощенный набор действий, которые можно использовать для сбора, применения и проверки среды выполнения одного или нескольких экземпляров AIX.

Инструменты, предлагаемые компонентами AIX, такими как RAS, Защита или Ядро, позволяю изменить параметры каждого уровня компонентов для настройки операционной системы с учетом конкретных требований. AIX Runtime Expert поддерживает настройку уровня системы за счет применения расширяемой среды, отвечающей за обработку различных методов настройки, предусмотренных в AIX.

AIX Runtime Expert позволяет выполнять команды настройки нескольких компонентов в виде одного действия с помощью профайла конфигурации. С помощью профайла конфигурации можно применить один и тот же набор параметров системы в нескольких системах. AIX Runtime Expert предлагает упрощенный способ управления конфигурацией рабочей среды одной или нескольких систем, однако не запрещает применение других способов изменения параметров системы.

Концепции AIX Runtime Expert

Перед тем, как приступить к работе с продуктом AIX Runtime Expert, необходимо получить общее представление о нем.

AIX Runtime Expert предназначен для управления профайлами конфигурации в отдельной системе AIX. Для применения одного и того же профайла системы AIX могут загружать описание профайла с сервера LDAP во время запуска или под управлением административных операций в конечных точках AIX. Для удаленного управления AIX Runtime Expert можно использовать только компонент Network Install Manager (NIM). NIM обеспечивает удаленную работу с AIX Runtime Expert в нескольких автономных клиентах NIM из основной системы NIM.

Профайлы AIX Runtime Expert:

Профайлы AIX Runtime Expert применяются для настройки параметров активной системы, извлечения конфигурации активной системы, а также для сравнения значений с активной системой или другим профайлом.

Профайл описывает один или несколько элементов управления конфигурации, а также связанные параметры соответствующей функциональной области. Профайл может представлять полный набор или произвольное подмножество элементов управления. Профайлы конфигурации сохраняются в виде файлов XML. AIX Runtime Expert позволяет управлять профайлами и применять их в системах.

Профайл может содержать параметры конфигурации без значений. Профайл без параметров предназначен для извлечения текущих значений системы из указанного профайла. Профайлы, содержащие по крайней мере один параметр без значения, имеют следующие ограничения:

- При выполнении команды **artexset** возникает ошибка.

- Команда **artexdiff** возвращает предупреждение для каждого параметра без значения.

Параметр профайла может содержать следующие значения:

- Нет значения
- Значения типа blob (двоичные данные в кодировке base64 в виде текстового файла). Значение blob применяется для замены существующих файлов, таких как `/etc/motd` и `/etc/hosts`.
- Значения, отличные от blob (отдельные значения, присваиваемые параметрам конфигурации, такие как integer или string).

Каталог `/etc/security/artex/samples` содержит примеры профайлов. Примеры профайлов могут содержать только имена параметров, поддерживаемых конфигурацией по умолчанию, устанавливаемой вместе с AIX Runtime Expert. Для параметров из примеров профайлов не указаны значения. Примеры профайлов доступны только для чтения. Их следует использовать в качестве шаблонов для создания новых профайлов конфигурации. Существующие примеры нельзя применить в активной системе.

Ниже приведены примеры команд настройки, которыми можно управлять с помощью профайлов конфигурации:

- Конфигурация сети
  - нет
  - mktcpip
- Конфигурация ядра
  - ioo
  - schedo
- Конфигурация RAS
  - alog
- Конфигурация защиты
  - setsecattr

Пример

Ниже приведен пример профайла конфигурации для различных каталогов и подкаталогов, в котором указаны значения различных параметров. Этот профайл можно изменить с помощью редактора XML или с помощью команды **vi**.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Profile origin="get" version="1.0" date="2009-04-25T15:33:37Z">
<Catalog id="vmoParam">
<Parameter name="kernel_heap_psize" value="0" applyType="nextboot" reboot="true" />
<Parameter name="maxfree" value="1088" />
</Catalog>
<Catalog id="noParam">
<SubCat id="tcp_network">
<Parameter name="tcp_recvspace" value="16384" />
<Parameter name="tcp_sendspace" value="16384" />
</SubCat>
<SubCat id="general_network">
<Parameter name="use_sndbufpool" value="1" applyType="nextboot" reboot="true" />
</SubCat>
</Catalog>
<Catalog id="lvmoParam">
<Parameter name="max_vg_pbuf_count" value="0">
<Target class="vg" instance="rootvg" />
</Parameter>
<Parameter name="pv_pbuf_count" value="512">
<Target class="vg" instance="rootvg" />
</Parameter>
</Catalog>
```

Задачи, связанные с данной:

“Изменение профайлов AIX Runtime Expert” на стр. 69
Профайлы AIX Runtime Expert - это файлы XML, которые можно изменить с помощью любого редактора XML или текстового редактора.

“Создание профайлов AIX Runtime Expert” на стр. 68
Новый профайл можно создать на основе примеров из каталога `/etc/security/artex/samples` с помощью команды **artexget**. Примеры профайлов - это шаблоны, с помощью которых можно создать собственный профайл и сохранить его в отдельном файле.

“Получение значений профайла AIX Runtime Expert” на стр. 71
Команда **artexget** позволяет найти информацию о профайле.

“Применение профайлов AIX Runtime Expert” на стр. 71
Команда **artexset** позволяет применить в системе параметры конфигурации из профайла.

Каталоги AIX Runtime Expert:

Каталоги применяются для описания элементов управления конфигурацией, применяемых в AIX Runtime Expert.

Каталоги предусмотрены для элементов управления, поддерживаемых AIX Runtime Expert. Они представляют собой файлы определения, связывающие значения профайлов конфигурации с параметрами, применяемыми для выполнения команд и действий по настройке.

Значения, доступные для изменения, представлены в виде каталогов, расположенных в каталоге `/etc/security/artex/catalogs`. Их не следует изменять.

Каждый каталог содержит параметры отдельного компонента. Однако некоторые каталоги могут содержать параметры нескольких тесно связанных компонентов. Имена каталогов описывают соответствующие компоненты. Элемент XML `<description>` содержит описание каталога.

AIX Runtime Expert и LDAP:

AIX Runtime Expert может получить профайлы с сервера Упрощенного протокола доступа к каталогам (LDAP).

Профайлы AIX Runtime Expert должны храниться в качестве объектов ibm-artexProfile и содержать следующие обязательные атрибуты:

- Ibm-artexProfileName. Имя профайла AIX Runtime Expert.
- Ibm-artexProfileXMLData. Содержимое XML профайла AIX Runtime Expert в формате octetString.

Перед сохранением профайлов AIX Runtime Expert на сервере LDAP должна быть установлена схема AIX Runtime Expert. Настройка сервера LDAP для поддержки AIX Runtime Expert аналогична настройке идентификации пользователей на сервере LDAP. Дополнительная информация о настройке LDAP приведена в разделе Настройка сервера идентификационной информации ITDS.

Настройка клиента LDAP для поддержки AIX Runtime Expert аналогична настройке идентификации пользователей на клиенте LDAP. Дополнительная информация приведена в разделе Настройка клиента LDAP. Для настройки клиента LDAP введите команду **mksecldap -c**, позволяющую правильно настроить демон **secldapclntd**. AIX Runtime Expert использует демон **secldapclntd** для обращения к серверу LDAP. По умолчанию AIX Runtime Expert выполняет поиск записей профайла с помощью идентификатора `DN: ou=artex,cn=AIXDATA`. При необходимости DN можно настроить, указав ключ artexbasedn в файле конфигурации secldapclntd `/etc/security/ldap/ldap.cfg`.

Загрузка профайла AIX Runtime Expert

Для загрузки профайла AIX Runtime Expert можно создать файл LDIF (формат обмена данными LDAP) и ввести команду **ldapadd** или воспользоваться инструментом администрирования LDAP, таким как Tivoli Directory Server Web Administration Tool.

Ниже приведен пример профайла, сохраненного в формате LDIF:

```
dn: ou=artex,cn=AIXDATA
objectClass: organizationalUnit
objectClass: top
ou: artex

dn: ibm-artexProfileName=alogProfile.xml,ou=artex,cn=AIXDATA
objectClass: ibm-artexProfile
objectClass: top
ibm-artexProfileName: alogProfile.xml
ibm-artexProfileXMLData:< file:///etc/security/artex/samples/alogProfile.xml
```

Ниже приведен пример загрузки профайла с помощью команды **ldapadd** (имя файла LDIF - sample.ldif):

```
ldapadd -c -h <хост-ldap> -D cn=admin -w <пароль> -f sample.ldif
```

Задачи, связанные с данной:

“Создание профайлов AIX Runtime Expert” на стр. 68
Новый профайл можно создать на основе примеров из каталога `/etc/security/artex/samples` с помощью команды **artexget**. Примеры профайлов - это шаблоны, с помощью которых можно создать собственный профайл и сохранить его в отдельном файле.

Информация, связанная с данной:

IBM Security Directory Server

AIX Runtime Expert и RBAC:

Для того чтобы предоставить пользователям, отличным от root, возможность выполнять команды AIX Runtime Expert, можно воспользоваться Role Based Access Control (RBAC).

Авторизации в AIX Runtime Expert

При установке набора файлов **artex.base.rte** создаются три системные авторизации, которые обеспечивают различные уровни доступа к функциональности AIX Runtime Expert:

- Авторизация **aix.system.config.artex.read** позволяет выполнять команды **artexlist** и **artexmerge**. Команды **artexget** и **artexdiff** также разрешены, но лишь для получения значений профайла. Значения нельзя получить из системы (т.е. команду **artexget** нельзя запустить с флагами *–r*, *–n* и *–p*, а команду **artexdiff** можно запускать только между двумя профайлами).
- Авторизация **aix.system.config.artex.get** позволяет выполнять те же операции, что и **artex.system.config.read** , а также выполнять команды **artexget** и **artexdiff** без ограничений.
- Авторизация **aix.system.config.artex.set** позволяет выполнять те же операции, что и **artex.system.config.get**, а также команду **artexset**.

Роли AIX Runtime Expert

AIX Runtime Expert не создает никакой новой роли, однако наборы файлов **artex.base.rte** добавляют авторизацию **aix.system.config.artex** к роли **SysConfig**. Любой пользователь с ролью **SysConfig** или любой объемлющей ролью (например, ролью **isso**), сможет выполнять команды **artexlist**, **artexmerge**, **artexdiff**, **artexget** и **artexset**.

Ограничения

В целях обеспечения безопасности применять переменную среды **ARTEX_CATALOG_PATH** разрешено только пользователю root. Пользователи, отличные от root, которым предоставлено право выполнять команды AIX Runtime Expert через RBAC, не могут применять переменную среды **ARTEX_CATALOG_PATH**.

Управление AIX Runtime Expert

AIX Runtime Expert использует несколько простых команд для создания, изменения, объединения и применения профайлов.

Настройка AIX Runtime Expert:

AIX Runtime Expert использует файл конфигурации `/etc/security/artex/config/artex.conf`.

Запись в файле конфигурации состоит из имени опции конфигурации, за которой следуют один или несколько пробелов и значение. Пустые строки и строки, начинающиеся со знака #, игнорируются.

Поддерживаются следующие опции:

Таблица 1. Опции конфигурации

Опции	Описание
ARTEX_CATALOG_PATH	Список каталогов через запятую, в которых выполняется поиск файлов-каталогов. Эта опция переопределяется переменной среды **ARTEX_CATALOG_PATH**. Путь по умолчанию - `/etc/security/artex/catalogs`.
ARTEX_PROFILE_PATH	Список каталогов через запятую, в которых выполняется поиск файлов профайла командой **artexlist**, если каталог не задан. Эта опция переопределяется переменной среды **ARTEX_PROFILE_PATH**. Путь по умолчанию - `/etc/security/artex/samples`.
DEBUG_LOG_CATEGORY	Категория отладки для файла протокола. Эту опцию можно повторять для выбора нескольких категорий отладки.
DEBUG_LOG_LEVEL	Уровень отладки файла протокола - от 0 (трассировка отладки отсутствует) до 3 (максимально подробный).
MAX_CMDS	Максимальное количество внешних команд, выполняемых параллельно. Внешние команды, выполняемые AIX Runtime Expert, помещаются в очередь, так что в каждый момент выполняется не более чем MAX_CMDS внешних команд одновременно. Значение по умолчанию - 10.

Создание профайлов AIX Runtime Expert:

Новый профайл можно создать на основе примеров из каталога `/etc/security/artex/samples` с помощью команды **artexget**. Примеры профайлов - это шаблоны, с помощью которых можно создать собственный профайл и сохранить его в отдельном файле.

Для создания профайла со всеми параметрами, поддерживаемыми AIX Runtime Expert, выполните следующие действия:

1. Настройте систему, указав параметры, которые должны быть отражены в профайле.
2. Перейдите в каталог примеров: `/etc/security/artex/samples`
3. Для создания нового профайла с именем `custom_all.xml` выполните следующую команду:

```
artexget -p all.xml >
/каталог-нового-профайла/custom_all.xml
```

Примечание: Профайл `custom_all.xml` можно использовать для настройки других систем с аналогичной конфигурацией.

Для создания профайла отдельного компонента выполните следующие действия:

1. Настройте систему, указав параметры, которые должны быть отражены в профайле.
2. Перейдите в каталог примеров: `/etc/security/artex/samples`
3. Создайте новый профайл `custom_no.xml` на основе примера профайла `noProfile.xml` с помощью следующей команды:

   ```
   artexget -p noProfile.xml > /каталог-нового-профайла/custom_no.xml
   ```

Новый профайл можно настроить путем изменения или удаления значений параметров с помощью редактора XML или текстового редактора.

Пользовательские профайлы можно загрузить на сервер LDAP для применения в нескольких системах AIX. Для загрузки профайлов на сервер LDAP применяются соответствующие инструменты LDAP.

Понятия, связанные с данным:

“AIX Runtime Expert и LDAP” на стр. 66
AIX Runtime Expert может получить профайлы с сервера Упрощенного протокола доступа к каталогам (LDAP).

“Профайлы AIX Runtime Expert” на стр. 64
Профайлы AIX Runtime Expert применяются для настройки параметров активной системы, извлечения конфигурации активной системы, а также для сравнения значений с активной системой или другим профайлом.

Задачи, связанные с данной:

“Получение значений профайла AIX Runtime Expert” на стр. 71
Команда **artexget** позволяет найти информацию о профайле.

“Применение профайлов AIX Runtime Expert” на стр. 71
Команда **artexset** позволяет применить в системе параметры конфигурации из профайла.

Информация, связанная с данной:

Команда artexget

Изменение профайлов AIX Runtime Expert:

Профайлы AIX Runtime Expert - это файлы XML, которые можно изменить с помощью любого редактора XML или текстового редактора.

Профайлы, созданные с помощью команды **artexget**, можно настроить путем изменения значений параметров или удаления отдельных параметров.

Для изменения профайлов AIX Runtime Expert выполните следующие действия:

1. Из каталога, в котором расположен файл `custom_all.xml`, выполните следующие команды для сохранения копии профайла:

   ```
   cp custom_all.xml custom_all_backup.xml
   ```

2. Из каталога, в котором расположен файл `custom_all.xml`, выполните следующую команду для изменения профайла:

   ```
   vi custom_all.xml
   ```

   Примечание: Для внесения изменений можно использовать редактор XML или текстовый редактор.

3. Измените значения параметров или удалите ненужные параметры.
4. Для проверки правильности внесенных изменений путем их сравнения с текущими параметрами системы выполните следующую команду:

   ```
   artexdiff -c -r custom_all.xml custom_all_backup.xml
   ```

Команда **artexdiff** отображает параметры, измененные с помощью редактора. `<FirstValue>` соответствует значению профайла, `<SecondValue>` - текущему значению системы.

Понятия, связанные с данным:

“Профайлы AIX Runtime Expert” на стр. 64
Профайлы AIX Runtime Expert применяются для настройки параметров активной системы, извлечения конфигурации активной системы, а также для сравнения значений с активной системой или другим профайлом.

Задачи, связанные с данной:

“Получение значений профайла AIX Runtime Expert” на стр. 71
Команда **artexget** позволяет найти информацию о профайле.

“Применение профайлов AIX Runtime Expert” на стр. 71
Команда **artexset** позволяет применить в системе параметры конфигурации из профайла.

Информация, связанная с данной:

Команда artexdiff

Объединение профайлов AIX Runtime Expert:

Профайл может представлять полный набор или произвольное подмножество элементов управления. Одним из удобных способов изменения профайлов является объединение профайлов с помощью команды **artxmerge**.

Команда **artexmerge** позволяет создать профайл на основе нескольких профайлов.

Для объединения профайлов выполните следующие действия:

1. Из каталога профайлов выполните следующую команду:
   ```
   artexmerge профайл1.xml профайл2.xml > новый-профайл.xml
   ```
2. Для проверки полученного профайла выполните следующую команду:
   ```
   artexget новый-профайл.xml
   ```

   Примечание: Если исходные профайлы содержат повторяющиеся параметры, то в ходе объединения произойдет ошибка. Если указан флаг **-f**, то применяются значения параметров из последнего профайла.

Информация, связанная с данной:

Команда artexmerge

Поиск профайлов AIX Runtime Expert:

Команда **artexlist** позволяет найти профайлы в указанном локальном каталоге, а также на сервере LDAP.

Для поиска профайлов выполните следующие действия:

1. Если профайл расположен в локальной системе, то выполните следующую команду:
   ```
   artexlist
   ```
2. Если профайл расположен на сервере LDAP, то выполните следующую команду:
   ```
   artexlist -l
   ```

По умолчанию команда отображает список профайлов из каталога `/etc/security/artex/samples`. При необходимости в переменной среды **ARTEX_PROFILE_PATH** через точку с запятой можно указать другие расположения.

Информация, связанная с данной:

Команда artexlist

Получение значений профайла AIX Runtime Expert:

Команда **artexget** позволяет найти информацию о профайле.

С помощью профайла можно просмотреть значения профайла или системы в различных форматах (XML, CSV или текст) с применением различных фильтров, таких как параметры, для применения которых требуется перезагрузка и параметры, требующие перезапуска служб.

Просмотр значений системы рекомендуется в следующих случаях:

Для создания моментальной копии системы
Если система настроена правильным образом, ее конфигурацию можно сохранить путем создания моментальной копии. Впоследствии с помощью этой моментальной копии можно определить, какие параметры были изменены. Профайл моментальной копии позволяет восстановить исходную конфигурацию системы.

Для копирования конфигурации системы для применения в других системах
Параметры настроенный системы можно сохранить в профайле AIX Runtime Expert и применить полученный профайл в других системах.

Для отладки неполадок
В случае обнаружения неполадки в рабочей системе с помощью профайла ее конфигурацию можно перенести в тестовую систему для отладки неполадок.

Для просмотра информацию о профайле выполните следующие действия:

1. Перейдите в каталог, в котором расположен нужный профайл.
2. Для просмотра информации о профайле выполните следующую команду:
   ```
   artexget профайл.xml
   ```

   Ограничение: Если в системе определено много пользователей, то для выполнения команд AIX Runtime Expert **artexget**, **artexset** и **artexdiff** над такими профайлами как chuserProfile.xml, coreProfile.xml или all.xml profiles потребуется больше времени, чем обычно.

Понятия, связанные с данным:

“Профайлы AIX Runtime Expert” на стр. 64
Профайлы AIX Runtime Expert применяются для настройки параметров активной системы, извлечения конфигурации активной системы, а также для сравнения значений с активной системой или другим профайлом.

Задачи, связанные с данной:

“Создание профайлов AIX Runtime Expert” на стр. 68
Новый профайл можно создать на основе примеров из каталога `/etc/security/artex/samples` с помощью команды **artexget**. Примеры профайлов - это шаблоны, с помощью которых можно создать собственный профайл и сохранить его в отдельном файле.

“Изменение профайлов AIX Runtime Expert” на стр. 69
Профайлы AIX Runtime Expert - это файлы XML, которые можно изменить с помощью любого редактора XML или текстового редактора.

Информация, связанная с данной:

Команда artexget

Применение профайлов AIX Runtime Expert:

Команда **artexset** позволяет применить в системе параметры конфигурации из профайла.

Для применения пользовательского профайла выполните следующие действия:

1. Перейдите в каталог, в котором расположен профайл.
2. Для применения профайла выполните следующую команду:

```
artexset -c профайл.xml
```

3. Необязательно: Если профайл должен применяться каждый раз при перезапуске системы, выполните следующую команду:

   ```
   artexset -b профайл.xml
   ```

   Примечание: Параметры с ограничениями поддерживаются как параметры только для чтения. Поэтому значения этих параметров можно получить с помощью команды **artexget**, но нельзя задать командой **artexset**.

Понятия, связанные с данным:

“Профайлы AIX Runtime Expert” на стр. 64
Профайлы AIX Runtime Expert применяются для настройки параметров активной системы, извлечения конфигурации активной системы, а также для сравнения значений с активной системой или другим профайлом.

Задачи, связанные с данной:

“Создание профайлов AIX Runtime Expert” на стр. 68
Новый профайл можно создать на основе примеров из каталога `/etc/security/artex/samples` с помощью команды **artexget**. Примеры профайлов - это шаблоны, с помощью которых можно создать собственный профайл и сохранить его в отдельном файле.

“Изменение профайлов AIX Runtime Expert” на стр. 69
Профайлы AIX Runtime Expert - это файлы XML, которые можно изменить с помощью любого редактора XML или текстового редактора.

Информация, связанная с данной:

Команда artexset

Откат профайлов AIX Runtime Expert:

Команда **artexset –u** позволяет восстановить предыдущие параметры конфигурации системы. Можно вернуть параметры системы, которые использовались до применения профайла.

Команда **rollback** недоступна, если параметры системы не изменялись в текущем сеансе.

Откат не считается изменением образа операционной системы. При использовании команды **rollback** не производится удаление или создание ресурсов. В ходе отката восстанавливаются предыдущие значения параметров конфигурации. Команда **rollback** не позволяет восстановить параметры за конкретные дату и время. Восстанавливаются параметры, которые использовались до внесения изменения.

Команду **rollback** можно использовать в следующих случаях:

- Тестирование изменений конфигурации. Если новая конфигурация работает неправильным образом, то можно быстро восстановить предыдущую конфигурацию.
- Отладка системы. Если система начинает работать недостаточно эффективно, то откат позволяет определить, связано ли это с внесенными изменениями.
- Реализация нового профайла для обработки исключительной ситуации. Например, если конкретное действие выполняется в системе только раз в месяц и после его выполнения требуется восстановить предыдущую конфигурацию системы.

Для отката системы выполните следующие действия:

1. Для того чтобы выполнить откат профайла, выполните следующие действия:

   ```
   artexset -u
   ```

2. Для проверки правильности отката профайла выполните следующую команду:

   ```
   artexdiff -f txt -r -профайл.xml
   ```

   Примечание: Где *профайл.xml* - это имя последнего действующего профайла системы.

Будут показаны различия между параметрами системы и профайлом.

Информация, связанная с данной:

Команда artexget

Команда artexlist

Сравнение профайлов AIX Runtime Expert:

Команда **artexdiff** позволяет сравнить два профайла или значения профайла с системными значениями.

Для сравнения профайлов выполните следующие действия:

1. Выполните следующую команду в системе 1:
   ```
   artexget -p all.xml > all_system1.xml
   ```
2. Выполните следующую команду в системе 2:
   ```
   artexget -p all.xml > all_system2.xml
   ```

Для определения параметров конфигурации, которые были изменены за указанный интервал времени (например, пока вы были в отпуске) выполните следующие команды:

- После возврата из отпуска выполните следующую команду:
  ```
  $ artexget -p all.xml > all_before_vacation.xml
  ```
- Для просмотра изменений, внесенных в конфигурацию за время отпуска, выполните следующую команду:
  ```
  $ artexdiff -c -p
  all_before_vacation.xml
  ```

Информация, связанная с данной:

Команда artexget

Команда artexlist

Написание профайлов AIX Runtime Expert

Область действия программы AIX Runtime Expert можно расширить, добавив доступные ей файлы-каталоги и профайлы. Прежде чем приступить к написанию новых файлов-каталогов, следует ознакомиться с концепциями AIX Runtime Expert.

Параметр - это наименьшая информационная единица, обрабатываемая AIX Runtime Expert. В качестве параметров могут выступать настраиваемые сущности, файлы конфигурации, переменные среды и свойства объектов, таких как пользователи, устройства или подсистемы (такие объекты называются целями в контексте AIX Runtime Expert).

Параметры объединяются в профайлы в соответствии с областью применения (пользователь, tcpi). Профайлы - это специальные средства взаимодействия между пользователями и средой AIX Runtime Expert. Профайл служит входными данными для команды **artexget**, извлекающей значение параметра в системе и возвращающей профайл. Профайл (включая значения) служит входными данными для команды **artexset**, присваивающей параметрам значения, считанные из профайла.

Концепции записи профайла AIX Runtime Expert:

Профайлы AIX Runtime Expert - это файлы XML, содержащие список параметров конфигурации, а также, при необходимости, значения параметров и флаги использования.

Профайлы могут быть расположены в системе, настраиваемой во время использования команд AIX Runtime Expert напрямую в командной строке.

Расположения профайлов:

Примеры профайлов AIX Runtime Expert находятся в каталоге `/etc/security/artex/samples`.

Когда вы записываете новый файл-каталог, который должен поддерживать AIX Runtime Expert, рекомендуется также записать пример профайла, который можно было бы использовать в качестве записи для команды **artexget**. Пример профайла - это предназначенный только для чтения профайл без присвоенных параметрам значений. Существующие примеры профайлов расположены в каталоге `/etc/security/artex/samples`. По умолчанию команда **artexlist** выдает только профайлы из каталога по умолчанию, но каталог по умолчанию можно изменить, задав переменную среды **ARTEX_PROFILE_PATH**. В этой переменной среды можно указать и несколько каталогов, разделив их двоеточием.

Все профайлы из каталога примеров объединяются во время установки набора файлов **artex.base.samples**, образуя профайл **default.xml**, используемый командой **snap**. Профайл, который не должен быть частью профайла **default.xml**, не следует включать в каталог примеров. Примерами профайлов, которые не следует включать в профайл **default.xml**, служат профайлы, которые могут содержать тысячи параметров (например, профайлы, применяющие пользователей в качестве целевого класса), и профайлы, которые должны запускаться только в конкретных системах (например, профайл атрибутов **vios**).

Присвоение имен профайлам:

Имена профайлам AIX Runtime Expert присваиваются на основе команд.

Профайлы обычно формируются вокруг одной команды или набора команд. Профайлы могут содержать несколько файлов-каталогов, если те тесно связаны. По соглашению имена файлам присваиваются на основе команды: например, примеру профайла присваивается имя **commandProfile.xml**, а файлу-каталогу - имя **commandParam.xml**, но это необязательно. Обязательно лишь расширение **.xml**.

Процесс создания профайла:

В этом разделе рассмотрен процесс написания нового профайла AIX Runtime Expert.

При написании нового профайла AIX Runtime Expert выполните следующие действия:

1. Создайте список параметров, которые вы хотите включить в профайл.
2. Создайте элемент **<Parameter name="...">** для каждого параметра, задав в атрибуте *name* имя, используемое в элементе **<ParameterDef>** файла-каталога.
3. Сгруппируйте все параметры, определенные в одном файле-каталоге, внутри одного элемента **<Catalog id="...">**, задав в атрибуте *id* идентификатор, используемый в элементе **<Catalog>** файла-каталога.
4. Для каждого элемента **<Parameter>** выполните следующие действия:
   a. Если параметр определен со значением *reboot*=`true` в файле-каталоге, добавьте атрибуты *reboot*=`true` и *applyType*=`nextboot`.
   b. Если параметр необходимо лишь извлечь, но не задавать, добавьте атрибут `readOnly=true`.
   c. Если параметр определен с непустым атрибутом *targetClass* в файле-каталоге, выполните следующие действия:
      1) Если для параметра требуется обнаружить цель, то определите отдельный элемент **<Parameter>** для этого параметра и укажите специальную цель **<Target class="" instance="" >** для этого элемента.
      2) Если для этого параметра необходимо определить конкретные цели, то определите элемент **<Parameter>** для каждой цели. Под каждым элементом **<Parameter>** определите соответствующие элементы **<Target class="..." instance="..." />**, чтобы полностью задать цель.
5. Проверьте профайл, выполнив команду **artexget –r**.

Элементы профайла AIX Runtime Expert:

Элемент <Profile>:

Элемент **<Profile>** - это корневой элемент для всех профайлов.

Синтаксис

Поддерживаются следующие атрибуты:

Таблица 2. Атрибуты

Атрибут	Обязательный	Тип	Описание
origin	нет	строка	Источник профайла.
date	нет	дата-время	Дата создания или последнего изменения профайла. Формат - ГГГГ-ММ-ДДТчч:мм:сс.
readOnly	нет	булевский	Указывает, можно ли использовать этот профайл в операции set. Значение по умолчанию - `false`.
version	нет	строка	Номер версии профайлов.

Поддерживаются следующие дочерние элементы:

Таблица 3. Дочерние элементы

Дочерний элемент	Обязательный	Количество	Описание
<ShortDescription>	нет	0 - 1	Краткое текстовое описание файла-каталога.
<Description>	нет	0 – 1	Длинное текстовое описание файла-каталога.
<Comments>	нет	0 – 1	Пользовательские комментарии.
<Catalog>	нет	0 – любое	Файл-каталог, необходимый для выполнения операций над профайлом.

Атрибуты

Атрибут *origin*

Атрибут *origin* - это информационный атрибут, которому могут быть присвоены следующие значения:

- При создании примера профайла атрибуту *origin* должно быть присвоено значение `reference`.
- При создании профайла с помощью команды **artexget** атрибуту *origin* автоматически присваивается значение `get`.

Дочерние элементы

Элемент **<Comments>** - это необязательная текстовая строка, зарезервированная для других целей. Этот элемент не следует указывать при создании профайла вручную, и он не используется базовыми командами AIX Runtime Expert.

Примеры

1. Пустой пример профайла будет выглядеть примерно следующим образом:

```
<?xml version="1.0" encoding="UTF-8"?>
<Profile origin="reference" version="2.0.0" readOnly="true">
</Profile>
```

2. Команда **artexget -r /etc/security/artex/samples/smtctmProfile.xml** выдает профайл примерно следующего вида:

```
<?xml version="1.0" encoding="UTF-8"?>
<Profile origin="get" version="2.0.1" date="2010-09-29T07:50:56Z">
  <Catalog id="smtctlParam" version="2.0">
    <Parameter name="enableSMT" value="1"/>
  </Catalog>
</Profile>
```

Связанная информация

Элемент <Catalog>

Элемент <Description> и <ShortDescription>.

Элементы <Description> и <ShortDescription>:

Элементы **<Description>** и **<ShortDescription>** позволяют задать текстовое описание профайлов и параметров.

Синтаксис

Родительский элемент элемента **<ShortDescription>**:

- Элемент **<Profile>**

Родительским элементом элемента **<Description>** может быть:

- Элемент **<Profile>**
- Элемент **<Parameter>**

Формат элементов **<Description>** и **<ShortDescription>** одинаков. Текст, содержащийся в элементе **<Description>**, - это строковое содержимое тега XML.

Формат

В настоящее время описания в файлах профайла не используются средой AIX Runtime Expert. Команды AIX Runtime Expert игнорируют все комментарии во входном профайле.

Примеры

Ниже приведен пример элементов **<Description>** и **<ShortDescription>**:

```
<ShortDescription>
   Краткое описание содержимого поля.
</ShortDescription>
<Description>
Это текстовое поле можно использовать для отображения подробных сведений об
использовании родительского элемента.
</Description>
```

Связанная информация

Элемент <Profile>.

Элемент <Parameter>.

Элемент <Catalog>:

Элемент **<Catalog>** указывает имя файла-каталога, содержащего определения дочерних элементов **<Parameter>**.

Синтаксис

Родительский элемент: **<Profile>**

Поддерживаются следующие атрибуты:

Таблица 4. Атрибуты

Атрибут	Обязательный	Тип	Описание
id	да	строка	Задает идентификатор каталога. Это имя должно быть уникально в системе.
version	нет	строка	Задает версию каталога, на основе которой формируется этот профайл.

Поддерживаются следующие дочерние элементы:

Таблица 5. Дочерние элементы

Дочерний элемент	Обязательный	Количество	Описание
<Parameter>	нет	0 – любое	Параметр, включенный в этот файл-каталог.
<SubCat>	нет	0 – любое	Подкатегория, включенная в этот каталог.
<Seed>	нет	0 – любое	Начальное значение, включенное в каталог.

Атрибуты

Атрибут *id*

Атрибуту *id* необходимо присвоить имя файла-каталога, определяющего параметры, перечисленные под элементом **<Catalog>**. Атрибут *id* - это базовое имя файла-каталога на диске, без расширения **.xml**. Например, профайл будет использовать элемент **<Catalog id="commandParam">** для ссылки на файл-каталог `commandParam.xml`.

По умолчанию поиск файлов-каталогов выполняется под каталогом `/etc/security/artex/catalogs`. Однако возможно - только пользователю root - выполнять поиск и в других каталогах, если задать переменную среды **ARTEX_CATALOG_PATH**. В этой переменной среды можно указать несколько каталогов, разделив их двоеточием.

Атрибут *version*

Атрибут *version* записывается в формате *ММ.мм*, где *ММ* называется основным, а *мм* - дополнительным номером.

Атрибут *version* должен соответствовать версии опорного файла-каталога (см. элемент <Catalog> в разделе Написание файлов-каталогов AIX Runtime Expert). Если команда AIX Runtime Expert применяется к профайлу, ссылающемуся на файл-каталог неправильной версии, то будет показано следующее предупреждающее сообщение:

```
0590-218 Версия файла-каталога отличается от той, что указана в профайле
Версия локального файла-каталога- '2.1'. Версия, использовавшаяся для формирования профайла, - '2.0'
```

Формат

Элемент **<Catalog>** указывает файл-каталог, содержащий определение перечисленных начальных значений и параметров. Все начальные значения и элементы параметра в профайле должны находиться в соответствующем элементе **<Catalog>**.

Профайл может ссылаться на несколько каталогов. Примером может служить профайл **default.xml**, который формируется во время установки набора файлов `artex.base.sample` путем объединения избранного набора других примеров файлов.

Примеры

Профайл атрибутов защиты `secattrProfile.xml` использует три файла-каталога, каждый из которых обрабатывает одну из таблиц защиты:

```
<Profile origin="reference" readOnly="true" version="2.0.0">
  <Catalog id="privcmdParam" version="2.0"
    <Parameter name="privatecommands" />
  </Catalog>
  <Catalog id="privdevParam" version="2.0">
    <Parameter name="privatedevices"/>
  </Catalog>
  <Catalog id="privfileParam" version="2.0">
    <Parameter name="privatefiles" />
  </Catalog>
</Profile>
```

Связанная информация

Элемент **<Catalog>** (в файлах-каталогах).

Элемент <SubCat>:

Элемент **<SubCat>** позволяет создавать логические подкатегории в элементе **<Catalog>**.

Синтаксис

Родительский элемент: **<Catalog>**,**<SubCat>**

Поддерживаются следующие атрибуты:

Таблица 6. Атрибуты

Атрибут	Обязательный	Тип	Описание
id	да	строка	Задает имя подкатегории. Это имя должно быть уникальным в пределах элемента **<Catalog>**.

Поддерживаются следующие дочерние элементы

Таблица 7. Дочерние элементы

Дочерний элемент	Обязательный	Количество	Описание
<Parameter>	нет	0 – любое	Содержит имя параметра.
<SubCat>	нет	0 – любое	Вложенная подкатегория

Атрибуты

Атрибут *id* (идентификатор) однозначно идентифицирует подкатегорию в каталоге. Профайл может содержать несколько подкатегорий с одинаковым идентификатором, при условии что они не находятся в одном элементе **<Catalog>**.

Дочерние элементы

Элемент **<SubCat>** может содержать другой элемент **<SubCat>** как дочерний. Количество возможных вложенных подкатегорий не ограничено.

Формат

Подкатегории предоставляются исключительно для удобочитаемости. Они не влияют на фактическую обработку параметров.

Примеры

Профайл `noProfile.xml` включает несколько подкатегорий. Например:

```
<Profile origin="reference" readOnly="true" version="2.0.0">
  <Catalog id="noParam" version="2.0">
    <SubCat id="general_network"
      <Parameter name="fasttimo"/>
      <Parameter name="nbc_limit"/>
    </SubCat>
    <SubCat id="tcp_network">
      <Parameter name="clean_partial_conns"/>
      <Parameter name="delayack"/>
    </SubCat>
    <SubCat id="restricted">
      <Parameter name="extendednetstats" readOnly="true"/>
      <Parameter name="inet_stack_size" readOnly="true"/>
    </SubCat>
  </Catalog>
</Profile>
```

Связанная информация

Элемент <Parameter>.

Элемент <Parameter>:

Элемент **<Parameter>** определяет параметр конфигурации.

Синтаксис

Поддерживаются следующие атрибуты:

Таблица 8. Атрибуты

Атрибут	Обязательный	Тип	Описание
name	да	строка	Задает имя параметра. Это имя должно быть уникальным в файле-каталоге.
value	нет	строка	Значение параметра, если оно определено.
applyType	нет	строка	Указывает, следует ли извлекать или задавать, если флаг не задан, текущее или присваиваемое при следующей перезагрузке значение параметра. Значение по умолчанию - `runtime`.

Таблица 8. Атрибуты (продолжение)

Атрибут	Обязательный	Тип	Описание
reboot	нет	булевский	Значение `true` означает, что перезагрузка обязательна, чтобы изменение значения вступило в силу. Значение по умолчанию - `false`.
readOnly	нет	булевский	Указывает, запрещено ли задавать значение параметра. Значение по умолчанию - `false`.
disruptive	нет	булевский	Указывает, порождает ли метод, используемый для задания параметра, прерывающие ограничения. Значение по умолчанию - `false`.
setDiscover	нет	булевский	Указывает, следует ли задать метод set равным значению, соответствующему всем обнаруженным экземплярам целевого класса. Значение по умолчанию - `false`.

Поддерживаются следующие дочерние элементы

Таблица 9. Дочерние элементы

Дочерний элемент	Обязательный	Количество	Описание
<Value>	нет	0 - 1	Значение параметра.
<Target>	нет	0 – любое	Цель, к которой применяется параметр.
<Description>	нет	0 - 1	Описание параметра.
<Property>	нет	0 – любое	Свойство параметра.

Атрибуты

Таблица 10. Атрибуты

Атрибут	Описание
name	Имя параметра - это единственный обязательный атрибут элемента **<Parameter>**. Вместе с именем файла-каталога, указанным в родительском элементе **<Catalog>**, имя параметра однозначно идентифицирует определение параметра в файле-каталоге.
value	Значение параметра может быть указано как атрибут или как дочерний элемент.

Таблица 10. Атрибуты (продолжение)

Атрибут	Описание
applyType	Атрибут *applyType* может принимать значения `runtime` (значение по умолчанию) или `nextboot`. Этот атрибут определяет команду, используемую для обработки параметра. В случае операций set атрибут *applyType*=`runtime` указывает, что для задания параметра следует применять команду **<Set type="permanent">** из файла-каталога. Атрибут *applyType*=`nextboot` указывает, что вместо нее следует применять команду **<Set type="nextboot">**. В случае операций get, если задан флаг *–p*, атрибут *applyType*=`runtime` указывает, что для получения параметра следует применять команду **<Get type="current">** из файла-каталога. Атрибут <*applyType*>=`nextboot` указывает, что вместо нее следует применять команду **<Get type="nextboot">**. Атрибут *applyType* должен быть равен `nextboot`, если атрибут *reboot* равен true.
reboot	Значение этого атрибута по умолчанию - false. Если он равен true, то это означает, что систему необходимо перезагрузить, прежде чем любое изменение параметра вступит в силу. Этот атрибут должен совпадать с атрибутом *reboot*, определенным в соответствующем элементе **<ParameterDef>** файла-каталога. Если этот атрибут равен true, то атрибут *applyType* должен быть равен `nextboot`. Когда вы задаете параметр, в котором задан атрибут *reboot*, пользователь получает предупреждение о необходимости перезагрузки: `0590-206 Для вступления изменений в силу необходимо вручную выполнить операцию POST Перезагрузите систему` Присваивать атрибуту *reboot* значение true следует только в том случае, если изменение значения параметра не вступит в силу до следующей перезагрузки.
readOnly	Этот атрибут указывает, что значение параметра может быть считано командой **artexget**, но не может быть ни задано командой **artexset**, ни учтено в операции сравнения с текущими значениями, выполняемой командами **artexdiff**. Значение по умолчанию - false. Ниже описаны несколько ситуаций, в которых можно рекомендовать задать атрибут *readOnly* равным true: • Параметр является статическим и его значение нельзя изменить (например, параметр *memory_frames* команды **vmo**). • Доступ к параметру ограничен и пользователю не рекомендуется изменять его в автоматических процедурах. В этом случае следует определить методы задания конфигурации set для этого параметра в файле-каталоге, а системный администратор должен вручную удалить атрибут *readOnly* из профайла, чтобы получить возможность задать параметр.

Таблица 10. Атрибуты (продолжение)

Атрибут	Описание
setDiscover	Атрибут *setDiscover*, будучи равным true, указывает, что при вызове команды **artexset** с флагом *–d* необходимо вызвать команду **discover**, чтобы обнаружить все экземпляры цели и задать их все равными значению, хранящемуся в профайле. Значение *setDiscover* по умолчанию - false. Значение true осмысленно лишь в том случае, если у параметра есть целевые классы, определенные в файле-каталоге. Не указывайте этот атрибут при создании примера профайла. Опытные пользователи должны добавить этот атрибут вручную, если сочтут это необходимым.

Другие атрибуты

Атрибуты *type* и *disruptive* - это информационные атрибуты, которые автоматически задаются командой **artextget** с флагом **-i**. Не указывайте эти атрибуты при создании примера профайла.

Примеры

1. Следующий пример представляет собой выдержку из примера файла-каталога `vmoProfile.xml`, демонстрирующую использование различных необязательных атрибутов:

```
<Profile origin="reference" readOnly="true" version="2.0.0">
  <Catalog id="vmoParam" version="2.1">
    <Parameter name="nokilluid"/>
    <Parameter name="memory_frames" readOnly="true"/>
    <Parameter name="kernel_heap_psize" reboot="true" applyType="nextboot"/>
  </Catalog>
</Profile>
```

2. Если вы примените команду **artexget –r** к профайлу из примера 1, то будет показан следующий профайл:

```
<Profile origin="get" version="2.0.1" date="2011-03-24T13:41:01Z">
  <Catalog id="vmoParam" version="2.1">
    <Parameter name="nokilluid" value="0"/>
    <Parameter name="memory_frames" value="393216" readOnly="true"/>
    <Parameter name="kernel_heap_psize" value="4096" applyType="nextboot" reboot="true"/>
  </Catalog>
</Profile>
```

Связанная информация

Раздел Значения параметра.

Элемент <ParameterDef>.

Значения параметров:

Значение параметра можно задать в профайле как атрибут, если оно достаточно коротко, или как дочерний элемент элемента **<Parameter>**.

Использование

При составлении примера профайла параметры указываются без значений. Значение параметра, если оно существует, автоматически включается в профайл в результате выполнения команды **artexget**.

Текущие и присваиваемые после перезагрузки значения

Принцип текущих (runtime) и присваиваемых после перезагрузки (nextboot) значений играет важную роль в среде AIX Runtime Expert.

Значение типа runtime - это текущее значение параметра, извлекаемое на момент выполнения команды **artexget**. Значение типа nextboot - это значение, которое будет присвоено параметру после следующей перезагрузки системы.

Например, рассмотрим параметр *type_of_dump* в профайле sysdumpdevProfile.xml. Текущим (runtime) значением этого параметра может быть traditional или firmware-assisted. При изменении этого значения (с помощью команды **artexset** или напрямую с помощью команды **sysdumpdev**) оно не вступит в силу до следующей перезагрузки. Тип значения этого параметра изменится на nextboot.

```
<Parameter name="type_of_dump" applyType="nextboot" reboot="true" />
```

Пример

В следующем примере показан параметр со значением, заданным как атрибут, и другой параметр со значением, заданным как дочерний элемент:

```
<Profile origin="get" version="2.0.1" date="2010-09-28T12:30:03Z">
<Catalog id="login.cfgParam" version="2.0">
<Parameter name="shells">
<Value>
/bin/sh,/bin/bsh,/bin/csh,/bin/ksh,/bin/tsh,
/bin/ksh93,/usr/bin/sh,/usr/bin/bsh,/usr/bin/csh,
/usr/bin/ksh,/usr/bin/tsh,/usr/bin/ksh93,
/usr/bin/rksh,/usr/bin/rksh93,
/usr/sbin/uucp/uucico,/usr/sbin/sliplogin,
/usr/sbin/snappd
</Value>
</Parameter>
<Parameter name="maxlogins" value="32767"/>
</Catalog>
</Profile>
```

Элемент <Property>:

Элемент **<Property>** присваивает значение свойству параметра.

Синтаксис

Родительский элемент: **<Parameter>**

Поддерживаются следующие атрибуты:

Таблица 11. Атрибуты

Атрибут	Обязательный	Тип	Описание
name	yes	строка	Задает имя свойства.
value	нет	строка	Задает значение свойства.

Формат

Элемент **<Property>** присваивает значение имени свойства родительского элемента. Это значение используется, когда последовательность **%p***[name]* расширена при генерации в командной строке.

Элемент **<Property>** обычно не добавляется вручную в профайлы. Элемент вставляется автоматически в выходной профайл при выполнении команд **artexget –r** и **artexget –n** на основании команды, определенной в

соответствующем элементе **<Property>** файла каталога.

Пример

Следующий пример устанавливает свойство **nodeId** для параметра `netaddr`. Значение свойства захватывается командой **artexget –r** и является выводом команды **uname –f**:

```
<Parameter name="netaddr" value="172.16.128.13">
 <Target class="device" instance="en0"/>
 <Property name="nodeId" value="8000108390E00009"/>
</Parameter>
```

Связанная информация

“Элемент <PropertyDef>” на стр. 118 (в файлах каталога).

Элемент <Seed>:

Элемент **<Seed>** определяет начальное значение, которое расширяется в один или несколько элементов **<ParameterDef>** при выполнении операции **<Get>**.

Синтаксис

Родительский элемент: **<Catalog>**

Поддерживается следующий атрибут:

Таблица 12. Атрибут

Атрибут	Обязательный	Тип	Описание
name	yes	строка	Указывает имя элемента seed, которое соответствует элементу **SeedDef** в файле каталога.

Поддерживаются следующие дочерние элементы:

Таблица 13. Дочерние элементы

Дочерний элемент	Обязательный	Номер	Описание
<Parameter>	нет	0 – любое	Обнаруженные фильтрами параметры на основании имен параметров.
<Target>	нет	0 – любое	Обнаруженные фильтрами параметры на основании их назначений.

Формат

Элемент **<Seed>** обнаруживает параметры динамически в процессе выполнения операции **<Get>**.

При выполнении команды **artexget** каждый элемент **<Seed>** во входном профайле расширяется в один или несколько элементов **<Parameter>**. Профайлы расширяются на основании правил, определенных в соответствующем элементе **<SeedDef>** файла каталога. Этот процесс называется обнаружением параметра. После завершения процесса поиска параметра команда **artexget** продолжает свою работу обычным образом с расширенным профайлом.

Необязательные дочерние элементы **<Parameter>** и **<Target>** используются для фильтрации обнаруженных параметров. Обнаруженные параметры, которые не соответствуют критерию, определенному в подэлементе **<Parameter>**, отбрасываются. Кроме того, те параметры, которые применяются к назначениям, не

соответствующим критерию, определенному в подэлементе **<Target>**, также отбрасываются.

Примеры

Этот пример использует каталог **devSeed** для определения начального значения, которое может быть использовано для обнаружения всех атрибутов всех устройств:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Catalog id="devSeed" version="3.0">
  <SeedDef name="devAttr">
    <Discover>
      <Command>
        /usr/sbin/lsdev -F 'name class subclass type' |
        while read DEV CLASS SUBCLASS TYPE
        do
          CAT=devParam.$CLASS.$SUBCLASS.$TYPE
          /usr/sbin/lsattr -F attribute -l $DEV |
          while read PAR
          do
            echo "device=$DEV $CAT $PAR"
          done
        done
      </Command>
      Mask target="1" catalog="2" name="3">(.*) (.*) (.*)</Mask>
    </Discover>
  </SeedDef>
</Catalog>
```

Следующий профайл может быть использован для обнаружения всех поддерживаемых атрибутов всех поддерживаемых устройств:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Profile>
  <Catalog id="devSeed" version="3.0">
    <Seed name="devAttr"/>
  </Catalog>
</Profile>
```

2. Используя тот же каталог, фильтр **<Target>** может обнаружить все поддерживаемые атрибуты всех адаптеров Ethernet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Profile>
  <Catalog id="devSeed" version="3.0">
    <Seed name="devAttr">
      <Target class="device" match="^en[0-9]+$"/>
    </Seed>
  </Catalog>
</Profile>
```

3. Можно добавить фильтр **<Parameter>** для захвата только атрибутов **netaddr**, **netaddr6**, **alias** и **alias6** всех адаптеров Ethernet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Profile>
  <Catalog id="devSeed" version="3.0">
    <Seed name="devattr">
      <Parameter match="^(netaddr|alias)6?$"/>
      <Target class="device" match="^en[0-9]+$"/>
    </Seed>
  </Catalog>
</Profile>
```

Связанная информация

Элемент “Элемент <SeedDef>” на стр. 112 (в файлах каталога).

Элемент <Target>:

Элемент **<Target>** определяет экземпляр целевого класса, к которому применяется параметр.

Синтаксис

Родительский элемент: **<Parameter>**

Несколько вхождений одного и того же параметра из одного и того же файла-каталога допускаются только в том случае, если они применяются к различным экземплярам своей цели.

Поддерживаются следующие атрибуты:

Таблица 14. Атрибуты

Атрибут	Обязательный	Тип	Описание
class	да	строка	Задает имя целевого класса.
instance	нет*	строка	Задает имя экземпляра класса.
match	нет*	строка	Задает регулярное выражение, применяемое к обнаруженным именам экземпляров.

* Атрибуты *instance* и *match* должны быть указаны один и только один раз.

Использование

Некоторые параметры применяются не к системе в целом, а к конкретному объекту. Примером может служить каталог `home` пользователя, указанный в профайле `chuserProfile.xml`; этот параметр применяется к конкретному пользователю (root, guest) в конкретном загружаемом модуле (файловом, LDAP). В этом примере пользователь и модуль - это два целевых класса. Параметр *home* применяется к конкретным экземплярам этих целевых классов. Например, к экземпляру guest пользовательского класса и экземпляру files модульного класса.

Если атрибуты *class* и *instance* оба пусты, то при применении команды **artexget** к такому профайлу выполняется операция discovery, запускается метод discovery, объявленный в соответствующем файле-каталоге, и в выходном профайле создается параметр для каждого обнаруженного экземпляра параметра. См. пример 1.

Если атрибуты *class* и *instance* оба заданы, то цель полностью определена и параметр применяется только к конкретному экземпляру целевого класса. См. пример 2.

Если атрибуты *class* и *match* оба заданы, то выполняется операция discovery, как описано выше, но будут обнаружены только те целевые экземпляры, имя которых соответствует регулярному выражению из атрибута *match*. См. пример 3.

При написании примера профайла атрибуты *class* и *instance* должны быть оставлены пустыми. Это означает, что при обнаружении пустого целевого класса команда **artexget** просмотрит список экземпляров этого целевого класса (перечень всех пользователей или подсистем системы), прежде чем извлекать значения.

При применении команды **artexset** к необнаруженному целевому классу будет выдано предупреждение:

```
0590-216 Некоторые параметры в профайле требуют обнаружения цели и будут проигнорированы
```

Примеры

1. Примером профайла с целями до обнаружения может служить профайл `chuserProfile.xml`, определяющий домашний каталог пользователя. Этот пример профайла приведен ниже:

```
<Profile version="2.0.0" origin="reference" readOnly="true">
  <Catalog id="chuserParam" version="2.0">
    <Parameter name="home">
      <Target class="" instance=""/>
    </Parameter>
  </Catalog>
</Profile>
```

2. После обнаружения профайл `chuserProfile.xml` будет содержать копию параметра home для каждого обнаруженного пользователя в каждом обнаруженном загружаемом модуле:

```
<Profile version="2.0.0" origin="get">
  <Catalog id="chuserParam" version="2.0">

    <Parameter name="home" value="/">
      <Target class="user" instance="root"/>
      <Target class="module" instance="files"/>
    </Parameter>

    <Parameter name="home" value="/etc">
      <Target class="user" instance="daemon"/>
      <Target class="module" instance="files"/>
    </Parameter>

    ...

  </Catalog>
</Profile>
```

3. Следующий профайл использует атрибут *match* для обнаружения домашнего каталога всех пользователей с именами, начинающимися на *u*, в файловом модуле:

```
<Profile version="2.0.0" origin="reference" readOnly="true"
  <Catalog id="chuserParam" version="2.0">
    <Parameter name="home">
      <Target class="user" match="^u"/>
      <Target class="module" instance="files"/>
    </Parameter>
</Catalog>
</Profile>
```

Связанная информация

Элемент **<Discover>** (в файлах-каталогах).

Запись каталогов AIX Runtime Expert

Файлы-каталоги используются внутри среды AIX Runtime Expert.

Файлы-каталоги содержат определения параметров и информацию о привязках для методов задания конфигурации, описывающих команды извлечения или задания значений параметров. Файлы-каталоги являются локальными для системы, в которой выполняется обычная и тонкая настройка.

Концепции файла-каталога AIX Runtime Expert:

Файлы-каталоги содержат всю необходимую информацию для выполнения операций над параметрами, включая определения, условия использования и методы задания конфигурации. Файлы-каталоги не предназначены для непосредственного применения пользователями; с ними можно работать только через базовую службу AIX Runtime Expert.

Файлы-каталоги устанавливаются в системе одновременно с базовой службой AIX Runtime Expert. Когда новые файлы-каталоги связываются с компонентами или приложениями третьих сторон, установленными в системе, важно проследить за тем, чтобы их уровень соответствовал установленной базовой службе AIX Runtime Expert.

Расположение файла-каталога:

Файлы-каталоги AIX Runtime Expert хранятся в каталоге `/etc/security/artex/catalogs`.

Имя файла-каталога должно в точности совпадать с его атрибутом *id*, к которому в качестве суффикса добавлено расширение `.xml`. Например, у файла-каталога `commandParam.xml` значением атрибута *id* должно быть `commandParam`.

Для того чтобы ссылающийся на него профайл мог его обнаружить, файл-каталог должен носить одно и то же имя в файле XML файла-каталога и в элементе **<Catalog>** файла XML профайла. По умолчанию базовая служба AIX Runtime Expert ищет файлы-каталоги в каталоге по умолчанию `/etc/security/artex/catalogs`. Это поведение можно изменить, только для пользователя root, путем задания переменной среды **ARTEX_CATALOG_PATH**. В этой переменной среды можно указать несколько каталогов через двоеточие.

Процесс файла-каталога:

Процедура записи нового файла-каталога AIX Runtime Expert.

При записи нового файла-каталога AIX Runtime Expert вы должны выполнить следующие действия:

1. Создайте список параметров для включения в файл-каталог.
2. Для каждого параметра создайте элемент **<ParameterDef>**
3. Если несколько параметров используют одну и ту же команду для операции **<Get>**, **<Set>**, **<Discover>** или **<Diff>**:
   - Определите элемент **<CfgMethod>** в начале файла-каталога.
   - С помощью атрибута *cfgmethod* настройте наследование от метода задания конфигурации.
4. Если несколько параметров подчиняются одному и тому же ограничению, определите элемент **<ConstraintDef>** в начале файла-каталога.
5. Для каждого параметра:
   a. Определите операции **<Get type="current">** и **<Get type="nextboot">** для каждого параметра - либо непосредственно под элементом **<ParameterDef>**, либо под опорным элементом **<CfgMethod>**, либо сочетая эти два способа.
   b. Определите все поддерживаемые операции **<Set>** для каждого параметра - либо непосредственно под элементом **<ParameterDef>**, либо под опорным элементом **<CfgMethod>**, либо сочетая эти два способа.
   c. Если параметру требуется цель:
      1) Определите поддерживаемые целевые классы с помощью атрибута *targetClass*
      2) Определите операцию discover - либо непосредственно под элементом **<ParameterDef>**, либо под опорным элементом **<CfgMethod>**, либо сочетая эти два способа. В большинстве случаев метод обнаружения будет определен в методе задания конфигурации.
   d. Если для вступления изменения в силу требуется перезагрузить параметр, добавьте атрибут *reboot* `=true`.
   e. Если параметр подчиняется ограничению, то либо определите элемент **<ConstraintDef>** под элементом **<ParameterDef>**, либо задайте ссылку на это ограничение с помощью атрибута constraint.
6. Для тестирования файла-каталога:
   a. Создайте профайл со всеми параметрами, определенными в файле-каталоге.
   b. С помощью команды **artexget –r** получите значения и протестируйте операции **<Discover>** и **<Get>**.
   c. Применив команду **artexset –c –F –R –l all** к полученному профайлу, протестируйте операции **<Set>** и **<Diff>**.
   d. Кроме того, к этим двум командам можно добавить флаги **–g 3 –g COMMANDS**, чтобы получить дополнительную информацию о командной строке, генерируемой для выполнения запрошенной операции.

Связанная информация

См. раздел о корневом элементе **<Catalog>**.

Элементы файла-каталога AIX Runtime Expert:

Элемент <Catalog>:

Элемент **<Catalog>** - это корневой элемент для всех файлов-каталогов.

Синтаксис

Поддерживаются следующие атрибуты:

Таблица 15. Атрибуты

Атрибут	Обязательный	Тип	Описание
id	да	строка	Задает имя файла-каталога. Это имя должно быть уникально в системе.
version	нет	строка	Задает номер версии файла-каталога.
date	нет	дата-время	Задает дату создания. Формат - *ГГГГ-ММ-ДДТчч:мм:сс*.
priority	нет	целый	Задает порядок выполнения файла-каталога относительно других в методах set. Значение по умолчанию - 0.
inherit	нет	строка	Задает имя каталога для наследования.

Поддерживаются следующие дочерние элементы. Столбец *number* определяет разрешенное количество вхождений дочернего элемента:

Таблица 16. Дочерние элементы

Дочерний элемент	Обязательный	Номер	Описание
<ShortDescription>	нет	0 – 1	Краткое текстовое описание файла-каталога.
<Description>	нет	0 – 1	Длинное текстовое описание файла-каталога.
<SubCat>	нет	0 – любое	Подкатегория
<ParameterDef>	нет	0 – любое	Содержит свойства параметра.
<ConstraintDef>	нет	0 – любое	Определение ограничений на параметры (условия и прерывающие команды).
<CfgMethod>	нет	0 – любое	Определение метода задания конфигурации
<PrereqDef>	нет	0 – любое	Определяет предварительное условие.
<PropertyDef>	нет	0 – любое	Определяет свойство.
<SeedDef>	нет	0 – любое	Определяет начальное значение.

Атрибуты

Таблица 17. Атрибуты

Атрибут	Описание
id	Атрибут *id* (идентификатор) должен совпадать с именем файла-каталога без расширения `.xml`. ИД каталога указывается в профайлах посредством элемента **<Catalog>**.
Priority	Атрибут *priority* (приоритет) применяется, когда методы set конкретного файла-каталога необходимо выполнить до или после методов set других файлов-каталогов, включенных в тот же профайл (например, составной профайл **default.xml**). По умолчанию для файла-каталога задан приоритет 0. Если у двух файлов-каталогов одинаковый приоритет, то их методы set выполняются в неопределенном порядке. Если у файла-каталога положительный приоритет, то его методы set выполняются до методов других файлов-каталогов, в порядке убывания приоритета. Если у файла-каталога отрицательный приоритет, то его методы set выполняются после методов других файлов-каталогов, в порядке убывания приоритета.
Version	Атрибут *Version* (версия) присутствует и в профайлах, и в файлах-каталогах. Версия позволяет определить, совместимы ли профайлы и каталоги с базовой службой AIX Runtime Expert и друг с другом. Дополнительная информация приведена в описании атрибута *Version*.
Date	Атрибут *date* (дата) в настоящее время не используется для элемента **<Catalog>**. Он включен для использования в будущем и возможности обслуживания.
inherit	Атрибут *inherit* указывает имя каталога для наследования без расширения .xml. Все элементы, определенные в унаследованном каталоге, доступны в главном каталоге, как если бы они были определены локально.

Пример

Ниже приведен пример файла-каталога с атрибутом *priority*. Файл-каталог `aixpertParam.xml` задает опции защиты и должен задаваться после всех остальных файлов-каталогов. В этом случае его приоритет будет равен большому по абсолютной величине отрицательному значению.

```
<Catalog id="aixpertParam" version="2.0" priority="-1000">
```

Связанная информация

Элемент <ConstraintDef>.

Элемент <CfgMethod>.

Элементы <Description> и <ShortDescription>.

Элемент <ParameterDef>.

Элемент <SubCat>.

Атрибут версии:

Синтаксис

Версия файла-каталога записывается как атрибут в формате *ММ.мм*, где *ММ* называется основным, а *мм* - дополнительным номером.

```
<Catalog id="commandParam" version="2.0">
```

Основной номер версии

Основной номер версии одинаков для всех установленных в системе файлов-каталогов AIX Runtime Expert и всей среды AIX Runtime Expert, в которой он применяется. Этот основной номер увеличивается при каждом значимом изменении схемы XML для профайлов и файлов-каталогов.

При создании нового файла-каталога задайте основной номер версии равным текущему номеру версии базовой службы AIX Runtime Expert; его можно определить, заглянув в любой из стандартных файлов-каталогов, поставляемых вместе с набором файлов `artex.base.rte`.

Если команда **artexget** применяется к профайлу, у которого основной номер версии отличен от указанного в базовой службе AIX Runtime Expert, то происходит сбой команды с выдачей следующего сообщения об ошибке:

```
0590-117 Ошибка версии
Этот профайл был создан в версии, не поддерживаемой ARTEX
```

Кроме того, для обеспечения совместимости рекомендуется использовать один и тот же основной номер версии для профайла и файла-каталога. Профайл ссылается на файлы-каталоги по конкретному номеру версии. Если основной номер версии профайла иной, нежели у файла-каталога, то любая команда AIX Runtime Expert выдаст предупреждение о том, что результаты могут быть непредвиденными:

```
0590-218 Версия файла-каталога отличается от
той, что указана в профайле
```

Дополнительный номер версии

Дополнительный номер версии уникален в каждом файле-каталоге и увеличивается всякий раз, когда значимое изменение файла-каталога делает его несовместимым с предыдущей версией. Профайл ссылается на файлы-каталоги по конкретному номеру версии. Если дополнительный номер версии профайла иной, нежели у файла-каталога, то любая команда AIX Runtime Expert выдаст предупреждение о том, что результаты могут быть непредвиденными:

```
0590-218 Версия файла-каталога отличается от
той, что указана в профайле
```

При создании нового примера профайла или каталога задайте дополнительный номер версии равным 0.

Элементы <Description> и <ShortDescription>:

Описания - это необязательные информационные текстовые поля, которые можно добавлять в различные элементы файлов-каталогов. Эти поля необязательны, но загрузчикам файлов-каталогов рекомендуется их использовать для документирования родительского элемента.

Синтаксис

У элемента **<ShortDescription>** может быть один из следующих родительских элементов:

- **<Catalog>**
- **<SubCat>**

У элемента **<Description>** может быть один из следующих родительских элементов:

- **<Catalog>**
- **<SubCat>**
- **<ParameterDef>**
- **<ConstraintDef>**

Содержимое элементов **<Description>** и **<ShortDescription>** - это либо простая строка, либо переведенное сообщение, определяемое одним из элементов **<NLSCatalog>**, **<NLSSmitHelp>** или **<NLSCommand>**. Дополнительная информация приведена в разделе Поддержка глобализации.

Формат

В настоящее время извлекается и отображается командой **artexget** с флагом **-i**. Рекомендуется обеспечивать глобализацию текста, вставляемого в эти поля описания.

Поля описания прочих элементов в настоящее время не используются средой AIX Runtime Expert, но их следует предоставлять для использования в будущем и документирования.

Пример

1. Ниже приведен простой пример полей описания:

```
<ShortDescription>
   chuser parameters
</ShortDescription>
<Description>
   Parameter definition for the chuser command
</Description>
```

2. Тот же пример, с использованием переведенных сообщений из файла сообщений `artexcat.cat`:

```
<ShortDescription>
<NLSCatalog catalog="artexcat.cat" setNum="12" msgNum="1">
  параметры chuser
</NLSCatalog>
</ShortDescription>
<Description>
<NLSCatalog catalog="artexcat.cat" setNum="12" msgNum="2">
  Определение параметра команды chuser
</NLSCatalog>
</Description>
```

Связанная информация

Поддержка глобализации

Поддержка глобализации:

В этом разделе указано, как реализована глобализация в описательных полях файлов-каталогов AIX Runtime Expert.

Синтаксис

Родительский элемент: **<Description>**, **<ShortDescription>**

Родительский элемент может содержать один (и только один) из следующих дочерних элементов:

Таблица 18. Дочерние элементы

Дочерний элемент	Обязательный	Количество	Описание
<NLSCatalog>	нет	0 – 1	Строка, включенная в файл-каталог сообщений
<NLSSmitHelp>	нет	0 – 1	Строка, включенная в HTML-файл справки по SMIT
<NLSCommand>	нет	0 – 1	Строка, выданная командой AIX

Файл-каталог NLS

Формат глобализации файла-каталога NLS применяется, когда отображаемое локализованное сообщение включается в существующий файл-каталог сообщений в формате **catgets()**.

Элемент **<NLSCatalog>** содержит следующие атрибуты:

Таблица 19. Атрибуты

Атрибут	Обязательный	Тип	Описание
catalog	да	строка	Имя файла-каталога, содержащего сообщение
setNum	да	целый	Номер набора сообщений, содержащего сообщение
msgNum	да	целый	Номер сообщения в наборе сообщений

Если локализованный файл-каталог сообщений не существует, то отображается сообщение по умолчанию. Сообщение по умолчанию включается (необязательно) в содержимое элемента **<NLSCatalog>**. Рекомендуется предоставлять сообщение по умолчанию.

Справка по SMIT NLS

Формат глобализации справки по SMIT NLS применяется, когда отображаемое локализованное сообщение уже существует в HTML-файле справки по SMIT.

Элемент **<NLSSmitHelp>** содержит следующий атрибут:

Таблица 20. Атрибуты

Атрибут	Обязательный	Тип	Описание
msgId	да	целый	Поле **help_msg_id** из раздела SMIT

Если локализованный файл справки не существует, то отображается сообщение по умолчанию. Сообщение по умолчанию включается (необязательно) в содержимое элемента **<NLSSmitHelp>**. Рекомендуется предоставлять сообщение по умолчанию.

Команда NLS

Формат глобализации команды NLS применяется, когда отображаемое локализованное сообщение выдается командой AIX. Так происходит в случае всех команд тонкой настройки (например, **no**, **vmo**), содержащих флаг *–h* для отображения справочного текста по конкретному параметру.

Элемент **<NLSCommand>** содержит следующий атрибут:

Таблица 21. Атрибут

Атрибут	Обязательный	Тип	Описание
команда	команда	строка	Выражение оболочки для выполнения

Примеры

1. Пример элемента **<NLSCatalog>** из файла-каталога `chssysParam.xml` AIX Runtime Expert, включая сообщение по умолчанию.

```
<Description>
  <NLSCatalog catalog="artexcat.cat" setNum="10" msgNum="2">
    Изменяет определение подсистемы в объектном классе
подсистем.
  </NLSCatalog>
</Description>
```

2. Пример элемента **<NLSSmitHelp>**:

```
<Description>
  <NLSSmitHelp msgId="055136"/>
</Description>
```

3. Пример элемента **<NLSCommand>** из файла-каталога `schedoParam.xml`:

```
<Description>
  <NLSCommand command="/usr/sbin/schedo -h maxspin | /usr/bin/tail -n +2"/>
</Description>
```

Элемент <SubCat>:

Подкатегории, необязательные параметры и подмножества внутри файла-каталога можно указывать с помощью элемента **<SubCat>** в файле-каталоге.

Синтаксис

Родительский элемент: **<Catalog>**,**<SubCat>**

Поддерживаются следующие атрибуты:

Таблица 22. Атрибуты

Атрибут	Обязательный	Тип	Описание
id	да	строка	Задает имя подкатегории файла-каталога. Это имя должно быть уникально в файле-каталоге.

Поддерживаются следующие дочерние элементы:

Таблица 23. Дочерние элементы

Дочерний элемент	Обязательный	Описание
<ShortDescription>	нет	Краткое текстовое описание подкатегории.
<Description>	нет	Длинное текстовое описание подкатегории.
<SubCat>	нет	Вложенная подкатегория. Этот элемент может встречаться несколько раз.
<ParameterDef>	нет	Содержит свойства параметра. Этот элемент может встречаться несколько раз.

Атрибут

Подкатегория является локальной в файле-каталоге:

- ИД подкатегории является уникальным внутри файла-каталога.
- Несколько файлов-каталогов могут использовать один и тот же идентификатор подкатегории.

Подкатегории, определенные в файле-каталоге, должны в точности совпадать с подкатегориями, указанными в связанном примере профайла.

Связанная информация

Элементы <Description> и <ShortDescription>.

Элемент <SubCat>.

Элемент <ParameterDef>.

Элемент <ParameterDef>:

AIX Runtime Expert определяются в файле-каталоге посредством элемента **<ParameterDef>**.

Синтаксис

Родительский элемент: **<Catalog>**,**<ParameterDef>**

Поддерживаются следующие атрибуты:

Таблица 24. Атрибуты

Атрибут	Обязательный	Тип	Описание
name	да	строка	Задает имя параметра. Это имя должно быть уникальным в файле-каталоге.
type	да	строка	Задает тип параметра, применительно к базовой службе.
targetClass	нет	строка	Задает целевые классы параметра, если они есть.
reboot	нет	булевский	True означает, что перезагрузка обязательна. Значение по умолчанию - false.
cfgmethod	нет	строка	Задает *id* (идентификатор) метода задания конфигурации, определенного на уровне **<Catalog>**, который содержит методы для этого параметра.
constraint	нет	строка	Задает идентификатор ограничения, определенного на уровне элемента <Catalog> для текущего файла-каталога.
priority	нет	целый	Ранг выполнения этого параметра в методе set относительно других параметров в этом файле-каталоге. Значение по умолчанию - 0.

Поддерживаются следующие дочерние элементы:

Таблица 25. Дочерние элементы

Дочерний элемент	Обязательный	Описание
<Description>	нет	Текстовое описание параметра.
<ConstraintDef>	нет	Определение ограничения на параметр (прерывающие команды).
<Get>	нет	Определение метода задания конфигурации для операции get. Этот элемент может встречаться несколько раз.
<Set>	нет	Определение метода задания конфигурации для операции set. Этот элемент может встречаться несколько раз.
<Diff>	нет	Определение метода задания конфигурации для операции **diff**.
<Discover>	нет	Определение метода задания конфигурации для поиска цели.

Атрибут

Таблица 26. Атрибуты

Атрибут	Описание
name	Атрибут name однозначно идентифицирует параметр в файле-каталоге. Дополнительная информация приведена в описании атрибута name параметра.
type	Обязательный атрибут type указывает тип значения параметра. Поддерживаются следующие значения: • строка - для алфавитно-цифровых строк; • целый - для числовых значений; • целый-bi - для числовых значений с необязательным строчным или прописным суффиксом K, M, G, T, P или E, обозначающим соответственно “кило”, “мега”, “гига”, “тера”, “пета” и “экса”. Эти суффиксы интерпретируются как степени 1024; • целый-si - для числовых значений с необязательным суффиксом SI. То же, что и тип целый-bi, за исключением того, что суффиксы интерпретируются как степени 1000. • булевский - для булевских значений. Поддерживаются значения 0 и 1. • двоичный - для двоичных значений, представленных в виде 64-значных строк в профайлах.

reboot	Значение по умолчанию для булевского атрибута ‘reboot’ - это “false”. Если для вступления изменения параметра в силу необходимо выполнить перезагрузку, то атрибуту ‘reboot’ этого параметра нужно присвоить значение “true”.
	
	Сам AIX Runtime Expert никогда не перезагружает системы. По умолчанию команда **artexset** не будет принудительно задавать параметры перезагрузки. Если профайл содержит параметры перезагрузки, то команда не будет выполнена:
	`0590-502: профайл содержит параметры, требующие перезагрузки. Профайл не задан. С помощью флага -l all выполните принудительное задание всех параметров`
	
	Команда **artexset**, вызванная с соответствующим флагом *–l*, задает значение и предупреждает пользователей о том, что для вступления изменений в силу необходима перезагрузка:
	`0590-206 Для вступления изменений в силу необходимо вручную выполнить операцию POST Перезагрузите систему`

priority	По умолчанию команда **artexset** задает параметры в произвольном порядке. С помощью атрибута *priority* можно изменить это поведение и принудительно задать параметр до или после других параметров. Приоритет по умолчанию - 0. С помощью атрибута priority можно изменить это значение на любое целое от -2147483648 до 2147483647. Чем выше приоритет параметра, тем раньше он обрабатывается. Параметры с одинаковым приоритетом обрабатываются в произвольном порядке.
targetClass	Некоторые параметры необходимо связать с целью, как объяснено в разделе о “целевом элементе” профайла. В таких параметрах атрибут *targetClass* должен быть задан равным списку поддерживаемых целевых классов через запятую.
cfgmethod	Элемент **<ParameterDef>** может унаследовать элементы командной строки из элемента **<CfgMethod>**, сославшись на атрибут *id* этого метода задания конфигурации в атрибуте *cfgmethod*. Дополнительная информация о методах задания конфигурации приведена в разделе об элементе <CfgMethod>.

Таблица 26. Атрибуты (продолжение)

Атрибут	Описание
constraint	Элемент **<ParameterDef>** может использовать атрибут constraint, чтобы ссылаться на атрибут id элемента **<ConstraintDef>**, указывая, что параметр подчиняется ограничению. Дополнительная информация об ограничениях приведена в разделе об “элементе **<ConstraintDef>**”.

Примеры

1. Ниже приведен пример определения параметра с альтернативным целым типом: *kernel_heap_size*, из файла-каталога `vmoParam.xml`:

   ```
   <ParameterDef name="kernel_heap_psize" type="integer-bi">
   ```

   При извлечении значения этого параметра посредством команды **artexget** результат будет примерно следующим (за исключением итогового профайла):

   ```
   <Parameter name="kernel_heap_psize" value="16M"... />
   ```

   Значение параметра будет интерпретировано по-разному в зависимости от типа:

   - Так как он объявлен как относящийся к типу целый-bi, его значением будет 16M= 16 777 216.
   - Если бы был указан тип целый-si, то значение равнялось бы “16M”=16 000 000.
2. Пример двоичного параметра: база данных сигнатуры защиты `tsd.dat` в файле-каталоге `tsdParam.xml`:

   ```
   <ParameterDef name="tsdatabase" type="binary">
   ```
3. Пример параметра с атрибутом *reboot*. Параметр типа дампа в файле-каталоге `sysdumpdevParam.xml`:

   ```
   <ParameterDef name="type_of_dump" type="string" reboot="true">
   ```
4. Пример параметра с одним целевым классом: параметр *addr* из файла-каталога `mktcpipParam.xml` применяется к конкретному сетевому интерфейсу:

   ```
   <ParameterDef name="addr" type="string" cfgmethod="mktcpip" targetClass="interface">
   ```
5. Пример параметра с несколькими целевыми классами: параметр спецификации имен из файла `coreParam.xml` применяется к конкретному пользователю (root, admin, guest и т.п.) в конкретном реестре (файловом, LDAP).

   ```
   <ParameterDef name="namingspecification" type="string" reboot="true" targetClass="user,registry"
   cfgmethod="coremgt">
   ```
6. Пример применения атрибута *cfgmethod*: для операции **<Get type="current">** параметр fixed из файла-каталога `chlicenseParam.xml` наследует элемент **<Command>** из метода задания конфигурации под названием *chlicense*, а также локально определяет собственные элементы **<Filter>** и **<Mask>** для той же операции:

   ```
   <CfgMethod id="chlicense">
     <Get type="current">
       <Command>lslicense -c -A</Command>
     </Get>
   </CfgMethod>
   <ParameterDef name="fixed" cfgmethod="chlicense" type="integer">
     <Get type="current">
       <Filter>tail -n 1 | cut -d: -f3</Filter>
       <Mask value="1">(.*)</Mask>
     </Get>
   </ParameterDef>
   ```
7. Пример применения атрибута constraint: параметр authorizations из файла-каталога `authParam.xml` подчиняется ограничению **setkst**, ранее определенному в элементе **<ConstraintDef>**:

   ```
   <ParameterDef name="authorizations" cfgmethod="cat" constraint="setkst" type="string">
   ```

Связанная информация

Атрибут *name*.

Атрибут name:

Имя параметра часто диктуется извлекающей или задающей параметр командой.

Имена параметров должны быть уникальными в файле-каталоге. Это гарантирует, что элемент **<Parameter>** в профайле будет связан с уникальным элементом **<ParameterDef>** в файле-каталоге.

- Если команда **get** выдает несколько пар параметр-значение, то для извлечения нескольких параметров из вывода такой команды можно воспользоваться элементом **<Mask>**. Это возможно только в том случае, если имя параметра совпадает с именем, указанным в выводе команды **get**.
- Если команда **set** принимает несколько пар параметр-значение, то с помощью последовательностей %n и %v1 в элементе **<Argument>** можно задать несколько параметров в одной команде. Это возможно только в том случае, если имя параметра совпадает с именем, указанным в команде **set**.

Примеры

1. Пример: команда **raso –a**, используемая в файле-каталоге `rasoParam.xml`, выдает по одному параметру в строке:

   ```
   kern_heap_noexec = 0
   kernel_noexec = 1
   mbuf_heap_noexec = 0
   mtrc_commonbufsize = 485
   ```

   В этом простом случае имена параметров - это *kernel_heap_noexec*, *kernel_noexec* и т.п.
2. Пример: команда, применяемая в методе задания конфигурации **get** из файла-каталога `acctctlParam.xml`, выдает результат, более трудный для синтаксического анализа. Не только имя параметра встроено в неотформатированную последовательность, но и имена и значения параметров локализованы. Методы задания конфигурации get должны будут запускать эту команду во время задания переменной среды **LANG=C** и в каждой строке заменять ключевые слова подходящими именами параметров:

   ```
   Расширенный учет не запущен.
   Уведомление по электронной почте отключено.
   Текущий адрес электронной почты не задан.
   Значение параметра Восстанавливать время учета CPU в турборежиме - false.
   ```

   В приведенном выше примере выбранные имена переменных - это `accounting`, `email`, `email_adddr` и `turacct`.

Связанная информация

- Элемент **<Parameter>**
- Элемент **<Mask>**
- Развертывание элементов командной строки

Элемент <ConstraintDef>:

Синтаксис

Родительский элемент: **<Catalog>**, **<ParameterDef>**

Поддерживаются следующие атрибуты:

Таблица 27. Атрибуты

Атрибут	Обязательный	Тип	Описание
id	нет*	строка	Задает имя ограничения на параметр.

*Этот атрибут должен быть задан для элементов **<Constraint>**, определенных на уровне файла-каталога.

Поддерживаются следующие дочерние элементы:

Таблица 28. Дочерние элементы

Дочерние элементы	Обязательный	Описание
<Description>	нет	Текстовое описание прерывающей команды.
<PreOp>	нет	Прерывающие операции, выполняемые до задания значения параметра.
<PostOp>	нет	Прерывающие операции, выполняемые после задания значения параметра.
<BuiltIn>	нет	Встроенная прерывающая операция. Этот элемент может встречаться несколько раз.

Формат

Для того чтобы измененные значения некоторых параметров конфигурации и тонкой настройки вступили в силу, может потребоваться выполнить прерывающие операции. Прерывающей называется любая операция, которая может временно прервать доступ к службе или устройству. Типичными примерами прерывающих операций могут служить перезапуск демона, размонтирование или монтирование файловой системы, включение или отключение карты сетевого адаптера. Программа AIX Runtime Expert применяет ограничения с целью продемонстрировать, что для вступления изменений параметра в силу требуется выполнить прерывающую операцию. Для определения такого ограничения используется элемент **<ConstraintDef>**.

Ограничение можно определить любым из следующих способов:

- Внутри элемента **<ParameterDef>**, если ограничение применяется только к одному параметру.
- На уровне файла-каталога у элемента **<ConstraintDef>** должен быть атрибут *id*, чтобы на ограничение можно было впоследствии ссылаться в элементах **<ParameterDef>**.

Встроенное ограничение

Элемент **<BuiltIn>** не содержит ни атрибутов, ни дочерних элементов.

Встроенное ограничение определяет операции, жестко закодированные в базовой службе. В настоящее время определено только одно встроенное ограничение: *bosboot*. Разница между встроенными ограничениями и другими прерывающими операциями заключается в том, что команда *bosboot* никогда не выполняется программой AIX Runtime Expert. Базовая служба лишь предупредит, что для вступления изменений в силу необходимо выполнить *bosboot*.

```
0590-206 Для вступления изменений в силу необходимо вручную выполнить операцию POST
Выполните bosboot
```

Ограничение PreOp и PostOp

Элемент **<PreOp>** определяет обязательные команды (выражения оболочки), которые должны быть выполнены до того, как значение параметра будет задано методом задания конфигурации. Элемент **<PostOp>** определяет обязательные команды, которые должны быть выполнены после выполнения метода задания конфигурации.

Элемент **<ConstraintDef>** должен содержать не более одного дочернего элемента **<PreOp>** и не более одного дочернего элемента **<PostOp>**.

Примеры

1. Пример встроенного ограничения (на уровне файла-каталога)

```
<ConstraintDef id="bosboot">
  <Description>
<NLSCatalog catalog="artexcat.cat" setNum="51" msgNum="3">
    bosboot
</NLSCatalog>
  </Description>
  <Built>Inbosboot</BuiltIn>
</ConstraintDef>
```

2. Пример ограничения **<PreOp>**: ограничение *clic* в файле-каталоге `trustchkParam.xml`. Заметьте, что в этом примере команда **preop** не выполняет никаких операций и лишь проверяет наличие расширения ядра, необходимого команде **set**. Если расширение ядра не установлено, то ограничение, определенное в элементе **<PreOp>**, будет нарушено и команда **set** не будет выполнена:

```
<ConstraintDef id="clic">
  <Description>
  <NLSCatalog catalog="artexcat.cat" setNum="48" msgNum="3">
     Проверить, установлено ли расширение ядра clic.rte.
  </NLSCatalog>
  </Description>
  <Pre>Oplslpp -l "clic*"</PreOp>
 </ConstraintDef>
```

3. Пример ограничения **<PostOp>**: ограничение set Kernel Security Tables в элементе `authParam.xml`. Модифицированные базы данных необходимо загружать в ядро лишь однажды, после внесения всех модификаций.

```
<ConstraintDef id="setkst">
  <Description>
<NLSCatalog catalog="artexcat.cat" setNum="5" msgNum="3">
Отправить базу данных прав доступа в KST (Kernel Security Tables)
    </NLSCatalog></Description>
  <PostOp>/usr/sbin/setkst -t auth &gt;/dev/null</PostOp>
  </ConstraintDef>
```

Элемент <CfgMethod>:

Синтаксис

Родительский элемент: **<Catalog>**

Поддерживается следующий атрибут:

Таблица 29. Атрибут

Атрибут	Обязательный	Тип	Описание
id	да	строка	Задает имя метода задания конфигурации.

Поддерживаются следующие дочерние элементы:

Таблица 30. Дочерние элементы

Дочерние элементы	Обязательный	Номер	Описание
<Get>	нет	0 – 1	Определение метода задания конфигурации для операции get. Этот элемент может встречаться несколько раз.
<Set>	нет	0 – 1	Определение метода задания конфигурации для операции set. Этот элемент может встречаться несколько раз.
<Diff>	нет	0 – 1	Определение метода задания конфигурации для операции **diff**.
<Discover>	нет	0 – 1	Определение метода задания конфигурации для поиска цели.
<Property>	нет	0 – любое	Присваивает свойство параметрам с помощью метода конфигурации.

Формат

Элемент **<CfgMethod>** определяет метод задания конфигурации, на который впоследствии можно будет ссылаться в параметре посредством атрибута *cfgmethod* элемента **<ParameterDef>**. Затем параметр наследует все элементы, определенные в опорном методе задания конфигурации.

В зависимости от параметра, применение конфигурации может дать некоторые преимущества по сравнению с локальным определением:

- Оно упрощает файл-каталог, позволяя избежать дублирования одних и тех же элементов командной строки для нескольких параметров.
- Оно позволяет обрабатывать несколько параметров одной командой.

Пример

Файл-каталог `vmoParam.xml` определяет большое количество параметров, использующих один и тот же метод задания конфигурации. Ниже приведена упрощенная версия этого файла-каталога:

```
<Catalog id="vmoParam" version="2.1">
 <CfgMethod id="vmo">
    <Get type="current">
      <Command>/usr/sbin/vmo -a</Command>
      <Mask name="1" value="2">[[:space:]]*(.*) = (.*)</Mask>
    </Get>

    <Get type="nextboot">
      <Command>/usr/sbin/vmo -r -a</Command>
      <Mask name="1" value="2">[[:space:]]*(.*) = (.*)</Mask>
    </Get>

    <Set type="permanent">
      <Command>/usr/sbin/vmo -p%a</Command>
   <Argument>%n=%v1</Argument>
    </Set>

    <Set type="nextboot">
      <Command>/usr/sbin/vmo -r%a</Command>
      <Argument>%n=%v1</Argument>
    </Set>
  </CfgMethod>

  <ParameterDef name="ame_maxfree_mem" cfgmethod="vmo" type="integer" />
```

```
  <ParameterDef name="ame_min_ucpool_size" cfgmethod="vmo" type="integer" />
  <ParameterDef name="ame_minfree_mem" cfgmethod="vmo" type="integer" />
  ...

</Catalog>
```

Связанная информация

- Генерация командной строки
- Элемент **<Get>**
- Элемент **<Set>**

Элемент <Get>:

Синтаксис

Родительский элемент: **<CfgMethod>**, **<ParameterDef>**

Поддерживается следующий атрибут:

Таблица 31. Атрибут

Атрибут	Обязательный	Тип	Описание
type	да	строка	Задает тип команды **get** (*current* или *nextboot*).

Поддерживаются следующие дочерние элементы:

Таблица 32. Дочерние элементы

Дочерние элементы	Обязательный	Номер	Описание
<Command>	нет	0 – 1	Команда
<Argument>	нет	0 – 1	Аргументы командной строки
<Stdin>	нет	0 – 1	Аргументы, поддерживаемые элементом **<Stdin>**
<Filter>	нет	0 – 1	Фильтр
<Mask>	нет	0 – 1	Маска захвата вывода
<Prereq>	нет	0 – любое	Назначает предварительное условие для операции get

Элемент **<Command>** должен быть определен для каждого параметра, либо на уровне **<CfgMethod>**, либо напрямую на уровне **<ParameterDef>**.

Формат

Элемент **<Get>** описывает способ захвата значения конкретного параметра. Его можно указать либо непосредственно под элементом **<ParameterDef>**, либо под элементом **<CfgMethod>**, на который ссылается элемент **<ParameterDef>** через атрибут *cfgmethod*, либо сочетая эти две возможности.

Для каждого параметра должно быть определено два элемента Get, по одному для каждого поддерживаемого значения атрибута *type*:

- Элемент Get **type="current"** указывает метод, который будет запущен для извлечения текущего значения параметра.
- Элемент Get **type="nextboot"** указывает метод, который будет запущен для извлечения значения, которое будет присвоено параметру после очередной перезагрузки системы.
- Запускаемый метод get зависит от выполняемой операции:
  - Если вызывается команда **artexget** с флагом *–r*, то используется метод get *current*.

– Если вызывается команда **artexget** с флагом *–n*, то используется метод get *nextboot*.
– Если вызывается команда **artexget** с флагом *–p*, то выбор метода зависит от входных параметров атрибута *applyType*. Метод get *current* применяется для параметров, у которых атрибут *applyType* равен runtime, а метод get *nextboot* - для параметров, у которых атрибут *applyType* равен reboot.

Связанная информация

Генерация командной строки

Элемент **<Маска>**.

Элемент <Set>:

Элемент **<Set>** определяет, как должна быть скомпонована командная строка для задания значения параметра.

Синтаксис

Родительский элемент: **<CfgMethod>**, **<ParameterDef>**

Поддерживается следующий атрибут:

Таблица 33. Атрибут

Атрибут	Обязательный	Тип	Описание
type	да	строка	Задает тип команды **set** (current или nextboot).

Поддерживаются следующие дочерние элементы:

Таблица 34. Дочерние элементы

Дочерние элементы	Обязательный	Номер	Описание
<Command>	нет	0 – 1	Команда
<Argument>	нет	0 – 1	Аргументы командной строки
<Stdin>	нет	0 – 1	Аргументы Stdin
<Prereq>	нет	0 – любое	Назначает предварительное условие для операции **<Set>**

Примечание: Элемент **<Command>** должен быть определен для каждого параметра, либо на уровне **<CfgMethod>**, либо напрямую на уровне **<ParameterDef>**.

Формат

Существует три типа элементов **<Set>**, которые могут быть определены для каждого параметра. Они идентифицируются по их обязательному атрибуту *type*:

- Set **type="current"** определяет операцию set, изменяющую только значение параметра для текущего сеанса. Любое изменение, вносимое с помощью операции set, будет утеряно после перезагрузки системы.
- Set **type="nextboot"** определяет операцию set, изменяющую только значение, которое будет присвоено параметру после следующей перезагрузки системы. Текущее значение останется без изменений.
- Set **type="permanent"** определяет операцию set, изменяющую и текущее, и присваиваемое после перезагрузки значение параметра.

Тип операции set определяется на основе параметров, заданных на момент запуска команды **artexset**, в зависимости от атрибута *applyType* параметра в профайле. В следующей таблице перечислены методы set,

запускаемые в зависимости от методов set, определенных в файле-каталоге, и атрибута *applyType* для параметра:

Таблица 35. Методы Set - определенные типы методов set и атрибут applyType параметра.

текущий	nextboot	permanent	динамическая	nextboot
0	0	0	не задан (ошибка)	не задан (ошибка)
0	0	1	set permanent	не задан (ошибка)
0	1	0	set nextboot + предупреждение	set nextboot
0	1	1	set permanent	не задан (ошибка)
1	0	0	set current + предупреждение	set nextboot
1	0	1	set permanent	не задан (ошибка)
1	1	0	set current set nextboot	set nextboot
1	1	1	set permanent	set nextboot

Связанная информация

Генерация командной строки.

Элемент <Diff>:

Элемент **<Diff>** определяет, как должна быть скомпонована командная строка для сравнения двух значений параметра.

Синтаксис

Родительский элемент: **<CfgMethod>**, **<ParameterDef>**

Поддерживаются следующие дочерние элементы:

Таблица 36. Дочерние элементы

Дочерние элементы	Обязательный	Описание
<Command>	нет	Команда
<Argument>	нет	Аргументы командной строки
<Stdin>	нет	Аргументы Stdin
<Filter>	нет	Фильтр
<Mask>	нет	Маска захвата вывода

Примечание: Элемент **<Command>** должен быть определен для каждого параметра, либо на уровне **<CfgMethod>**, либо напрямую на уровне **<ParameterDef>**.

Формат

Элемент **<Diff>** обычно необязателен, поскольку среде известно, как сравнить два значения параметра внутренне по их типу (строка, целый, целый-bi, двоичный и т.п.). Однако если внутреннее сравнение не подходит для конкретного параметра, то вместо него можно использовать команду внешнего сравнения.

Пример

Следующий элемент **<Diff>** пригоден для большинства параметров, хотя внутреннее сравнение эффективнее. Элемент **<Diff>** использует команду **diff** для сравнения двух файлов, содержащих два значения:

```
<Diff>
  <Command>/usr/bin/diff %f1 %f2; echo $?</Command>
</Diff>
```

Связанная информация

Генерация командной строки.

Элемент **<Маска>**.

Элемент <Discover>:

Элемент **<Discover>** определяет, как должна быть скомпонована командная строка для обнаружения целей для поддерживающего их параметра.

Синтаксис

Родительский элемент: **<CfgMethod>**, **<ParameterDef>**

Поддерживаются следующие дочерние элементы:

Таблица 37. Дочерние элементы

Дочерние элементы	Обязательный	Номер	Описание
<Command>	нет	0 – 1	Команда
<Prereq>	нет	0 – любое	Назначает предварительное условие для операции discover

Примечание: Элемент **<Command>** должен быть определен для каждого параметра, либо на уровне **<CfgMethod>**, либо напрямую на уровне **<ParameterDef>**.

Формат

Команда discover служит для получения списка целевых экземпляров заданного параметра.

Вывод команды discover для параметра, поддерживающего N целевых классов, представляется в следующем формате:

```
class_1=inst_1_1;class_2=inst_2_1;...;class_N=inst_N_1
class_1=inst_1_2;class_2=inst_2_2;...;
class_N=inst_N_2class_1=inst_1_3;
class_2=inst_2_3;...;class_N=inst_N_3
...
```

Команда **artexget** генерирует и запускает команду discover для параметров, удовлетворяющих одному из следующих критериев:

- Содержат элемент **<Target>** с пустыми атрибутами *class* и *instance*. **<Target class="" instance="" />**
- Содержат хотя бы один элемент **<Target>** с атрибутом *match*: **<Target class="..." match="..." />**

Команда **artexset** дополнительно требует соблюдения следующих двух критериев:

- Команда **artexset** вызывается с флагом *–d*.
- Атрибут *setDiscover* в элементе **<Parameter>** профайла равен true.

Примеры

1. Файл-каталог `mktcpipParam.xml` использует следующую команду discover для получения списка сетевых интерфейсов, определенных в системе:

```
<Discover>
  <Command>
   /usr/sbin/lsdev -C -c if -F "name" | /usr/bin/sed -e 's/^/interface=/'
 </Command>
</Discover>
```

Эта команда выдаст следующий вывод:

```
interface=en0
interface=et0
interface=lo0
```

2. Файл-каталог `chuserParam.xml` использует следующую команду **discover** для получения списка всех пользователей для всех загружаемых идентификационных модулей:

```
<Discover>
  <Command>
  /usr/sbin/lsuser -a registry ALL | /usr/bin/sed -e "s/\(.*\) registry=\(.*\)/module=\2;user=\1/g"
</Command>
</Discover>
```

Эта команда выдаст следующий вывод:

```
module=LDAP;user=daemon
module=LDAP;user=bin
module=LDAP;user=sys
module=LDAP;user=adm
...
module=files;user=root
module=files;user=daemon
module=files;user=bin
module=files;user=sys
module=files;user=adm
...
```

Элемент <Command>:

Элемент **<Command>** определяет базовую команду, применяемую для выполнения операции, определенной родительским элементом.

Синтаксис

Родительский элемент: **<Get>**, **<Set>**, **<Diff>**, **<Discover>**, **<PrereqDef>**, **<Prereq>**, **<PropertyDef>**, **<Property>**, **<Command>**

Формат

Содержимое элемента **<Command>** развертывается, как описано в разделе Развертывание элементов командной строки, и объединяется с другими элементами командной строки, образуя полную командную строку. Дополнительная информация приведена в разделе Генерация командной строки.

Некоторые символы, часто встречающиеся в выражениях оболочки, такие как <, > и &, не разрешены в документах XML. Эти символы следует заменить на соответствующую сущность XML:

Таблица 38. Сущности XML

Символ	Сущность XML
<	&lt;
>	&gt;
&	&amp;

Если выражение содержит много таких символов, то можно воспользоваться разделом CDATA. Разделы CDATA начинаются с **<![CDATA[** и оканчиваются на **]]>**.

Элемент **<Command>** должен быть определен для каждой поддерживаемой операции каждого параметра, либо на уровне **<CfgMethod>**, либо на уровне **<ParameterDef>**.

Пример

Файл-каталог `envParam.xml` определяет параметр под названием profile, представляющий содержимое файла `/etc/profile`. Для этого параметра элемент **<Get>** использует команду **cat** для захвата содержимого файла `/etc/profile`:

```
<ParameterDef name="profile">
  <Get type="current">
    <Command>/usr/bin/cat /etc/environment</Command>
  </Get>
</ParameterDef>
```

Связанная информация

Генерация командной строки

Развертывание элементов командной строки

Элемент <Argument>:

Синтаксис

Родительский элемент: **<Get>**, **<Set>**, **<Diff>**, **<PrereqDef>**, **<Prereq>**, **<PropertyDef>**, **<Property>**

Формат

Содержимое элемента **<Argument>** развертывается, как описано в разделе Развертывание элементов командной строки, и объединяется с элементом **<Command>** или <Stdin>, образуя полную командную строку. Дополнительная информация приведена в разделе Генерация командной строки.

Некоторые символы, часто встречающиеся в выражениях оболочки, такие как <, > и &, не разрешены в документах XML. Эти символы следует заменить на соответствующую сущность XML:

Таблица 39. Сущности XML

Символ	Сущность XML
<	&lt;
>	&gt;
&	&amp;

Если выражение содержит много таких символов, то можно воспользоваться разделом CDATA. Разделы CDATA начинаются с **<![CDATA[** и оканчиваются на **]]>**.

Пример

Файл-каталог `vmoParam.xml` использует элемент **<Argument>** для добавления аргумента в команду **vmo** для каждого параметра **vmo** в профайле:

```
<CfgMethod id="vmo">
  <Set type="permanent">
    <Command>/usr/sbin/vmo -p%a</Command>
    <Argument> -o %n=%v1</Argument>
  </Set>
</CfgMethod>
```

Связанная информация

Генерация командной строки

Развертывание элементов командной строки

Элемент <Stdin>:

Синтаксис

Родительский элемент: **<Get>**, **<Set>**, **<Diff>**, **<PrereqDef>**, **<Prereq>**, **<PropertyDef>**, **<Property>**

Формат

Содержимое элемента **<Stdin>** развертывается, как описано в разделе Развертывание элементов командной строки, и полученные данные записываются в стандартный ввод командной строки, генерируемой для операции, определенной в родительском элементе.

Пример

Файл-каталог `envParam.xml` определяет параметр под названием profile, представляющий содержимое файла `/etc/profile`. Для этого параметра операция set записывает значение параметра в стандартный ввод команды **cat**, чтобы заменить файл `/etc/profile`:

```
<ParameterDef name="profile">
  <Set type="permanent">
    <Command>/usr/bin/cat &gt; /etc/profile</Command>
     <Stdin>%v1</Stdin>
   </Set>
 </Get>
```

Связанная информация

Генерация командной строки

Развертывание элементов командной строки

Элемент <Filter>:

Синтаксис

Родительский элемент: **<Get>**, **<Diff>**, **<PropertyDef>**, **<Property>**

Формат

Содержимое элемента **<Filter>** - это команда, в которую в качестве ввода передается вывод командной строки, сгенерированной для операции, определенной в родительском элементе.

Некоторые символы, часто встречающиеся в выражениях оболочки, такие как <, > и &, не разрешены в документах XML. Эти символы следует заменить на соответствующую сущность XML:

Таблица 40. Сущности XML

Символ	Сущность XML
<	&lt;
>	&gt;
&	&amp;

Если выражение содержит много таких символов, то можно воспользоваться разделом CDATA. Разделы CDATA начинаются с **<![CDATA[** и оканчиваются на **]]>**.

Пример

Файл-каталог `nfsParam.xml` использует элемент **<Filter>** операции get параметра *v4_root_node* для извлечения корневого узла из вывода команды **nfds –getnode**:

```
<ParameterDef id="v4_root_node">
  <Get type="current">
    <Command>
       /usr/sbin/nfsd -getnodes
    </Command>
    <Filter>
       /usr/bin/awk -F: 'NR == 2 { printf("%s", $1) }'
    </Filter>
  </Get>
</ParameterDef>
```

Связанная информация

Генерация командной строки

Элемент <Mask>:

Синтаксис

Родительский элемент: **<Get>**, **<Diff>**, **<Discover>** (только под <**SeedDef**>),**<PropertyDef>**, **<Property>**

Следующие атрибуты поддерживаются при использовании в элементе **<Get>** или **<Diff>**:

Таблица 41. Атрибуты

Атрибут	Обязательный	Тип	Описание
name	нет	целый	Указывает индекс подвыражения, который соответствует имени параметра. Допустимы значения 1 и 2.
value	нет	целый	Указывает индекс подвыражения, который соответствует значению параметра. Допустимы значения 1 и 2.

Следующие атрибуты поддерживаются при использовании в подэлементе **<Discover>** элемента **<SeedDef>**:

Таблица 42. Атрибуты

Атрибут	Обязательный	Тип	Описание
catalog	yes	целый	Указывает индекс подвыражения, который соответствует имени каталога. Допустимые значения: 1, 2 и 3.
name	yes	целый	Указывает индекс подвыражения, который соответствует имени параметра. Допустимые значения: 1, 2 и 3.
target	нет	целый	Указывает индекс подвыражения, который соответствует назначению параметра. Допустимые значения: 1, 2 и 3.

Следующий атрибут поддерживается при использовании в элементе **<PropertyDef>** или **<Property>**:

Таблица 43. Атрибут

Атрибут	Обязательный	Тип	Описание
value	нет	целый	Указывает индекс подвыражения, который соответствует имени параметра. Должен иметь значение “1”, если указан.

Формат

Элемент **<Mask>** определяет регулярное выражение, которое применяется к каждой строке выходной команды для извлечения данных из этих строк. То, какие данные будут извлечены, зависит от того, где используется элемент **<Mask>**.

Если никакие атрибуты не указаны, то для извлечения данных используется последняя строка выходной команды, которая соответствует регулярному выражению. Извлеченные данные - это часть строки, соответствующая регулярному выражению. При использовании в элементе **<Get>** или **<Diff>** извлеченные данные применяются в качестве значения параметра. При использовании в элементе **<PropertyDef>** или **<Property>** извлеченные данные применяются в качестве значения свойства.

Если указан только атрибут *value*, для него должно быть установлено значение 1, и регулярное выражение должно содержать только одно подвыражение. Для извлечения данных используется последняя строка выходной команды, которая соответствует регулярному выражению. Извлеченные данные - это часть строки, соответствующая первому (и только ему) подвыражению. При использовании в элементе **<Get>** или **<Diff>** извлеченные данные применяются в качестве значения параметра. При использовании в элементе **<PropertyDef>** или **<Property>** извлеченные данные применяются в качестве значения свойства.

Если заданы атрибуты *name* и *value*, для одного из них должно быть установлено значение 1, а для другого значение 2, и регулярное выражение должно содержать два подвыражения. Атрибуты *name* и *value* извлекаются из каждой строки выходной команды, которая соответствует регулярному выражению. При использовании в элементе **<Get>** name используется в качестве имени параметра, а value используется в качестве значения параметра. При использовании в элементе **<Diff>** name используется в качестве имени параметра, а value используется как результат сравнения. Благодаря этой функциональности можно извлекать значения нескольких параметров из одной команды **get** и сравнивать несколько параметров в одной команде **diff**.

При использовании в подэлементе **<Discover>** элемента **<SeedDef>** должны быть указаны атрибуты каталога и имени. Имя каталога и имя параметра извлекаются из каждой строки выходной команды, которая

соответствует регулярному выражению. Если в системе найден каталог, соответствующий извлеченному имени каталога, и если он содержит определение для параметра, которое соответствует извлеченному имени параметра, то параметр вставляется в профайл. Может быть добавлен необязательный целевой аргумент для извлечения целевого определения для каждого обнаруженного параметра. Целевое определение должно следовать за списком через точку с запятой пар ⌂class=instance⌂, например, class1=instance1;class2=instance2;....

Примеры

1. Файл-каталог vmoParam.xml использует элемент **<Mask>** с атрибутами *name* и *value* для извлечения всех значений параметров из одной команды **vmo –a**:

```
<CfgMethod id="vmo">
  <Get type="current">
    <Command>/usr/sbin/vmo -a</Command>
    <Mask name="1" value="2">[[:space:]]*(.*) = (.*)</Mask>
  </Get>
</CfgMethod>
```

2. Если бы файл-каталог vmoParam.xml был написан таким образом, что содержал по одной отдельной команде для каждого извлекаемого значения параметра, то элемент **<Mask>** можно было бы использовать лишь с заданным атрибутом *value*, без атрибута *name*:

```
<CfgMethod id="vmo">
  <Get type="current">
    <Command>/usr/sbin/vmo -o %n</Command>
    <Mask value="1"> = (.*)</Mask>
  </Get>
</CfgMethod>
```

3. Или используя регулярное выражение, соответствующее лишь значению:

```
<CfgMethod id="vmo">
  <Get type="current">
    <Command>/usr/sbin/vmo -o %n</Command>
    <Mask>[^ ]*$</Mask>
  </Get>
</CfgMethod>
```

   Из трех приведенных выше примеров первый наиболее эффективен, так как только он обходится одной командой для захвата всех параметров команды **vmo**. Примеры 2 и 3 создадут отдельную команду для каждого параметра команды **vmo**, поскольку имя параметра используется в элементе **<Command>**.

4. Следующий элемент **<SeedDef>** определяет начальное значение, которое может быть использовано для обнаружения всех атрибутов всех устройств. Он использует назначение для обозначения устройства, с которым он работает:

```
<SeedDef name="devAttr">
  <Discover>
    <Command>
      /usr/sbin/lsdev -F 'name class subclass type' |
      while read DEV CLASS SUBCLASS TYPE
      do
        /usr/sbin/lsattr -F attribute -l $DEV |
        while read PAR
        do
          echo device=$DEV devParam.$CLASS.$SUBCLASS.$TYPE $PAR
        done
      done
    </Command>
    <Mask target="1" catalog="2" name="3">(.*) (.*) (.*) <Mark>
  </Discover>
</SeedDef>
```

Команда discovery распечатывает каждый обнаруженный атрибут устройства в отдельной строке в следующем формате:

```
device=DeviceName devParam.Class.Subclass.Type AttributeName
```

Пример:

```
device=en0 devParam.if.EN.en tcp_recvspace
device=en0 devParam.if.EN.en tcp_sendspace
device=ent0 devParam.adapter.vdevice.IBM,l-lan alt_addr
device=ent0 devParam.adapter.vdevice.IBM,l-lan chksum_offload
```

Связанная информация

Генерация командной строки

Элемент <SeedDef>:

Элемент **<SeedDef>** определяет начальное значение, которое может быть использовано в элементе **<Seed>**.

Синтаксис

Родительский элемент: **<Catalog>**

Поддерживается следующий атрибут:

Таблица 44. Атрибут

Атрибут	Обязательный	Тип	Описание
name	yes	строка	Задает имя начального значения. Это имя должно быть уникальным для каждого каталога.

Поддерживается следующий дочерний элемент:

Таблица 45. Дочерний элемент

Дочерний элемент	Обязательный	Описание
<Discover>	yes	Задает команду, которая используется для обнаружения параметров.

Формат

Начальные значения используются для динамического обнаружения параметров при выполнении операции get.

При выполнении команды **artexget** каждый элемент **<Seed>** во входном профайле расширяется в один или несколько элементов **<Parameter>** на основании правил, определенных в соответствующем элементе **<SeedDef>** файла каталога. Этот процесс называется обнаружением параметра. Затем команда **artexget** продолжает свою работу обычным образом с расширенным профайлом.

Элемент **<SeedDef>** содержит только подэлемент **<Discover>**, который определяет команду для выполнения и маску для извлечения имен параметров, имен каталогов (в виде списков через точку с запятой без расширения **.xml**) и необязательных назначений из вывода команды (в формате *class1=instance1;class2=instance2;...*). Для каждой строки вывода загружается первый каталог из списка через двоеточие, который найден в системе. Если определение параметра найдено в этом каталоге, то параметр создается в выходном профайле, который имеет назначения, извлеченные из этой строки. Строки из вывода команды, которые не соответствуют маске, или для которых не найден файл каталога, или которые не имеют определения параметра, если находятся в файле каталога, игнорируются.

Примеры

1. Следующий каталог определяет элемент **<SeedDef>** по имени *vmoTunables*, который обнаруживает все неограниченные начальные значения *vmo tunables*, поддерживаемые AIX Runtime Expert:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Catalog id="vmoSeed">
  <SeedDef name="vmoTunables">
    <Discover>
      <Command>/usr/sbin/vmo -x | /usr/bin/awk -F, '{ print "vmoParam:" $1 }'</Command>
      <Mask catalog="1" name="2">(.*):(.*)/Mask>
    </Discover>
  </SeedDef>
</Catalog>
```

   Команда discovery распечатывает каждый tunable в отдельной строке после имени каталога, который их определяет:

```
...
vmoParam:enhanced_affinity_vmpool_limit
vmoParam:esid_allocator
vmoParam:force_relalias_lite
vmoParam:kernel_heap_psize
...
```

   Следующий профайл использует начальное значение *vmo tunables* для захвата всех неограниченных начальных значений *vmo tunables*, поддерживаемых AIX Runtime Expert:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Profile>
  <Catalog id="vmoSeed">
    <Seed name="vmoTunables"/>
  </Catalog>
</Profile>
```

   Когда команда **artexget –r** выполняется для профайла, она генерирует профайл, подобный следующему:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Profile>
 <Catalog id="vmoParam">
  ...
  <Parameter name="enhanced_affinity_vmpool_limit" value="10"/>
  <Parameter name="esid_allocator" value="0"/>
  <Parameter name="force_relalias_lite" value="0"/>
  <Parameter name="kernel_heap_psize" value="65536" applyType="nextboot" reboot="true"/>
  ...
</Catalog>
</Profile>
```

2. Следующий элемент **<SeedDef>** определяет начальное значение, которое может быть использовано для обнаружения всех атрибутов всех устройств. Элемент использует целевое начальное значение для обозначения устройства, с которым он работает:

```
<SeedDef name="devAttr">
  <Discover>
    <Command>
      /usr/sbin/lsdev -F 'name class subclass type' |
      while read DEV CLASS SUBCLASS TYPE
      do
        /usr/sbin/lsattr -F attribute -l $DEV |
        while read PAR
        do
          echo device=$DEV devParam.$CLASS.$SUBCLASS.$TYPE:devParam.$CLASS
.$SUBCLASS:devParam.$CLASS $PAR
        done
      done
    </Command>
    <Mask target="1" catalog="2" name="3">(.*) (.*) (.*)</Mask>
  </Discover>
</SeedDef>
```

Команда discovery распечатывает каждый обнаруженный атрибут устройства в отдельной строке в следующем формате:

```
device=DeviceName devParam.Class.Subclass.Type:devParam.Class.Subclass:devParam.Class
AttributeName
```

Пример:

```
device=en0 devParam.if.EN.en:devParam.if.EN:devParam.if tcp_recvspace
device=en0 devParam.if.EN.en:devParam.if.EN:devParam.if tcp_sendspace
device=ent0 devParam.adapter.vdevice.IBM,l-lan:devParam.adapter.vdevice:devParam.adapter
alt_addr
device=ent0 devParam.adapter.vdevice.IBM,l-lan:devParam.adapter.vdevice:devParam.adapter
chksum_offload
```

Элемент <Prereq>:

Элемент **<Prereq>** назначает предварительное условие операциям **<Get>**, **<Set>** и **<Discover>**.

Синтаксис

Родительский элемент: **<Get>**, **<Set>** и **<Discover>**

Поддерживается следующий атрибут:

Таблица 46. Атрибут

Атрибут	Обязательный	Тип	Описание
id	нет	строка	Указывает уникальный идентификатор

Поддерживаются следующие дочерние элементы:

Таблица 47. Дочерние элементы

Дочерний элемент	Обязательный	Описание
<Command>	нет	Команда
<Argument>	нет	Аргументы командной строки
<Stdin>	нет	Аргументы, поддерживаемые элементом <Stdin>
<ErrMessage>	нет	Сообщение для печати в случае невыполнения предварительного условия

Примечание: Элемент **<Command>** должен быть определен для каждого предварительного условия: на уровне **<ParameterDef>**, на уровне **<CfgMethod>** или в элементе **<PrereqDef>**.

Формат

Prereqs являются командами, которые обуславливают выполнение операций **<Get>**, **<Set>** и **<Discover>** для параметров, использующих операции **<Get>**, **<Set>** и **<Discover>**. Параметры, для которых не выполнена команда **prereq** (ненулевой код возврата), игнорируются, и показывается сообщение об ошибке, определенное в предварительном условии.

Элемент **<Prereq>** назначает предварительное условие родительским операциям **<Get>**, **<Set>** и **<Discover>**. Предварительное условие или определено локально в элементе **<Prereq>**, или унаследовано от элемента верхнего уровня **<Prereq>** или **<PrereqDef>**, который имеет соответствующий атрибут *id*.

Для параметра все предварительные условия определены локально в элементе **<ParameterDef>**. Кроме того, предварительное условие имеет свойства, определенные в методе конфигурации параметра, если он используется. В результате этого, если предварительное условие определено в элементе **<CfgMethod>**, то все

элементы **<ParameterDef>**, которые используют метод конфигурации, будут автоматически иметь это предварительное условие (хотя некоторые из них могут переопределить предварительное условие локально).

Элементы **<Command>**, **<Argument>**, **<Stdin>** и **<ErrMessage>**, которые определяют предварительное условие, ищутся в следующем порядке:

- В подэлементе **<Prereq>** соответствующей операции элемента **<ParameterDef>**.
- Если элемент **<ParameterDef>** имеет атрибут *cfgmethod*, в подэлементе **<Prereq>**, который имеет соответствующий атрибут *id* для данной операции метода конфигурации.
- В элементе **<PrereqDef>** каталога, который имеет соответствующий атрибут id.

Пример

Следующий пример определяет предварительное условие, которое проверяет, применяются ли параметры **netaddr** и **netaddr6** к той же системе, в которой они были захвачены:

```
<ParameterDef name="netaddr" type="string" targetClass="device" cfgmethod="attr">
  <Set type="permanent">
    <Prereq>
      <Command>[[ `/usr/bin/uname -f` = %p[nodeId] ]]</Command>
      <ErrMessage>Parameter cannot be applied to a different node</ErrMessage>
    </Prereq>
  </Set>
</ParameterDef>

<ParameterDef name="netaddr6" type="string" targetClass="device" cfgmethod="attr">
  <Set type="permanent">
    <Prereq>
      <Command>[[ `/usr/bin/uname -f` = %p[nodeId] ]]</Command>
      <ErrMessage>Parameter cannot be applied to a different node</ErrMessage>
    </Prereq>
  </Set>
</ParameterDef>
```

В этом примере проверка выполняется дважды: один раз для параметра **netaddr** и один раз для параметра **netaddr6**. Эта обработка выполняется дважды, потому что каждый параметр имеет свое собственное предварительное условие со своим собственным элементом **<Command>**. В разделе “Элемент <PrereqDef>” приведен пример, который требует только одного выполнения проверки.

Связанная информация

- “Генерация командной строки” на стр. 119
- “Элемент <PrereqDef>”

Элемент <PrereqDef>:

Элемент **<PrereqDef>**, который может быть впоследствии использован в элементе **<Prereq>**.

Синтаксис

Родительский элемент: **<Catalog>**

Поддерживается следующий атрибут:

Таблица 48. Атрибут

Атрибут	Обязательный	Тип	Описание
name	yes	строка	Задает имя свойства.

Поддерживаются следующие дочерние элементы:

Таблица 49. Дочерние элементы

Дочерний элемент	Обязательный	Описание
<Command>	нет	Команда
<Argument>	нет	Аргументы командной строки
<Stdin>	нет	Аргументы, поддерживаемые элементом <Stdin>
<ErrMessage>	нет	Сообщение для печати в случае невыполнения предварительного условия

Примечание: Элемент **<Command>** должен быть определен для каждого предварительного условия: на уровне **<ParameterDef>**, на уровне **<CfgMethod>** или в элементе **<PrereqDef>**.

Формат

Prereq являются командами, которые обуславливают выполнение операций **<Get>**, **<Set>** и **<Discover>** для параметров, использующих операции **<Get>**, **<Set>** и **<Discover>**. Параметры, для которых не выполнена команда **prereq** (ненулевой код возврата), игнорируются, и показывается сообщение об ошибке, определенное в предварительном условии.

Элемент **<PrereqDef>** определяет предварительное условие. Эти предварительные условия могут затем быть связаны с операцией параметра или методом конфигурации с помощью элемента **<Prereq>**, который имеет такой же атрибут *id*.

Пример

Следующий пример определяет предварительное условие *nodeId* и назначает его параметрам **netaddr** и **netaddr6**:

```
<PrereqDef id="nodeId">
  <Command>[[ `/usr/bin/uname -f` = %p[nodeId] ]]</Command>
  <ErrMessage>Parameter cannot be applied to a different node</ErrMessage>
</PrereqDef>

<ParameterDef name="netaddr" type="string" targetClass="device" cfgmethod="attr">
  <Set type="permanent">
    <Prereq id="nodeId"/>
  </Set>
  <Property name="nodeId"/>
</ParameterDef>

<ParameterDef name="netaddr6" type="string" targetClass="device" cfgmethod="attr">
  <Set type="permanent">
    <Prereq id="nodeId"/>
  </Set>
  <Property name="nodeId"/>
</ParameterDef>
```

В этом примере тест выполняется только один раз, потому что два параметра используют одинаковый элемент **<Command>** для своих предварительных условий, и сгенерированная командная строка одинакова для двух параметров.

Связанная информация

- “Генерация командной строки” на стр. 119
- “Элемент <Prereq>” на стр. 114

Элемент <Property>:

Элемент **<Property>** присваивает свойство параметру или методу конфигурации.

Синтаксис

Родительский элемент: **<CfgMethod>**, **<ParameterDef>**

Поддерживается следующий атрибут:

Таблица 50. Атрибут

Атрибут	Обязательный	Тип	Описание
name	yes	строка	Задает имя свойства.

Поддерживаются следующие дочерние элементы:

Таблица 51. Дочерний элемент

Дочерний элемент	Обязательный	Описание
<Command>	нет	Команда
<Argument>	нет	Аргументы командной строки
<Stdin>	нет	Аргументы, поддерживаемые элементом <Stdin>
<Filter>	нет	Фильтр
<Mask>	нет	Маска захвата вывода

Примечание: Элемент **<Command>** должен быть определен для каждого свойства: на уровне **<ParameterDef>**, на уровне **<CfgMethod>** или в элементе **<PropertyDef>**.

Формат

Свойства — это пары ключ-значение, связанные с параметром. Значение пары ключ-значение получается командами **artexget –r** и **artexget –n** и сохраняется в выходном профайле. Сохраненные в профайле значения свойств могут быть вставлены в командную строку с помощью последовательности %p[имя_свойства].

Элемент **<Property>** присваивает свойство параметру или методу конфигурации. Свойство или определено локально в элементе **<Property>**, или унаследовано от элемента верхнего уровня **<Property>** или **<PropertyDef>**, который имеет соответствующий атрибут имени.

Для параметра все свойства определены локально в элементе **<ParameterDef>**. Кроме того, все свойства параметра определены в методе конфигурации параметров, если он используется. В результате этого, если свойство определено в элементе **<CfgMethod>**, то все элементы **<ParameterDef>**, которые используют метод конфигурации, будут автоматически иметь это свойство (хотя некоторые из них могут переопределить свойство локально).

Значения свойств извлекаются из вывода командной строки. Командная строка составляется из комбинации элементов **<Command>**, **<Argument>**, **<Stdin>** и **<Filter>**, как описано в разделе Генерация командной строки. Необходимо использовать одно из следующих значений свойств: необработанный вывод командной строке или часть вывода, которая соответствует маске, если задан элемент **<Mask>**.

Элементы **<Command>**, **<Argument>**, **<Stdin>**, **<Filter>** и **<Mask>**, которые определяют свойство, ищутся в следующем порядке:

- В элементе **<Property>** на уровне **<ParameterDef>**.
- Если элемент **<ParameterDef>** имеет атрибут *cfgmethod*, в методе конфигурации элемента **<Property>**, который имеет соответствующий атрибут *name*.
- В элементе **<PropertyDef>** каталога, который имеет соответствующий атрибут name.

Пример

Следующий пример назначает свойство *nodeId* параметрам **netaddr** и **netaddr6**:

```
<ParameterDef name="netaddr" type="string" targetClass="device" cfgmethod="attr">
  <Property name="nodeId">
    <Command>/usr/bin/uname -f/<Command>
    <Mask>.*/<Mask>
  </Property>
</ParameterDef>

<ParameterDef name="netaddr6" type="string" targetClass="device" cfgmethod="attr">
  <Property name="nodeId">
    <Command>/usr/bin/uname -f</Command>
    <Mask>.*</Mask>
  </Property>
</ParameterDef>
```

В этом примере маска соответствует всей строке и используется только для исключения символа *новой строки* в конце вывода команды.

В этом примере команда **uname** выполняется дважды: один раз для параметра **netaddr** и один раз для параметра **netaddr6**. Команда выполняется дважды, потому что каждый параметр имеет свое собственное свойство со своим собственным элементом **<Command>**. В разделе “Элемент <PropertyDef>” приведен пример, который требует только одного выполнения команды **uname**.

Связанная информация

- “Генерация командной строки” на стр. 119
- “Развертывание элементов командной строки” на стр. 122
- “Элемент <PropertyDef>”

Элемент <PropertyDef>:

Элемент **<PropertyDef>** определяет свойство, которое может быть использовано в элементе **<Property>**.

Синтаксис

Родительский элемент: **<Catalog>**

Поддерживается следующий атрибут:

Таблица 52. Атрибут

Атрибут	Обязательный	Тип	Описание
name	yes	строка	Задает имя свойства.

Поддерживаются следующие дочерние элементы:

Таблица 53. Дочерний элемент

Дочерний элемент	Обязательный	Описание
<Command>	нет	Команда
<Argument>	нет	Аргументы командной строки
<Stdin>	нет	Аргументы, поддерживаемые элементом <Stdin>
<Filter>	нет	Фильтр
<Mask>	нет	Маска захвата вывода

Примечание: Элемент **<Command>** должен быть определен для каждого свойства: на уровне **<ParameterDef>**, на уровне **<CfgMethod>** или в элементе **<PropertyDef>**.

Формат

Свойства — это пары ключ-значение, связанные с параметром. Значение пары ключ-значение получается командами **artexget –r** и **artexget –n** и сохраняется в выходном профайле. Сохраненные в профайле значения свойств могут быть вставлены в командную строку с помощью последовательности %p[имя_свойства].

Элемент **<PropertyDef>** определяет свойство. Это свойство может быть затем связано с параметром или методом конфигурации с помощью элемента **<Property>**, который имеет такой же атрибут name.

Пример

Следующий пример назначает свойство *nodeId* параметрам **netaddr** и **netaddr6**:

```
<PropertyDef name="nodeId">
  <Command>/usr/bin/uname -f</Command>
  <Mask>.*</Mask>
</PropertyDef>

<ParameterDef name="netaddr" type="string" targetClass="device" cfgmethod="attr">
  <Property name="nodeId"/>
</ParameterDef>

<ParameterDef name="netaddr6" type="string" targetClass="device" cfgmethod="attr">
  <Property name="nodeId"/>
</ParameterDef>
```

В этом примере команда **uname** выполняется только один раз, потому что два параметра используют одинаковый элемент **<Command>** для своих свойств, и сгенерированная командная строка одинакова для двух параметров.

Связанная информация

- “Генерация командной строки”
- “Развертывание элементов командной строки” на стр. 122
- “Элемент <Property>” на стр. 117

Генерация командной строки

Среда AIX Runtime Expert опирается на внешние команды при получении, задании и, при необходимости, сравнении значений параметров. В этом разделе указано, как происходит компоновка командных строк на основе синтаксической информации из файлов-каталогов.

Операции

Для каждого параметра могут быть определены следующие операции:

- Операция get **type="current"**, извлекающая текущее значение параметра.

- Операция get **type="nextboot"**, извлекающая значение, которое будет присвоено параметру после перезагрузки.
- Операция set **type="current"**, задающая текущее значение параметра. Это значение параметра будет утеряно при перезагрузке.
- Операция set **type="nextboot"**, задающая значение, которое будет присвоено параметру после перезагрузки.
- Операция set **type="permanent"**, задающая текущее значение параметра, которое останется присвоенным параметру после перезагрузки.
- Операция **diff**, сравнивающая два значения параметра.
- Операция discover, находящая цели для параметров, которые их поддерживают.
- Свойство, используемое для захвата свойства параметра.
- Предварительное условие, используемое для условия выполнения операции get, set или discover для данного параметра.

Не все операции требуется определять для всех параметров. две операции **get** и все операции **set**, поддерживаемые параметрами, должны быть определены. Операция **diff** необязательна; если она не определена, то сравнения значений параметра будут проведены внутренне на основе типа параметра, такого как строка или целое число. Операция **discover** должна быть определена только для тех параметров, у которых есть цели. Свойства и предварительные условия определяются только при необходимости.

Элементы командной строки

Для каждой операции, поддерживаемой параметром, можно использовать до пяти различных элементов, чтобы определить способ компоновки командной строки для выполнения операции:

- Элемент **<Command>**, определяющий базовую команду для обработки параметров.
- Элемент **<Stdin>**, определяющий данные, которые будут записаны в стандартный ввод командной строки.
- Элемент **<Argument>**, вставляющий данные, относящиеся к параметру, в элемент **<Command>** или **<Stdin>**.
- Элемент **<Filter>**, фильтрующий вывод командной строки для операций **get** и **diff**.
- Элемент **<Mask>**, используемый для извлечения данных из вывода командной строки для операций **get**, **diff** и **property**.

Когда требуется выполнить операцию, элементы **<Command>**, **<Stdin>**, **<Argument>** и **<Filter>**, определенные для запрошенной операции, объединяются, образуя набор командных строк, как объяснено в разделе “Алгоритм генерации командной строки” на стр. 121. Генерируемые командные строки затем выполняются оболочкой. В случае операций **get**, **diff** и **property** элемент **<Mask>** извлекает запрошенные данные (значения параметров, результаты сравнения или значения свойств) из вывода команды.

Метод задания конфигурации

Элементы командной строки могут быть определены локально внутри элемента **<ParameterDef>** или унаследованы из элемента **<CfgMethod>**, на который ссылается элемент **<ParameterDef>** через атрибут *cfgmethod*.

Набор элементов командной строки, определенных для конкретной операции конкретного параметра, может представлять собой объединение элементов командной строки, определенных локально в элементе **<ParameterDef>**, и элементов командной строки, определенных для той же операции в элементе **<CfgMethod>**, на который ссылается атрибут *cfgmethod* элемента **<ParameterDef>**. Если один и тот же элемент командной строки определен и локально, и в методе задания конфигурации, то приоритет отдается локальному определению.

Рассмотрим, например, следующий неоптимизированный файл-каталог.

```
<CfgMethod id="vmo">
  <Get type="nextboot">
    <Command>/usr/sbin/vmo -r%a</Command>
    <Mask name="1" value="2">[[:space:]]*(.*) = (.*)</Mask>
  </Get>

  <Set type="permanent">
    <Command>/usr/sbin/vmo -p -o%a</Command>
    <Argument> -o %n=%p</Argument>
  </Set>

</CfgMethod>

<ParameterDef name="lgpg_size" cfgmethod="vmo">
  <Get type="current">
    <Command>/usr/sbin/vmo -o lgpg_size</Command>
    <Mask name="1" value="2">[[:space:]]*(.*) = (.*)</Mask>
  </Get>

  <Get type="nextboot">
    <Argument> -o lgpg_size</Argument>
  </Get>

</ParameterDef>
```

Мы можем видеть, что:

- Операция **<Get type="current">** полностью определяется на уровне **<ParameterDef>**.
- В операции **<Get type="nextboot">** есть несколько элементов, определенных на уровне **<CfgMethod>** (**<Command>** и **<Mask>**), и несколько элементов, определенных на уровне **<ParameterDef>** (**<Argument>**).
- Операция **<Get type="current">** полностью определяется на уровне **<CfgMethod>**.

Применение метода задания конфигурации дает два главных преимущества:

- Оно упрощает файл-каталог. Во многих случаях определения параметров унаследуют все свои элементы командной строки из метода задания конфигурации, а элемент **<ParameterDef>** будет пустым.
- Оно позволяет объединить различные параметры в одной командной строке, если это возможно.

Алгоритм генерации командной строки

Командные строки генерируются по алгоритму, позволяющему сгруппировать несколько параметров в одной команде.

Группировка параметров не только желательна с точки зрения производительности и эффективности, но и необходима для некоторых параметров. Например, для параметров **vmo** *lgpg_regions* и *lgpg_size*, которые не могут быть заданы независимо и должны быть заданы совместно в одном вызове команды **vmo**.

Алгоритм генерирования командной строки функционально эквивалентен следующей процедуре:

1. В каждом параметре входного профайла происходит частичное развертывание элементов **<Command>** и **<Stdin>**. На этом этапе последовательности %a, %v1[имя], %v2[имя], %f1[имя] и %f2[имя] игнорируются и не развертываются.
2. Параметры, удовлетворяющие всем пяти перечисленным ниже условиям, группируются:
   - Параметры используют один и тот же элемент **<Command>**.
   - Параметры используют один и тот же элемент **<Stdin>**.
   - Параметры используют один и тот же элемент **<Filter>**.
   - Развертывание элемента **<Command>**, выполняемое на шаге 1, дает одинаковые строки.
   - Развертывание элемента **<Stdin>**, выполняемое на шаге 1, дает одинаковые строки.

   Теперь в группе есть ее собственные, частично развернутые, элементы **<Command>** и **<Stdin>** и ее собственный элемент **<Filter>**, совместно используемый всеми параметрами группы.

3. В каждой группе параметров, в групповых элементах **<Command>** и **<Stdin>** развернуты последовательности %v1[имя], %v2[имя], %f1[имя] и %f2[имя]. Поиск имени параметра выполняется только в пределах группы.
4. В каждой группе параметров, в групповых элементах **<Command>** и **<Stdin>** развернуты последовательности %a: в каждом параметре группы развернут элемент **<Argument>**, а конкатенация этих развернутых элементов **<Argument>** заменяет любую последовательность %a в элементах **<Command>** и **<Stdin>**.

В результате этого процесса возникает набор командных строк, а также, возможно, данные для записи в их стандартный ввод и команда для фильтрации их вывода.

Развертывание элементов командной строки:

Элементы **<Command>**, **<Stdin>** и **<Argument>** поддерживают специальные последовательности, которые развертываются средой AIX Runtime Expert для создания окончательных командных строк.

В приведенной ниже таблице дана краткая справка обо всех поддерживаемых последовательностях. Более подробная информация о последовательностях приведена в следующих разделах.

Таблица 54. Последовательность

Последовательность	**Результат ее развертывания**
%%	Литерал %.
%a	Конкатенация развернутых строк **Argument** для всех параметров, которые могут быть развернуты в той же командной строке.
%n	Имя параметра.
%v1	Значение параметра.
%v2	Второе значение параметра. Допустимо только для операций **diff**.
%f1	Имя временного файла, в который будет записано значение параметра.
%f2	Имя временного файла, в который будет записано второе значение параметра. Допустимо только для операций **diff**.
%v1[имя]	Значение имени параметра.
%v2[имя]	Второе значение имени параметра. Допустимо только для операций **diff**.
%f1[имя]	Имя временного файла, в который будет записано значение имени параметра.
%f2[имя]	Имя временного файла, в который будет записано второе значение имени параметра. Допустимо только для операций **diff**.
%t[класс]	Имя целевого экземпляра для целевого класса.
%p[name]	Значение свойства *name*.
%c	ИД каталога.

Заключение последовательностей % в кавычки

Имена параметров, значения параметров и целевые имена, развернутые посредством AIX Runtime Expert, заключаются в одинарные кавычки, когда используются внутри элемента **<Command>** или элемента **<Argument>**, который должен быть вставлен (через последовательность %) в элемент **<Command>**. Это позволяет гарантировать, что эти строки будут переданы в оболочку как одно слово, даже если они содержат пробелы или другие специальные символы. Кроме того, любая отдельная кавычка в развернутом выражении будет правильно взята в кавычки.

Загрузчики файла-каталога не должны использовать последовательности %n, %v1, %v2, %v1[имя], %v2[имя] и %t[класс] внутри строк, взятых в кавычки. Если эти последовательности необходимо включить в состав строки, то строку необходимо закрыть до последовательности %, как показано в следующем примере:

```
echo "Parameter "%n" is set to "%v1
```

Если этого не сделать, то командные строки будут сформированы неверно и безопасность системы окажется под угрозой.

Последовательность %%

Последовательность %% развертывается в литерал %.

Например, строка:

```
/bin/ps -aeF"%%a"
```

развертывается в следующую строку:

```
/bin/ps -aeF"%a"
```

Последовательность %

Последовательность %a можно использовать либо в строке <Command>, либо в строке <Stdin>. Она заменяется на конкатенацию всех развернутых строк <Argument> всех параметров, которые можно рассматривать в той же команде (формальное описание группировки параметров приведено в разделе Генерация командной строки).

Например, следующий файл-каталог (заметьте, что его можно упростить с помощью последовательности %n):

```
<CfgMethod id="vmo">
  <Get type="current"
    <Command>/usr/sbin/vmo%a</Command>
  </Get>
</CfgMethod>
<ParameterDef name="lgpg_size" cfgmethod="vmo">
  <Get type="current">
    <Argument> -o lgpg_size</Argument>
  </Get>
</ParameterDef>
<ParameterDef name="lgpg_regions" cfgmethod="vmo">
  <Get type="current">
    <Argument> -o lgpg_regions</Argument>
  </Get>
</ParameterDef>
```

и следующий профайл:

```
<Parameter name="lgpg_size" />
<Parameter name="lgpg_regions" />
```

породят следующую командную строку для операции “get current”:

```
/usr/sbin/vmo -o lgpg_size -o lgpg_regions
```

Последовательность %

Последовательность %n заменяется на имя параметра.

С помощью последовательности %n пример из раздела %a можно упростить следующим образом:

```
<CfgMethod id="vmo">
<Get type="current">
    <Command>/usr/sbin/vmo%a</Command>
    <Argument> -o %n</Argument>
```

```
</Get>
</CfgMethod>
<ParameterDef name="lgpg_size" cfgmethod="vmo" />
<ParameterDef name="lgpg_regions" cfgmethod="vmo" />
```

со следующим профайлом:

```
<Parameter name="lgpg_size" />
<Parameter name="lgpg_regions" />
```

Для операции get current будет сгенерирована следующая командная строка:

```
/usr/sbin/vmo -o 'lgpg_size' -o 'lgpg_regions'
```

Последовательности %v1 и %v2

Последовательность %v1 заменяется на значение параметра.

Последовательность %v2 допустима только для операций **<Diff>** и заменяется на второе значение параметра.

Например, следующий файл-каталог:

```
<CfgMethod id="vmo">
  <Set type="permanent">
    <Command>/usr/sbin/vmo -p%a</Command>
    <Argument> -o %n=%v1</Argument>
  </Set>
</CfgMethod>
<ParameterDef name="lgpg_size" cfgmethod="vmo" />
<ParameterDef name="lgpg_regions" cfgmethod="vmo" />
```

со следующим профайлом:

```
<Parameter name="lgpg_size" value="16M"/>
<Parameter name="lgpg_regions" value="128" />
```

породят следующую командную строку для операции **set permanent**:

```
/usr/sbin/vmo -p -o 'lgpg_size'='16M' -o
'lgpg_regions'='128'
```

Последовательности %f1 и %f2

Последовательности %f1 и %f2 заменяются на имя временного файла, создаваемого до выполнения команды. Содержимое этого файла - это значение параметра для %f1 и второе значение параметра для %f2. Последовательность %f2 можно использовать только в операциях **<Diff>**.

Например, следующий файл-каталог:

```
<ParameterDef name="some_file">
  <Diff>
    <Command>/usr/bin/diff %f1 %f2</Command>
  </Diff>
</ParameterDef>
```

Если **artexdiff** выполняется между двумя профайлами, включающими один и тот же параметр с разными значениями:

```
<Parameter name="some_file" value="foo" />
<Parameter name="some_file" value="bar" />
```

то будут созданы два временных файла `/tmp/file1` и `/tmp/file2` (фактические имена файлов будут другими), содержащие соответственно строки “foo” и “bar”, и будет выполнена следующая команда:

```
/usr/bin/diff /tmp/file1 /tmp/file2
```

Последовательности %v1[имя] и %v2[имя]

Последовательность %v1[имя] заменяется на значение имени параметра.

Последовательность %v2[имя] допустима только для операций **<Diff>** и заменяется на второе значение имени параметра.

Эти последовательности полезны, когда команда задания конфигурации принимает несколько параметров одновременно, но требует, чтобы некоторые из них были помещены в определенную позицию в командной строке. Примером может служить команда **chcons**, которая требует, чтобы путь к консоли или файлу располагался последним в командной строке. С помощью последовательности %v1[имя] файл-каталог **chcons** можно записать следующим образом:

```
<CfgMethod id="chcons">
  <Set type="nextboot">
    <Command>/usr/sbin/chcons%a %v1[console_device]</Command>
    <Argument> -a %n=%v1</Argument>
  </Set>
</CfgMethod>
<ParameterDef name="console_device" cfgmethod="chcons" reboot="true" />
<ParameterDef name="console_logname" cfgmethod="chcons" reboot="true" />
<ParameterDef name="console_logsize" cfgmethod="chcons" reboot="true" />
```

со следующим профайлом:

```
<Parameter name="console_device" value="/dev/vty0"/>
<Parameter name="console_logname" value="/var/adm/ras/conslog" />
<Parameter name="console_logverb" value="9" />
```

Этот файл-каталог породит следующую командную строку для операции **set nextboot**:

```
/usr/sbin/chcons -a 'console_logname'='/var/adm/ras/conslog' -a 'console_logverb'='9' /dev/vty0
```

Последовательности %f1[имя] и %f2[имя]

Последовательности %f1[имя] и %f2[имя] заменяются на имя временного файла, создаваемого до выполнения команды. Содержимое этого файла - это значение имени параметра для %f1[имя] и второе значение имени параметра для %f2[имя]. Последовательность %f2[имя] можно использовать только в операциях **<Diff>**.

Последовательности %t[класс]

Последовательность %t[класс] заменяется на имя целевого экземпляра, рассматриваемого для целевого класса.

Последовательность %t[класс] используется для параметров, применяемых к конкретному объекту, а не к системе в целом. Примером может служить команда **chuser**, параметры которой применяются к конкретному пользователю (root, guest) для конкретного реестра (файлового, LDAP). Файл-каталог для команды **chuser** можно записать следующим образом:

```
<CfgMethod id="chuser">
  <Set type="permanent">
<Command>/usr/bin/chuser -R %t[module]%a %t[user]</Command>
    <Argument> %n=%v1</Argument>
  </Set>
</CfgMethod>
<ParameterDef name="shell" cfgmethod="chuser" targetClass="module,user">
<ParameterDef name="histsize" cfgmethod="chuser" targetClass="module,user" />
```

Со следующим профайлом, который задает параметры оболочки и параметр *histsize* для пользователей *adam* и *bob* в файловом реестре и реестре LDAP:

```
<Parameter name="shell" value="/usr/bin/ksh">
  <Target class="module" instance="LDAP" />
  <Target class="user" instance="adam" />
</Parameter>
<Parameter name="histsize" value="5000">
  <Target class="module" instance="LDAP" />
  <Target class="user" instance="adam" />
</Parameter>
<Parameter name="shell" value="/usr/bin/ksh">
  <Target class="module" instance="files" />
  <Target class="user" instance="adam" />
</Parameter>
<Parameter name="histsize" value="5000">
  <Target class="module" instance="files" />
  <Target class="user" instance="adam" />
</Parameter>
<Parameter name="shell" value="/usr/bin/bash">
  <Target class="module" instance="LDAP" />
  <Target class="user" instance="bob" />
</Parameter>
<Parameter name="histsize" value="10000">
  <Target class="module" instance="LDAP" />
  <Target class="user" instance="bob" />
</Parameter>
<Parameter name="shell" value="/usr/bin/bash">
  <Target class="module" instance="files" />
  <Target class="user" instance="bob" />
</Parameter>
<Parameter name="histsize" value="10000">
  <Target class="module" instance="files" />
  <Target class="user" instance="bob" />
</Parameter>
```

он выполнит следующие команды:

```
/usr/bin/chuser -R 'LDAP' 'shell'='/usr/bin/ksh' 'histsize'='5000' 'adam'
/usr/bin/chuser -R 'files' 'shell'='/usr/bin/ksh' 'histsize'='5000' 'adam'
/usr/bin/chuser -R 'LDAP' 'shell'='/usr/bin/bash' 'histsize'='10000' 'bob'
/usr/bin/chuser -R 'files' 'shell'='/usr/bin/bash' 'histsize'='10000' 'bob'
```

Обратите внимание на то, как генерируются четыре команды. Причина в том, что последовательности %t[модуль] и %t[пользователь] использовались в строке **<Command>**, что означает, что каждая команда предназначена для конкретного модуля и пользователя. Из-за этого сгруппированы лишь те параметры, которые применяются к одному модулю и пользователю.

Последовательность %p[name]

Последовательность %p[name] заменяется на значение, указанное во входном профайле для имени свойства. Например, следующее предварительное условие использует последовательность %p[nodeId] для проверки соответствия ИД узла локальной системы (возвращаемого командой **uname –f**) ИД узла, сохраненного в свойстве nodeId профайла:

```
<PrereqDef id="nodeId">
  <Command>[[ `/usr/bin/uname -f` = %p[nodeId] ]]</Command>
  <ErrMessage>Parameter cannot be applied to a different node</ErrMessage>
</PrereqDef>
```

Последовательность %c

Последовательность %c заменяется на ИД файла каталога, которому принадлежит параметр. Этот ИД каталога указан в профайле, и он может отличаться от ИД каталога, который фактически определяет параметр, если используется наследование каталога.

Например, следующее предварительное условие использует последовательность %c для проверки соответствия *uniquetype* целевого устройства имени файла каталога:

```
<PrereqDef id="devUniqueType">
  <Command>[[ "devParam.`/usr/sbin/lsdev -F uniquetype -l %t[device] | /usr/bin/tr / .`" = %c ]]</Command>
  <ErrMessage>Parameter cannot be applied to a different device type</ErrMessage>
</PrereqDef>
```

Команды и процессы

Команда - это запрос на выполнение операции или запуск программы. *Процесс* - это программа или команда, выполняемая на компьютере.

С помощью команд вы сообщаете операционной системе, какую задачу ей следует выполнить. Введенные команды расшифровываются интерпретатором команд (называемым *оболочкой*), после чего задача выполняется.

Операционная система может одновременно выполнять множество различных процессов.

Операционная система позволяет управлять потоками ввода и вывода данных с помощью специальных символов и команд ввода-вывода. При управлении вводом вы можете задать источник данных. Например, входные данные можно считывать с клавиатуры (стандартный ввод) или из файла. При управлении выводом вы можете указывать, куда следует направить полученные выходные данные. Например, вы можете указать, что выходные данные следует показать на экране (стандартный вывод) или записать в файл.

Команды

Для запуска некоторых команд достаточно ввести одно слово. Кроме того, команды можно комбинировать, чтобы вывод одной команды становился входными данными для другой.

Сочетание команд таким образом, чтобы вывод одной являлся вводом для другой, называется *конвейером*.

Флаги уточняют действие команд. *Флаг* - это модификатор, который указывается в командной строке вместе с именем команды, обычно после дефиса.

Команды можно объединять в группы и хранить в файле. Эти файлы называются *сценариями оболочки* или *процедурами оболочки*. В этом случае вместо того чтобы выполнять команды по отдельности, вы обрабатываете файл, содержащий команды.

Для запуска команды введите ее имя в командной строке и нажмите Enter.

```
$ Имя_команды
```

Понятия, связанные с данным:

“Функции оболочки” на стр. 210
В использовании оболочки в качестве системного интерфейса есть определенные преимущества.

Задачи, связанные с данной:

“Создание и выполнение сценария оболочки” на стр. 212
Сценарий оболочки - это файл, содержащий одну или несколько команд. Сценарии оболочки позволяют достаточно просто выполнять однообразные команды, длинные или сложные последовательности команд и процедуры. Когда вы вводите имя файла сценария оболочки, система выполняет содержащуюся в нем последовательность команд.

Синтаксис и имена команд:

Для запуска некоторых команд достаточно ввести одно слово; другие указываются с флагами и параметрами. Для каждой команды определен ее формат, задающий обязательные и необязательные флаги и параметры.

Общий формат команды выглядит следующим образом:

Имя_команды флаг(и)
параметр(ы)

Ниже описаны общие правила ввода команд:

- Пробелы, разделяющие команды, флаги и параметры, являются значащими.
- Две команды можно ввести на одной строке, разделив их точкой с запятой (;). Например:

  ```
  $
  Первая_команда;Вторая_команда
  ```

  Оболочка выполнит команды по очереди.
- В командах учитывается регистр символов. Оболочка различает строчные и прописные буквы. Команды `print`, `PRINT` и `Print` в оболочке считаются различными.
- Если команда очень длинная, то ее можно продолжить на следующей строке, поставив в конце строки обратную косую черту (\). Этот символ означает продолжение команды на следующей строке. В следующем примере одна команда занимает две строки:

  ```
  $ ls Mail info temp \
  (нажмите Enter)

  > diary
  (появится приглашение >)
  ```

  Символ **>** - это вспомогательное приглашение (основное приглашение обычных пользователей - $), означающая, что текущая строка представляет собой продолжение предыдущей. Учтите, что в оболочке C (`csh`) не появляется второе приглашение, разрыв в ней допустим только между словами и основное приглашение выглядит как %.

Первое слово в команде - это ее имя. Некоторые команды состоят только из имени.

Флаги команд:

Флаги указываются после имени команды. Они задают параметры операции, и иногда называются *опциями*.

Флаг отделяется от остального текста пробелами или символами и обычно начинается с дефиса (-). Исключение составляют команды **ps**, **tar** и **ar**, в которых перед некоторыми флагами не нужно указывать дефис. Например, в следующей команде:

```
ls -a -F
```

ls - это имя команды, а **-a -F** - флаги.

Флаги указываются после имени команды. Перед флагом, состоящим из одного символа, можно указать дефис. Например, предыдущую команду можно записать следующим образом:

```
ls -aF
```

Иногда дефис (**-**) - это первый символ параметра. В этом случае необходимо поставить два дефиса (**--**). Знаки **--** сообщают команде, что следующие за ними символы - это параметр, а не флаг.

Например, если для создания каталога **-tmp** вы укажете следующую команду:

```
mkdir -tmp
```

то система выдаст примерно следующее сообщение об ошибке:

```
mkdir: Флаг не распознан: t
Формат: mkdir [-p] [-m режим] Каталог ...
```

Эту команду необходимо задать следующим образом:

```
mkdir -- -tmp
```

Будет создан каталог `-tmp`.

Параметры команд:

После имени команды и флагов могут быть указаны параметры. Параметры иногда называют *аргументами* или *операндами*. Параметры задают информацию, необходимую для выполнения команды.

Если параметр не указан, то в некоторых случаях команда принимает значение по умолчанию. Например, в следующей команде:

```
ls -a temp
```

ls - это имя команды, **-a** - флаг, а *temp* - параметр. Эта команда выдает список всех (**-a**) файлов в каталоге `temp`.

В следующем примере:

```
ls -a
```

параметр не задан, поэтому по умолчанию принимается текущий каталог.

В следующем примере:

```
ls temp mail
```

флаги отсутствуют, а *temp* и *mail* - это параметры. В этом случае `temp` и `mail` задают имена двух различных каталогов. Команда **ls** выдаст список всех файлов из этих каталогов, кроме скрытых файлов.

Числа в параметрах или аргументах опций интерпретируются как десятичные целые, если не указано противное. Числовые константы от 0 до `INT_MAX`, согласно определению в файле `/usr/include/sys/limits.h`, распознаются как числовые значения.

Если в команде допустимы отрицательные значения параметров или опций, то вы можете указывать числовые константы в диапазоне от `INT_MIN` до `INT_MAX`, согласно определению в файле `/usr/include/sys/limits.h`. Это не означает, что допустимы все числовые значения из указанного диапазона. В некоторых командах, например, в командах печати, заданы ограничения, сужающие диапазон допустимых чисел. В случае ошибки появится сообщение о том, что число не входит в допустимый диапазон, а не о том, что команда синтаксически неправильна.

Формат:

Формат вызова описывает синтаксис команды и состоит из таких символов, как квадратные скобки ([ ]), круглые скобки ({ }) и вертикальная черта (|).

Ниже приведен пример формата вызова для команды **unget**:

unget [**-r***SID*] [**-s**] [**-n**] *Файл* ...

В описании формата вызова действуют следующие соглашения:

- Элементы, которые необходимо вводить так, как они указаны в описании формата, выделены **полужирным шрифтом**. В число таких элементов входят имена команд, флаги и литеральные символы.
- Элементы, представляющие переменные, вместо которых необходимо подставить фактическое значение, указаны *курсивом*. В число таких элементов входят параметры, задаваемые с флагами, и параметры команды, например, *файлы* и *каталоги*.
- Параметры, указанные в квадратных скобках, считаются необязательными.
- Параметры, указанные в фигурных скобках, считаются обязательными.
- Параметры, не заключенные ни в квадратные, ни в круглые скобки, считаются обязательными.

- Вертикальная черта указывает, что следует выбрать только один из перечисленных параметров. Например, [a | b] означает, что вы *можете* выбрать a, b или ничего не выбирать. Аналогично, { a | b } означает, что вы *должны* выбрать a или b.
- Многоточия (...) означают, что параметр можно повторять в командной строке.
- Дефис (-) задает стандартный ввод.

Команда shutdown:

Если у вас есть права доступа root, то вы можете завершить работу системы командой **shutdown**. Если у вас нет прав на применение команды **shutdown**, выйдите из системы, не выключая ее.

Внимание: Не выключайте систему, не завершив ее работу. Это приведет к завершению всех ее процессов. Если в системе работают другие пользователи или выполняются задания в фоновом режиме, то данные будут утеряны. Перед выключением системы необходимо выполнить специальные процедуры закрытия.

Введите в командной строке:

```
завершение работы
```

Когда выполнение команды **shutdown** закончится, и работа системы завершится, будет выдано следующее сообщение:

```
....Работа завершена....
```

Дополнительные сведения о синтаксисе приведены в описании команды **shutdown**.

Поиск команд и программ (команда whereis):

Команда **whereis** выполняет поиск в исходном, двоичном и справочном разделах указанных файлов. Команда пытается найти нужную программу в списке стандартных каталогов.

Примеры:

- Для того чтобы найти в текущем каталоге файлы, не содержащие документации, введите:

  ```
  whereis -m -u *
  ```
- Для того чтобы найти все файлы, содержащие слово Mail, введите:

  ```
  whereis Mail
  ```

  Появится приблизительно следующая информация:

  ```
  Mail: /usr/bin/Mail /usr/lib/Mail.rc
  ```

Сведения о синтаксисе приведены в описании команды **whereis** книги *Справочник по командам, том 6*.

Просмотр сведений о команде (команда man):

Команда **man** выдает информацию о командах, процедурах и файлах.

Общий формат команды **man** выглядит следующим образом:

```
man Имя_команды
```

Для просмотра информации о команде **pg** введите:

```
man pg
```

Появится приблизительно следующая информация:

```
Команда pg

Назначение
```

```
Форматирует файлы на экране.

Синтаксис

pg [ - число ] [ -c ] [ -e ] [ -f ] [ -n ] [ -p строка ]
[ -s ] [ +номер-строки | +/шаблон/ ] [ файл ... ]

Описание

Команда pg предназначена для постраничного просмотра содержимого файлов,
указанных в параметре Файл. Если указать
в параметре Файл дефис (-) или команда pg будет запущена
без параметров, то команда pg будет считывать данные из стандартного ввода. После
каждой страницы данных появляется приглашение. Если вы нажмете клавишу Enter,
то появится следующая страница. Подкоманды pg
позволяют перемещаться по файлу и выполнять поиск текста.
```

Сведения о синтаксисе приведены в описании команды **man** книги *Справочник по командам, том 3*.

Просмотр описания команды (команда whatis):

Команда **whatis** выполняет поиск команды, системного вызова, функции библиотеки или специального файла, указанных в параметре **Команда**, в базе данных, созданной командой **catman -w**.

Информация о команде **catman -w** приведена в описании команды **catman -w**. Команда **whatis** выдает заголовок справочного раздела. После этого для получения дополнительных сведений можно выполнить команду **man**. Дополнительная информация о команде **man** приведена в разделе **man**.

Команда **whatis** эквивалентна команде **man -f**.

Например, если вас интересует назначение команды **ls**, введите:

```
whatis ls
```

Появится приблизительно следующая информация:

```
ls(1)  -Показывает содержимое каталога.
```

Сведения о синтаксисе приведены в описании команды **whatis** книги *Справочник по командам, том 6*.

Просмотр введенных ранее команд с помощью команды (history):

Команда **history** применяется для просмотра команд, введенных ранее.

Команда **history** - это встроенная команда оболочки Korn, выдающая список последних 16 введенных команд. Оболочка Korn сохраняет введенные команды в файле хронологии команд, который обычно называется `$HOME/.sh_history`. Это позволяет ускорить повторный ввод команд.

По умолчанию оболочка Korn сохраняет текст последних 512 команд. Размер файла хронологии (заданный в переменной среды *HISTSIZE*) не ограничен, однако если он будет очень большим, что время запуска оболочки Korn может значительно возрасти.

Примечание: В оболочке Bourne хронологический список команд не ведется.

Для просмотра списка ранее введенных команд введите в командной строке:

```
history
```

Команда **history**, введенная без параметров, перечисляет последние 16 введенных команд. Появится приблизительно следующая информация:

```
928   ls
929   mail
930   printenv MAILMSG
931   whereis Mail
932   whatis ls
933   cd /usr/include/sys
934   ls
935   man pg
936   cd
937   ls | pg
938   lscons
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
```

В списке сначала указана позиция команды в файле `$HOME/.sh_history`, а затем - сама команда.

Для просмотра пяти предыдущих команд введите в командной строке:

```
history -5
```

Будет показан список следующего вида:

```
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
944   history -5
```

Команда **history**, в которой указан номер, выдает список всех ранее введенных команд, начиная с заданного номера.

Для просмотра команд, начиная с номера 938, введите в командной строке:

```
history 938
```

Будет показан список следующего вида:

```
938   lscons
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
944   history -5
945   history 938
```

Понятия, связанные с данным:

“Оболочки операционной системы” на стр. 206
Интерфейс операционной системы называется *оболочкой*.

“Подстановка хронологи команд” на стр. 256
С помощью встроенной команды **fc** можно просматривать и изменять содержимое файла хронологии. Для выбора части файла или редактирования строки укажите номер или начальные символы команды.

Повтор команд с помощью псевдонима r:

Псевдоним оболочки Korn **r** позволяет повторять ранее введенные команды.

Введите `r` и нажмите Enter (можно также указать номер или начальные символы нужной команды).

Для просмотра списка доступных дисплеев введите в командной строке `lsdisp`. Вывод команды будет показан на экране. Для того чтобы еще раз получить эту информацию, введите в командной строке:

```
r
```

Система еще раз запустит последнюю введенную команду. В данном примере это команда **lsdisp**.

Для повторного вызова команды**ls *.txt** введите в командной строке:

```
r ls
```

Псевдоним оболочки Korn **r** находит последнюю команду, начинающуюся с указанных символов.

Подстановка строк с помощью псевдонима r:

Псевдоним оболочки Korn **r** позволяет изменить команду перед ее запуском.

В этом случае можно изменить выполняемую команду с помощью параметра *старая строка***=***новая строка*.

Ниже приведены примеры применения псевдонима **r**:

- Например, если командой номер 940 была **ls** `*.txt`, и необходимо запустить команду **ls** `*.exe`, введите в командной строке:

  ```
  r txt=exe 940
  ```

  Будет запущена команда номер 940, в которой `exe` будет заменено на `txt`.
- Если команда 940 была последней командой, которая начинается со строчной буквы *l*, можно также ввести:

  ```
  r txt=exe l
  ```

Примечание: Только первое вхождение *старой строки* будет заменено на *новую строку*. Если вы введете псевдоним оболочки Korn **r** без номера команды и символа, то будет выполнена предыдущая введенная команда.

Изменение хронологии команд:

Встроенная команда оболочки Korn **fc** позволяет просматривать и редактировать файл хронологии команд.

Для выбора части файла или редактирования строки укажите номер или начальные символы команды. Вы можете указать как одну, так и несколько команд.

Если вы не укажете программу-редактор в качестве аргумента встроенной команды оболочки Korn **fc**, то будет применен редактор, заданный переменной *FCEDIT*. Если переменная *FCEDIT* не задана, то применяется редактор `/usr/bin/ed`. По окончании работы с редактором измененные команды будут напечатаны и запущены. Для просмотра значения переменной **FCEDIT** воспользуйтесь командой *printenv*.

Ниже приведены примеры редактирования хронологии команд:

- Например, для выполнения следующей команды:

  ```
  cd /usr/tmp
  ```

  которая очень похожа на команду, введенную под номером `933`, введите в командной строке:

  ```
  fc 933
  ```

  Будет запущен редактор по умолчанию с командой номер `933`. Измените `include/sys` на `tmp`. Когда вы завершите работу с редактором, команда будет запущена.
- Вы можете также указать редактор для команды **fc**. Например, для того чтобы отредактировать команду с помощью редактора `/usr/bin/vi`, введите в командной строке:

  ```
  fc -e vi 933
  ```

Будет запущен редактор `vi` с командой номер `933`.

- Вы можете также указать диапазон команд для редактирования. Например, для того чтобы отредактировать команды с 930 по 940, введите в командной строке:

  ```
  fc 930 940
  ```

  Будет запущен редактор по умолчанию с командами с 930 по 940. Когда вы завершите работу с редактором, все эти команды будут выполнены по очереди.

Создание псевдонима команды (команда alias):

Псевдоним позволяет создать удобное, короткое имя для команды, имени файла или произвольного текста оболочки. Применение псевдонимов экономит время при частом выполнении одних и тех же задач. Вы можете создать псевдоним команды.

Встроенная команда оболочки Korn **alias** определяет слово как псевдоним для некоторой команды. С помощью псевдонимов вы можете переопределять встроенные команды, но не зарезервированные слова.

Первым символом имени псевдонима может быть любой печатаемый символ, за исключением метасимволов. Все остальные символы должны быть такими же, как в допустимом имени файла.

Ниже приведен формат команды создания псевдонима:

```
alias
Имя=Строка
```

где *Имя* - это имя псевдонима, а *Строка* - строка символов. Если *Строка* содержит пробелы, она должна быть заключена в кавычки.

Ниже приведены примеры создания псевдонимов:

- Например, для создания псевдонима для команды **rm -i** (удаление файлов с подтверждением) введите в командной строке:

  ```
  alias rm="/usr/bin/rm -i"
  ```

  Теперь каждый раз, когда вы будете вводить команду **rm**, фактически будет выполняться команда `/usr/bin/rm -i`.

- ля создания псевдонима **dir** для команды **ls -alF | pg** (постранично выводит подробную информацию обо всех файлах текущего каталога, включая скрытые файлы; помечает выполняемые файлы символом `*`, а каталоги - символом `/`) введите в командной строке:

  ```
  alias dir="/usr/bin/ls -alF | pg"
  ```

  Теперь каждый раз, когда вы будете вводить команду **dir**, фактически будет выполняться команда `/usr/bin/ls -alF | pg`.

- Для просмотра всех псевдонимов введите в командной строке:

  ```
  alias
  ```

  Появится приблизительно следующая информация:

  ```
  rm="/usr/bin/rm -i"
  dir="/usr/bin/ls -alF | pg"
  ```

Понятия, связанные с данным:

“Псевдонимы команд в оболочке Korn (POSIX)” на стр. 256
Оболочка Korn (другое название - POSIX) позволяет создавать псевдонимы команд.

Поддержка символов расширенного международного набора при форматировании текста:

Команды форматирования текста позволяют работать с текстом, составленным из символов расширенного международного набора, предназначенного для европейских языков.

Международный расширенный набор символов содержит символы, применяемые во многих европейских языках, а также подмножество символов ASCII, состоящее из символов английского языка, цифр и знаков препинания.

Все символы в этом наборе заданы в формате символов ASCII. Этот формат позволяет представлять расширенные символы при вводе; кроме того, символы расширенного набора можно вводить непосредственно с устройства (клавиатуры), поддерживающего такой набор символов.

Следующие команды форматирования текста поддерживают все международные наборы однобайтовых символов. Команды находятся в каталоге `/usr/bin`. (Команды, помеченные звездочкой (*), поддерживают обработку текстов с многобайтовыми символами.

```
addbib*         hyphen          pic*            pstext
checkmm         ibm3812         ps4014          refer*
checknr*        ibm3816         ps630           roffbib*
col*            ibm5587G*       psbanne         soelim*
colcrt          ibm5585H-T*     psdit           sortbib*
deroff*         indxbib*        psplot          tbl*
enscript        lookbib*        psrev           troff*
eqn*            makedev*        psroff          vgrind
grap*           neqn*           psrv            xpreview*
hplj            nroff*
```

Команды форматирования текста и макрокоманды, отсутствующие в этом списке, не предназначены для обработки международных наборов символов.

Понятия, связанные с данным:

“Поддержка многобайтовых символов при форматировании текста” на стр. 136
Некоторые команды форматирования текста позволяют обрабатывать текст, содержащий многобайтовые символы.

Форматирование текста с помощью расширенного набора однобайтовых символов:

Если устройство ввода поддерживает символы из расширенного набора символов европейских языков, то вы можете вводить эти символы непосредственно на устройстве.

В противном случае для представления этих символов воспользуйтесь следующей Escape-последовательностью ASCII:

\[*N*], где *N* - это 2- или 4-значный шестнадцатиричный код символа.

> **Примечание:** формат NCesc \<xx> более не поддерживается.

Вывод текста, содержащего символы из расширенного набора, выполняется согласно правилам форматирования, установленным для применяемого языка. Символы, не определенные в интерфейсе устройства вывода, не выводятся или рассматриваются как ошибочные.

Хотя основным языком имен запросов, команд и макрокоманд является английский, большинство из них могут обрабатывать ввод (например, имена файлов и параметры), содержащий символы из расширенного набора для европейских языков.

Входные данные команд **nroff** и **troff** и их препроцессоров должны быть заданы в формате ASCII, иначе возникнет неустранимая синтаксическая ошибка. Символы из международного набора, как однобайтовые,

так и многобайтовые, можно вводить только в кавычках и без другого текста, предназначенного для форматирования. Рассмотрим пример применения макрокоманд команды **pic**:

```
define foobar %
Текст %
```

Указанное после директивы `define` имя `foobar` должно содержать только символы ASCII. Однако замещающий текст, `Текст`, может содержать и другие символы.

Поддержка многобайтовых символов при форматировании текста:

Некоторые команды форматирования текста позволяют обрабатывать текст, содержащий многобайтовые символы.

Такие команды помечены звездочкой (*) в списке из раздела Поддержка международных наборов символов при форматировании текста. Команды, отсутствующие в списке, не предназначены для обработки международных наборов символов.

Многобайтовые символы можно вводить непосредственно с устройства ввода, если оно поддерживает такие символы. В противном случае можно ввести любой многобайтовый символ в формате ASCII \[*N*], где *N* - это 2-, 4-, 6-, 7- или 8-значный шестнадцатеричный код символа.

Хотя основным языком имен запросов, команд и макрокоманд является английский, большинство из них могут обрабатывать ввод (например, имена файлов и параметры), содержащий многобайтовые символы.

Если вам уже знакомы команды форматирования текста, состоящего из однобайтовых символов, то просмотрите следующий список - в нем приведены специальные или уникальные характеристики многобайтовых локалей:

- Знаки переноса в тексте не применяются.
- Для вывода многобайтовых числовых символов необходимые специальные форматы. Вы можете воспользоваться форматами японского языка.
- Текст выводится горизонтально слева направо.
- Расстояние между символами фиксировано, т.е. символы автоматически выровнены по столбцам.
- Символы, не определенные в интерфейсе устройства вывода, не выводятся или рассматриваются как ошибочные.

Понятия, связанные с данным:

“Поддержка символов расширенного международного набора при форматировании текста” на стр. 135
Команды форматирования текста позволяют работать с текстом, составленным из символов расширенного международного набора, предназначенного для европейских языков.

Просмотр календаря:

С помощью команды **cal** вы можете направить в стандартный поток вывода календарь.

В параметре **месяц** указывается месяц, на который будет показан календарь. Допустимы значения от 1 до 12, соответствующие месяцам с января по декабрь. Если **месяц** не указан, команда **cal** показывает календарь на текущий месяц.

В параметре **год** указывается год, на который будет показан календарь. Поскольку команда **cal** может показать календарь для любого года с 1 по 9999, номер года необходимо указывать полностью, а не только последние две цифры. Если **год** не указан, команда **cal** показывает календарь на текущий год.

Ниже приведены примеры применения команды **cal**:

1. Для того чтобы просмотреть календарь на февраль 2002 года, введите:

   ```
   cal 2 2002
   ```

2. Нажмите Enter.
3. Для того напечатать календарь на весь 2002 год, введите:
   ```
   cal 2002 | qprt
   ```
4. Нажмите Enter.

Полный синтаксис приведен в описании команды **cal** в *Справочник по командам, том 1*.

Просмотр напоминаний:

Можно просмотреть напоминание, прочитав файл `calendar`. Этот файл создается в домашнем каталоге с помощью команды **calendar**. Эта команда записывает в стандартный вывод все строки файла, содержащие сегодняшнюю или завтрашнюю дату.

Для просмотра файла `calendar`, созданного в домашнем каталоге, введите команду **calendar**. Эта команда записывает в стандартный вывод все строки файла, содержащие сегодняшнюю или завтрашнюю дату.

Команда **calendar** распознает даты в формате 10 октября или 10.10. Кроме того, она допускает применение специального символа * (звездочка), если перед ним указана косая черта (/). Например, значение */7 интерпретируется как 7-е число каждого месяца.

По пятницам команда **calendar** показывает все строки с текущей датой и датами ближайшей субботы, воскресенья и понедельника. Эта команда не распознает праздничные дни. В праздничные дни команда выполняется обычным образом и показывает расписание только для следующего дня.

Стандартный файл календаря

Обычно файл `calendar` выглядит следующим образом:

```
25.* - Подготовить ежемесячный отчет
12 августа - Отлет в Магадан
23 августа - встреча
Жены нет дома - 23.8, 24.8, 25.8
24.8 - Забрать права из ГАИ
Суббота, 25 августа - отдых
27 августа - Встреча с клиентами
28 августа - Встреча с адвокатом
```

Для вызова команды **calendar** введите:

```
calendar
```

В пятницу, 24 августа, команда **calendar** покажет следующее:

```
25.* - Подготовить ежемесячный отчет
Жены нет дома - 23.8, 24.8, 25.8
24.8 - Забрать права из ГАИ
Суббота, 25 августа - отдых
27 августа - Встреча с клиентами
```

Стандартный файл calendar, содержащий оператор include

Ниже приведен пример файла `calendar`, содержащего оператор include:

```
#include </tmp/out>
21.1 - Ежегодная проверка
21.1 - Еженедельное обсуждение проекта
22.1  *Встреча с асфальтоукладчиками*
Прием у врача - 23.1
23.1 - Свадьба Павлова
```

Для вызова команды **calendar** введите:

```
calendar
```

В среду, 21 января, команда **calendar** покажет следующее:

```
21 января - прощальная вечеринка у Ивана
22 января  Встреча акционеров в Москве
21.1 - Ежегодная проверка
21.1 - Еженедельное обсуждение проекта
22.1  *Встреча с асфальтоукладчиками*
```

В выводе команды **calendar** использовалось содержимое файла `/tmp/out`:

```
21 января - прощальная вечеринка у Ивана
22 января  Встреча акционеров в Москве
```

Полный синтаксис приведен в описании команды **calendar** в *Справочник по командам, том 1*.

Разложение числа на простые множители:

Команда **factor** предназначена для разложения чисел на простые множители.

Если команда **factor** вызвана без параметра **число**, то она будет ожидать ввода натурального числа, меньшего 1E14 (100 000 000 000 000). Простые множители, составляющие это число, будут записаны в стандартный вывод. Множители выводятся по порядку. Каждый множитель указывается столько раз, сколько раз он используется в разложении. Для завершения работы команды введите 0 или любой символ, отличный от цифры.

Если команда **factor** вызвана с параметром **число**, то она определяет простые множители числа, записывает их в стандартный вывод и завершает работу.

Ниже приведен пример вычисления простых множителей:

1. Для разложения числа 123 на простые множители введите:
   ```
   factor 123
   ```
2. Нажмите Enter. Появится следующая информация:
   ```
   123 3 41
   ```

Полный синтаксис приведен в описании команды **factor** в *Справочник по командам, том 2*.

Поиск команды по ключевому слову:

С помощью команды **apropos** можно просмотреть разделы справочного руководства, заголовки которых содержат указанные *Ключевые слова*.

Команда **apropos** рассматривает отдельные слова без учета регистра символов. Также выводятся слова, входящие в состав других слов. Например, при поиске слова *compile* команда **apropos** также найдет все вхождения слова *compiler*.

Примечание: Вначале необходимо создать базу данных ключевых слов `/usr/share/man/whatis` с помощью команды **catman -w**.

Команда **apropos** эквивалентна команде **man** с опцией **-k**.

Например, для того чтобы найти разделы справочного руководства, в заголовках которых есть слово *password*, введите:

```
apropos password
```

Нажмите Enter.

Полный синтаксис приведен в описании команды **apropos** в книге *Справочник по командам, том 1*.

Процессы

Программа или команда, фактически выполняемая на компьютере, называется *процессом*.

Структура процессов иерархическая. Процесс, запущенный программой или командой, называется *родительским*; порожденный им процесс - *дочерним*. У родительского процесса может быть несколько дочерних процессов, однако у каждого дочернего процесса есть только один родительский процесс.

Каждому запускаемому процессу присваивается идентификационный номер (ИД процесса). Если вы несколько раз запускаете одну и ту же программу, то каждый раз ей будет присваиваться уникальный ИД процесса.

После запуска процесс начинает использовать различные системные ресурсы. Если процессов несколько, то встроенный в операционную систему планировщик распределяет время процессора между ними на основании приоритетов процессов. Эти приоритеты можно изменить командами **nice** и **renice**.

Примечание: Повысить приоритет процесса может только пользователь root. Обычный пользователь может понизить приоритет запускаемого процесса командой **nice**, а уже запущенного процесса - командой **renice** .

В следующем списке описаны типа процессов:

Интерактивные и фоновые процессы

Интерактивными называются процессы, которые запускаются пользователем, и те процессы, в работе которых принимает участие пользователь. Процессы, выполняемые независимо от пользователя, называются *фоновыми*. По умолчанию программы и команды выполняются как интерактивные процессы. Для запуска процесса в фоновом режиме укажите после команды амперсанд (&).

Демоны

Демоны - это процессы, выполняемые автономно. Они постоянно работают в фоновом режиме. Обычно демоны запускаются вместе с системой и завершают работу вместе с ней. Как правило, демоны обеспечивают работу системных служб и доступны в каждый момент времени нескольким задачам и пользователям. Демоны запускаются пользователем root или оболочкой root и могут быть завершены только пользователем root. Например, демон **qdaemon** предоставляет доступ к системным ресурсам (принтерам и т.п.). Другой пример стандартного демона - **sendmail**.

Процессы зомби

Процесс-зомби - это неактивный процесс, который по-прежнему находится в таблице процессов (иными словами, он сохраняет свой ИД процесса). Ему не выделяется системная память. Процессы-зомби - это процессы, которые были прерваны или завершены, но продолжают существовать в таблице процессов до тех пор, не будет завершен родительский процесс или не произойдет перезагрузка системы. Процессы-зомби отображаются как `<defunct>` в выводе команды **ps**.

Запуск процесса:

Для запуска интерактивного процесса на дисплейной станции введите имя программы или команды в командной строке.

После запуска интерактивного процесса он будет выдавать сообщения и принимать ваши ответы с дисплейной станции до тех пор, пока не будет завершен. Вы не сможете вводить другие команды и выполнять какие-либо действия на этой станции до тех пор, пока процесс не будет завершен или вы не прервете его.

Пользователь может одновременно запустить максимум 40 процессов.

Запуск интерактивного процесса

Для запуска интерактивного процесса введите команду со всеми необходимыми параметрами и флагами:

```
$ Имя_команды
```

Запуск фонового процесса

Для запуска процесса в фоновом режиме введите команду со всеми необходимыми параметрами и флагами и поставьте в конце амперсанд (&):

```
$ Имя_команды&
```

Если процесс работает в фоновом режиме, то вы можете выполнять другие задачи, вводя команды с дисплейной станции.

Фоновый режим рекомендуется применять в случае, если выполнение команды занимает много времени. Однако следует учесть, что фоновые процессы увеличивают нагрузку на процессор и, как следствие, могут замедлить работу системы.

Большинство процессов направляют результаты работы в стандартный вывод, даже если они работают в фоновом режиме. По умолчанию вывод отправляется на экран дисплея. Поскольку вывод фонового процесса может помешать выполнению остальных операций, его рекомендуется направлять в файл или на принтер. В этом случае, вы можете просмотреть вывод, когда он вам понадобится.

Примечание: В некоторых случаях порядок вывода процесса в фоновом режиме может быть несколько иным, чем в интерактивном. Обеспечить правильный порядок вывода независимо от режима выполнения процесса можно с помощью процедуры **fflush**.

Во время выполнения фонового процесса его состояние можно узнать с помощью команды **ps**.

Команда проверки состояния процесса (ps):

Если система работает, в ней обязательно выполняются процессы. С помощью команды **ps** вы можете выяснить, какие процессы активны, и просмотреть информацию о них.

У команды **ps** есть несколько флагов, позволяющих уточнить набор процессов, включаемых в список, и вид информации, выдаваемой о каждом из них.

Для просмотра всех запущенных процессов системы введите в командной строке:

```
ps-ef
```

Появится приблизительно следующая информация:

```
ПОЛЬЗ. PID  PPID   C    STIME    TTY  ВРЕМЯ CMD
root     1     0   0   Jun 28      -  3:23 /etc/init
root  1588  6963   0   Jun 28      -  0:02 /usr/etc/biod 6
root  2280     1   0   Jun 28      -  1:39 /etc/syncd 60
mary  2413 16998   2 07:57:30      -  0:05 aixterm
mary 11632 16998   0 07:57:31  lft/1  0:01 xbiff
mary 16260  2413   1 07:57:35  pts/1  0:00 /bin/ksh
mary 16469     1   0 07:57:12  lft/1  0:00 ksh /usr/lpp/X11/bin/xinit
mary 19402 16260  20 09:37:21  pts/1  0:00 ps -ef
```

Ниже приведено описание полей вывода:

Элемент	Описание
ПОЛЬЗ.	Имя пользователя
PID	ИД процесса
PPID	ИД родительского процесса
C	Время обработки процесса в CPU
STIME	Время запуска процесса
TTY	Управляющая рабочая станция процесса
ВРЕМЯ	Общее время выполнения процесса
CMD	Команда

В предыдущем примере ИД процесса, запущенного командой **ps -ef**, равен 19402. ИД родительского процесса - 16260, команда /bin/ksh.

Если список не умещается на экране, то будет показана только его нижняя часть. Для постраничного просмотра списка направьте вывод команды **ps** в команду **pg**. Введите в командной строке:

```
ps -ef | pg
```

Для просмотра состояния активных процессов системы введите:

```
ps gv
```

Будет выдана статистическая информация о каждом активном процессе. Вывод этой команды выглядит примерно следующим образом:

```
   PID    TTY STAT ВРЕМЯ PGIN  РАЗМЕР RSS   LIM  TSIZ   TRS %CPU %MEM КОМАНДА
     0      - A     0:44    7     8     8    xx     0     0  0.0  0.0 swapper
     1      - A     1:29  518   244   140    xx    21    24  0.1  1.0 /etc/init
   771      - A     1:22    0    16    16    xx     0     0  0.0  0.0 kproc
  1028      - A     0:00   10    16     8    xx     0     0  0.0  0.0 kproc
  1503      - A     0:33  127    16     8    xx     0     0  0.0  0.0 kproc
  1679      - A     1:03  282   192    12 32768   130     0  0.7  0.0 pcidossvr
  2089      - A     0:22  918    72    28    xx     1     4  0.0  0.0 /etc/sync
  2784      - A     0:00    9    16     8    xx     0     0  0.0  0.0 kproc
  2816      - A     5:59 6436  2664   616     8   852   156  0.4  4.0 /usr/lpp/
  3115      - A     0:27  955   264   128    xx    39    36  0.0  1.0 /usr/lib/
  3451      - A     0:00    0    16     8    xx     0     0  0.0  0.0 kproc
  3812      - A     0:00   21   128    12 32768    34     0  0.0  0.0 usr/lib/lpd/
  3970      - A     0:00    0    16     8    xx     0     0  0.0  0.0 kproc
  4267      - A     0:01  169   132    72 32768    16    16  0.0  0.0 /etc/sysl
  4514  lft/0 A     0:00   60   200    72    xx    39    60  0.0  0.0 /etc/gett
  4776  pts/3 A     0:02  250   108   280     8   303   268  0.0  2.0 -ksh
  5050      - A     0:09 1200   424   132 32768   243    56  0.0  1.0 /usr/sbin
  5322      - A     0:27 1299   156   192    xx    24    24  0.0  1.0 /etc/cron
  5590      - A     0:00    2   100    12 32768    11     0  0.0  0.0 /etc/writ
  5749      - A     0:00    0   208    12    xx    13     0  0.0  0.0 /usr/lpp/
  6111      - T     0:00   66   108    12 32768    47     0  0.0  0.0 /usr/lpp/
```

Сведения о синтаксисе приведены в описании команды **ps** книги *Справочник по командам, том 4*.

Настройка начального приоритета процесса (команда nice):

Вы можете уменьшить начальный приоритет процесса по сравнению с запланированным значением.

Чтобы задать начальный приоритет процесса ниже по сравнению с основным запланированным приоритетом, воспользуйтесь командой **nice** для запуска процесса.

Примечание: Для повышения приоритета процесса необходимы права доступа root.

Для того чтобы задать начальный приоритет процесса, введите:

```
nice -n Число Команда
```

где *Число* - число от 0 до 39 (39 - самый низкий приоритет). *Значение nice* - это десятичное представление приоритета процесса в системном планировщике заданий. Чем больше число, тем ниже приоритет. Ноль означает, что процесс необходимо запустить с базовым приоритетом планирования. *Команда* - это запускаемая команда с флагами и параметрами.

Сведения о синтаксисе приведены в описании команды **nice** книги *Справочник по командам, том 4*.

Для выполнения этой задачи можно также воспользоваться командой **smit nice**.

Изменение приоритета выполняемого процесса (команда renice):

Вы можете изменить приоритет уже выполняемого процесса с помощью команды **renice**, введенной в командной строке. Эта команда изменяет приоритет процесса, заданный командой nice.

Примечание: Повышать приоритет процессов и изменять приоритет чужих процессов может только пользователь root.

Для изменения приоритета активного процесса введите:

```
renice Приоритет
-p ИД_процесса
```

где *Приоритет* - число от -20 до 20. Чем больше число, тем ниже приоритет. Ноль означает, что процесс необходимо запустить с базовым приоритетом планирования. *ИД_процесса* - это ИД процесса, приоритет которого необходимо изменить.

Для выполнения этой задачи можно также воспользоваться командой **smit renice**.

Отмена интерактивного процесса:

Если вы решили прервать работу запущенного вами процесса, нажмите клавишу INTERRUPT. Обычно это клавиши Ctrl-C или Ctrl-Backspace.

Примечание: С помощью INTERRUPT (Ctrl-C) нельзя отменить фоновые процессы. Для отмены фонового процесса воспользуйтесь командой **kill**.

Большинство простых команд выполняются так быстро, что вы не просто не успеете их отменить. В этом разделе выбрана команда, выполнение которой занимает определенное время: **find / -type f**. Эта команда покажет пути ко всем файлам вашей системы. Вам не обязательно знать, как работает команда **find**, - она взята здесь только для примера.

В следующем примере команда **find** запускает процесс. Через несколько секунд после запуска вы можете прервать процесс, нажав клавишу INTERRUPT:

```
$ find / -type f
/usr/sbin/acct/lastlogin
/usr/sbin/acct/prctmp
/usr/sbin/acct/prdaily
/usr/sbin/acct/runacct
/usr/sbin/acct/sdisk
/usr/sbin/acct/shutacct INTERRUPT (Ctrl-C)
$ _
```

На экране вновь появится системное приглашение. Теперь вы можете ввести другую команду.

Задачи, связанные с данной:

“Просмотр функций клавиш терминала (команда stty)” на стр. 312
Команда **stty** показывает параметры терминала. В частности, с помощью этой команды можно просмотреть управляющие клавиши терминала.

Команды клавиатуры для остановки фонового интерактивного процесса:

Вы можете приостановить интерактивный процесс, не удаляя его ИД из таблицы процессов. Для этого нажмите клавиши Ctrl-Z.

Примечание: Сочетание клавиш Ctrl-Z работает в оболочках Korn (**ksh**) и C (**csh**), но не работает в оболочке Bourne (**bsh**).

Перезапуск остановленного процесса:

Эта процедура описывает, как возобновить процесс, приостановленный нажатием клавиш Ctrl-Z.

Примечание: Сочетание клавиш Ctrl-Z работает в оболочках Korn (**ksh**) и C (**csh**), но не работает в оболочке Bourne (**bsh**). Возобновить приостановленный процесс может только запустивший его пользователь или пользователь root.

1. Для просмотра списка запущенных и приостановленных процессов, то есть всех процессов системы, за исключением убитых, введите:

   ```
   ps -ef
   ```

   Эту команду можно объединить в конвейер с командой **grep** для просмотра только нужных вам процессов. Например, если вы хотите возобновить сеанс **vi**, введите:

   ```
   ps -ef | grep vi
   ```

   Эта команда выдаст только те строки из списка команды **ps**, которые содержат слово `vi`. Вывод будет выглядеть примерно так:

   ```
   UID    PID   PPID   C      STIME       TTY  ВРЕМЯ КОМАНДА
   root   1234  13682  0      00:59:53    -    0:01  vi test
   root  14277  13682  1      01:00:34    -    0:00  grep vi
   ```

2. В выводе команды **ps** найдите процесс, который вы хотите возобновить, и запишите его ИД. В данном примере ИД равен `1234`.
3. Для отправки приостановленному процессу сигнала CONTINUE (продолжить) введите:

   ```
   kill -19 1234
   ```

   Вместо `1234` укажите ИД нужного процесса. Флаг `-19` означает сигнал CONTINUE. Эта команда возобновляет процесс в фоновом режиме. Если процесс может продолжить работу в фоновом режиме, то процедура выполнена. Если процесс должен выполняться в интерактивном режиме (примером может служить сеанс **vi**), то перейдите к следующему шагу.
4. Для перевода процесса в интерактивный режим введите:

   ```
   fg 1234
   ```

   Вместо `1234` укажите ИД нужного процесса. Теперь процесс будет выполняться в интерактивном режиме. (Вы работаете с сеансом редактора **vi**).

Планирование запуска:

Вы можете запланировать запуск процесса как *пакетного* в заданное время.

Команды **at** и **smit** позволяют ввести имена команд, которые должны быть выполнены в будущем, и указать, когда именно следует выполнить эти команды.

Примечание: Доступ к команде **at** контролируют файлы `/var/adm/cron/at.allow` и `/var/adm/cron/at.deny`. Пользователь root может создавать, редактировать и удалять эти файлы. Записи в них - это имена пользователей, по одному имени на строку. Ниже приведен пример файла `at.allow`:

```
root
nick
dee
sarah
```

Если файл `at.allow` существует, то команду **at** могут применять только указанные в нем пользователи. Системный администратор может запретить пользователю применять команду **at**, указав его имя в файле `at.deny`. Если существует только файл `at.deny`, то команду **at** могут применять все пользователи, кроме тех, которые указаны в этом файле.

Команду **at** нельзя применять в следующих случаях:

- Файлы `at.allow` и `at.deny` не существуют (команду at разрешено применять только пользователю root).
- Файл `at.allow` существует, но имя пользователя в нем не указано.
- Файл `at.deny` существует и в нем указано имя пользователя.

Если файл `at.allow` не существует, а файл `at.deny` также не существует или пуст, то запустить задание командой **at** может только пользователь root.

Синтаксис команды **at** позволяет указать строку даты, строку времени и даты или строку приращения, задающую момент запуска процесса. Кроме того, вы можете указать оболочку или очередь для запуска процесса. Ниже приведено несколько примеров применения команды:

Например, если ваш ИД пользователя - `joyce` и вы хотите запустить сценарий `WorkReport` в полночь, выполните следующие действия:

1. Введите время планируемого запуска программы:
   ```
   at midnight
   ```
2. Введите имена запускаемых программ, нажимая Enter после каждого имени. После фамилии введите символ конца файла (Ctrl-D), сигнализируя об окончании списка.
   ```
   WorkReport^D
   ```

   После нажатия Ctrl-D появится примерно следующая информация:
   ```
   Время запуска задания joyce.741502800.a : пятница, 6 июля 2002 г. 00:00:00
   ```

   Программе `WorkReport` будет присвоен номер задания `joyce.741502800.a`; она будет запущена в полночь 6 июля.
3. Для просмотра списка программ, запланированных для запуска в будущем, введите:
   ```
   at -l
   ```

   Появится приблизительно следующая информация:
   ```
   joyce.741502800.a      пятница, 6 июля 2002 г. 00:00:00
   ```

Дополнительные сведения о синтаксисе приведены в описании команды **at**.

Задачи, связанные с данной:

“Просмотр списка запланированных процессов (команда at или atq)”
Список запланированных процессов можно просмотреть с помощью команды **at** с флагом **-l** или с помощью команды **atq**.

“Удаление процесса из расписания” на стр. 145
Вы можете удалить запланированный процесс с помощью команды **at** с флагом **-r**.

Просмотр списка запланированных процессов (команда at или atq):

Список запланированных процессов можно просмотреть с помощью команды **at** с флагом **-l** или с помощью команды **atq**.

Эти команды практически эквиваленты, однако команда **atq** может упорядочить процессы по моментам времени, когда вводилась команда **at**, и показать только номера процессов в очереди.

Вы можете просмотреть список запланированных процессов следующим образом:

- Введя **at** в командной строке
- Введя **atq** в командной строке

Команда at

Для просмотра запланированных процессов введите:

```
at -l
```

Эта команда выдает список запланированных процессов в очереди. Если вы пользователь root, то команда перечислит все запланированные процессы для всех пользователей. Полное описание синтаксиса приведено в справке по команде **at**.

Команда atq

Ниже приведены примеры применения команды **atq**:

- Для просмотра списка запланированных процессов в очереди введите:

  ```
  atq
  ```

- Если вы пользователь root, то вы можете просмотреть запланированные процессы в очереди конкретного пользователя, введя:

  ```
  atq Имя-пользователя
  ```

- Для просмотра количества запланированных процессов в очереди введите:

  ```
  atq -n
  ```

Задачи, связанные с данной:

“Планирование запуска” на стр. 143
Вы можете запланировать запуск процесса как *пакетного* в заданное время.

“Удаление процесса из расписания”
Вы можете удалить запланированный процесс с помощью команды **at** с флагом **-r**.

Удаление процесса из расписания:

Вы можете удалить запланированный процесс с помощью команды **at** с флагом **-r**.

Ниже приведены примеры применения команд **at** и **atq**:

1. Для удаления запланированного процесса необходимо знать его номер. Узнать номер можно с помощью команды **at -l** или команды **atq**.
2. Выяснив номер процесса, удаляемого из плана, введите:

   ```
   at -r Номер_процесса
   ```

Для выполнения этой задачи можно также воспользоваться командой `smit rmat`.

Задачи, связанные с данной:

“Просмотр списка запланированных процессов (команда at или atq)” на стр. 144
Список запланированных процессов можно просмотреть с помощью команды **at** с флагом **-l** или с помощью команды **atq**.

“Планирование запуска” на стр. 143
Вы можете запланировать запуск процесса как *пакетного* в заданное время.

Отмена фонового процесса (команда kill):

Если прервать интерактивный процесс нажатием клавиши INTERRUPT не удалось, либо если после запуска фонового процесса вы решили его прервать, то отменить процесс можно командой **kill**.

Перед применением команды **kill** необходимо узнать ИД (номер) процесса. Общий формат вызова команды **kill** выглядит следующим образом:

```
kill ИД_процесса
```

Примечание:

- Удалить процесс может только запустивший его пользователь или пользователь root. По умолчанию команда **kill** отправляет процессу сигнал -15 (SIGTERM).
- Для удаления процесса-зомби необходимо удалить его родительский процесс.

1. С помощью команды **ps** определите ИД процесса, который вы хотите удалить. Эту команду можно объединить в конвейер с командой **grep** для просмотра только нужных вам процессов. Например, если вы хотите узнать ИД процесса сеанса vi, введите:

   ```
   ps -l | grep vi
   ```

2. Предположим, что вы запустили команду **find** в фоновом режиме. Затем вы решили прервать процесс. Введите команду **ps** для просмотра списка номеров процессов.

   ```
   $ find / -type f > dir.paths &
   [1]    21593
   $ ps
      PID   TTY  ВРЕМЯ КОМАНДА
     1627  pts3  0:00  ps
     5461  pts3  0:00  ksh
    17565  pts3  0:00  -ksh
    21593  pts3  0:00  find / -type f
   $ kill 21593
   $ ps
      PID   TTY  ВРЕМЯ КОМАНДА
     1627  pts3  0:00  ps
     5461  pts3  0:00  ksh
    17565  pts3  0:00  -ksh
   [1]  +  Завершено 21593     find  /  -type f > dir.paths  &
   ```

   Команда **kill 21593** завершила работу фонового процесса **find**. В выводе второй команды **ps** отсутствует информация о состоянии процесса с ИД 21593. Система не выдаст сообщения о завершении процесса, пока вы не введете следующую команду (причем отличную от **cd**).

   Команда **kill** позволяет отменять фоновые процессы. Необходимость в этом возникает, когда процесс ошибочно переведен в фоновый режим или выполняется слишком долго.

Сведения о синтаксисе приведены в описании команды **kill** книги *Справочник по командам, том 3*.

Команду **kill** также можно вызвать с помощью **smit**. Для этого нужно ввести:

```
smit kill
```

Обзор команд для команд и процессов

Ниже описаны команды для работы с командами и процессами.

Таблица 55. Обзор команд для команд

Элемент	Описание
alias	Команда оболочки, записывающая список псевдонимов в стандартный вывод
history	Команда оболочки, выдающая хронологический список событий
man	Выдает электронную справку по командам, функциям и файлам
whatis	Описывает функцию, выполняемую командой
whereis	Выполняет поиск исходного кода, двоичного кода или электронной справки для установленных программ

Таблица 56. Обзор команд для работы с процессами

Элемент	Описание
at	Запускает команды в запланированное время, выдает список запланированных процессов или удаляет процесс из расписания
atq	Показывает очередь заданий, ожидающих запуска
kill	Отправляет сигнал выполняемым процессам
nice	Запускает команду с большим или меньшим приоритетом
ps	Показывает текущее состояние процессов
renice	Изменяет приоритет выполнения процессов

Устранение зависания системы

Средства устранения зависания системы позволяют пользователям не прекращать работу важных приложений во время устранения неполадок в системе. Средство обнаружения зависания системы предупреждает системного администратора о возможных неполадках и позволяет ему войти в систему с правами доступа root или перезапустить систему и устранить неполадку.

Команда shconf

Команда **shconf** вызывается, если включено **Средство обнаружения зависания системы**. Команда **shconf** настраивает отслеживание нужных событий и задает действия, которые будут предприняты в случае наступления события. Вы можете указать любое из описанных ниже действий, а также приоритет проверки, тайм-аут, во время которого не выполняются процессы и нити с более низким или заданным приоритетом, терминал, на который должно быть отправлено предупреждение, а также действие команды **getty**:

- Занести сообщение об ошибке в файл `errlog`
- Вывести сообщение с предупреждением на системную консоль (алфавитно-цифровую) или указанный терминал
- Перезагрузить систему
- Выполнить команду **getty**, чтобы пользователь мог войти с правами доступа root и вводить команды
- Вызвать команду

Если выбран вариант **Вызвать команду** или **Выполнить специальную команду getty**, то средство обнаружения зависания системы запустит специальную команду **getty** или указанную команду с самым высоким приоритетом. Специальная команда **getty** выдает сообщение о восстановлении **getty** с приоритетом 0. Приведенная ниже таблица содержит действия и значения, которые применяются средством обнаружения зависания системы по умолчанию. Для каждого типа обнаружения задается только одно действие.

Опция	Состояние	Приоритет	Тайм-аут (в секундах)
Занести сообщение об ошибке в файл `errlog`	отключена	60	120
Вывести сообщение с предупреждением	отключена	60	120
Выполнить команду getty для восстановления системы	разрешено	60	120
Вызвать команду	отключена	60	120
Перезагрузить систему	отключена	39	300

Примечание: Если включена опция **Выполнить команду getty для восстановления консоли**, то команда **shconf** добавляет флаг **-u** в команду **getty** в сценарии **inittab**, связанном с входом в систему с консоли.

При утере событий ввода-вывода вы можете указать значение тайм-аута и задать следующие действия:

Опция	Состояние
Вывести сообщение с предупреждением	отключена
Перезагрузить систему	отключена

Демон shdaemon

Демон **shdaemon** - это процесс, запускаемый сценарием **init** и работающий с приоритетом 0 (ноль). Он обеспечивает обнаружение зависания системы, получая информацию о конфигурации, инициализируя рабочие структуры и запуская обнаружение с указанными пользователем опциями.

Понятия, связанные с данным:

"Обнаружение по приоритету"
AIX может обнаружить, что система зависла, и попытаться устранить ошибку.

"Обнаружение по потерянному вводу-выводу" на стр. 149
AIX может обнаружить, что система зависла, и попытаться устранить ошибку.

Настройка средств обнаружения зависания системы

Вы можете управлять конфигурацией средств обнаружения зависания системы с помощью SMIT.

Опции меню SMIT позволяют включать и отключать механизм обнаружения, просматривать его текущее состояние и изменять и просматривать параметры конфигурации. Ниже перечислены команды настройки:

smit shd
Управление средствами обнаружения зависания системы

smit shstatus
Состояния средств обнаружения зависания системы

smit shpriocfg
Просмотр/изменение параметров приоритетного обнаружения неполадок

smit shreset
Восстановление конфигурации приоритетного обнаружения неполадок по умолчанию

smit shliocfg
Просмотр/изменение параметров обнаружения утерянных событий ввода-вывода

smit shlioreset
Восстановление конфигурации обнаружения утерянных операций ввод-вывода по умолчанию

Также управлять средствами обнаружения зависания системы можно с помощью команды **shconf**.

Обнаружение по приоритету

AIX может обнаружить, что система зависла, и попытаться устранить ошибку.

Все процессы (нити) выполняются с определенным приоритетом. Приоритет обозначается числом в диапазоне от 0 до 126. Самый высокий приоритет - ноль, самый низкий - 126. По умолчанию для всех нитей задан приоритет 60. Любой пользователь может понизить приоритет процесса с помощью команды **nice**. Пользователь с правами доступа root может повысить приоритет процесса.

Планировщик ядра всегда передает на выполнение процессору нить с наивысшим приоритетом. Таким образом, достаточное количество нитей с высоким приоритетом могут полностью занять систему, что приведет к невозможности выполнения нитей с более низким приоритетом. При работе нитей с приоритетом выше 60 (заданного по умолчанию) система может зависнуть в результате блокирования всех стандартных оболочек.

Функция Обнаружение зависания системы позволяет обнаружить подобную ситуацию и предоставляет системному администратору возможность восстановить работу системы. Эта функция реализуется демоном

(**shdaemon**), работающим с самым высоким приоритетом. Через определенный интервал времени демон проверяет, выполнялась ли нить с самым низким приоритетом. Если приоритет превышает заданное пороговое значение, демон может выполнить одно из нескольких действий. Для каждого из этих действий можно независимо настроить пороговое значение и интервал его проверки. Ниже приведены эти действия и параметры, заданные для них по умолчанию:

```
    Действие         Включено   Приоритет   Тайм-аут  Устройство
                     ------------- по умолчанию ---------------

1)  Регистрация
    ошибки               нет        60          2
2)  Выдача сообщения
    на консоль           нет        60          2      /dev/console
3)  Запуск оболочки с    да         60          2      /dev/tty0
    высоким приоритетом
4)  Запуск команды       нет        60          2
    с высоким приоритетом
5)  Перезагрузка         нет        39          5
```

Понятия, связанные с данным:

“Устранение зависания системы” на стр. 147
Средства устранения зависания системы позволяют пользователям не прекращать работу важных приложений во время устранения неполадок в системе. Средство обнаружения зависания системы предупреждает системного администратора о возможных неполадках и позволяет ему войти в систему с правами доступа root или перезапустить систему и устранить неполадку.

Обнаружение по потерянному вводу-выводу

AIX может обнаружить, что система зависла, и попытаться устранить ошибку.

Ошибки ввода-вывода могут вызвать блокирование путей ввода-вывода. Операционная система должна уметь выявлять такие ситуации, предупреждать о них пользователя и предпринимать указанные действия. Демон **shdaemon** совместно с администратором логических томов отслеживает состояние буферов ввода-вывода и выявляет данные, остающиеся в буфере слишком долго. Когда время ожидания превышает порог, заданный в файле **shconf**, ввод-вывод считается потерянным и предпринимаются действия по исправлению. Информация о потерянном вводе-выводе заносится в протокол ошибок. В зависимости от параметров, заданных в файле **shconf**, система может быть перезагружена для исправления ошибки.

Вы можете задать тайм-аут и выбрать следующие действия по исправлению:

Действие	Включено по умолчанию	Устройство по умолчанию
Сообщение на консоли	нет	/dev/console
Перезагрузка	нет	-

Дополнительная информация об обнаружении зависания системы приведена в разделе “Устранение зависания системы” на стр. 147.

Понятия, связанные с данным:

“Устранение зависания системы” на стр. 147
Средства устранения зависания системы позволяют пользователям не прекращать работу важных приложений во время устранения неполадок в системе. Средство обнаружения зависания системы предупреждает системного администратора о возможных неполадках и позволяет ему войти в систему с правами доступа root или перезапустить систему и устранить неполадку.

Управление процессами

Процессы - это объекты, применяемые операционной системой для управления распределением ресурсов. *Нити* предназначены для дополнительного контроля над распределением времени процессора, однако большая часть инструментов управления системой по-прежнему требуют указать процесс, в котором работает нить, а не саму нить.

В системе предусмотрены средства для выполнения следующих операций:

- Наблюдение за созданием, отменой, идентификацией и распределением ресурсов между процессами.
  - Команда **ps** показывает идентификаторы, владельцев и распределение времени CPU между различными процессами.
  - Команда **who -u** показывает идентификаторы процессов оболочек для вошедших в систему пользователей.
  - Команда **svmon** показывает распределение физической памяти между процессами. (Дополнительная информация о команде **svmon** приведена в разделе *Performance Toolbox версии 3: Справочник и руководство*).
  - Команда **acct** регистрирует объем занимаемой процессом памяти во время его завершения.
- Управления уровнем приоритета, с которым процесс конкурирует за время центрального процессора.
  - Команда **nice** запускает команду с указанным приоритетом.
  - Команда **renice** изменяет приоритет выполнения указанного процесса.
- Завершение процессов.
  - Команда **kill** отправляет одному или нескольким процессам сигнал завершения.

Понятия, связанные с данным:

“Учет ресурсов системы” на стр. 156
Утилита учета ресурсов системы позволяет собирать данные и создавать отчеты об индивидуальном и групповом использовании различных ресурсов системы.

Отслеживание процессов

В качестве системного администратора вы можете отслеживать процессы.

Основное средство отслеживания выполнения процессов - это команда **ps**. Большую часть флагов команды **ps** можно отнести к одной из следующих категорий:

- Флаги, задающие тип процессов, которые нужно включить в вывод
- Флаги, задающие атрибуты процессов, которые нужно включить в вывод

Для управления системой наиболее часто применяются следующие форматы команды **ps**:

Элемент	Описание
ps -ef	Показывает для всех процессов, кроме процессов ядра, ИД пользователя, ИД процесса, полную и текущую нагрузку на CPU и команду запуска процесса (включая параметры).
ps -fu *ИД-пользователя*	Показывает для всех процессов, принадлежащих *ИД-пользователя*, ИД процесса, текущую и полную нагрузку на CPU и команду запуска процесса (включая параметры).

Для определения пользователей, процессы которых наиболее сильно загружают CPU, введите:

```
ps -ef | egrep -v "STIME|$LOGNAME" | sort +3 -r | head -n 15
```

Команда выдаст список из 15 наиболее интенсивно загружающих CPU процессов других пользователей, в порядке убывания.

Вы можете указать несколько флагов в команде **ps**, руководствуясь следующими таблицами:

Флаги выбора процессов

	-A	-a	-d	-e	-G -g	-k	-p	-t	-U -u	a	g	t	x
Все процессы	+	-	-	-	-	-	-	-	-	-	+	-	-
Не связанные с термина- лом, и не лидеры групп процессов	-	+	-	-	-	-	-	-	-	-	-	-	-
Все, кроме лидеров групп процессов	-	-	+	-	-	-	-	-	-	-	-	-	-
Все, кроме процессов ядра	-	-	-	+	-	-	-	-	-	-	-	-	-
Члены указанной группы процессов	-	-	-	-	+	-	-	-	-	-	-	-	-
Процессы ядра	-	-	-	-	-	+	-	-	-	-	-	-	-
Указанные в списке номеров процессов	-	-	-	-	-	-	+	-	-	-	-	-	-
Связанные с терминалами из списка	-	-	-	-	-	-	-	Y (*n* терм)	-	-	-	Y (1 терм)	-
Указанного пользователя	-	-	-	-	-	-	-	-	+	-	-	-	-
Связанные с терминалами	-	-	-	-	-	-	-	-	-	+	-	-	-
Не связанные с терминалами	-	-	-	-	-	-	-	-	-	-	-	-	+

Флаги выбора столбцов

По умолч. 1	-f	-l	-U -u	По умолч. 2	e	l	s	u	v	
PID	+	+	+	+	+	+	+	+	+	+
TTY	+	+	+	+	+	+	+	+	+	+
ВРЕМЯ	+	+	+	+	+	+	+	+	+	+
CMD	+	+	+	+	+	+	+	+	+	+
ПОЛЬЗ.	-	+	-	-	-	-	-	-	+	-
UID	-	-	+	+	-	-	+	-	-	-
PPID	-	+	+	-	-	-	+	-	-	-
C	-	+	+	-	-	-	+	-	-	-
STIME	-	+	-	-	-	-	-	-	+	-
F	-	-	+	-	-	-	-	-	-	-
S/STAT	-	-	+	-	+	+	+	+	+	+
PIR	-	-	+	-	-	-	+	-	-	-
NI/NICE	-	-	+	-	-	-	+	-	-	-
ADDR	-	-	+	-	-	-	+	-	-	-
SIZE	-	-	-	-	-	-	-	-	+	-
SZ	-	+	-	-	-	+	-	+	-	-

Флаги выбора столбцов

По умолч. 1	-f	-l	-U -u	По умолч. 2	e	l	s	u	v	
WCHAN	-	-	+	-	-	-	+	-	-	-
RSS	-	-	-	-	-	-	+	-	+	+
SSIZ	-	-	-	-	-	-	-	+	-	-
%CPU	-	-	-	-	-	-	-	-	+	+
%MEM	-	-	-	-	-	-	-	-	+	+
PGIN	-	-	-	-	-	-	-	-	-	+
LIM	-	-	-	-	-	-	-	-	-	+
TSIZ	-	-	-	-	-	-	-	-	-	+
TRS	-	-	-	-	-	-	-	-	-	+
Environment (после команды)	-	-	-	-	-	+	-	-	-	-

Если команда **ps** запущена без флагов или с флагом выбора процессов, начинающимся со знака минус, будут показаны столбцы По умолчанию 2. Если команда запущена с флагом выбора процессов, не начинающимся со знака минус, будут показаны столбцы По умолчанию 2. Флаги **-u** и **-U** выбирают одновременно и процессы, и столбцы.

Ниже приведено краткое описание содержимого столбцов:

Элемент	Описание
PID	ИД процесса
TTY	Терминал или псевдотерминал, связанный с процессом.
ВРЕМЯ	Полное время процессора, затраченное на выполнение процесса, в минутах и секундах.
CMD	Команда запуска процесса.
ПОЛЬЗ.	Имя пользователя - владельца процесса
UID	ИД пользователя - владельца процесса
PPID	ИД родительского процесса
C	Использованное время CPU
STIME	Время запуска процесса, если он был запущен менее 24 часов назад. Иначе - дата запуска процесса
F	Восемь шестнадцатеричных цифр, описывающих флаги, связанные с процессом (см. описание команды **ps**)
S/STAT	Состояние процесса (см. описание команды **ps**)
PRI	Текущий приоритет процесса
NI/NICE	Значение параметра nice процесса
ADDR	Номер сегмента стека процесса
SIZE	(Флаг **-v**) Виртуальный размер блока данных процесса (в КБ)
SZ	(Флаги **-l** и **l**) Размер образа ядра процесса (в КБ).
WCHAN	Событие, ожидаемое процессом
RSS	Число страниц памяти, выделенных процессу под рабочие сегменты и под сегменты кода, умноженное на 4
SSIZ	Размер стека ядра
%CPU	Использование процессом CPU, в процентах, за все время его работы
%MEM	Номинальное количество реальной памяти, выделенной процессу, в процентах. Это число не связано с другими значениями использования памяти
PGIN	Число обращений к страницам, при которых были обнаружены страничные ошибки. Поскольку весь ввод-вывод в системе организован на основе страничных ошибок, обычно это число характеризует интенсивность ввода-вывода.
LIM	Всегда **xx**
TSIZ	Размер текстового раздела исполняемого файла
TRS	Число страниц, выделенных под сегменты кода, умноженное на 4
Среда	Значения всех переменных среды процесса

Изменение приоритета процесса

Обычно, если вы обнаружили процесс, занимающий слишком много времени CPU, вы можете уменьшить его приоритет, увеличив значение nice командой **renice**.

Пример:

renice +5 *ИД-процесса*

Значение nice процесса с указанным *ИД_процесса* будет увеличено от обычного для интерактивного процесса значения 20 до значения 25. Для изменения величины nice указанного *ИД-процесса* на значение 20 у вас должны быть права доступа пользователя root. Введите:

renice -5 *ИД-процесса*

Завершение процесса

Обычно для завершения процесса используется команда **kill**.

Команда **kill** отправляет указанному процессу определенный сигнал. В зависимости от типа сигнала и программы, процесс может остановиться или продолжить выполнение. Для завершения обычно применяются следующие сигналы:

Элемент	Описание
SIGTERM	(Сигнал 15) - запрос на завершение программы. Если программа создала процедуру обработки сигнала SIGTERM, не завершающую приложение, вызов команды **kill** не приведет к желаемым результатам. Этот сигнал отправляется командой **kill** по умолчанию.
SIGKILL	(Сигнал 9) - директива немедленного завершения процесса. Этот сигнал не может быть перехвачен или проигнорирован.

Обычно рекомендуется применять сигнал SIGTERM, а не SIGKILL. Обработчик сигнала SIGTERM программы может выполнять очистку и нормальное завершение программы. Введите:

kill -term *ИД-процесса*

(Флаг **-term** можно опустить.) Если процесс не завершается сигналом SIGTERM, введите:

kill -kill *ИД-процесса*

В таблице процессов вы можете найти не работающие процессы, называемые также *процессами-зомби*. Это процессы, которые больше не обрабатываются, не имеют выделенной системной памяти, но их ИД процесса еще не освобожден. Для процессов-зомби в столбце Команда указано значение <не функц.>. Например:

```
 UID    PID   PPID   C      STIME       TTY  ВРЕМЯ КОМАНДА
                            .
                            .
                            .
 lee 22392 20682   0   Июл 10      -  0:05 xclock
 lee 22536 21188   0   Июл 10  pts/0  0:00 /bin/ksh
 lee 22918 24334   0   Июл 10  pts/1  0:00 /bin/ksh
 lee 23526 22536  22                   0:00 <не функц.>
 lee 24334 20682   0   Июл 10      ?  0:00 aixterm
 lee 24700     1   0   Июл 16      ?  0:00 aixterm
root 25394 26792   2   Июл 16  pts/2  0:00 ksh
 lee 26070 24700   0   Июл 16  pts/3  0:00 /bin/ksh
 lee 26792 20082   0   Июл 10  pts/2  0:00 /bin/ksh
root 27024 25394   2 17:10:44  pts/2  0:00 ps -ef
```

Процессы-зомби остаются в таблице процессов до тех пор, не будет завершен их родительский процесс или не произойдет перезагрузка системы. В приведенном выше примере родительский процесс (PPID) - это команда **ksh**. После выхода из оболочки Korn процесс-зомби будет удален из таблицы процессов.

Иногда число не работающих процессов в таблице увеличивается из-за того, что приложение создает дочерние процессы и не завершает свою работу. Простейшим выходом из такой ситуации может быть изменение приложения, чтобы входящая в его состав функция **sigaction** игнорировала сигнал **SIGCHLD**.

Информация, связанная с данной:

Команда sigaction

Связывание и отмена связывания процесса

Процесс можно привязать к процессору или отменить связывание ранее привязанного процесса.

Для управления связыванием процессов, не принадлежащих вам, у вас должны быть права пользователя root.

В многопроцессорных системах можно связать процесс с процессором или отменить это связывание с помощью:

- SMIT
- Командной строки

Примечание: Хотя связывание процесса с процессором может увеличить производительность этого процесса (за счет уменьшения числа ошибок аппаратного кэша), злоупотребление этой возможностью может привести к перегрузке одних процессоров при недостаточном использовании других. В результате общее быстродействие может уменьшиться. При обычной работе рекомендуется позволить операционной системе автоматически распределять процессы из соображений равномерной загрузки всех процессоров. Применяйте связывание только к тем процессам, для которых выполнение одним процессором существенно повышает производительность.

Задачи управления связыванием процессов

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Связывание процесса	**smit bindproc**	**bindprocessor -q**
Отмена связывания	**smit ubindproc**	**bindprocessor -u**

Устранение зависших или ненужных процессов:

Зависшие или ненужные процессы могут стать причиной возникновения неполадок на терминале. При возникновении некоторых неполадок на экран будут выводиться сообщения, содержащие информацию о возможных причинах неполадок.

Для выполнения описанных ниже процедур необходим второй терминал, модем или возможность сетевого входа в систему. Если ни одно из этих условий не выполнено, перезагрузите систему, чтобы устранить неполадку терминала.

Выберите необходимую процедуру для устранения неполадки терминала:

Высвобождение терминала, занятого процессом:

Устаревшие или ненужные процессы можно остановить.

Для идентификации и завершения зависшего или ненужного процесса выполните следующие действия:

1. Для определения активных процессов, использующих терминал, введите **ps**:
   ```
   ps -ef | pg
   ```
   Команда **ps** показывает сведения о состоянии процесса. Флаг **-e** означает, что должна быть показана информация обо всех процессах (кроме процессов ядра), а флаг **-f** указывает, что в списке процессов должны быть показаны имена команд и значения параметров, с помощью которых был запущен процесс. Команда **pg** показывает информацию по одной странице, чтобы большой объем информации не прокручивался на экране слишком быстро.

   Найдите системные или пользовательские процессы, использующие слишком большой объем системных ресурсов, таких как время CPU или дисковое пространство. Среди системных процессов это могут быть процессы **sendmail**, **routed** и **lpd**. Проверьте использование процессора с помощью команды **ps -u**.
2. Для определения списка пользователей, запустивших процессы на данной системе, введите команду **who**:
   ```
   who
   ```

Команда **who** показывает следующую информацию о пользователях работающих в данный момент в системе: имя пользователя, имя рабочей станции, дата и время входа в систему.

3. Выберите действие, которое необходимо выполнить над пользовательским процессом: завершить, приостановить, или изменить приоритет.

   Примечание: Для завершения процессов других пользователей необходимы права доступа root. Удалив пользовательский процесс или изменив его приоритет, сообщите об этом владельцу процесса.

   - Остановите процесс с помощью команды **kill**. Например:

     ```
     kill 1883
     ```

     Команда **kill** отправляет работающему процессу определенный сигнал. Для того, чтобы завершить процесс, необходимо указать его ИД (PID) (в данном примере - 1883). Определить PID процесса можно с помощью команды **ps**.

   - Для приостановки процесса и его выполнения в фоновом режиме применяется амперсанд (&). Пример:

     ```
     /u/bin1/prog1 &
     ```

     Символ **&** означает, что процесс должен работать в фоновом режиме. В фоновом процессе оболочка не ожидает завершения обработки команды перед возвратом управления командной строке. Если на выполнение процесса требуется более нескольких секунд, запустите команду в фоновом режиме, добавив в конце символ **&**. Обычная команда **ps** позволяет просмотреть задания, выполняемые в фоновом режиме.

   - Для изменения приоритета процессов применяется команда **renice**:

     ```
     renice 20 1883
     ```

     Команда **renice** позволяет изменить приоритет планирования одного или нескольких активных процессов. Чем больше число, тем ниже приоритет, наименьший приоритет - 20.

     В предыдущем примере команда **renice** присваивает процессу с номером 1883 наименьший приоритет. Этот процесс будет выполняться, когда появится небольшой интервал свободного времени процессора.

Ответ на выводимые на дисплее сообщения:

С помощью данной процедуры можно отвечать на сообщения, выводящиеся на дисплее.

1. Проверьте правильность настройки переменной среды **DISPLAY**. Для проверки переменной **DISPLAY** воспользуйтесь одним из следующих способов:

   - Для просмотра переменных среды используйте команду **setsenv**.

     ```
     setsenv
     ```

     Команда **setsenv** показывает защищенные переменные среды после входа в систему.

     Определите, была ли задана переменная **DISPLAY**. В приведенном ниже примере строка **DISPLAY** не показана, что свидетельствует о том, что значение переменной **DISPLAY** не задано.

     ```
     SYSENVIRON:
     NAME=casey
     TTY=/dev/pts/5
     LOGNAME=casey
     LOGIN=casey
     ```

     ИЛИ

   - Измените значение переменной **DISPLAY**. Например, для того чтобы задать в качестве дисплея систему с именем `bastet` и терминал 0, введите:

     ```
     DISPLAY=bastet:0
     export DISPLAY
     ```

     Если значение переменной среды **DISPLAY** не задано, то по умолчанию применяется значение `unix:0` (консоль). Значение переменной имеет формат *имя*:*число*, где *имя* - имя хоста определенной системы, а *число* - X-сервер в названной системе.

2. Установите значение терминала по умолчанию с помощью следующей команды **stty**:

   ```
   stty sane
   ```

Команда **stty sane** восстанавливает нормальное состояние драйверов терминала. Эта команда выводит код сброса терминала, заданный в файле `/etc/termcap` (или `/usr/share/lib/terminfo`, если есть).

3. Если клавиша Return работает неправильно, сбросьте ее с помощью следующей команды:

   ```
   ^J stty sane ^J
   ```

   Строка ^J соответствует комбинации клавиш Ctrl-J.

Запуск нескольких очередей с помощью переменных среды RT_MPC и RT_GRQ:

Применение нескольких очередей увеличивает возможности процессора по обработке нитей, однако существуют особые ситуации, в которых этот эффект необходимо нейтрализовать.

Если существует только одна очередь выполнения, то нить, выполнение которой возобновляется (возобновляемая нить) другой нитью (возобновляющей нитью), будет использовать ресурсы процессора, освобожденные возобновляющей нитью. В случае применения нескольких очередей выполнения возобновляющая нить может находиться в очереди выполнения другого процессора, который не будет выполнять возобновляющую нить до следующего решения планировки. В результате задержка может составить до 10 мс.

Подобная ситуация возникала в предыдущих выпусках данной операционной системы при использовании опции bindprocessor. Если все процессоры постоянно заняты, и возобновляется выполнение нескольких взаимозависимых нитей, возможно два варианта.

- В первом варианте, в котором применяется одна очередь выполнения, задается значение переменной среды **RT_GRQ=ON**, в результате чего несвязанные выбранные нити убираются из глобальной очереди выполнения.
- Вы также можете выбрать опцию ядра реального времени (введите команду `bosdebug -R on`, затем `bosboot`) и переменную среды **RT_GRQ=ON** для выбранных процессов. Необходимо вести протокол работы систем, чтобы выявить воздействие внесенных изменений.

Учет ресурсов системы

Утилита учета ресурсов системы позволяет собирать данные и создавать отчеты об индивидуальном и групповом использовании различных ресурсов системы.

Эта учетная информация может быть использована для выставления счета пользователям за используемые ими ресурсы системы и для отслеживания выбранных аспектов работы системы. Для помощи при выставлении счетов система учета предоставляет итоговые суммы использования ресурсов, определенные членами административной группы, и если включена команда **chargefee**, коэффициенты оплаты.

Система учета также предоставляет данные для оценки достаточности выделенных в данный момент ресурсов, установки квот и ограничений ресурсов, прогнозов на будущие потребности и заказа новых принтеров и других устройств.

Следующая информация должна помочь понять, как можно реализовать утилиту учета ресурсов в системе.

Понятия, связанные с данным:

“Управление процессами” на стр. 149
Процессы - это объекты, применяемые операционной системой для управления распределением ресурсов. *Нити* предназначены для дополнительного контроля над распределением времени процессора, однако большая часть инструментов управления системой по-прежнему требуют указать процесс, в котором работает нить, а не саму нить.

“Управление рабочей схемой” на стр. 483
Управление рабочей схемой (WLM) предоставляет системному администратору возможность более точно управлять распределением ресурсов между процессами, выполняемым планировщиком и Администратором виртуальной памяти (VMM).

“Учет ресурсов для классов” на стр. 490
Система учета ресурсов AIX позволяет собирать статистическую информацию о ресурсах конкретного пользователя, группы или класса WLM.

Задачи, связанные с данной:

“Устранение переполнения в файловой системе /var” на стр. 452
При заполнении корневой файловой системы `/var` выполните следующие действия.

Информация, связанная с данной:

AIX Version 6.1: Подсистема расширенного учета

Отчеты по учету ресурсов

После сбора информации об использовании различных ресурсов полученные данные обрабатываются и преобразуются в отчеты.

Команды учета ресурсов автоматически преобразуют слишком большие числа в записях в экспоненциальный формат. Экспоненциальным называется следующий формат записи числа:

*Мантисса***e+***Порядок*

*Мантисса***e-***Порядок*

Само число при этом равно *Мантиссе*, умноженной на 10 в степени **+***Порядок* или **-***Порядок*. Например, экспоненциальная форма `1.345e+9` означает 1.345×10^{9} или 1,345,000,000. Соответственно, число `1.345e-9` равно 1.345×10^{-9} или 0.000000001345.

Понятия, связанные с данным:

“Данные о процессе” на стр. 176
В ходе выполнения процесса система учета собирает данные об используемых им ресурсах.

Ежедневные отчеты:

Для создания ежедневных отчетов служит команда **runacct**.

Эта команда записывает итоговые данные в текстовый файл `/var/adm/acct/sum(x)/rprt`*ММДД*. *ММДД* в названии файла указывает месяц и день создания отчета. Создаются отчеты следующих видов:

- Ежедневный отчет
- Ежедневный отчет об использовании ресурсов
- Ежедневный отчет об использовании команд
- Ежемесячный отчет об использовании команд
- Отчет о последнем входе в систему

Ежедневный отчет:

Ежедневные отчеты содержат сведения о времени подключения, процессах, использовании диска, использовании принтеров и сведения для подсчета оплаты.

С помощью команды **acctmerg** можно объединить необработанные учетные данные о времени подключения, процессах, использовании диска, использовании принтеров и сведения для подсчета оплаты в ежедневные отчеты. Команда **runacct** ежедневно вызывает команду **acctmerg**, которая создает следующие файлы:

/var/adm/acct/nite(x)/dacct
: Промежуточный отчет, создаваемый при переполнении одного из входных файлов.

/var/adm/acct/sum(x)/tacct
: Накопительный общий отчет в формате `tacct`. Этот файл используется командой **monacct** для создания текстового итогового отчета за месяц.

Команда **acctmerg** преобразует записи из текстового формата в двоичный и обратно, и объединяет записи из различных источников в единую запись для каждого пользователя. Дополнительная информация о команде **acctmerg** приведена в разделе **acctmerg**.

Первая строка дневного отчета содержит время начала и конца периода сбора информации, а также список событий системного уровня, произошедших за этот период, таких как закрытие, перезагрузка системы и изменение ядра. Указано также полное число минут в отчетном периоде (обычно 1440, если отчет создается каждые 24 часа). В отчете содержится следующая информация:

Элемент	Описание
LINE	Использование консоли, терминалов и псевдотерминалов.
MINUTES	Полное число минут, в течение которых линия связи была занята.
PERCENT	Процент времени, в течение которого линия связи была занята.
# SESS	Число запущенных сеансов
# ON	То же, что и **# SESS**
# OFF	Число выходов из системы плюс число разрывов соединения

Дневной отчет об использовании ресурсов:

Дневной отчет об использовании ресурсов содержит итоговые значения использования ресурсов каждым пользователем (ИД) за отчетный период.

Некоторые значения учитываются отдельно в рабочее и в нерабочее время, задаваемое администратором учета ресурсов в каталоге `/usr/lib/acct/holidays`. В отчете содержится следующая информация:

Элемент	Описание
UID	ИД пользователя
LOGIN NAME	Имя пользователя
CPU (PRIME/NPRIME)	Полное время работы всех процессов пользователя, в минутах
KCORE (PRIME/NPRIME)	Полный объем памяти, занятой работавшими процессами, в килобайт-минутах
CONNECT (PRIME/NPRIME)	Полное время соединения (время работы пользователя в системе), в минутах
DISK BLOCKS	Средний объем дискового пространства, занятый пользователем во всех контролируемых файловых системах
FEES	Итоговый счет, созданный командой **chargefee**
# OF PROCS	Полное число процессов, принадлежащих пользователю
# OF SESS	Число независимых дисплейных сеансов пользователя
# DISK SAMPLES	Число проверок использования дисковой памяти, выполненных за отчетный период. Если величина DISK BLOCKS равна нулю, это значение также будет равно нулю

Ежедневный отчет об использовании команд:

В ежедневном отчете об использовании команд показаны все команды, выполненные за день. В каждой строке содержится информация об одной команде.

Таблица отсортирована по величине TOTAL KCOREMIN (см. объяснение ниже); первая строка содержит итоговую информацию о всех командах. Данные о каждой команде собираются при каждом ее запуске в учетный период. В столбцах таблицы содержится следующая информация:

Элемент	Описание
COMMAND NAME	Имя команды
NUMBER CMDS	Число вызовов команды
TOTAL KCOREMIN	Полный объем памяти, занятый командой, в килобайт-минутах
TOTAL CPU-MIN	Полное время работы команды, в минутах
TOTAL REAL-MIN	Фактическое время, затраченное на выполнение команды, в минутах
MEAN SIZE-K	Средний объем памяти, занимаемый командой во время работы
MEAN CPU-MIN	Среднее время каждого выполнения команды, в минутах
HOG FACTOR	Коэффициент, показывающий, насколько команда загружает центральный процессор во время работы. Равен отношению величины **TOTAL CPU-MIN** к величине **TOTAL REAL-MIN**
CHARS TRNSFD	Число символов, переданных командой при чтении и записи
BLOCKS READ	Число физических блоков, прочитанных и записанных командой

Ежемесячный отчет об использовании команд:

Ежемесячные отчеты об использовании команд создаются командой **monacct**. Они содержат информацию о всех командах, запущенных с момента создания прошлого ежемесячного отчета.

Отчет за месяц содержит те же поля, что и ежедневный отчет об использовании команд.

Отчет о последнем входе в систему:

Отчет о последнем входе в систему содержит по два поля на каждый ИД пользователя. В первом поле указана дата последнего входа в систему. Во втором поле указано имя учетного файла пользователя.

Нулевая дата означает, что пользователь никогда не входил в систему.

Итоговые отчеты об использовании ресурсов:

Можно создать отчет с итоговыми данными об использовании ресурсов.

Для объединения исходных данных учета ресурсов введите команду **sa**. Эта команда считывает исходные данные учета ресурсов, обычно сохраняемые в файле `/var/adm/pacct`, и текущие итоговые данные об использовании ресурсов, сохраняемые в файле `/var/adm/savacct`, если они существуют. Затем она объединяет всю информацию в новый итоговый отчет об использовании ресурсов и очищает файл исходных данных, чтобы освободить место для новых данных.

Предварительные требования

Команде **sa** необходим файл с исходными данными учета ресурсов, например `pacct` (файл учета ресурсов для процессов). Для сбора исходных данных система учета ресурсов должна быть установлена и запущена.

Процедура

Команда **sa** предназначена для объединения информации учета ресурсов, занимаемых процессами, и просмотра или сохранения полученных результатов. Простой запуск команды показывает статистическую информацию о каждом процессе, запущенном за время создания файла `pacct`. Для простого запуска команды введите:

```
/usr/sbin/sa
```

Для объединения информации учета ресурсов в итоговый файл, введите:

```
/usr/sbin/sa -s
```

Команда **sa** поддерживает большое количество дополнительных флагов, задающих способ обработки и вывода информации учета ресурсов. Дополнительная информация приведена в описании команды **sa**.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Ежемесячный отчет:

Можно создать ежемесячный отчет об учете ресурсов.

Команда **monacct** вызывается демоном **cron** и создает следующие файлы:

Элемент	Описание
`/var/adm/acct/fiscal`	Итоговый отчет за весь период, создаваемый командой **monacct** из отчета `/var/adm/acct/sum/tacct`. Команда **monacct** может выполняться один раз в месяц или в конце каждого отчетного периода.

Отчеты о времени соединения:

Отчеты включают в себя сведения о входе в систему, выходе из системы, завершении работы системы и инфор мацию о последнем входе в систему.

Команда **runacct** вызывает команды **acctcon1** и **acctcon2** для обработки записей о входе и выходе из системы, а также записей о завершении работы системы, хранящихся в файле `/var/adm/wtmp`. Команда **acctcon1** преобразует эти записи в записи о сеансах и заносит результат в файл `/var/adm/acct/nite(x)/lineuse`. Команда `acctcon2` преобразует записи о сеансах в итоговую запись учета ресурсов и помещает ее в файл `/var/adm/logacct`. Эта запись применяется командой **acctmerg** для составления ежедневных отчетов. Сведения об этих командах приведены в описании команд **runacct**, **acctcon1** и **acctcon2**.

Для создания отчета об использовании линии связи в файле `/var/adm/acct/nite(x)/lineuse` запустите из командной строки команду **acctcon1** с флагом **-l**. Для создания полного отчета о сеансе за указанный период времени в файле `/var/adm/acct/nite(x)/reboots` введите команду **acctcon1** с флагом **-o**.

Команда **lastlogin** создает отчет о дате последнего входа в систему каждого из пользователей. Информация о команде **lastlogin** приведена в разделе **lastlogin**.

Понятия, связанные с данным:

“Данные учета времени соединения” на стр. 175
Данные о времени соединения собираются с помощью команды **init** и **login**.

“Учет использования диска” на стр. 177
Большая часть информации об использовании ресурсов относится к их расходу. Команда **dodisk**, запускаемая демоном **cron** в соответствии с расписанием, добавляет в файл `/var/adm/acct/nite(x)/dacct` записи об объеме дисковой памяти, занятой различными пользователями.

Отчет об использовании диска:

Записи об использовании дисковой памяти, собранные в файле `/var/adm/acct/nite(x)/dacct`, добавляются в ежедневный отчет командой **acctmerg**.

Информация о команде **acctmerg** приведена в разделе **acctmerg**.

Отчет об использовании принтера:

Текстовые записи файла `/var/adm/qacct` могут быть преобразованы в итоговые записи учета ресурсов и добавлены в ежедневный отчет командой **acctmerg**.

Информация о команде **acctmerg** приведена в разделе **acctmerg**.

Понятия, связанные с данным:

“Данные об учете использования принтере” на стр. 177
Данные об использовании принтеров собираются совместно командой **enq** и демоном работы с очередью.

Отчет об оплате:

Если вы применяете команду **chargefee** для создания счетов за такие услуги, как восстановление файлов, консультации и использование расходных материалов, в файле `/var/adm/fee` создаются текстовые записи учета ресурсов. Этот файл добавляется в ежедневный отчет командой **acctmerg**.

Сведения о командах **chargefee** и **acctmerg** приведены в разделах **chargefee** и **acctmerg**.

Понятия, связанные с данным:

“Данные об оплате” на стр. 177
В файле `/var/adm/fee` можно создать итоговую запись учета ресурсов в формате ASCII.

Суммарные отчеты об использовании ресурсов:

Обычно такие отчеты создаются ежемесячно командой **monacct**.

Полученный отчет сохраняется в файле `/var/adm/acct/fiscal(x)/fiscrptMM`, где *MM* - месяц, в котором была запущена команда **monacct**. Отчет содержит ту же информацию, что и ежедневные отчеты, но накопленную за весь месяц.

Отчеты о работе системы и у чете ресурсов:

Вы можете создать отчет, содержащий сведения о работе системы учета ресурсов.

Для того чтобы создать отчет о работе системы, выполните команду **prtacct**. Эта команда печатает данные из общего файла учета ресурсов (файл в формате `tacct`) в отформатированном виде. Общие файлы учета ресурсов содержат ежедневные отчеты с информацией о времени соединения, времени обработки, использовании дисков и принтера.

Предварительные требования

В качестве входного файла команда **prtacct** получает файл в формате `tacct`. Следовательно, в системе должна быть настроена и запущена система учета ресурсов.

Процедура

Создайте отчет о работе системы, выполнив команду:

```
prtacct -f спецификации -v заголовок файл
```

Спецификация - это список номеров или диапазонов полей, элементы которого разделяются запятыми. Этот список применяется командой **acctmerg**. Необязательный флаг **-v** позволяет получить подробный вывод, в котором числа с плавающей точкой отображаются с максимальной точностью. *Заголовок* - это строка, которая должна быть показана в качестве заголовка создаваемого отчета. Этот параметр необязательный. *Файл* - это полное имя общего файла учета ресурсов, который должен применяться в качестве входного файла. В этом параметре можно задать несколько файлов.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Поддержка имен пользователей, состоящих более, чем из восьми символов:

Для обеспечения обратной совместимости со всеми сценариями длинные имена пользователей по умолчанию не поддерживаются приложениями учета. Все идентификаторы пользователей усекаются до первых восьми символов.

Для поддержки длинных имен в большинство команд добавлен дополнительный флаг **-X**, позволяющий задавать и выводить длинные идентификаторы пользователей (в формате ASCII и в двоичном формате). Кроме того, если включена поддержка длинных имен пользователей, то команды и сценарии будут обрабатывать файлы в каталогах `/var/adm/acct/sumx` , `/var/adm/acct/nitex` и `/var/adm/acct/fiscalx`, а не в каталогах `/var/adm/acct/sum`, `/var/adm/acct/nite` и `/var/adm/acct/fiscal`.

Команды учета ресурсов

Команды учета ресурсов выполняют несколько различных задач.

Примеры команд, выполняющих разные задачи:

- Сбор данных или создание отчетов об использовании ресурсов определенного типа: времени соединения, дисков, принтеров, процессов или команд.
- Вызов других команд. Например, команда **runacct**, автоматически запускаемая демоном **cron**, в свою очередь вызывает множество других команд, которые собирают и обрабатывают данные об использовании ресурсов, а также создают отчеты. Для автоматического сбора сведений об учете ресурсов настройте демон **cron** на запуск команды **runacct**. Дополнительная информация о настройке демона **cron** на запуск других команд через определенные промежутки времени приведена в описании команды **crontab**. Сведения об этих командах приведены в описании команды **runacct**, демона **cron** daemon и **crontab**.
- Обслуживание активных файлов и обеспечение их целостности.
- Выполнение задач пользователями из группы adm, например, просмотр отдельных записей, путем ввода команд.
- Просмотр различной информации. Есть только одна команда, **acctcom**, предназначенная для просмотра итоговой информации об учете ресурсов процесса.

Автоматически запускаемые команды:

Некотоые команды выполняют автоматический сбор данных учета ресурсов.

Демон **cron** автоматически запускает несколько команд сбора данных об учете ресурсов. К ним относятся:

runacct
: Выполняет главную процедуру дневного учета ресурсов. Команда **runacct** обычно запускается демоном **cron** в нерабочие часы. Она вызывает другие команды учета ресурсов для обработки файлов данных и создания итоговых записей учета ресурсов и команд, отсортированных по имени пользователя. Кроме того, эта команда вызывает команду **acctmerg** для создания файлов с итоговым ежедневным отчетом, и команду **ckpacct** для обслуживания файлов данных о работе системы.

ckpacct
: Ограничивает размер файла `pacct`. Обычно удобнее иметь несколько небольших файлов `pacct`, поскольку иногда возникает необходимость перезапустить процедуру **runacct** после сбоя при обработке записей. Команда **ckpacct** вычисляет размер файла `/var/adm/pacct`, и, если он превышает 500 блоков, вызывает команду **turnacct switch** для временного выключения учета ресурсов процессов. Данные перемещаются в новый файл `pacct - /var/adm/pacct` *x*. (*x* - порядковый номер файла `pacct`). Если количество свободных блоков на диске становится меньше 500, то команда **ckpacct** выключает учет ресурсов процессов с помощью команды **turnacct off**.

dodisk Вызывает команду **acctdisk**, а затем команду **diskusg**или **acctdusg** для создания записей учета использования дисковой памяти в файле `/var/adm/acct/nite/dacct`. Позднее эти данные добавляются в ежедневные отчеты.

dodisk Вызывает команду **acctdisk**, а затем команду **diskusg**или **acctdusg** для создания записей учета использования дисковой памяти в файле /var/adm/acct/nite/dacct. Позднее эти данные добавляются в ежедневные отчеты.

monacct
Создает из дневных отчетов отчет за больший промежуток времени.

sa1 Собирает и сохраняет двоичные данные в файле /var/adm/sa/sa *dd*, где *dd* - текущее число месяца.

sa2 Записывает ежедневный отчет в файл /var/adm/sa/sa*dd*, где *dd* - текущее число месяца. Эта команда удаляет отчеты из файлов /var/adm/sa/sa*dd*, созданные более недели назад.

Приведенные ниже команды также запускаются автоматически, но не демоном **cron**:

startup
Если команда /etc/rc добавлена в файл **startup**, то она выполняет процедуры запуска системы учета ресурсов.

shutacct
Вызывает команду **acctwtmp** для записи в файл /var/adm/wtmp строки о прекращении учета ресурсов времени. Затем она вызывает команду **turnacct off** для выключения учета ресурсов процессов.

Команды клавиатуры:

Пользователь из группы adm может выполнить следующие команды учета ресурсов.

ac Печатает записи о времени соединения. Эта команда добавлена для совместимости с системами BSD.

acctcom
Показывает итоговые записи учета ресурсов для процессов. Эта команда доступна всем пользователям.

acctcon1
Показывает итоговые записи учета ресурсов о времени соединения. Необходимо указать флаг **-l** или **-o**.

accton Включает и выключает учет ресурсов для процессов.

chargefee
Создает для пользователя счет на оплату проделанной им работы. Эти счета добавляются к ежедневному отчету командой **acctmerg**.

fwtmp Преобразует файлы между двоичным и текстовым форматами.

last Показывает информацию о предыдущих входах в систему. Эта команда добавлена для совместимости с системами BSD.

lastcomm
Показывает информацию о последних выполненных командах. Эта команда добавлена для совместимости с системами BSD.

lastlogin
Показывает время последнего входа в систему для каждого пользователя.

pac Подготавливает записи учета ресурсов принтера или графопостроителя. Эта команда добавлена для совместимости с системами BSD.

prctmp
Показывает запись для сеанса.

prtacct
Показывает итоговые файлы учета ресурсов.

sa Объединяет исходную информацию учета ресурсов для упрощения управления большими объемами данных. Эта команда добавлена для совместимости с системами BSD.

sadc Создает отчет о работе локальной системы, например, о работе с буфером, дисками, магнитной лентой, терминалами и файлами.

sar Записывает в стандартный вывод содержимое некоторых счетчиков деятельности операционной системы. Команда **sar** создает отчет только о локальной системе.

time Создает отчет о полном, пользовательском и системном времени выполнения команды.

timex Показывает затраченное, пользовательское время и время выполнения команды в секундах.

Понятия, связанные с данным:

“Сбор и отчет о системных данных” на стр. 175
Можно настроить систему для автоматического сбора данных и создания отчетов.

Файлы учета ресурсов

Система учета ресурсов работает со следующими каталогами: `/usr/sbin/acct`, в котором находятся все программы на языке C и процедуры оболочки, необходимые для работы этой системы, и `/var/adm`, в котором хранятся данные, отчеты и итоговые файлы.

Файлы с данными учета ресурсов принадлежат группе adm; все файлы данных о работе системы (такие как `wtmp` и `pacct`) находятся в домашнем каталоге этой группы, `/var/adm`.

Файлы учета ресурсов:

Следующие файлы располагаются в каталоге `/var/adm`.

Элемент	Описание
`/var/adm/diskdiag`	Диагностическая информация, создаваемая программами учета использования диска.
`/var/adm/dtmp`	Вывод команды **acctdusg**.
`/var/adm/fee`	Вывод команды **chargefee** в записях ASCII `tacct`
`/var/adm/pacct`	Файл учета ресурсов активных процессов.
`/var/adm/wtmp`	Файл учета ресурсов активных процессов.
`/var/adm/Spacct` *.mmdd*	Файлы учета ресурсов процессов за указанный месяц и день (*mmdd*), созданные командой **runacct**.

Итоговые файлы и отчеты об учете ресурсов:

Перед включением системы учета требуется создать некоторые подкаталоги.

Файлы отчетов и итоговые файлы создаются в каталоге `/var/adm/acct`. Перед включением системы учета ресурсов должны быть созданы перечисленные ниже каталоги.

`/var/adm/acct/nite(x)`
Содержит файлы, ежедневно обновляемые командой **runacct**.

`/var/adm/acct/sum(x)`
Содержит итоговые файлы, ежедневно обновляемые командой **runacct**.

`/var/adm/acct/fiscal(x)`
Содержит файлы с ежемесячными отчетами, создаваемые командой **monacct**.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Запуск команды runacct для учета ресурсов:

Можно запустить команду **runacct**.

Предварительные требования

1. Должна быть установлена система учета ресурсов.
2. У вас должны быть права пользователя root или члена группы adm.

Примечания:

1. Запуск команды **runacct** без параметров означает первый запуск команды в этот день. При повторном запуске команды **runacct** укажите параметр *ммдд*, соответствующий текущему месяцу и дню. Если состояние не задано, то программа **runacct** определяет точку входа с помощью файла `/var/adm/acct/nite(x)/statefile`. Для того, чтобы переопределить значение из файла `/var/adm/acct/nite(x)/statefile`, укажите в командной строке параметр состояния.
2. При выполнении следующей задачи рекомендуется указать полное имя команды `/usr/sbin/acct/runacct`, а не **runacct**.

Процедура

Для запуска команды **runacct** введите в командной строке:

```
nohup runacct 2> \
/var/adm/acct/nite/accterr &
```

В результате команда, выполняемая в фоновом режиме, будет игнорировать сигналы **INTR** и **QUIT**. Стандартный вывод сообщений об ошибках будет перенаправлен в файл `/var/adm/acct/nite/accterr`.

Повторный запуск команды runacct для учета ресурсов:

Если команда **runacct** завершилась неудачно, можно перезапустить ее.

Для этой процедуры имеются следующие предварительные требования:

- Должна быть установлена система учета ресурсов.
- У вас должны быть права пользователя root или члена группы adm.

Примечание: Ниже перечислены наиболее распространенные причины сбоев команды **runacct**:

- Сбой системы
- Отсутствие свободного места в файловой системе `/usr`
- Файл `/var/adm/wtmp` содержит записи с неправильными метками даты.

Если команда **runacct** завершилась неудачно, то выполните следующие действия:

1. Просмотрите сообщения об ошибках в файле `/var/adm/acct/nite(x)/active` *ММДД*.
2. Если в каталоге `acct/nite` есть активные и заблокированные файлы, то просмотрите файл `accterr`, в который перенаправляется вывод сообщений об ошибках, когда демон **cron** вызывает команду **runacct**.
3. Исправьте ошибку.
4. Запустите команду **runacct.**
5. Для перезапуска команды **runacct** для заданной даты введите следующую команду:

   ```
   nohup runacct 0601 2>> \
   /var/adm/acct/nite/accterr &
   ```

   В приведенном примере программа **runacct** будет перезапущена для 1 июня (`0601`). Команда **runacct** считывает информацию о начальном состоянии из файла `/var/adm/acct/nite/statefile`. Вывод сообщений об ошибках добавляется к файлу `/var/adm/acct/nite/accterr`.

6. Для перезапуска команды **runacct** в заданную дату с начальным состоянием MERGE введите в командной строке:

```
nohup runacct 0601 MERGE 2>> \
/var/adm/acct/nite/accterr &
```

runacct, создаваемые файлы:

Команда **runacct** создает отчеты и итоговые файлы.

Ниже описаны некоторые файлы отчетов и итоговые файлы, создаваемые командой **runacct**:

Элемент	Описание
`/var/adm/acct/nite(x)/lineuse`	Статистика использования ресурсов для каждой линии связи с терминалом. Этот отчет особенно полезен при диагностике неполадок в линиях связи. Если отношение количества операций выхода из системы к количеству операций входа в систему превышает 3:1, то велика вероятность повреждения линии связи.
`/var/adm/acct/nite(x)/daytacct`	Итоговый файл учета ресурсов за предыдущий день.
`/var/adm/acct/sum(x)/tacct`	Содержит информацию, полученную из создаваемых ежедневно файлов `nite/daytacct`; применяется для создания счетов. Команда **monacct** очищает этот файл каждый месяц или в конце заданного отчетного периода.
`/var/adm/acct/sum(x)/cms`	Содержит информацию, полученную из создаваемых ежедневно итоговых файлов. Команда **monacct** считывает и очищает двоичную версию этого файла. Текстовая версия хранится в файле `nite/cms`.
`/var/adm/acct/sum(x)/daycms`	Содержит дневной отчет об использовании команд. Текстовая версия этого файла хранится в `nite/daycms`.
`/var/adm/acct/sum(x)/loginlog`	Содержит запись о последнем применении ИД пользователя.
`/var/adm/acct/sum(x)/rprt` *mmdd*	Содержит копию отчета, ежедневно создаваемого командой **runacct**.

В каталоге /var/adm/acct/nite(x) хранятся следующие файлы::

Следующие файлы располагаются в каталоге `/var/adm/acct/nite(x)`.

Элемент	Описание
`active`	Применяется командой **runacct** для вывода информации о состоянии, предупреждений и сообщений об ошибках. Файл `active`. *ммдд* - это копия файла `active`, создаваемая командой **runacct** при обнаружении ошибки.
`cms`	Содержит итоговую текстовую информацию об использовании команд, применяемую командой **prdaily**.
`ctacct`.*ммдд*	Итоговые записи учета ресурсов времени соединения.
`ctmp`	Записи о соединении для сеансов.
`daycms`	Содержит итоговую текстовую информацию об использовании команд за день, применяемую командой **prdaily**.
`daytacct`	Итоговые записи учета ресурсов за день.
`dacct`	Итоговые записи учета ресурсов дисковой памяти, созданные командой **dodisk**.
`accterr`	Диагностический вывод, созданный при выполнении команды **runacct**.
`lastdate`	Дата последнего выполнения команды **runacct** в формате *дата +%$м%д*.
`lock1`	Применяется для управления доступом к команде **runacct**.
`lineuse`	Отчет об использовании линий терминалов, применяемый командой **prdaily**.
`log`	Диагностический вывод команды **acctcon1**.
`log`*ммдд*	Совпадает с файлом **log** на момент обнаружения ошибки командой **runacct**.
`reboots`	Содержит начальные и конечные даты из файла `wtmp`, а также список с указанием моментов перезапуска системы.
`statefile`	Применяется для записи текущего состояния при работе команды **runacct**.
`tmpwtmp`	Файл `wtmp`, исправленный командой **wtmpfix**.
`wtmperror`	Содержит сообщения об ошибках **wtmpfix**.
`wtmperr`*ммдд*	Совпадает с файлом `wtmperror` на момент обнаружения ошибки командой **runacct**.
`wtmp`.*ммдд*	Файл `wtmp` за предыдущий день. Этот файл удаляется при очистке командой **runacct**.

В каталоге /var/adm/acct/sum(x) хранятся следующие файлы:

Следующие файлы располагаются в каталоге `/var/adm/acct/sum(x)`.

Элемент	Описание
`cms`	Итоговый файл учета использования команд за текущий отчетный период, в двоичном формате.
`cmsprev`	Совпадает с итоговым файлом учета использования команд, за исключением последнего обновления.
`daycms`	Итоговый файл учета использования команд за предыдущий день, в двоичном формате.
`lastlogin`	Файл, создаваемый командой **lastlogin**.
`pacct.`*ммдд*	Объединенная версия всех файлов `pacct` за *ммдд*. Этот файл удаляется командой **remove**. Информация о команде **remove** приведена в разделе **remove**.
`rprt`*ммдд*	Сохраненный вывод команды **prdaily**.
`tacct`	Итоговый файл учета ресурсов за текущий отчетный период.
`tacctprev`	Совпадает с файлом `tacct`, за исключением последнего обновления.
`tacct`*ммдд*	Итоговый файл учета ресурсов за *ммдд*.

В каталоге /var/adm/acct/fiscal(x) хранятся следующие файлы::

Следующие файлы располагаются в каталоге `/var/adm/acct/fiscal(x)`:

Элемент	Описание
`cms`*?*	Итоговый файл с информацией об использовании команд за отчетный период, задаваемый символом *?*, в двоичном формате.
`fiscrpt`*?*	Отчет, аналогичный создаваемому командой **prdaily**, но за отчетный период, задаваемый символом *?*, в двоичном формате.
`tacct`*?*	Итоговый файл учета ресурсов за определенный период, задаваемый символом *?*, в двоичном формате.

Форматы файла учета ресурсов:

В следующей таблице приведены сведения о форматах и выводе файла учета ресурсов.

Элемент	Описание
`wtmp`	Файл учета ресурсов для активных процессов. Формат файла `wtmp` описан в файле `utmp.h`. Информация о файле `utmp.h` приведена в разделе `utmp.h`.
`ctmp`	Записи о времени соединения. Формат вывода описан в файле `ctmp.h`.
`pacct*`	Записи учета ресурсов для активных процессов. Формат вывода описан в файле `/usr/include/sys/acct.h`.
`Spacct*`	Записи учета ресурсов процессов за указанную дату (*ммдд*), созданные командой **runacct**. Формат файлов описан в файле `sys/acct.h`.
`daytacct`	Итоговые записи учета ресурсов за день. Файл записывается в формате `tacct`.
`sum/tacct`	Двоичный файл с информацией, полученной из ежедневных итоговых записей об использовании команд. Формат этого файла описан в файле `/usr/include/sys/acct.h`.
`ptacct`	Объединенная версия файлов `pacct`. Формат этих файлов определен в файле `tacct`.
`ctacct`	Итоговые записи учета ресурсов о времени соединения. Формат вывода определен в файле `tacct`.
`cms`	Итоговые записи об использовании команд, применяемые командой **prdaily**, в двоичном формате. Текстовая версия хранится в файле `nite/cms`.
`daycms`	Ежедневно создаваемый отчет об использовании команд, применяемый командой **prdaily**, в двоичном формате. Текстовая версия хранится в файле `nite/daycms`.

Управление учетом ресурсов системы

Для учета ресурсов системы можно выполнить несколько задач. В число этих задач входит настройка системы учета ресурсов системы, отображение использования CPU и отображение процессов учета ресурсов.

Настройка системы учета ресурсов:

Вы можете настроить систему учета ресурсов.

Для выполнения этой процедуры необходимы права доступа root.

Ниже описаны действия, которые необходимо выполнить для настройки системы учета ресурсов. Дополнительная информация приведена в описании команд, применяемых в этих действиях.

1. Введите команду **nulladm**, предоставляющую следующие права доступа к файлам: на чтение (r) и запись (w) - владельцу файла и группы файла, и на чтение (r) - всем остальным пользователям:

   ```
   /usr/sbin/acct/nulladm wtmp pacct
   ```

   Указанная команда предоставляет необходимые права доступа к файлам `pacct` и `wtmp`.

2. Укажите в файле `/etc/acct/holidays` начало и конец рабочего дня, а также все праздничные дни.

   Примечание: Любая строка этого файла, начинающаяся с символа звездочки (*), считается комментарием.

   a. Для задания рабочего времени заполните поля в первой строке данных (в первой строке, не являющейся комментарием), указывая время в 24-часовом формате. Эта строка состоит из трех полей (по четыре цифры в каждом), имеющих следующее значение:

      1) Текущий год
      2) Начало рабочего дня (*ччмм*)
      3) Конец рабочего дня (*ччмм*)

      Начальные пробелы игнорируются. Полночь можно указать как числом 0000, так и числом 2400.

      Например, для задания в 2000 году рабочего времени с 08:00 по 17:00 введите:

      ```
      2000  0800  1700
      ```

   b. В следующих строках задайте праздники. Каждая строка состоит из четырех полей, в следующем порядке:

      1) День года
      2) Месяц
      3) День месяца
      4) Название праздника

      В поле День года указывается номер дня, на который выпадает праздник в этом году, в виде числа в диапазоне от 1 до 365 (в високосный год - до 366). Например, 1 февраля - это 32-й день года. Оставшиеся три поля рассматриваются системой как комментарий.

      Ниже приведен пример двух таких строк:

      ```
        1  Янв  1  Новый год
      332  Ноя 28  День благодарения
      ```

3. Включите процесс учета ресурсов, добавив в файл `/etc/rc` следующую строку, либо удалив из этой строки символ #, если она уже существует в виде комментария:

   ```
   /usr/bin/su - adm -c /usr/sbin/acct/startup
   ```

   Процедура **startup** записывает время начала учета ресурсов и очищает файлы учета ресурсов, оставшиеся от предыдущего дня.

4. Обозначьте каждую файловую систему, которую необходимо включить в отчет о дисках, добавив следующую строку в файл `/etc/filesystems` настройки файловой системы:

   ```
   account = true
   ```

5. Задайте файл данных принтера, указав его в разделе queue файла `/etc/qconfig`:

   ```
   acctfile = /var/adm/qacct
   ```

6. Войдя в систему под именем пользователя adm, создайте каталоги `/var/adm/acct/nite`, `/var/adm/acct/fiscal` и `/var/adm/acct/sum`, в которых будут храниться записи ежедневных и суммарных отчетов.

   ```
   su - adm
   cd /var/adm/acct
   mkdir nite fiscal sum
   exit
   ```

В случае длинных имен пользователей используйте следующие команды:

```
su - adm
cd /var/adm/acct
mkdir nitex fiscalx sumx
exit
```

7. Для настройки автоматического выполнения процедур ежедневного учета ресурсов можно отредактировать файл /var/spool/cron/crontabs/adm и добавить в него команды **dodisk**, **ckpacct** и **runacct**. Например:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   5 * * * * /usr/sbin/acct/ckpacct
   0 4 * * 1-6 /usr/sbin/acct/runacct
               2>/var/adm/acct/nite/accterr
   ```

   В случае длинных имен добавьте вместо них следующие строки:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk -X
   5 * * * * /usr/sbin/acct/ckpacct
   0 4 * * 1-6 /usr/sbin/acct/runacct -X
               2>/var/adm/acct/nitex/accterr
   ```

   Первая строка предназначена для запуска учета использования дисковой памяти в 02:00 (0 2) каждый четверг (4). Вторая строка запускает проверку целостности активных файлов данных в 5-ю минуту каждого часа (5 *) каждый день (*). В третьей строке указаны процедуры учета ресурсов, которые будут запускаться для обработки файлов данных о работе системы в 04:00 (0 4) каждого дня, кроме воскресенья (1-6). Если указанные часы не соответствуют режиму работы вашей системы, замените их своими.

   Примечание: Для изменения файла /var/spool/cron/crontabs/adm необходимы права доступа пользователя root.

8. Настройте автоматический запуск процедур создания отчетов об использовании ресурсов за месяц, добавив в файл **monacct** команду /var/spool/cron/crontabs/adm. Например, введите:

   ```
   15 5 1 * * /usr/sbin/acct/monacct
   ```

   В случае длинных имен добавьте следующую строку:

   ```
   15 5 1 * * /usr/sbin/acct/monacct -X
   ```

   При указании времени запуска этой процедуры учтите, что создание отчета за месяц занимает определенное время. В указанном примере эта процедура будет запускаться в 05:15 первого числа каждого месяца.

9. Для отправки отредактированного файла cron введите:

   ```
   crontab /var/spool/cron/crontabs/adm
   ```

Понятия, связанные с данным:

“Команды для автоматической очистки файловой системы” на стр. 374
Для очистки файловой системы путем удаления ненужных файлов вызовите команду **skulker**.

“Сбор и отчет о системных данных” на стр. 175
Можно настроить систему для автоматического сбора данных и создания отчетов.

“Отчеты о работе системы и у чете ресурсов” на стр. 161
Вы можете создать отчет, содержащий сведения о работе системы учета ресурсов.

“Итоговые отчеты об использовании ресурсов” на стр. 159
Можно создать отчет с итоговыми данными об использовании ресурсов.

Задачи, связанные с данной:

“Ограничение доступа пользователей к определенным каталогам” на стр. 374
Вы можете освободить пространство на диске и поддерживать его свободным с помощью ограничения доступа к каталогам и отслеживания использования диска.

“Устранение переполнения пользовательской файловой системы” на стр. 447
При переполнении пользовательской файловой системы выполните следующие действия.

“Просмотр времени выполнения активных процессов системы учета” на стр. 171
Можно просмотреть время работы процессов системы учета.

“Просмотр времени выполнения завершенных процессов системы учета” на стр. 171
Можно просмотреть время выполнения готовых процессов.

“Просмотр потребления CPU для каждого процесса учета ресурсов” на стр. 172
Для просмотра форматированных отчетов об использовании CPU различными пользователями воспользуйтесь командой **acctprc1**.

“Просмотр информации об использовании CPU пользователями” на стр. 172
Для просмотра информации об использовании CPU различными пользователями воспользуйтесь комбинацией команд **acctprc1** и **prtacct**.

“Отображает отчет об использовании принтера или графопостроителя” на стр. 174
Вы можете просмотреть записи системы учета об использовании принтера или графопостроителя с помощью команды **pac**.

Ссылки, связанные с данной:

“Итоговые файлы и отчеты об учете ресурсов” на стр. 164
Перед включением системы учета требуется создать некоторые подкаталоги.

Просмотр сведений о работе системе:

Для просмотра отформатированных данных о работе системы выполните команду **sar**.

Для просмотра статистических данных о работе системы должен быть запущен процесс **sadc**.

Примечание: Обычно для запуска команды **sadc** в корневой файл `crontab` помещается запись о команде **sa1**. Команда **sa1** - это процедура оболочки, аналогичная команде **sadc**, которая предназначена для работы с демоном **cron**.

Для просмотра основной информации о работе системы введите:

```
sar 2 6
```

где первое число задает частоту обновления информации, а второе число - количество сообщений, которое нужно показать. Ниже приведен пример вывода этой команды:

```
    arthurd 2 3 000166021000    05/28/92

14:03:40    %usr    %sys    %wio   %idle
14:03:42       4       9       0      88
14:03:43       1      10       0      89
14:03:44       1      11       0      88
14:03:45       1      11       0      88
14:03:46       3       9       0      88
14:03:47       2      10       0      88

Среднее        2      10       0      88
```

В команде **sar** предусмотрены флаги для просмотра расширенного набора статистических данных. Для просмотра всех статистических данных укажите флаг **-A**. Список типов статистических данных и флагов, которые применяются для их просмотра, приведен в описании команды **sar**.

Примечание: Для того чтобы в файл `/var/adm/sa/sa`*dd* ежедневно записывался отчет о работе системы, поместите в файл `crontab` запись о команде **sa2**. Команда **sa2** - это процедура оболочки, аналогичная команде **sar**, которая работает вместе с демоном **cron**.

Просмотр сведений о работе системы при выполнении команды:

Можно просмотреть отформатированные данные о работе системы во время выполнения определенной команды.

Если вы указываете в команде **timex** флаг **-o** или **-p**, то предварительно должен быть запущен учет ресурсов системы.

С помощью команд **time** и **timex** можно просмотреть отформатированные данные о работе системы во время выполнения определенной команды.

Для того чтобы узнать время, прошедшее с момента запуска команды, время пользователя и системное время выполнения, введите:

```
time команда
```

ИЛИ

```
timex команда
```

Для просмотра всей информации о работе системы (всех данных из отчета команды **sar**) во время выполнения отдельной команды введите:

```
timex -s команда
```

В команде **timex** предусмотрено два флага. Флаг **-o** позволяет узнать общее число блоков памяти, которые были считаны или записаны указанным процессом и всеми его дочерними процессами. Флаг **-p** позволяет просмотреть список всех записей учета ресурсов об указанном процессе и всех его дочерних процессах.

Просмотр времени выполнения активных процессов системы учета:

Можно просмотреть время работы процессов системы учета.

Команда **acctcom** считывает входные данные из общей формы записей учета (формат файла `acct`). При этом подразумевается, что процесс учета запущен или вызывался ранее.

В команде **ps** предусмотрен набор флагов, позволяющих настраивать формат показываемой информации.

Для просмотра полного списка всех активных процессов ядра введите:

```
ps -ef
```

Также можно просмотреть список всех процессов, связанных с терминалами. Для этого введите:

```
ps -al
```

В обоих случаях, в нескольких колонках показана информация о каждом процессе, в том числе текущее время CPU в минутах и секундах.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Просмотр времени выполнения завершенных процессов системы учета:

Можно просмотреть время выполнения готовых процессов.

Команда **acctcom** считывает входные данные из общей формы записей учета (формат файла `acct`). При этом подразумевается, что процесс учета запущен или вызывался ранее.

Функции учета процессов вызываются командой **startup**, которая обычно запускается при инициализации системы из файла `/etc/rc` file. Во время выполнения функций учета процессов в файл `/var/adm/pacct` (файл записей общего учета) заносится запись для каждого завершенного процесса, включая время его запуска и завершения. Вы можете просмотреть данные о времени, записанные в файл `pacct`, с помощью команды **acctcom**. Для этой команды предусмотрен набор флагов, обеспечивающих гибкость выбора процессов для просмотра.

Например, для просмотра всех процессов, которые CPU выполнял не менее указанного времени (в секундах), укажите флаг **-O**, введите:

```
acctcom -O 2
```

Будут показаны записи для каждого процесса, выполнявшегося не менее 2 секунд. Если вы не укажете входной файл, то команда **acctcom** получит данные из каталога /var/adm/pacct.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Просмотр потребления CPU для каждого процесса учета ресурсов:

Для просмотра форматированных отчетов об использовании CPU различными пользователями воспользуйтесь командой **acctprc1**.

Команда **acctprc1** считывает входные данные из общей формы записей учета (формат файла acct). При этом подразумевается, что процесс учета запущен или вызывался ранее.

Для создания отчета об использовании CPU каким-либо процессом введите:

```
acctprc1 </var/adm/pacct
```

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Просмотр информации об использовании CPU пользователями:

Для просмотра информации об использовании CPU различными пользователями воспользуйтесь комбинацией команд **acctprc1** и **prtacct**.

Команда acctprc1 считывает входные данные из общей формы записей учета (формат файла acct). При этом подразумевается, что процесс учета запущен или вызывался ранее.

Для просмотра использования CPU пользователями выполните следующие действия:

1. Создайте файл вывода с информацией об использовании CPU процессами, введя следующую команду:

   ```
   acctprc1 </var/adm/pacct >out.file
   ```

   Файл /var/adm/pacct - это файл вывода по умолчанию, используемый при обработке учетных записей. Вместо него можно указать архивный файл pacct.
2. Создайте двоичный файл учетных записей с помощью предыдущей процедуры, введя следующую команду:

   ```
   acctprc2 <out.file >/var/adm/acct/nite/daytacct
   ```

   Примечание: Для создания итоговой записи за день (/var/adm/acct/sum(x)/tacct) файл daytacct объединяется с другими итоговыми записями командой **acctmerg**.
3. С помощью команды prtacct можно получить сгруппированный по пользователям отчет об использовании CPU:

   ```
   prtacct </var/adm/acct/nite/daytacct
   ```

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Просмотр сведений о времени подключения к системе:

Вы можете просмотреть информацию о времени подключения к системе для всех пользователей, отдельной группы или конкретного пользователя с помощью команды **ac**.

Команда **ac** выбирает информацию о пользователях из файла `/var/adm/wtmp`, поэтому этот файл обязательно должен существовать. Если этот файл не был создан, то будет возвращено следующее сообщение об ошибке:

```
Отсутствует /var/adm/wtmp
```

При переполнении этого файла создаются дополнительные файлы `wtmp`. Вы можете просмотреть информацию о времени соединения, хранящуюся в этих файлах, указав их с флагом **-w**. Дополнительная информация о команде **ac** приведена в разделе **ac**.

Для просмотра общего времени соединения для всех пользователей введите:

```
/usr/sbin/acct/ac
```

Эта команда показывает десятичное число, являющееся суммой полного времени соединения (в минутах) для всех пользователей, входивших в систему за время существования текущего файла `wtmp`.

Для просмотра полного времени соединения для одного или нескольких пользователей введите:

```
/usr/sbin/acct/ac Пользователь1 Пользователь2 ...
```

Эта команда показывает десятичное число, являющееся суммой полного времени соединения (в минутах) для указанных вами пользователей, если они входили в систему за время существования текущего файла `wtmp`.

Для просмотра времени соединения для конкретного пользователя вместе с общим временем соединения введите:

```
/usr/sbin/acct/ac -p Пользователь1 Пользователь2 ...
```

Эта команда показывает десятичное число, являющееся значением полного времени соединения (в минутах) для каждого указанного пользователя, если он входил в систему за время существования текущего файла `wtmp`. Она также показывает значение суммарного времени соединения для всех указанных пользователей. Если в команде не были заданы имена пользователей, то в списке будет показана информация о всех пользователях, вошедших в систему за время существования файла `wtmp`.

Просмотр информации об использовании диска:

Для просмотра информации об использовании диска служит команда **acctmerg**.

Команде **acctmerg**, вызываемой для просмотра информации об использовании диска, необходимы входные данные из файла `dacct` (учет дисковых ресурсов). Сбор записей об использовании диска выполняется командой **dodisk**.

Для просмотра данных об использовании дискового пространства введите:

```
acctmerg -a1 -2,13 -h </var/adm/acct/nite(x)/dacct
```

Эта команда показывает записи об учете работы диска, в которых указано число блоков по 1 КБ, занятых каждым пользователем.

Примечание: Команда **acctmerg** всегда считывает стандартный ввод, однако для нее допустимо также применение входных данных из девяти дополнительных файлов. Если входные данные не передаются команде с помощью конвейера, то следует перенаправить в нее входные данные из одного файла. Остальные файлы могут быть указаны без перенаправления.

Отображает отчет об использовании принтера или графопостроителя:

Вы можете просмотреть записи системы учета об использовании принтера или графопостроителя с помощью команды **pac**.

- Для сбора данных об использовании принтера необходимо настроить и запустить систему учета. См. раздел “Настройка системы учета ресурсов” на стр. 167.
- Для принтера или графопостроителя, о котором вы хотите просмотреть записи системы учета, в разделе описания принтера в файле `/etc/qconfig` должен быть указан параметр `acctfile=`. Файл, указанный в операторе `acctfile=`, должен содержать разрешения на чтение и запись для пользователя root или группы printq.
- Если в команде **-s** указан флаг **pac**, то команда изменяет общий путь, добавляя **_sum** к пути, заданному параметром `acctfile=` в файле `/etc/qconfig`. Этот файл должен существовать и содержать разрешения на чтение и запись для пользователя root или группы printq.

Для просмотра информации о работе с конкретным принтером всех пользователей введите:

```
/usr/sbin/pac -Рпринтер
```

Если вы не укажете принтер, то по умолчанию имя принтера будет получено из переменной среды **PRINTER**. Если переменная среды **PRINTER** не определена, то по умолчанию будет приято имя `lp0`.

Для просмотра информации о работе конкретных пользователей с конкретным принтером введите:

```
/usr/sbin/pac -Рпринтер Пользователь1 Пользователь2 ...
```

С помощью других флагов команды **pac** можно настраивать формат показываемой информации.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Обновление устаревшего файла holidays:

Файл Holidays считается устаревшим в том случае, если прошел последний указанный праздник или начался новый год. Файл Holidays можно обновить.

Команда **acctcon1** (запускаемая из сценария **runacct**) отправляет сообщение пользователям **root** и **adm**, если файл `/usr/lib/acct/holidays` устарел.

Обновите устаревший файл Holidays, отредактировав файл `/var/adm/acct/holidays` и указав в нем рабочее и нерабочее время.

Рабочее время - это период, когда система используется наиболее активно, например, рабочие дни. Свободное время для системы учета ресурсов - это суббота и воскресенье, а также все праздники, указанные в файле.

Файл holidays содержит записи трех типов: комментарии, год и рабочее время, а также список праздников, как показано в следующем примере:

```
*Таблица рабочего и свободного времени для системы учета ресурсов
*
*   Тек.       Запуск в          Запуск в
*   год        рабочее время     нерабочее время
    1992       0830              1700
*
*   День       Календарная       Название
*   года       дата              праздника
*
*   1           1 Января         Новый год
*   7           7 Января         Рождество
```

```
*   54          23 Февраля        День защитника отечества
*   67          8 Марта           Международный женский день
*   121         1 Мая             День труда
*   129         9 Мая             День победы
*   163         12 Июня           День независимости
*   244         1 Сентября        День знаний
*   280         7 Октября         День Конституции 1977 года
*   311         7 Ноября          День общественного примирения и согласия
*   339         5 Декабря         День Конституции 1936 года
*   346         12 Декабря        День Конституции Российской Федерации
*   359         25 Декабря        Католическое Рождество
```

Первая строка без комментария должна задавать текущий год, а также начало и конец рабочего периода времени в виде четырех цифр. Рабочее и нерабочее время влияет только на то, как программа учета ресурсов обрабатывает записи учета ресурсов.

Если список праздников очень длинный, то команда **acctcon1** отправит сообщение об ошибке и вам придется его сократить. Список может содержать не более 20 праздников. Для увеличения списка праздников обновляйте файл holidays каждый месяц.

Сбор данных учета ресурсов

После настройки учета ресурсов системы можно начать сбор и обработку разного рода данных учета ресурсов.

Сбор и отчет о системных данных:

Можно настроить систему для автоматического сбора данных и создания отчетов.

Для запуска автоматического сбора информации пользователь из группы adm должен быть настроен как система учета. Установка системы учета настраивает демон **cron** на запуск команд, собирающих данные о следующих параметрах:

- Время работы в системе каждого пользователя
- Использование процессора, памяти и устройств ввода/вывода
- Объем дисковой памяти, занимаемой файлами каждого пользователя
- Использование принтеров и графопостроителей
- Количество запусков различных команд

Система создает запись для каждого сеанса и обрабатывает ее после завершения сеанса. В дальнейшем эти записи преобразуются в итоговые записи учета ресурсов (`tacct`) для различных пользователей и объединяются в ежедневный отчет. Периодически дневные отчеты объединяются для создания итоговых отчетов за больший период времени. Способы сбора данных и создания отчетов, а также различные команды и файлы учета обсуждаются в следующих разделах.

Хотя большая часть данных учета ресурсов собирается автоматически, пользователь из группы adm для получения конкретной информации может ввести некоторые команды с клавиатуры.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Ссылки, связанные с данной:

“Команды клавиатуры” на стр. 163
Пользователь из группы adm может выполнить следующие команды учета ресурсов.

Данные учета времени соединения:

Данные о времени соединения собираются с помощью команды **init** и **login**.

При каждом входе в систему программа **login** добавляет запись в файл /etc/utmp. Эта запись содержит имя пользователя, дату, время и порт входа в систему. Другие команды, например, команда **who**, считывают содержимое этого файла для получения информации о том, какие пользователи работают на различных дисплейных станциях. Если в системе существует файл учета /var/adm/wtmp, то команда **login** добавляет в него копию записи о входе в систему. Сведения о командах **init** и **login** приведены в описании команд **init** и **login**.

После завершения начальной программы (обычно при выходе из системы) процесс **init** добавляет в файл /var/adm/wtmp запись о завершении сеанса. Записи о выходе из системы отличаются от записей о входе в систему тем, что в них указано пустое имя пользователя. Формат обоих типов записей описан в файле utmp.h. Информация о файле utmp.h приведена в разделе utmp.h.

Команда **acctwtmp** заносит в файл /var/adm/wtmp записи при запуске и завершении работы системы.

Понятия, связанные с данным:

“Отчеты о времени соединения” на стр. 160
Отчеты включают в себя сведения о входе в систему, выходе из системы, завершении работы системы и инфор мацию о последнем входе в систему.

Данные о процессе:

В ходе выполнения процесса система учета собирает данные об используемых им ресурсах.

К этим данным относится:

- Номер пользователя, запустившего процесс, а также номер его группы
- Первые восемь символов имени команды
- 64-разрядный числовой ключ, соответствующий классу управления рабочей схемой, к которому относится процесс
- Полное время выполнения процесса и затраченное процессорное время
- Доля занятой памяти
- Число переданных символов
- Число блоков диска, считанных или записанных процессом

Команда **accton** записывает эти данные в указанный файл, обычно в /var/adm/pacct. Дополнительная информация о команде **accton** приведена в разделе **accton**.

К связанным командам относятся **startup**, **shutacct**, **dodisk**, **ckpacct** и **turnacct**. Информация об этих командах приведена в разделах **startup**, **shutacct**, **dodisk**, **ckpacct** и **turnacct**.

Понятия, связанные с данным:

“Отчеты по учету ресурсов” на стр. 157
После сбора информации об использовании различных ресурсов полученные данные обрабатываются и преобразуются в отчеты.

Отчеты о процессах:

Для обработки данных из файла /var/adm/pacct или другого файла, применяемого для подсчета оплаты, служат две команды.

Команда **acctprc1** преобразует ИД в имена пользователей и создает текстовые записи об использовании различных ресурсов (использование процессора в рабочее и нерабочее время, средний объем памяти, объем данных ввода-вывода). Команда **acctprc2** преобразует эти записи в итоговые записи учета ресурсов, которые добавляются к ежедневному отчету командой **acctmerg**. Дополнительная информация о команде **acctmerg** приведена в разделе **acctmerg**.

Кроме того, учет ресурсов процессов позволяет получить информацию об использовании ресурсов различными командами. Для создания таких отчетов служит команда **acctcms**. При запуске этой команды будет показано, сколько раз была запущена каждая команда, сколько памяти и процессорного времени потребовалось на ее выполнение, и насколько интенсивно использовались при этом ресурсы (величина *hog factor*). Команда **acctcms** накапливает статистическую информацию, такую как полный объем использованных ресурсов и частоту запуска отдельных команд, за длительный период времени. Дополнительная информация о команде **acctcms** приведена в разделе **acctcms**.

Команда **acctcom** обрабатывает те же данные, что и команда **acctcms**, но выводит подробную информацию о процессах. Вы можете просматривать все записи учета ресурсов для процесса или выбрать только часть из них. Критерий выбора может учитывать использование процессора, время завершения, команду запуска, пользователя, группу и порт процесса. В отличие от прочих команд учета ресурсов, команду **acctcom** может запустить любой пользователь. Дополнительная информация о команде **acctcom** приведена в разделе **acctcom**.

Учет использования диска:

Большая часть информации об использовании ресурсов относится к их расходу. Команда **dodisk**, запускаемая демоном **cron** в соответствии с расписанием, добавляет в файл `/var/adm/acct/nite(x)/dacct` записи об объеме дисковой памяти, занятой различными пользователями.

Для этого команда **dodisk** вызывает другие команды. В зависимости от требуемой подробности учета ресурсов, для сбора данных применяется команда **diskusg** или **acctdusg**. Для создания итоговой записи учета ресурсов применяется команда **acctdisk**. Итоговая запись, в свою очередь, применяется командой **acctmerg** для создания ежедневного отчета об использовании ресурсов.

Команда **dodisk** просматривает все файлы в домашних каталогах пользователей и делит стоимость каждого файла на число ссылок на него. При этом стоимость файла будет распределена между всеми пользователями, работавшими с этим файлом. Остальные пользователи не будут вовлечены в эту процедуру. Дополнительная информация о команде **dodisk** и демоне **cron** приведены в разделах **dodisk** и **cron**.

Понятия, связанные с данным:

“Отчеты о времени соединения” на стр. 160
Отчеты включают в себя сведения о входе в систему, выходе из системы, завершении работы системы и инфор мацию о последнем входе в систему.

Данные об учете использования принтере:

Данные об использовании принтеров собираются совместно командой **enq** и демоном работы с очередью.

Команда **enq** ставит в очередь имя пользователя, номер задания и имя печатаемого файла. После завершения печати файла команда **qdaemon** добавляет в определенный файл (обычно в `/var/adm/qacct` текстовую запись с именем, номером пользователя и числом напечатанных страниц. Эти записи затем можно отсортировать и преобразовать в итоговые записи учета ресурсов. Дополнительная информация о этих командах приведена в разделах **enq** и **qdaemon**.

Понятия, связанные с данным:

“Отчет об использовании принтера” на стр. 160
Текстовые записи файла `/var/adm/qacct` могут быть преобразованы в итоговые записи учета ресурсов и добавлены в ежедневный отчет командой **acctmerg**.

Данные об оплате:

В файле `/var/adm/fee` можно создать итоговую запись учета ресурсов в формате ASCII.

Для создания в файле /var/adm/fee итоговой записи учета ресурсов введите команду **chargefee**. Этот файл затем будет добавлен в ежедневный отчет командой **acctmerg**.

Сведения о командах **chargefee** и **acctmerg** приведены в разделах **chargefee** и **acctmerg**.

Понятия, связанные с данным:

“Отчет об оплате” на стр. 161
Если вы применяете команду **chargefee** для создания счетов за такие услуги, как восстановление файлов, консультации и использование расходных материалов, в файле /var/adm/fee создаются текстовые записи учета ресурсов. Этот файл добавляется в ежедневный отчет командой **acctmerg**.

Устранение неполадок учета ресурсов системы

Следующие приемы устранения неполадок предназначены для разрешения некоторых наиболее распространенных неполадок, которые могут возникнуть при учете ресурсов системы. Если эта информация не помогла устранить неполадку, обратитесь в службу поддержки.

Исправление ошибок tacct:

При изменении владельца системных ресурсов с помощью системы учета ресурсов необходимо поддерживать целостность файла /var/adm/acct/sum/tacct. Случайным образом в файле **tacct** иногда появляются ошибочные записи, содержащие отрицательные числа, дублирующие номера пользователей или номера пользователей 65535. Эти ошибки можно исправить.

У вас должны быть права пользователя root или члена группы adm.

Для исправления файла tacct выполните следующие действия:

1. Перейдите в каталог /var/adm/acct/sum, введите:
   ```
   cd /var/adm/acct/sum
   ```
2. С помощью команды **prtacct** проверьте общий файл учета ресурсов tacctprev, введите:
   ```
   prtacct tacctprev
   ```
   Команда **prtacct** показывает отформатированное содержание файла tacctprev. Проверьте время соединения, время обработки, использование дисков и принтера.
3. Если в файле tacctprev содержится правильная информация, то преобразуйте последний файл tacct *.ммдд* из двоичного формата в формат ASCII. В приведенном ниже примере команда **acctmerg** преобразует файл tacct.*ммдд* в файл ASCII с именем tacct.new:
   ```
   acctmerg -v < tacct.mmdd > tacct.new
   ```
   Примечание: Команда **acctmerg** с флагом **-a** также создает вывод ASCII. Флаг **-v** задается для вывода чисел с плавающей точкой с максимальной точностью.

   Команда **acctmerg** вставляет промежуточные отчеты с записями учета ресурсов в общий отчет (**tacct**). Общий отчет обрабатывается командой **monacct** для создания ежемесячного обзорного отчета в формате ASCII. Команда **monacct** удаляет все файлы tacct.*mmdd*, поэтому вам нужно снова создать файл tacct, объединив эти файлы.
4. Удалите из файла tacct.new неверные записи и скопируйте записи с повторяющимися номерами пользователей в другой файл, введите:
   ```
   acctmerg -i < tacct.new > tacct.mmdd
   ```
5. Создайте файл tacct еще раз, введите:
   ```
   acctmerg tacctprev < tacct.mmdd > tacct
   ```

Исправление ошибок wtmp:

Файл /var/adm/wtmp ("who temp") может вызвать сбои в повседневных процедурах, выполняемых системой учета. Ошибки wtmp можно исправить.

У вас должны быть права пользователя root или члена группы adm для выполнения данной поцедуры.

Если при изменении даты система находится в многопользовательском режиме, то записи об изменении даты заносятся в файл /var/adm/wtmp. При изменении даты команда **wtmpfix** исправляет системное время в записях wtmp. Изменение даты с последующей перезагрузкой системы может вызвать пропуск команды **wtmpfix** и привести к сбою при выполнении команды **acctcon1**, а также к тому, что команда **runacct** отправит в учетные файлы **root** и **adm** сообщение с записями об ошибках при определении времени.

Для исправления ошибок wtmp выполните следующую процедуру:

1. Перейдите в каталог /var/adm/acct/nite, введите:
   ```
   cd /var/adm/acct/nite
   ```
2. Преобразуйте двоичный файл wtmp в текстовый файл, который можно редактировать, введите:
   ```
   fwtmp < wtmp.mmdd > wtmp.new
   ```
   Команда **fwtmp** преобразует wtmp из двоичного кода в текстовый формат.
3. Отредактируйте текстовый файл wtmp.new, удалив поврежденные записи или все записи с начала файла до записи с требуемой датой, введите:
   ```
   vi wtmp.new
   ```
4. Преобразуйте текстовый файл wtmp.new обратно в двоичный формат, введите:
   ```
   fwtmp -ic < wtmp.new > wtmp.ммдд
   ```
5. Если файл wtmp восстановить невозможно, то с помощью команды **nulladm** создайте пустой файл wtmp. Это избавит вас от ошибок при вычислении оплаты за время соединения.
   ```
   nulladm wtmp
   ```
   Команда **nulladm** создает файл с правами доступа на запись и чтение для владельца и группы файла, и с правами доступа на чтение для других пользователей. Это позволяет гарантировать, что у владельца и группы файла будут права доступа **adm**.

Задачи, связанные с данной:

“Исправление ошибок учета ресурсов” на стр. 180
Можно исправить несогласованность времени и даты.

Исправление неверных прав доступа к файлу системы учета ресурсов:

Для применения системы учета ресурсов нужно задать права доступа к файлу и владельца файла.

У вас должны быть права пользователя root или члена группы adm для выполнения данной поцедуры.

Команды и сценарии учета ресурсов принадлежат административной группе **adm**, за исключением команды /var/adm/acct/accton, которая принадлежит пользователю root.

Для исправления неверных прав доступа к файлу системы учета ресурсов выполните следующие действия:

1. Для того чтобы с помощью команды **ls** узнать права доступа, установленные для файла, введите следующую команду:
   ```
   ls -l /var/adm/acct

   -rws--x--- 1 adm adm 14628 Mar 19 08:11 /var/adm/acct/fiscal
   -rws--x--- 1 adm adm 14628 Mar 19 08:11 /var/adm/acct/nite
   -rws--x--- 1 adm adm 14628 Mar 19 08:11 /var/adm/acct/sum
   ```
2. При необходимости измените права доступа с помощью команды **chown**. Для файла должны быть установлены права доступа 755 (у владельца есть все права доступа, а у всех остальных пользователей - права на чтение и выполнение). Кроме того, у всех остальных пользователей не должно быть прав на запись в каталог. Например:
   a. Перейдите в каталог /var/adm/acct с помощью следующей команды:

```
cd /var/adm/acct
```

b. Установите в качестве владельца каталогов sum, nite и fiscal группу **adm** с помощью следующей команды:

```
chown adm sum/* nite/* fiscal/*
```

Отключите разрешение на запись для всех остальных пользователей. Установите в качестве владельца файла **accton** группу **adm** и установите этой группы права доступа 710, то есть запретите доступ всем остальным пользователям. Процессы, относящиеся к группе **adm**, смогут выполнять команду **accton**, а все остальные пользователи - нет.

3. Файл /var/adm/wtmp также должен принадлежать группе **adm**. Если /var/adm/wtmp принадлежит пользователю root, то во время запуска появится следующее сообщение:

```
/var/adm/acct/startup: /var/adm/wtmp: Доступ запрещен
```

Для того чтобы изменить принадлежность файла /var/adm/wtmp, установите в качестве его владельца группу **adm** с помощью следующей команды:

```
chown adm /var/adm/wtmp
```

Исправление ошибок учета ресурсов:

Можно исправить несогласованность времени и даты.

У вас должны быть права пользователя root или члена группы adm для выполнения данной поцедуры.

Во время обработки файла /var/adm/wtmp пользователю root может быть отправлено несколько предупреждений. Файл wtmp содержит информацию, полученную процессами /etc/init и /bin/login. Этот файл применяется сценариями учета ресурсов для определения продолжительности работы пользователей в системе. К сожалению, программа обработки файла wtmp не рассчитана на изменение даты. В результате команда**runacct** отправляет пользователям root и adm сообщение со списком ошибок, возникших с момента последнего изменения даты, то есть с момента последнего запуска подсистемы учета ресурсов.

1. Узнайте, получили ли вы какие-либо сообщения об ошибках. Команда **acctcon1** выдает сообщения об ошибках, которые передаются adm и root командой **runacct**. Например, если процесс **acctcon1** приостановился после изменения даты и не собрал информацию о времени соединения, то adm может получить электронное сообщение приблизительно следующего содержания:

```
Mon Jan 6 11:58:40 CST 1992
acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
new: Mon Jan 6 11:57:59 1992
acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
new: Mon Jan 6 11:57:59 1992
acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
new: Mon Jan 6 11:57:59 1992
```

2. Исправьте ошибку в файле wtmp с помощью следующей команды:

```
/usr/sbin/acct/wtmpfix wtmp
```

Команды **wtmpfix** проверяет файл wtmp на согласованность значений системной даты и времени, и исправляет неполадки, которые могут вызвать сбой при выполнении команды **acctcon1**. Тем не менее, команда **wtmpfix** не учитывает некоторые изменения даты.

3. Запустите учет ресурсов непосредственно перед завершением работы системы или сразу после запуска системы. Выполнение команды **runacct** в это время позволит минимизировать число записей с неверными значениями времени. Команда **runacct** будет отправлять электронные сообщения в файл учета ресурсов root и adm, пока вы не отредактируете сценарий **runacct**. Найдите в сценарии раздел WTMPFIX и установите символ комментария в строке, которая управляет отправкой файла протокола пользователям root и adm.

Задачи, связанные с данной:

“Исправление ошибок wtmp” на стр. 178
Файл /var/adm/wtmp ("who temp") может вызвать сбои в повседневных процедурах, выполняемых системой

учета. Ошибки wtmp можно исправить.

Неполадки учета ресурсов при работе с командой runacct:

При выполнении команды **runacct** могут возникнуть неполадки.

Примечание: У вас должны быть права пользователя root или члена группы adm для выполнения команды **runacct**.

Команда **runacct** часто обрабатывает файлы очень большого размера. Для этого необходимо несколько раз просматривать определенные файлы, что потребляет значительные ресурсы системы. Именно поэтому команда **runacct** обычно выполняется рано утром, когда она может получить все необходимые ресурсы, не влияя на работу пользователей.

Команда **runacct** - это сценарий, разделенный на несколько этапов. После завершения каждого этапа выполнение команды можно приостановить, а затем снова возобновить с прерванного места.

Если во время выполнения **runacct** возникает неполадка, то целевое устройство, в которое отправляются сообщения об ошибке, зависит от этапа, на котором произошла ошибка. Обычно дата и сообщение отправляются на консоль. В сообщении говорится, что нужно просмотреть файл active*ММДД* (например, active0621 за 21 июня), расположенный в каталоге /usr/adm/acct/nite. Если команда **runacct** преждевременно завершает работу, то весь файл active помещается в файл active*ММДД*, к которому добавляется сообщение с описанием неполадки.

В приведенных ниже таблицах описаны сообщения об ошибках, которые могут возникнуть при выполнении команды **runacct**.

Примечание:

- Аббревиатура *ММДД* означает месяц и день, например, 0102 соответствует 2 января. Если во время выполнения процесса CONNECT1 2 января произойдет неисправимая ошибка, то будет создан файл active0102, содержащий сообщение об ошибке.
- Аббревиатура "сообщение SE" означает стандартное сообщение об ошибке, например:
  ```
  ********* ОШИБКА УЧЕТА РЕСУРСОВ : см. файл active0102 *********
  ```

Сообщения о предварительном состоянии и об ошибках команды **runnacct**

Состояние	Команда	Неисправимая?	Сообщение об ошибке	Получатели
pre	**runacct**	да	* 2 демона cron или НЕПОЛАДКА УЧЕТА РЕСУРСОВ * ОШИБКА: обнаружена блокировка, выполнение прекращено	консоль, почта, файл active
pre	**runacct**	да	runacct: в каталоге /usr недостаточно свободной памяти (*nnn* блоков); Процедура прервана	консоль, почта, файл active
pre	**runacct**	да	сообщение SE; ОШИБКА: для 'дата' уже запущен acctg: проверьте дату последнего изменения	консоль, почта, файл active*ММДД*
pre	**runacct**	нет	* УЧЕТ РЕСУРСОВ СИСТЕМЫ ЗАПУЩЕН*	консоль
pre	**runacct**	нет	повторный запуск acctg для 'дата' в состоянии СОСТОЯНИЕ	активная консоль, консоль

Сообщения о предварительном состоянии и об ошибках команды **runnacct**

Состояние	Команда	Неисправимая?	Сообщение об ошибке	Получатели
pre	**runacct**	нет	повторный запуск acctg для 'дата' в состоянии, определяемым аргументом $2, предыдущее состояние - СОСТОЯНИЕ	файл active
pre	**runacct**	да	сообщение SE; Ошибка: неверные аргументы runacct	консоль, почта, файл active*ММДД*

Сообщения о состоянии и об ошибках команды **runacct**

Состояние	Команда	Неисправимая?	Сообщение об ошибке	Получатели
НАСТРОЙКА	**runacct**	нет	ls -l fee pacct* /var/adm/wtmp	файл active
НАСТРОЙКА	**runacct**	да	сообщение SE; Ошибка: переключатель turnacct вернул rc=error	консоль, почта, файл active*ММДД*
НАСТРОЙКА	**runacct**	да	сообщение SE; ошибка: файл SpacctMMDD уже существует. Возможно, настройка уже запущена	файл active*ММДД*
НАСТРОЙКА	**runacct**	да	сообщение SE; ошибка: файл wtmpMMDD уже существует; запустите настройку вручную	консоль, почта, файл active*ММДД*
WTMPFIX	**wtmpfix**	нет	сообщение SE; Ошибка: см. сообщения об ошибках wtmpfix в xtmperrorММДД	файлы active*ММДД*, wtmperror*ММДД*
WTMPFIX	**wtmpfix**	нет	обработка wtmp завершена	файл active
CONNECT1	**acctcon1**	нет	сообщение SE; (сообщения об ошибках из протокола acctcon1)	консоль, почта, файл active*ММДД*
CONNECT2	**acctcon2**	нет	соединение acctg завершено	файл active
ПРОЦЕСС	**runacct**	нет	ПРЕДУПРЕЖДЕНИЕ: для pacct*N* учет ресурсов уже запущен	файл active
ПРОЦЕСС	**acctprc1 acctprc2**	нет	команда acctg выполнена для SpacctNMMДД	файл active
ПРОЦЕСС	**runacct**	нет	выполнены все процессы actg для указанной даты	файл active
ВСТАВКА	**acctmerg**	нет	выполнено объединение tacct для создания dayacct	файл active
FEES	**acctmerg**	нет	объединенная оплата ИЛИ нет оплаты	файл active
ДИСК	**acctmerg**	нет	объединение записей о диске ИЛИ нет записей о диске	файл active

Сообщения о состоянии и об ошибках команды **runacct**

Состояние	Команда	Неисправимая?	Сообщение об ошибке	Получатели
MERGEACCT	**acctmerg**	нет	ПРЕДУПРЕЖДЕНИЕ: восстановление sum/tacct	файл active
MERGEACCT	**acctmerg**	нет	обновление sum/tacct	файл active
CMS	**runacct**	нет	ПРЕДУПРЕЖДЕНИЕ: восстановление sum/cms	файл active
CMS	**acctcms**	нет	вычисление итоговых данных завершено	файл active
ОЧИСТКА	**runacct**	нет	выполнен учет ресурсов системы, 'дата'	файл active
ОЧИСТКА	**runacct**	нет	*УЧЕТ РЕСУРСОВ СИСТЕМЫ ВЫПОЛНЕН*	консоль
<ошибка>	**runacct**	да	сообщение SE; ошибка: недопустимое состояние, просмотрите поле состояния	консоль, почта, файл active*ММДД*

Примечание: Метка <ошибка> в приведенной выше таблице не соответствует никакому состоянию; она означает, что в файл состояний /usr/adm/acct/nite/statefile было записано неверное состояние.

Список терминов

Получатель	Описание
консоль	Устройство /dev/console
почта	Сообщение будет отправлено по электронной почте и записано в файл учета ресурсов **root** и **adm**
файл active	Файл /usr/adm/acct/nite/active
activeММДД	Файл /usr/adm/acct/nite/active*ММДД*
wtmperrММДД	Файл /usr/adm/acct/nite/wtmperror*ММДД*
СОСТОЯНИЕ	Текущее состояние, указанное в файле /usr/adm/acct/nite/statefile
fd2log	Любое другое сообщение об ошибке

Контроллер системных ресурсов

Контроллер системных ресурсов (SRC) предоставляет набор команд и подпрограмм, которые упрощают задачи создания и управления подсистемами для системного администратора и программиста.

Подсистема - это любая программа или процесс (либо набор программ или процессов), которые выполняются независимо или под управлением системы. Подсистема реализуется как модуль, выполняющий определенную функцию.

SRC позволяет свести к минимуму деятельность оператора. Он предоставляет механизм для управления подсистемами с помощью командной строки и интерфейса языка C. В этот механизме предусмотрены следующие функции:

- Общий интерфейс для запуска, останова и просмотра состояния подсистем
- Занесение в протокол сообщений об аварийном завершении подсистемы
- Программа уведомления, вызываемая при аварийном завершении связанного процесса
- Трассировка подсистем, групп подсистем и субсервера
- Управление удаленной системой
- Обновление подсистемы (например, после изменения конфигурации).

SRC применяется в том случае, если вам нужен универсальный способ для запуска, завершения работы и сбора информации о состоянии процесса.

Понятия, связанные с данным:

“Введение в AIX системных администраторов BSD” на стр. 322
Ниже приведены советы для системных администраторов Berkeley Software Distribution (BSD), начинающих работать с AIX.

Компоненты подсистемы

Ниже описаны свойства и компоненты подсистемы.

Возможные свойства подсистемы:

- Идентифицируется системой по имени
- Требует более сложной среды выполнения, чем функция или обычная программа
- Содержит кроме кода подсистемы также приложения и библиотеки
- Управляет запуском и остановом ресурсов по имени
- Требует уведомления, если связанному процессу не удается выполнить очистку или получить необходимые ресурсы
- Требует более сложного управления, чем простой демон
- Управляется удаленно
- Реализует субсерверы, предназначенные для управления специальными ресурсами
- Не переходит в фоновый режим автоматически.

Примерами подсистем могут служить ypserv, ntsd, qdaemon, inetd, syslogd и sendmail.

Примечание: За информацией о свойствах SRC конкретной подсистемы обращайтесь к ее описанию.

Для просмотра списка активных и неактивных подсистем введите команду **lssrc -a**.

Ниже приведены описания групп подсистем и субсерверов:

Группа подсистем

Группа подсистем - это набор любых заданных подсистем. Объединение подсистем в группы позволяет одновременно управлять несколькими подсистемами. Примерами группы подсистем могут служить TCP/IP, Службы SNA, Служба информации о сети (NIS) и Сетевая файловая система (NFS).

Субсервер

Субсервер - это программа или процесс, принадлежащий подсистеме. Подсистема может содержать несколько субсерверов. Она отвечает за запуск, завершение работы и предоставление информации о состоянии субсерверов. Субсерверы определяются только для подсистем, которые обмениваются информацией через очереди сообщений IPC или сокеты. Подсистемы, в которых применяется отправка сигналов, не поддерживают субсерверы.

Субсерверы запускаются при запуске родительской подсистемы. Если во время запуска субсервера с помощью команды **startsrc** родительская подсистема не была активна, то она также будет запущена.

Иерархия SRC

Корень иерархии Контроллера системных ресурсов - операционная система, затем следует группа подсистем (например, **tcpip**), содержащая подсистему (например, **inetd**), управляющую несколькими субсерверами (например, демоном **ftp** и командой **finger**).

Команды администрирования SRC

Выполнять администрирование SRC можно с помощью командной строки.

К командам администрирования SRC относятся:

Элемент	Описание
Демон **srcmstr**	Запускает Контроллер системных ресурсов
Команда **startsrc**	Запускает подсистему, группу подсистем или субсервер
Команда **stopsrc**	Останавливает подсистему, группу подсистем или субсервер
Команда **refresh**	Обновляет подсистему
Команда **traceson**	Включает трассировку подсистемы, группы подсистем или субсервера
Команда **tracesoff**	Выключает трассировку подсистемы, группы подсистем или субсервера
Команда **lssrc**	Выводит информацию о состоянии подсистемы.

Запуск контроллера системных ресурсов

Контроллер системных ресурсов (SRC) запускается во время инициализации системы, если в файле `/etc/inittab` содержится запись о демоне `/usr/sbin/srcmstr`.

Ниже перечислены предварительные требования для запуска SRC:

- Для чтения и записи в файл `/etc/inittab` у вас должны быть права пользователя root.
- Для запуска команды **mkitab** у вас должны быть права доступа пользователя root.
- В файле `/etc/inittab` должна быть запись о демоне **srcmstr**.

По умолчанию в файле `/etc/inittab` уже есть эта запись. Кроме того, SRC можно запустить из командной строки, профайла или сценария оболочки, однако рекомендуется запускать его во время инициализации по следующим причинам:

- Запуск SRC из файла `/etc/inittab` позволяет перезапустить SRC с помощью команды **init**, если его работа была прервана по каким-либо причинам.
- SRC позволяет в значительной степени автоматизировать управление подсистемами. Запуск SRC из источника, отличного от `/etc/inittab`, должен выполняться вручную.
- По умолчанию файл `/etc/inittab` содержит запись для запуска подсистемы планирования печати (**qdaemon**) с помощью команды **startsrc**. Обычно с помощью команды **startsrc** из файла `/etc/inittab` запускаются и другие подсистемы. Для выполнения команды **srcmstr** нужно запустить SRC, поэтому удаление демона **srcmstr** из файла `/etc/inittab` приведет к ошибке при выполнении команды **startsrc**.

Примечание: Эту процедуру нужно выполнять только в том случае, если в файле `/etc/inittab` еще не содержится запись о демоне **srcmstr**.

1. Создайте запись о демоне **srcmstr** в файле `/etc/inittab` с помощью команды **mkitab**. Например, для того чтобы создать запись, аналогичную записи в файле по умолчанию `/etc/inittab`, введите:

   ```
   mkitab -i fbcheck srcmstr:2:respawn:/usr/sbin/srcmstr
   ```

   Флаг **-i fbcheck** позволяет разместить эту запись перед всеми записями о подсистемах.
2. Запустите обработку файла `/etc/inittab` с помощью **init** , выполнив команду:

   ```
   telinit q
   ```

   После просмотра файла `/etc/inittab` команда **init** обработает новую запись о демоне **srcmstr** и запустит SRC.

Понятия, связанные с данным:

“Управление подсистемой” на стр. 187
С помощью команды **traceson** можно включить, а с помощью команды **traceoff** - отключить трассировку ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера.

Задачи, связанные с данной:

“Обновление подсистемы или группы подсистем” на стр. 187
Команда **refresh** предназначена для обновления ресурса Контроллера системных ресурсов (SRC), например, подсистемы или группы подсистем.

Запуск и остановка подсистемы, группы подсистем или субсервера

Команда **startsrc** позволяет получить информацию о состоянии ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера. Команда **stopsrc** позволяет получить информацию о состоянии ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера.

Ниже приведены предварительные требования для запуска и остановки подсистемы, группы подсистем или субсервера:

- Для запуска или завершения работы ресурса SRC должна быть запущена программа SRC. Обычно SRC запускается при инициализации системы. Файл по умолчанию `/etc/inittab` , в котором указан список процессов, запускаемых во время инициализации системы, содержит запись запуска демона **srcmstr** (the SRC). Для того чтобы узнать, активен ли SRC, введите команду `ps -A` и найдите процесс с именем `srcmstr`.
- У пользователя или процесса, запускающего ресурс SRC, должны быть права пользователя root. У процесса, инициализирующего систему (команда **init**), есть права пользователя root.
- У пользователя или процесса, завершающего работу ресурса SRC, должны быть права пользователя root.

Команду **startsrc** можно ввести:

- Файл `/etc/inittab` позволяет задать запуск ресурса во время инициализации системы
- С помощью командной строки
- С использованием SMIT

При запуске группы подсистем запускаются все входящие в нее подсистемы. При запуске подсистемы запускаются все ее субсерверы. При запуске субсервера запускается его родительская подсистема, если она еще не активна.

При завершении работы подсистемы завершается работа всех ее субсерверов. Однако при завершении работы сервера состояние его родительской подсистемы не изменяется.

В командах **startsrc** и **stopsrc** предусмотрены флаги, позволяющие отправлять запросы на локальные и удаленные хосты. За информацией о требованиях, которые предъявляются к конфигурации для поддержки удаленных запросов SRC, обратитесь к описанию команды **srcmstr**.

Запуск и остановка задач подсистемы

Задача	*SMIT, команда быстрого доступа*	*Команда или файл*
Запустить подсистему	**smit startssys**	`/bin/startsrc -s` *подсистема* , или измените файл `/etc/inittab`
Остановить подсистему	**smit stopssys**	`/bin/stopsrc -s` *подсистема*

Информация, связанная с данной:

команда stopsrc

команда startsrc

srcmstr, команда

Просмотр состояния подсистемы или подсистем

Команда **lssrc** позволяет получить информацию о состоянии ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера.

Во всех подсистемах предусмотрена возможность создания короткого отчета о состоянии, в котором содержится группа, к которой относится подсистема, ИД процесса (PID), и указывается, активна ли подсистема. Если подсистема не поддерживает отправку сигналов, то в ней может быть предусмотрена возможность отправки подробного отчета о состоянии, содержащего дополнительную информацию.

В команде **lssrc** предусмотрены флаги и параметры для указания имени или PID подсистемы, просмотра списка всех подсистем, запроса краткого или подробного отчета о состоянии и запроса информации о состоянии ресурсов SRC локального или удаленного хоста.

За информацией о требованиях, которые предъявляются к конфигурации для поддержки удаленных запросов SRC, обратитесь к описанию команды **srcmstr**.

Задачи просмотра информации о состоянии подсистем

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Просмотр информации о состоянии подсистемы (полный формат)	**smit qssys**	**lssrc -l -s** *Имя_подсистемы*
Просмотр информации о состоянии всех подсистем	**smit lsssys**	**lssrc -a**
Просмотр информации о состоянии подсистем отдельного хоста		**lssrc -h***хост* **-a**

Обновление подсистемы или группы подсистем

Команда **refresh** предназначена для обновления ресурса Контроллера системных ресурсов (SRC), например, подсистемы или группы подсистем.

Ниже описаны предварительные требования для обновления подсистемы или группы подсистем:

- Должна быть запущена программа SRC.
- Обновляемый ресурс не должен обмениваться информацией с помощью сигналов.
- В обновляемом ресурсе должна быть предусмотрена возможность ответа на запрос об обновлении.

В команде **refresh** предусмотрены флаги и параметры, позволяющие указать подсистему по имени и PID. Команда позволяет отправить запрос на обновление подсистемы или группы подсистем как в локальной системе, так и на удаленном хосте. За информацией о требованиях, которые предъявляются к конфигурации для поддержки удаленных запросов SRC, обратитесь к описанию команды **srcmstr**.

Обновление подсистемы или группы подсистем

Задача	**Команда быстрого доступа SMIT**	**Команда или файл**
Обновить подсистему	**smit refresh**	**refresh -s подсистема**

Задачи, связанные с данной:

“Запуск контроллера системных ресурсов” на стр. 185
Контроллер системных ресурсов (SRC) запускается во время инициализации системы, если в файле `/etc/inittab` содержится запись о демоне `/usr/sbin/srcmstr`.

Управление подсистемой

С помощью команды **traceson** можно включить, а с помощью команды **traceoff** - отключить трассировку ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера.

Команда **traceson** позволяет включить трассировку ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера.

Команда **tracesoff** позволяет выключить трассировку ресурса Контроллера системных ресурсов (SRC), например, подсистемы, группы подсистем или субсервера.

Команды **traceson** и **traceoff** позволяют включать и выключать трассировку на удаленном хосте. За информацией о требованиях, которые предъявляются к конфигурации для поддержки удаленных запросов SRC, обратитесь к описанию команды **srcmstr**.

`Предварительные требования`

- Чтобы включить или выключить трассировку ресурса SRC в системе должен быть запущен контроллер системных ресурсов.

- Ресурс, для которого нужно включить трассировку, не должен обмениваться информацией с помощью сигналов.
- В ресурсе, для которого нужно включить трассировку, должна быть предусмотрена возможность ответа на запрос о трассировке.

Задачи включения и выключения трассировки подсистемы, группы подсистем и субсервера

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Включение трассировки подсистемы (сокращенный формат)	**smit tracessyson**	**traceson -s подсистема**
Включение трассировки подсистемы (полный формат)	**smit tracessyson**	**traceson -l -s подсистема**
Выключение трассировки подсистемы	**smit tracessysoff**	**tracesoff -s подсистема**

Задачи, связанные с данной:

“Запуск контроллера системных ресурсов” на стр. 185
Контроллер системных ресурсов (SRC) запускается во время инициализации системы, если в файле `/etc/inittab` содержится запись о демоне `/usr/sbin/srcmstr`.

Файлы операционной системой

Файлы используются для всей входной и выходной информации в операционной системе, для стандартизации доступа к программному и аппаратному обеспечению.

Вход происходит, когда содержимое файла изменяется или записывается в него. *Выход* происходит при чтении содержимого файла или передаче его в другой файл. Например, для создания распечатанной копии файла система читает информацию из текстового файла и записывает ее в файл, который представляет принтер.

Типы файлов

Система распознает файлы типа **обычные**, **файлы каталогов** или **специальные**. Однако в операционной системе также применяются многие производные типы файлов.

Существуют следующие основные типы файлов:

Элемент	**Описание**
обычный	Предназначен для хранения данных (текстовых или двоичных данных и исполняемого кода)
каталог	Содержит информацию, применяемую для доступа к другим файлам
специальный	Представляет собой файл канала FIFO или файл физического устройства

Система распознает только файлы этих типов. Однако в операционной системе также применяются многие производные типы файлов.

Стандартные файлы

Большая часть файлов относится к обычному типу и применяется для хранения данных. Обычные файлы делятся на текстовые файлы и двоичные файлы:

Текстовые файлы

Текстовые файлы - это обычные файлы, содержащие информацию в формате ASCII, которая может быть прочитана пользователем. Их можно просмотреть или напечатать. Строки текстового файла не должны содержать символы `NUL`, а их длина с учетом символа новой строки не должна превышать `{LINE_MAX}` байт.

Текстовый файл может содержать управляющие и непечатаемые символы (отличные от `NUL`). Если стандартная служебная программа получает текстовый файл в качестве ввода или записывает его в качестве вывода, то она должна уметь обрабатывать встреченные

специальные символы. В противном случае, в описании этой программы должны быть явно заданы ограничения, которые накладываются на входные текстовые файлы.

Двоичные файлы

Двоичные файлы - это обычные файлы, содержащие данные, которые может прочитать компьютер. Примером двоичного файла может быть исполняемый файл, в котором содержатся инструкции для системы по выполнению некоторого задания. Команды и программы хранятся в двоичном виде в форме исполняемых файлов. Для преобразования текста ASCII в двоичный код предназначены специальные программы-компиляторы.

Единственное различие между текстовыми и двоичными файлами состоит в том, что текстовые файлы разбиты на строки, не содержащие символов NUL и заканчивающиеся символом новой строки, длина которых не превосходит {LINE_MAX} байт.

Файлы каталогов

Файлы каталогов содержат информацию, которая необходима системе для доступа к файлам всех типов, но при этом в них не хранится сама информация из файлов. В результате каталоги занимают меньше места, чем обычные файлы. Кроме того, каталоги позволяют создать гибкую многоуровневую структуру файловой системы. Каждая запись каталога связана с файлом или подкаталогом и содержит имя файла и номер индексного дескриптора файла (*номер i-узла*). Номер i-узла ссылается на уникальный индексный дескриптор, присвоенный файлу. В индексном дескрипторе описано физическое расположение данных, связанных с файлом. Для создания каталогов и управления ими предназначен отдельный набор команд.

Специальные файлы

К специальным файлам относятся файлы устройств и временные файлы, создаваемые процессами. Существуют следующие основные типы специальных файлов: FIFO, блочные и символьные. Файлы FIFO также называются *каналами*. Канал создается на ограниченное время одним процессом для обмена данными с другим процессом. Такие файлы удаляются после завершения работы первого процесса. Блочные и символьные файлы определяют устройства.

Для каждого файла устанавливается набор разрешений (*прав доступа*), задающих группы пользователей, которые могут читать, изменять или выполнять файл.

Понятия, связанные с данным:

“Режимы доступа к файлам и каталогам” на стр. 297
У каждого файла есть владелец. Владельцем нового файла становится пользователь, создавший этот файл. *Права доступа* к файлу определяет его владелец. Права доступа указывают для других пользователей системы, могут ли они читать, изменять и исполнять файл. Права доступа к файлу могут быть изменены только владельцем файла или пользователем root.

Соглашения об именах файлов:

Имя каждого файла должно быть уникальным в каталоге, где он хранится. Таким образом обеспечивается уникальность пути к файлу в системе.

Существуют следующие основные правила присвоения имен файлам:

- Имя файла должно содержать не более 255 символов - букв, цифр и знаков подчеркивания.
- В операционной системе учитывается регистр символов, т.е. в именах файлов различаются прописные и строчные буквы. Например, FILEA, FiLea и filea - это три разных файла, и они могут быть расположены в одном каталоге.
- Имена файлов должны быть осмысленными и должны отражать содержимое файлов.
- Для каталогов применяются те же соглашения о присвоении имен, что и для файлов.
- Некоторые символы имеют специальное значение в операционной системе. Их нельзя указывать в именах файлов. Это следующие символы:

  / \ " ' * ; - ? [ ] ( ) ~ ! $ { } &lt > # @ & | пробел табуляция новая строка

- Имя файла не выводится в обычном списке файлов каталога, если оно начинается с точки (`.`). Однако список скрытых файлов и каталогов выводится вместе со обычными файлами и каталогами, если в команде **ls** задан флаг **-a**.

Пути к файлам:

Путь к объекту файловой системе состоит из имен всех каталогов, которые расположены выше данного файла или каталога в дереве каталогов.

Любой путь в файловой системе начинается с корневого каталога (`/`), поэтому для всех файлов существует уникальный путь от корневого каталога, или *полный путь*. Имя полного пути начинается с косой черты (`/`). Примером полного пути к файлу `h` может служить `/B/C/h`. Обратите внимание, что в файловой системе могут существовать два файла с именем `h`. Если полные пути к этим файлам различны, например, `/B/h` и `/B/C/h`, то с точки зрения системы у обоих файлов `h` есть уникальное имя. Все компоненты пути файла, за исключением последнего, - это каталоги. Последний компонент пути может быть именем файла.

Примечание: Длина пути не должна превышать 1023 символов.

Соответствие шаблона с символами подстановки и мета-символами:

Шаблон с символами подстановки позволяет задать сразу несколько имен файлов или каталогов.

Роль символов подстановки играют звездочка (`*`) и вопросительный знак (`?`). Метасимволы - квадратные скобки (`[ ]`), дефис (`-`) и восклицательный знак (`!`).

*Соответствие шаблонов с помощью символа подстановки *:*

Звездочка (`*`) заменяет любую последовательность или строку символов.

Звездочка (`*`) обозначает любое количество символов (ноль и более).

Примеры:

- Если в каталоге находятся следующие файлы:
  ```
  1test 2test afile1 afile2 bfile1 file file1 file10 file2 file3
  ```
  и вам нужно задать файлы, имена которых начинаются со слова `file`, то введите:
  ```
  file*
  ```
  Будут выбраны следующие файлы: `file`, `file1`, `file10`, `file2` и `file3`.
- Если вы хотите указать файлы, в имени которых содержится слово `file`, введите:
  ```
  *file*
  ```
  Будут выбраны следующие файлы: `afile1`, `afile2`, `bfile1`, `file`, `file1`, `file10`, `file2` и `file3`.

Соответствие шаблона с помощью символа подстановки ?:

Символ **?** заменяет любой символ.

Символ **?** соответствует одному символу. Примеры:

- Если вы хотите выбрать файлы, имена которых начинаются со слова **file**, после которого указан любой символ, то введите:
  ```
  file?
  ```
  Будут выбраны следующие файлы: `file1`, `file2`, `file3`.

- Для того чтобы выбрать файлы, имена которых начинаются со слова **file**, после которого указано два любых символа, введите:
  ```
  file??
  ```
  Будут выбраны следующие файлы: `file10`.

Соответствие шаблонов с помощью мета-символов оболочки [ ]:

Метасимволы представляют собой другой тип символов подстановки. В данном случае нужные символы заключаются в скобки `[ ]`. Действие таких метасимволов аналогично действию символа `?`, однако эта конструкция заменяет любой символ, заключенный в скобки.

Внутри скобок `[ ]` можно задать интервал значений с помощью дефиса (`-`). Конструкция `[[:alpha:]]` заменяет любой символ алфавита. Конструкция `[[:lower:]]` заменяет любую строчную букву алфавита.

Примеры:

- Если вы хотите выбрать файлы, имена которых оканчиваются на 1 или 2, введите:
  ```
  *file[12]
  ```
  Будут выбраны следующие файлы: `afile1`, `afile2`, `file1` и `file2`.
- Для того чтобы задать файлы, имена которых начинаются с цифры, введите:
  ```
  [0123456789]* или [0-9]*
  ```
  Будут выбраны следующие файлы: `1test` и `2test`.
- Для того чтобы задать файлы, имена которых не начинаются с буквы `a`, введите:
  ```
  [!a]*
  ```
  Будут выбраны следующие файлы: `1test`, `2test`, `bfile1`, `file`, `file1`, `file10`, `file2` и `file3`.

Соответствия шаблона и регулярные выражения:

Регулярные выражения предназначены для выбора отдельных строк из набора символьных строк. Чаще всего регулярные выражения применяются при обработке текстов.

В виде регулярных выражений можно записать большое число разнообразных строк. Большинство регулярных выражений интерпретируются в зависимости от текущей локали, однако существуют определенные правила записи регулярных выражений, общие для всех локалей.

См. следующие примеры:

Шаблон	Регулярное выражение
*	.*
?	.
[!a]	[^a]
[abc]	[abc]
[[:alpha:]]	[[:alpha:]]

Сведения о синтаксисе приведены в описании команды **awk** книги *Справочник по командам, том 1*.

Управление файлами

В AIX предусмотрены разнообразные средства для работы с файлами. Обычно текстовые файлы создаются с помощью текстовых редакторов.

В среде UNIX чаще всего применяются редакторы `vi` и `ed`. Выберите тот редактор, который удобнее для вас.

Для создания файлов могут применяться операции перенаправления ввода-вывода. Вывод команды можно записать в новый файл или добавить в существующий файл.

Кроме операций создания и изменения файла, предусмотрены операции копирования и перемещения файла из одного каталога в другой, а также операция переименования файлов, предназначенная для создания различных версий файла или записи одного и того же файла под разными именами. Во время работы над различными проектами вам, скорее всего, потребуется создавать каталоги.

Кроме того, может возникнуть необходимость удалить некоторые файлы. Хранение устаревших или ненужных файлов в каталоге может затруднить работу с ним. Для освобождения памяти в системе следует своевременно удалять те файлы, которые больше не нужны.

Понятия, связанные с данным:

“Перенаправление ввода и вывода” на стр. 342
Операционная система AIX позволяет управлять потоками ввода и вывода данных с помощью специальных символов и команд ввода-вывода.

Удаление файлов (команда rm):

Команда **rm** применяется для удаления ненужных файлов.

Команда **rm** позволяет удалить из каталога записи об указанном файле, группе файлов или выбранных файлах. Перед удалением файла команда **rm** не запрашивает подтверждение у пользователя. Для применения этой команды не нужны права на чтение и запись в файл. Однако у вас должны быть права на запись в каталог, содержащий данный файл.

Ниже приведены примеры применения команды **rm**:

- Для удаления файла `myfile` введите:
  ```
  rm myfile
  ```
- Для удаления всех файлов из каталога `mydir` введите:
  ```
  rm -i mydir/*
  ```

  После появления имени очередного файла введите `y` и нажмите Enter, если вы хотите удалить этот файл. Для того чтобы сохранить файл, нажмите Enter.

Сведения о синтаксисе приведены в описании команды **rm** в книге *Справочник по командам, том 4*.

Перемещение и переименование файлов (команда mv):

Для перемещения и переименования файлов и каталогов применяется команда **mv**. Если вы перемещаете файл или каталог в другой каталог и при этом не указываете новое имя, то этот файл или каталог сохраняет свое исходное имя.

Внимание: Если вы не укажете флаг **-i**, команда **mv** может заменить существующие файлы. Если вы укажете флаг **-i**, то перед заменой файла будет запрашиваться подтверждение. Если вы укажете флаг **-f**, то подтверждение запрашиваться не будет. Если вы укажете оба флага **-f** и **-i**, то первый из них будет игнорироваться.

Перемещение файлов с помощью команды mv

Ниже приведены примеры применения команды **mv**:

- Для переименования файла и перемещения его в другой каталог введите:
  ```
  mv intro manual/chap1
  ```

  Файл `intro` будет перемещен в файл `manual/chap1`. При этом файл **intro** удаляется из текущего каталога и добавляется в каталог `manual` под именем `chap1`.

- Для перемещения файла в другой каталог с сохранением имени файла введите:
  ```
  mv chap3 manual
  ```

  Файл `chap3` будет перемещен в файл `manual/chap3`.

Переименование файлов с помощью команды mv

Команда **mv** позволяет изменить имя файла, не перемещая его в другой каталог.

Для переименования файла введите:

```
mv appendix apndx.a
```

Файл `appendix` будет переименован в `apndx.a`. Если файл с именем `apndx.a` уже существует, его содержимое будет заменено на содержимое файла `appendix`.

Сведения о синтаксисе приведены в описании команды **mv** в книге *Справочник по командам, том 3*.

Копирование файлов (команда cp):

Команда **cp** копирует файл или каталог, указанный в параметре *Исходный_файл* или *Исходный_каталог*, в файл или каталог, заданный в параметре *Целевой_файл* или *Целевой_каталог*.

Если *Целевой_файл* существует, то его содержимое заменяется без предупреждения. При копировании нескольких *Исходных_файлов* должен быть указан целевой каталог.

Если файл с указанным именем уже существует в целевом каталоге, то его содержимое будет заменено на содержимое исходного файла. В связи с этим рекомендуется указывать *новое* имя для копии файла, чтобы в целевом каталоге не было файла с тем же именем.

Для помещения копии файла *Исходный_файл* в определенный каталог укажите путь к этому каталогу в параметре *Целевой_каталог*. При копировании в другой каталог файлы сохраняют свои имена, если только в конце пути не было указано новое имя файла. Команда **cp** также копирует полные каталоги в другие каталоги. Для этого нужно указать флаг **-r** или **-R**.

Для копирования специальных файлов устройств нужно указать флаг **-R**. Флаг **-R** позволяет скопировать специальный файл в файл с новым именем в другом каталоге. Если вы укажете флаг **-r**, то команда **cp** скопирует специальный файл в стандартный файл.

Ниже приведены примеры применения команды **cp**:

- Для создания копии файла в текущем каталоге введите:
  ```
  cp prog.c prog.bak
  ```

  Эта команда копирует файл `prog.c` в файл `prog.bak`. Если файл `prog.bak` не существует, то команда **cp** создаст его. Если такой файл существует, команда **cp** заменит его на копию файла `prog.c`.
- Для того чтобы скопировать файл из текущего каталога в другой каталог, введите:
  ```
  cp jones /home/nick/clients
  ```

  Эта команда копирует файл `jones` в файл `/home/nick/clients/jones`.
- Для того чтобы скопировать все файлы каталога в другой каталог, введите:
  ```
  cp /home/janet/clients/* /home/nick/customers
  ```

  Все файлы из каталога `clients` будут скопированы в каталог `customers`.
- Для того чтобы скопировать набор файлов в другой каталог, введите:
  ```
  cp jones lewis smith /home/nick/clients
  ```

  Эта команда копирует файлы `jones`, `lewis` и `smith` текущего каталога в каталог `/home/nick/clients`.

- Для копирования всех файлов, соответствующих указанному шаблону, введите:
  ```
  cp programs/*.c .
  ```

  Все файлы из каталога `programs` с расширением `.c` будут скопированы в текущий каталог, обозначенный точкой (`.`). Между символом `c` и последней точкой должен стоять пробел.

Сведения о синтаксисе приведены в описании команды **cp** в книге *Справочник по командам, том 1*.

Поиск файлов (команда find):

Команда **find** просматривает каталоги, пути к которым заданы в параметре *Путь*, и выполняет рекурсивный поиск файлов, соответствующих булевскому выражению, формат которого описан ниже.

Вывод команды **find** зависит от параметра *Выражение*.

Ниже приведены примеры применения команды `find`:

- Для просмотра списка всех файлов с именами `.profile` введите:
  ```
  find / -name .profile
  ```

  Поиск файлов будет выполняться во всей файловой системе, а выводом команды будет список каталогов, где хранятся файлы с именем `.profile`. Косая черта (/) означает, что команда **find** должна просмотреть корневой каталог (`/root`) и все его подкаталоги.

  Для экономии времени можно сузить область поиска, указав каталоги, в которых предположительно находятся файлы.
- Для получения списка файлов текущего каталога, для которых установлены права доступа `0600`, введите:
  ```
  find . -perm 0600
  ```

  Появится список файлов, для которых установлено разрешение на чтение и запись *только* для владельца. Точка (`.`) сообщает команде **find** о необходимости поиска в текущем каталоге и во всех его подкаталогах. Информация о значениях прав доступа приведена в описании команды **chmod**.
- Для получения списка файлов с определенным кодом прав доступа из нескольких каталогов введите:
  ```
  find manual clients proposals -perm -0600
  ```

  Появится список имен файлов, для которых установлено разрешение на чтение и запись только для владельца и, возможно, другие разрешения. Поиск будет выполняться в каталогах `manual`, `clients` и `proposals` и их подкаталогах. В предыдущем примере было задано выражение `-perm 0600`, позволяющее найти файлы с кодом прав доступа, который в точности равен `0600`. В данном примере указано выражение `-perm -0600`, позволяющее найти все файлы с кодом прав доступа `0600` и выше. В этом случае будут также выбраны файлы с кодами прав доступа 0622 и 2744.
- Для просмотра списка всех файлов текущего каталога, которые были изменены в течение текущих суток, введите:
  ```
  find . -ctime 1
  ```
- Для получения списка обычных файлов, для которых создано несколько связей, введите:
  ```
  find . -type f -links +1
  ```

  Появится список обычных файлов (`-type f`), для которых создано несколько связей ((`-links +1`).

  Примечание: Для каждого каталога создается по крайней мере две связи: запись в родительском каталоге и запись `.` (точка) внутри самого каталога. Дополнительная информация приведена в описании команды **ln**.
- Для того чтобы получить список файлов, размер которых равен 414 байт, введите:
  ```
  find . -size 414c
  ```

Сведения о синтаксисе приведены в описании команды **find** в книге *Справочник по командам, том 2*.

Просмотр типа файла (команда file):

Команда **file** просматривает содержимое файлов, указанных в параметре *Файл* или **-f** *Список_файлов*, и выполняет набор тестов. Типы файлов записываются в стандартный вывод.

В файле ASCII команда **file** проверяет первые 512 байт и определяет его язык. Если файл отличен от ASCII, то команда **file** пытается определить, является ли он двоичным или текстовым файлом, содержащим символы из расширенного набора.

Если в параметре *Файл* указан исполняемый или объектный файл, версия которого больше 0, то команда **file** выводит номер версии.

Информацию о файлах, содержащих сигнатуру, т.е. числовую или строковую константу, задающую тип, команда **file** получает из файла `/etc/magic`.

Ниже приведены примеры применения команды **file**:

- Для просмотра типа информации, содержащейся в файле `myfile`, введите:
  ```
  file myfile
  ```
  Будет показан тип файла `myfile` (например, каталог, файл данных, ASCII, исходный файл программы на языке C или архив).
- Для просмотра сведений о типах всех файлов, перечисленных в файле `filenames.lst`, введите:
  ```
  file -f filenames.lst
  ```
  Появится список типов всех файлов, указанных в `filenames.lst`. Тип каждого файла будет указан на отдельной строке.
- Для создания файла `filenames.lst`, содержащего список имен файлов из текущего каталога, введите:
  ```
  ls > filenames.lst
  ```
  Внесите в файл `filenames.lst` необходимые изменения.

Сведения о синтаксисе приведены в описании команды **file** в книге *Справочник по командам, том 2*.

Команды просмотра содержимого файла (pg, more, page и cat):

Команды **pg**, **more** и **page** предназначены для просмотра содержимого файла и позволяют управлять скоростью просмотра файлов.

С помощью команды **cat** вы можете просмотреть содержимое одного или нескольких файлов. Для постраничного просмотра файла укажите команду **cat** в сочетании с командой **pg**.

Кроме того, содержимое файлов можно просмотреть с помощью операций перенаправления ввода-вывода.

Понятия, связанные с данным:

“Перенаправление ввода и вывода” на стр. 342
Операционная система AIX позволяет управлять потоками ввода и вывода данных с помощью специальных символов и команд ввода-вывода.

Применение команды pg:

Команда **pg** предназначена для постраничного просмотра содержимого файлов, указанных в параметре **Файл**.

Если в параметре **Файл** вы укажете дефис (-) или запустите команду **pg** без параметров, то команда **pg** будет считывать данные из стандартного потока ввода. После каждой страницы данных появляется приглашение. Если вы нажмете клавишу Enter, то появится следующая страница. Для повторного просмотра предыдущих страниц предусмотрены подкоманды команды **pg**.

Например, для постраничного просмотра содержимого файла `myfile` введите:

```
pg myfile
```

Сведения о синтаксисе приведены в описании команды **pg** книги *Справочник по командам, том 4*.

Применение команда more или page:

Команды **more** и **page** предназначены для постраничного просмотра текста.

Они выводят очередную страницу текста, а также *имя файла* и индикатор доли просмотренного текста в процентах (например, `myfile (7%)`). Если вы нажмете клавишу Enter, то появится следующая строка. Если вы нажмете пробел, то команда **more** покажет следующую страницу текста.

Примечание: В некоторых моделях терминалов при выполнении команды **more** происходит не прокрутка, а очистка экрана перед выводом следующей страницы текста.

Например, для просмотра содержимого файла `myfile` введите:

```
more myfile
```

Для просмотра следующей страницы нажмите пробел.

Сведения о синтаксисе приведены в описании команды **more** книги *Справочник по командам, том 3*.

Команда cat:

Команда **cat** последовательно считывает содержимое файлов, указанных в параметре *Файл*, и записывает его в стандартный поток вывода.

Примеры:

- Для просмотра содержимого файла `notes` введите:

  ```
  cat notes
  ```

  Если длина файла превышает 24 строки, то для просмотра дополнительных строк необходимо прокрутить содержимое окна. Для постраничного просмотра файла введите команду **pg**.
- Для просмотра содержимого файлов `notes`, `notes2` и `notes3` введите:

  ```
  cat notes notes2 notes3
  ```

Сведения о синтаксисе приведены в описании команды **cat** книги *Справочник по командам, том 1*.

Поиск текстовых строк в файлах (команда grep):

Команда **grep** предназначена для поиска строк, соответствующих шаблону, заданному в параметре *Шаблон*. Каждая найденная строка записывается в стандартный поток вывода.

Ниже приведены примеры применения команды **grep**:

- Для поиска файла с именем `pgm.s` шаблон может содержать специальные символы *, ^, ?, [,], \(, \), \{ и \}. Например, для поиска строк, начинающихся с прописной или строчной буквы, введите:

  ```
  $grep "^[a-zA-Z]" pgm.s
  ```

  Появится список строк файла `pgm.s`, начинающихся с буквы.

- Для просмотра всех строк файла `sort.c`, которые не совпадают с шаблоном, введите:
  ```
  grep -v bubble sort.c
  ```

  оявится список строк файла `sort.c`, не содержащих слова `bubble`.
- Для просмотра всех строк вывода команды **ls**, которые содержат слово `staff`, введите:
  ```
  ls -l | grep staff
  ```

Сведения о синтаксисе приведены в описании команды **grep** в книге *Справочник по командам, том 2*.

Сортировка текстовых файлов (команда sort):

Команда **sort** упорядочивает строки файлов, заданных в параметре **Файл**, в алфавитном порядке и записывает результат в стандартный вывод.

Если в параметре **Файл** задано несколько файлов, то команда **sort** выполняет конкатенацию содержимого файлов, а затем сортирует строки полученного текста.

Примечание: Команда **sort** учитывает регистр символов, причем у прописных букв приоритет выше, чем у строчных (это зависит от локали).

В следующих примерах предполагается, что файл `names` содержит следующие данные:

```
marta
denise
joyce
endrica
melanie
```

а файл `states` содержит следующие данные:

```
texas
colorado
ohio
```

- Для сортировки содержимого файла `names` введите:
  ```
  sort names
  ```

  Появится приблизительно следующая информация:
  ```
  denise
  endrica
  joyce
  marta
  melanie
  ```
- Для сортировки содержимого файлов `names` и `states` введите:
  ```
  sort names states
  ```

  Появится приблизительно следующая информация:
  ```
  colorado
  denise
  endrica
  joyce
  marta
  melanie
  ohio
  texas
  ```
- Для замены исходного содержимого файла `names` отсортированными данными введите:
  ```
  sort -o names names
  ```

  В результате файл `names` будет содержать те же данные, но в упорядоченном виде.

Сведения о синтаксисе приведены в описании команды **sort** в книге *Справочник по командам, том 5*.

Сравнение файлов (команда diff):

Для сравнения текстовых файлов служит команда **diff**. С ее помощью можно сравнивать как отдельные файлы, так и содержимое каталогов.

Если команда **diff** сравнивает обычные файлы или текстовые файлы из разных каталогов, она выдает список несовпадающих строк файлов.

Ниже приведены примеры применения команды **diff**:

- Для сравнения двух файлов введите:
  ```
  diff chap1.bak chap1
  ```
  Появится список различий файлов `chap1.bak` и `chap1`.
- Для сравнения двух файлов с игнорированием пробелов введите:
  ```
  diff -w prog.c.bak prog.c
  ```
  Если файлы отличаются друг от друга только количеством пробелов и символов табуляции, разделяющих слова, то команда **diff -w** будет считать их одинаковыми.

Сведения о синтаксисе приведены в описании команды **diff** в книге *Справочник по командам, том 2*.

Подсчет числа слов, строк и байт в файле (команда wc):

Команда **wc** подсчитывает число строк, слов и байт в файлах, указанных в параметре *Файл*.

Если параметр *Файл* не задан, то данные считываются из стандартного потока ввода. Результат работы команды записывается в стандартный поток вывода и в нем указывается суммарное число строк, слов и байт во всех заданных файлах. Флаги команды определяют порядок вывода. Под *словом* понимается символьная строка, отделенная пробелом, символом табуляции или символом новой строки.

Если в команде заданы имена файлов, то эти имена выводятся вместе с результатом подсчета.

Примеры:

- Для подсчета числа строк, слов и байт в файле с именем `chap1` введите:
  ```
  wc chap1
  ```
  Будет показано число строк, слов и байт в файле `chap1`.
- Для подсчета только числа слов и байт введите:
  ```
  wc -cw chap*
  ```
  Появится число байт и слов для каждого файла, имя которого начинается с символов `chap`, и общий результат.

Сведения о синтаксисе приведены в описании команды **wc** книги *Справочник по командам, том 6*.

Просмотр первых строк файла (команда head):

Команда **head** записывает в стандартный поток вывода первые несколько строк каждого из заданных файлов или первые несколько строк из стандартного потока ввода.

Если в команде **head** не указаны флаги, то по умолчанию выводятся первые 10 строк.

Например, для просмотра первых пяти строк файла `Test` введите:

```
head -5 Test
```

Сведения о синтаксисе приведены в описании команды **head** книги *Справочник по командам, том 2*.

Просмотр последних строк файла (команда tail):

Команда **tail** записывает в стандартный поток вывода содержимое файла, заданного в параметре *Файл*, начиная с указанной позиции.

Примеры:

- Для просмотра последних 10 строк файла `notes` введите:
  ```
  tail notes
  ```
- Для просмотра указанного числа строк с конца файла `notes` введите:
  ```
  tail -20 notes
  ```
- Для постраничного просмотра файла `notes`, начиная с двухсотого байта, введите:
  ```
  tail -c +200 notes | pg
  ```
- Для отслеживания изменений файла `accounts` введите:
  ```
  tail -f accounts
  ```

  Будут показаны последние 10 строк файла `accounts`. Команда **tail** будет продолжать показывать новые строки по мере их добавления к файлу `accounts`. Для окончания просмотра нажмите клавиши Ctrl-C.

Сведения о синтаксисе приведены в описании команды **tail** книги *Справочник по командам, том 5*.

Вырезание разделов текстовых файлов (команда cut):

Команда **cut** записывает выбранные байты, символы или поля из каждой строки файла в стандартный вывод.

Примеры:

- Для просмотра отдельных полей из каждой строки файла введите:
  ```
  cut -f1,5 -d: /etc/passwd
  ```

  Будут показаны поля имени входа в систему и полного имени пользователя из системного файла паролей. Это первое и пятое поля (`-f1,5`), разделенные двоеточием (`-d:`).
- Если в файле `/etc/passwd` указаны следующие сведения:
  ```
  su:*:0:0:User with special privileges:/:/usr/bin/sh
  daemon:*:1:1::/etc:
  bin:*:2:2::/usr/bin:
  sys:*:3:3::/usr/src:
  adm:*:4:4:system administrator:/var/adm:/usr/bin/sh
  pierre:*:200:200:Pierre Harper:/home/pierre:/usr/bin/sh
  joan:*:202:200:Joan Brown:/home/joan:/usr/bin/sh
  ```

  то результат работы команды **cut** будет выглядеть следующим образом:
  ```
  su:User with special privileges
  daemon:
  bin:
  sys:
  adm:system administrator
  pierre:Pierre Harper
  joan:Joan Brown
  ```

Сведения о синтаксисе приведены в описании команды **cut** книги *Справочник по командам, том 1*.

Вставка фрагментов текстовых файлов (команда paste):

Команда **paste** объединяет выбранные строки из нескольких (не более 12) файлов в один.

Примеры:

- Пусть у вас есть файл с именем `names`, содержащий следующий текст:

  ```
  rachel
  jerry
  mark
  linda
  scott
  ```

  и файл с именем `places`, содержащий следующий текст:

  ```
  New York
  Austin
  Chicago
  Boca Raton
  Seattle
  ```

  и файл с именем `dates`, содержащий следующий текст:

  ```
  February 5
  March 13
  June 21
  July 16
  November 4
  ```

  Для того чтобы объединить содержимое файлов `names`, places и dates, введите:

  ```
  paste names places dates > npd
  ```

  Будет создан файл с именем `npd`, в первом столбце которого будут размещены данные из файла `names`, во втором столбце - данные из файла `places`, а в третьем столбце - данные из файла `dates` Теперь файл `npd` содержит следующую информацию:

  ```
  rachel          New York        February 5
  jerry           Austin          March 13
  mark            Chicago         June 21
  linda           Boca Raton      July 16
  scott           Seattle         November 4
  ```

  Имя, город и дата в каждой строке отделены друг от друга символом табуляции. Эти столбцы не выровнены так как символ табуляции вставляется через каждые восемь символов.

- Для того чтобы отделить столбцы друг от друга символом, отличным от знака табуляции, введите:

  ```
  paste -d"!@" names places dates > npd
  ```

  Эта команда устанавливает в качестве альтернативных разделителей символы `!` и `@`. Если взять те же файлы `names`, `places` и `dates`, что и в примере 1, то файл `npd` будет содержать следующую информацию:

  ```
  rachel!New York@February 5
  jerry!Austin@March 13
  mark!Chicago@June 21
  linda!Boca Raton@July 16
  scott!Seattle@November 4
  ```

- Для просмотра списка файлов текущего каталога в виде четырех столбцов введите:

  ```
  ls | paste - - - -
  ```

  Для команды **paste** каждый дефис (`-`) означает создание столбца, содержащего данные из стандартного потока ввода. Первая строка размещается в первом столбце, вторая - во втором, и т.д.

Сведения о синтаксисе приведены в описании команды **paste** книги *Справочник по командам, том 4*.

Нумерация строк в текстовых файлах (команда nl):

Команда **nl** считывает указанный файл (по умолчанию - стандартный поток ввода), нумерует строки и записывает их в стандартный поток вывода.

Примеры:

- Для нумерации непустых строк введите:
  ```
  nl chap1
  ```
  Будет показан пронумерованный список непустых строк файла `chap1`.
- Для нумерации всех строк введите:
  ```
  nl -ba chap1
  ```
  Будет показан пронумерованный список всех строк файла `chap1`, с учетом пустых строк.

Сведения о синтаксисе приведены в описании команды **nl** книги *Справочник по командам, том 4*.

Удаление столбцов в текстовых файлах (команда colrm):

Команда **colrm** предназначена для удаления указанных столбцов символов из файла. Входные данные считываются из стандартного потока ввода. Результат работы команды записывается в стандартный поток вывода.

Если в команде указан только один параметр, то удаляются все столбцы, начиная с указанного. Если в команде заданы два параметра, то удаляются все столбцы из указанного диапазона.

Примечание: Нумерация столбцов символов начинается с 1.

Примеры:

- Для удаления столбцов из файла `text.fil` введите:
  ```
  colrm 6 < text.fil
  ```
  Если в файле `text.fil` содержится:
  ```
  123456789
  ```
  то результат работы команды **colrm** будет выглядеть следующим образом:
  ```
  12345
  ```

Сведения о синтаксисе приведены в описании команды **colrm** книги *Справочник по командам, том 1*.

Связи файлов и каталогов

Связями называются записи о соответствии между именами файлов и номерами индексных узлов (i-узлов). Номера индексных узлов - это внутренние абсолютные идентификаторы файлов в AIX. Поскольку записи каталогов состоят из имен файлов и соответствующих номеров i-узлов, их можно считать связями.

Фактически уникальным идентификатором файла как физического объекта является i-узел, а не имя этого файла. Любой файл (номер i-узла) может быть связан с несколькими именами одновременно. Например, предположим, что номер i-узла 798 - это файл с информацией о продажах ярославского филиала компании за июнь. Пусть этому файлу присвоено следующее имя:

Номер i-узла	Имя файла
798	**memo**

Поскольку логически этот файл можно считать относящимся к каталогам `sales` и `yaroslavl`, в этих каталогах можно создать связи с ним. Для создания связей с этими каталогами можно воспользоваться командой **ln**. После создания связей у файла будет три имени:

Номер i-узла	Имя файла
798	`memo`
798	`sales/june`
798	`yaroslavl/june.sales`

Теперь при просмотре любого из этих файлов с помощью команды **pg** или **cat** будет выдаваться одна и та же информация. Если вы измените содержимое любого из этих файлов, то одновременно будут изменены и остальные два файла, поскольку все они связаны с одним номером i-узла.

Типы связей:

Есть два типа связей: жесткие и символьные.

Для создания связей применяется команда **ln**. Связи бывают следующих типов:

Элемент	Описание
жесткие связи	Позволяют создавать альтернативные имена файлов. Наличие жесткой связи с файлом гарантирует существование этого файла. При удалении последней жесткой связи автоматически удаляются номер i-узла и хранящиеся в нем данные. Жесткие связи можно создавать только между файлами, находящимися в одной файловой системе.
символьные связи	Позволяют создавать альтернативные имена файлов в других файловых системах. Символьная связь - это особый файл, в котором хранится путь к исходному файлу. При обращении к символьной связи процесс пытается открыть файл, указанный в ней. Наличие символьной связи не гарантирует существования файла и не препятствует его удалению из файловой системы.

Примечание: Независимо от числа символьных связей, файл принадлежит пользователю, создавшему его. Права доступа к файлу могут изменять только его владелец и пользователь root. Однако изменять связь с файлом (а, значит, и сам файл) могут все пользователи, которым разрешен доступ к этой связи.

Файл или каталог существует до тех пор, пока с ним существует хотя бы одна жесткая связь. Команда **ls -l** в числе прочей информации для каждого файла и каталога выдает число жестких связей. Для операционной системы все жесткие связи равнозначны, независимо от того, какая из них была создана первой.

Создание связей между файлами (команда ln):

Для создания связей между файлами применяется команда **ln**. Фактически связи - это альтернативные имена одного и того же файла.

Связи создаются с помощью присваивания альтернативных имен к исходным файлам. С помощью связей можно предоставить различным пользователям доступ к одному большому файлу (например, к базе данных или списку рассылки), не создавая дополнительных копий этого файла. Применение связей позволяет не только сэкономить дисковую память, но и избавляет от необходимости синхронизировать один и тот же файл в различных каталогах.

Команда **ln** создает связь для файла **исходный файл** с именем **целевой файл** или с тем же именем в каталоге **целевой каталог**. По умолчанию команда **ln** создает жесткие связи. Для создания символьной связи нужно указать в команде **ln** флаг **-s**.

Примечание: Без флага **-s** можно создавать связи только в тех же файловых системах, в которых находятся исходные файлы.

При создании связи с новым именем в текущем каталоге в команде ln можно указывать только один файл. При создании связей в новом каталоге можно указывать несколько файлов одновременно.

Параметр **целевой файл** указывать не обязательно. Если он не указан, команда **ln** создает файл в текущем каталоге. Новый файл наследует имя, указанное в параметре **исходный файл**.

Примеры:

- Для создания связи с файлом chap1 введите следующую команду:
  ```
  ln -f chap1 intro
  ```
  Для файла chap1 будет создана связь с именем intro. Флаг **-f** указывает, что если файл intro не существует, его нужно создать. Если файл intro существует, его нужно заменить на связь с файлом chap1. В результате объекты chap1 и intro будут соответствовать одному и тому же файлу.
- Для того чтобы создать для файла index одноименную связь в каталоге manual, введите следующую команду:
  ```
  ln index manual
  ```
  В результате для файла index будет создана связь manual/index.
- Для того чтобы создать связи для нескольких файлов в другом каталоге, введите следующую команду:
  ```
  ln chap2 jim/chap3 /home/manual
  ```
  В результате для файла chap2 будет создана связь /home/manual/chap2, а для файла jim/chap3 - связь /home/manual/chap3.
- В команде **ln** можно указывать шаблоны имен файлов, например:
  ```
  ln manual/* .
  ```
  Примечание: В предыдущей команде между звездочкой и точкой должен быть указан пробел.
  Для всех файлов из каталога manual будут созданы связи в текущем каталоге, обозначенном точкой (.), с теми же именами, что и в каталоге manual.
- Для создания символьной связи введите:
  ```
  ln -s /tmp/toc toc
  ```
  Эта команда создает символьную связь с именем **toc** в текущем каталоге. Файл toc указывает на файл /tmp/toc. Если файл /tmp/toc существует, команда **cat toc** показывает его содержимое.
- Следующая команда равносильна предыдущей, хотя в ней и не указан параметр **целевой файл**:
  ```
  ln -s /tmp/toc
  ```

Сведения о синтаксисе приведены в описании команды **ln** книги *Справочник по командам, том 3*.

Команды для удаления связей с файлами:

С помощью команды **rm** можно удалить вязи с файлами.

При удалении жесткой связи файл удаляется только в том случае, если это последняя жесткая связь с файлом. Одновременно с файлом (i-узлом) удаляются и содержащиеся в нем данные. После удаления i-узла его номер освобождается и может быть присвоен какому-либо вновь созданному файлу.

Сведения о синтаксисе приведены в описании команды **rm** книги *Справочник по командам, том 3*.

Файлы DOS

Операционная система AIX позволяет работать с файлами DOS.

Скопируйте на дискету файлы DOS, с которыми вы будете работать. Эти файлы можно скопировать в каталог базовой операционной системы в нужном формате и записать их обратно на дискету в формате DOS.

Примечание: Символы подстановки * и ? (звездочка и вопросительный знак) не поддерживаются в командах, описанных в этом разделе (хотя они и применяются в оболочке базовой операционной системы). Если расширение имени файла не указано, то считается, что вы указали пустое расширение.

Копирование файлов DOS в файлы базовой операционной системы:

Команда **dosread** копирует указанный файл DOS в указанный файл базовой операционной системы.

Примечание: Все соглашения о присвоении имен DOS остаются в силе, за единственным исключением. Поскольку обратная косая черта (\) может иметь специальное значение в базовой операционной системе, для разделения имен подкаталогов в пути к файлу DOS следует применять косую черту (/).

Примеры:

- Для копирования текстового файла с именем `chap1.doc` с дискеты в формате DOS в файловую систему базовой операционной системы введите:

  ```
  dosread -a chap1.doc chap1
  ```

  Текстовый файл DOS `\CHAP1.DOC` из устройства по умолчанию `/dev/fd0` будет скопирован в файл базовой операционной системы `chap1`, расположенный в текущем каталоге.
- Для копирования двоичного файла с дискеты в формате DOS в файловую систему базовой операционной системы введите:

  ```
  dosread -D/dev/fd0 /survey/test.dta /home/fran/testdata
  ```

  Файл данных `\SURVEY\TEST.DTA` в формате DOS из устройства `/dev/fd0` будет скопирован в файл базовой операционной системы `/home/fran/testdata`.

Сведения о синтаксисе приведены в описании команды **dosread** книги *Справочник по командам, том 2*.

Копирование файлов базовой операционной системы в файлы DOS:

Команда **doswrite** копирует указанный файл базовой операционной системы в указанный файл DOS.

Примечание: Все соглашения о присвоении имен DOS остаются в силе, за единственным исключением. Поскольку обратная косая черта (\) может иметь специальное значение в базовой операционной системе, для разделения имен подкаталогов в пути к файлу DOS следует применять косую черту (/).

Примеры:

- Для копирования текстового файла с именем `chap1.doc` из файловой системы базовой операционной системы на дискету в формате DOS введите:

  ```
  doswrite -a chap1 chap1.doc
  ```

  Файл `chap1` из текущего каталога базовой операционной системы будет скопирован в текстовый файл `\CHAP1.DOC` DOS в устройстве `/dev/fd0`.
- Для копирования двоичного файла `/survey/test.dta` из файловой системы базовой операционной системы на дискету в формате DOS введите:

  ```
  doswrite -D/dev/fd0 /home/fran/testdata /survey/test.dta
  ```

  Файл данных базовой операционной системы `/home/fran/testdata` будет скопирован в файл DOS `\SURVEY\TEST.DTA` в устройстве `/dev/fd0`.

Сведения о синтаксисе приведены в описании команды **doswrite** книги *Справочник по командам, том 2*.

Удаление файлов DOS:

Команда **dosdel** удаляет указанный файл DOS.

Примечание: Все соглашения о присвоении имен DOS остаются в силе, за единственным исключением. Поскольку обратная косая черта (\) может иметь специальное значение в базовой операционной системе, для разделения имен подкаталогов в пути к файлу DOS следует применять косую черту (/).

Перед проверкой диска команда **dosdel** преобразует строчные буквы в прописные. Поскольку все имена файлов рассматриваются как полные (не относительные), нет необходимости добавлять в начало косую черту (/).

Например, для удаления файла DOS с именем `file.ext` в устройстве по умолчанию (`/dev/fd0`) введите:

```
dosdel file.ext
```

Сведения о синтаксисе приведены в описании команды **dosdel** книги *Справочник по командам, том 2*.

Просмотр содержимого каталога DOS:

Команда **dosdir** показывает информацию об указанных файлах и каталогах DOS.

Примечание: Все соглашения о присвоении имен DOS остаются в силе, за единственным исключением. Поскольку обратная косая черта (\) может иметь специальное значение в базовой операционной системе, для разделения имен подкаталогов в пути к файлу DOS следует применять косую черту (/).

Перед проверкой диска команда **dosdir** преобразует строчные буквы в прописные. Поскольку все имена файлов рассматриваются как полные (не относительные), нет необходимости добавлять в начало символ / (косая черта).

Например, для чтения каталога файлов DOS в устройстве `/dev/fd0` введите:

```
dosdir
```

Команда вернет список имен файлов и информацию об объеме свободного пространства на диске в формате, показанном ниже.

```
PG3-25.TXT
PG4-25.TXT
PG5-25.TXT
PG6-25.TXT
Свободное пространство: 312320 байт
```

Сведения о синтаксисе приведены в описании команды **dosdir** книги *Справочник по командам, том 2*.

Обзор команд для файлов

Ниже описаны команды для работы с файлами, процедуры обработки файлов и файлы DOS. Приведен список команд для привязки файлов и каталогов.

Таблица 57. Команды для работы с файлами

Элемент	Описание
*	Символ подстановки, соответствует любым символам
?	Символ подстановки, соответствует любому одному символу
[ ]	Метасимволы, соответствуют указанным внутри символам.

Таблица 58. Команды для процедур работы с файлами

Элемент	Описание
cat	Объединяет или показывает файлы
cmp	Сравнивает два файла
colrm	Выдает отдельные столбцы файла
cp	Копирует файлы
cut	Записывает выбранные байты, символы или поля из каждой строки файла
diff	Сравнивает текстовые файлы
file	Определяет тип файла
find	Находит файлы, содержащие указанное выражение
grep	Выполняет поиск шаблона в файле
head	Показывает начальные строки или байты одного или нескольких файлов
more	Показывает следующую страницу текста на экране
mv	Перемещает файлы
nl	Нумерует строки файла
pg	Форматирует файлы для вывода на экран
rm	Удаляет файлы или каталога (удаляет ссылки на них)
paste	Вставляет строки из нескольких файлов или смежные строки в один файл
sort	Упорядочивает файлы, вставляет уже упорядоченные файлы и проверяет, упорядочены ли файлы
tail	Записывает файл в стандартный вывод, начиная с указанного места
wc	Подсчитывает число строк, слов и байтов в файле

Таблица 59. Команды для создания ссылок на файлы и каталоги

Элемент	Описание
ln	Связывает файлы и каталоги

Таблица 60. Команды для работы с файлами DOS

Элемент	Описание
dosdel	Удаляет файлы DOS
dosdir	Показывает файлы DOS в каталоге
dosread	Копирует файлы DOS в файлы Базовой операционной системы
doswrite	Копирует файлы Базовой операционной системы в файлы DOS

Оболочки операционной системы

Интерфейс операционной системы называется *оболочкой*.

Оболочка - это самый внешний уровень операционной системы. Оболочки содержат в себе язык программирования для управления процессами и файлами, а также запуска и управления другими программами. Оболочка управляет взаимодействием между вами и операционной системой, показывая вам приглашения для ввода, интерпретируя ввод для операционной системы, а затем обрабатывая результирующий вывод операционной системы.

Оболочки предоставляют вам способ взаимодействия с операционной системой. Это взаимодействие проводится или интерактивно (ввод с клавиатуры обрабатывается немедленно), или в качестве сценария оболочки. *Сценарий оболочки* - это последовательность команд оболочки и операционной системы, которая хранится в файле.

Когда вы входите в систему, она находит имя программы оболочки для выполнения. После запуска оболочка показывает приглашение командной строки. Этим приглашением обычно является $ (знак доллара). При вводе команды и нажатии клавиши Enter оболочка вычисляет команду и пытается ее

выполнить. В зависимости от инструкций команды оболочка записывает вывод команды на экран или перенаправляет вывод. Затем она снова показывает приглашение командной строки и ожидает ввода другой команды.

Командная строка - это строка, в которой вы осуществляете ввод. Она содержит приглашение оболочки. Основным форматом каждой строки является следующий:

```
$ Команда Аргумент(ы)
```

Оболочка рассматривает первое слово командной строки (до первого пробела) как команду, а все последующие слова как аргументы.

Примечание: Когда файл `libc.a` перемещен или переименован, сообщение об ошибке `Killed` показывается в оболочке, так как файл `libc.a` не доступен в системе для загрузки и выполнения утилит. Команда **recsh** вызывает оболочку восстановления, которая предоставляет возможность переименовать файл `libc.a`, если он случайно перемещен.

Задачи, связанные с данной:

“Просмотр введенных ранее команд с помощью команды (history)” на стр. 131
Команда **history** применяется для просмотра команд, введенных ранее.

Основные сведения об оболочке

Перед началом работы с различными типами оболочек, доступных для AIX, необходимо ознакомиться с основными терминами и функциями.

Доступные оболочки:

Ниже описаны оболочки, поставляемые с AIX.

- Оболочка Korn (запускается командой **ksh**)
- Оболочка Bourne (запускается командой **bsh**)
- Оболочка с ограничениями (ограниченная версия оболочки Bourne, запускается командой **Rsh**)
- Оболочка POSIX (или Korn, для запуска применяется команда **psh**)
- Оболочка с ограничениями для оболочки Korn (**ksh** и **ksh93**). Оболочки **ksh** и **ksh93** предоставляются со своими ограниченными эквивалентами **rksh** и **rksh93**.
- Оболочка по умолчанию (запускается командой **sh**)
- Оболочка C (запускается командой **csh**)
- Защищенная оболочка (ограниченная версия оболочки Korn, запускается командой **tsh**)
- Удаленная оболочка (запускается командой **rsh**)

Начальной оболочкой называется оболочка, загружаемая при входе в систему. Начальная оболочка задается в файле `/etc/passwd`. Оболочка Korn - это стандартная начальная оболочка системы, совместимая с предыдущими версиями оболочки Bourne.

Оболочка Korn (`/usr/bin/ksh`) установлена по умолчанию. Оболочкой по умолчанию или стандартной оболочкой называется оболочка, соединенная и запущенная командой `/usr/bin/sh`. Оболочка Bourne (`/usr/bin/sh`) может служить оболочкой по умолчанию. Оболочке POSIX соответствует команда `/usr/bin/psh`, которая является символьной связью для команды `/usr/bin/sh`.

Понятия, связанные с данным:

“Оболочка Bourne” на стр. 258
Оболочка Bourne - это интерактивный интерпретатор команд и командный язык программирования.

“Команды оболочки Korn (POSIX)” на стр. 250
Оболочка Korn - это интерактивный интерпретатор и командный язык программирования. Она разработана в соответствии с Международным стандартом интерфейса переносимых операционных систем (POSIX).

Терминология, связанная с оболочками:

Приведенные в данной таблице термины и определения помогут вам лучше познакомиться с оболочками.

Элемент	Описание
blank	Пробел - это один из символов класса пробелов, определенного в категории **LC_CTYPE**. В оболочке POSIX пробел - это символ табуляции или обычный пробел.
встроенная команда	Команда, которую оболочка выполняет без поиска и без создания отдельного процесса.
команда	Последовательность символов, заданная в соответствии с синтаксисом языка оболочки. Оболочка считывает каждую команду и выполняет требуемое действие либо непосредственно, либо путем запуска утилит.
комментарий	Любое слово, начинающееся со знака #. Само слово и все следующие за ним символы до конца строки игнорируются.
идентификатор	Последовательность букв, цифр и знаков подчеркивания из универсального набора символов, начинающаяся с буквы или знака подчеркивания. Идентификатор не должен начинаться с цифры. Идентификаторы применяются в качестве имен псевдонимов, функций и любых именованных параметров.
список	Последовательность из одного или нескольких конвейеров, разделенных одним из следующих символов: точка с запятой (**;**), амперсанд (**&**), двойной амперсанд (**&&**) или двойная вертикальная черта (**||**). Список может оканчиваться на один из следующих символов: точка с запятой (**;**), амперсанд (**&**) или вертикальная черта с амперсантом (**|&**). **;** Задает последовательную обработку предшествующего конвейера. Оболочка поочередно выполняет каждую команду и ожидает завершения последней команды. **&** Задает асинхронную обработку предшествующего конвейера. Оболочка поочередно выполняет каждую команду, обрабатывая конвейер в фоновом режиме и не дожидаясь завершения. **|&** Задает асинхронную обработку предшествующего конвейера и устанавливает двусторонний конвейер с родительской оболочкой. Оболочка поочередно выполняет каждую команду, обрабатывая конвейер в фоновом режиме и не дожидаясь завершения. Родительская оболочка может считывать стандартный вывод и записывать в стандартный ввод созданной команды с помощью команд **read -p** и **print -p**. В каждый момент времени может быть активна только одна такая команда. **&&** Обрабатывает последующий список только в том случае, если предыдущий конвейер возвратил нулевое значение. **||** Обрабатывает последующий список только в том случае, если предыдущий конвейер возвратил ненулевое значение. Точка с запятой (**;**), амперсанд (**&**) и вертикальная черта с амперсантом (**|&**) имеют более низкий приоритет, чем двойной амперсанд (**&&**) и двойная вертикальная черта (**||**). Приоритет символов **;**, **&** и **|&** одинаковый. Приоритет символов **&&** и **||** также одинаковый. Для отделения одной команды от другой в списке вместо точки с запятой можно указать один или несколько символов новой строки. **Примечание:** Символ **|&** допустим только в оболочке Korn.
метасимвол	Метасимвол - это специальный символ оболочки; если метасимвол указан без кавычек, то он считается концом слова. Существуют следующие метасимволы: конвейер (**|**), амперсанд (**&**), точка с запятой (**;**), знак меньше (**<**), знак больше (**>**), открывающая круглая скобка (**(**), закрывающая круглая скобка (**)**), знак доллара (**$**), обратная кавычка (**`**), обратная косая черта (****), одинарная кавычка (**'**), двойная кавычка (**"**), символ новой строки, пробел и символ табуляции. Все символы, заключенные между одинарными кавычками, рассматриваются как обычные, а не специальные символы. Метасимволы сохраняют свое специальное значение, если они не взяты в кавычки. (В оболочке C метасимволы называют также *метасимволами компилятора*.)
список назначения параметров	Содержит одно или несколько слов в формате *Идентификатор=Значение*, в котором количество пробелов перед и после знака равенства (**=**) должно быть одинаковым. Иными словами, либо начальные и конечные пробелы вообще не указываются, либо их должно быть равное число. **Примечание:** В оболочке C список присвоений параметров задается в формате **set***Идентификатор=Значение*. Пробелы перед и после знака равенства (**=**) обязательны.

Элемент	Описание
конвейер	Последовательность из нескольких команд, разделенных вертикальной чертой (=). Каждая команда конвейера, за исключением, быть может, последней, запускается как отдельный процесс. Стандартный вывод предыдущей команды в конвейере становится стандартным вводом следующей. Если список заключен в круглые скобки, то он рассматривается как простая команда, выполняемая в отдельной подоболочке. Если перед конвейером не указан зарезервированный символ `!` , то состоянием выхода будет состояние выхода последней команды конвейера. В противном случае состоянием выхода будет логическое отрицание состояния выхода последней команды. Иными словами, если последняя команда возвратит ноль, то состоянием выхода будет 1, а если положительное значение, то - ноль. Конвейер задается в следующем формате: `[!] command1 [ | command2 ...]` **Примечание:** В предыдущих версиях оболочки Bourne конвейер обозначался знаком ^.
переменная оболочки	Имя или параметр, которому присваивается некоторое значение. Для присвоения значения переменной введите имя переменной, знак равенства (=) и значение. Имя переменной будет заменено на присвоенное ей значение, если перед именем переменной стоит знак доллара ($). Переменные особенно полезны при создании кратких обозначений длинных путей, например, `$HOME` для домашнего каталога. Предопределенной называется переменная, значение которой присваивается оболочкой. Пользовательской - переменная, значение которой присваивается пользователем.
простая команда	Произвольная последовательность списков присвоений параметров и перенаправлений ввода и вывода. За ними могут следовать команды, слова и опции перенаправления. Последовательность должна заканчиваться символом ;, |, &, ||, &&, |& или символом новой строки. Имя команды передается как параметр `0` (согласно определению процедуры **exec**). Значение простой команды - это ее состояние выхода, равное нулю в случае нормального завершения обработки, и положительному значению в противном случае. Функция sigaction, sigvec или signal содержит список значений состояний сигнала выхода.
подоболочка	Оболочка, запущенная как дочерний процесс начальной или текущей оболочки.
символ подстановки	Другое название - *символ шаблона*. Оболочка присваивает символам подстановки некоторые значения. Наиболее часто используются символы подстановки `?`, `*`, `[set]` и `[!set]`. Символы подстановки удобно применять при задании шаблона имен файлов.
слово	Последовательность символов, не содержащая пробелов. Слова разделяются одним или несколькими метасимволами.

Выбор оболочки для файла сценария:

Если исполнимый сценарий был запущен в оболочке Korn (POSIX) или Bourne, то команды этого сценария выполняются под управлением текущей оболочки (оболочки, из которой был запущен сценарий), если не была указана другая оболочка. Если исполняемый сценарий был запущен в оболочке C, команды этого сценария выполняются под управлением оболочки Bourne (`/usr/bin/bsh`), если не была указана другая оболочка.

Для того чтобы сценарий выполнялся в определенной оболочке, укажите эту оболочку в сценарии.

Для этого введите `#!`*Путь* в первой строке сценария оболочки и нажмите Enter. Символы `#!` задают тип файла. Переменная *Путь* задает путь к оболочке, из которой следует запустить сценарий.

Например, для запуска сценария **bsh** в оболочке Bourne введите:

```
#!/usr/bin/bsh
```

Если перед именем файла сценария будет задана команда оболочки, то оболочка, указанная в командной строке, переопределит оболочку, указанную в файле сценария. Например, если будет вызвана команда `ksh myfile`, то сценарий `myfile` будет выполняться под управлением оболочки Korn, даже если в первой строке файла `myfile` указано `#!/usr/bin/csh`.

Функции оболочки:

В использовании оболочки в качестве системного интерфейса есть определенные преимущества.

Ниже перечислены основные возможности и функции оболочки как интерфейса между пользователем и системой:

- **Подстановка символов в именах файлов (шаблоны)**

  Команды над группой файлов выполняются по единому шаблону, а не по фактическим именам файлов.

  Дополнительная информация приведена в следующих разделах:
  - “Подстановка имен файлов в оболочке Korn (POSIX)” на стр. 229
  - “Подстановка имен файлов в оболочке Bourne” на стр. 261
  - “Подстановка имен файлов в оболочке C” на стр. 277
- **Обработка в фоновом режиме**

  Сложные задачи рекомендуется выполнять в фоновом режиме, освобождая терминал для параллельной интерактивной обработки.

  Дополнительная информация приведена в описании команды **bg** в следующих разделах:
  - “Управление заданиями в оболочке Korn (POSIX)” на стр. 243
  - “Встроенные команды оболочки C” на стр. 284

  Примечание: В оболочке Bourne управление заданиями не поддерживается.
- **Псевдонимы команд**

  Команде или словосочетанию можно присвоить псевдоним. Когда оболочка обнаруживает псевдоним в командной строке или в сценарии оболочки, она подставляет текст, соответствующий псевдониму.

  Дополнительная информация приведена в следующих разделах:
  - “Псевдонимы команд в оболочке Korn (POSIX)” на стр. 256
  - “Подстановка псевдонимов в оболочке C” на стр. 274

  Примечание: В оболочке Bourne псевдонимы команд не поддерживаются.
- **Хронология команд**

  Записывает вводимые команды в файл хронологии. С помощью этого файла вы легко можете обращаться к ранее введенным командам, изменять и повторять их.

  Дополнительная информация приведена в описании команды **history** в следующих разделах:
  - “Хронология команд оболочки Korn (POSIX)” на стр. 255
  - “Встроенные команды оболочки C” на стр. 284
  - “Подстановка хронологии в оболочке C” на стр. 291

  Примечание: В оболочке Bourne хронологический список команд не ведется.
- **Подстановка имен файлов**

  Автоматически создает список имен файлов в командной строке согласно шаблону.

  Дополнительная информация приведена в следующих разделах:
  - “Подстановка имен файлов в оболочке Korn (POSIX)” на стр. 229
  - “Подстановка имен файлов в оболочке Bourne” на стр. 261
  - “Подстановка имен файлов в оболочке C” на стр. 277
- **Перенаправление ввода и вывода**

  Перенаправляет ввод с клавиатуры или перенаправляет вывод в файл или устройство, отличное от терминала. Например, входные данные для программы могут поступать из файла и перенаправляться на принтер или в другой файл.

  Дополнительная информация приведена в следующих разделах:

- “Перенаправление ввода и вывода в оболочке Korn (POSIX)” на стр. 230
- “Перенаправление ввода и вывода в оболочке Bourne” на стр. 261
- “Перенаправление ввода и вывода в оболочке C” на стр. 294

- **Конвейер**

  Объединяет последовательность команд в сложную программу. Стандартный вывод предыдущей команды становится стандартным вводом следующей.

  Дополнительная информация приведена в определении *конвейера* в книге “Терминология, связанная с оболочками” на стр. 208.

- **Подстановка переменных оболочки**

  Сохраняет данные в пользовательских и предопределенных переменных оболочки.

  Дополнительная информация приведена в следующих разделах:

  - “Подстановка параметров в оболочке Korn и POSIX” на стр. 227
  - “Подстановка переменных в оболочке Bourne” на стр. 270
  - “Подстановка переменных в оболочке C” на стр. 275

Понятия, связанные с данным:

“Команды” на стр. 127
Для запуска некоторых команд достаточно ввести одно слово. Кроме того, команды можно комбинировать, чтобы вывод одной команды становился входными данными для другой.

Классы символов:

Для указания диапазонов символов можно применять классы.

При создании шаблонов имен файлов можно применять классы символов:

`[[:`*класс_символов*`:]]`

Данный формат обозначает любой символ, принадлежащий указанному классу. Набор классов соответствует функциям **ctype**, как указано ниже:

Класс символов	Определение
alnum	Алфавитно-цифровые символы
alpha	Строчные и прописные буквы
blank	Пробел или символ горизонтальной табуляции
cntrl	Управляющие символы
digit	Цифры
graph	Графические символы
lower	Строчные буквы
print	Печатаемые символы
punct	Знаки препинания
space	Пробел, символ горизонтальной табуляции, символ возврата каретки, символ новой строки, символ вертикальной табуляции или символ перевода страницы
upper	Прописные буквы
xdigit	Шестнадцатеричные цифры

оболочка с ограничениями:

Оболочка с ограничениями позволяет задать имена пользователей и сред, возможности которых будут ограничены по сравнению с обычной оболочкой Bourne.

Команда **Rsh** или **bsh -r** открывает ограничивающую оболочку. Такая оболочка поддерживает все команды оболочки **bsh**, за исключением следующих:

- Переход в другой каталог (команда **cd**)
- Настройка значений переменных *PATH* и *SHELL*

- Ввод имен и команд, содержащих косую черту (/)
- Перенаправление вывода

Если оболочка с ограничениями определяет, что запускаемая команда - это процедура оболочки, то команда запускается в оболочке Bourne. Таким образом, пользователю можно предоставить процедуры оболочки, эквивалентной оболочке Bourne, но с ограниченным набором команд. В этом случае предполагается, что пользователю не предоставлены права на запись и выполнение для текущего каталога.

Если в команде запуска оболочки Bourne указан параметр *Файл* [*Опции*], то оболочка запустит файл сценария, указанный в параметре *Файл*, с учетом опций. Файл сценария должен быть доступен для чтения. Все значения **setuid** и **setgid**. Затем оболочка выполнит команды. Файл сценария не следует указывать вместе с флагами **-c** и **-s**.

Если для запуска оболочки применяется команда **Rsh**, то ограничения вступают в силу после считывания файлов `.profile` и `/etc/environment`. Таким образом, владелец файла `.profile` полностью контролирует действия пользователя, выполняя настройку и задавая каталог для пользователя (возможно, отличный от начального). Администратор может создать в `/usr/rbin` каталог команд, доступных команде **Rsh** , указав этот каталог в переменной *PATH*. Если для запуска оболочки применяется команда **bsh -r**, то ограничения вступают в силу после считывания файла `.profile`.

Если оболочка с ограничениями была вызвана как **Rsh**, она считывает пользовательский файл `.profile` (`$HOME/.profile`). Такая оболочка полностью эквивалентна обычной оболочке Bourne, за исключением того, что прерывание приводит к немедленному выходу из оболочки, а не возврату в режим командной строки.

Оболочка Korn может быть запущена как оболочка с ограничениями с помощью команды **ksh -r**.

inodes для **ksh** и **rksh** идентичны, и inodes для **ksh93** и **rksh93** идентичны.

Создание и выполнение сценария оболочки:

Сценарий оболочки - это файл, содержащий одну или несколько команд. Сценарии оболочки позволяют достаточно просто выполнять однообразные команды, длинные или сложные последовательности команд и процедуры. Когда вы вводите имя файла сценария оболочки, система выполняет содержащуюся в нем последовательность команд.

Сценарий оболочки можно создать в обычном текстовом редакторе. В сценарий можно включить как команды операционной системы, так и встроенные команды оболочки.

Ниже приведены основные рекомендации по созданию сценариев оболочки:

1. Создайте и сохраните файл в текстовом редакторе. В файл можно включить любое сочетание команд оболочки и операционной системы. По умолчанию сценарии оболочки, не предназначенные для применения всеми пользователями, хранятся в каталоге `$HOME/bin`.

   Примечание: Операционная система не поддерживает функции **setuid** и **setgid** в сценарии оболочки.
2. Если вы хотите запретить запускать сценарий всем пользователям, кроме его владельца, воспользуйтесь командой **chmod**. Например, если файл называется `script1`, введите:

   ```
   chmod u=rwx script1
   ```
3. Для запуска сценария введите его имя в командной строке. Например, для запуска сценария оболочки `script1` введите:

   ```
   script1
   ```

   Примечание: Вы можете запустить сценарий оболочки, не делая его исполняемым, если укажете команду оболочки (**ksh**, **bsh** или **csh**) перед именем файла сценария в командной строке. Например, для запуска неисполняемого файла `script1` в оболочке Korn введите:

   ```
   ksh script1
   ```

Понятия, связанные с данным:

“Команды” на стр. 127
Для запуска некоторых команд достаточно ввести одно слово. Кроме того, команды можно комбинировать, чтобы вывод одной команды становился входными данными для другой.

Оболочка Korn

Оболочка Korn (команда ksh) совместима с предыдущими версиями оболочки Bourne (команда bsh) и содержит большинство функций оболочки Bourne, а также несколько наилучших функций оболочки C.

Переменные, задаваемые оболочкой Korn или POSIX:

Ниже описаны переменные, задаваемые оболочкой.

Элемент	Описание
знак подчеркивания (_)	Первоначально равна абсолютному пути к исполняемому файлу оболочки или сценария. Затем этой переменной присваивается значение последнего аргумента предыдущей команды. Этот параметр не определен для команд, выполняемых в асинхронном режиме. Кроме того, параметру - присваивается имя файла `MAIL` при проверке наличия новой почты.
ERRNO	Содержит код ошибки, выданный последней подпрограммой, завершившейся с ошибкой. Это значение зависит от конкретной системы и предназначено только для отладки.
LINENO	Содержит номер текущей строки в выполняемом сценарии или функции.
OLDPWD	Содержит имя предыдущего рабочего каталога, установленное с помощью команды **cd**.
OPTARG	Содержит значение последнего аргумента опции, обработанного обычной встроенной командой **getopts**.
OPTIND	Содержит номер последнего аргумента опции, обработанного обычной встроенной командой **getopts**.
PPID	Содержит номер родительского процесса оболочки.
PWD	Содержит имя текущего рабочего каталога, установленного с помощью команды **cd**.
RANDOM	Содержит случайное целое число, равномерно распределенное в интервале от 0 до 32767. Последовательность псевдослучайных чисел можно инициализировать, присвоив произвольное число переменной *RANDOM*.
REPLY	Значение этой переменной присваивается оператором **select**, а также обычной встроенной командой **read**, если она вызвана без аргументов.
SECONDS	Содержит число секунд, прошедшее с момента запуска оболочки. Если этой переменной будет присвоено какое-либо число, то в результате она будет содержать сумму числа секунд, прошедшего с момента запуска оболочки, и присвоенного ей числа.

Переменные, используемые оболочкой Korn или POSIX:

Ниже описаны переменные, используемые оболочкой.

Элемент	Описание
CDPATH	Задает список каталогов, в которых выполняется поиск нужного каталога при выполнении команды **cd**.
COLUMNS	Задает ширину окна редактирования в режиме редактирования и при выдаче списков оператора **select**.
EDITOR	Если значение этого параметра оканчивается на символы emacs, gmacs или vi и в данном сеансе оболочки переменная *VISUAL* не была определена с помощью особой встроенной команды **set**, то оболочка включает соответствующий встроенный редактор для командной строки.
ENV	Если эта переменная определена, то при запуске оболочка пытается выполнить указанный файл. Данный файл выполняется после файла $HOME/.profile и, как правило, применяется для определения функций и псевдонимов. Для неинтерактивных оболочек этот флаг игнорируется.
FCEDIT	Задает имя редактора по умолчанию для встроенной команды **fc**.

Элемент	Описание
FPATH	Задает список каталогов для поиска определений функций. В случае, если какая-либо функция будет вызвана с флагом **-u**, и не будет найдена соответствующая команда, оболочка попытается найти определение функции в этих каталогах. Если будет найден выполняемый файл, он будет прочитан и выполнен в текущей среде.
HISTFILE	Если эта переменная определена на момент запуска оболочки, она задает имя файла хронологии выполненных команд. Процесс инициализации файла `history` зависит от содержимого файлов запуска системы, поскольку в некоторых из этих файлов могут находиться переопределения переменных HISTFILE и HISTSIZE. Например, в файле `history` сохраняются команды определения функций. Если администратор системы включит определения функций в системный файл запуска, вызываемый до файла ENV или до определения переменных *HISTFILE* или *HISTSIZE*, файл `history` будет инициализирован до того, как пользователь сможет повлиять на его характеристики.
HISTSIZE	Если эта переменная определена на момент запуска оболочки, она задает число команд, сохраняемых в файле хронологии. Значение по умолчанию - 512 команд для пользователя root и 128 команд для других пользователей.
HOME	Содержит имя каталога, который становится текущим после входа в систему. Эта переменная инициализируется программой **login**. Значение переменной *$HOME* используется командой **cd** , если она указана без аргументов. Применение этой переменной вместо абсолютных имен позволяет каждому пользователю без изменений выполнять процедуру в своей среде из различных каталогов.
IFS	Задает IFS (внутренние разделители полей), которыми обычно служат пробел, символ табуляции и символ новой строки. Эти символы применяются для разделения слов, получаемых в результате подстановки параметров или команд, и для разделения слов при выполнении обычной встроенной команды **read**. Первый символ параметра *IFS* вставляется между именами, полученными в результате подстановки параметра `$*`.
LANG	Содержит значение по умолчанию для переменных *LC_**.
LC_ALL	Переопределяет значение переменной *LANG* и переменных *LC_**.
LC_COLLATE	Задает режим проверки принадлежности значений диапазону при подстановке шаблонов.
LC_CTYPE	Задает классификацию символов, соответствие между строчными и прописными буквами и прочие атрибуты символов.
LC_MESSAGES	Задает язык, на котором выдаются сообщения.
LINES	Задает высоту колонок при печати списков оператора select. Высота списков оператора select (в строках) составляет около двух третей значения *LINES*.
MAIL	Задает имя файла, применяемого для проверки наличия новой почты. Если этой переменной будет присвоено имя файла почты, а переменная *MAILPATH* не будет определена, то оболочка будет проверять наличие новой почты в указанном файле.
MAILCHECK	Задает интервал (в секундах), через который оболочка будет проверять, изменились ли файлы, имена которых присвоены переменным *MAILPATH* и *MAIL*. Значение по умолчанию - 600 секунд. По истечении указанного времени с момента последней проверки оболочка вновь проверяет файлы почты перед тем, как выдать приглашение.
MAILPATH	Задает список имен файлов, разделенных двоеточиями. Если эта переменная определена, оболочка информирует пользователя о любых изменениях в указанных файлах. Файлы проверяются с интервалом, заданным переменной *MAILCHECK*. За каждым именем файла может быть указан символ **?** и сообщение для печати. Для данного сообщения будет выполнена подстановка переменных, и на момент подстановки имя измененного файла будет присвоено переменной *$_*. По умолчанию выдается сообщение `Вам пришла новая почта в $_`.
NLSPATH	Задает расположение каталогов с сообщениями. Эта переменная применяется вместе с переменной **LC_MESSAGES**.
PATH	Содержит список каталогов, в которых выполняется поиск команд. Каталоги должны быть разделены двоеточиями. При поиске команд каталоги просматриваются в указанном порядке. Пустая строка (два двоеточия подряд, начальное или конечное двоеточие) в этом списке соответствует текущему каталогу.
PS1	Задает основное приглашение системы. Это приглашение применяется в качестве основного приглашения оболочки, значение по умолчанию - `$`. Символ `!` в основном приглашении оболочки заменяется на номер текущей команды.
PS2	Задает вспомогательное приглашение системы, по умолчанию - символ `>`.

Элемент	Описание
PS3	Задает приглашение выбора, применяемое в цикле **select**. Значение по умолчанию - #? .
PS4	Значение этой переменной после подстановки параметров добавляется в начало каждой строки трассировочной информации. Если она не определена, по умолчанию в качестве приглашения трассировки используется символ +.
SHELL	Задает путь к программе оболочки.
SHELL PROMPT	В интерактивном режиме перед чтением очередной команды оболочка выдает в качестве приглашения значение параметра *PS1*. Если после ввода символа новой строки оболочке потребуется дополнительная входная информация для выполнения команды, то она выдаст дополнительное приглашение - значение параметра *PS2*.
TMOUT	Задает максимальное время (в секундах), в течение которого оболочка может бездействовать. Если переменной *TMOUT* присвоено положительное значение, то оболочка автоматически завершает работу в случае, если в течение указанного времени в ответ на приглашение *PS1* не будет введена ни одна команда. (Учтите, что в оболочке может быть установлено жесткое ограничение, которое нельзя превысить путем увеличения этого значения.) **Примечание:** По истечении указанного времени оболочка дополнительно ожидает ввода команды в течение 60 секунд.
VISUAL	Если значение этой переменной заканчивается на символы emacs, gmacs или vi, то включается соответствующая опция редактирования.

В оболочке предусмотрены значения по умолчанию для параметров *PATH*, *PS1*, *PS2*, *MAILCHECK*, *TMOUT* и *IFS*. Параметры *HOME*, *SHELL*, *ENV* и *MAIL не* задаются оболочкой (однако параметр *HOME* задается командой **login**).

Подстановка команд в оболочке Korn и POSIX:

В оболочке Korn (другое название - POSIX) можно выполнять подстановку команд. При подстановке оболочка выполняет указанную команду в среде подоболочки и заменяет команду на ее вывод.

Для подстановки команд в оболочке Korn или POSIX введите:

```
$(команда)
```

или воспользуйтесь обратными кавычками:

```
`команда`
```

Примечание: Хотя обратные кавычки распознаются оболочкой **ksh**, в стандарте X/Open Portability Guide Issue 4 и POSIX они не поддерживаются. В этих стандартах рекомендуется применять в приложениях формат `$(команда)`.

Оболочка подставляет команду, выполняя `команду` в среде подоболочки и заменяя ее (текст `команды` и окружающие его символы `$( )` или обратные кавычки) на полученный стандартный вывод, удаляя при этом все символы новой строки в конце вывода.

В следующем примере символы `$( )` , в которые заключена команда, указывают, что выполняется подстановка для команды **whoami**:

```
echo Меня зовут: $(whoami)
```

Ту же подстановку можно выполнить и по-другому:

```
echo Меня зовут: `whoami`
```

В обоих случаях вывод для пользователя `dee` будет следующим:

```
Меня зовут: dee
```

Можно подставлять и арифметические выражения, заключая их в круглые скобки `( )`. Например, команда:

```
echo В одном часе $((60 * 60)) секунд
```

выдаст следующий результат:

```
В одном часе 3600 секунд
```

При подстановке команд оболочка Korn (POSIX) удаляет все конечные символы новой строки. Например, если текущий каталог содержит файлы **файл1**, **файл2** и **файл3**, то команда:

```
echo $(ls)
```

удалит символы новой строки и создаст следующий вывод:

```
файл1 файл2 файл3
```

Если вы хотите сохранить символы новой строки, укажите команду в двойных кавычках (" "):

```
echo "$(ls)"
```

Арифметические операции в оболочке Korn (POSIX):

Стандартная встроенная команда **let** оболочки Korn (POSIX) позволяет выполнять арифметические операции над целыми числами.

Константы задаются в формате [**Основание**]**Число**. Параметр **Основание** представляет собой десятичное число в диапазоне от 2 до 36 включительно, указывающее основание системы счисления. В параметре **Число** указывается число в этой системе. Если параметр **Основание** не указан, принимается значение 10.

Синтаксис, приоритет операторов и ассоциативность операторов в арифметических выражениях подчиняются правилам языка программирования C. Поддерживаются все операторы, кроме двойного плюса (++), двойного дефиса (--), вопросительного знака с двоеточием (**?:**) и запятой (**,**). В следующей таблице перечислены операторы оболочки Korn (POSIX) в порядке убывания приоритета:

Оператор	Определение
-	Унарный минус
!	Логическое отрицание
~	Поразрядное отрицание
*	Умножение
/	Деление
%	Остаток
+	Сложение
-	Вычитание
<<, >>	Сдвиг влево, сдвиг вправо
<=,>=, <>, ==, !=	Сравнение
&	Поразрядное И
^	Поразрядное исключающее ИЛИ
|	Поразрядное ИЛИ
&&	Логическое И
||	Логическое ИЛИ
= *=, /=, &= +=, -=, <<=, > >=, &=, ^=, |=	Присвоение

Многие арифметические операторы, такие как *****, **&**, **<** и **>**, в оболочке Korn (POSIX) интерпретируются специальным образом. Такие символы должны быть заключены в кавычки. Например, для умножения текущего значения переменной *y* на 5 и присвоения нового значения переменной *y* укажите выражение:

```
let "y = y * 5"
```

Заключение выражения в кавычки отменяет специальное значение символа *.

Операции в команде **let** можно объединять в группы скобками. Например, в выражении:

```
let "z = q * (z - 10)"
```

переменная *q* будет умножено на уменьшенное на 10 значение переменной *z*.

Если необходимо вычислить только одно выражение, оболочка Korn (или POSIX) допускает альтернативную форму команды **let**. Оболочка рассматривает команды, заключенные в скобки `(( ))`, как выражения в кавычках. Таким образом, выражение

```
((x = x / 3))
```

эквивалентно

```
let "x = x / 3"
```

На именованные параметры внутри арифметических выражений можно ссылаться просто по их имени, без какого-либо синтаксиса подстановки параметра. При этом значение параметра вычисляется как арифметическое выражение.

Внутреннее числовое представление именованного параметра можно указать с помощью флага **-i** специальной встроенной команды **typeset**. С флагом **-i** арифметические вычисления выполняются над каждым значением, присваиваемым именованному параметру. Если основание системы счисления не указано, оно определяется по первому присвоенному значению. Это основание применяется только при подстановке параметра.

Понятия, связанные с данным:

“Команды оболочки Korn (POSIX)” на стр. 250
Оболочка Korn - это интерактивный интерпретатор и командный язык программирования. Она разработана в соответствии с Международным стандартом интерфейса переносимых операционных систем (POSIX).

“Параметры в оболочке Korn” на стр. 226
Ниже описаны параметры оболочки Korn.

Разделение полей в оболочке Korn (POSIX):

После подстановки команд оболочка Korn ищет в результирующей строке символы разделения полей, указанные в переменной **IFS** (Разделители внутренних полей). В тех местах, где оболочка находит такие символы, она разделяет строку между различными аргументами.

Затем она оставляет явные пустые аргументы (`""` и `''`) и отбрасывает неявные пустые аргументы (полученные от параметров, у которых нет значений).

- Если в переменной *IFS* указаны пробел, знак табуляции и символ новой строки, либо если эта переменная не задана, то любые пробелы, знаки табуляции и символы новой строки в начале и в конце ввода будут игнорироваться, а в середине ввода будет восприниматься как разделители полей. Например, следующая строка будет разделена на два поля, **school** и **days**:

  ```
  <newline><space><tab>school<tab><tab>days<space>
  ```

- Если значение переменной **IFS** непустое и отлично от вышеперечисленных, последовательно действуют следующие правила. *Пробелом **IFS*** называется любая (в том числе пустая) последовательность символов, каждый из которых содержится в переменной **IFS** (например, если **IFS** содержит пробел, запятую и знак табуляции, то любая последовательность пробелов и знаков табуляции рассматривается как пробел **IFS**).
  1. Пробелы **IFS** в начале и в конце строки игнорируются.
  2. Любой символ **IFS**, отличный от пробела **IFS**, вместе со смежными пробелами **IFS** рассматривается как разделитель поля.
  3. Пробелы **IFS** ненулевой длины разделяют поля.

Список особых встроенных команд оболочки Korn или POSIX:

В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

Элемент	Описание
:(двоеточие)	Только интерпретирует аргументы.
. (точка)	Считывает указанный файл, а затем выполняет содержащиеся в нем команды.
break	Прерывает цикл команды **for**, **while**, **until** или **select**, если он выполняется.
continue	Возобновляет следующий внешний цикл команды **for** , **while**, **until** или **select**.
eval	Выполняет команду или команды, указанные в качестве параметров.
exec	Запускает команду, указанную в параметре *Аргумент*, не создавая нового процесса (команда заменяет оболочку).
exit	Завершает работу оболочки с кодом возврата, заданным в параметре *n*.
export	Заносит указанные имена в список имен, которые автоматически экспортируются в среду выполнения следующих команд.
newgrp	Аналогична команде `exec/usr/bin/newgrp` [*группа ...*].
readonly	Для указанных файлов устанавливаются права доступа только на чтение.
return	Возвращает управление из текущей функции в сценарий, из которого она была вызвана.
set	Если команда выполняется без параметров, то она выводит список всех переменных оболочки и соответствующие им значения в алфавитном порядке, определяемом текущей локалью.
shift	Предназначена для переименования позиционных переменных.
times	Показывает общее время, которое заняло выполнение пользовательских и системных процессов в оболочке.
trap	Выполняет указанную команду, когда оболочка получает один из перечисленных сигналов.
typeset	Позволяет задавать атрибуты и значения параметров оболочки.
unset	Удаляет значения и атрибуты указанных параметров.

Понятия, связанные с данным:

“Встроенные команды оболочки Korn или POSIX” на стр. 233
В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

Обычные встроенные команды оболочки Korn или POSIX:

Ниже приведен список обычных встроенных команд оболочки Korn или POSIX.

Элемент	Описание
alias	Записывает в стандартный вывод список псевдонимов.
bg	Переводит указанные задания в фоновый режим.
cd	Предназначена для перехода из текущего каталога в указанный каталог или замены текущей строки указанной строкой.
echo	Записывает строки символов в устройство стандартного вывода.
fc	Выбирает интервал команд из списка хронологии, размер которого равен значению переменной *HISTSIZE*. После подстановки новых значений вместо старых выбранные команды последовательно выполняются.
fg	Переводит указанный процесс в фоновый режим.
getopts	Проверяет правильность опций в параметре *аргумент*.
jobs	Предназначена для просмотра информации об указанных заданиях.
kill	Отправляет сигнал **TERM** (сигнал завершения) указанным заданиям и процессам.
let	Вычисляет указанное арифметическое выражение.
print	Направляет указанный аргумент в стандартный вывод оболочки.
pwd	Аналогична команде **print -r -$PWD**.
read	Считывает входную информацию оболочки.
ulimit	Задает или показывает ограничения на ресурсы пользовательских процессов из файла `/etc/security/limits`.
umask	Определяет права доступа к файлу.
unalias	Удаляет имена из списка псевдонимов.
wait	Ожидает завершения указанного задания и завершается.
whence	Показывает, как будет интерпретироваться слово, если оно будет задано в качестве имени команды.

За дополнительной информацией обратитесь к разделу “Встроенные команды оболочки Korn или POSIX” на стр. 233.

Понятия, связанные с данным:

“Встроенные команды оболочки Korn или POSIX” на стр. 233
В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

Условные выражения в оболочке Korn (POSIX):

Условное выражение применяется в составной команде [[(две квадратные скобки) для проверки атрибутов файла и сравнения строк.

В словах, расположенных между символами [[и]], не выполняются разбиение слов и подстановка имен файлов. Каждое выражение образуется из одного или нескольких унарных или бинарных операторов следующего вида:

Элемент	Описание
-a *файл*	Истинно, если указанный файл - это символьная ссылка на существующий файл.
-b *файл*	Истинно, если указанный файл существует и является блочным специальным файлом.
-c *файл*	Истинно, если указанный файл существует и является символьным специальным файлом.
-d *файл*	Истинно, если указанный файл существует и является каталогом.
-e *файл*	Истинно, если указанный файл существует.
-f *файл*	Истинно, если указанный файл существует и является обычным файлом.
-g *файл*	Истинно, если указанный файл существует и задан его бит **setgid**.
-h *файл*	Истинно, если указанный файл существует и является символьной ссылкой.
-k *файл*	Истинно, если указанный файл существует и задан его бит привязки.
-n *строка*	Истинно, если длина указанной строки ненулевая.
-o *Опция*	Истинно, если указанная опция включена.
-p *файл*	Истинно, если указанный файл существует и является специальным файлом FIFO или конвейером.
-r *файл*	Истинно, если указанный файл существует и может быть прочитан текущим процессом.
-s *файл*	Истинно, если указанный файл существует и не пуст.
-t *дескриптор_файла*	Истинно, если файл с указанным дескриптором открыт и связан с терминалом.
-u *файл*	Истинно, если указанный файл существует и задан его бит **setuid**.
-w *файл*	Истинно, если указанный файл существует и его бит записи равен 1. Однако файл может быть недоступен для записи в файловой системе, в которой разрешено только чтение, даже если результатом проверки будет истина.
-x *файл*	Истинно, если указанный файл существует и флаг **execute** установлен. Если указанный файл существует и является каталогом, то текущему процессу будет разрешено выполнять в нем поиск.
-z *строка*	Истинно, если длина указанной строки нулевая.
-L *файл*	Истинно, если указанный файл существует и является символьной ссылкой.
-O *файл*	Истинно, если указанный файл существует и принадлежит фактическому ИД пользователя этого процесса.
-G *файл*	Истинно, если указанный файл существует и его группа соответствует фактическому ИД пользователя этого процесса.
-S *файл*	Истинно, если указанный файл существует и является сокетом.
файл1 **-nt** *файл2*	Истинно, если *Файл1* существует и создан позже, чем *Файл2*.
файл1 **-ot** *файл2*	Истинно, если *Файл1* существует и создан раньше, чем *Файл2*.
файл1 **-ef** *файл2*	Истинно, если *Файл1* и *Файл2* существуют и обозначают один и тот же файл.
строка1 **=** *строка2*	Истинно, если *Строка1* и *Строка2* совпадают.
строка1 **!=** *строка2*	Истинно, если *Строка1* и *Строка2* не совпадают.
строка **=** *шаблон*	Истинно, если указанная строка соответствует указанному шаблону.
строка **!=** *шаблон*	Истинно, если указанная строка не соответствует указанному шаблону.
строка1 **<** *строка2*	Истинно, если *Строка1* предшествует *Строке2*, согласно ASCII-кодам символов.
строка1 **>** *строка2*	Истинно, если *Строка1* следует за *Строкой2*, согласно ASCII-кодам символов.
выражение1 **-eq** *выражение2*	Истинно, если *Выражение1* равно *Выражению2*.
выражение1 **-ne** *выражение2*	Истинно, если *Выражение1* не равно *Выражению2*.
выражение1 **-lt** *выражение2*	Истинно, если *Выражение1* меньше *Выражения2*.
выражение1 **-gt** *выражение2*	Истинно, если *Выражение1* больше *Выражения2*.
выражение1 **-le** *выражение2*	Истинно, если *Выражение1* меньше или равно *Выражению2*.

Элемент	Описание
выражение1 **-ge** *выражение2*	Истинно, если *Выражение1* больше или равно *Выражению2*.

Примечание: В каждом из предыдущих выражений, если переменная *Файл* схожа с `/dev/fd/n`, где *n* - целое число, то проверка выполняется по отношению к файлу с дескриптором *n*.

Вы можете сформировать составное выражение из этих базовых конструкций, или более мелких элементов, с помощью любых выражений, перечисленных ниже в порядке убывания приоритета:

Элемент	Описание
(*Выражение***)**	Истинно, если указанное выражение истинно. Служит для объединения выражений в группы.
! *Выражение*	Истинно, если указанное выражение ложно.
Выражение1 **&&** *Выражение2*	Истинно, если истинны и *Выражение1*, и *Выражение2*.
Выражение1 **||** *Выражение2*	Истинно, если истинно *Выражение1* или *Выражение2*.

Заключение символов в кавычки в оболочке Korn (POSIX):

Если вы хотите, чтобы оболочка Korn (POSIX) обработала специальный символ как обычный, *заключите этот символ в кавычки*.

Каждый метасимвол имеет специальное значение для оболочки; если метасимвол указан без кавычек, то он обозначает конец слова. Ниже перечислены метасимволы (т.е. специальные символы) оболочки Korn (POSIX):

- конвейер (`|`)
- амперсанд (`&`)
- точка с запятой (`;`)
- знаки меньше (`<`) и больше (`>`)
- открывающая (`(`) и закрывающая (`)`) круглые скобки
- знак доллара (`$`)
- обратные (`` ` ``) и обычные (`'`) одинарные кавычки
- обратная косая черта (`\`)
- двойные кавычки (`"`)
- символ новой строки
- пробел
- символ табуляции

Ниже описаны приемы, позволяющие отменить специальное значение метасимволов.

Элемент	Описание
Обратная косая черта	Обратная косая черта (\), не заключенная в кавычки, сохраняет обычное значение следующего за ней символа, за исключением символа новой строки. Если перед символом новой строки указана обратная косая черта, он интерпретируется оболочкой как символ продолжения строки.
Одинарные кавычки	Символы, заключенные в одинарные кавычки (`' '`), теряют специальное значение. Одинарная кавычка не может быть заключена в одинарные кавычки. Если необходимо указать одинарную кавычку внутри одинарных кавычек, то одной обратной косой черты для этого недостаточно. Необходимо ввести следующую последовательность символов: `'a'\''b'` (результатом будет `a'b`).

Элемент	Описание	
Двойные кавычки	Символы, заключенные в двойные кавычки (" "), теряют специальное значение, за исключением следующих символов: знак доллара, обратная кавычка, обратная косая черта.	
	$	Знак доллара сохраняет свое специальное значение - подстановка параметров, команд и арифметических выражений. На символы, заключенные между `$(` и `)` в строке, взятой в двойные кавычки, двойные кавычки не действуют. Эти символы определяют команду, вывод которой заменит `$(...)` после подстановки команды. Внутри строки символов, находящихся между ${ и $}, должно быть четное число двойных и одинарных кавычек, не являющихся Escape-последовательностями. Для отмены специального значения символа { или } перед ним необходимо поставить обратную косую черту.
	`	Обратная кавычка сохраняет свое специальное значение - другая форма подстановки команд. Символы от начальной обратной кавычки до следующей обратной кавычки, перед которыми нет обратной косой черты, определяют команду, вывод которой заменит ` ... ` после подстановки команды.
	\	Косая черта сохраняет свое специальное значение (Escape-символ), только если за ней следует один из символов $, `, ", \ или символ новой строки.

Если внутри двойных кавычек необходимо указать двойную кавычку, то перед ней следует поставить обратную косую черту. Если строка заключена в двойные кавычки, то в случае, когда за обратной косой чертой следует специальный символ, обратная косая черта будет удалена, а символ потеряет специальное значение. Если же обратная косая черта перед специальным символом отсутствует, то символ останется без изменений, так же как и непосредственно следующий за ним. Например:

```
"\$"    ->      $
"\a"    ->      \a
```

Следующие правила относятся к метасимволам и символам кавычек в оболочке Korn (POSIX):

- Сочетания знака доллара со звездочкой ($*) и знака доллара с символом at ($@) эквивалентны в качестве значения параметра или имени файла, если они не заключены в кавычки.
- Если символы двойная кавычка, знак доллара, звездочка, двойная кавычка ("$*") указаны в качестве аргумента команды, то они эквивалентны символам "$*d*$2*d*...", где *d* - первый символ параметра IFS.
- Последовательность из двойной кавычки, символа at, звездочки и двойной кавычки ("$@") равносильна "$1" "$2" ....
- Внутри обратных кавычек (``) косая черта заключает в кавычки символы обратная косая черта (\), одинарная кавычка (') и знак доллара ($). Если обратные кавычки стоят внутри двойных кавычек (" "), то обратная косая черта заключает в кавычки также символы двойных кавычек.
- Внутри двойных кавычек (" ") выполняется подстановка команд и параметров.
- Специальное значение зарезервированных слов и псевдонимов аннулируется, если любой символ этого слова заключен в кавычки. Заключать в кавычки имена функций и встроенные команды нельзя.

Ограниченная оболочка Korn:

Оболочка Korn с применяется для пользователей и сред, права которых должны быть ограничены по сравнению с обычной оболочкой Korn.

Команда **rksh** or **ksh -r** открывает ограничивающую оболочку Korn. Такая оболочка поддерживает все команды оболочки **ksh**, за исключением следующих:

- Смена текущего рабочего каталога
- Изменение значений переменных *SHELL*, *ENV* или *PATH*.
- Применение символа / (косая черта) в путях

- Перенаправление вывода команд с помощью символов >, >|, <> или >>.

Если оболочка с ограничениями определяет, что запускаемая команда - это процедура оболочки, то команда запускается в оболочке Korn. Таким образом, конечному пользователю можно предоставить процедуры оболочки, эквивалентной оболочке Korn, но с ограниченным набором команд. В этом случае предполагается, что пользователю не предоставлены права на запись и выполнение для текущего каталога.

Если в команде запуска оболочки Korn указан параметр **Файл** [*Параметр*], то оболочка запустит файл сценария, указанный в параметре **Файл**, с учетом опций. Файл сценария должен быть доступен для чтения. Все значения **setuid** и **setgid**. Затем оболочка выполнит команды. Файл сценария не следует указывать вместе с флагами **-c** и **-s**.

Если для запуска оболочки применяется команда **rksh**, то ограничения вступают в силу после считывания файлов `.profile` и `/etc/environment`. Таким образом, владелец файла `.profile` полностью контролирует действия пользователя, выполняя настройку и задавая каталог для пользователя (возможно, отличный от начального). Администратор может создать в `/usr/rbin` каталог команд, доступных команде **rksh** , указав этот каталог в переменной *PATH*. Если для запуска оболочки применяется команда **ksh -r**, то ограничения вступают в силу после считывания файла `.profile`.

Если оболочка с ограничениями Korn была вызвана как **rksh**, она считывает пользовательский файл `.profile` (`$HOME/.profile`). Такая оболочка полностью эквивалентна обычной оболочке Korn, за исключением того, что прерывание приводит к немедленному выходу из оболочки, а не возврату в режим командной строки.

Зарезервированные слова в оболочках Korn или POSIX:

Следующие слова зарезервированы оболочкой Korn или POSIX и имеют для нее специальное значение.

```
!         case    do
done      elif    else
esac      fi      for
function  if      in
select    then    time
until     while   {
}         [[      ]]
```

Зарезервированные слова распознаются только при условии, что они указаны без кавычек в качестве:

- Первого слова команды
- Первого слова, стоящего за зарезервированным словом, отличным от **case**, **for** и **in**
- Третьего слова в команде **case** или **for** (относится только к слову **in**)

Расширенная оболочка Korn (ksh93):

Помимо стандартной оболочки Korn (`/usr/bin/ksh`), AIX содержит ее расширенную версию `/usr/bin/ksh93`. Расширенная версия практически полностью совместима со стандартной оболочкой, но содержит несколько дополнительных функций, отсутствующих в `/usr/bin/ksh`.

Некоторые сценарии могут выполняться в оболочке `ksh93` не так, как в стандартной оболочке, поскольку в ней немного по-другому реализована обработка переменных.

Примечание: Также предусмотрен расширенный вариант оболочки Korn с ограничениями, он вызывается командой `rksh93`.

Следующие функции недоступны в оболочке Korn `/usr/bin/ksh`, но доступны в оболочке Korn `/usr/bin/ksh93`:

Элемент	Описание
Дополнительные арифметические функции	Вы можете вызывать функции **libm** (математические функции, реализованные в языке программирования C) в стандартных арифметических выражениях, например `$ value=$((sqrt(9)))`. В новой оболочке реализовано больше операторов, включая унарные конструкции **+**, **++**, **--** и **?:** (например, "`x ? y : z`"), а также оператор **,** (запятая). Поддерживаются арифметические основания, не превышающие 64. Также поддерживаются вычисления с плавающей точкой. Для указания количества значащих разрядов можно ввести "**typeset -E**", а для указания количества десятичных позиций в вещественной переменной - "**typeset -F**". Переменная **SECONDS** теперь округляет число секунд сверху до ближайшей сотни, а не до целого числа секунд.

Составные переменные	В оболочке поддерживаются составные переменные. Составные переменные позволяют указывать несколько значений с помощью одного имени переменной. Конкретные значения указываются по индексу, отделенному от имени переменной с помощью точки (`.`). Например:
	`$ myvar=( x=1 y=2 )`
	`$ print "${myvar.x}"`
	`1`

Присвоения составных переменных	Присвоение составных переменных поддерживается при инициализации массивов, как индексированных, так и именованных. Присваиваемые значения нужно заключить в скобки, как показано в следующем примере:
	`$ numbers=( zero one two three )`
	`$ print ${numbers[0]} ${numbers[3]}`
	`zero three`

Именованные массивы	Именованный массив - это массив, в котором в качестве индексов применяются строковые значения.
	
	Для работы с именованными массивами применяется команда **typeset** с флагом **-A**. Например:
	`$ typeset -A teammates`
	`$ teammates=( [john]=smith [mary]=jones )`
	`$ print ${teammates[mary]}`
	`jones`

Ссылки на переменные	С помощью команды **typeset** с флагом **-n** можно присвоить одной переменной значение ссылки на другую. В этом случае изменение значения одной переменной немедленно вызовет изменение значения другой. Например:
	`$ greeting="hello"`
	`$ typeset -n welcome=greeting     # создание ссылки`
	`$ welcome="Привет!"     # переопределение значения`
	`$ print $greeting`
	`Привет!`

Развертывание параметров	Существуют следующие конструкции развертывания параметров: • `${!переменная}` - имя переменной. • `${!`*`переменная`*`[@]}` - имена индексов массива *переменная*. • `${`*`параметр:смещение`*`}` подстрока *параметра*, начиная со *смещения*. • `${параметр:смещение:число}` подстрока *параметра* , начиная со *смещения*, заданное *число* символов. • `${@:`*`смещение`*`}` задает все позиционные параметры, начиная со *смещения*. • `${@:`*`смещение:число`*`}` задает требуемое *число* позиционных параметров, начиная со *смещения*. • `${`*`параметр/шаблон/замена`*`}` заменяет в *параметре* первое вхождение *шаблона* на значение *замены*. • `${`*`параметр//шаблон/замена`*`}` заменяет в *параметре* все вхождения *шаблона* на значение *замены*. • `${`*`параметр/#шаблон/замена`*`}` - если *параметр* начинается с *шаблона*, то *параметр* заменяется на значение *замены*. • `${`*`параметр/%шаблон/замена`*`}` - если *параметр* заканчивается на *шаблон*, то *параметр* заменяется на значение *замены*.

Элемент	Описание

Дисциплинарные функции	Дисциплинарная функция - это функция, связанная с конкретной переменной. Эта функция будет вызываться каждый раз при обращении, присвоении или сбросе переменной. Функции задаются в виде *переменная.функция*, где *переменная* - это имя переменной, а *функция* - дисциплинарная функция. Существуют следующие предопределенные дисциплинарные функции: **get**, **set** и **unset**.
	• Функция ***переменная*.get** вызывается каждый раз при обращении к *переменной*. Если этой функцией задается специальная переменная **.sh.value**, то значение *переменной* изменяется на это значение. Например:
	`$ function time.get`
	`> {`
	`>  .sh.value=$(date +%r)`
	`> }`
	`$ print $time`
	`09:15:58 AM`
	`$ print $time  # изменится через несколько секунд`
	`09:16:04 AM`
	• Функция ***переменная*.set** вызывается каждый раз при присвоении *переменной*. Переменная **.sh.value** становится равной присваиваемому значению. Значение *переменной* будет равно значению **.sh.value** после выполнения функции. Например:
	`$ function adder.set`
	`> {`
	`>  let .sh.value="`
	`$ {.sh.value} + 1"`
	`> }`
	`$ adder=0`
	`$ echo $adder`
	`1`
	`$ adder=$adder`
	`$ echo $adder`
	`2`
	• Функция ***переменная*.unset** вызывается каждый раз при сбросе *переменной*. Переменная сбрасывается только в том случае, если эта операция явно выполняется функцией; в противном случае, переменная сохраняет свое значение.
	
	Во всех дисциплинарных функциях специальная переменная **.sh.name** равна имени переменной, а **.sh.subscript** - ее индексу, если он есть.

Среда выполнения функции	Функции, объявленные в формате *function функция*, выполняются в отдельной среде и поддерживают локальные переменные. Функции, объявленные в формате *функция*`()`, выполняются в той же среде, что и родительская оболочка.
Переменные	Переменные, начинающиеся с `.sh.`, зарезервированы оболочкой и имеют специальное значение. Переменные **.sh.name**, **.sh.value** и **.sh.subscript** описаны в разделе таблицы Дисциплинарные функции. Кроме того, переменная **.sh.version** содержит значение версии оболочки.
Значения, возвращаемые командами	Команды возвращают следующие значения: • Если выполняемая команда не найдена, то возвращается значение 127. • если команда найдена, но не является выполняемым файлом, то возвращается значение 126. • Если команда выполнялась, но была завершена по сигналу, то возвращается значение 256 плюс номер сигнала.
Правила поиска в PATH	Сначала выполняется поиск специальных встроенных команд, затем всех остальных функций (включая находящиеся в каталогах **FPATH**), а затем всех встроенных функций.
Хронология оболочки	Команда **hist** позволяет просматривать список выполнявшихся ранее команд и редактировать выбранные команды. В оболочке ksh применялась команда **fc**. Теперь команда **fc** - это псевдоним команды **hist**. Переменная *HISTCMD* увеличивается на единицу каждый раз при вызове команды в оболочке, а переменная *HISTEDIT*указывает, какой редактор нужно применять при вызове команды **hist**.

Элемент	Описание
Встроенные команды	В расширенной оболочке Korn предусмотрены следующие встроенные команды: • Команда **builtin** показывает список всех встроенных команд. • Команда **printf** выполняет те же действия, что и библиотечная функция C **printf()**. См. команду **printf**. • Команда **disown** запрещает оболочке отправлять сигнал SIGHUP указанной команде. • Команда **getconf** аналогична обычной команде **/usr/bin/getconf**. Обратитесь к описанию команды **getconf**. • Для встроенной команды **read** предусмотрены следующие флаги: – **read -d {*символ*}** позволяет указать символьный разделитель вместо встроенного значения новой строки. – **read -t {*секунды*}** позволяет задать число секунд, по истечении которых команда **read** будет завершена по тайм-ауту. Если команда **read** завершена по тайм-ауту, она возвратит значение FALSE. • Для встроенной команды **exec** предусмотрены следующие флаги: – **exec -a {*имя*} {*команда*}** указывает, что аргумент 0 *команды* следует заменить на *имя*. – **exec -c {*команда*}** указывает команде **exec**, что перед вызовом *команды* нужно очистить среду выполнения. • Для встроенной команды **kill** предусмотрены следующие флаги: – **kill -n {*номер*}** применяется для указания номера сигнала, который нужно отправить процессу, а **kill -s {*имя*}** применяется для указания имени сигнала. – **kill -l** без аргументов показывает список имен сигналов, без их номеров. • Для встроенной команды **whence** предусмотрены следующие флаги: – Флаг **-a** показывает все совпадения, а не только первое совпадение. – Флаг **-f** указывает команде **whence**, что функции искать не надо. • Для команд **print** и **echo** обычно применяется esc-последовательность. Клавишу Esc (Escape) можно представить с помощью \E. • Все обычные встроенные команды распознают флаг **-?**, показывающий формат вызова команды. • Встроенная команда **getopts** требует, чтобы флаг **optstring** содержал начальный знак **+**, чтобы разрешить опции, начинающиеся с **+**.
Прочие различия между оболочкой Korn ksh и Korn ksh93	К прочим различиям относятся: • В оболочке Korn ksh93 нельзя экспортировать функции с помощью встроенной команды **typeset -fx**. • В оболочке Korn ksh93 нельзя экспортировать псевдоним с помощью встроенной команды **alias -x**. • В оболочке Korn ksh93 знак доллара с одинарной кавычкой ($') считается строкой ANSI C. Для воссоздания старого алгоритма поведения (ksh) следует заключать знак доллара в кавычки (\"$\"'). • Логика обработки аргументов для встроенных команд оболочки Korn ksh93 была изменена. Недокументированные сочетания аргументов, передаваемых встроенным командам оболочки Korn ksh не работают в оболочке Korn ksh93. Например, **typeset -4i** работает аналогично **typeset -i4** в оболочке Korn ksh, но не работает в оболочке Korn ksh93. • В оболочке Korn ksh93 подстановка команд и расширенные арифметические функции выполняются в особых переменных среды PS1, PS3 и ENV при расширении. Поэтому рекомендуется избегать использования символа ударения (`) и знака доллара с открывающейся кавычкой ($(), применяя обратную косую черту (\) для сохранения старого алгоритма поведения. Например, оболочка Korn ksh явно присваивает x=$'*имя*\t*оператор*' как $*имя*\t*оператор*; оболочка Korn ksh93 расширяет \t и присваивает значение *имя*<\t расширено>*оператор*. Для сохранения поведения оболочки Korn ksh следует заключать символ $ в кавычки. Например, x="$'*имя*\t*оператор*'. • Переменная *ERRNO* в оболочке Korn ksh93 отсутствует. • В оболочке Korn ksh93 имена файлов не расширяются для неинтерактивных оболочек после получения символа перенаправления. • В оболочке Korn ksh93 необходимо использовать опцию **-t** команды **alias** для отображения псевдонимов-следов. Функция отслеживаемых псевдонимов считается устаревшей и отображаемые псевдонимы могут не отслеживаться. • В оболочке Korn ksh93 в режиме emacs при нажатии клавиш Ctrl+T текущий и предыдущий символы будут поменяны местами. В ksh при нажатии Ctrl+T происходит замена текущего символа на следующий. • В оболочке Korn ksh93 запрещены несбалансированные скобки в ${*имя оператор значение*}. Например, для ${*имя*-(} требуется escape-последовательность, такая как ${*имя*-\(} для работы в обеих версиях. • В оболочке Korn ksh93 команда **kill -l** показывает только список имен сигналов, но не их числовые значения.

Код завершения в оболочке Korn (POSIX):

Если оболочка обнаруживает ошибки, например синтаксические, она возвращает ненулевой код завершения. Иначе оболочка возвращает код завершения последней выполненной команды.

После ошибок времени выполнения выводится имя команды или функции и причина ошибки. Если номер строки, на которой была обнаружена ошибка, больше 1, в квадратных скобках (**[ ]**) после имени команды или функции будет указан номер этой строки.

При обнаружении ошибок в специальных встроенных и прочих командах неинтерактивная оболочка выдаст диагностическое сообщение, согласно следующей таблице:

Ошибка	**Особая встроенная команда**	**Прочие утилиты**
Синтаксическая ошибка языка оболочки	будут завершены	будут завершены
Ошибка синтаксиса утилиты (ошибка в опции или операнде)	будут завершены	не будут завершены
Ошибка перенаправления	будут завершены	не будут завершены
Ошибка присвоения переменной	будут завершены	не будут завершены
Ошибка подстановки	будут завершены	будут завершены
Команда не найдена	неприменимо	могут быть завершены
Точечный сценарий не найден	будут завершены	неприменимо

При обнаружении ошибок, для которых указано "будут (могут быть) завершены", оболочка выполнения команды будет (может быть) завершена с ненулевым состоянием, однако сценарий, из которого запущена эта оболочка, не будет завершен.

Во всех перечисленных в таблице случаях интерактивная оболочка запишет в стандартный вывод диагностическое сообщение и продолжит работу.

Параметры в оболочке Korn:

Ниже описаны параметры оболочки Korn.

Параметр определяется следующим образом:

- Идентификатор с одним из следующих символов: звездочка (`*`), знак at (`@`), решетка (`#`), вопросительный знак (`?`), дефис (`-`), знак доллара (`$`) и восклицательный знак (`!`). Эти идентификаторы называются *особыми параметрами*.
- Аргумент, заданный в виде номера (*позиционный параметр*)
- Параметр, заданный в виде идентификатора, для которого определено значение и, возможно, дополнительные атрибуты (*именованный параметр/переменные*).

Значения и атрибуты именованных параметров можно изменять с помощью команды **typeset**. Информация об атрибутах, поддерживаемых оболочкой Korn, приведена в описании особой встроенной команды **typeset**. Экспортированными называются параметры, значения и атрибуты которых передаются в среду после выполнения команды.

Значения именованных параметров можно изменять с помощью следующей команды:

```
Имя=Значение [ Имя=Значение ] ...
```

Если для параметра **имя** будет задан атрибут **-i** (указывающий, что значение параметра - целое число), то для параметра **Значение** будет вычислено арифметическое выражение.

Оболочка поддерживает одномерные массивы. К элементам массива можно обращаться по их номерам. Номер элемента может быть задан как арифметическое выражение, заключенное в квадратные скобки `[ ]`.

Для того чтобы присвоить значения всем элементам массива, нужно выполнить команду `set -A` *имя значение ...* . Значения номеров элементов должны лежать в диапазоне от 0 до 511. Массивы не требуется объявлять. Для любого параметра всегда можно указать произвольный допустимый номер элемента, и при необходимости массив будет создан автоматически. Обращение к массиву без указания номера элемента равносильно обращению к элементу 0.

Позиционным параметрам значения присваиваются с помощью особой команды **set**. Параметру **$0** присваивается значение аргумента `0` при вызове оболочки. Символ `$` представляет параметры, вместо которых можно подставить их значение.

Понятия, связанные с данным:

"Запуск оболочки" на стр. 253
Оболочку Korn можно запустить с помощью команды **ksh**, **psh** (оболочка POSIX) или **exec**.

"Функции оболочки Korn" на стр. 254
Зарезервированное слово **function** предназначено для определения функций оболочки. Оболочка считывает определение функции и сохраняет его в текущей среде. При разборе определений функций разворачиваются псевдонимы. Оболочка выполняет функции точно так же, как команды, и передает в них аргументы как позиционные параметры.

"Арифметические операции в оболочке Korn (POSIX)" на стр. 216
Стандартная встроенная команда **let** оболочки Korn (POSIX) позволяет выполнять арифметические операции над целыми числами.

Ссылки, связанные с данной:

"Составные команды оболочки Korn" на стр. 252
Составная команда представляет собой список простых команд, конвейер или команду, начинающуюся с зарезервированного слова. Как правило, составные команды (например, **if**, **while** или **for**) применяются в сценариях оболочки.

Подстановка параметров в оболочке Korn и POSIX:

В оболочке Korn (POSIX) предусмотрены функции подстановки значений параметров.

Предусмотрены следующие операции подстановки параметров:

Элемент	Описание
${*Параметр*}	Оболочка считывает символы, заключенные между ${ и }, и интерпретирует их как одно слово, даже если в нем содержатся фигурные скобки или метасимволы. Вместо имени указанного в скобках параметра подставляется его значение. Фигурные скобки требуется указывать в случаях, когда после значения *параметр* стоит буква, цифра или знак подчеркивания (иначе они будут считаться частью имени параметра), а также при обращении к элементам именованного параметра-массива. Если в имени параметра содержится хотя бы одна цифра, он называется *позиционным параметром*. Позиционные параметры, состоящие из нескольких цифр, должны заключаться в фигурные скобки. Если вам нужно обработать все позиционные параметры одновременно, укажите особый параметр `*` или `@` - вместо него подставляются значения всех позиционных параметров, начиная с `$1` (значения параметров будут разделены символом разделения полей). Идентификатор массива с индексом (номером элемента) `*` или `@` заменяется на значения всех элементов этого массива.
${*#параметр*}	Если значение *параметра* - `*` или `@`, то подставляется количество позиционных параметров. В противном случае подставляется длина позиционного параметра с номером *параметр*.
${*#идентификатор*[*]}	Заменяется на число элементов в массиве *идентификатор*.
${*параметр*:-*значение*}	Если *параметр* определен, и ему присвоено непустое значение, то подставляется его значение, в противном случае подставляется значение параметра *значение*.
${*параметр*:=*значение*}	Если *параметр* не определен, или ему присвоено пустое значение, то будет присвоено указанное *значение*. Этот способ неприменим для изменения значений позиционных параметров.

Элемент	Описание
${*параметр*:?*значение*}	Если *параметр* определен, и ему присвоено непустое выражение, то подставляется его значение. В противном случае выдается указанное *значение* и работа оболочки завершается. Если *значение* не указано, то выдается стандартное сообщение.
${*параметр*:+*значение*}	Если *параметр* определен и ему присвоено непустое значение, то подставляется указанное *значение*.
${*параметр*#*шаблон*} | ${*параметр*##*шаблон*}	Если значение *параметра* начинается с указанного *шаблона*, то подставляется значение *параметра* за вычетом символов шаблона. В противном случае подставляется полное значение *параметра*. В первом формате из подставляемого значения удаляется шаблон минимальной возможной длины; во втором формате - шаблон максимальной возможной длины.
${*параметр*%*шаблон*} | ${*параметр*%%*шаблон*}	Если значение *параметра* оканчивается на указанный *шаблон*, то подставляется значение *параметра* за вычетом символов шаблона. В противном случае подставляется полное значение *параметра*. В первом формате из подставляемого значения удаляется шаблон минимальной возможной длины; во втором формате - шаблон максимальной возможной длины. Во всех выражениях, указанных выше, параметр *значение* вычисляется только в случае, если оно должно быть подставлено. Таким образом, в следующем примере команда **pwd** будет выполнена только в том случае, если флаг **-d** не определен или пуст: `echo ${d:-$(pwd)}`

Примечание: Если из любого из предыдущих выражений удалить символ **:**, то оболочка будет проверять только наличие параметра *параметр* (но не будет проверять его значение).

Понятия, связанные с данным:

“Неконтролируемые терминалы” на стр. 296
Если пользователь вошел в систему с некоторого терминала, а затем оставил терминал без присмотра, то уязвимость системы значительно повышается. Наиболее серьезные проблемы возникают в случае, когда таким пользователем является системный администратор (пользователь root). В общем случае, пользователи должны выходить из системы всякий раз, когда они оставляют терминал без присмотра.

Предустановленные особые параметры в оболочке Korn (POSIX):

Некоторые параметры задаются оболочкой Korn или POSIX автоматически.

Значения следующих параметров автоматически задаются оболочкой:

Элемент	Описание
@	Выдает значения позиционных параметров, начиная с $1. Параметры будут разделены пробелами. Если вы заключите символы $@ в двойные кавычки ", то каждый позиционный параметр будет считаться отдельной строкой. Если не определен ни один позиционный параметр, то вместо $@ будет подставлена пустая строка.
*****	Выдает значения позиционных параметров, начиная с $1. Параметры отделяются друг от друга при помощи символа, который указан первым в параметре **IFS**. Если символы $* заключены в двойные кавычки " , то выданные значения позиционных параметров также будут заключены в двойные кавычки. Как и в первом случае, значения параметров будут разделены первым символом параметра **IFS**.
#	Выдает число позиционных параметров, переданных в оболочку, не считая собственно имени процедуры оболочки. Можно сказать, что параметр **$#** выдает номер последнего позиционного параметра. Одно из основных применений этого параметра - проверка наличия нужного числа аргументов.
-	Передает флаги в оболочку при ее запуске или при выполнении команды **set**.
?	Выдает код завершения последней выполненной команды. Это десятичная строка. Большинство команд возвращают нулевое значение (код успешного завершения). Сама оболочка возвращает текущее значение переменной **$?** в качестве кода завершения.

Элемент	Описание

$	Равен номеру процесса текущей оболочки. Поскольку в каждый момент времени номера всех активных процессов различны, они часто применяются в качестве уникальных имен для временных файлов.
	
	Ниже приведен пример создания такого временного файла:
	`temp=$HOME/temp/$$`
	`ls >$temp`
	`.`
	`.`
	`.`
	`rm $temp`

!	Задает номер процесса последней команды, запущенной в фоновом режиме.
ноль (0)	Передает имя оболочки или сценария оболочки.

Подстановка имен файлов в оболочке Korn (POSIX):

Оболочка Korn (POSIX) подставляет имена файлов вместо всех слов команды, заданных в переменной *Word*. Имя файла определяется с помощью специальных символов, указанных в слове.

Если слово команды содержит символы ***)**, **?** или **[**, а флаг **-f** не был задан, то оболочка рассматривает это слово как шаблон. Вместо указанного слова оболочка подставляет список имен файлов, удовлетворяющих шаблону, отсортированный в соответствии с последовательностью упорядочения для текущей локали. Если ни одно имя файла не соответствует шаблону, то слово не изменяется.

Если для подстановки имен файлов оболочка применяет шаблон, то символы **.** и **/** должны сопоставляться явно.

Примечание: Оболочка Korn не рассматривает эти символы как специальные.

Символы подстановки имеют следующее значение:

Элемент	Описание
*	Соответствует любой строке, в том числе пустой.
?	Заменяет любой символ.
[...]	Заменяет любой из заключенных в скобки символов. Пара символов, разделенных символом (-), заменяет любой символ из указанного интервала в соответствии с последовательностью упорядочения, определяемой текущей локалью. Если сразу за открывающей скобкой [указан символ !, то заменяются все символы, кроме символов в скобках. Для того чтобы указать в наборе символов дефис (-), задайте его в качестве первого или последнего символа.

Выражение `[:класс-символов:]` заменяет имена файлов из указанного интервала. Система выберет все имена файлов, в которых содержится хотя бы один символ из указанного `класса`. Определение классов задается в разделе **LC_CTYPE** подпрограммы $_{\text{setlocale}}$. Распознаются все классы символов, заданные в текущей локали.

Ниже перечислены некоторые имена классов символов:

- **alnum**
- **alpha**
- **cntrl**
- **digit**
- **graph**
- **lower**
- **print**
- **punct**
- **space**

- **upper**
- **xdigit**

Например, `[[:upper:]]` соответствует любой прописной букве.

Оболочка Korn поддерживает расширение имен файлов на основе элементов последовательности упорядочения, символов и классов эквивалентности.

Список_шаблонов - это список из одного или нескольких шаблонов, элементы которого отделены друг от друга символом |. Составные шаблоны содержат одно или несколько выражений следующего вида:

Элемент	Описание
?(*PatternList***)**	Заменяет любой из указанных шаблонов
(Список-шаблонов***)**	Заменяет ноль и более указанных шаблонов
+(*Список-шаблонов***)**	Заменяет по крайней мере один указанный шаблон
@(*Список-шаблонов***)**	Заменяет в точности один из указанных шаблонов
!(*PatternList***)**	Заменяет любую строку, за исключением одного из указанных шаблонов

На соответствие шаблону накладываются определенные ограничения. Если имя файла начинается с точки (`.`), то он соответствует только тому шаблону, который также начинается с точки. Например, символ `*` соответствует именам файлов `myfile` и `yourfile`, но не соответствует именам `.myfile` и `.yourfile`. Для таких имен файлов нужно задать шаблон следующего вида:

```
.*file
```

Если нет ни одного имени файла, соответствующего шаблону, то в качестве результата поиска возвращается сам шаблон.

В именах файлов и каталогов не допускаются символы `*`, `?`, `[` и `]`, поскольку при разборе шаблона они могут вызвать бесконечную рекурсию (зацикливание).

Удаление кавычек:

Некоторые символы будут удалены если они не заключены в кавычки.

Символы обратной косой черты ((\)), одинарных кавычек (('')) и двойных кавычек (("")) будут удалены из исходного текста, если только они сами не были взяты в кавычки.

Перенаправление ввода и вывода в оболочке Korn (POSIX):

Перед выполнением команды оболочка Korn выполняет поиск символов перенаправления в командной строке. Такие символы означают, что необходимо перенаправить ввод или вывод.

Они могут быть заданы как перед, так и после команды. Сами символы не передаются вызываемой команде.

За исключением описанных случаев, оболочка всегда выполняет подстановку команд и параметров в переменных **Слово** и **Цифра**. Подстановка имени файла выполняется только тогда, когда шаблону соответствует единственное имя, и он не может быть интерпретирован, как пустая строка.

Элемент	Описание
<*Слово*	Стандартный ввод связывается с файлом, заданным в параметре **Слово** (ему присваивается дескриптор файла, равный 0).
>*Слово*	Стандартный вывод связывается с файлом, заданным в параметре **Слово**(файлу присваивается дескриптор, равный 1). Если файл не существует, то оболочка создает его. Если файл существует и включена опция **noclobber,** то будет отправлено сообщение об ошибке; если же опция выключена, то все данные из файла удаляются. **Примечание:** Если для нескольких оболочек включен режим **noclobber** и их вывод перенаправляется в один и тот же файл, то возможно возникновение ситуации, при которой несколько оболочек пишут данные в один и тот же файл. Оболочка не отслеживает и не предотвращает возникновение таких ситуаций.
>|*Слово*	Действие этой команды аналогично действию команды **>***Слово*, однако этот оператор перенаправления подавляет опцию **noclobber**.
> >*Слово*	Стандартный вывод связывается с файлом, заданным в параметре **Слово.** Если файл уже существует, то оболочка добавляет вывод к этому файлу (за символом конца файла). Если файл не существует, то оболочка создает его.
<>*Слово*	Стандартный ввод связывается с файлом, заданным в параметре **Слово,** который открывается на чтение и запись.
<<[-]*Слово*	Строки текста считываются до тех пор, пока не будет обнаружена строка, заданная в параметре **Слово**, или символ конца файла. Оболочка не выполняет подстановку параметров, команд и имен файлов в указанном файле. Получившийся текст, который называется документом ввода с консоли, применяется в качестве стандартного ввода. Если строка **Слово** указана в кавычках, то подстановки в тексте документа выполняться не будут.

Документ ввода с консоли рассматривается как одно слово, которое начинается с символа новой строки и заканчивается строкой, содержащей только заданный разделитель, после которого нет дополнительных пробелов. Затем начинается следующий документ ввода с консоли. Применяется следующий формат:

```
[n]<<слово
   документ ввода с консоли
разделитель
```

Если параметр *слово* задан в кавычках, то разделителем считается строка *слово* без кавычек. В строках *документа ввода с консоли* подстановки выполняться не будут. Если параметр *слово* указан без кавычек, то разделителем считается сама строка. В этом случае во всех строках документа ввода с консоли выполняется подстановка параметров, команд и чисел.

Для перенаправленных потоков данных оболочка выполняет подстановку параметров. Если символ \, $, ` или первый символ параметра **Слово** не должен интерпретироваться как специальный, то укажите перед ним символ \.

Если вместе с **<<** задан символ **-** , то в параметре **Слово** и строках вводимого документа оболочка удалит все начальные символы табуляции.

Элемент	Описание
<&*Цифра*	Стандартный ввод связывается с файлом, дескриптор которого указан в параметре **Цифра**
>&*Цифра*	Стандартный вывод связывается с файлом, дескриптор которого указан в параметре **Цифра**
<&-	Закрывает файл стандартного ввода
>&-	Закрывает файл стандартного вывода
<&p	Связывает ввод параллельного процесса со стандартным вводом
>&p	Записывает вывод параллельного процесса в стандартный вывод

Если перед одной из опций перенаправления задан дескриптор файла, то вместо стандартного ввода или вывода будет применяться этот файл. В приведенном ниже примере оболочка открывает файл с дескриптором 2 на запись. Этот файл будет применяться вместо стандартного вывода (дескриптора 1):

```
... 2>&1
```

Важен порядок символов перенаправления. С каждым символом перенаправления оболочка связывает пару (*Дескриптор_файла*, *Файл*). Например, в операторе:

```
... 1>Файл 2>&1
```

дескриптор 1 связывается с файлом, заданным в параметре **Файл**. Затем дескриптор файла 2 связывается с файлом, дескриптор которого равен 1 (*Файл*). Если изменить порядок операций перенаправления, то дескриптор файла 2 будет связан с терминалом (раньше связанным с дескриптором файла 1), а затем дескриптор файла 1 будет связан с файлом, заданным в параметре **Файл**.

Если после текста команды указан амперсанд (&), и управление заданиями выключено, то по умолчанию стандартный ввод команды связывается с пустым файлом `/dev/null`. Если же применяется управление заданием, то среда выполнения команды содержит те дескрипторы файлов вызывающей оболочки, которые заданы в спецификациях ввода-вывода.

Понятия, связанные с данным:

“Перенаправление ввода и вывода” на стр. 342
Операционная система AIX позволяет управлять потоками ввода и вывода данных с помощью специальных символов и команд ввода-вывода.

Задачи, связанные с данной:

“Перенаправление вывода во ввод” на стр. 346
Можно перенаправить вывод в документы ввода.

Создание подчиненных процессов:

Оболочка Korn (POSIX) позволяет запустить несколько параллельных фоновых процессов. Соответствующие команды, которые запускаются из сценария оболочки, называются *параллельными процессами*.

Для запуска параллельного процесса после текста команды нужно указать оператор `|&`. Стандартный ввод и вывод команды перенаправляется в соответствии с конвейером, определяемым вашим сценарием.

На параллельный процесс накладываются следующие ограничения:

- В конце каждого сообщения должен находиться символ новой строки
- Все сообщения должны отправляться в стандартный вывод
- После записи каждого сообщения должна выполняться очистка стандартного вывода

В следующем примере показано, как входные данные передаются параллельному процессу, и как формируется его вывод:

```
echo "Исходный процесс"
./FileB.sh |&
read -p a b c d
echo "Считать с параллельного процесса: $a $b $c $d"
print -p "Передано параллельному процессу"
read -p a b c d
echo "Передано обратно из параллельного процесса: $a $b $c $d"
```

```
FileB.sh
      echo "Параллельный процесс выполняется"
   read a b c d
   echo $a $b $c $d
```

В результате в стандартный вывод будет записано следующее:

```
Исходный процесс
Считано из параллельного процесса: Параллельный процесс выполняется
Передано из параллельного процесса: Передано в параллельный процесс
```

Для записи данных в параллельный процесс применяется команда **print -p**. Для чтения данных из параллельного процесса применяется команда **read -p**.

Понятия, связанные с данным:

“Команды оболочки Korn (POSIX)” на стр. 250
Оболочка Korn - это интерактивный интерпретатор и командный язык программирования. Она разработана

в соответствии с Международным стандартом интерфейса переносимых операционных систем (POSIX).

Перенаправление ввода и вывода:

Для того чтобы связать стандартный ввод и вывод параллельного процесса с файлом с заданным дескриптором, укажите операторы перенаправления ввода-вывода.

Например, команда:

```
exec 5>&p
```

задает в качестве устройства ввода параллельного процесса файл с дескриптором 5.

Аналогично с помощью стандартных операторов перенаправления можно перенаправить вывод параллельного процесса. Кроме того, можно запустить еще один параллельный процесс. Оба параллельных процесса будут выводить данные в один и тот же канал, чтение из которого выполняется с помощью команды **read -p**. Для того чтобы остановить параллельный процесс, введите:

```
read -u5
```

Встроенные команды оболочки Korn или POSIX:

В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

По умолчанию вывод этих команд направляется в файл с дескриптором 1, и если в команде нет синтаксических ошибок, то для нее устанавливается код завершения ноль (0). Разрешается перенаправление ввода и вывода. Встроенные команды подразделяются на два типа - *особые встроенные команды* и *обычные встроенные команды*.

Особые встроенные команды отличаются от обычных следующим:

- Синтаксическая ошибка в особой встроенной команде может привести к завершению работы оболочки. Если синтаксическая ошибка будет допущена в обычной команде, работа оболочки не будет завершена. В случае, если в особой встроенной команде допущена синтаксическая ошибка, но работа оболочки не завершается, то устанавливается ненулевой код завершения.
- После завершения особых встроенных команд остаются в силе все значения переменных.
- Опции перенаправления ввода-вывода обрабатываются после присвоения значений параметрам.

Кроме того, значения, указанные в операторах присваивания после особых команд **export**, **readonly** и **typeset**, анализируются по тем же правилам, что и значения обычных операторов присваивания. Для строки, указанной после знака равенства (=), выполняется подстановка по тильде, но при этом не выполняется разбиение на слова и подстановка имен файлов.

Понятия, связанные с данным:

“Команды оболочки Korn (POSIX)” на стр. 250
Оболочка Korn - это интерактивный интерпретатор и командный язык программирования. Она разработана в соответствии с Международным стандартом интерфейса переносимых операционных систем (POSIX).

“Функции оболочки Korn” на стр. 254
Зарезервированное слово **function** предназначено для определения функций оболочки. Оболочка считывает определение функции и сохраняет его в текущей среде. При разборе определений функций разворачиваются псевдонимы. Оболочка выполняет функции точно так же, как команды, и передает в них аргументы как позиционные параметры.

Ссылки, связанные с данной:

“Список особых встроенных команд оболочки Korn или POSIX” на стр. 218
В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

“Обычные встроенные команды оболочки Korn или POSIX” на стр. 218
Ниже приведен список обычных встроенных команд оболочки Korn или POSIX.

Описания особых встроенных команд для оболочки Korn или POSIX:

В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

Ниже описаны особые встроенные команды оболочки Korn:

```
:           eval        newgrp      shift
.           exec        readonly    times
break       exit        return      trap
continue    export      set         typeset
                                    unset
```

Элемент	Описание
: [*аргумент* ...]	Только интерпретирует аргументы. Эта команда применяется в случае, если необходимо указать какую-либо команду (например, в условии *then* команды **if**), но при этом данная команда не должна выполнять никаких действий.
. *файл* [*аргумент* ...]	Считывает указанный файл, а затем выполняет содержащиеся в нем команды. Команды выполняются в среде текущей оболочки. Путь поиска, указанный переменной *PATH*, используется для поиска каталога, содержащего указанный файл. Если заданы какие-либо аргументы, они переопределяют позиционные параметры. В противном случае позиционные параметры остаются без изменений. Код завершения равен коду завершения последней выполненной команды. Дополнительная информация о позиционных параметрах приведена в разделе “Подстановка параметров в оболочке Korn и POSIX” на стр. 227. **Примечание:** Команда *.файл* [*аргумент* ...] сначала считывает весь файл, а затем выполняет указанные в нем команды. Поэтому команды **alias** и **unalias**, содержащиеся в файле, не действуют для любых функций, определенных в файле.
break [*n*]	Прерывает цикл команды **for**, **while**, **until** или **select**, если он выполняется. С помощью необязательного параметра *n* можно задать число циклов, которые нужно завершить. Значение *n* должно быть натуральным числом.
continue [*n*]	Возобновляет следующий внешний цикл команды **for** , **while**, **until** или **select**. Если вы укажете переменную *n*, то команда возобновит цикл n^{th}-го уровня. Значение *n* должно быть натуральным числом.
eval [*аргумент* ...]	Разбирает указанные аргументы как входную информацию для оболочки и выполняет полученную в итоге команду или команды.
exec [*аргумент* ...]	Выполняет команду, указанную в аргументе, в процессе текущей оболочки (без создания нового процесса). Входные и выходные аргументы выполняемой команды применяются к текущему процессу. Если команда **exec** будет указана без аргументов, она изменит дескрипторы файлов в соответствии со списком перенаправления ввода и вывода. В этом случае все дескрипторы файлов с номерами, превышающими 2, открытые с помощью этой команды, автоматически закрываются при вызове любой другой команды.
exit [*n*]	Завершает работу оболочки с кодом выхода *n*. Параметр *n* должен быть целым числом от 0 до 255. Если параметр *n* не будет указан, то код завершения оболочки будет равен коду завершения последней выполненной команды. Кроме того, если не включена опция **ignoreeof** команды **set**, работа оболочки завершается при обнаружении символа конца файла.
export -p [*имя*[**=** *значение*]] ...	Помечает указанные имена для автоматического экспорта в среду при выполнении последующих команд. Флаг **-p** выдает имена и значения всех экспортируемых переменных в следующем формате: `"export %s= %s\n", <имя> <значения>`
newgrp [*группа*]	Аналогична команде **exec/usr/bin/newgrp** [*группа*]. **Примечание:** Эта команда не возвращает результатов.
readonly -p [*имя*[**=** *значение*]] ...	Помечает имена, указанные в параметре *имя*, как неизменяемые. Указанные значения нельзя будет изменять в дальнейшем. Флаг **-p** выдает имена и значения всех экспортируемых переменных в следующем формате: `"export %s= %s\n", <имя> <значения>`

Элемент	Описание
возврат [*n*]	Возвращает управление из текущей функции в сценарий, из которого она была вызвана. С помощью необязательного параметра *n* можно задать код возврата. Если параметр *n* не будет указан, то код завершения оболочки будет равен коду завершения последней выполненной команды. Вне функций и сценариев команда **return** равносильна команде **exit**.
set [**+** |**-abCefhkmnostuvx**] [**+** |**-o** *опция*]... [**+** |**-A***имя*] [*аргумент* ...]	Если команда **set** указана без опций и аргументов, она выдает список переменных среды в соответствии с последовательностью упорядочения для текущей локали. Если эта команда указана с какими-либо опциями, то она задает или отменяет соответствующие атрибуты оболочки, как описано ниже: **-A** Присваивает значения элементам массива. Сначала значение параметра *имя* будет сброшено, а затем его элементам будут последовательно присвоены значения из списка *аргументов*. Если будет указан флаг **+A**, то значение параметра *имя* не будет предварительно сброшено. **-a** Автоматически экспортирует все последующие определенные параметры. **-b** В асинхронном режиме уведомляет пользователя о завершении фоновых заданий. **-C** Аналогична опции `set -o noclobber`. **-e** Выполняет прерывание ERR, если оно установлено, и завершает работу при ненулевом коде возврата команды, если только эта простая команда не относится к следующим категориям: + содержится в списке && или || + команда следует непосредственно после if, while или until + содержится в конвейере после символа ! Этот режим отключается при чтении профайлов. **-f** Отключает подстановку имен файлов. **-h** Включает режим, в котором при первом вводе каждой команды она должна быть определена как псевдоним-след. **-k** Указывает, что в среду выполнения команды нужно поместить все аргументы присвоения параметров, а не только те, которые указаны непосредственно в строке команды перед ее именем. **-m** Запускает фоновые задания в отдельных процессах и информирует пользователя об их завершении. В сообщении о завершении фонового задания указан его код возврата. В системах с управлением заданиями этот флаг автоматически включается для интерактивных оболочек. За дополнительной информацией обратитесь к разделу “Управление заданиями в оболочке Korn (POSIX)” на стр. 243. **-n** Проверяет наличие синтаксических ошибок в командах, но не выполняет их. Для интерактивных оболочек этот флаг игнорируется.

Элемент	Описание
-o ***Опция***	Если опция -o указана без аргумента, то будут показаны текущие значения опций. Для одной командной строки **ksh** можно определить несколько опций. С помощью флага **+o** можно аннулировать указанную опцию. Если эта команда указана с аргументами, она задает или отменяет позиционные параметры оболочки. Для *Опции* допустимы следующие значения: **allexport** Равносильно флагу **-a**. **bgnice** Запускает все фоновые задания с более низким приоритетом. Этот режим включен по умолчанию. **emacs** Включает встроенный редактор, подобный emacs, при вводе команд. **errexit** Равносильно флагу **-e**. **gmacs** Включает встроенный редактор, подобный gmacs, при вводе команд. **ignoreeof** Не завершает работу оболочки при обнаружении символа конца файла. Для выхода из оболочки нужно ввести команду **exit** или нажать клавиши Ctrl-D более 11 раз. **keyword** Равносильно флагу **-k**. **Примечание:** Этот флаг предусмотрен только для совместимости с оболочкой Bourne. Применять его настоятельно не рекомендуется. **markdirs** Добавляет символ косой черты (/) к именам каталогов, получаемым в результате подстановки имен файлов. **monitor** Равносильно флагу **-m**. **noclobber** Запрещает усечение существующих файлов при перенаправлении вывода команд. Если включена эта опция, то для усечения существующего файла нужно указать вертикальную черту после символа (>|). **noexec** Равносильно флагу **-n**. **noglob** Равносильно флагу **-f**. **nolog** Запрещает сохранять определения функций из файлов `.profile` и `$ENV`. **nounset** Равносильно флагу **-u**. **privileged** Равносильно флагу **-p**.

Элемент	Описание
	trackall Равносильно флагу **-h**. **verbose** Равносильно флагу **-v**. **vi** Переходит в режим встроенного редактора, подобного vi, при вводе команд. Для перехода в режим перемещения курсора нужно ввести escape-символ 033. Для выполнения введенной команды нужно нажать Enter. **viraw** Если будет включена эта опция, то в режиме vi каждый символ будет обрабатываться непосредственно после ввода. **xtrace** Равносильно флагу **-x**. **-p** Отключает обработку файла `$HOME/.profile` и применяет файл `/etc/suid_profile` вместо файла `ENV`. Этот режим включается всегда, когда действительный ИД пользователя (UID) или группы (GID) не совпадает с фактическим UID или GID. Если вы отключите эту опцию, то действительные ИД пользователя и группы будут всегда равны фактическим UID и GID. **Примечание:** Система не поддерживает опцию **-p**, поскольку операционная система не поддерживает сценарии оболочки для команды **setuid**. **-s** Упорядочивает позиционные параметры по алфавиту. **-t** Завершает работу после считывания и выполнения одной команды. **Примечание:** Этот флаг предусмотрен только для совместимости с оболочкой Bourne. Применять его настоятельно не рекомендуется. **-u** Включает режим, в котором наличие неопределенных параметров при подстановке считается ошибкой. **-v** Показывает входные строки оболочки по мере их считывания. **-x** Показывает команды и их аргументы перед выполнением. **-** Отключает флаги **-x** и **-v** и заканчивает разбор строки аргументов. — Запрещает изменять флаги. Эта опция часто применяется для того, чтобы присвоить параметру *$1* значение, начинающееся с символа `-`. Если помимо этого флага не указан ни один аргумент, то позиционные параметры не будут определены. Если перед каким-либо флагом команды **set** будет указан символ `+` вместо символа `-`, то данный флаг будет отключен. Все перечисленные флаги можно применять при запуске оболочки. Если команда '`set +o`' вызвана без аргументов, она показывает текущие значения опций в формате, удобном для ввода в оболочку. Текущий набор флагов хранится в параметре **$-**. Если не указан флаг **-A**, оставшиеся параметры рассматриваются как позиционные аргументы, и их значения присваиваются переменным `$1`, `$2`, `...` и т.д. Если не будет указан ни один аргумент, то имена и значения всех именованных параметров будут направлены в стандартный вывод.
shift [*n*]	Переименовывает позиционные параметры с `$n+1` по `$1`. По умолчанию *n* равно 1. Для *n* допустимы любые арифметические выражения, результатом которых будет неотрицательное число, не превышающее значение параметра `$#`.
times	Показывает общее время, которое заняло выполнение пользовательских и системных процессов в оболочке.

Элемент	Описание
trap [*команда*] [*сигнал*] ...	Выполняет указанную команду, когда оболочка получает один из перечисленных сигналов. Параметр *команда* считывается один раз при определении прерывания и еще один раз при его активизации. В этой команде можно указывать как номера, так и имена сигналов. Команды прерываний выполняются в соответствии с номерами сигналов. Любая попытка задать прерывание (trap) по сигналу, проигнорированному при входе в текущую оболочку, будет неудачной. Если в качестве команды будет указано значение **-**, то будут восстановлены исходные состояния всех прерываний. Если в команде trap не будет указана команда прерывания, то будут восстановлены первоначальные значения всех указанных сигналов. **Примечание:** В этом случае первый из перечисленных сигналов должен быть задан по номеру, так как в противном случае команда trap интерпретирует его как команду прерывания. Если для сигнала будет задано значение **ERR** , то указанная команда будет автоматически выполняться после любых команд с ненулевым кодом завершения. Если для сигнала будет задано значение **DEBUG**, то указанная команда будет выполняться после каждой команды. Если для сигнала будет задано значение **0** или **EXIT**, а команда **trap** будет выполняться в теле функции, то указанная команда будет выполнена после завершения этой функции. Если значение сигнала - **0** или **EXIT**, а команда **trap** будет выполняться вне функции, то указанная команда будет выполнена после выхода из оболочки. **Примечание:** Если сценарий получает сигнал `SIGINT` внутри функции, то сигнал `EXIT` нельзя перехватить, если существует оболочка. Команда **trap** без аргументов показывает список команд, связанных с каждым номером сигнала. Если указана пустая команда с помощью пустых кавычек (""), то команда **ksh** игнорирует сигнал. Дополнительная информация о чтении символа в качестве регулярного символа оболочкой Korn или POSIX приведена в разделе “Заключение символов в кавычки в оболочке Korn (POSIX)” на стр. 220. Полный список значений параметра *Сигнал* команды **trap** без префикса **SIG** приведен в описании функций **sigaction**, **sigvec** или **signal** в книге *Technical Reference: Base Operating System and Extensions, Volume 2*.

Элемент	Описание
typeset [**+HLRZfilrtux** [*n*]] [*имя*[**=***значение*]] ...	Позволяет задавать атрибуты и значения параметров оболочки. Если эта команда выполняется в теле функции, то она создает новый экземпляр параметра *имя*. После завершения функции восстанавливаются исходные тип и значение параметра. Команда **typeset** поддерживает следующие флаги: **-H** Задает соответствие между файлами AIX и файлами локальной системы для систем, отличных от AIX. **-L** Удаляет начальные пробелы из параметра *значение* и выравнивает его по левой границе поля. Если при этом будет указано ненулевое значение для параметра *n*, то оно задает ширину поля; в противном случае ширина поля будет равна ширине первого значения, присвоенного параметру. Если длина нового значения параметра не будет равна длине поля, то значение будет усечено или дополнено пробелами справа. Если дополнительно будет указан флаг **-Z**, то из значений будут удаляться незначащие нули. Этот флаг отключает флаг **-R**. **-R** Выравнивает значения по правой границе поля и при необходимости дополняет их начальными пробелами. Если при этом будет указано ненулевое значение для параметра *n*, то оно задает ширину поля; в противном случае ширина поля будет равна ширине первого значения, присвоенного параметру. Новое значение параметра при необходимости дополняется пробелами или усекается справа. Этот флаг отключает флаг **L**. **-Z** Выравнивает значение по правой границе поля и дополняет его начальными нулями, если первый символ значения, отличный от пробела, - цифра, и не задан флаг **-L**. Если при этом будет указано ненулевое значение для параметра *n*, то оно задает ширину поля; в противном случае ширина поля будет равна ширине первого значения, присвоенного параметру. **-f** Указывает, что в команде задано имя функции, а не имя параметра. С этим флагом запрещено присваивание и допустимы только флаги **-t**, **-u** и **-x**. Флаг **-t** включает трассировку выполнения функции. Флаг **-u** помечает функцию как неопределенную. При необходимости поиск определения функции выполняется в каталогах *FPATH*. Если указан флаг **-x**, то определение функции будет доступно только сценариям оболочки, запущенным из текущего сеанса **ksh**. **-i** Указывает, что параметр - это целочисленное значение (для ускорения арифметических операций). Если при этом будет указано ненулевое значение параметра *n*, то оно задает систему счисления, в которой будут выдаваться выходные данные; в противном случае система счисления определяется автоматически по первому значению, присваиваемому данному параметру. **-l** Заменяет прописные буквы на строчные. Этот флаг отключает флаг **-u**. **-r** Помечает имена, указанные в параметре *имя*, как неизменяемые. Указанные значения нельзя будет изменять в дальнейшем. **-t** Помечает указанные параметры тегами. Теги могут быть определены пользователем; они игнорируются оболочкой. **-u** Заменяет строчные буквы на прописные. Этот флаг отключает флаг **-l**. **-x** Помечает *имя* для автоматического экспорта в среду при выполнении последующих команд. Если вместо символа **-** будет указан символ **+**, то флаги команды **typeset** будут отключены. Если в этой команде будут указаны какие-либо флаги, но не будет указано имя, то будет показан список имен (и, возможно, их значений), для которых установлены эти флаги. (Если перед флагами будет указан символ **+**, то будут выданы только имена.) Если не будут указаны ни имена, ни флаги, то будет выдан список имен и атрибутов всех параметров.
unset [**-fv**] *имя* ...	Аннулирует значения и атрибуты указанных параметров. Если при этом будет указан флаг **-v**, то параметр *имя* считается именем переменной, и эта переменная будет удалена из среды. С помощью этой команды нельзя отменять неизменяемые переменные. Если вы сбросите какую-либо из особых переменных *ERRNO*, *LINENO*, *MAILCHECK*, *OPTARG*, *OPTIND*, *RANDOM*, *SECONDS*, *TMOUT* и _ (символ подчеркивания), то она потеряет свое особое значение, даже если будет впоследствии восстановлена. Если будет указан флаг **-f**, то *имя* считается именем функции, и определение этой функции будет отменено.

Описания обычных встроенных команд для оболочки Korn или POSIX:

Ниже описаны встроенные команды оболочки Korn или POSIX.

В оболочке Korn предусмотрены следующие обычные встроенные команды:

```
alias           fg              print           ulimit
bg              getopts         pwd             umask
cd              jobs            read            unalias
command         kill            setgroups       wait
echo            let             setsenv            test            whence
fc
```

Элемент	Описание
alias [**-t**] [**-x**] [*псевдоним*[**=** *строка*]] ...	Создает или переопределяет псевдонимы, либо отправляет текущий список псевдонимов в стандартный вывод. Дополнительная информация приведена в описании команды **alias**.
bg [*ИД-задания*...]	Переключает указанные задания в фоновый режим. Если *ИД-задания* не указан, то в фоновый режим переключается текущее задание. За дополнительной информацией об управлении заданиями обратитесь к разделу “Управление заданиями в оболочке Korn (POSIX)” на стр. 243. За дополнительной информацией о выполнении заданий в фоновом режиме обратитесь к описанию команды **bg**.
cd [*аргумент*] **cd***текущий новый*	Эту команду можно указывать в двух различных форматах. В первом формате она делает текущим каталог, заданный командой *аргумент*. Если в качестве *аргумента* указан символ **-**, то текущим будет сделан предыдущий каталог. Если **аргумент** не указан, то вместо него будет подставлено значение переменной оболочки *HOME*. После выполнения команды cd имя нового текущего каталога присваивается переменной **PWD**. Переменная оболочки **CDPATH** задает список каталогов, в которых выполняется поиск каталога, указанного в параметре *аргумент*. Имена каталогов в этой переменной разделяются символом **:** (двоеточие). По умолчанию переменная CDPATH не определена, и поиск указанного каталога выполняется в текущем каталоге. В переменной CDPATH текущий каталог обозначается пустым символом, поэтому чтобы добавить его в эту переменную, нужно указать в ней двоеточие сразу после знака равно или два двоеточия подряд между какими-либо каталогами. Если имя каталога, в который нужно перейти, начинается с символа косой черты (/), то его поиск выполняется относительно корневого каталога. В противном случае поиск этого каталога выполняется в каталогах, перечисленных в переменной CDPATH, или в текущем каталоге, если эта переменная не определена. Во втором формате команда **cd** находит в значении переменной **PWD** (имя текущего каталога) строку *текущий*, заменяет ее на строку *новый* и пытается перейти в каталог с полученным именем.
command [**-p**] *команда* [*аргумент* ...] **command** [**-v** | **-V**] *команда*	Указанная команда и ее аргументы будут интерпретироваться как обычная команда. Поиск функций оболочки выполняться не будет. Дополнительная информация приведена в описании команды **command**.
echo [*строка* ...]	Записывает строки символов в устройство стандартного вывода. Дополнительные сведения о формате приведены в описании команды **echo**. Флаг **-n** не поддерживается.
fc [**-r**] [**-e** *редактор*] [*первый* [*последний*]] **fc -l** [**-n**] [**-r**] [*первый* [*последний*]]	
fc -s [*старый***=** *новый*] [*первый*]	Позволяет просматривать и изменять содержимое файла хронологии команд, а также повторять введенные ранее команды. Дополнительная информация приведена в описании команды **fc**.

Элемент	Описание
fg [*ИД-задания*]	Переключает указанные задания в интерактивный режим. Если ИД задания не указан, то в интерактивный режим будет переключено текущее задание. За дополнительной информацией о выполнении заданий в интерактивном режиме обратитесь к описанию команды **fg**.
getopts*опции имя* [*аргумент*...]	Проверяет правильность опций в параметре *аргумент*. Дополнительная информация приведена в описании команды **getopts**.
jobs [**-l** | **-n** | **-p**] [*ИД-задания* ...]	Показывает состояние заданий, запущенных в среде текущей оболочки. Если *ИД-задания* не указан, то выдается информация о состоянии всех активных заданий. Учтите, что при завершении задания оболочка удаляет ИД его процесса из списка активных заданий. Дополнительная информация приведена в описании команды **jobs**.
kill [**-s** { *имя-сигнала* | *номер-сигнала* }] *ИД-процесса...* **kill** [*-имя-сигнала* | *-номер-сигнала*] *ИД-процесса...* **kill -l** [*состояние-завершения*]	Передает сигнал (по умолчанию **SIGTERM**) выполняющемуся процессу. Как правило, это приводит к завершению процесса. Для того чтобы завершить процесс, укажите его ИД в параметре *ИД-процесса*. Оболочка передает ИД всех процессов, выполняющихся в фоновом режиме (кроме случая, когда процессы запущены конвейером - тогда выдается только номер последнего процесса). Для определения ИД процесса можно воспользоваться командой **ps**. Содержит список имен сигналов. Дополнительная информация приведена в описании команды **kill**.
let*выражение ...*	Вычисляет указанное арифметическое выражение. Если результат последнего полученного выражения отличен от 0, то код завершения команды let будет равен 0, иначе 1. Дополнительная информация приведена в разделе “Арифметические операции в оболочке Korn (POSIX)” на стр. 216.
print [**-Rnprsu** [*n*]] [*аргумент ...*]	Направляет указанный аргумент в стандартный вывод оболочки. Если эта команда будет указана без флагов или с флагом минус (**-**) или два минуса (**-**), то аргументы будут направлены в стандартный вывод в том же формате, что и с помощью команды **echo**. Остальные флаги выполняют следующие функции: **-R** Выдает информацию "как есть" (т.е. escape-функции команды **echo** игнорируются). При этом также выдаются все аргументы и флаги, указанные после флага **-R**, за исключением флага **-n**. **-n** Отменяет добавление символа новой строки в вывод команды. **-p** Передает аргументы в конвейер процесса, запущенного с символами `|&`, а не в стандартный вывод. **-r** Выдает информацию в режиме "как есть". Escape-функции команды **echo** игнорируются. **-s** Выдает аргументы в файл хронологии, а не в стандартный вывод. **-u** Позволяет задать номер блока дескриптора файла (этот номер должен состоять из одной цифры), в который будет направлен вывод. Значение по умолчанию - 1.
pwd	Эквивалентна команде `print -r - $PWD`. **Примечание:** Внутренняя команда **pwd** оболочки Korn не поддерживает символьные связи.
read [**-prsu** [**n**]] [*имя?приглашение*] [*имя...*]	Считывает входную информацию оболочки. Одна команда read считывает одну строку и разделяет ее на поля в соответствии с тем, какие символы-разделители указаны в переменной **IFS**. Дополнительная информация приведена в описании команды **read**.
setgroups	Выполняет команду **/usr/bin/setgroups** в отдельной оболочке. Дополнительная информация приведена в описании команды **setgroups**. Однако, разницы в этом нет. Встроенная команда **setgroups** выполняется в отдельной оболочке, а внешняя заменяет текущую оболочку. Поскольку встроенная команда поддерживается только для обеспечения совместимости, рекомендуется указывать в сценариях полное имя команды `/usr/bin/setgroups`, а не встроенную команду оболочки.
setsenv	Позволяет запустить команду `/usr/bin/setsenv`, заменяющую текущую оболочку. Дополнительная информация приведена в описании команды **setsenv**.
test	То же, что и [*выражение*]. Информацию о формате и описание команды можно найти в разделе “Условные выражения в оболочке Korn (POSIX)” на стр. 219.

Элемент	Описание

ulimit [**-HSacdfmst**] [*ограничения*]	Задает или показывает ограничения на ресурсы пользовательских процессов из файла `/etc/security/limits`. В этом файле по умолчанию заданы следующие ограничения:
	`fsize = 2097151`
	`core = 2048`
	`cpu = 3600`
	`data = 131072`
	`rss = 65536`
	`stack = 8192`
	`threads = -1`
	
	Эти значения применяются по умолчанию при добавлении пользователя в систему. Их можно изменить с помощью команды **mkuser** при создании или команды **chuser** при изменении пользователя.
	
	Все ограничения делятся на гибкие и жесткие. С помощью команды **ulimit** пользователи могут изменять свои гибкие ограничения в пределах от 0 до жестких ограничений. Жесткие ограничения может изменять только пользователь с правами доступа root.
	
	Во многих случаях не обязательно устанавливать все ограничения. Для того чтобы задать ограничения на конкретный тип ресурсов, нужно указать параметр *ограничение*. Параметр *ограничение* должен быть числом, допустимым для указанного типа ресурсов, или значением `unlimited`. Допустимы следующие флаги команды **ulimit**:
	
	-H Указывает, что нужно изменить жесткое ограничение на указанный ресурс. Увеличивать жесткие ограничения могут только пользователи с правами root. Уменьшать их могут любые пользователи.
	
	-S Указывает, что нужно изменить гибкое ограничение на указанный ресурс. Гибкое ограничение не может превышать жесткое ограничение. Если не указана ни одна из опций **-H** и **-S**, то изменяются оба ограничения.
	
	-a Показывает текущий список ограничений на ресурсы.
	
	-c Задает ограничение на размер базового дампа в блоках по 512 байт.
	
	-d Задает размер области данных в блоках по 512 байт.
	
	-f Задает максимальный размер создаваемых файлов в блоках по 512 байт (при этом считывать можно файлы любого размера).
	
	-m Задает объем физической памяти в КБ.
	
	-n Задает максимальное число дескрипторов файлов, которое может открыть процесс.
	
	-r Задает предельное количество нитей за процесс.
	
	-s Задает размер области стека в КБ.
	
	-t Задает максимальное время (в секундах), которое может быть выделено каждому процессу.
	
	Для просмотра значения какого-либо ограничения нужно вызвать эту команду без параметра *ограничение*. По умолчанию выдается значение гибкого ограничения (с флагом **-H** - значение жесткого ограничения). При просмотре ограничений для нескольких типов ресурсов одновременно команда ulimit выдает перед каждым значением тип ресурсов, к которым оно относится. Если команда указана без опций, то считается, что указан флаг **-f**. Для того чтобы изменить только гибкое или только жесткое ограничение, укажите опцию **-H** или **-S**, так как по умолчанию и гибкое, и жесткое ограничения устанавливаются равными значению параметра *ограничение*.
	
	Дополнительная информация об ограничениях пользовательских и системных ресурсов приведена в описании функций **getrlimit**, **setrlimit**и **vlimit**.

umask [**-S**] [*маска*]	Определяет права доступа к файлу. Это значение, совместно с правами доступа создающего процесса, определяет права доступа к создаваемому файлу. Значение по умолчанию: 022. Если маска не задана, команда **umask** выдает значение маски прав доступа, применяемое в среде текущей оболочки. За дополнительной информацией о правах доступа к файлам обратитесь к описанию команды **umask**.
unalias { **-a** | *псевдоним...* }	Отменяет указанные (или все, если задан флаг **-a**) псевдонимы. Действие этой команды распространяется только на среду текущей оболочки. Дополнительная информация приведена в описании команды **unalias**.

Элемент	Описание
wait [*ИД-процесса...*]	Ожидает завершения указанного задания и завершается. Если ИД процесса не указан, то команда wait будет ожидать завершения всех активных подчиненных процессов. Состояние завершения этой команды равно состоянию завершения указанного процесса. Дополнительная информация приведена в описании команды **wait**.
whence [**-pv**] *имя ...*	Указывает способ интерпретации указанного имени, если оно введено в качестве команды. Команда **whence** без флагов показывает полное имя, соответствующее заданному имени (если оно есть). **-p** Выполняет поиск указанного имени или имен даже в случае, если эти имена - псевдонимы, функции или зарезервированные слова. **-v** Выдает подробную информацию о типе указанного имени.

Управление заданиями в оболочке Korn (POSIX):

Оболочка Korn (POSIX) содержит функции управления командными последовательностями, или *заданиями*.

При выполнении специальной команды **set -m** оболочка Korn связывает задание с каждым конвейером. Она также хранит таблицу текущих заданий, которую можно просмотреть командой **jobs**, и присваивает заданиям небольшие целые числа.

При запуске задания в фоновом режиме с помощью символа & оболочка выдает строку следующего вида:

```
[1] 1234
```

Данная строка означает, что запущенному в фоновом режиме заданию присвоен номер 1, и в нем есть один процесс (верхнего уровня) с ИД 1234.

Для запуска другой команды во время работы задания нажмите Ctrl-Z. Эта комбинация клавиш отправляет текущему заданию сигнал **STOP**. Обычно оболочка выдает сообщение об останове задания, а затем показывает командную строку. После этого можно изменить состояние приостановленного задания (например, перевести его в фоновый режим командой **bg**), запустить другую команду или вернуть задание в интерактивный режим командой **fg**. Комбинация клавиш Ctrl-Z действует немедленно, аналогично прерыванию, отменяя ожидание ввода и весь непрочитанный ввод оболочки.

Задание, работающее в фоновом режиме, при попытке чтения из терминала останавливается. Обычно заданиям в фоновом режиме разрешен вывод. Эту возможность можно отменить командой **stty tostop**. При этом фоновые задания будут останавливаться при попытке записи на терминал и чтения из терминала.

Указать фоновое задание в оболочке Korn можно несколькими путями. На задание можно сослаться по ИД любого его процесса, или одним из следующих способов:

Элемент	Описание
%*Номер*	Представляет задание с указанным номером.
%*Строка*	Представляет любое задание, запущенное с помощью команды, начинающейся со *Строки*.
%?*Строка*	Представляет любое задание, запущенное с помощью команды, содержащей *Строку*.
%%	Представляет текущее задание.
%+	Равносильно %%.
%-	Представляет предыдущее задание.

Данная оболочка немедленно фиксирует изменение состояния процесса. Обычно оболочка информирует пользователя о том, что задание блокировано и не может продолжать работу. Для того чтобы не помешать работе интерактивного задания, соответствующее сообщение показывается оболочкой перед выводом командной строки.

Если режим отслеживания включен, каждое завершение фонового задания приводит к запуску процедуры обработки сигнала **CHLD**.

При попытке выхода из оболочки, в которой запущены фоновые задания, с помощью команды `exit` или клавиши Ctrl-D система показывает предупреждающее сообщение `Есть остановленные (работающие) задания.` Для просмотра списка этих заданий введите команду **jobs**. При немедленной повторной попытке выхода оболочка завершит все приостановленные и работающие задания без предупреждения.

Обработка сигналов:

Если после команды введен символ & и опция отслеживания заданий (**monitor**) выключена, сигналы **SIGINT** и **SIGQUIT** в запущенных командах игнорируются. В противном случае, значения этих сигналов наследуются от родительского процесса.

Если в то время, когда оболочка ожидает завершения интерактивной команды, был принят сигнал, для которого есть процедура обработки, эта процедура будет вызвана только после завершения команды. Таким образом, процедура обработки сигнала **CHILD** не будет выполнена до завершения интерактивного задания.

Встроенный редактор в оболочке Korn (POSIX):

Обычно пользователь вводит команды с клавиатуры терминала и завершает их символами новой строки (**RETURN** или **LINE FEED**). Если вы включите опцию встроенного редактора emacs, gmacs или vi, то сможете пользоваться функциями редактирования командной строки.

Для того чтобы включить встроенный редактор, нужно выполнить одну из следующих команд:

Элемент	Описание
set -o emacs	Включает режим редактирования emacs и запускает встроенный редактор, имитирующий emacs.
set -o gmacs	Включает режим редактирования emacs и запускает встроенный редактор, имитирующий gmacs.
set -o vi	Включает режим редактирования vi и запускает встроенный редактор vi.

Опция редактирования автоматически выбирается всякий раз, когда переменной *VISUAL* или *EDITOR* присваивается значение, заканчивающееся на имя любой из этих опций.

Примечание: Для работы со встроенным редактором ваш терминал должен интерпретировать символ **RETURN** как символ возврата каретки без символа новой строки. Пробел должен заменять символ в текущей позиции курсора.

При переходе в режим редактирования открывается новое окно в текущей строке. Ширина этого окна равна значению переменной *COLUMNS*, если она определена, или 80 символам. Если длина строки превышает ширину окна за вычетом двух символов, то в конце строки будет показан специальный маркер, указывающий, что строка поместилась на экране не полностью. Когда курсор будет достигать границы экрана, окно будет центрироваться относительно текущей позиции курсора. Применяются следующие маркеры:

Элемент	Описание
>	Указывает, что строка показана не до конца.
<	Указывает, что строка показана не с начала.
*	Указывает, что строка показана не с начала и не до конца.

Действие команд поиска в обоих режимах редактирования распространяется на все содержимое файла хронологии Korn. Поиск выполняется по строкам. Если перед искомым значением указан символ ^, то будут найдены только строки, начинающиеся с этого значения.

Понятия, связанные с данным:

“Команды оболочки Korn (POSIX)” на стр. 250
Оболочка Korn - это интерактивный интерпретатор и командный язык программирования. Она разработана в соответствии с Международным стандартом интерфейса переносимых операционных систем (POSIX).

Режим редактирования emacs:

Режим редактирования emacs применяется встроенными редакторами **emacs** и **gmacs**. Единственное различие между этими встроенными редакторами заключается в способе обработки клавиш Ctrl-T.

Для редактирования строки нужно поместить курсор в нужную позицию и вставить или удалить требуемые символы или слова. Все команды редактирования представляют собой управляющие символы или escape-последовательности.

Команды редактирования действуют в любой позиции курсора (не только в начале строки). Если не указано иное, после ввода команды не нужно нажимать ни клавишу Enter, ни клавишу перевода строки (стрелку вниз).

Элемент	Описание
Ctrl-F	Перемещает курсор на один символ вперед (вправо).
Esc-F	Перемещает курсор на одно слово вперед. Словом считается произвольная последовательность символов, в которую входят только буквы, цифры и знаки подчеркивания.
Ctrl-B	Перемещает курсор на один символ назад (влево).
Esc-B	Перемещает курсор на одно слово назад.
Ctrl-A	Перемещает курсор в начало строки.
Ctrl-E	Перемещает курсор в конец строки.
Ctrl-] *c*	Перемещает курсор на указанный символ в текущей строке.
Esc-Ctrl-] *c*	Перемещает курсор назад до указанного символа в текущей строке.
Ctrl-X Ctrl-X	Меняет местами символы в помеченной позиции и в текущей позиции курсора.
ERASE	Удаляет предыдущий символ. (Пользовательский символ удаления, заданный с помощью команды **stty**; как правило соответствует клавишам Ctrl-H.)
Ctrl-D	Удаляет текущий символ.
Esc-D	Удаляет текущее слово.
Esc-Backspace	Удаляет предыдущее слово.
Esc-H	Удаляет предыдущее слово.
Esc-Delete	Удаляет предыдущее слово. Если символ прерывания вводится при нажатии клавиши Delete, то эта команда работать не будет.
Ctrl-T	Меняет местами текущий и следующий символы в режиме emacs. Меняет местами два предыдущих символа в режиме gmacs.
Ctrl-C	Если текущий символ - строчная буква, заменяет его на соответствующую прописную букву.
Esc-C	Заменяет в текущем слове все строчные буквы на прописные.
Esc-L	Заменяет в текущем слове все прописные буквы на строчные.
Ctrl-K	Удаляет все символы с текущей позиции курсора до конца строки. Если перед этой командой будет указано число, не превышающее текущую позицию курсора, то будут удалены все символы, начиная с указанной позиции и вплоть до позиции курсора. Если перед этой командой будет указано число, превышающее текущую позицию курсора, то будут удалены все символы, начиная с текущей и заканчивая заданной позицией курсора.
Ctrl-W	Удаляет символы с текущей позиции курсора до метки.
Esc-P	Заносит в стек символы с текущей позиции курсора до метки.

Элемент	Описание
KILL	Пользовательский символ KILL, определенный с помощью команды **stty**; как правило, соответствует клавишам Ctrl-G или символу @. Удаляет текущую строку. Если будет последовательно указано несколько символов KILL, то все символы KILL, кроме первого, будут заменены на символы новой строки (эту функцию удобно применять при работе с печатающими устройствами).
Ctrl-Y	Восстанавливает последний символ или слово, удаленные из строки. (Возвращает удаленное.)
Ctrl-L	Вводит символ новой строки и выдает текущую строку.
Ctrl-@	(Пустой символ) Маркирует текущую позицию.
Esc-пробел	Маркирует текущую позицию.
Ctrl-J	(Символ новой строки) Выполняет текущую строку.
Ctrl-M	(Возврат каретки) Выполняет текущую строку.
EOF	Символ конца файла (обычно соответствует клавишам Ctrl-D) интерпретируется как конец файла только в случае, если текущая строка пустая.
Ctrl-P	Выдает предыдущую строку списка хронологии (предыдущую команду). Последовательно нажимая клавиши Ctrl-P, можно пролистать список хронологии назад до нужной команды. Команды, состоящие из нескольких строк, выдаются по одной строке за одно нажатие клавиш Ctrl-P.
Esc-<	Выдает первую строку списка хронологии (первую из сохраненных команд).
Esc->	Выдает последнюю строку списка хронологии (последнюю выполненную команду).
Ctrl-N	Выдает следующую строку списка хронологии. Последовательно нажимая клавиши Ctrl-N, можно пролистать список хронологии вперед до нужной команды.
Ctrl-R *строка*	Просматривает файл хронологии от текущей позиции к началу и пытается найти строку, в которой содержится указанная **подстрока**. Если будет указано значение 0, то поиск будет выполнен по направлению к концу файла. Подстрока должна заканчиваться символом Enter или символом новой строки. Если перед строкой будет указан символ (^), то искомая строка должна начинаться с указанной **подстроки**. Если параметр **подстрока** не будет указан, то будет найдена следующая строка с последней указанной **подстрокой**. В этом случае значение 0 изменяет направление поиска.
Ctrl-O	Выполняет текущую строку и выдает следующую строку из файла хронологии.
Esc *цифры*	Задает числовой параметр. Указанные цифры будут переданы в качестве параметра в следующую команду. Числовые параметры допустимы для следующих команд: **Ctrl-F**, **Ctrl-B**, **ERASE**, **Ctrl-C**, **Ctrl-D**, **Ctrl-K**, **Ctrl-R**, **Ctrl-P**, **Ctrl-N**, **Ctrl-]**, **Esc-.**, **Esc-Ctrl-]**, **Esc-_**, **Esc-B**, **Esc-C**, **Esc-D**, **Esc-F**,**Esc-H**, **Esc-L** и **Esc-Ctrl-H**.
Esc *буква*	Пытается найти псевдоним *_буква* в списке псевдонимов. Если такой псевдоним определен, то в очередь ввода будет помещено его значение. Параметр *буква* не должен соответствовать ни одной из escape-функций.
Esc-[*буква*	Пытается найти псевдоним *__буква* (два знака подчеркивания и указанная буква) в списке псевдонимов. Если такой псевдоним определен, то в очередь ввода будет помещено его значение. В большинстве терминалов эту команду можно применять для программирования функциональных клавиш.
Esc-.	Вставляет в текущую строку последнее слово из предыдущей команды. Если перед этой командой будет задан параметр-число, то будет вставлено слово с указанным номером.
Esc-_	Равносильна команде **Esc-**. .
Esc-*	Выполняет подстановку имен файлов для текущего слова. Если слово не соответствует ни одному файлу или содержит специальные символы шаблона, то к нему будет добавлена звездочка (*).
Esc-Esc	Дополнение имени файла. Находит все файлы, соответствующие маске, полученной путем добавления звездочки к текущему слову. Выдает общий префикс имен этих файлов. Если маске соответствует только один объект, то к его имени добавляется либо символ /, если это каталог, либо пробел, если это файл.
Esc-=	Выдает список файлов, соответствующих маске, полученной путем добавления звездочки (*) к текущему слову.
Ctrl-U	Умножает параметр следующей команды на 4.

Элемент	Описание
\	Игнорирует следующий специальный символ, считая его обычным символом. С помощью этой функции можно ввести в командной строке или в строке поиска такие символы, как **ERASE**, **KILL** и **INTERRUPT** (обычно соответствует клавише Delete), а также символы, соответствующие функциям редактирования \). Обратная косая черта аннулируют функции редактирования следующего введенного символа.
Ctrl-V	Показывает версию оболочки.
Esc-#	Вставляет символ # в начало строки и выполняет текущую строку. Эта функция позволяет вставить комментарий в файл хронологии.

Режим редактирования vi:

В режиме редактирования vi предусмотрено два состояния командной строки.

Имеются следующие режимы:

- **Режим ввода текста.** При вводе команд редактор vi находится в режиме ввода текста.
- **Режим управления.** Для перехода в режим управления нажмите клавишу Esc.

Перед большинством команд можно указывать необязательный параметр **число**. В большинстве систем в режиме редактирования vi действуют стандартные правила обработки. Эхоповтор команды выдается в случае, если выполнено хотя бы одно из следующих условий:

- Быстродействие линии - 1200 бод или выше.
- В команде есть хотя бы один управляющий символ.
- С момента выдачи приглашения до приема команды прошло менее одной секунды.

Символ Esc отменяет обработку оставшейся части команды и позволяет редактировать командную строку. Данный способ обладает рядом преимуществ перед стандартной обработкой с опережающим эхоповтором линейного режима. Если будет дополнительно установлена опция **viraw**, то стандартная обработка будет отключена всегда. Этот режим по умолчанию включается для систем, не поддерживающих два различных символа конца строки, и его удобно применять с некоторыми терминалами.

Команды редактирования vi разбиты на категории. Эти категории описаны ниже:

Команды редактирования текста:

Ниже описаны команды редактирования текста для оболочки Korn.

Примечание: По умолчанию редактор находится в режиме ввода текста.

Элемент	Описание
ERASE	Удаляет предыдущий символ. (Пользовательский символ Erase, определенный с помощью команды **stty**; обычно соответствует клавише Ctrl-H или #.)
Ctrl-W	Удаляет предыдущее слово (все символы правее последнего пробела).
Ctrl-D	Завершает работу оболочки.
Ctrl-V	Игнорирует следующий специальный символ, считая его обычным символом. С помощью этой функции можно ввести в командной строке или в строке поиска символы редактирования, например, **ERASE** или **KILL**. Клавиши Ctrl-V аннулируют функции редактирования следующего введенного символа.
\	Игнорирует следующий специальный символ **ERASE** или **KILL**.

Команды перемещения курсора:

Ниже описаны команды перемещения курсора для оболочки Korn.

Для перемещения курсора по тексту предназначены следующие команды:

Элемент	Описание
[*число*]**l**	Перемещает курсор на один символ вперед (вправо).
[*число*]**w**	Перемещает курсор на одно слово вправо (слово - последовательность символов, в которую входят только буквы и цифры).
[*число*]**W**	Перемещает курсор на начало следующего "длинного слова" (первый символ за следующим пробелом).
[*число*]**e**	Перемещает курсор в конец текущего слова.
[*число*]**E**	Перемещает курсор в конец текущего "длинного слова" (последний символ перед следующим пробелом).
[*число*]**h**	Перемещает курсор на один символ назад (влево).
[*число*]**b**	Перемещает курсор на одно слово назад.
[*число*]**B**	Перемещает курсор в начало текущего "длинного слова" (первый символ после предыдущего пробела).
[*число*]**|**	Перемещает курсор в столбец, указанный параметром *число*. Если число не задано, то перемещает курсор в первый столбец.
[*число*]**f** *c*	Находит следующее вхождение символа *c* в текущей строке.
[*число*]**F***c*	Находит предыдущее вхождение символа *c* в текущей строке.
[*число*]**t***c*	Равносильна команде **f h**.
[*число*]**T***c*	Равносильна команде **F l**.
[*число*]**;**	Повторяет последнюю команду поиска символа - **f**, **F**, **t** или **T** - *число* раз.
[*число*]**,**	Повторяет в обратном направлении последнюю команду поиска символа указанное в параметре *число* раз.
0	Перемещает курсор в начало строки.
^	Перемещает курсор на первый символ строки, отличный от пробела.
$	Перемещает курсор в конец строки.

Команды поиска:

Команды поиска применяют хронологию команд, как указано ниже:

Элемент	Описание
[*число*]**k**	Выдает предыдущую строку списка хронологии (предыдущую команду).
[*число*]**-**	Равносильна команде **k**.
[*число*]**j**	Выдает следующую команду. С помощью клавиши **j** можно последовательно пролистать содержимое файла хронологии до нужной строки.
[*число*]**+**	Равносильна команде **j**.
[*число*]**G**	Выдает команду, номер которой задан параметром *число*. По умолчанию выдается последняя выполненная команда.
/*строка*	Выдает последнюю команду, в которой содержалась указанная строка. Строка должна заканчиваться символом **RETURN** или символом новой строки. Если перед строкой будет указан символ (^), то искомая строка должна начинаться с указанной *подстроки*. Если в этой команде строка не указана, то будет выполнен повторный поиск последней указанной строки.
?*строка*	То же, что /*строка*, только поиск выполняется в направлении к концу файла хронологии.
n	Находит следующее вхождение шаблона, указанного в команде /*строка* или **?** .
N	Находит следующее вхождение шаблона, указанного в команде /*строка* или **?** , но в обратном порядке. Выполняется поиск строки, указанной в последней команде /*строка*.

Команды изменения текста в командной строке:

Эти команды позволяют изменять текст в командной строке:

Элемент	Описание
a	Включает режим ввода текста; текст будет вводиться справа от текущего символа.
A	Включает режим ввода текста; текст будет вводиться в конце строки. Равносильна команде **$a**.
[*число*]**c***команда* **c**[*число*]*команда*	Удаляет символы от текущего до символа, на который будет перемещен курсор с помощью *команды* (исключая этот символ), а затем переходит в режим ввода текста. Если значение параметра *команда* - **c**, то будет удалена вся строка.
C	Удаляет символы от текущего до конца строки, а затем переходит в режим ввода текста. Равносильна команде **c$**.
S	Равносильная команде **cc**.
D	Удаляет символы от текущего до конца строки. Равносильна команде **d$**.

Элемент	Описание
[*число*]**d***команда* **d**[*число*]*команды*	Удаляет все символы от текущего до символа, на который будет перемещен курсор с помощью параметра *команда* (включая этот символ). Если значение параметра *команда* - **d**, то будет удалена вся строка.
i	Включает режим ввода текста; текст будет вводиться перед текущим символом.
I	Включает режим ввода текста; текст будет вводиться в начале строки. Равносильна команде **0i**.
[*число*]**P**	Вставляет перед текущим символом текст, полученный в результате выполнения последней команды изменения текста.
[*число*]**p**	Вставляет после текущего символа текст, полученный в результате выполнения последней команды изменения текста.
R	Переходит в режим ввода текста; вводимый текст будет заменять текущее содержимое строки.
[*число*]**r***c*	Заменяет *число* символов, начиная с текущей позиции курсора, на символ *с*. После замены курсор перемещается вправо за последний замененный символ.
[*число*]**x**	Удаляет текущий символ.
[*число*]**X**	Удаляет предыдущий символ.
[*число*]**.**	Повторяет предыдущую команду изменения текста.
[*число*]~	Заменяет строчные буквы на прописные, а прописные - на строчные. Замена выполняется для указанного *числа* символов, начиная с текущей позиции курсора.
[*число*]_	Добавляет слово, указанное в параметре *число* предыдущей команды, и переходит в режим ввода текста. Если параметр *число* не указан, то добавляется последнее указанное слово.
*	Добавляет звездочку (*) к текущему слову и пытается выполнить подстановку имен файлов. Если подстановка невозможна (ни одно имя файла не соответствует указанному шаблону), то будет выдан звуковой сигнал. В противном случае текущее слово будет заменено на список файлов, соответствующих шаблону, и будет включен режим ввода текста.
\	Дополнение имени файла. Добавляет звездочку (*) к текущему слову и находит все файлы, соответствующие полученному шаблону. После этого текущее слово заменяется на максимальный общий префикс имен найденных файлов. Если маске соответствует только один объект, к его имени добавляется символ косой черты (/), если это каталог, или пробел, если это файл.

Различные команды редактирования:

В этом разделе описываются часто используемые команды редактирования.

Элемент	Описание
[*число*]**y***команда*	
y[*число*]*команда*	Копирует в буфер удаления все символы от текущего до символа, на который будет перемещен курсор в результате выполнения команды *команда*. Текст и позиция курсора не изменяются.
Y	Копирует в буфер обмена все символы с текущей позиции курсора до конца строки. Равносильна команде **y$**.
u	Отменяет последнюю команду изменения текста.
U	Отменяет все команды изменения текста, выполненные в текущей строке.
[*число*]**v**	Отправляет команду `fc -e ${VISUAL:-${EDITOR:-vi}}` *число* в буфер ввода. Если параметр *число* не указан, то редактор будет вызван для текущей строки.
Ctrl-L	Вводит символ новой строки и выдает текущую строку. Эта команда работает только в режиме управления.
Ctrl-J	(Новая строка) Выполняет текущую строку. Эта команда действует во всех режимах.
Ctrl-M	(Возврат каретки) Выполняет текущую строку. Эта команда действует во всех режимах.
#	Добавляет в начало текущей строки символ # и выдает полученную строку. С помощью этой команды можно добавить текущую строку в файл хронологии, не выполняя ее. Если в командной строке есть символ конвейера, точка с запятой или символ новой строки, то после каждого из этих символов будут вставлены дополнительные символы #. Для того чтобы удалить все символы комментария, нужно скопировать эту команду из файла хронологии в командную строку, а затем нажать клавишу #.
=	Показывает список файлов, соответствующих шаблону, полученному путем добавления звездочки к текущему слову.
@*буква*	Пытается найти псевдоним *_буква*. Если такой псевдоним определен, то в очередь ввода будет помещено его значение.

Команды оболочки Korn (POSIX):

Оболочка Korn - это интерактивный интерпретатор и командный язык программирования. Она разработана в соответствии с Международным стандартом интерфейса переносимых операционных систем (POSIX).

POSIX - это не операционная система, а *стандарт*, разработанный для обеспечения переносимости приложений на уровне исходного текста. Функции POSIX реализованы в оболочке Korn. В оболочке Korn (ее также называют оболочкой POSIX) поддерживается большинство функций оболочек Bourne и C, а также возможности по перенаправлению ввода-вывода, подстановке значений переменных и имен файлов. Кроме того, в ней предусмотрено несколько дополнительных команд и средств программирования:

Примечание: Также доступна ограниченная версия оболочки Korn, имеющая название **rksh**. Дополнительные сведения приведены в описании команды **rksh**.

Элемент	Описание
Вычисление арифметических выражений	Оболочка Korn (POSIX) позволяет вычислять целочисленные арифметические выражения с помощью встроенной команды **let**. В выражениях могут применяться любые системы счисления от 2 до 36. Для того чтобы включить в оболочке Korn распознавание чисел, начинающихся с 0 (восьмеричное) и 0x (шестнадцатеричное), выполните следующие команды: **export XPG_SUS_ENV=ON** Экспортирование переменной XPG_SUS_ENV делают выполняемые команды и используемые ими библиотеки полностью совместимыми с POSIX. **Примечание:** Так как вся система библиотек становится совместимой с POSIX, алгоритм поведения определенных команд может измениться. **export OCTAL_CONST=ON** Экспортирование этой переменной делает обработку констант, объявленных в оболочке Korn, совместимой с POSIX, когда речь идет о распознавании восьмеричных или шестнадцатеричных констант.
Хронология команд	Оболочка Korn (POSIX) сохраняет текст всех введенных команд в специальном файле. С помощью текстового редактора можно изменять содержимое этого файла хронологии и повторно выполнять нужные команды.
Создание подчиненных процессов	Позволяет запускать программы в фоновом режиме и обмениваться с ними данными.
Редактирование	В оболочке Korn (POSIX) предусмотрены встроенные функции редактирования командной строки. Поддерживается эмуляция редакторов emacs, gmacs и vi.

Каждую команду оболочки Korn можно отнести к одному из следующих типов:

- Простая команда
- Конвейер
- Список
- Составная команда
- Функция

При вызове команды оболочка Korn (POSIX) анализирует ее и выполняет следующие действия:

- Выполняет все необходимые подстановки.
- Определяет, есть ли в команде символ /. Если он есть, оболочка выполняет программу, расположенную в указанном каталоге.

Если в команде нет символа /, оболочка Korn (POSIX) выполняет следующие действия:

- Определяет, является ли команда особой встроенной командой. Если да, то команда выполняется в текущем процессе оболочки.
- Сравнивает команду с пользовательскими функциями. Если команда совпадает с одной из них, то оболочка сохраняет позиционные параметры, а затем анализирует аргументы вызова *функции*. После завершения функции (или выполнения в ней команды return) оболочка восстанавливает позиционные параметры и выполняет прерывание EXIT для данной функции, если она определена. Значением функции будет код завершения последней выполненной команды. Функция выполняется в текущем процессе оболочки.
- Если имя команды совпадает с именем одной из обычных встроенных команд, то выполняется эта обычная встроенная команда.
- Создает процесс и пытается выполнить команду с помощью команды **exec** (если указанная команда не является встроенной командой или пользовательской функцией).

Оболочка Korn (POSIX) пытается найти выполняемый файл во всех каталогах, указанных в переменной *PATH*. Имена каталогов в этой переменной разделяются символом : (двоеточие). По умолчанию значение этой переменной - /usr/bin: (каталог /usr/bin, затем текущий каталог). Текущий каталог в переменной PATH обозначается двумя символами двоеточия, если он находится в середине списка, или одним двоеточием, если он находится в начале или конце списка.

Если указанный файл не является каталогом, его имя отлично от `a.out`, и при этом разрешено его выполнение, то оболочка предполагает, что в нем содержатся команды оболочки. Для чтения такого файла текущий процесс оболочки создает подоболочку. Все неэкспортируемые псевдонимы, функции и именованные параметры не передаются в файл при его выполнении. Если для файла с командами оболочки разрешено *чтение* или установлен один из битов **setuid** и **setgid**, то оболочка запускает специальный агент, который устанавливает необходимые права доступа, запускает копию оболочки и передает ей на вход данный файл. Если команда указана в скобках, то при ее выполнении из среды не удаляются неэкспортируемые объекты (псевдонимы и т.п.)

Понятия, связанные с данным:

“Доступные оболочки” на стр. 207
Ниже описаны оболочки, поставляемые с AIX.

“Создание подчиненных процессов” на стр. 232
Оболочка Korn (POSIX) позволяет запустить несколько параллельных фоновых процессов. Соответствующие команды, которые запускаются из сценария оболочки, называются *параллельными процессами*.

“Встроенный редактор в оболочке Korn (POSIX)” на стр. 244
Обычно пользователь вводит команды с клавиатуры терминала и завершает их символами новой строки (**RETURN** или **LINE FEED**). Если вы включите опцию встроенного редактора emacs, gmacs или vi, то сможете пользоваться функциями редактирования командной строки.

“Арифметические операции в оболочке Korn (POSIX)” на стр. 216
Стандартная встроенная команда **let** оболочки Korn (POSIX) позволяет выполнять арифметические операции над целыми числами.

“Встроенные команды оболочки Korn или POSIX” на стр. 233
В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

Составные команды оболочки Korn:

Составная команда представляет собой список простых команд, конвейер или команду, начинающуюся с зарезервированного слова. Как правило, составные команды (например, **if**, **while** или **for**) применяются в сценариях оболочки.

Ниже приведен список составных команд оболочки Korn (POSIX):

Формат команды	Описание
for *Идентификатор* [**in** *Слово* ...] **;do** *Список* **;done**	При каждом выполнении команды **for** параметру **идентификатор** присваивается следующее слово из списка **in** *слово*... Если параметр **in** *слово* ... не указан, то команда **for** выполняет команды, указанные в параметре **do** *команды*, один раз для каждого определенного позиционного параметра. Выполнение завершается после обработки последнего слова в списке или последнего позиционного параметра.
select *Идентификатор* [**in** *Слово* ...] **;do** *Список* **;done**	Команда **select** выдает в stderr (файл с дескриптором 2) указанные слова с порядковым номером перед каждым из них. Если параметр **in** *слово*... не указан, то выдается список позиционных параметров. Выдается приглашение **PS3**, после чего из стандартного ввода считывается одна строка. Если в этой строке указан номер одного из перечисленных значений, то переменной *идентификатор* присваивается данное слово. Если из стандартного ввода будет считана пустая строка, то будет повторно выдан список вариантов. В противном случае параметру *идентификатор* будет присвоено пустое слово. Строка, считанная из стандартного ввода, сохраняется в параметре **REPLY**. Параметр *команды* выполняется для каждого выбранного варианта до тех пор, пока не будет нажата клавиша прерывания или не встретится символ конца файла.
case *слово* **in** [[**(**] *шаблон* [| *шаблон*] ... **)** *список* **;;**] ... **esac**	Команда **case** выполняет параметр *команды* для первого параметра *шаблон*, соответствующего параметру *слово*. Шаблоны должны быть указаны в формате, который применяется при подстановке имен файлов.

Формат команды	Описание
if *список* **;then***список* [**elif***список* **;then***список*] ... [**;else***список*] **;fi**	Параметр *команды* задает список выполняемых команд. Сначала оболочка выполняет команды, указанные в параметре **if** *команды*. Если в результате будет получен нулевой код завершения, то будут выполнены команды, указанные в параметре **then** *команды*. В противном случае будут выполнены команды, указанные в параметре *список* команды **elif**. Если последняя команда в списке **elif** *команды* завершится с кодом 0, то будут выполнены команды из соответствующего блока **then** *команды*. Если код завершения последней команды из параметра **then** *команды* будет равен нулю, то будут выполнены команды из параметра **else** *команды*. Если блок *команды* не будет выполнен ни для одного параметра **else** и **then**, то команда **if** завершится с кодом 0.
while *список* **;do***список* **;done until** *список* **;do***список* **;done**	Параметр *команды* задает список выполняемых команд. Команда **while** выполняет в цикле команды, указанные в параметрах *команды*. Если код завршения последней команды в списке **while** *команды* равен нулю, то выполняются команды из списка **do** *команды*. Если код завершения последней команды в списке **while** *команды* отличен от нуля, то выполнение цикла завершается. Если команды из списка **do** *команды* не были выполнены ни разу, то команда **while** завершается с кодом 0. Команда **until** применяется вместо команды **while** в случаях, когда нужно изменить условие выхода из цикла на противоположное.
(*Список***)**	Параметр *команды* задает список выполняемых команд. Оболочка выполнит *команды* в отдельной среде. **Примечание:** Если необходимо указать две открывающие скобки подряд (для вложенного выполнения команд), то между ними должен быть указан пробел, иначе они будут интерпретированы как арифметическое выражение.
{ *Список***; }**	Параметр *команды* задает список выполняемых команд. Оболочка выполнит указанные *команды.* **Примечание:** В отличие от команды ( ), в команде { } распознаются зарезервированные символы. Зарезервированные символы должны быть указаны в начале строки или после символа **;**.
[[*выражение*]]	Вычисляет значение параметра *выражение*. Если оно истинно, команда завершается с кодом 0.
function *идентификатор* { *список* **;}** или **function** *идентификатор* **()** {*список* ;}	Определяет функцию с именем, указанным в параметре *идентификатор*. Тело функции - список команд, заключенный в символы { }. Символы ( ) представляют собой два отдельных оператора, поэтому между параметром *идентификатор* и символами (и) можно указывать пробелы, хотя это не обязательно.
time *конвейер*	Выполняет команды, указанные в параметре *конвейер*. В stderr выдается информация о том, сколько времени было затрачено на их выполнение (отдельные значения для полного, системного и пользовательского времени).

Понятия, связанные с данным:

“Параметры в оболочке Korn” на стр. 226
Ниже описаны параметры оболочки Korn.

Запуск оболочки:

Оболочку Korn можно запустить с помощью команды **ksh**, **psh** (оболочка POSIX) или **exec**.

Если оболочка будет запущена с помощью команды **exec** и первым символом аргумента 0 (**$0**) будет дефис (**-**), то оболочка будет считаться оболочкой входа в систему. В этом случае сначала будут выполнены команды из файла `/etc/profile`, а затем команды либо из файла `.profile` в текущем каталоге, либо из файла `$HOME/.profile`, если хотя бы один из них существует. Затем оболочка выполнит команды из файла, указанного в переменной среды **ENV**, если он существует.

Если при запуске оболочки Korn (POSIX) будет указан параметр *файл* [*параметр*], то оболочка выполнит файл сценария, указанный в параметре *файл*, со всеми его параметрами. Файл сценария должен быть доступен для чтения; все значения **setuid** и **setgid** игнорируются. Затем оболочка выполнит команды.

Примечание: При вызове оболочки Korn (POSIX) не указывайте файл сценария с флагами **-c** и **-s**.

За дополнительной информацией о позиционных параметрах обратитесь к разделу “Параметры в оболочке Korn” на стр. 226.

Понятия, связанные с данным:

“Параметры в оболочке Korn” на стр. 226
Ниже описаны параметры оболочки Korn.

Среда оболочки Korn:

Совокупность всех переменных (вместе с их значениями), известных команде на момент начала ее выполнения, образует так называемую *среду*.

В эту среду входят как переменные, которые команда наследует от своего родительского процесса, так и переменные, указанные в виде ключевых слов в командной строке. Существует несколько способов взаимодействия оболочки и среды. При запуске оболочка просматривает среду, создает для каждого найденного имени параметр, которому она присваивает соответствующее значение, и помечает его для экспорта. Команды, выполняемые из оболочки, полностью наследуют среду.

Если вы измените или создадите новые параметры оболочки с помощью команды **export** или **typeset -x**, то измененные (новые) параметры станут частью среды. Соответственно, среда каждой выполняемой команды состоит из всех переменных, первоначально унаследованных оболочкой (в дальнейшем они могут быть изменены), с учетом изменений, внесенных с помощью команд **export** и **typeset -x**. Команда (подоболочка) может изменять значения переменных, но для того, чтобы эти изменения остались в силе для ее дочерних оболочек и процессов, все измененные переменные должны быть экспортированы.

Для того чтобы изменить среду конкретной команды или функции, можно задать необходимые изменения в командной строке непосредственно перед именем этой команды или функции. Эти изменения должны быть заданы в форме *Идентификатор=значение*. Поэтому для выполняемой команды следующие выражения равносильны:

```
TERM=450 команда аргументы
(export TERM; TERM=450; команда аргументы)
```

Функции оболочки Korn:

Зарезервированное слово **function** предназначено для определения функций оболочки. Оболочка считывает определение функции и сохраняет его в текущей среде. При разборе определений функций разворачиваются псевдонимы. Оболочка выполняет функции точно так же, как команды, и передает в них аргументы как позиционные параметры.

Оболочка Korn (POSIX) выполняет функции в среде, из которой они вызываются. Следующие объекты являются общими для функции и сценария, из которого она вызвана, и их изменение приводит к побочным эффектам:

- Атрибуты и значения переменных (если для объявления локальной переменной внутри функции не применяется команда **typeset**)
- Рабочий каталог
- Псевдонимы, определения функций и атрибуты
- Особый параметр $
- Открытые файлы

Следующие объекты функции не связаны с соответствующими объектами сценария, из которого была вызвана функция, и их изменение не приводит к побочным эффектам:

- Позиционные параметры
- Особый параметр #
- Переменные в списке значений переменных, заданном при вызове функции

- Переменные, объявленные с помощью команды **typeset** внутри функции
- Опции
- Прерывания. (Учтите, что сигналы, игнорируемые сценарием, всегда игнорируются и вызываемыми из него функциями).

Примечание: В ранних версиях оболочки Korn все прерывания, за исключением **EXIT** и **ERR**, были общими для функции и сценария, из которого она была вызвана.

Если прерывание **0** или **EXIT** сработает *внутри* тела функции, то соответствующее действие будет выполнено после завершения функции в среде, из которой она была вызвана. Если прерывание сработает *вне* тела функции, соответствующее действие будет выполнено после выхода из оболочки Korn. В ранних версиях оболочки Korn прерывания **0** и **EXIT**, сработавшие вне тела функции, не выполнялись после завершения работы оболочки.

При выполнении функции действуют все правила присвоения переменных и обработки синтаксических ошибок, описанные во встроенных командах оболочки Korn или оболочки POSIX.

Если имя функции будет указано в качестве простой команды, то фактически будет выполнена составная команда. Операнды этой простой команды на время выполнения составной команды станут ее позиционными параметрами. Кроме того, значение особого параметра # будет изменено в соответствии с числом операндов. Особый параметр `0` остается без изменения.

Для возврата управления из тела функции применяется особая команда **return**. Кроме того, управление автоматически возвращается в вызывающий сценарий в случае ошибки в теле функции.

Список определенных функций можно просмотреть с помощью команды **typeset** с флагом **-f** или **+f**. Если будет указана опция **-f**, то помимо имен будет показан текст функций. Для того чтобы аннулировать определение функции, нужно выполнить особую команду **unset** с флагом **-f**.

По умолчанию определения функций аннулируются при выполнении сценариев оболочки. Флаг **-xf** особой команды **typeset** позволяет экспортировать функции в сценарии, выполняемые в текущем экземпляре оболочки. Если какая-либо функция должна действовать в различных экземплярах оболочки, ее определение нужно указать в файле `ENV` с опцией **-xf** команды **typeset**.

При разборе определения функции код завершения 0 устанавливается в случае, если оболочке не удалось объявить функцию. В противном случае код завершения будет больше нуля. Код завершения функции равен коду завершения последней выполненной в ней команды.

Понятия, связанные с данным:

“Параметры в оболочке Korn” на стр. 226
Ниже описаны параметры оболочки Korn.

“Встроенные команды оболочки Korn или POSIX” на стр. 233
В оболочку Korn, или POSIX, встроен ряд команд, которые можно выполнять в процессах этой оболочки.

Хронология команд оболочки Korn (POSIX):

Оболочка Korn (POSIX) сохраняет текст команд, введенных с терминала, в файле хронологии.

С помощью переменной *HISTFILE* можно задать имя этого файла. Если переменная *HISTFILE* не определена или в указанный файл запрещена запись, то применяется файл хронологии `$HOME/.sh_history`. Если файл хронологии не существует и оболочка Korn не может создать его, или если он существует, но оболочке Korn не разрешена запись в него, то будет применяться временный файл хронологии. Все экземпляры оболочки пользуются одним и тем же файлом хронологии.

По умолчанию оболочка Korn (POSIX) сохраняет текст последних 512 команд. Размер файла хронологии (указанный в переменной *HISTSIZE*) не ограничен, но если он будет очень большим, то время запуска оболочки Korn может значительно возрасти.

Подстановка хронологи команд:

С помощью встроенной команды **fc** можно просматривать и изменять содержимое файла хронологии. Для выбора части файла или редактирования строки укажите номер или начальные символы команды.

Вы можете указать как одну, так и несколько команд.

Если в команде **fc** не задан редактор, то будет применяться редактор, указанный в переменной *FCEDIT*. Если переменная *FCEDIT* не задана, то применяется редактор `/usr/bin/ed`. По окончании работы с редактором измененные команды будут напечатаны и запущены.

Для того чтобы повторить последнюю команду, не запуская редактор, нужно в качестве редактора указать дефис (`-`). В этом случае можно изменить выполняемую команду с помощью параметра `старая строка=новая строка`. Например, если `r` - псевдоним команды `fc -e -`, то команда `r bad=good c` повторит последнюю введенную команду, начинающуюся с символа `c`, заменив первое вхождение строки `bad` на строку `good`.

Задачи, связанные с данной:

“Просмотр введенных ранее команд с помощью команды (history)” на стр. 131
Команда **history** применяется для просмотра команд, введенных ранее.

Псевдонимы команд в оболочке Korn (POSIX):

Оболочка Korn (другое название - POSIX) позволяет создавать псевдонимы команд.

Команда **alias** определяет псевдоним с помощью оператора `Имя=Строка`. Если в качестве первого слова команды указан псевдоним, то оболочка Korn проверяет, не обрабатывается ли уже псевдоним с тем же именем. Если да, оболочка Korn не заменяет псевдоним. Если нет, то оболочка Korn заменяет псевдоним на его значение.

Первым символом имени псевдонима может быть любой печатаемый символ, за исключением метасимволов. Все остальные символы должны быть такими же, как в допустимом идентификаторе. Строка замещения может содержать любой допустимый текст оболочки, включая метасимволы.

Если последний символ значения псевдонима - пробел, то при подстановке псевдонима оболочка будет проверять также следующее слово. С помощью псевдонимов вы можете переопределять встроенные команды, но не зарезервированные слова. Определения псевдонимов не наследуются при новом вызове **ksh**. Однако если вы укажете **alias -x**, то псевдоним будет действителен в сценариях, инициализированных по имени и не запускающих отдельную оболочку. Для того чтобы экспортировать определение псевдонима и предоставить к нему доступ дочерним процессам, укажите в файле переменных среды **alias -x** и определение псевдонима.

Для создания, просмотра и экспорта псевдонимов служит команда **alias**.

Для удаления псевдонимов применяется команда **unalias**.

Ниже приведен формат команды создания псевдонима:

```
alias Имя=Строка
```

где **`Имя`** - это имя псевдонима, а **`Строка`** - его значение.

Следующие экспортированные псевдонимы предопределены в оболочке Korn, но могут быть отменены или заменены. Рекомендуется не изменять эти псевдонимы, поскольку другие пользователи ожидают, что эти псевдонимы соответствуют определенным значениям.

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
nohup='nohup '
r='fc -e -'
true=':'
type='whence -v'
```

Псевдонимы не поддерживаются в оболочке Korn (**ksh**), если она была вызвана в неинтерактивном режиме, например, с помощью сценария оболочки или команды **ksh** с опцией **-c**:

```
ksh -c alias
```

Задачи, связанные с данной:

“Создание псевдонима команды (команда alias)” на стр. 134
Псевдоним позволяет создать удобное, короткое имя для команды, имени файла или произвольного текста оболочки. Применение псевдонимов экономит время при частом выполнении одних и тех же задач. Вы можете создать псевдоним команды.

Псевдонимы-следы:

Часто псевдонимы применяются для сокращения длины пути к файлу. Специальная опция позволяет автоматически задать значение псевдонима равным полному пути соответствующей команды. Такой специальный тип псевдонима называется псевдонимом-*следом*.

Следы ускоряют выполнение, устраняя необходимость в поиске полного пути в переменной *PATH*.

Команда **set -h** включает *отслеживание* команд, что означает, что при каждом обращении к команде оболочка создает соответствующий псевдоним-след. После сброса переменной *PATH* это значение становится неопределенным.

Отслеживание команд и создание следов продолжается, поэтому при следующем обращении к команде это значение вновь будет определено. Псевдонимы-следы компилируются в оболочке.

Замена тильды:

После того, как оболочка заменит псевдонимы на соответствующие значения, она проверит каждое слово, не начинается ли оно с тильды (~) без кавычек. При обнаружении такого слова оболочка сравнит символы в этом слове до первой косой черты (/) с именами пользователей в файле `/etc/passwd`. Если оболочка найдет соответствие, то она заменит тильду (~) и имя на начальный каталог соответствующего пользователя. Это процесс называется *заменой тильды*.

Если соответствие не будет найдено, то исходный текст останется без изменений. Кроме того, оболочка Korn выполняет специальные замены, если тильда (~) является единственным символом в слове или за ней стоит знак плюс (+) или дефис (-):

Элемент	Описание
~	Символ будет заменен на значение переменной *HOME*
~+	Символы будут заменены на значение переменной *$PWD* (полный путь к текущему каталогу)
~-	Символы будут заменены на значение переменной *$OLDPWD* (полный путь к предыдущему каталогу)

Кроме того, оболочка пытается заменить тильду (~), когда значение параметра присвоения переменной начинается с тильды.

Оболочка Bourne

Оболочка Bourne - это интерактивный интерпретатор команд и командный язык программирования.

Команда **bsh** запускает оболочку Bourne.

Оболочку Bourne можно запустить либо как начальную оболочку, либо как подоболочку начальной оболочки. Первый вариант запуска выполняется только командой **login**. Для этого предназначен специальный формат команды **bsh**: `-bsh`. Если оболочка вызывается со знаком дефис (`-`), то сначала она считывает и выполняет команды из системного файла `/etc/profile` и вашего профайла `$HOME/.profile`, если последний существует. Файл `/etc/profile` применяется для настройки переменных, необходимых всем пользователям. После этого оболочка будет готова к обработке команд, которые вы введете обычным образом.

Если в команде запуска оболочки Bourne указан параметр **Файл** [*Параметр*], то оболочка запустит файл сценария, указанный в параметре **Файл**, с учетом опций. Файл сценария должен быть доступен для чтения; все значения `setuid` и `setgid` игнорируются. Затем оболочка выполнит команды. Файл сценария не следует указывать вместе с флагом **-c** и **-s**.

Понятия, связанные с данным:

“Доступные оболочки” на стр. 207
Ниже описаны оболочки, поставляемые с AIX.

Среда оболочки Bourne:

Совокупность всех переменных (вместе с их значениями), известных команде на момент начала ее выполнения, образует так называемую *среду*. В эту среду входят как переменные, которые команда наследует от своего родительского процесса, так и переменные, указанные в виде ключевых слов в командной строке.

Оболочка передает дочерним процессам переменные в виде аргументов встроенной команды **export**. Эта команда помещает указанные переменные в среду как самой оболочки, так и всех ее будущих дочерних процессов.

Ключевые слова - это пары "переменная-значение", которые обычно указываются в виде оператора присвоения перед именем процедуры в командной строке (возможны и другие варианты - см., например, флаг команды **set**). Эти переменные задаются в среде выполнения вызываемой процедуры.

Примеры:

- Рассмотрим следующую процедуру, которая показывает значения двух переменных (процедура хранится в командном файле `key_command`):

  ```
  # key_command
  echo $a $b
  ```

  В командной строке будет показан следующий вывод:

  ```
  Ввод                              Вывод
  a=key1   b=key2  key_command      key1 key2
  a=tom    b=john  key_command      tom john
  ```

Ключевые слова процедуры не учитываются в счетчике параметров, хранящемся в $#.

Процедура может обращаться к значениям любых переменных своей среды. Но если она изменит любое из этих значений, то изменение не отразится на среде оболочки. Эти изменения отразятся только на среде рассматриваемой процедуры. Если вы хотите передать изменения в среде дочерним процессам процедуры, экспортируйте новые значения в этой процедуре.

Примеры:

- Для просмотра списка переменных, которые можно экспортировать из текущей оболочки, введите команду:

  ```
  export
  ```

- Для просмотра списка переменных, доступных в текущей оболочке только для чтения, введите команду:

  ```
  readonly
  ```

- Для просмотра списка пар "переменная-значение" в текущей среде введите команду:

  ```
  env
  ```

Дополнительная информация о пользовательских средах приведена в разделе “Файл /etc/environment” на стр. 315.

Подстановка команд в оболочке Bourne:

Обычно оболочка заменяет выражение $*Переменная* на строковое значение, присвоенное переменной *Переменная*, если оно есть. Однако вы можете выполнить и *условную подстановку*, т.е. подстановку в зависимости от того, задано значение переменной или нет, пустое оно или нет, или обоих условий.

По определению переменная считается заданной, если ей когда-либо присваивалось значение. Значением переменной может быть пустая строка, которую можно присвоить любым из следующих способов:

Элемент	Описание
`A=`	
`bcd=""`	
`Efg=''`	Присваивает пустую строку переменным `A`, `bcd` и `Efg`.
`set '' ""`	Задает первый и второй позиционные параметры равными пустой строке и сбрасывает все остальные позиционные параметры.

Ниже приведен список выражений, с помощью которых можно выполнять условную подстановку:

Элемент	Описание
${*Переменная***-***Строка***}**	Если переменная задана, то вместо этого выражения будет подставлено значение *Переменной*. Если нет, то выражение будет заменено на значение *Строка*.
${*Переменная***:-***Строка***}**	Если переменная задана и не пуста, то вместо этого выражения будет подставлено значение *Переменной*. Если нет, то выражение будет заменено на значение *Строка*.
${*Переменная***=***Строка***}**	Если переменная задана, то вместо этого выражения будет подставлено значение *Переменной*. Если нет, то *Переменной* будет присвоено значение *Строка*, после чего это значение будет подставлено вместо выражения. Присваивать таким образом значения позиционным параметрам нельзя.
${*Переменная***:=***Строка***}**	Если переменная задана и не пуста, то вместо этого выражения будет подставлено значение *Переменной*. Если нет, то *Переменной* будет присвоено значение *Строка*, после чего это значение будет подставлено вместо выражения. Присваивать таким образом значения позиционным параметрам нельзя.

Элемент	Описание

${*Переменная***?***Строка***}**	Если переменная задана, то вместо этого выражения будет подставлено значение *Переменной*. Если нет, то будет выдано следующее сообщение:
	`Переменная: Строка`
	
	Затем работа оболочки будет завершена (если оболочка не начальная). Если вы не укажете значение *Строка*, то появится следующее сообщение:
	`Переменная: параметр пуст или не задан`

${*Переменная***:?***Строка***}**	Если переменная задана и не пуста, то вместо этого выражения будет подставлено значение *Переменной*. Если нет, то будет выдано следующее сообщение:
	`Переменная : Строка`
	
	Затем работа оболочки будет завершена (если оболочка не начальная). Если вы не укажете значение *Строка*, то появится следующее сообщение:
	`Переменная: параметр пуст или не задан`

${*Переменная***+***Строка***}**	Если переменная задана, то вместо этого выражения будет подставлено значение *Строка*. Если нет, то будет подставлена пустая строка.
${*Переменная***:+***Строка***}**	Если переменная задана и не пуста, то вместо этого выражения будет подставлено значение *Строка*. Если нет, то будет подставлена пустая строка.

При условной подстановке оболочка не вычисляет значение *Строка* до тех пор, пока не воспользуется им в качестве подставленной строки. Таким образом, в следующем примере оболочка выполнит команду **pwd** только в том случае, если переменная `d` не задана или пуста:

```
echo ${d:-`pwd`}
```

Понятия, связанные с данным:

“Пользовательские переменные в оболочке Bourne” на стр. 270
Оболочка Bourne распознает алфавитно-цифровые переменные, которым могут быть присвоены строковые значения.

Позиционные параметры в оболочке Bourne:

При запуске процедуры оболочки она неявно создает по одному позиционному параметру для каждого слова, указанного в командной строке.

Слову в позиции 0 (имени процедуры) будет соответствовать параметр `$0`, следующему слову (первому параметру) - параметр `$1`, и т.д. до `$9`. Если в командной строке указано больше девяти параметров, воспользуйтесь встроенной командой **shift**.

Вы можете сбросить значения позиционных параметров явно, воспользовавшись встроенной командой **set**.

Примечание: Если аргумент в некоторой позиции не задан, то соответствующему позиционному параметру присваивается пустое значение. Позиционные параметры глобальны и могут быть переданы вложенным процедурам оболочки.

Понятия, связанные с данным:

“Пользовательские переменные в оболочке Bourne” на стр. 270
Оболочка Bourne распознает алфавитно-цифровые переменные, которым могут быть присвоены строковые значения.

Ссылки, связанные с данной:

“Предустановленные специальные переменные в оболочке Bourne” на стр. 273
Некоторые переменные в оболочке Bourne являются специальными. Следующие переменные задаются только в оболочке Bourne:

Подстановка имен файлов в оболочке Bourne:

Оболочка Bourne позволяет выполнить подстановку имен файлов.

В роли параметров команды часто выступают имена файлов. Вы можете автоматически создать список имен файлов, которые будут параметрами командной строки. Для этого укажите символ, распознаваемый оболочкой в качестве шаблона. Если команда содержит такой символ, оболочка заменит его на имена файлов.

Примечание: Оболочка Bourne не поддерживает подстановку имен файлов на основе классов эквивалентности символов.

Большинство символов в шаблоне будут соответствовать самим себе, однако вы можете указать и специальные символы подстановки. Эти символы перечислены ниже:

Элемент	Описание
*	Заменяет любую строку, в том числе пустую
?	Соответствует любому символу
[. . .]	Соответствует любому из заключенных в скобки символов
[! . . .]	Соответствует любому символу, *отличному от* тех, что указаны после восклицательного знака в квадратных скобках

Если в квадратных скобках указана пара символов, разделенных дефисом (-), то она задает множество всех промежуточных символов согласно двоичной последовательности упорядочения символов.

На соответствие шаблону накладываются определенные ограничения. Если имя файла начинается с точки (.), то он соответствует только тому шаблону, который также начинается с точки. Например, символ * соответствует именам файлов *myfile* и *yourfile*, но не соответствует именам *.myfile* и *.yourfile*. Для таких имен файлов нужно задать шаблон следующего вида:

```
.*file
```

Если нет ни одного имени файла, соответствующего шаблону, то в качестве результата поиска возвращается сам шаблон.

В именах файлов и каталогов не допускаются символы *, ?, [и], поскольку при разборе шаблона они могут вызвать бесконечную рекурсию (зацикливание).

Перенаправление ввода и вывода в оболочке Bourne:

Существуют опции перенаправления, используемые в командах.

Работа большинства команд не зависит от того, с чем связан стандартный ввод или вывод: с клавиатурой, экраном или файлом. Это позволяет применять команды как отдельно, так и в конвейере.

Ниже приведены опции перенаправления, которые применяются в простых командах. Эти символы могут быть указаны перед или после команды, однако они не передаются на вход команды.

Элемент	Описание
<*Файл*	Указанный файл будет применяться в качестве стандартного ввода.
>*Файл*	Указанный файл будет применяться в качестве стандартного вывода. Если файл не существует, то он будет создан; в противном случае все данные файла удаляются.
> >*Файл*	Указанный файл будет применяться в качестве стандартного вывода. Если файл не существует, то он будет создан; в противном случае вывод будет добавлен в конец файла.
<<[-]*eofstr*	В качестве стандартного ввода считываются все строки, указанные между первой последовательностью символов *eofstr* и строкой, содержащей только последовательность *eofstr*, или символом конца файла. Если последовательность *eofstr* указана в кавычках, то оболочка не обрабатывает специальные символы и команды, которые содержатся в строках вводимого текста. В противном случае, вместо переменных и команд оболочка подставляет нужные значения и игнорирует символ новой строки в формате **\символ_новой_строки**. Роль кавычек в переменной *eofstr* и строках вводимого текста играет обратная косая черта (\). Если в опции перенаправления **<<** задан символ **-**, то в переменной *eofstr* и строках вводимого текста будут удаляться начальные пробелы.
<&*Цифра*	Стандартный ввод связывается с файлом, дескриптор которого указан в параметре *Цифра.*
>&*Цифра*	Стандартный вывод связывается с файлом, дескриптор которого указан в параметре *Цифра.*
<&-	Закрывает файл стандартного ввода.
>&-	Закрывает файл стандартного вывода.

Примечание: Оболочка с сокращенным набором команд не поддерживает перенаправление вывода.

Дополнительная информация о перенаправлении ввода-вывода приведена в разделе “Перенаправление ввода и вывода” на стр. 342.

Список встроенных команд оболочки Bourne:

Ниже приведен список встроенных команд оболочки Bourne.

Элемент	Описание
:	Возвращает нуль.
.	Читает и выполняет команды, заданные в качестве параметра.
break	Прерывает внешний цикл команды **for**, **while** или **until**, если он выполняется.
cd	Предназначена для перехода из текущего каталога в указанный каталог.
continue	Возобновляет следующий внешний цикл команды **for**, **while** или **until**.
echo	Записывает строки символов в устройство стандартного вывода.
eval	Выполняет команду или команды, указанные в качестве параметров.
exec	Запускает команду, указанную в параметре **Аргумент**, не создавая нового процесса (команда заменяет оболочку).
exit	Завершает работу оболочки с кодом возврата, заданным в параметре **n**.
export	Заносит указанные имена в список имен, которые автоматически экспортируются в среду выполнения следующих команд.
hash	Находит и запоминает расположение в пути поиска всех команд, указанных в параметре команды.
pwd	Показывает текущий каталог.
read	Считывает одну строку из стандартного ввода.
readonly	Помечает имя, указанное в параметре **Имя**, как доступное только для чтения.
return	Завершает работу функции с указанным кодом возврата.
set	Управляет записью в стандартное устройство вывода различных параметров.
shift	Сдвигает аргументы команды влево.
test	Вычисляет условные выражения.
times	Показывает общее время, которое заняло выполнение пользовательских и системных процессов в оболочке.
trap	Выполняет указанную команду, когда оболочка получает один из перечисленных сигналов.
тип	Указывает, как способ интерпретации оболочкой указанного в качестве параметра имени.
ulimit	Показывает или настраивает выделенные ресурсы оболочки.
umask	Определяет права доступа к файлу.
unset	Удаляет связанную с данным именем переменную или функцию.

Элемент	Описание
wait	Ждет завершения дочернего процесса и сообщает его код возврата.

Ссылки, связанные с данной:

“Встроенные команды оболочки Bourne” на стр. 265
Особые команды встроены в оболочку Bourne и выполняются как ее процессы.

Команды оболочки Bourne:

В оболочке Bourne можно выполнять определенные команды.

Когда вы вводите команду в оболочке Bourne, оболочка сначала выполняет предварительную обработку команды, делая все подстановки. Затем она выполняет команду, если соблюдены следующие условия:

- Указанная команда является особой встроенной командой оболочки Bourne.

  ИЛИ
- Имя команды совпадает с именем определенной функции. В этом случае позиционные параметры становятся параметрами функции.

Если команда не совпадает ни со встроенной командой, ни с именем определенной функции, но указывает на выполняемый файл, являющийся откомпилированной программой (кодом), то оболочка становится *родительской* и создает новый (*дочерний*) процесс, немедленно запускающий программу. Если файл помечен как выполняемый, но не является откомпилированной программой, то оболочка предполагает, что это процедура оболочки. В этом случае оболочка создает новый экземпляр самой себя (*подоболочку*) с целью прочесть файл и выполнить указанные в нем команды. Кроме того, оболочка запускает заключенную в круглые скобки команду в подоболочке. С точки зрения пользователя откомпилированная программа выполняется точно так же, как процедура оболочки. Обычно оболочка просматривает каталоги файловой системы в поисках команд в следующем порядке:

1. `/usr/bin`
2. `/etc`
3. `/usr/sbin`
4. `/usr/ucb`
5. `$HOME/bin`
6. `/usr/bin/X11`
7. `/sbin`
8. Текущий каталог

Оболочка просматривает каталоги по очереди. Если в просматриваемом каталоге команда не найдена, оболочка переходит к следующему каталогу.

Примечание: Порядок просмотра каталогов определяется переменной *PATH*. Вы можете изменить его, задав другое значение переменной *PATH*.

Если при запуске команды будет указан путь (например, `/usr/bin/sort`), оболочка не будет просматривать другие каталоги. Если имя команды содержит косую черту (/), то путь поиска не применяется.

Вы можете указать полный путь от корневого каталога (например, `/usr/bin/sort`). Вы можете также указать неполный путь (путь относительно текущего каталога). Например, если вы укажете:

```
bin/myfile
```

то оболочка будет искать каталог `bin` и его файл `myfile` в текущем каталоге.

Примечание: В оболочке с ограничениями не выполняются команды, в которых есть символ косой черты (/).

Для каждой выполненной команды оболочка запоминает ее расположение в пути поиска (во избежание выполнения впоследствии ненужных команд **exec**). Если команда находится в некорневом каталоге, то ее расположение должно определяться заново при каждом изменении текущего каталога. Оболочка "забывает" все известные ей расположения в случае изменения переменной *PATH* или запуска команды **hash -r**.

Заключение символов в кавычки:

Некоторые символы для оболочки являются специальными. Иногда возникает необходимость указать такие специальные символы как обычные. Для этого следует заключить строку символов в одинарные (') или двойные кавычки (") или поставить обратную косую черту (\) перед одним символом.

Все символы (за исключением скобочных одинарных кавычек) рассматриваются как обычные, без учета их специального назначения. Таким образом, команда:

```
stuff='echo $? $*; ls * | wc'
```

присвоит строку символов `echo $? $*; ls * | wc` переменной `stuff`. Оболочка не будет выполнять команды **echo**, **ls** и **wc**, а также раскрывать переменные `$?` и `$*` и специальный символ `*`.

В двойных кавычках сохраняется значение символов $, ` и ", а все остальные символы не интерпретируются. Таким образом, в строке, заключенной в двойные кавычки, выполняется подстановка команд и переменных. Кроме того, кавычки не влияют на команды, полученные в результате подстановки в строке, заключенной в кавычки, поэтому специальные символы в таких командах сохраняют свое значение.

Рассмотрим следующую последовательность строк:

```
ls *
файл1 файл2 файл3
message="Этот каталог содержит `ls * ` "
echo $message
Этот каталог содержит файл1 файл2 файл3
```

Приведенный пример показывает, что специальный символ * (звездочка) был раскрыт при подстановке команды.

Если нужно отменить специальное значение символов $, ` и ", укажите перед ними символ обратной косой черты (\). Если двойные кавычки не применяются, то постановка обратной косой черты перед символом эквивалентна заключению символа в одинарные кавычки. Следовательно, обратная косая черта, стоящая перед символом перехода на новую строку (т.е. обратная косая черта в конце строки), скрывает символ перехода на новую строку и позволяет продолжить команду на следующей физической строке.

Обработка сигналов:

Оболочка игнорирует сигналы **INTERRUPT** и **QUIT** для выполняемой команды, если команда оканчивается символом &, т.е. если команда выполняется в фоновом режиме. В остальных случаях значения сигналов наследуются от родительской оболочки, за исключением сигнала **SEGMENTATION VIOLATION** (нарушение сегментации).

Дополнительная информация приведена в описании встроенной команды оболочки Bourne **trap**.

Составные команды оболочки Bourne:

Составными называются следующие команды.

- Конвейер (одна или несколько простых команд, отделенных друг от друга символом |.
- Список простых команд
- Команды, начинающиеся с зарезервированного слова
- Команды, начинающиеся с управляющего оператора ((открывающая круглая скобка)

Если не указано противное, вывод составной команды - это значение, выданное последней простой командой.

Зарезервированные слова:

Следующие зарезервированные слова для оболочки Bourne распознаются только при условии, что они указаны без кавычек в качестве первого слова команды.

```
for         do          done
case        esac
if          then         fi
elif        else
while       until
{           }
(           )
```

Элемент	Описание
for*Идентификатор* [**in***Слово*. . .] **do***Список* **done**	Присваивает параметру *Идентификатор* одно или несколько слов, заданных параметром *Слово* (по одному слову поочередно), и запускает команды, перечисленные в параметре *Список*. Если вы опустите **in** *Слово*. . ., то команда **for** запустит команды, перечисленные в параметре *Список*, ля каждого заданного позиционного параметра; обработка будет завершена, когда все позиционные параметры будут использованы.
case *Слово* **in** *Шаблон* [|*Шаблон*] . . . **)** *Список***;;** [*Шаблон* [|*Шаблон*] . . . **)** *Список***;;**] **. . . esac**	Запускает команды, перечисленные в параметре *Список* и связанные с первым из параметров *Шаблон*, соответствующим значению параметра *Слово*. В шаблонах применяются те же формат и обозначения, что при подстановке имен файлов, за исключением того что косая черта **/**, начальная точка **.**и точка, стоящая сразу за косой чертой (**/.**), не должны совпадать явным образом.
if *Список* **then** *Список* [**elif** *Список* **then** *Список*] . . . [**else** *Список*] **fi**	Запускает команды, перечисленные в параметре *Список* после команды **if**. Если команда возвращает нуль, то оболочка запускает команды, перечисленные в параметре *Список* после команды **then**. Иначе оболочка запускает команды, перечисленные в параметре *Список* после команды **elif** (если последняя указана). Если результирующим значением будет ноль, то оболочка запустит команды, перечисленные в параметре Список после следующей команды **then**. В противном случае оболочка запустит команды, перечисленные в параметре *Список* после команды **else** (если последняя указана). Если команды **else** - *Список* и **then** - *Список* отсутствуют, то команда **if** возвращает нулевое значение.
while *Список* **do** *Список* **done**	Запускает команды, перечисленные в параметре *Список* после команды **while**. Если последняя команда в параметре *Список* **while** возвращает нуль, оболочка выполняет команды, перечисленные в параметре *Список* команды **do**. Выполнение цикла по всем спискам продолжается до тех пор, пока последняя команда из параметра *Список* команды **while** не возвратит ненулевое значение. Если команды в параметре *Список* команды **do** отсутствуют, то команда **while** возвращает нулевое значение.
until *Список* **do** *Список* **done**	Запускает команды, перечисленные в параметре *Список* после команды **until**. Если последняя команда в параметре *Список* команды **until** возвращает нуль, оболочка выполняет команды, перечисленные в параметре *Список* команды **do**. Выполнение цикла по всем спискам продолжается до тех пор, пока последняя команда из параметра *Список* команды **until** не возвратит нуль. Если команды в параметре *Список* команды **do** отсутствуют, то команда **until** возвращает нулевое значение.
(*Список***)**	Запускает команды, перечисленные в параметре *Список*, в подоболочке.
{*Список***; }**	Запускает команды, перечисленные в параметре *Список*, в текущей оболочке, не запуская подоболочку.
Имя **()** {*Список*}	Определяет функцию, указанную в параметре *Имя*. Тело функции образует список команд в фигурных скобках в параметре *Список*.

Встроенные команды оболочки Bourne:

Особые команды встроены в оболочку Bourne и выполняются как ее процессы.

Если не указано иное, вывод команды записывается в файл с дескриптором 1 (стандартный вывод), а код завершения равен нулю, если команда не содержит синтаксических ошибок. Разрешается перенаправление ввода и вывода.

Следующие особые команды обрабатываются несколько иначе, чем остальные встроенные команды:

```
:
(двоеточие) exec shift
. (точка)       exit            times
break         export          trap
continue      readonly        wait
eval          return
```

Оболочка Bourne обрабатывает эти команды следующим образом:

- Списки соответствия ключевых слов параметрам, предшествующие команде, остаются в силе по окончании выполнения команды.
- Опции перенаправления ввода-вывода обрабатываются после присвоения значений параметрам.
- Ошибки в сценарии оболочки приводят к завершению работы сценария.

Ссылки, связанные с данной:

“Список встроенных команд оболочки Bourne” на стр. 262
Ниже приведен список встроенных команд оболочки Bourne.

Описания особых команд:

В оболочке Bourne предусмотрены следующие особые встроенные команды.

Элемент	Описание
:	Возвращает нуль.
. *Файл*	Считывает и выполняет команды из файла, указанного в параметре *Файл*, после чего возвращает к исходному уровню. Подоболочка не запускается. Каталог указанного файла определяется по пути поиска, заданного переменной *PATH*.
break [*n*]	Прерывает внешний цикл команды **for**, **while** или **until**, если он выполняется. Если вы укажете переменную *n*, то команда **break** прервет *n* уровней циклов.
continue [*n*]	Возобновляет следующий внешний цикл команды **for**, **while** или **until**. Если вы укажете переменную *n*, то команда возобновит цикл *n*-го уровня.
cd*Каталог*]	Переходит к каталогу *Каталог*. Если *Каталог* не указан, будет использовано значение переменной оболочки *HOME*. Переменная оболочки *CDPATH* задает путь поиска для *Каталога*. Переменная *CDPATH* - это список альтернативных имен каталогов, перечисленных через двоеточие. Пустой путь (путь по умолчанию) соответствует текущему каталогу. Этот пустой путь указан сразу после знака равенства в операторе присвоения или между двоеточиями-разделителями в списке путей. Если *Каталог* начинается с символа / (косая черта), то путь поиска не применяется. В противном случае, поиск выполняется в каждом каталоге, указанном в переменной *CDPATH*. **Примечание:** В оболочке с ограничениями команду оболочки **cd** запускать нельзя.
echo *Строка . . .*]	Записывает строки символов в устройство стандартного вывода. Информация о применении и параметрах приведена в справке по команде **echo**. Флаг **-n** не поддерживается.
eval [*Аргумент . . .*]	Считывает аргументы в качестве входных данных оболочки и запускает сформированные команды.
exec [*Аргумент . . .*]	Запускает команду, указанную в параметре *Аргумент*, вместо данной оболочки, не создавая новый процесс. Допустимы аргументы ввода и вывода; если других аргументов нет, то с их помощью можно изменять входные и выходные данные оболочки. Эту команду не рекомендуется выполнять в начальной оболочке.
exit [*n*]	Завершает работу оболочки с результирующим значением, указанным в параметре *n*. Если этот параметр не указан, то значением выхода будет результат выполнения последней команды (работу оболочки можно также завершить нажатием клавиш Ctrl-D). В качестве *n* допустимы значения от 0 до 255 включительно.
export [*Имя . . .*]	Помечает указанные имена для автоматического экспортирования в среды выполняемых впоследствии команд. Если параметр *Имя* опущен, то команда **export** выдает список всех имен, экспортированных в данной оболочке. Имена функций экспортировать нельзя.

Элемент	Описание
hash [**-r**][*Команда* . . .]	Находит и запоминает расположение в пути поиска всех команд, указанных в параметре *Команда*. Флаг **-r** удаляет всю хранящуюся в оболочке информацию о расположениях. Если ни флаг, ни команды не указаны, то оболочка выдаст информацию о командах, расположения которых известны, в следующем формате: `Запуски  Стоимость  Команда` В поле `Запуски` указано, сколько раз оболочка запускала команду. В поле `Стоимость` указан объем работы, который потребовался на поиск команды в пути поиска. В поле `Команда` указан путь к команде. В некоторых случаях необходимо повторно определить путь к команде, например, если путь относительный, т.е. указан от текущего каталога, а текущий каталог изменился. Команды, для которых допустима такая операция, помечены звездочкой (`*`) в столбце `Запуски`. Значение в колонке `Стоимость` увеличивается при каждом вычислении пути.
pwd	Показывает текущий каталог. Описание опций команды приведено в справке по команде **pwd**.
read [*Имя* . . .]	Считывает одну строку из стандартного ввода. Первое слово в этой строке присваивается первому параметру *Имя*, второе - второму параметру *Имя* и т.д.; последнему параметру *Имя* присваиваются все оставшиеся слова. Если команда не обнаруживает символ конца файла, то она возвращает нуль.
readonly [*Имя* . . .]	Помечает имя, указанное в параметре *Имя*, как доступное только для чтения. Сбросить значение имени нельзя. Если *Имя* не указано, то команда **readonly** выдает список всех имен, доступных только для чтения.
return [*n*]	Завершает работу функции с кодом возврата *n*. Если переменная *n* опущена, то функция возвратит состояние последней команды, которую она выполнила. Эта команда применима только к функциям оболочки.
set [*Флаг* [*Аргумент*] . . .]	Задает один или несколько следующих флагов: **-a** Помечает для экспортирования все переменные, для которых выполняется присвоение. Если присвоение предшествует имени команды, то атрибут экспортирования действует только в данной среде выполнения команды, за исключением случаев, когда присвоение предшествует одной из особых встроенных команд. В последнем случае, атрибут экспортирования остается действительным и после обработки встроенной команды. Если присвоение не предшествует имени команды или если аргумент получен в результате выполнения команды **getopts** или **read**, то атрибут экспортирования остается действительным, пока переменная не будет сброшена. **-e** Немедленно завершает работу, если для команды выполнены все перечисленные ниже условия: • Команда завершает работу с положительным значением выхода. • Команда не входит в составной список команды **while**, **until** или **if**. • Команда не участвует в проверке по спискам AND или OR. • Команда не является командой конвейера, перед которой стоит зарезервированное слово ! (восклицательный знак). **-f** Отключает подстановку имен файлов. **-h** Находит и запоминает команды, вызываемые внутри определений функций. (Обычно поиск этих команд происходит при выполнении функции. Более подробные сведения приведены в справке по команде **hash**). **-k** Помещает в среду выполнения команды все параметры ключевых слов, а не только те, которые предшествуют имени команды. **-n** Считывает команды, но не запускает их. Флаг **-n** применяется для проверки сценария оболочки на наличие синтаксических ошибок. **-t** Завершает работу после считывания и выполнения одной команды. **-u** Рассматривает незаданную переменную как ошибку и немедленно завершает работу при подстановке переменных. Работа интерактивной оболочки при этом не завершается. **-v** Показывает входные строки оболочки по мере их считывания. **-x** Показывает команды и их аргументы перед выполнением. **—** Не изменяет ни один из флагов. Это полезно при указании позиционного параметра **$1** в строке, начинающейся со знака дефис (**-**).

Элемент	Описание
	Указание плюса (+) вместо дефиса (-) сбрасывает флаги. Эти флаги можно также задать в командной строке оболочки. Специальная переменная *$-* содержит текущий набор флагов. Любой *Аргумент*, указанный в команде **set**, становится позиционным параметром и присваивается переменной $1, $2, **...** и т.д. Если *флаг* и *Аргумент* опущены, команда **set** выдает все имена и значения текущих переменных оболочки.
shift [*n*]	Сдвигает аргументы командной строки влево, т.е. повторно присваивает значения позиционных параметров, аннулируя текущее значение **$1** и присваивая значение **$2** параметру **$1**, значение **$3** - параметру **$2** и т.д. Если в командной строке указано больше 9 аргументов, то 10-й присваивается параметру **$9**, а остальные остаются неприсвоенными (до следующей команды **shift**). Если аргументов 9 или меньше, то команда **shift** сбрасывает позиционный параметр с наибольшим номером, которому присвоено значение. Позиционный параметр **$0** никогда не сдвигается. Команда **shift** *n* - это сокращенная форма записи *n* команд последовательного сдвига. Значение параметра *n* по умолчанию равно 1.
test*Выражение* | [*Выражение*]	Вычисляет условные выражения. Описание флагов и параметров приведено в справке по команде **test**. Флаг **-h** не поддерживается встроенной командой тестирования в **bsh**.
times	Показывает общее время, которое заняло выполнение пользовательских и системных процессов в оболочке.
trap [*Команда*] [*n*] . . .	Запускает команду, указанную в параметре *Команда*, когда оболочка принимает один или несколько сигналов, заданных параметром *n*. Порядок запуска команд **trap** определяется номерами сигналов. Любая попытка задать прерывание (trap) по сигналу, проигнорированному при входе в текущую оболочку, будет неудачной. **Примечание:** Оболочка просматривает параметр *Команда* первый раз, когда прерывание устанавливается, а второй раз - когда оно происходит.Если команда опущена, то все прерывания, указанные в параметре *n*, сбрасываются к их текущим значениям. Если указана пустая строка, то сигнал игнорируется оболочкой и выполняемыми ей командами. Если параметр *n* равен нулю, то указанная команда выполняется при выходе из оболочки. Если ни команда, ни сигнал не заданы, то команда **trap** выдает список команд, связанных с номерами сигналов.
type [*Имя* . . .]	Указывает, как оболочка проинтерпретирует указанное в качестве параметра *Имя*.

Элемент	Описание
ulimit [**-HS**] [**-c** | **-d** | **-f** | **-m** | **-s** | **-t**] [*ограничение*]	Показывает или настраивает выделенные ресурсы оболочки. Параметры ресурсов оболочки можно просматривать по-отдельности или группами. Режим по умолчанию - просмотр ресурсов, которым присвоено "мягкое" значение (нижняя граница), в группе. Значения ресурсов оболочки зависят от ИД пользователя текущей оболочки. "Жесткий" уровень ресурса может задать только пользователь текущей оболочки с правами доступа root. Если пользователь, отличный от пользователя root, попытается изменить "жесткий" уровень ресурса, возникнет ошибка. По умолчанию пользователь root задает и "жесткий", и "мягкий" уровни конкретного ресурса. По этой причине, пользователь root должен быть осторожен при указании флагов **-S**, **-H** или флагов по умолчанию для установки ограничений. Пользователи, не имеющие прав доступа root, могут задавать только "мягкую" границу ресурса. После того как ограничение было уменьшено пользователем, отличным от пользователя root, его нельзя увеличить, даже до первоначального системного ограничения. Для того чтобы задать ограничение на ресурс, выберите соответствующий флаг и ограничение для нового ресурса, которое должно быть целым числом. За один раз можно задать только одно ограничение на ресурс. Если вы укажете несколько флагов ресурсов, результаты будут непредсказуемыми. По умолчанию команда **ulimit** с одним новым значением в командной строке задает размер файла оболочки. Флаг **-f** необязателен. Допустимы следующие флаги команды **ulimit**: **-c** Задает или показывает сегмент ядра оболочки. **-d** Задает или показывает сегмент данных оболочки. **-f** Задает или показывает размер файла оболочки. **-H** Задает или показывает "жесткую" границу ресурса (доступен только пользователю root). **-m** Задает или показывает объем памяти оболочки. **-r** Задает или показывает максимальное количество нитей за процесс. **-s** Задает или показывает сегмент стека оболочки. **-S** Задает или показывает "мягкую" границу ресурса. **-t** Задает или показывает максимальное время CPU для оболочки. **-u** Задает или показывает максимальное количество процессов на пользователя.
umask [*nnn*]	Определяет права доступа к файлу. Это значение, совместно с правами доступа создающего процесса, определяет права доступа к создаваемому файлу. Значение по умолчанию: 022. Если значение не указано, команда **umask** показывает текущее значение.
unset [*Имя . . .*]	Сбрасывает переменные и функции, имена которых перечислены в параметре *Имя*. Сбросить переменные оболочки *PATH*, *PS1*, *PS2*, *MAILCHECK* и *IFS* нельзя.
wait [*n*]	Ожидает завершения дочернего процесса, номер которого указан в параметре *n*, и возвращает состояние выхода этого процесса. Если параметр *n* опущен, то оболочка ожидает завершения всех текущих активных дочерних процессов и возвращает нуль.

Подстановка команд в оболочке Bourne:

Подстановка команд позволяет сделать вывод одной команды аргументом другой команды.

Если вы заключите команду в обратные кавычки (``), то оболочка сначала запустит все перечисленные команды, а затем заменит все выражение, включая обратные кавычки, на данные вывода. Этот прием часто применяется для присвоения значений переменным оболочки. Например, оператор:

```
today=`date`
```

присваивает строку, задающую текущую дату, переменной *today*. Следующая операция присвоения сохраняет в переменной *files* количество файлов в текущем каталоге:

```
files=`ls | wc -l`
```

Вы можете выполнить подстановку команд в любой команде, создающей стандартный вывод.

Если вы хотите указать вложенные подстановки, поставьте обратную косую черту (\) перед каждой из открывающих обратных кавычек, например:

```
logmsg=`echo Ваш начальный каталог: \`pwd\``
```

Кроме того, вы можете присваивать значения переменным оболочки косвенно с помощью особой команды **read**. Эта команда считывает строку из стандартного ввода (с клавиатуры) и последовательно присваивает ее слова любым указанным переменным. Например:

```
read first init last
```

считывает строку:

```
И. П. Иванов
```

то результат будет тем же, как если бы вы ввели:

```
first=И. init=П. last=Иванов
```

Особая команда **read** присваивает все лишние слова последней переменной.

Подстановка переменных в оболочке Bourne:

Оболочка Bourne позволяет выполнить подстановку переменных.

В оболочке Bourne существует несколько способов создания переменных (т.е. присвоения строкового значения некоторому имени). Некоторые переменные, а также позиционные и ключевые параметры обычно задаются только в командной строке. Другие переменные - это просто имена, которым вы или оболочка можете присваивать строковые значения.

Понятия, связанные с данным:

“Неконтролируемые терминалы” на стр. 296
Если пользователь вошел в систему с некоторого терминала, а затем оставил терминал без присмотра, то уязвимость системы значительно повышается. Наиболее серьезные проблемы возникают в случае, когда таким пользователем является системный администратор (пользователь root). В общем случае, пользователи должны выходить из системы всякий раз, когда они оставляют терминал без присмотра.

Пользовательские переменные в оболочке Bourne:

Оболочка Bourne распознает алфавитно-цифровые переменные, которым могут быть присвоены строковые значения.

Для присвоения строкового значения некоторому имени введите:

```
Имя=Строка
```

Имя - это последовательность букв, цифр и знаков подчеркивания, начинающаяся со знака подчеркивания или буквы. Для того чтобы узнать значение, присвоенное переменной, поставьте знак доллара ($) перед именем переменной. Таким образом, при указании *$Имя* будет выдано значение *Строка*. Учтите, что по обеим сторонам знака равенства (=) не должно быть пробелов. (Указывать позиционные параметры в операторе присвоения нельзя. Вы можете ввести несколько операторов присвоения в командной строке, но учтите, что оболочка выполняет присвоение в обратном порядке, т.е. справа налево.

Если *Строка* заключена в двойные (") или одинарные (') кавычки, то оболочка будет рассматривать пробелы, знаки табуляции, точки с запятой и символы новой строки, указанные внутри строки, не как ограничители слов, а как обычные символы.

Если *Строка* заключена в двойные кавычки ("), то оболочка по-прежнему будет распознавать имена переменных, указанные в строке, и выполнять подстановку переменных, т.е. заменять ссылки на позиционные параметры и указанные со знаком доллара ($) имена переменных на соответствующие значения. Кроме того, в таких строках будет выполняться подстановка команд.

Если *Строка* заключена в одинарные кавычки (''), то подстановка переменных и команд внутри строки выполняться не будет. Следующий пример иллюстрирует это различие:

```
Ввод:           num=875
                number1="Add $num"
                number2='Add $num'
                echo $number1
Вывод:          Add 875
Ввод:           echo $number2
Вывод:          Add $num
```

После подстановки переменных пробелы не обрабатываются. Таким образом, следующие операторы присвоения дадут одинаковые значения в `$first` и `$second`:

```
first='a string with embedded blanks'
second=$first
```

При указании ссылки на переменную вы можете заключить имя переменной (или цифру, задающую позиционный параметр) в фигурные скобки ({ }), чтобы отделить ее от последующих символов. Если символ, стоящий сразу за именем - буква, цифра или знак подчеркивания, а переменная не является позиционным параметром, то фигурные скобки необходимы:

```
Ввод:           a='This is a'
                echo "${a}n example"
Вывод:          This is an example
Ввод:           echo "$a test"
Вывод:          This is a test
```

Понятия, связанные с данным:

“Позиционные параметры в оболочке Bourne” на стр. 260
При запуске процедуры оболочки она неявно создает по одному позиционному параметру для каждого слова, указанного в командной строке.

Ссылки, связанные с данной:

“Подстановка команд в оболочке Bourne” на стр. 259
Обычно оболочка заменяет выражение $*Переменная* на строковое значение, присвоенное переменной *Переменная*, если оно есть. Однако вы можете выполнить и *условную подстановку*, т.е. подстановку в зависимости от того, задано значение переменной или нет, пустое оно или нет, или обоих условий.

Переменные, используемые оболочкой Bourne:

Ниже перечислены переменные, применяемые оболочкой. Оболочка задает только некоторые из них; вы можете задать или сбросить любые переменные.

Элемент	Описание
CDPATH	Задает путь поиска для команды **cd** (изменить каталог).
HOME	Указывает имя *начального каталога*, т.е. каталога, который становится текущим после входа в систему. Эта переменная инициализируется программой `login`. Команда **cd** по умолчанию применяет значение переменной *$HOME*. Применение этой переменной вместо абсолютных имен позволяет каждому пользователю без изменений выполнять процедуру в своей среде из различных каталогов.
IFS	Задает символы, которые могут служить внутренними разделителями поля (IFS) и учитываются оболочкой при обработке пробелов. Первоначально переменная *IFS* содержит пробел, символ табуляции и символ новой строки.
LANG	Определяет локаль, которая будет применяться, если переменная *LC_ALL* и соответствующая переменная среды (начинающаяся с `LC_`) не заданы.
LC_ALL	Задает локаль, переопределяющую значения переменной среды *LANG*, а также всех переменных среды, начинающихся с `LC_`.
LC_COLLATE	Определяет последовательность упорядочения, применяемую при сортировке имен и выборе символов по шаблону.

Элемент	Описание
LC_CTYPE	Определяет локаль для обработки байтов текстовых данных как символов (однобайтовых или многобайтовых), а также для проверки того, является ли данный символ буквой (класс алфавитных символов - **alpha**), и выбора символов по шаблону.
LC_MESSAGES	Задает язык, на котором выдаются сообщения.
LIBPATH	Задает путь поиска общих библиотек.
LOGNAME	Задает имя пользователя, под которым вы вошли в систему, помеченное как `readonly` в файле `/etc/profile`.
MAIL	Указывает путь к файлу, с помощью которого функция почты проверяет поступление новой почты. Если эта переменная задана, то оболочка периодически проверяет, не изменился ли файл. Если да, причем файл не пуст, то оболочка выдает сообщение `$MAILMSG`. Переменная *MAIL* задается в файле `.profile`. Обычно пользователи команды **mail** присваивают этой переменной значение `/usr/spool/mail/$LOGNAME`.
MAILCHECK	Интервал времени в секундах, через который оболочка проверяет поступление почты в файлах, заданных в переменных *MAILPATH* и *MAIL*. Значение по умолчанию - 600 секунд (10 минут). Если вы присвоите переменной *MAILCHECK* нулевое значение, то проверка будет выполняться перед каждой выдачей приглашения.
MAILMSG	Сообщение, уведомляющее о поступлении почты. Если вы зададите пустую строку в переменной *MAILMSG* (`MAILMSG=""`), то сообщение выдаваться не будет.
MAILPATH	Список имен файлов, разделенных двоеточиями. Если эта переменная задана, то оболочка будет информировать вас о поступлении почты в любой из файлов, перечисленных в этом списке. После любого имени файла вы можете ввести символ *%* и текст сообщения, которое должно выдаваться при поступлении почты в этот файл. По умолчанию оболочка выдает сообщение из переменной *MAILMSG*, а если она не задана, то сообщение `[Вам пришла почта]`. **Примечание:** Если переменная *MAILPATH* задана, то проверяются файлы, заданные в ней, а не в переменной *MAIL*. Если необходимо проверять файлы, заданные в обеих переменных - *MAILPATH* и *MAIL*, укажите файл `MAIL` в списке файлов в переменной `MAILPATH`.
PATH	Путь для поиска команд, заданный в виде упорядоченного списка имен каталогов, разделенных двоеточиями. При поиске команд каталоги просматриваются в указанном порядке. Пустая строка (два двоеточия подряд, начальное или конечное двоеточие) в этом списке соответствует текущему каталогу. Переменная *PATH* обычно инициализируется в файле `/etc/environment`, и обычно ее значение равно `/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin`. Вы можете сбросить значение этой переменной. Переменная *PATH* в вашем файле `.profile` содержит также каталог `$HOME/bin` и ваш текущий каталог. Если вы создали каталог, специально предназначенный для какого-либо проекта, например, `/project/bin`, и хотите, чтобы прежде всего просматривался именно этот каталог, задайте переменную *PATH* следующим образом: `PATH=/project/bin:$PATH` Если вы хотите задать для переменной *PATH* значение, отличное от значения по умолчанию, то лучше всего сделать это в файле `$HOME/.profile`. Учтите, что в оболочке с ограничениями сбросить значение переменной *PATH* нельзя.
PS1	Задает основное приглашение системы. Интерактивная оболочка выдает его, когда ожидает ввода информации. Значение переменной *PS1* по умолчанию для пользователей без прав доступа root - $ и пробел.
PS2	Задает вспомогательное приглашение системы. Если оболочка обнаруживает символ новой строки в данных ввода и ожидает продолжения ввода, то она выдает приглашение, указанное в переменной *PS2*. Значение переменной *PS2* по умолчанию - знак "больше" (>) и пробел.
SHACCT	Имя принадлежащего вам файла. Если эта переменная задана, то оболочка считывает учетную запись из файла для каждого выполняемого сценария оболочки. Для анализа собранных данных вы можете воспользоваться программами учета, такими как `acctcom` и `acctcms`.
SHELL	Задает путь к программе оболочки. Эта переменная должна быть задана и экспортирована файлом `$HOME/.profile` для каждой оболочки с ограничениями.
TIMEOUT	Время простоя (в минутах), после которого оболочка завершает работу. Если этой переменной присвоено положительное значение, то оболочка автоматически завершает работу в случае, если в течение указанного времени в ответ на приглашение `PS1` не будет введена ни одна команда. (Учтите, что в оболочке может быть установлено жесткое ограничение, которое нельзя превысить путем увеличения этого значения.) Нулевое значение означает неограниченное время простоя.

Понятия, связанные с данным:

“Обработка пробелов”
После того как оболочка выполнит подстановку переменных и команд, она просматривает результаты в поисках внутренних разделителей полей (определенных в переменной оболочки *IFS*).

Предустановленные специальные переменные в оболочке Bourne:

Некоторые переменные в оболочке Bourne являются специальными. Следующие переменные задаются только в оболочке Bourne:

Элемент	Описание
$@	Выдает значения позиционных параметров, начиная с **$1**. Параметры будут разделены пробелами. Если вы заключите символы **$@** в двойные кавычки (" "), то каждый позиционный параметр будет считаться отдельной строкой. Если позиционные параметры отсутствуют, то оболочка Bourne преобразует выражение в пустую строку без кавычек.
$*	Выдает значения позиционных параметров, начиная с **$1**. Параметры отделяются друг от друга при помощи символа, который указан первым в переменной *IFS*. Если символы **$*** заключены в двойные кавычки (" "), то выданные значения позиционных параметров также будут заключены в двойные кавычки. Как и в первом случае, значения параметров будут разделены первым символом переменной *IFS*.
$#	Задает число позиционных параметров, переданных оболочке, без учета имени самой процедуры оболочки. Таким образом, переменная *$#* содержит номер последнего позиционного параметра. Эта переменная позволяет убедиться, что оболочке передано необходимое число аргументов. Учтите, что в оболочке будут доступны только позиционные параметры с **$0** по **$9**.
$?	Выдает код завершения последней выполненной команды. Это десятичная строка. Большинство команд возвращают нулевое значение (код успешного завершения). Сама оболочка возвращает текущее значение переменной *$?* в качестве кода завершения.

$$	Указывает номер текущего процесса. Поскольку номера процессов уникальны, эта строка часто применяется для создания уникальных имен временных файлов.
	
	Ниже приведен пример создания такого временного файла:
	`temp=/tmp/$$`
	`ls >$temp`
	`.`
	`.`
	`.`
	`rm $temp`

$!	Задает номер последнего выполненного фонового процесса (процесс выполняется в фоновом режиме, если в конце команды его запуска указан символ &).
$-	Эта строка состоит из флагов выполнения, заданных в оболочке в данный момент.

Понятия, связанные с данным:

“Позиционные параметры в оболочке Bourne” на стр. 260
При запуске процедуры оболочки она неявно создает по одному позиционному параметру для каждого слова, указанного в командной строке.

Обработка пробелов:

После того как оболочка выполнит подстановку переменных и команд, она просматривает результаты в поисках внутренних разделителей полей (определенных в переменной оболочки *IFS*).

Оболочка разбивает строку на отдельные слова в тех местах, где стоят разделители. Затем она оставляет явные пустые аргументы ("" и '') и отбрасывает неявные пустые аргументы (полученные от параметров, у которых нет значений).

Ссылки, связанные с данной:

“Переменные, используемые оболочкой Bourne” на стр. 271
Ниже перечислены переменные, применяемые оболочкой. Оболочка задает только некоторые из них; вы можете задать или сбросить любые переменные.

Оболочка C

Оболочка C - это интерактивный интерпретатор команд и командный язык программирования. Ее синтаксис похож на синтаксис языка программирования C.

Для запуска оболочки C предназначена команда **csh**.

При входе в систему команда **csh** прежде всего пытается найти системный файл конфигурации `/etc/csh.cshrc`. При наличии такого файла оболочка C выполняет указанные в нем команды. Затем оболочка C выполняет команды из системного файла `/etc/csh.login`, если он существует. Затем она пытается установить домашний каталог на основе файлов `.cshrc` и `.login`. В этих файлах хранится информация, необходимая для работы оболочки C. Все переменные из файлов `/etc/csh.cshrc` и `/etc/csh.login` могут быть переопределены в файлах `.cshrc` и `.login` из вашего домашнего каталога (`$HOME`). Файлы `/etc/csh.cshrc` и `/etc/csh.login` может изменять только пользователь root.

Файлы `/etc/csh.login` и `$HOME/.login` обрабатываются только один раз при входе в систему. Обычно они используются для хранения переменных среды, команд, которые требуется выполнять при входе в систему, или параметров терминала.

Файлы `/etc/csh.cshrc` и `$HOME/.cshrc` обрабатываются при входе в систему и каждый раз при вызове команды **csh** или сценария оболочки C. Обычно они используются для задания параметров оболочки C, например, псевдонимов и переменных среды (таких как *history*, *noclobber* и *ignoreeof*). В файлах `/etc/csh.cshrc` и `$HOME/.cshrc` рекомендуется указывать только встроенные команды оболочки C, так как применение других команд увеличивает время запуска сценариев оболочки.

Ссылки, связанные с данной:

“Список встроенных команд оболочки C” на стр. 281
Ниже приведены примеры встроенных команд оболочки C.

Ограничения оболочки C:

В оболочке C существуют следующие ограничения.

- Длина слова не может превышать 1024 байт.
- Длина списка аргументов не может превышать *ARG_MAX* байт. Значение переменной *ARG_MAX* задается в файле `/usr/include/sys/limits.h`.
- Число аргументов команды, требующих подстановки имен файлов, не может превышать 1/6 от максимального числа байт в списке аргументов.
- При подстановке команд может быть подставлено число байт, не превышающее максимальную длину списка аргументов.
- Для предотвращения зацикливания оболочка ограничивает число подстановок псевдонимов в одной строке двадцатью.
- Команда **csh** не поддерживает подстановку имен файлов на основе отношений эквивалентности символов.
- Дескрипторы файлов (отличные от стандартного потока ввода, стандартного потока вывода и стандартного потока вывода ошибок), открытые до запуска приложения в оболочке **csh**, будут недоступны этому приложению.

Подстановка псевдонимов в оболочке C:

Псевдоним - это имя, присвоенное команде или командной строке. Оболочка C позволяет создавать псевдонимы и применять их в качестве команд. Для этого оболочка хранит список псевдонимов, определенных пользователем.

После того как оболочка считывает командную строку, она делит команду на слова и проверяет, является ли первое слово команды (если смотреть слева направо) псевдонимом. При обнаружении псевдонима оболочка заменяет его на текст соответствующей команды, используя механизм хронологии. При этом заменяется как команда, так и список ее аргументов. Если в списке хронологии нет соответствующих ссылок, список аргументов остается неизменным.

Список псевдонимов создается, просматривается и изменяется встроенными командами **alias** и **unalias**. Команда alias имеет следующий формат:

```
alias [Имя [Список_слов]]
```

В необязательной переменной *Имя* указывается имя, для которого создается псевдоним. Если в переменной *Список_слов* указан список слов, команда делает их псевдонимами *Имени*. Команда **alias** без переменных показывает все псевдонимы оболочки C.

Если для команды **ls** задан псевдоним **ls -l**, то команда

```
ls /usr
```

будет заменена командой

```
ls -l /usr
```

Список аргументов при этом не меняется, поскольку в команде из списка хронологии отсутствуют ссылки. Аналогично, если псевдоним **lookup** соответствует команде

```
grep \!^ /etc/passwd
```

то оболочка заменит строку `lookup bill` на

```
grep bill /etc/passwd
```

В этом примере `!^` заменяется на первый аргумент строки ввода, в данном случае `bill`.

В псевдониме можно применять специальные символы-шаблоны. Следующая команда:

```
alias lprint 'pr &bslash2.!* >

> print'
```

создает команду, форматирующую свои аргументы для построчного принтера. Символ `!` защищен одинарными кавычками, поэтому псевдоним не будет развернут до запуска команды **pr**.

Если оболочка обнаруживает псевдоним, она выполняет преобразование слов исходного текста и начинает обработку псевдонимов заново, в новой строке ввода. Если первое слово нового текста совпадает со старым, цикл обработки псевдонимов прерывается. При обнаружении других вложенных циклов выдается ошибка.

Понятия, связанные с данным:

“Подстановка хронологии в оболочке C” на стр. 291
Подстановка хронологии позволяет получить новые команды путем замены слов в предыдущих командах. С помощью подстановки хронологии можно повторить команду, перенести аргументы предыдущей команды в текущую команду или исправить орфографические ошибки в предыдущей команде.

Подстановка переменных в оболочке C:

Оболочка C поддерживает набор переменных, каждая из которых хранит в качестве значения одно или несколько слов. Некоторые из этих переменных задаются оболочкой или используются ей. Например, переменная *argv* представляет собой список параметров оболочки, причем на любое слово этого списка можно задать ссылку специальным образом.

Для просмотра и изменения значений переменных служат команды **set** и **unset**. Некоторые из переменных, используемых оболочкой, являются переключателями (включают и выключают какую-либо функцию). На

работу оболочки влияет не их значение, а только тот факт, установлены они или нет. Например, переменная оболочки *verbose* - переключатель, включающий эхоповтор выполняемых команд. Установка этой переменной приводит к добавлению в командную строку флага **-v**.

Другие переменные хранят численные значения. Например, команда **@** присваивает переменной результат вычисления арифметического выражения. Однако значения переменных всегда представляются в виде некоторого количества строк. При численных расчетах пустая строка считается равной нулю; второе и последующие слова строки игнорируются.

При анализе команды оболочка разделяет строку ввода и выполняет подстановку псевдонимов. Затем, перед запуском команды, выполняется подстановка переменных. Подстановка выполняется при обнаружении в строке символа $. Однако если за этим символом следует пробел, знак табуляции или символ новой строки, то подстановка не выполняется. Если перед знаком $ стоит символ \, то подстановка также отменяется, за исключением следующих двух случаев:

- Команда заключена в двойные кавычки (" "). В этом случае оболочка всегда выполняет подстановку.
- Команда заключена в одинарные кавычки (' '). В этом случае оболочка никогда не выполняет подстановку. Подстановка команд выполняется в строках, заключенных в ' '.

Оболочка находит операторы перенаправления ввода и вывода перед подстановкой переменных, и обрабатывает их отдельно. Переменные в имени команды и списке аргументов подставляются вместе. Поэтому первое слово (команда) может быть заменено несколькими словами, первое из которых затем будет рассматриваться как имя команды, а остальные станут ее аргументами.

Если строка не заключена в кавычки " " и не указан модификатор `:q`, то результаты подстановки переменных могут быть переданы для подстановки имен файлов и команд. Если переменная, значение которой включает несколько слов, заключена в двойные кавычки, она заменяется одним словом или частью одного слова, исходные слова в котором разделены пробелами. Если указан модификатор `:q`, переменная заменяется несколькими словами. Все эти слова будут разделены пробелами и заключены в кавычки, чтобы предотвратить дальнейшую подстановку имен файлов и команд.

Значение переменной оболочки можно представить несколькими способами. Если не указано обратное, ссылка не переменную, не заданную явно командой **set**, приводит к ошибке.

К следующим подстановкам можно применять модификаторы `:gh`, `:gt`, `:gr`, `:h`, `:r`, `:q` и `:x`. Если переменная указана в фигурных скобках ({ }), модификаторы должны находится внутри скобок. Для каждой подставляемой переменной можно указать только один модификатор **:**.

Элемент	Описание
$*Имя*	
${ *имя***}**	Будет заменено значением переменной Имя, слова в котором будут разделены пробелами. Фигурные скобки отделяют параметр *Имя* от следующих символов. Имена переменных оболочки должны начинаться с буквы и включать до 20 букв, цифр и знаков подчеркивания (_). Если переменной оболочки Имя не существует, но есть такая переменная среды, то подставляется ее значение. Модификаторы, указываемые после двоеточия, в данном случае недопустимы.
$*Имя***[***номер***]**	
${*Имя***[***Номер***]}**	Выбирает только некоторые из слов, составляющих значение переменной *Имя*. Номер также может быть объектом для подстановки; в нем может быть указано одно число, либо два числа через дефис (-). Первое слово значения переменной имеет номер 1. Если первое число опущено, по умолчанию оно считается равным 1. Если последнее число опущено, результат эквивалентен подстановке вместо него значения **$#***Имя*. Символ * выбирает все слова. Пустой результат, если второй аргумент указан в допустимом диапазоне или вовсе не указан, не приводит к ошибке.
$#*Имя*	
${# *Имя***}**	Заменяется числом слов в переменной *Имя*. Это значение можно использовать при подстановке **[***число***]**, как показано выше. Например, $*Имя*[$#*Имя*].
$0	Заменяется именем файла, из которого прочитан ввод команды. Если имя неизвестно, выдается сообщение об ошибке.

Элемент	Описание
$*номер*	
${*номер***}**	Эквивалентно `$argv[`*Номер*`]`.
$*	Эквивалентно `$argv[*]`.

Следующие подстановки не могут быть изменены модификаторами **:** (двоеточие):

Элемент	Описание
$?*имя*	
${?*имя***}**	Заменяется 1 (единицей), если переменная *Имя* установлена, и 0 (нулем), если переменная не установлена.
$?0	Заменяется 1 (единицей), если имя текущего файла ввода известно, и 0 (нулем), если имя этого файла неизвестно.
$$	Заменяется (десятичным) номером процесса родительской оболочки.
$<	Заменяется одной строкой из стандартного ввода без какой-либо обработки. Эта подстановка полезна в процедурах оболочки для чтения с клавиатуры.

Понятия, связанные с данным:

“Подстановка команд в оболочке C” на стр. 290
Подстановка команд означает, что оболочка выполняет указанную команду и заменяет эту команду ее выводом.

Подстановка имен файлов в оболочке C:

Оболочка C позволяет использовать подстановку имен файлов.

В оболочке C предусмотрено несколько функций для сокращения размера вводимых команд. Если слово содержит символы `*`, `?`, `[ ]` или `{ }`, либо начинается с тильды (~), то для этого слова будет выполнена подстановка. Оболочка C рассматривает это слово как шаблон и заменяет его списком имен файлов, соответствующих этому шаблону, отсортированных по алфавиту.

При этом применяется текущий порядок сортировки, заданный переменными среды *LC_COLLATE* и *LANG*. Если ни один из указанных шаблонов не соответствует ни одному из существующих файлов, возникает ошибка. Однако существование файлов для каждого из шаблонов не требуется. Подстановку по шаблону или расширение имени файла обозначают только символы подстановки `*`, `?` и `[ ]`. Тильда (~) и символы { } применяются для сокращения имен файлов.

Расширение имен файлов в оболочке C:

Символ `*` соответствует любой подстроке символов, в том числе и пустой.

Например, в каталоге с файлами:

```
a aa aax alice b bb c cc
```

команда **echo a*** напечатает имена всех файлов, начинающихся с `a`:

```
a aa aax alice
```

Примечание: При сравнении имени файла с шаблоном символы точка (`.`) и (`/`) должны сопоставляться явно.

Вопросительный знак (`?`) соответствует любому отдельному символу. Следующая команда:

```
ls a?x
```

покажет все состоящие из трех букв имена файлов, начинающиеся с `a` и заканчивающиеся на `x`:

```
aax
```

Для нахождения символов из определенного набора перечислите эти символы в квадратных скобках ([ ]). Следующая команда:

```
ls [abc]
```

покажет все файлы, имена которых равны одному из следующих символов:

```
a b c
```

Весь алфавит можно указать следующим образом: `[a-z]`. Символы, соответствующие этому шаблону, определяются текущим порядком сортировки.

Сокращения имен файлов в оболочке C:

Тильда (~) и открывающая фигурная скобка ({) применяются для сокращения имен файлов. Символ ~ в начале имени файла заменяется именем домашнего каталога. Отдельно стоящий символ ~ заменяется именем текущего домашнего каталога, которое берется из переменной оболочки *home*.

Например, следующая команда:

```
ls ~
```

покажет все файлы и подкаталоги каталога `$HOME`.

Если за тильдой следует имя, состоящее из букв, цифр и символов `-`, то оболочка подставит каталог `$HOME` пользователя с заданным именем.

Примечание: Если после тильды (~) стоит символ, отличный от буквы и косой черты (/), либо тильда указана не в начале слова, то подстановка не выполняется.

Для подстановки в имя файла нескольких различных слов укажите их в фигурных скобках ({ }). Например, шаблон `a{b,c,d}e` эквивалентен записи `abe ace ade`. Порядок слов при этом сохраняется. Подобная конструкция может быть вложенной. Например:

```
~source/s1/{oldls,ls}.c
```

будет заменено строкой

```
/usr/source/s1/oldls.c /usr/source/s1/ls.c
```

если домашний каталог пользователя **source** - это `/usr/source`. Аналогично, значение

```
../{memo,*box}
```

будет заменено на

```
../memo ../box ../mbox
```

Примечание: Слово `memo` не сортируется вместе с результатами подстановки слова `*box`. В некоторых случаях символы {, } и { } не заменяются.

Классы символов в оболочке C:

Для указания диапазонов символов можно применять классы.

Указанный ниже формат обозначает любой символ, принадлежащий указанному классу:

```
[:класс_символов:]
```

Следующие классы соответствуют функциям **ctype**:

Класс символов	Определение
alnum	Алфавитно-цифровые символы
alpha	Строчные и прописные буквы
cntrl	Управляющие символы
digit	Цифры
graph	Графические символы
lower	Строчные буквы
print	Печатаемые символы
punct	Знаки препинания
space	Пробел, символ горизонтальной табуляции, символ возврата каретки, символ новой строки, символ вертикальной табуляции или символ перевода страницы
upper	Прописные буквы
xdigit	Шестнадцатеричные цифры

Предположим, что текущий каталог содержит следующие файлы:

```
a aa aax Alice b bb c cc
```

Введите в командной строке оболочки C следующую команду:

```
ls [:lower:]
```

Оболочка C покажет все имена файлов, начинающиеся со строчных букв:

```
a aa aax b bb c cc
```

Дополнительные сведения о выражениях классов символов приведены в описании команды **ed**.

Переменные среды в оболочке C:

Некоторые переменные в оболочке C имеют специальное значение. Из них переменные *argv*, *cwd*, *home*, *path*, *prompt*, *shell* и *status* всегда задаются средой.

Для всех перечисленных переменных, кроме *cwd* и *status*, установка выполняется только один раз при инициализации оболочки. Затем эти переменные сохраняют свои значения, если только вы не измените их явно.

Команда **csh** копирует переменные среды *USER*, *TERM*, *HOME* и *PATH* соответственно в переменные *csh* variables, *user*, *term*, *home* и *path*. При любом изменении переменных оболочки выполняется обратное копирование их значений в переменные среды. Переменную *path* можно задать только в файле **.cshrc**, так как оболочка **csh** автоматически импортирует значение переменной path из среды и экспортирует ее в среду при каждом изменении.

Следующие переменные среды имеют особое значение:

Элемент	Описание
argv	Содержит аргументы, переданные сценарию оболочки. Из этой переменной подставляются позиционные параметры.
cdpath	Содержит список альтернативных каталогов для поиска подкаталогов, указанных в аргументах команд **chdir** и **cd**.
cwd	Содержит полное имя текущего каталога.
echo	Устанавливается флагом **-x** командной строки; выводит на экран каждую команду (вместе с аргументами) перед ее выполнением. Для внешних команд перед этим выполняются все подстановки. Встроенные команды выводятся на экран до выполнения подстановок, поскольку эти подстановки затем выполняются выборочно.
histchars	Задает строку, переопределяющую символы подстановки хронологии. Первый символ этой строки будет применяться в качестве символа подстановки хронологии вместо символа по умолчанию, !. Второй символ этой строки заменяет символ ^ быстрой подстановки. **Примечание:** Наличие в значении переменной **histchars** символов, применяемых в именах команд и файлов, может привести к нежелательным подстановкам.

Элемент	Описание
history	Содержит числовое значение, управляющее размером списка хронологии. Сохраняются все команды в указанном диапазоне. Слишком большое значение переменной *history* может привести к выходу за границу памяти. Независимо от значения этой переменной, оболочка C всегда сохраняет в списке хронологии последнюю команду.
home	Содержит имя вашего домашнего каталога, полученное из соответствующей переменной среды. Эта переменная подставляется вместо тильды (~) в именах файлов.
ignoreeof	Указывает оболочке игнорировать символ конца файла в данных, получаемых от рабочих станций. Это предотвращает случайное закрытие оболочки после чтения символа конца файла (Ctrl-D).
mail	Задает файлы, в которых оболочка будет проверять почту. Проверка почты выполняется при выводе приглашения после завершения команды, если прошел определенный интервал времени. Если время последнего обращения к одному из указанных файлов станет меньше времени последнего изменения файла, то оболочка выдаст сообщение `Почта в файле имя-файла`. Если первое слово в значении переменной *mail* - число, оно задает интервал проверки почты (в секундах); значение по умолчанию - 600 (10 минут). Если указано несколько файлов, и в одном из них обнаружена новая почта, то оболочка выдаст сообщение `Новая почта в файле имя_файла`.
noclobber	Устанавливает ограничения на перенаправление вывода, позволяющие быть уверенным в том, что файлы не будут случайно уничтожены или изменены.
noglob	Отменяет операции подстановки в именах файлов. Эта возможность применяется в сценариях оболочки, не работающих с именами файлов, либо когда получен список имен файлов, дальнейшая подстановка в котором нежелательна.
nonomatch	Указывает, что не следует показывать сообщение об ошибке, если в результате подстановки получилось имя файла, не соответствующее ни одному из существующих файлов; вместо этого возвращается простой шаблон. В любом случае, неправильные простые шаблоны приводят к ошибке.
notify	Указывает оболочке, что отправлять уведомления об изменении состояния заданий следует асинхронно. По умолчанию эта информация показывается перед выводом приглашения оболочки.
path	Задает каталоги, в которых выполняется поиск введенных команд. Пустое слово обозначает текущий каталог. Если переменная *path* не задана, при вводе команд следует указывать полные имена. Путь поиска по умолчанию (из файла `/etc/environment`, применявшегося во время входа в систему) выглядит следующим образом: `/usr/bin /etc /usr/sbin /usr/ucb /usr/bin/X11 /sbin` Обычно оболочка, при запуске которой не был указан ни флаг **-c**, ни флаг **-t**, после чтения файла `.cshrc` и при каждом изменении переменной *path* создает хэш-таблицу содержимого каталогов, перечисленных в переменной *path*. Если во время работы оболочки в эти каталоги были добавлены новые команды, необходимо ввести команду **rehash**. Иначе эти команды не будут найдены.
prompt	Задает строку, показываемую перед каждым чтением очередной команды с интерактивной рабочей станции. Если в этой строке встречается символ !, то он заменяется текущим номером события. Если в этой строке встречается символ ! , находится в одинарных или двойных кавычках, то перед ним необходимо указать символ \ (обратная косая черта). По умолчанию для пользователей без прав доступа root используется приглашение %. Для пользователя root по умолчанию выводится приглашение #.
savehist	Содержит числовое значение, управляющее числом записей в списке хронологии, сохраняемых при выходе из системы в файле `~/.history`. Будут сохранены все команды в указанном диапазоне. При запуске оболочка загружает файл `~/.history` в список хронологии, таким образом сохраняя хронологию предыдущего сеанса работы в системе. Увеличение значения переменной *savehist* замедляет запуск оболочки.
shell	Указывает файл, в котором находится оболочка C. Это значение применяется при запуске новых оболочек для файлов, выполнение которых разрешено, но которые не могут быть выполнены системой (сценариев оболочки). В эту переменную записывается полное имя файла оболочки C.
status	Содержит состояние возврата из последней запущенной команды. Если команда завершилась аварийно, к состоянию добавляется `0200`. Встроенные команды, завершившиеся неудачно, возвращают состояние 1. Успешно выполненные встроенные команды возвращают значение 0.
time	Управляет автоматическим определением времени выполнения команд. После каждой команды, выполнение которой заняло больше секунд CPU, чем указано в этой переменной (если она задана), показывается строка с информацией об использовании ресурсов. Дополнительная информация о формате вывода по умолчанию приведена в описании встроенной команды **time**.
verbose	Эта переменная, устанавливаемая с помощью флага **-v** командной строки, включает вывод на экран всех слов каждой команды после подстановки хронологии.

Управление заданиями в оболочке C:

С каждым процессом оболочка связывает определенный номер задания. Оболочка хранит таблицу текущих заданий и присваивает им небольшие целые числа.

При запуске задания в фоновом режиме с помощью символа & оболочка выдает строку следующего вида:

```
[1] 1234
```

Данная строка означает, что номер задания равен 1 и задание состоит из единственного процесса с ИД 1234. Для просмотра таблицы текущих заданий используйте команду **jobs**.

Задание, работающее в фоновом режиме, может заблокировать ввод при попытке чтения из рабочей станции. Кроме того, вывод фоновых заданий на рабочую станцию не отделяется от вывода других заданий.

Указать задание в оболочке можно несколькими способами. Например, введите символ % и имя задания. Именем может быть либо номер задания, либо имя команды запуска задания, если последнее имя уникально. Например, если процесс **make** выполняется в задании 1, то его можно указать как %1. Кроме того, на него можно сослаться, указав %make, если только у одного приостановленного задания имя начинается со слова make. Кроме того, указав значение

```
%?Строка
```

Последний вариант допустим только в случае, если такое задание единственно.

Изменение состояния процесса немедленно фиксируется оболочкой. Если задание блокируется и его дальнейшая работа становится невозможной, оболочка отправляет рабочей станции сообщение. Это сообщение выводится на экран только после нажатия клавиши Enter. Однако, если установлена переменная оболочки с именем *notify*, оболочка отправляет сообщения об изменении состояния фоновых заданий немедленно. Для выборочной установки этой переменной для отдельных процессов служит встроенная команда **notify**. По умолчанию команда **notify** применяется к текущему процессу.

Список встроенных команд оболочки C:

Ниже приведены примеры встроенных команд оболочки C.

Элемент	Описание
@	Показывает значение указанной переменной оболочки.
alias	Предназначена для просмотра всех или некоторых псевдонимов.
bg	Переводит текущее или указанное задание в фоновый режим.
break	Возобновляет выполнение после обработки оператора end ближайшего внешнего цикла **foreach** или **while**.
breaksw	Прерывает выполнение команды **switch**.
case	Определяет метку в команде **switch**.
cd	Предназначена для перехода из текущего каталога в указанный каталог.
chdir	Предназначена для перехода из текущего каталога в указанный каталог.
continue	Продолжает выполнение ближайшего внешнего цикла **foreach** или **while**, начиная с нового витка.
default	Метка действия по умолчанию в операторе **switch**.
dirs	Показывает стек каталогов.
echo	Записывает символьные строки в стандартный вывод оболочки.
else	Запускает команды, указанные за вторым оператором else в последовательности команд if (*Выражение*) then ...else if (*Выражение2*) then ... else ... endif.
end	Означает конец последовательности команд, которая начинается с оператора **foreach**.
endif	Запускает команды, указанные за вторым оператором then в последовательности команд if (*Выражение*) then ... else if (*Выражение2*) then ... else ... endif.
endsw	Обозначает конец последовательности команд **switch** (*Строка*) case *Строка* : ... breaksw default: ... breaksw endsw. Эта команда сравнивает каждую метку оператора case со значением переменной *Строка*. Выполнение программы продолжается после команды **endsw**, если была выполнена команда **breaksw**, или если не было найдено ни одного совпадения, и действие по умолчанию не задано.
eval	Считывает значения указанных переменных как входные данные оболочки, а затем выполняет получившиеся команды в среде текущей оболочки.

Элемент	Описание
exec	Запускает указанную команду вместо текущей оболочки.
exit	Завершает работу оболочки с кодом возврата, равным значению переменной оболочки status или значению указанного выражения.
fg	Переводит текущее или указанное задание в фоновый режим и возобновляет его выполнение, если оно было остановлено.
foreach	Поочередно присваивает переменной *Имя* значения из *Списка* и выполняет последовательность команд до оператора **end**.
glob	Предназначена для просмотра списка, в котором перечислены подстановки хронологии, переменных и имен файлов.
goto	Оператор перехода; выполнение продолжается с указанной строки.
hashstat	Показывает статистику работы хэш-таблицы поиска команд.
history	Показывает список хронологии.
if	Выполняет указанную команду, если значение заданного выражения равно true.
jobs	Показывает список активных заданий.
kill	Отправляет указанному заданию или процессу сигнал **TERM** (сигнал завершения) или сигнал, заданный в переменной *Сигнал*.
limit	Устанавливает ограничение на использование указанного ресурса для текущего процесса и его дочерних процессов.
login	Завершает работу начальной оболочки и запускает вместо нее экземпляр процесса `/usr/sbin/login`.
logout	Завершает работу начальной оболочки.
nice	Устанавливает приоритет команд, выполняющихся в оболочке.
nohup	В результате выполнения этой команды сигнал hangup будет игнорироваться до завершения процедуры.
notify	В результате выполнения этой команды оболочка будет асинхронно отправлять уведомления об изменении состояния текущего или указанного задания.
onintr	Управляет реакцией оболочки на прерывания.
popd	Считывает стек каталогов и переходит к верхнему каталогу.
pushd	Помещает в стек новый список каталогов.
rehash	Предназначена для обновления внутренних хэш-таблиц, в которых содержится список файлов из каталогов, указанных в переменной оболочки path.
repeat	Выполняет заданную команду указанное число раз с тем же ограничением, что и команда **if**.
set	Показывает список значений всех переменных оболочки.
setenv	Изменяет значение указанной переменной среды.
shift	Для заданной переменной выполняет сдвиг влево.
source	Показывает исходный текст команды, заданной в параметре *Имя*.
stop	Завершает работу текущего или указанного фонового задания.
suspend	Действие команды аналогично отправке оболочке сигнала **STOP**.
switch	Запускает последовательность команд **switch** (*Строка*) `case` *Строка* `:` ... `breaksw default:` ... `breaksw endsw`. Эта команда сравнивает каждую метку оператора case со значением переменной *Строка*. Если ни одно совпадение не найдено, то выполняется действие, связанное с меткой default.
time	Показывает суммарное время работы оболочки и ее дочерних процессов.
umask	Определяет права доступа к файлу.
unalias	Удаляет все псевдонимы с именами, соответствующими *Шаблону*.
unhash	Отменяет применение внутренней хэш-таблицы для нахождения запускаемых программ.
unlimit	Снимает ограничение на использование ресурсов.
unset	Удаляет все переменные с именами, соответствующими *Шаблону*.
unsetenv	Удаляет все переменные среды с именами, соответствующими *Шаблону*.
wait	Ожидает завершения всех фоновых заданий.
while	Последовательность команд, расположенная между операторами `while` и `end`, будет выполняться до тех пор, пока значение *Выражения* отлично от нуля.

Ниже приведена связанная информация:

Оболочка Korn

Команды **ksh** и **stty**.

Команды оболочки Korn **alias**, **cd**, **export**, **fc**, **getopts**, **read**, **set** и **typeset**.

Файл `/etc/passwd`.

оболочка Bourne

Команда **bsh**, **Rsh** или **login**.

Специальная команда оболочки Bourne **read**.

Функция **setuid**, **setgid**.

Особый файл `null`.

Формат файла `environment`, `profile`.

Оболочка C

Команды **csh**, **ed**.

Встроенные команды оболочки C **alias**, **unalias**, **jobs**, **notify** и **set**.

Понятия, связанные с данным:

“Оболочка C” на стр. 274
Оболочка C - это интерактивный интерпретатор команд и командный язык программирования. Ее синтаксис похож на синтаксис языка программирования C.

“Встроенные команды оболочки C” на стр. 284
Обычно встроенные команды выполняются внутри оболочки. Но если встроенная команда встретится как любой компонент конвейера, кроме последнего, для нее будет создана отдельная оболочка.

Обработка сигналов в оболочке C:

Обычно оболочка C игнорирует сигналы QUIT. Автономные задания не реагируют на сигналы с клавиатуры (**INTERRUPT** , **QUIT** и **HANGUP**).

Значения остальных сигналов наследуются от родительского процесса. Вы можете управлять обработкой сигналов **INTERRUPT** и **TERMINATE** в процедурах оболочки с помощью **onintr**. Оболочки входа в систему могут игнорировать или обрабатывать сигналы **TERMINATE** в зависимости от своей настройки. Другие оболочки передают сигналы **TERMINATE** своим дочерним процессам. Сигналы **INTERRUPT** игнорируются во время чтения оболочкой файла `.logout`.

Команды оболочки C:

Простая команда - это последовательность слов, разделенных пробелами или знаками табуляции. *Слово* - это последовательность символов и цифр, не содержащая пробелов вне кавычек.

Кроме того, следующие символы и удвоенные символы также рассматриваются как отдельное слово, если применяются в качестве разделителей или признаков конца команды.

```
&       |       ;
&&      ||      <<          > >
<       >       (           )
```

Эти специальные символы могут быть частью других слов. Однако если перед ними указать символ \, то оболочка не будет интерпретировать такие символы как специальные. Строки, заключенные в символы ' ' или " " (пары совпадающих кавычек) или обратные кавычки, также могут быть частью слов. Пробелы, символы табуляции и специальные символы не рассматриваются как отдельные слова, если не заключены в кавычки. В текст можно вставить символ новой строки, указав перед ним символ \.

Первым словом простой командной последовательности (под номером 0) обычно является имя команды. Остальные слова, за некоторыми исключениями, передаются этой команде как аргументы. Если в команде указан исполняемый файл, в котором хранится откомпилированная программа, оболочка запускает его. Если файл отмечен как исполняемый, но не является откомпилированной программой, оболочка рассматривает этот файл как сценарий. В этом случае запускается другой экземпляр оболочки (подоболочка), читающий указанный файл и выполняющий его команды.

Встроенные команды оболочки C:

Обычно встроенные команды выполняются внутри оболочки. Но если встроенная команда встретится как любой компонент конвейера, кроме последнего, для нее будет создана отдельная оболочка.

Примечание: При вводе команды в приглашении оболочки C система сначала ищет встроенную команду. Если встроенная команда не существует, выполняется поиск команды системного уровня в каталогах, указанных в переменной оболочки *path*. Имена некоторых встроенных команд оболочки C совпадают с именами команд операционной системы. Однако эти команды не обязательно работают одинаково. Для получения дополнительной информации о той или иной команде, ознакомьтесь с ее описанием.

Если первая строка запускаемого сценария начинается с `#!/`*`Полное_имя_оболочки`*, то оболочка C для выполнения этого сценария запускает указанную оболочку. В противном случае, запускается оболочка по умолчанию (связанная с `/usr/bin/sh`). Оболочка по умолчанию может не поддерживать встроенные команды оболочки C. Для того чтобы сценарий выполнялся оболочкой C, в первой строке этого сценария должно быть указано `#!/usr/bin/csh`.

Ссылки, связанные с данной:

“Список встроенных команд оболочки C” на стр. 281
Ниже приведены примеры встроенных команд оболочки C.

Описания команд оболочки C:

Оболочка C содержит следующие встроенные команды.

Элемент	Описание
alias [*Имя* [*Список_слов*]]	Если параметры не указаны, то команда показывает все определенные псевдонимы. С параметром *Имя* команда показывает все псевдонимы этого имени. Если указан параметр *Список_слов*, команда присваивает значение *Список_слов* псевдониму *Имя*. В качестве *Имени* нельзя указывать команды `alias` и `unalias`.
bg [*%Задание* ...]	Переводит текущее или указанное в параметре *Задание* задание в фоновый режим, возобновляя его работу, если она была остановлена.
break	Возобновляет выполнение после обработки оператора `end` ближайшего внешнего цикла **foreach** или **while**.
breaksw	Прерывает выполнение команды **switch**; возобновляет выполнение после оператора **endsw**.
case *Метка*:	Задает *Метку* для команды **switch**.
cd[*Имя*]	Эквивалентна команде **chdir** (см. ниже).
chdir[*Имя*]	Изменяет текущий каталог на указанный в параметре *Имя*. Если *Имя* не указано, команда изменяет текущий каталог на ваш домашний. Если *Имя* - не подкаталог текущего каталога и не начинается с `/`, `./` или `../`, то оболочка просматривает все элементы переменной среды ***cdpath*** в поисках подкаталога с именем *Имя*. Если *Имя* - это имя переменной среды, значение которой начинается с символа косой черты (/), то оболочка пытается рассматривать это значение как каталог. Команда **chdir** эквивалентна команде **cd**
continue	Продолжает выполнение после оператора `end` ближайшей команды **while** или **foreach**.
default:	Отмечает вариант по умолчанию (`default`) в операторе `switch`. Оператор **default** должен находиться после всех остальных меток `case`.
dirs	Показывает стек каталогов.
echo	Записывает символьные строки в стандартный вывод оболочки.

Элемент	Описание
else	Запускает команды, указанные за вторым оператором `else` в последовательности команд `if` (*Выражение*) `then` ...`else if` (*Выражение2*) `then` ... `else` ... `endif`. **Примечание:** Оператор **иначе** является встроенной командой **csh**, использующей сначала оператор **если(***expr***), а затем** ..**иначе** ...**конец, если**. Если **(***expr***)** верно, то выполняются все команды до оператора **иначе**. Если **(***expr***)** ложно, то выполняются команды между операторами **иначе** и **конец, если**. Все отдельные цитаты берутся дословно и не интерпретируются.
конец	Присваивает переменной *Имя* последовательно элементы из списка, указанного в параметре *Список*, и запускает последовательность *Команд*, расположенную между оператором **foreach** и соответствующим ему **end**. Операторы **foreach** и **end** должны находиться на отдельных строках. Оператор `continue` начинает следующий цикл, оператор `break` прерывает цикл. При вводе команды **foreach** с терминала оболочка C выдает приглашение `?` для ввода *Команд*. Команды, введенные в приглашении `?` и выполненные внутри цикла, не добавляются в список хронологии.
endif	Если *Выражение* истинно, запускает *Команды*, следующие за первым оператором `then`. Иначе, если *Выражение2* из оператора `else if` истинно, запускает *Команды*, следующие за вторым оператором `then`. Если *Выражение2* после `else if` ложно, запускает *Команды*, следующие за последним `else`. Допустимо любое количество пар `else if`. При этом необходим только один завершающий оператор `endif`. Раздел, начинающийся с `else`, необязателен. Слова `else` и `endif` могут стоять только в начале строк. Раздел `if` должен либо быть единственным в строке ввода, либо следовать за командой `else`.
endsw	Сравнивает метки **case** со значением переменной *Строка*. *Строка* - это команда или имя файла с выполненными подстановками. В метках **case** переменной длины следует применять символы сравнения с шаблоном `*`, `?` и `[ . . . ]`. Если ни одна из меток перед **default** не будет выбрана, управление будет передано на метку **default**. Метки **case** и метка **default** должны находиться в начале строк. Команда **breaksw** продолжает выполнение после команды **endsw**. В ее отсутствие управление будет передаваться по меткам **case** и **default**, как в языке C. Если ни одна из меток не будет выбрана и метка **default** отсутствует, то выполнение будет продолжено после команды **endsw**.
eval *Параметр . . .*	Рассматривает значение *Параметр* как входные данные для оболочки и выполняет соответствующие команды внутри текущей оболочки. Эта команда часто применяется для запуска других команд, имена которых получены в результате выполнения команды или подстановки переменных, поскольку анализ выполняется до подстановки.
exec *Команда*	Запускает указанную *Команду* вместо текущей оболочки.
exit (*Выражение***)**	Завершает работу оболочки, используя значение переменной среды ***status*** (если не задано *Выражение*), либо значение *Выражения*.
fg [%*Задание ...*]	Переводит текущее или указанное в параметре *Задание* задание в интерактивный режим, возобновляя его работу, если она была остановлена.
foreach *Имя* (*Список*) *Команда. . .*	Поочередно присваивает переменной *Имя* значения из *Списка* и выполняет последовательность команд до оператора **end**.
glob *Список*	Показывает *Список* с учетом подстановки хронологии, переменных и имен файлов. Помещает между словами символ с нулевым кодом и не добавляет в конец строки возврат каретки.
goto *Слово*	Продолжает выполнение со строки, указанной в параметре *Слово*. *Слово* - это имя файла и команда, в результате подстановки которых получается строка в форме *Метка:* . Оболочка просматривает входные данные на максимально возможную длину для поиска строки в форме *Метка***:**, перед которой может стоять несколько пробелов или знаков табуляции.
hashstat	Показывает статистику работы хэш-таблицы поиска команд.
history [**-r** | **-h**] [*n*]	Показывает список хронологии. Самые старые события показываются первыми. Если указать число в параметре *n*, будет показано только n последних событий. С флагом **-r** события будут показаны в обратном порядке - начиная с самых новых. С флагом **-h** список хронологии выводится без нумерации. При этом могут быть созданы файлы, допустимые в качестве входных для команды **source** с флагом **-h**.

Элемент	Описание
if (*Выражение*) *Команда*	Запускает указанную *Команду* (с аргументами), если *Выражение* истинно. Подстановка переменных в *Команде* выполняется в самом начале, одновременно с оператором `if`. *Команда* должна представлять собой простую команду (а не конвейер или список команд, даже заключенный в скобки). **Примечание:** Перенаправление ввода и вывода выполняется даже в том случае, если *Выражение* ложно и *Команда* не выполняется.
jobs [**-l**]	Показывает список активных заданий. Команда **jobs** с флагом **-l** (строчная *L*) кроме номеров и имен заданий показывает идентификаторы процессов.
kill -l | [[-*Сигнал*] % *Задание*...|*ИД-процесса*...]	Отправляет сигнал **TERM** (завершить) или другой *Сигнал* указанному *Заданию* или процессу *ИД-процесса*. Сигналы указываются либо по номеру, либо по имени (согласно файлу `/usr/include/sys/signal.h`, но без префикса **SIG**). Если указан флаг **-l** (строчная *L*), то будет показан список сигналов.
limit [**-h**] [*Ресурс* [*Максимум*]]	Ограничивает использование указанного ресурса для текущего процесса и всех процессов, им созданных. Ограничения на использование ресурсов хранятся в файле `/etc/security/limits`. Вы можете ограничить время центрального процессора (CPU), размер файлов, размер данных, размер дампа памяти и размер памяти. Максимальные значения задаются командой **mkuser** при добавлении пользователя в систему. Эти значения затем можно изменить командой **chuser**. Все ограничения делятся на гибкие и жесткие. Пользователи могут увеличивать гибкие ограничения вплоть до жестких. Для увеличения жесткого ограничения или для указания такого гибкого ограничения, которое будет превышать жесткое, необходимы права пользователя root. С флагом **-h** будут показаны жесткие ограничения, без него - гибкие. Если параметр *Максимум* не указан, команда **limit** показывает текущее ограничение на использование указанного ресурса. Если параметр *Ресурс* также не указан, команда **limit** показывает текущие ограничения на использование всех ресурсов. Дополнительная информация об управляемых командой **limit** ресурсах приведена в описании функций**getrlimit**, **setrlimit** или **vlimit** в книге *Technical Reference: Base Operating System and Extensions, Volume 1*. Параметр *Максимум* для времени использования CPU указывается в формате *чч*`:`*мм*`:`*сс*. Для других ресурсов *Максимум* указывается в виде числа с плавающей точкой или целого, которое может быть дополнено масштабным коэффициентом. Допустимы следующие масштабные коэффициенты: k (килобайты, 1 килобайт=1024 байт), m (мегабайты) и b (блоки - единицы, применяемые функцией **ulimit**; см. руководство *Technical Reference: Base Operating System and Extensions, Volume 2*). Если масштабный коэффициент не указан, то считается, что размер всех ресурсов задан в килобайтах. Для имен ресурсов и масштабных коэффициентов можно применять однозначные сокращения. **Примечание:** Данная команда ограничивает объем физической памяти, доступной процессу, только в том случае, если ее не хватает другим активным процессам.
login	Завершает работу начальной оболочки и заменяет ее экземпляром команды `/usr/bin/login`. Это один из способов выхода из системы (поддерживается в целях совместимости с командами оболочек **ksh** и **bsh**).
logout	Завершает работу начальной оболочки. Данная команда необходима в случае, если установлена опция `ignoreeof`.
nice [**+***n*] [*Команда*]	Без параметров устанавливает приоритет выполнения команд в оболочке равным 24. С флагом **+***n* увеличивает приоритет на указанное значение (n). Если с параметром **+***n* указана *Команда*, запускает *Команду* с приоритетом 24 плюс указанное значение. Пользователь с правами root может вызывать команды **nice** с отрицательным числом в параметре. *Команда* всегда запускается в отдельной оболочке с теми же ограничениями, что и команды в простом операторе `if`.
nohup [*Команда*]	Если *Команда* не указана, сигналы **hangups** игнорируются до завершения сценария. Если *Команда* указана, сигналы **hangups** игнорируются во время работы этой *Команда*. Для запуска конвейера или списка команд поместите их в сценарий оболочки, разрешите для него выполнение и укажите этот сценарий в качестве *Команды*. Все процессы, запущенные в фоновом режиме со знаком амперсанда (`&`), уже защищены от получения сигнала **hangup** при выходе пользователя из системы. Однако этим процессам все равно можно отправить сигнал **hangup** явно, если для них не был запущен оператор **nohup**.

Элемент	Описание
notify [**%***Задание*...]	Указывает заданию отправлять уведомления об изменении состояния текущего или указанного *Задания* асинхронно. Обычно оболочка отправляет уведомления перед выводом приглашения. При задании переменной оболочки ***notify*** это свойство включается автоматически.
onintr [**-** | *Метка*]	Управляет реакцией оболочки на прерывания. Если аргументы не указаны, восстанавливает реакцию оболочки на прерывания по умолчанию, при которой сценарии оболочки завершаются, и управление передается командной строке. Если указан флаг **-**, все прерывания игнорируются. Если указана *Метка*, оболочка выполняет команду `goto` *Метка*, когда получает прерывание или когда ее подпроцесс завершается прерыванием. В любом случае, если оболочка работает автономно, и все прерывания игнорируются, то все формы команды **onintr** не выполняют никаких действий. Прерывания будут и дальше игнорироваться оболочкой и всеми запущенными командами.
popd [**+***n*]	Выталкивает имя каталога из стека и переходит в этот каталог. Если указан параметр **+*n***, команда извлекает *n*-й элемент стека. Элементы нумеруются от вершины стека, начиная с нуля.
pushd [**+***n*|*Имя*]	Без аргументов меняет местами два верхних элемента стека каталогов. С параметром *Имя* команда сохраняет текущий каталог в стеке каталогов (указанном в переменной среды ***cwd***) и переходит в новый каталог. Если указано **+*n***, команда получает из стека *n*-й каталог и переходит в него. Элементы стека каталогов нумеруются от вершины стека, начиная с нуля.
rehash	Повторно создает внутреннюю хэш-таблицу содержимого каталогов, указанных в переменной оболочки ***path***. Это необходимо, если после входа в систему в каталоги ***path*** были добавлены новые команды. Команда **rehash** применяется только в том случае, если команды были добавлены в собственные каталоги пользователя или если содержимое системных каталогов было изменено другим пользователем.
repeat *Число Команда*	Запускает *Команду*, удовлетворяющую тем же ограничениям, что и команда в простом операторе `if`, указанное *Число* раз. **Примечание:** Перенаправления ввода-вывода выполняются ровно один раз, даже если *Число* равно 0.
set [[*Имя*[*n*]] [**=** *Слово*]] | [*Имя* = (*Список*)]	Без аргументов показывает значения всех переменных оболочки. Переменные, значение которых состоит из нескольких слов, показываются в виде списка, заключенного в скобки. Если указано только *Имя*, оболочка C присваивает переменной *Имя* пустую строку. Иначе переменной *Имя* присваивается значение *Слово* или *Список* слов. Если указано число *n*, то *n*-му компоненту переменной *Имя* присваивается значение *Слово*; *n*-й компонент должен существовать. Во всех случаях в команде и имени файла выполняются все подстановки. Для изменения нескольких значений в одной команде **set** аргументы могут повторяться. Однако подстановка переменных во всех аргументах выполняется до присваивания значений.
setenv*Имя Значение*	Присваивает переменной среды с именем *Имя* строку символов *Значение*. Наиболее часто используемые переменные среды - ***USER***, ***TERM***, ***HOME*** и ***PATH*** - автоматически импортируются и экспортируются в переменные ***user***, ***term***, ***home*** и ***path*** оболочки C. К ним нет необходимости применять команду **`setenv`**.
shift [*Variable*]	Сдвигает элементы переменной оболочки ***argv*** или другой *Переменной* влево. Если переменная оболочки **argv** (или указанная *Переменная*) не установлена или длина ее значения менее одного слова, выдается сообщение об ошибке.
source[**-h**] *Имя*	Считывает команды, записанные в файле *Имя*. Команды **source** могут быть вложены друг в друга. Однако при слишком большой глубине вложенности у оболочки могут закончиться дескрипторы файлов. Ошибка в команде **source** любого уровня вложенности завершает все команды **source**. Обычно входные данные во время выполнения команды **source** не добавляются в список хронологии. Команда с флагом **-h** помещает команды в список хронологии, не выполняя их.
stop [**%***Задание* ...]	Останавливает текущее или указанное *Задание*, работающее в фоновом режиме.
suspend	Приостанавливает оболочку, имитируя получение сигнала **STOP**.
switch (*строка*)	Запускает последовательность команд **switch** (*Строка*) **case** *Строка* **:** ... **breaksw default:** ... **breaksw endsw**. Эта команда сравнивает каждую метку оператора case со значением переменной *Строка*. Если ни одно совпадение не найдено, то выполняется действие, связанное с меткой default.

Элемент	Описание

time [*Команда*]	Команда **time** автоматически определяет время выполнения команд. Если *Команда* не указана, **time** показывает отчет о времени, использованном данной оболочкой и ее потомками. Если *Команда* указана, будет показано время ее выполнения. Формат отчета об использованном времени определяется значением переменной оболочки ***time***. При необходимости, для показа статистики по времени после завершения команды создается дополнительная оболочка.
	
	Ниже приведен пример применения команды **time** совместно с командой **sleep**:
	`time sleep`
	
	Вывод этой команды выглядит примерно следующим образом:
	`0.0u 0.0s 0:00 100% 44+4k 0+0io 0pf+0w`
	
	Ниже приведено описание полей вывода:
	1 Время, затраченное CPU на пользовательский процесс, в секундах.
	2 Время, затраченное CPU на выполнение функций ядра для пользовательского процесса.
	3 Внешнее время, прошедшее при выполнении команды.
	4 Полное пользовательское время (1) плюс системное время (2), в процентах от прошедшего времени (3).
	5 Средний объем задействованной общей памяти плюс средний объем частного пространства данных, в килобайтах.
	6 Число операций блочного ввода и вывода.
	7 Число страничных сбоев плюс число операций подкачки.

umask [*Значение*]	Определяет права доступа к файлу. Это *Значение*, вместе с правами доступа создающего процесса, задает права доступа к создаваемому файлу. Значение по умолчанию: 022. Если *Значение* не указано, будет показано текущее значение.
unalias *|*Шаблон*	Удаляет все псевдонимы с именами, соответствующими *Шаблону*. Для удаления всех псевдонимов введите **unalias ***. Отсутствие псевдонимов не приводит к ошибке.
unhash	Отменяет применение внутренней хэш-таблицы для нахождения запускаемых программ.
unlimit [**-h**][*Ресурс*]	Отменяет ограничение на использование *Ресурса*. Если *Ресурс* не указан, удаляются все ограничения на использование ресурсов. Список имен *Ресурсов* приведен в описании команды **limit**. С флагом **-h** команда удаляет соответствующие жесткие ограничения. Жесткие ограничения могут быть изменены только пользователем с правами доступа root.
unset *|*Шаблон*	Удаляет все переменные с именами, соответствующими *Шаблону*. Для удаления всех переменных введите **unset ***. Если ни одна переменная не задана, это не приводит к ошибке.
unsetenv*Шаблон*	Удаляет все переменные среды, имена которых соответствуют *Шаблону*. (См. также встроенную команду **setenv**)
wait	Ожидает завершения всех фоновых заданий. В интерактивном задании ожидание может быть прервано сигналом **INTERRUPT** (обычно вызываемым комбинацией клавиш Ctrl-C). После этого оболочка показывает имена и номера всех оставшихся заданий.
while (*Выражение*) *Команда*. . . `end`	Выполняет *Команды*, указанные между операторами **while** и соответствующим ему **end**, пока *Выражение* не примет нулевое значение. Для завершения цикла служит команда **break**; для продолжения цикла - **continue**. Операторы **while** и **end** должны находится на отдельных строках. При вводе с терминала после команды **while** (*Выражение*) будет показано приглашение, похожее на приглашение команды **foreach**.

Элемент	Описание
@ [*Имя*[*n*] = *Выражение*]	Без аргументов показывает значения всех переменных среды. С аргументами присваивает переменной *Имя* значение *Выражения*. Если выражение содержит символы <, >, & или |, они должны быть заключены в скобки. Если указано число *n*, значение *Выражения* присваивается *n*-му компоненту переменной *Имя*. Переменная *Имя* и ее *n*-ый компонент должны существовать. Допустимы операторы языка C, такие как *= и +=. Пробелы, отделяющие *Имя* от оператора присвоения, необязательны. Однако для отделения компонентов *Выражения*, которые иначе читались бы как одно слово, пробелы необходимы. Специальные постфиксные операторы двойной плюс (++) и двойной минус (--) увеличивают и уменьшают значение переменной *Имя*.

Выражения и операторы оболочки C:

Встроенная команда **@** и операторы **exit**, **if** и **while** допускают использование выражений, соответствующих правилам и приоритету операций языка C.

Допустимы следующие операторы:

Оператор	Значение
()	изменение порядка действий
~	поразрядное отрицание
!	отрицание
*/ %	умножение, деление и взятие остатка
+ -	сложение и вычитание
<< > >	Сдвиг влево, сдвиг вправо
<= >= < >	сравнение
== != =~ !~	сравнение строк со строками и шаблонами
&	поразрядное И
^	поразрядное исключающее ИЛИ
|	поразрядное ИЛИ
&&	логическое И
||	логическое ИЛИ

В предыдущем списке операторы перечислены в порядке уменьшения приоритета (сверху вниз, слева направо).

Примечание: Операторы + и - вычисляются справа налево. Например, a + b - c вычисляется как

```
a + (b - c)
```

а не как

```
(a + b) - c
```

Операторы ==, !=, =~ и !~ сравнивают аргументы как строки, все остальные операторы работают с числами. Операторы =~ и !~ действуют аналогично операторам == и != за исключением того, что в правой части может стоять *шаблон*, с которым сравнивается левая часть. Эта возможность в некоторых случаях избавляет от необходимости применения в процедурах оболочки оператора **switch**.

Также поддерживаются логические операторы **или** (||) и **и** (&&). С их помощью можно проверить, относится ли значение к указанному диапазону, как показано ниже:

```
if ($#argv > 2 && $#argv < 7) then
```

В этом примере число аргументов должно быть больше двух, но меньше семи.

Строки, начинающиеся с нуля (0), рассматриваются как восьмеричные числа. Пустые или отсутствующие аргументы считаются нулевыми. Все результаты вычисления выражений сохраняются в строках в виде

десятичных чисел. Обратите внимание, что два компонента выражения могут быть объединены в одном слове. Кроме некоторых операторов, важных для анализа (`& | < > ( )`), компоненты выражения должны быть разделены пробелами.

В качестве простых операндов в выражениях также допустимы команды, заключенные в `( )`, и запросы к файлам в форме (**-оператор** *Имя_файла*), где **оператор** - один из следующих символов:

Элемент	Описание
r	Доступ для чтения
w	Доступ для записи
x	Доступ для выполнения
e	Существование
o	Владелец
z	Нулевой размер
f	Простой файл
d	Каталог

После выполнения всех подстановок система проверяет, есть ли у текущего пользователя указанные права доступа к заданному файлу *Имя_файла*. Если файл *Имя_файла* не существует или недоступен, все запросы возвращают значение `ложь (0)`. Если команда выполняется успешно, запрос возвращает истину `true(1)`. Если команда выполняется успешно, запрос возвращает значение `ложь (0)`. Для более подробного разбора состояния запустите команду вне выражения, затем проанализируйте переменную оболочки *status*.

Подстановка команд в оболочке C:

Подстановка команд означает, что оболочка выполняет указанную команду и заменяет эту команду ее выводом.

Для выполнения подстановки команд в оболочке C заключите текст этой команды в обратные апострофы (` `). Обычно оболочка разбивает вывод команды на отдельные слова, разделенные пробелами, знаками табуляции символами переноса строки. Затем исходная команда заменяется полученным выводом.

В следующем примере обратные апострофы (` `) вокруг команды **date** указывают на необходимость подстановки вывода этой команды:

```
echo Текущие дата и время: `date`
```

Вывод этой команды может выглядеть следующим образом:

```
Текущие дата и время: суббота 17 апреля 2004 г. 19.38.41
```

Оболочка C выборочно выполняет подстановку в аргументах встроенных команд оболочки. Это означает, что оболочка не подставляет невычисленные части выражений. Для внешних команд оболочка подставляет имя команды отдельно от списка аргументов. Подстановка выполняется в отдельной оболочке после перенаправления ввода и вывода.

Если текст команды заключен в двойные кавычки (`" "`), оболочка разделяет слова только по символам переноса строки, оставляя пробелы и символы табуляции внутри слов. Во всех случаях заключительный символ переноса строки не создает нового слова.

Понятия, связанные с данным:

“Подстановка переменных в оболочке C” на стр. 275
Оболочка C поддерживает набор переменных, каждая из которых хранит в качестве значения одно или несколько слов. Некоторые из этих переменных задаются оболочкой или используются ей. Например, переменная *argv* представляет собой список параметров оболочки, причем на любое слово этого списка можно задать ссылку специальным образом.

Выполнение не встроенных в оболочку C команд:

Если оболочка C определяет, что введенная команда не является встроенной, она пытается запустить эту команду с помощью функции **execv**.

Каждое слово в переменной оболочки *path* интерпретируется как каталог для запуска команды. Если ни флаг **-c**, ни флаг **-t** не были указаны, оболочка хранит имена файлов из этих каталогов во внутренней хэш-таблице. Оболочка вызывает функцию **execv** только тогда, когда существует вероятность наличия команды в одном из каталогов. Если выключить этот механизм командой **unhash** или указать для среды флаг **-c** или **-t**, к указанному имени команды будет добавлен каталог для образования полного пути. Эта операция в любом случае выполняется для каждого компонента переменной *path*, не начинающегося с (/). Затем оболочка пытается запустить команду.

Команды, заключенные в скобки, всегда выполняются в отдельной оболочке. Например:

```
(cd ; pwd) ; pwd
```

показывает домашний каталог, не меняя текущий каталог, а команда

```
cd ; pwd
```

изменит текущий каталог на домашний. Наиболее часто команды в скобках применяются для того, чтобы избежать влияния команды **chdir** на текущую оболочку.

Если разрешено выполнение файла, но файл не является двоичным исполняемым файлом, оболочка рассматривает его как командный файл, для выполнения которого запускает новую оболочку.

Если в оболочке определены псевдонимы, их слова подставляются в начало списка аргументов для образования команды оболочки. Первым словом псевдонима должен быть полный путь к оболочке.

Подстановка хронологии в оболочке C:

Подстановка хронологии позволяет получить новые команды путем замены слов в предыдущих командах. С помощью подстановки хронологии можно повторить команду, перенести аргументы предыдущей команды в текущую команду или исправить орфографические ошибки в предыдущей команде.

Подстановки хронологии начинаются с символа ! и могут располагаться в любом месте командной строки. Однако подстановки не могут быть вложенными (другими словами, подстановка хронологии не может содержать подстановку хронологии). Для того чтобы отменить специальное значение символа !, укажите перед ним символ обратной косой черты (\). Кроме того, если символ ! указан перед символом пробела, табуляции, новой строки, = или (, то подстановка хронологии не выполняется.

Подстановка хронологии выполняется и для тех строк, которые начинаются с символа ^. Перед обработкой строки, содержащей подстановки хронологии, оболочка выводит ее на экран рабочей станции.

Понятия, связанные с данным:

“Подстановка псевдонимов в оболочке C” на стр. 274
Псевдоним - это имя, присвоенное команде или командной строке. Оболочка C позволяет создавать псевдонимы и применять их в качестве команд. Для этого оболочка хранит список псевдонимов, определенных пользователем.

Списки хронологии в оболочке C:

В список хронологии заносятся команды длиной в одно или несколько слов, считанные оболочкой из командной строки. Подстановка хронологии заключается в том, что из строк сохраненных команд выбираются последовательности слов, которые записываются в стандартный ввод.

Размер списка хронологии задается в переменной оболочки *history*. Значение переменной *history* можно указать в файле `.cshrc` или в параметре встроенной команды **set**. Независимо от значения переменной *history*, последняя выполненная команда всегда сохраняется. В списке хронологии команды последовательно нумеруются, начиная с 1. Вывод встроенной команды **history** схож со следующим примером:

```
9 write michael
10 ed write.c
11 cat oldwrite.c
12 diff *write.c
```

Будут показаны команды и номера соответствующих событий (порядковые номера). Номер события указывается слева от команды и показывает, когда была введена команда сравнительно с другими командами в хронологии. Для ссылки на событие не обязательно применять номер события, однако номер текущего события может отображаться в приглашении системы. Для этого укажите символ `!` в строке приглашения, заданной в переменной среды *PROMPT*.

Полная ссылка на хронологию содержит указание события, ссылку на слово и один или несколько модификаторов в следующем формате:

```
Событие[.]Слово:Модификатор[:Модификатор] . . .
```

Примечание: Можно изменить только одной слово. Строка не может содержать пробелы.

В предыдущем примере вывода команды **history** номер текущего события равен 13. В этом примере можно задать следующие ссылки на предыдущие события:

Элемент	Описание
`!10`	Событие номер 10.
`!-2`	Событие номер 11 (номер текущего события минус 2).
`!d`	Команда, начинающаяся с символа `d` (номер события 12).
`!?mic?`	Команда, содержащая подстроку `mic` (номер события 9).

Такие выражения просто переносят команды, связанные с указанными событиями, без какого-либо изменения. Специальное выражение `!!` представляет собой ссылку на предыдущую команду; отдельная команда **!!** в строке ввода запускает предыдущую команду.

Спецификация событий в оболочке C:

Для выбора отдельных слов из команд, связанных с событиями, укажите событие с символом `:` и одну из следующих ссылок на слово (слова в строке ввода последовательно нумеруются с 0)

Элемент	Описание
0	Первое слово (имя команды)
n	*n*-ый аргумент
^	Первый аргумент
$	Последний аргумент
%	Слово, найденное при поиске предыдущей строки ?*строка*? ?
x-y	Интервал слов с порядковыми номерами от *x* до *y*
-y	Интервал слов с порядковыми номерами от 0 до *y*
*	Все аргументы или пустая строка, если событию соответствует только одно слово (имя команды)
*n**	аргумент с номером *x*, считая с конца
x-	То же самое, что и *x**, за исключением последнего аргумента

Если ссылка на слово начинается с символа ^, $, *, - или %, то двоеточие между событием и ссылкой на слово можно опустить. За необязательной ссылкой на слово можно указать последовательность следующих модификаторов, указав перед каждым из них двоеточие:

Элемент	Описание
h	Удаляет расширение в полном имени.
r	Удаляет последние компоненты *.xxx*, оставляя только имя.
e	Удаляет все, кроме расширения *.xxx*.
s/*Старое_слово*/*Новое_слово*/	Вместо значения переменной *Старое_слово* подставляет значение переменной *Новое_слово*.

В качестве левой части подстановки указывается не шаблон строки, применяемый в редакторе, а некоторое слово, представляющее собой единый блок без пробелов. Обычно в качестве разделителя между словами *Старое-слово* и *Новое_слово* указывается символ /. Однако в качестве разделителя можно задать любой символ. В следующем примере в качестве разделителя применяется символ %, поэтому символ / может выступать в качестве части слова:

```
s%/home/myfile%/home/yourfile%
```

Оболочка заменит выражение, состоящее из амперсанда (&), за которым указано *Старое_слово*, на *Новое_слово*. В следующем примере имя /home/myfile будет заменено на /temp/home/myfile.

```
s%/home/myfile%/temp&%
```

Пустое слово при подстановке заменяется оболочкой на последнюю подстановку или на последнюю строку, которая применялась в контекстном поиске строки !?*Строка*?. Последний ограничитель (/) можно опустить, если сразу за выражением следует символ новой строки. Для изменения элементов списка хронологии применяются следующие модификаторы:

Элемент	Описание
t	Удаляет все начальные компоненты пути, за исключением последнего
&	Повторяет предыдущую подстановку
g	Указывается, если необходимо применить изменения на глобальной основе, то есть для всех вхождений каждой строки
p	Показывает новую команду, но не запускает ее
q	Заключает в кавычки слова для подстановки, блокируя дальнейшие подстановки
x	Действует так же, как и модификатор **q,** но разбивает строку на слова, считая разделителями символы пробела, табуляции и новой строки.

Если модификатор **g** предшествует выбранному модификатору, то изменяется только первое указанное слово.

Если в ссылке на хронологию не задано событие (например, !$), то по умолчанию применяется предыдущая команда. Если предыдущая ссылка на хронологию задана в той же строке, то оболочка повторяет предыдущую ссылку. Например, следующая последовательность команд выбирает первый и последний аргумент команды, имя которой содержит строку ?foo?.

```
!?foo?^ !$
```

Строка ввода, которая начинается с символа ^, представляет собой специальный тип ссылки на хронологию. Это выражение эквивалентно команде !:s^, являясь короткой записью ссылки на текст предыдущей строки. Например, команда ^ lb^ lib исправляет опечатку в слове lib из предыдущей команды.

При необходимости подстановку хронологии можно заключить в фигурные скобки ({ }), чтобы отделить ее от последующих символов. Например, если вы хотите задать ссылку на следующую команду:

```
ls -ld ~paul
```

для того, чтобы выполнить команду:

```
ls -ld ~paula
```

то укажите следующее выражение:

```
!{l}a
```

В данном примере команда !{l} выполняет поиск команды, имя которой начинается с символы l, а затем добавляет к ней символ a.

Одинарные и двойные кавычки:

Для того чтобы прекратить дальнейшие подстановки, заключите строку в одинарные или двойные кавычки.

Если вы заключите строку в ' ', то дальнейшая интерпретация символов строки выполняться не будет; если вы заключите строку в " ", то будет выполняться дальнейшая обработка строки. В обоих случаях результатом обработки строки строки или части строки будет одно слово.

Перенаправление ввода и вывода в оболочке C:

Перед запуском команды оболочка C выполняет поиск символов перенаправления в командной строке. Такие символы означают, что необходимо перенаправить ввод или вывод.

Для перенаправления ввода и вывода команды применяются следующие операторы:

Элемент	Описание
< *Файл*	Открывает в качестве стандартного ввода указанный *Файл* (перед этим выполняется подстановка команд, переменных и имен файлов).
<<*Слово*	В качестве стандартного устройства ввода будет применяться клавиатура до тех пор, пока не будет введено указанное *Слово*. Для параметра *Слово* подстановка переменных, имен файлов и команд не выполняется. Каждая введенная строка сравнивается с параметром *Слово*, а затем в строке выполняются все необходимые подстановки. Если в переменной *Слово* нет символов кавычек (\, ", ' и `), оболочка выполняет подстановку переменных и команд в строках вводимого текста. При этом можно отменять специальное значение символов $, \ и `, указывая перед ними символ \. В командах, для которых выполняется подстановка, сохраняются все пробелы, символы табуляции и символы новой строки, за исключением последнего символа новой строки, который отбрасывается. Получившийся текст записывается во временный файл без имени, который передается команде в качестве стандартного ввода.
> *Файл* >!*Файл* >& *Файл* >&! *Файл*	Стандартный вывод будет связан с указанным *Файлом*. Если *Файл* не существует, то он будет создан. Если *Файл* существует, то все его содержимое удаляется. Если установлена переменная оболочки *noclobber*, то в параметре *Файл* должно быть задано имя нового файла, специального символьного файла или файла вывода сообщений об ошибках. Это позволяет предотвратить случайное повреждение файлов. Для подавления этой опции укажите конструкцию с символом ! . Подстановки в параметре *Файл* выполняются так же, как и для входных файлов <. Символ >& задает перенаправление стандартного вывода и вывода сообщений об ошибках в указанный *Файл*. В приведенном ниже примере показано, как перенаправить стандартный вывод в файл /dev/tty, а вывод сообщений об ошибках - в файл /dev/null. Для разделения стандартного вывода и вывода сообщений об ошибках необходимо указать круглые скобки. `% (find / -name vi -print > /dev/tty) >& /dev/null`
> >*Файл* > >!*Файл* > >& *Файл* > >&! *Файл*	Стандартный вывод связывается с указанным *Файлом*, как и в случае оператора >, однако вывод *добавляется* в конец *Файла*. Если установлена переменная оболочки *noclobber*, то в параметре *Файл* нужно задать имя существующего файла или указать конструкцию с символом ! . Если файл не существует, то действие этого оператора аналогично действию >.

Команде передаются переменные среды, которые были установлены при запуске оболочки, с учетом заданного перенаправления ввода-вывода. В отличие от многих других оболочек, у команды из сценария оболочки по умолчанию нет доступа к командной строке, из которой эта команда была вызвана. Вместо этого такие команды получают исходные данные из стандартного устройства ввода оболочки. Конструкция << позволяет предоставить внутренние данные, что, в свою очередь, позволяет применять командные файлы оболочек как элементы конвейера и как самостоятельные команды. Обратите внимание, что если команда выполняется в фоновом режиме, то ее стандартный ввод по умолчанию не перенаправляется в файл /dev/null. Вместо этого стандартный ввод по-прежнему связан с вводом с клавиатуры.

Для того чтобы перенаправить вывод сообщений об ошибках и стандартный вывод в конвейере, вместо символа | нужно указать символы |&.

Управление выполнением файлов в оболочке C:

В оболочке C предусмотрены операторы, позволяющие управлять ходом выполнения командных файлов (сценариев). В некоторых случаях эти операторы могут использоваться и в командной строке. В число операторов оболочки C входят стандартные операторы, присущие большинству языков программирования.

Для применения операторов **foreach**, **switch**, **while** и **if** (в форме **if-then-else**) необходимо задать все основные ключевые слова в строке ввода в виде одной команды.

Если оболочка выполняет чтение в цикле и полученный на одном из витков ввод не поддается синтаксическому разбору, то он помещается во внутренний буфер, и оболочка читает данные на следующем витке. Для выполнения такой последовательности действий за каждым нераспознанным блоком ввода указывается оператор безусловного перехода **goto**.

Защита операционной системы

Целью защиты системы является защита хранящейся в ней информации.

При этом защита информации должна обеспечивать следующее:

Элемент	Описание
Целостность	Ценность любой информации заключается в ее точности. Если в данные могут вноситься произвольные изменения, то их ценность существенно снижается.
Секретность	Ценность многих типов информации зависит от ее секретности.
Доступность	К информации должен обеспечиваться простой доступ.

Стратегию защиты системы рекомендуется разработать и применить до начала работы с системой. Стратегию защиты изменить очень сложно, поэтому лучше сразу принять правильное решение.

Идентификация и проверка прав доступа

Идентификация позволяет выяснить, является ли пользователь тем, за кого он себя выдает.

Идентификация выполняется при входе в систему. Вы указываете свой ИД пользователя и пароль, если он определен для учетной записи (в защищенной системе пользователям без пароля должно быть запрещено выполнение важных операций). Если пароль введен правильно, то вам назначается учетная запись: вы получаете права доступа этой учетной записи.

Поскольку пароль - это единственный способ защиты учетной записи, важно правильно выбрать пароль и хранить его в тайне. Часто попытки взломать систему начинаются с угадывания пароля. Операционная система достаточно надежно защищает пароли, храня их отдельно от остальной пользовательской информации. Зашифрованные пароли и другие конфиденциальные данные хранятся в файле `/etc/security/passwd`. Этот файл должен быть доступен только пользователю root. Если доступ к зашифрованным паролям запрещен, то взломщик не сможет расшифровать пароль с помощью программы, перебирающей все возможные сочетания символов.

Кроме того, существует возможность угадать пароль, повторяя попытки входа в систему. Если пароль слишком простой или редко изменяется, то такие попытки могут увенчаться успехом.

ИД входа в систему

Операционная система может распознать пользователей по их *идентификаторам (ИД) входа в систему*.

Идентификатор позволяет отслеживать все действия, выполняемые пользователем. После входа пользователя в систему, но до запуска начальной программы пользователя система устанавливает в качестве ИД пользователя процесса тот ИД пользователя, который указан в пользовательской базе данных. Кроме

того, этот ИД добавляется ко всем последующим процессам, запускаемым в сеансе. Это позволяет отслеживать все операции, выполняемые пользователем.

Пользователь может сбросить *действующий ИД пользователя*, *фактический ИД пользователя*, *действующий ИД группы*, *фактический ИД группы* и *дополнительный ИД группы*, но не может изменить ИД, под которым он вошел в систему.

Неконтролируемые терминалы

Если пользователь вошел в систему с некоторого терминала, а затем оставил терминал без присмотра, то уязвимость системы значительно повышается. Наиболее серьезные проблемы возникают в случае, когда таким пользователем является системный администратор (пользователь root). В общем случае, пользователи должны выходить из системы всякий раз, когда они оставляют терминал без присмотра.

Вы можете установить режим принудительного выхода простаивающего терминала из системы, задав параметры **TMOUT** и **TIMEOUT** в файле `/etc/profile`. Параметр **TMOUT** применяется в оболочке **ksh** (Korn), а параметр **TIMEOUT** - в оболочке **bsh** (Bourne).

В следующем примере, взятом из файла `.profile`, принудительный выход из системы терминала происходит через час простоя:

```
TO=3600
echo "Установка автоматического выхода из системы через $TO"
TIMEOUT=$TO
TMOUT=$TO
export TIMEOUT TMOUT
```

Примечание: Можно переопределить значения **TMOUT** и **TIMEOUT** в файле `/etc/profile`, указав их в файле `.profile` в своем домашнем каталоге.

Понятия, связанные с данным:

“Подстановка переменных в оболочке Bourne” на стр. 270
Оболочка Bourne позволяет выполнить подстановку переменных.

Ссылки, связанные с данной:

“Подстановка параметров в оболочке Korn и POSIX” на стр. 227
В оболочке Korn (POSIX) предусмотрены функции подстановки значений параметров.

Владельцы файлов и группы пользователей

Владельцем нового файла становится пользователь, который его создал.

Владелец файла задает права на чтение, запись (изменение) или выполнение файла. Изменить владельцев можно с помощью команды **chown**.

Любой ИД пользователя относится к определенной группе с уникальным ИД группы. Группы пользователей создаются системным администратором при настройке системы. Во время создания файла операционная система задает права доступа для пользователя, создавшего файл, группы владельца файла, а также группы `others`, включающей всех остальных пользователей. Команда **id** позволяет узнать ИД текущего пользователя (UID), ИД группы (GID) этого пользователя и имена всех групп, в которые он входит.

В списке файлов (например, в выводе команды **ls**) всегда указываются три класса пользователей в следующем порядке: пользователь, группа и другие. Если вам нужно узнать имя своей группы, введите команду **groups**, которая показывает список групп, в которые входит пользователь с данным ИД.

Изменение принадлежности файла или каталога:

С помощью команды **chown** можно изменить принадлежность своих файлов.

Если указана опция **-R**, то команда **chown** выполняет рекурсивный просмотр структуры каталогов, начиная с указанного каталога. При наличии символьной связи изменяется принадлежность только того файла или каталога, для которого создана эта связь; принадлежность самой символьной связи не изменяется.

Примечание: Только пользователь root может изменять принадлежность файлов, владельцем которых он не является. Если задана опция **-f**, сообщения об ошибках не выводятся.

Например, для того чтобы изменить владельца файла `program.c`, введите:

```
chown jim program.c
```

Права доступа владельца файла `program.c` теперь предоставлены пользователю `jim`. Как владелец файла, пользователь `jim` может вызвать команду **chmod** и изменить права доступа, предоставленные другим пользователям по отношению к файлу `program.c`.

Дополнительные сведения о синтаксисе приведены в описании команды **chown**.

Режимы доступа к файлам и каталогам:

У каждого файла есть владелец. Владельцем нового файла становится пользователь, создавший этот файл. *Права доступа* к файлу определяет его владелец. Права доступа указывают для других пользователей системы, могут ли они читать, изменять и исполнять файл. Права доступа к файлу могут быть изменены только владельцем файла или пользователем root.

Существует три класса пользователей: пользователь/владелец, пользователи группы и все остальные пользователи. Этим классам пользователей выделяются права доступа трех типов: на чтение, на запись и на выполнение. При создании файла для пользователя, создавшего файл, по умолчанию устанавливаются права на чтение, запись и выполнение. Для остальных двух классов пользователей устанавливаются права на чтение и выполнение. В приведенной ниже таблице описаны права доступа к файлу, предоставляемые по умолчанию для трех классов пользователей:

Элемент	**Описание**		
Классы	**Чтение**	**Запись**	**Выполнение**
Владелец	Да	Да	Да
Группа	Да	Нет	Да
Другие	Да	Нет	Да

Для каждого класса пользователей система устанавливает права доступа в виде набора разрешений. В операционной системе права доступа могут задаваться в символьном и числовом виде.

Понятия, связанные с данным:

“Каталоги” на стр. 475
Каталог - это особый тип файла, предназначенный для хранения информации, которая необходима для доступа к файлам или другим каталогам. Каталоги обычно занимают меньше памяти, чем файлы других типов.

“Типы файлов” на стр. 188
Система распознает файлы типа **обычные**, **файлы каталогов** или **специальные**. Однако в операционной системе также применяются многие производные типы файлов.

Символьное представление режима доступа:

Режимы доступа представляются в символьном виде.

Элемент	Описание
r	Означает разрешение на чтение, которое дает пользователю права на просмотр файла.
w	Означает разрешение на запись, которое дает пользователям права на изменение содержимого файла.
x	Означает разрешение на выполнение. Для исполняемых файлов (обычные файлы, содержащие программы), разрешение на выполнение предоставляет права на запуск программы. Для каталогов разрешение на выполнение предоставляет права на просмотр содержимого каталога.

Права доступа к файлам и каталогам задаются в виде строки, содержащей девять символов. Первые три символа представляют текущие права доступа **владельца**, следующие три символа - текущие права доступа **группы**, а последние три символа - текущие права доступа **других пользователей**. Дефис (**-**) в последовательности из девяти символов означает, что права доступа не предоставлены. Например, для файла с правами доступа `rwxr-xr-x` установлены разрешения на чтение и выполнение для всех трех классов пользователей, а разрешение на запись задано только для владельца файла. Это символьное представление прав доступа по умолчанию.

Команда **ls** с флагом **-l** (строчная буква L) показывает подробный список файлов каталога. Первые 10 символов в списке команды **ls-l** задают тип файла и разрешения, установленные для каждого из трех классов пользователей. Кроме того, в списке команды **ls -l** указываются владелец и группа, связанная с каждым файлом или каталогом.

Первый символ задает тип файла. Остальные девять символов - разрешения, установленные для каждого из трех классов пользователей. В качестве типа файла могут быть указаны следующие символы:

Элемент	Описание
-	Стандартные файлы
d	Каталог
b	Специальный блочный файл
c	Специальный символьный файл
p	Специальный файл конвейера
l	Символьная ссылка
s	Сокеты

Например, ниже приведен простой пример вывода команды **ls -l**:

```
-rwxrwxr-x   2   janet   acct   512 01 марта 13:33 january
```

Здесь первый дефис (-) означает, что это обычный файл. Следующие девять символов (`rwxrwxr-x`) представляют права доступа пользователя, группы и всех остальных пользователей, описанные ниже. `janet` - это имя владельца файла, а `acct` - имя группы Janet. `512` - это размер файла в байтах, `01 марта 13:33` - дата и время последнего изменения, а `january` - имя файла. Значение `2` задает количество ссылок на файл.

Числовое представление режима доступа:

В числовом виде разрешение на чтение представляется в виде цифры 4, разрешение на запись - в виде цифры 2, а разрешение на выполнение - в виде цифры 1. Суммарное значение от 1 до 7 задает права доступа для каждого класса (пользователя, группы и других).

В следующей таблице указано числовое представление прав доступа:

Суммарное значение	Чтение	Запись	Выполнение
0	-	-	-
1	-	-	1
2	-	2	-
3	-	2	1
4	4	-	-
5	4	-	1
6	4	2	-
7	4	2	1

При создании файла для него по умолчанию устанавливаются права доступа 755. Это означает, что для пользователя установлены права на чтение, запись и выполнение (4+2+1=7), для группы - права на чтение и выполнение (4+1=5), и для всех остальных пользователей - права на чтение и выполнение (4+1=5). Владелец файла может изменить права доступа к своему файлу с помощью команды **chmod**.

Просмотр информации о группе:

С помощью команды **lsgroup** можно просмотреть атрибуты всех групп системы или указанных групп. Если один или несколько атрибутов недоступны для чтения, то команда **lsgroup** показывает все доступные атрибуты.

Информация об атрибутах выводится в виде пар *Атрибут*=*Значение*, разделенных пробелами.

1. Для просмотра всех групп системы введите:

   ```
   lsgroup ALL
   ```

   Появится приблизительно следующий список, содержащий для каждой группы имя, ИД и описание всех пользователей группы:

   ```
   system  0       arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
   staff   1       john,ryan,flynn,daveb,jzitt,glover,maple,ken,gordon,mbrady
   bin     2       root,bin
   sys     3       root,su,bin,sys
   ```

2. Просмотреть отдельные атрибуты всех групп можно следующими способами:
   - Атрибуты можно просмотреть в виде пар *Атрибут=Значение*, разделенных пробелами. Этот способ применяется по умолчанию. Например, для просмотра списков идентификаторов и пользователей всех групп введите:

     ```
     lsgroup -a id users ALL | pg
     ```

     Будет показан список следующего вида:

     ```
     system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build
     staff id=1 users=john,ryan,flynn,daveb,jzitt,glover,maple,ken
     ```

   - Кроме того, информация может быть сгруппирована по разделам. Например, для просмотра отформатированного списка всех идентификаторов и групп пользователей введите:

     ```
     lsgroup -a -f id users ALL | pg
     ```

     Будет показан список следующего вида:

     ```
     system:
        id=0
        users=pubs,ctw,geo,root,chucka,noer,su,dea,backup,build

     staff:
        id=1
        users=john,ryan,flynn,daveb,jzitt,glover,maple,ken
     ```

```
bin:
   id=2
   users=root,bin

sys:
   id=3
   users=root,su,bin,sys
```

3. Для просмотра всех атрибутов отдельной группы применяются те же способы, что и для просмотра отдельных атрибутов всех групп:
   - Атрибуты можно просмотреть в виде пар *Атрибут=Значение*, разделенных пробелами. Этот способ применяется по умолчанию. Например, для просмотра всех атрибутов группы system введите:

     ```
     lsgroup system
     ```

     Будет показан список следующего вида:

     ```
     system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
     ```
   - Кроме того, информация может быть сгруппирована по разделам. Например, для просмотра отформатированного списка всех атрибутов группы `bin` введите:

     ```
     lsgroup -f system
     ```

     Будет показан список следующего вида:

     ```
     system:
        id=0   users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
     ```
4. Для просмотра отдельных атрибутов конкретной группы введите:

   ```
   lsgroup -a Атрибуты Группа
   ```

   Например, для просмотра ИД и списка пользователей группы `bin` введите:

   ```
   lsgroup -a id users bin
   ```

   Будет показан список следующего вида:

   ```
   bin id=2 users=root,bin
   ```

Дополнительные сведения о синтаксисе приведены в описании команды **lsgroup**.

Изменение прав доступа к файлам или каталогам:

С помощью команды **chmod** можно изменить принадлежность своих файлов.

1. Для того чтобы установить права доступа для файлов `chap1` и `chap2`, введите:

   ```
   chmod g+w chap1 chap2
   ```

   Эта команда предоставляет права на запись в файлы `chap1` и `chap2` пользователям группы.
2. Если вы хотите добавить или удалить несколько разрешений для каталога `mydir`, введите:

   ```
   chmod go-w+x mydir
   ```

   Эта команда аннулирует (**-**) права на создание и удаление файлов (**w**) в каталоге **mydir**, предоставленные пользователям группы (**g**) и другим пользователям (`o`), а также предоставляет этим пользователям (**+**) права на поиск в каталоге `mydir` и его применение в качестве элемента пути (**x**). Аналогичные действия выполняет следующая последовательность команд:

   ```
   chmod g-w mydir
   chmod o-w mydir
   chmod g+x mydir
   chmod o+x mydir
   ```
3. Если права на выполнение процедуры оболочки с именем **cmd** должны быть предоставлены только владельцу, введите команду:

   ```
   chmod u=rwx,go= cmd
   ```

В результате для владельца файла будут установлены разрешения на чтение, запись и выполнение (**u=rwx**). Для пользователей группы и всех остальных пользователей будет запрещен любой доступ к файлу `cmd` (**go=**).

4. Если вы хотите задать новые права доступа к файлу `text` с помощью команды **chmod**, указав их в числовом виде, то введите:

   ```
   chmod 644 text
   ```

   В результате для владельца будут установлены разрешения на чтение и запись, а для пользователей группы и всех остальных пользователей будут установлены права только на чтение.

Дополнительные сведения о синтаксисе приведены в описании команды **chmod**.

Списки управления доступом

Списки управления доступом - это защищенные информационные ресурсы, в которых указано, кому и к каким ресурсам разрешен доступ.

Операционная система обеспечивает раздельную защиту для разных режимов доступа. Владелец информации может предоставить другим пользователям права на запись или чтение для своего ресурса. Пользователь, которому предоставлены права доступа к ресурсу, может передавать эти права другим пользователям. Такая схема позволяет управлять распространением информации в системе; права доступа к объекту задаются его владельцем.

Права доступа пользовательского типа разрешают пользователю доступ только к его собственным объектам. Обычно пользователи получают также права доступа группы или права доступа по умолчанию для ресурса. Основная задача управления правами доступа состоит в определении групп пользователей, поскольку именно принадлежность к группе дает пользователю права доступа к чужим объектам.

Списки управления доступом для объектов файловых систем:

Как правило, объекты файловых систем связаны со списками управления доступом (ACL), состоящими из набора записей управления доступом (ACE). Каждая запись ACE задает субъект и права доступа к нему.

Для обслуживания списков управления доступом используйте команды **aclget**, **acledit**, **aclput** и **aclconvert**.

Обычно ACL хранятся и обслуживаются физической файловой системой (PFS). Операционная система AIX предоставляет инфраструктуру для поддержки нескольких типов ACL в физических файловых системах. Файловая система JFS2, входящая в комплект поставки AIX, поддерживает два типа ACL:

- AIXC
- NFS4

В более ранних файловых системах, а также в предыдущих версиях AIX поддерживались только ACL типа AIXC. Подробная информация об этих типах ACL приведена в руководстве *Защита*.

Тип списка управления доступом AIXC:

ACL типа AIXC (AIX Classic) - это стандартные списки управления доступом, применявшиеся в предыдущих версиях AIX. ACL этого типа состоят из разрядов обычного режима и расширенных прав доступа (ACE).

Расширенные права доступа позволяют разрешить или запретить доступ к ресурсу конкретным пользователям и группам, не меняя базовые права доступа.

Примечание: Размер ACL типа AIXC для файла не может превышать размер страницы (около 4096 байт).

С помощью команды **chmod** базовые права доступа и атрибуты объекта можно задать в числовом формате (в восьмеричной нотации). Функция **chmod**, которую вызывает эта команда, отключает расширенные права доступа. После применения команды **chmod** в числовом режиме к файлу с ACL расширенные права доступа

будут отключены. Команда **chmod** в символьном режиме не отключает расширенные права доступа для объектов с ACL типа AIXC. Информация о числовом и символьном режимах приведена в описании команды **chmod**. Информация о команде chmod приведена в разделе **chmod**.

Базовые права доступа

Базовые права доступа в системе ACL AIXC представляют собой стандартные режимы доступа для владельца файла, группы и остальных пользователей. Режимы доступа: это чтение (r), запись (w) и выполнение/поиск (x).

Примечание: Базовые права доступа ACL AIXC такие же, как биты режима доступа к файлу, хранящиеся в заголовках соответствующего i-узла. Иными словами, в битах режима доступа содержится та же информация, что возвращается функцией **stat** для объекта файловой системы.

В списке управления доступом базовые права хранятся в приведенном ниже формате, где вместо параметра **Режим** стоит строка rwx (прочерк (-) означает, что права доступа не заданы):

```
базовые права доступа:
   owner(имя пользователя): Режим
   group(имя группы): Режим
   others: Режим
```

Атрибуты

В список управления доступом можно добавить три атрибута:

setuid (SUID)
Флаг setuid. Этот атрибут устанавливает в качестве действующего и сохраненного ИД пользователя процесса ИД владельца файла.

setgid (SGID)
Флаг setgid. Этот атрибут устанавливает в качестве действующего и сохраненного ИД группы процесса ИД группы файла.

savetext (SVTX)
Сохраняет текст в формате текстового файла.

Эти атрибуты добавляются в следующем формате:

```
атрибуты: SUID, SGID, SVTX
```

Расширенные права доступа

Расширенные права доступа ACL AIXC позволяют владельцу файла более точно настроить режим доступа к файлу. Расширенные права доступа изменяют базовые права (для владельца, группы и остальных) разрешая, запрещая или указывая режимы доступа для отдельных пользователей, групп или комбинаций пользователей и групп. Для изменения прав доступа применяются ключевые слова.

Ключевые слова permit, deny и specify означают следующее:

permit Разрешает пользователю или группе указанный режим доступа к файлу

deny Запрещает пользователю или группе указанный режим доступа к файлу

specify Задает для пользователя или группы права доступа к файлу

Если пользователю запрещен определенный режим доступа ключевым словом deny или specify, его уже нельзя разрешить каким-либо другим образом.

Для того чтобы разрешить применение расширенных прав доступа, нужно указать в ACL ключевое слово enabled. По умолчанию вместо него стоит ключевое слово disabled (отключено).

В ACL AIXC расширенные права доступа записываются в следующем формате:

```
расширенные права доступа:
  разрешено | запрещено
    permit   Режим  Пользователи...:
    deny     Режим  Пользователи...:
    specify  Режим  Пользователи...:
```

Каждое из ключевых слов permit, deny и specify должно находиться на отдельной строке. Параметр **Режим** заменяется на `rwx` (прочерк (-) означает, что права доступа не заданы). Параметр **сведения_о_пользователе** выражается как `u:имя_пользователя` или `g:имя_группы`, либо с помощью `u:имя_пользователя` и `g:имя_группы`, указанными через запятую.

Примечание: Если в строке указано несколько имен пользователей, эта строка не может применяться для управления доступом, поскольку каждому процессу соответствует только один ИД пользователя.

Тип списка управления доступом NFS4:

Файловая система JFS2 операционной системы AIX поддерживает ACL типа NFS4. Эта реализация ACL соответствует спецификациям RFC для протокола NFS4 версии 4.

Она позволяет задавать гораздо более детальные права доступа и оснащена рядом дополнительных функций, например возможностью наследования. ACL NFS4 представляет собой массив ACE. В каждой записи ACE заданы права доступа для определенного субъекта. В RFC основные компоненты ACE NFS4 определены следующим образом:

```
struct nfsace4 {

                acetype4          type;
                aceflag4           flag;
                acemask4        access_mask;
                utf8str_mixed   who;
};
```

где

type Маска, задающая тип ACE. В этом поле задается такая информация, как то, разрешает или запрещает данный ACE доступ к ресурсу.

flag Маска, задающая параметры наследования ACE. Указывает, к чему применяется данная ACE: к объекту файловой системы, его дочерним объектам или и к тому, и к другому.

access_mask
Маска, задающая права доступа. В число возможных прав доступа входят чтение, запись, выполнение, создание, удаление, создание дочерних объектов, удаление дочерних объектов и т.д.

who Эта строка, оканчивающаяся символом NULL, задает пользователя, к которому относится данная ACE. В RFC длина этой строки не ограничена, что позволяет создавать в NFS версии 4 домены управления доступом. В обычном режиме (он задействован большую часть времени) AIX игнорирует эту строку, и каждая запись ACE привязывается к субъекту AIX (например **uid** или **gid**). Предполагается, что данные строки интерпретируются и сопоставляются пользователям и группам файловой системой NFS 4. AIX распознает только некоторые строки **who**, описанные в RFC.

Для обслуживания списков управления доступом NFS4 в AIX используйте команды **aclget**, **acledit**, **aclput** и **aclconvert**.

Примечание: При использовании любого типа команды **chmod**, ACL файла будет удален.

Пример списка управления доступом для AIXC:

Ниже приведен пример списка управления доступом (ACL) AIXC.

Ниже приведен пример ACL AIXC:

```
атрибуты: SUID
базовые права доступа:
      owner(frank):  rw-
      group(system): r-x
      others: ---
расширенные права доступа:
      разрешено
        permit  rw-  u:dhs
        deny    r--  u:chas, g:system
        specify r--  u:john, g:gateway, g:mail
        permit  rw-  g:account, g:finance
```

Расшифровка строк ACL приведена ниже:

- Первая строка устанавливает значение бита setuid.
- Следующая строка, начинающая раздел базовых прав доступа, не обязательна.
- В следующих трех строках задаются базовые права доступа. Имена пользователя и группы в скобках указаны только для удобства. Изменение этих имен не изменит владельца или группу файла. Изменять эти атрибуты файла могут лишь команды **chown** и **chgrp**. Дополнительная информация о этих командах приведена в разделах **chown** и **chgrp**.
- Следующая строка, начинающая раздел расширенных прав доступа, также не обязательна.
- Следующая строка означает, что расширенные права доступа включены.
- В последних четырех строках приведены расширенные записи.
- Первая расширенная запись предоставляет пользователю `dhs` права на чтение (`r`) и на запись (`w`) к файлу.
- Вторая расширенная запись запрещает чтение (`r`) пользователю `chas`, если он входит в группу system.
- Третья расширенная запись указывает, что пользователю `john` разрешено чтение (`r`), пока он состоит в группах gateway и mail. Если пользователь `john` не является членом ни одной из этих групп, то данная расширенная запись к нему не применяется.
- Последняя запись расширенных прав доступа разрешает чтение (`r`) и запись (`w`) всем пользователям, которые входят в **обе** группы `account` и `finance`.

Примечание: Если к процессу применяются несколько расширенных записей, запреты имеют больший приоритет, чем разрешения.

Дополнительные сведения и информация о синтаксисе приведены в описании команды **acledit** в книге *Справочник по командам, том 1*.

Проверка прав доступа к спискам управления доступом:

За управление доступом к информационному ресурсу отвечает владелец этого ресурса. Ресурсы защищены битами прав доступа, входящими в режим объекта.

Для AIXC ACL биты прав доступа задают права, предоставленные владельцу объекта, группе объекта и `остальным` пользователям. Тип AIXC ACL поддерживает три различных режима доступа (чтение, запись и выполнение), которые могут предоставляться независимо друг от друга.

Когда пользователь входит в систему (командой **login** или **su**, ИД пользователя и группы, присвоенные его учетному файлу, связываются с пользовательскими процессами. Эти ИД и определяют права доступа процесса.

Для файлов, каталогов, именованных конвейеров и устройств (особых файлов) с AIX ACL права доступа определяются следующим образом:

- Для каждой записи управления доступом (ACE) из списка управления доступом (ACL) список идентификаторов сравнивается с идентификаторами процесса. В случае их соответствия к процессу применяется соответствующее разрешение или запрещение, указанное в записи. Логические значения запрещений и разрешений из различных записей ACL умножаются. Если запрашивающему процессу не соответствует ни одна из записей ACL, он получает права доступа из записи по умолчанию.
- Если запрошенный режим доступа разрешен (включен в число разрешений) и не запрещен (не включен в число запретов), доступ предоставляется. Иначе запрос на доступ отклоняется.

Список идентификаторов в ACL типа AIXC соответствует процессу, если все идентификаторы из этого списка соответствуют одному из действующих идентификаторов запрашивающего процесса. Идентификатор типа USER (пользовательского типа) считается подходящим, если он равен действующему ИД пользователя процесса. Идентификатор типа GROUP (группы) считается подходящим, если он равен действующему ИД группы или одной дополнительных групп. Например, ACE со следующим списком идентификаторов:

```
USER:fred, GROUP:philosophers, GROUP:software_programmer
```

будет соответствовать процессу с действующим ИД пользователя `fred` и следующим набором групп:

```
philosophers, philanthropists, software_programmer, doc_design
```

но не будет соответствовать процессу с действующим ИД пользователя `fred` и следующим набором групп:

```
philosophers, iconoclasts, hardware_developer, graphic_design
```

Обратите внимание, что ACE со следующим списком идентификаторов будет соответствовать обоим процессам:

```
USER:fred, GROUP:philosophers
```

Другими словами, список идентификаторов в ACE работает как набор условий, которые должны быть выполнены для предоставления (запрещения) определенного режима доступа.

Разделение режимов доступа позволяет более эффективно управлять доступом к информационным ресурсам и предназначено для отдельной защиты секретности и целостности информации. Этот механизм работает только в той мере, в которой владельцы объектов настраивают и поддерживают его. В связи с этим все пользователи должны знать способы предоставления, аннулирования и настройки прав доступа.

Для тех объектов файловой системы, с которыми связаны ACL типа NFS4, проверка прав доступа выполняется в форме анализа ACE, составляющих ACL, согласно спецификациям RFC по протоколу NFS версии 4. Субъекты сопоставляются по ИД пользователя, ИД группы и специальным строкам `who` в ACE. При наличии совпадения проверяются соответствие запрошенных прав доступа правам доступа, указанным в ACE. Для формирования прав доступа последовательно просматриваются права, указанные в ACE, до тех пор пока не закончится ACL, не будут найдены все необходимые права доступа, либо не будет обнаружен запрет на по крайней мере одно из запрошенных прав доступа. Ниже описана процедура проверки прав доступа для объекта файловой системы, к которому привязан ACL NFS4:

1. Для каждой записи управления доступом (ACE) из списка управления доступом (ACL) список идентификаторов сравнивается с идентификаторами процесса. Проверяются ИД пользователей и групп, указанные в ACE. Кроме того, если субъект помечен как **особый** с такими строками как `OWNER@`, то совпадением будет считаться принадлежность файла вызывающему процессу. В случае их соответствия к процессу извлекаются права доступа, указанные в записи. В противном случае выполняется переход к следующей ACE.
2. Запрошенные права доступа сравниваются с правами доступа, извлеченными из ACE. Если какое-либо из запрошенных прав доступа явно запрещается данной ACE, то процедура проверки прав доступа заканчивается, и в доступе будет отказано.

3. Если ACE предоставляет некоторые из запрошенных прав доступа, то предоставляемые права будут исключены из списка необходимых прав доступа, и будет выполнен переход к следующей ACE.
4. Если таким образом будут получены все запрошенные права доступа, то доступ разрешается.
5. Если к концу списка ACL будут получены не все из запрошенных прав доступа, доступ запрещается.

Учтите, что помимо проверки по ACL, в отдельных файловых системах могут быть установлены собственные ограничения на доступ к объектам. Например, у владельца объекта может быть право на изменение ACL независимо от текущего содержания ACL. Процесс с ИД пользователя 0 называется процессом пользователя root. Такие процессы обычно обладают всеми правами доступа. Тем не менее, если пользователь root запрашивает выполнение программы, соответствующий режим доступа предоставляется ему только в случае, если права на выполнение предоставлены по крайней мере одному пользователю.

Все проверки прав доступа для данных объектов выполняются на уровне системных вызовов при первом обращении к объекту. Поскольку для объектов SVIPC (System V Interprocess Communication) информация о состоянии не хранится, проверка выполняется при каждом обращении. Однако возможна ситуация, когда проверки выполняются при открытии объекта файловой системы, но не при выполнении операций чтения и записи. Для доступа к объектам по имени файловой системы необходимо знать фактическое имя объекта. Имена указываются либо относительно рабочего каталога процесса, либо полностью, начиная с корневого каталога. Все имена могут быть получены одним из этих способов.

Просмотр информации об управлении доступом к файлу (команда aclget):

Команда **aclget** позволяет просмотреть информацию об управлении доступом к файлу. При этом будут показаны атрибуты, базовые и расширенные права доступа.

Например, для просмотра информации об управлении доступом к файлу `status` введите:

```
aclget status
```

Будет показана информация об управлении доступом: атрибуты, базовые и расширенные права доступа.

Сведения о синтаксисе приведены в описании команды **aclget** в книге *Справочник по командам, том 1*.

Понятия, связанные с данным:

“Пример и описание списка управления доступом” на стр. 307
Ниже приведен пример и описание списков управления доступом (ACL).

Настройка информации об управлении доступом к файлу (команда aclput):

Для того чтобы задать информацию об управлении доступом для файла, вызовите команду **aclput**.

Примечание: Список прав доступа для файла не может занимать более одной страницы памяти (приблизительно 4096 байт).

Примеры:

Например, для того чтобы задать для файла status параметры управления доступом, указанные в файле `acldefs`, введите:

```
aclput -i acldefs status
```

Чтобы задать для файла status такие же параметры доступа, как у файла `plans`, введите:

```
aclget plans | aclput status
```

Дополнительные сведения и информация о синтаксисе приведены в описании команды **aclput** в книге *Справочник по командам, том 1*.

Пример и описание списка управления доступом:

Ниже приведен пример и описание списков управления доступом (ACL).

Ниже приведен пример ACL:

```
атрибуты: SUID
базовые права доступа:
      owner(frank):  rw-
      group(system): r-x
      others: ---
расширенные права доступа:
      разрешено
        permit  rw-  u:dhs
        deny    r--  u:chas, g:system
        specify r--  u:john, g:gateway, g:mail
        permit  rw-  g:account, g:finance
```

Расшифровка строк ACL приведена ниже:

- Первая строка устанавливает значение бита **setuid**.
- Следующая строка, начинающая раздел базовых прав доступа, не обязательна.
- В следующих трех строках задаются базовые права доступа. Имена пользователя и группы в скобках указаны только для удобства. Изменение этих имен не изменит владельца или группу файла. Изменять эти атрибуты файла могут лишь команды **chown** и **chgrp**. Дополнительная информация о этих командах приведена в разделах **chown** и **chgrp**.
- Следующая строка, начинающая раздел расширенных прав доступа, также не обязательна.
- Следующая строка означает, что расширенные права доступа включены.
- В последних четырех строках приведены расширенные записи. Первая расширенная запись предоставляет пользователю `dhs` права на чтение (`r`) и на запись (`w`) к файлу.
- Вторая расширенная запись запрещает чтение (`r`) пользователю `chas`, если он входит в группу `system`.
- Третья расширенная запись указывает, что пользователю `john` разрешено чтение (`r`), пока он состоит в группах `gateway` и `mail`. Если пользователь `john` не является членом ни одной из этих групп, то данная расширенная запись к нему не применяется.
- Последняя запись расширенных прав доступа разрешает чтение (`r`) и запись (`w`) всем пользователям, которые входят в **обе** группы `account` и `finance`.

  Примечание: Если к процессу применяются несколько расширенных записей, запреты имеют больший приоритет, чем разрешения.

  Сведения о синтаксисе приведены в описании команды **acledit** книги *Справочник по командам, том 1*.

Понятия, связанные с данным:

“Просмотр информации об управлении доступом к файлу (команда aclget)” на стр. 306
Команда **aclget** позволяет просмотреть информацию об управлении доступом к файлу. При этом будут показаны атрибуты, базовые и расширенные права доступа.

Задачи, связанные с данной:

“Изменении информации об управлении доступом к файлу (команда acledit)”
Команда **acledit** предназначена для изменения информации об управлении доступом к файлу. Эта команда показывает текущую информацию об управлении доступом и позволяет владельцу файла изменять ее.

Изменении информации об управлении доступом к файлу (команда acledit):

Команда **acledit** предназначена для изменения информации об управлении доступом к файлу. Эта команда показывает текущую информацию об управлении доступом и позволяет владельцу файла изменять ее.

Перед сохранением любых изменений команда запрашивает подтверждение. Информация о команде **acledit** приведена в разделе **acledit**.

Примечание: В переменной среды *EDITOR* должен быть указан полный путь, иначе при выполнении команды **acledit** возникнет ошибка.

Будет показана информация об управлении доступом, зависящая от типа ACL: атрибуты, базовые и расширенные права доступа.

Например, для того чтобы изменить информацию об управлении доступом для файла `plans`, введите:

```
acledit plans
```

Сведения о синтаксисе приведены в описании команды **acledit** в книге *Справочник по командам, том 1*.

Понятия, связанные с данным:

“Пример и описание списка управления доступом” на стр. 307
Ниже приведен пример и описание списков управления доступом (ACL).

Блокирование терминала (команда lock или xlock)

Команда **lock** позволяет блокировать терминал. Команда **lock** запрашивает у пользователя пароль, считывает его и запрашивает подтверждение пароля.

Затем команда блокирует терминал до тех пор, пока пароль не будет введен во второй раз. Тайм-аут по умолчанию равен 15 минутам; его можно изменить с помощью флага **-***Число*.

Примечание: В интерфейсе AIXwindows для этой цели служит команда **xlock**.

Например, для блокирования терминала и его защиты с помощью пароля введите:

```
lock
```

Вам будет дважды предложено указать пароль, чтобы система могла его проверить. Если пароль не будет указан в течение 15 минут, терминал будет заблокирован.

Для блокирования терминала и его защиты с помощью пароля с тайм-аутом в `10` минут введите:

```
lock -10
```

Дополнительные сведения о синтаксисе приведены в описании команд **lock** и **xlock** в книге в книге *Справочник по командам*.

Обзор команд для защиты файлов и системы

Ниже описаны команды для работы с файловой системой и защитой.

Элемент	Описание
acledit	Изменяет информацию об управлении доступом к файлу
aclget	Показывает информацию об управлении доступом к файлу
aclput	Задает информацию об управлении доступом к файлу
chmod	Изменяет режимы доступа
chown	Изменяет пользователя, связанного с файлом
lock	Резервирует терминал
lsgroup	Показывает атрибуты групп
xlock	Блокирует локальный X-дисплей до ввода пароля

Пользовательская среда

С каждым именем пользователя связана определенная системная среда.

Системная среда представляет собой область, в которой хранится информация, общая для всех процессов данного сеанса. Для просмотра информации о системе предусмотрен ряд команд.

Файлы и процедуры настройки пользовательской среды

Эти файлы и процедуры помогают пользователю при настройке системной среды.

Системные файлы запуска

Элемент	Описание
/etc/profile	Системный файл, который содержит команды, выполняемые при входе в систему.
/etc/environment	Системный файл, который содержит переменные, задающие базовую среду для всех процессов.
$HOME/.profile	Файл в вашем домашнем каталоге, который содержит команды, переопределяющие значения в **/etc/profile** при входе в систему. За дополнительной информацией обратитесь к описанию файла .profile.
$HOME/.env	Файл в вашем домашнем каталоге, переопределяющий системный файл **/etc/environment** и содержащий переменные, задающие базовую среду для всех процессов. За дополнительной информацией обратитесь к описанию файла .env.

Файлы запуска AIXwindows

Элемент	Описание
$HOME/.xinitrc	Файл в вашем домашнем каталоге, управляющий окнами и приложениями, которые запускаются вместе с AIXwindows. За дополнительной информацией обратитесь к описанию файла .xinitrc.
$HOME/.Xdefaults	Файл в вашем домашнем каталоге, управляющий параметрами графики и работой с устройствами AIXwindows. За дополнительной информацией обратитесь к разделу “Файл .Xdefaults” на стр. 318.
$HOME/.mwmrc	Файл в вашем домашнем каталоге, задающий назначения клавиш, кнопок мыши и определений меню для Администратора окон. За дополнительной информацией обратитесь к разделу “Файл .mwmrc” на стр. 319.

Процедуры настройки

Элемент	Описание
PS1	Обычное системное приглашение
PS2	Системное приглашение для ввода дополнительной информации
PS3	Системное приглашение пользователя root
chfont	Изменяет шрифт, которым выводится текст на экране, при перезагрузке системы
stty	Задает, сбрасывает и сообщает параметры рабочей станции

Список системных устройств (команда lscfg)

Команда **lscfg** показывает список всех устройств системы, в котором указано имя, расположение и описание каждого устройства. Список упорядочен по расположению устройств.

Например, для просмотра списка устройств системы введите в командной строке:

```
lscfg
```

Появится примерно следующая информация:

```
СПИСОК УСТАНОВЛЕННЫХ РЕСУРСОВ

В вашей системе установлены следующие ресурсы.

+/- = Добавлено/Удалено из списка диагностики.
*   = Не поддерживается диагностикой.


  Архитектура модели: chrp
  Реализация модели: Многопроцессорная система с шиной PCI

+  sysplanar0   00-00           Системный планар
+  fpa0         00-00           Математический процессор
+  mem0         00-0A           Модуль памяти
+  mem1         00-0B           Модуль памяти
+  ioplanar0    00-00           Планар ввода-вывода
```

```
+  rs2320       00-01           Карта RS232
+  tty0         00-01-0-01      Порт карты RS232
-  tty1         00-01-0-02      Порт карты RS232
   ..
   ..
   ..
```

Список упорядочен не только по расположению устройств. Он также представляет иерархию родительских и дочерних устройств. Если у родительского устройства есть несколько дочерних устройств, они будут отсортированы по расположению. Если расположения всех дочерних устройств совпадают, они выводятся в том порядке, в котором они были получены программой. Для просмотра информации об отдельном устройстве введите флаг **-l**. Например, для просмотра информации об устройстве **sysplanar0** введите в командной строке:

```
lscfg -l sysplanar0
```

Появится примерно следующая информация:

```
УСТРОЙСТВО     РАСПОЛОЖЕНИЕ      ОПИСАНИЕ

sysplanar0     00-00        Системный планар
```

Кроме того, команда **lscfg** применяется для просмотра таких основных сведений о продукте (VPD), как код продукта, серийный номер и уровень модификации. Для некоторых устройств сбор основных сведений о продукте и добавление их к конфигурации системы выполняются автоматически. Для остальных устройств информация VPD добавляется вручную. Введенные вручную данные помечены символами ME.

Например, для просмотра данных реестра для всех устройств системы введите в командной строке:

```
lscfg -v
```

Появится примерно следующая информация:

```
СПИСОК УСТАНОВЛЕННЫХ РЕСУРСОВ С VPD
В вашей системе установлены следующие ресурсы.
  Архитектура модели: chrp
  Реализация модели: Многопроцессорная система с шиной PCI
sysplanar0     00-00        Системный планар
   Код.........342522
   Уровень EC............254921
   Серийный номер.......353535
fpa0   00-00   Математический процессор
mem0   00-0A   Модуль памяти
   Уровень EC............990221
.
.
.
```

Сведения о синтаксисе приведены в описании команды **lscfg** в книге *Справочник по командам, том 3*.

Просмотр имен консолей

Команда **lscons** записывает в стандартный вывод (обычно - на экран) имя текущей консоли.

Например, введите в командной строке:

```
lscons
```

Появится примерно следующая информация:

```
/dev/lft0
```

Дополнительные сведения о синтаксисе приведены в описании команды **lscons**.

Просмотр имени терминала (команда tty)

Для просмотра имени терминала введите команду **tty**.

Например, введите в командной строке:

```
tty
```

Появится приблизительно следующая информация:

```
/dev/tty06
```

В этом примере `tty06` - имя терминала, а **/dev/tty06** - файл устройства с интерфейсом к этому терминалу.

Сведения о синтаксисе приведены в описании команды **tty** в книге *Справочник по командам, том 5*.

Просмотр доступных устройств (команда lsdisp)

Команда **lsdisp** показывает список доступных дисплеев системы, содержащий идентификационное имя, номер разъема, обычное имя и описание дисплея.

Например, для просмотра всех доступных дисплеев введите:

```
lsdisp
```

Ниже приведен пример вывода. Элементы списка упорядочены в порядке возрастания номеров разъемов.

```
Имя   Разъем    Имя        Описание
ppr0   00-01   POWER_G4    Промежуточный графический адаптер
gda0   00-03   colorgda    Цветной графический адаптер
ppr1   00-04   POWER_Gt3   Промежуточный базовый графический адаптер
```

Сведения о синтаксисе приведены в описании команды **lsdisp** в книге *Справочник по командам, том 3*.

Просмотр доступных шрифтов (команд lsfont)

Для просмотра списка шрифтов, доступных для вашего дисплея, введите команду **lsfont**.

Например, для просмотра списка всех доступных шрифтов дисплея введите:

```
lsfont
```

Ниже приведен пример вывода команды, содержащего идентификатор, имя файла, размер глифа и кодировку шрифта:

```
ИД    ИМЯ             РАЗМ   КОДИРОВКА
ШРИФТА ФАЙЛА          ГЛИФА  ШРИФТА
====  ==============  =====  =========
0     Erg22.iso1.snf  12x30  ISO8859-1
1     Erg11.iso1.snf   8x15  ISO8859-1
```

Сведения о синтаксисе приведены в описании команды **lsfont** в книге *Справочник по командам, том 3*.

Просмотр текущей раскладки клавиатуры (команда lskbd)

Для просмотра полного имени файла, содержащего текущую раскладку клавиатуры, загруженную в систему, введите команду **lskbd**.

Например, для просмотра текущей раскладки клавиатуры введите:

```
lskbd
```

Ниже приведен пример вывода команды **lskbd**:

```
Текущая раскладка клавиатуры = /usr/lib/nls/loc/C.lftkeymap
```

Просмотр доступных в системе программных продуктов (lslpp команда)

Для просмотра информации о программных продуктах, доступных в системе, введите команду **lslpp**.

Например, для просмотра списка всех имеющихся в системе программных продуктов, введите в командной строке:

```
lslpp -l -a
```

Ниже приведен пример вывода:

```
Набор файлов  Уровень Состояние  Описание
-------------------  ------- --------  -----------------
Путь: /usr/lib/objrepos
   X11_3d.gl.dev.obj           APPLIED   AIXwindows/3D GL
                                         Development Utilities
Шрифты
  X11fnt.oldX.fnt             APPLIED   AIXwindows Miscellaneous
                                         X Fonts
X11mEn_US.msg                 APPLIED   AIXwindows NL Message
                                         файлы
.
.
.
```

Если список не умещается на экране, то будет показана только нижняя часть списка. Для постраничного просмотра списка объедините команду **lslpp** в конвейер вместе с командой **pg**. Введите в командной строке:

```
lslpp -l -a | pg
```

Сведения о синтаксисе приведены в описании команды **lslpp** в книге *Справочник по командам, том 3*.

Просмотр функций клавиш терминала (команда stty)

Команда **stty** показывает параметры терминала. В частности, с помощью этой команды можно просмотреть управляющие клавиши терминала.

Например, введите в командной строке:

```
stty -a
```

Появится приблизительно следующая информация:

```
.
.
.
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D;
eol = ^@ start = ^Q; stop = ^S; susp = ^Z; dsusp = ^Y;
reprint = ^R discard = ^O; werase = ^W; lnext = ^V
.
.
.
```

В данном примере строки вида `intr = ^C; quit = ^\; erase = ^H;` содержат информацию об управляющих клавишах. Клавиша `^H` - это клавиша Backspace, а erase - это действие, назначенное этой клавише.

Если список не умещается на экране, то будет показана только нижняя часть списка. Для постраничного просмотра списка объедините команду **stty** в конвейер вместе с командой **pg**. Введите в командной строке:

```
stty -a | pg
```

Сведения о синтаксисе приведены в описании команды **stty** в книге *Справочник по командам, том 5*.

Понятия, связанные с данным:

“Отмена интерактивного процесса” на стр. 142
Если вы решили прервать работу запущенного вами процесса, нажмите клавишу INTERRUPT. Обычно это

клавиши Ctrl-C или Ctrl-Backspace.

Просмотр переменных среды (команда env)

Для просмотра текущих переменных среды вызовите команду **env**. Переменные среды, доступные для всех процессов, называются *глобальными переменными*.

Например, для просмотра списка переменных среды и их значений введите:

```
env
```

Ниже приведен пример вывода:

```
TMPDIR=/usr/tmp
myid=denise
LANG=En_US
UNAME=barnard
PAGER=/bin/pg
VISUAL=vi
PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/u/bin1
MAILPATH=/usr/mail/denise?denise has mail !!!
MAILRECORD=/u/denise/.Outmail
EXINIT=set beautify noflash nomesg report=1 showmode showmatch
EDITOR=vi
PSCH=>
HISTFILE=/u/denise/.history
LOGNAME=denise
MAIL=/usr/mail/denise
PS1=denise@barnard:${PWD}>
PS3=#
PS2=>
epath=/usr/bin
USER=denise
SHELL=/bin/ksh
HISTSIZE=500
HOME=/u/denise
FCEDIT=vi
TERM=lft
MAILMSG=**YOU HAVE NEW MAIL. ЕЕ МОЖНО ПРОСМОТРЕТЬ С ПОМОЩЬЮ КОМАНДЫ mail
ENV=/u/denise/.env
```

Если список не умещается на экране, то будет показана только его нижняя часть. Для постраничного просмотра списка объедините команду **env** в конвейер вместе с командой **pg**. Введите в командной строке:

```
env | pg
```

Сведения о синтаксисе приведены в описании команды **env** в книге *Справочник по командам, том 2*.

Просмотр значения переменных среды (команда printenv)

Для просмотра значений переменных среды вызовите команду **printenv**.

Указав параметр **Имя**, можно просмотреть значение заданной переменной среды. Если параметр **Имя** не указан, команда **printenv** показывает все текущие переменные среды. Каждая пара **Имя**=*Значение* выводится на отдельной строке.

Например, для того чтобы узнать текущее значение переменной среды *MAILMSG*, введите:

```
printenv MAILMSG
```

Команда покажет значение переменной среды *MAILMSG*. Например:

```
Вам пришла почта
```

Сведения о синтаксисе приведены в описании команды **printenv** в книге *Справочник по командам, том 4*.

Двунаправленные языки (команда aixterm)

Команда **aixterm** поддерживает иврит и арабский язык.

Двунаправленные языки характеризуются возможностью чтения и написания в двух направлениях: как слева направо, так и справа налево. Для работы с арабским языком или ивритом в приложениях необходимо открыть окно с указанием локали соответствующего языка.

Сведения о синтаксисе приведены в описании команды **aixterm** книги *Справочник по командам, том 1*.

Обзор команд для работы с пользовательской средой и системной информацией

Ниже описаны команды для работы с пользовательской средой и системной информацией.

Элемент	Описание
aixterm	Позволяет работать с двунаправленными языками
env	Показывает текущую среду или задает среду для выполнения команды
lscfg	Показывает диагностическую информацию об устройстве
lscons	Показывает имя текущей консоли
lsdisp	Показывает список доступных дисплеев
lsfont	Показывает список шрифтов, доступных для дисплея
lskbd	Показывает список раскладок клавиатур, загруженных в систему
lslpp	Выдает список программных продуктов
printenv	Показывает значения переменных среды
stty	Показывает параметры системы
tty	Показывает полный путь к терминалу

Настройка пользовательской среды

В операционной системе предусмотрены различные команды и файлы инициализации, позволяющие настраивать рабочие характеристики и параметры пользовательской среды.

Кроме того, вы можете настраивать некоторые ресурсы по умолчанию, применяемые приложениями в вашей системе. Значения по умолчанию инициализируются при запуске программы. После изменения этих значений необходимо перезапустить программу, для того чтобы новые значения вступили в силу.

За информацией о настройке функций и внешнего вида Общей среды рабочего стола обратитесь к руководству *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*.

Системные файлы запуска:

Когда вы входите в систему, оболочка считывает ваши файлы инициализации и на основе этой информации настраивает пользовательскую среду. Параметры среды определяются значениями, которые вы указали в переменных среды. Эти значения будут действительны до выхода из системы.

При входе в систему оболочка применяет файлы profile двух типов. Она выполняет предварительную обработку команд, указанных в файлах, и затем запускает команды для настройки среды. Функции этих файлов схожи, за исключением того, что файл `/etc/profile` управляет переменными профайла для всех пользователей системы, а файл `.profile` позволяет вам настроить свою собственную среду.

Оболочка сначала выполняет команды, содержащиеся в файле `/etc/environment`, а затем команды из файла `/etc/profile`. После запуска файлов система проверяет, есть ли файл `.profile` в вашем домашнем каталоге. Если файл `.profile` существует, то система запускает этот файл. Кроме того, в файле `.profile` указано, существует ли файл среды. Если да (обычно он называется `.env`), то система запускает его и задает переменные среды.

Файлы `/etc/environment`, `/etc/profile` и `.profile` запускаются один раз во время входа в систему. Файл `.env` - каждый раз, когда вы открываете новую оболочку или окно.

Файл /etc/environment:

При входе в систему первым считывается файл /etc/environment. Файл /etc/environment содержит переменные, задающие базовую среду для всех процессов.

При запуске нового процесса процедура **exec** создает массив из строк вида *Имя=Значение*. Этот массив строк называется *средой*. Каждое имя, указанное в строке, называется *переменной среды* или *переменной оболочки*. Процедура **exec** позволяет сразу задать всю среду.

При входе в систему вначале задаются переменные среды из файла /etc/environment, а затем считывается файл .profile. Основную среду образуют следующие переменные:

Элемент	Описание
HOME	Полный путь к начальному каталогу пользователя или каталогу HOME. Программа login присваивает этой переменной имя, указанное в файле /etc/passwd.
LANG	Имя текущей действующей локали. Переменная *LANG* первоначально задается в файле /etc/profile во время установки.
NLSPATH	Полный путь к каталогам сообщений.
LOCPATH	Полный путь к таблицам поддержки национальных языков.
PATH	Набор каталогов, в которых команды вида **sh**, **time**, **nice** и **nohup** выполняют поиск команды, путь к которой указан не полностью.
TZ	Информация о часовом поясе. Переменная среды *TZ* первоначально задается в файле /etc/profile - системном профайле входа в систему.

Дополнительная информация о файле /etc/environment приведена в *Справочник по файлам* .

Файл /etc/profile:

Вторым при входе в систему обрабатывается файл /etc/profile.

Файл /etc/profile управляет системными переменными по умолчанию, такими как:

- Переменные экспорта
- Маску создания файла (umask)
- Типы терминала
- Почтовые сообщения, указывающие на поступление новой почты

Системный администратор настраивает файл /etc/profile для всех пользователей системы. Только он может изменять этот файл.

Ниже приведен пример стандартного файла /etc/profile:

```
#Задать маску создания файлов
umask 022
#Сообщить о доставке новой почты
MAIL=/usr/mail/$LOGNAME
#Добавить каталог /bin в последовательность поиска оболочки
PATH=/usr/bin:/usr/sbin:/etc::
#Задать тип терминала
TERM=lft
#Сделать некоторые переменные среды глобальными
export MAIL PATH TERM
```

Дополнительная информация о файле /etc/profile приведена в *Справочник по файлам* .

Файл .profile:

Он находится в домашнем каталоге (`$HOME`) и позволяет вам настраивать собственную среду.

Поскольку файл `.profile` скрыт, для его просмотра воспользуйтесь командой **ls -a**.

После того как программа **login** добавит *LOGNAME* (начальное имя пользователя) и *HOME* (начальный каталог) в среду, будут выполнены команды в файле `$HOME/.profile` (если файл существует). Файл `.profile` содержит ваш профайл, переопределяющий переменные из файла `/etc/profile`. Файл `.profile` часто применяется для установки экспортированных переменных среды и режимов терминала. Можно настраивать среду по своему усмотрению, внося изменения в файл `.profile`. Например, с помощью файла `.profile` можно управлять следующими значениями по умолчанию:

- Список открываемых оболочек
- Внешний вид приглашения
- Звуковой сигнал клавиатуры

Ниже приведен пример стандартного файла `.profile`:

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user::
epath=/home/gsc/e3:
export PATH epath
csh
```

В этом примере заданы операции определения двух переменных пути (*PATH* и *epath*), экспортирования этих переменных и открытия оболочки C (`csh`).

С помощью файла `.profile` (или, если он отсутствует, файла `/etc/profile`) вы можете определять переменные начальной оболочки. Вы можете также настраивать среды других оболочек. Например, с помощью файлов `.cshrc` и `.kshrc` можно настроить оболочки C и Korn, соответственно.

Файл .env:

Наконец, четвертым при входе в систему обрабатывается файл `.env`, если в файле `.profile` есть следующая строка: `export ENV=$HOME/.env`

Файл `.env` позволяет настраивать отдельные переменные вашей среды. Поскольку файл `.env` скрыт, для его просмотра воспользуйтесь командой **ls -a**. Дополнительная информация о команде **ls** приведена в разделе **ls**. Файл `.env` содержит некоторые пользовательские переменные среды, переопределяющие переменные, которые заданы в файле `/etc/environment`. Можно настраивать эти переменные по своему усмотрению, внося изменения в файл `.env`.

Ниже приведен пример стандартного файла `.env`:

```
export  myid=`id | sed -n -e 's/).*$//' -e 's/^.*(//p'`
#задать приглашение: ИД пользователя & имя системы & текущий каталог
if [ $myid = root ]
        then    typeset -x PSCH='#:\${PWD}> '
                PS1="#:\${PWD}> "
        else    typeset -x PSCH='>'
                PS1="$LOGNAME@$UNAME:\${PWD}> "
                PS2=">"
                PS3="#?"
fi
export PS1 PS2 PS3
#задать псевдонимы команд
alias   ls="/bin/ls -CF" \
        d="/bin/ls -Fal | pg" \
        rm="/bin/rm -i" \
        up="cd .."
```

Примечание: При изменении файла `.env` следите за тем, чтобы новые переменные среды не совпадали со стандартными переменными, в частности, *MAIL*, *PS1*, *PS2* и *IFS*.

AIXwindows, файлы запуска:

В разных системах X-сервер и AIXwindows запускаются по-разному.

Поскольку в разных системах X-сервер и AIXwindows запускаются по-разному, для выяснения особенностей своей системы обратитесь к системному администратору. Обычно X-сервер и AIXwindows автоматически запускаются в сценарии оболочки после входа в систему. Тем не менее, в некоторых случаях может потребоваться запустить X-сервер или AIXwindows вручную.

Если после входа в систему на экране показан только один терминал и нет оконной среды, то для запуска X-сервера введите следующее:

```
xinit
```

Примечание: Перед вводом команды убедитесь, что указатель находится в окне с приглашением системы.

Если X-сервер не был запущен, проверьте вместе с системным администратором наличие каталога X11 с исполняемыми программами в пути для поиска. В разных системах правильный путь может быть различным.

Если после входа в систему на экране показано несколько окон без рамок, запустите Администратор окон AIXwindows с помощью следующей команды:

```
mwm &
```

Поскольку AIXwindows допускает настройку как программистами, создающими приложения AIXwindows, так и пользователями, кнопки мыши и прочие функции могут работать не так, как описано в данной документации. Для того чтобы восстановить функции среды AIXwindows по умолчанию, нажмите клавиши:

Alt-Ctrl-Shift-!

Для возврата к предыдущему режиму нажмите эту комбинацию клавиш еще раз. Если ваша система не допускает такой комбинации клавиш, вы можете восстановить поведение по умолчанию из меню по умолчанию.

Файл .xinitrc:

Команда **xinit** применяет для запуска программ X-клиентов настраиваемый сценарий среды. Приложения, запускаемые вместе с AIXwindows, перечислены в файле `.xinitrc` , который находится в вашем домашнем каталоге.

Команда **xinit** обрабатывает сценарии оболочки в следующем порядке:

1. При запуске AIXwindows команда **xinit** сначала получает значение переменной среды *$XINITRC*.
2. Если переменная *$XINITRC* отсутствует, команда **xinit** выполняет поиск сценария оболочки `$HOME/.xinitrc`.
3. Если сценарий оболочки `$HOME/.xinitrc` также не удается найти, команда **xinit** ищет сценарий оболочки `/usr/lib/X11/$LANG/xinitrc`.
4. Если `/usr/lib/X11/$LANG/xinitrc` также не существует, выполняется поиск сценария `/usr/lpp/X11/defaults /$LANG/xinitrc`. Если и этот файл отсутствует, выполняется сценарий `/usr/lpp/X11/defaults/xinitrc`.
5. Сценарий оболочки `xinitrc` запускает такие команды, как **mwm** (Администратор окон AIXwindows), **aixterm** и **xclock**.

Команда **xinit** выполняет следующие действия:

- Запускает X-сервер на текущем дисплее
- Задает переменную среды *$DISPLAY*
- Запускает файл `xinitrc` для запуска клиентов X

В следующем примере показана настраиваемая часть файла `xinitrc`:

```
# Этот сценарий вызывается /usr/lpp/X11/bin/xinit

.
.
.
#************************************************************
#  Запускает X-клиенты. Измените следующие строки, добавив     *
#  нужные команды. По умолчанию загружаются                        *
#  аналоговые часы (xclock), эмулятор терминала lft            *
#  (aixterm) и Администратор окон Motif (mwm).                 *
#  (mwm).  *
#************************************************************
exec mwm
```

Файл .Xdefaults:

Интерфейс AIXwindows можно настраивать, изменяя файл `.Xdefaults`. Таким образом задаются такие визуальные характеристики AIXwindows, как цвета и шрифты.

Многие параметры внешнего вида приложений для системы Windows и способов работы с ними управляются набором переменных под общим названием *ресурсы*. Внешний вид и поведение ресурса определяется присвоенным ему значением. Предусмотрено несколько типов значений. Например, ресурсам, управляющим цветом, можно присвоить такие предопределенные значения, как *DarkSlateBlue* и *Black*. Ресурсы, определяющие размер, имеют числовые значения. Некоторые ресурсы принимают булевские значение (*Истина* или *Ложь*).

Если файла `.Xdefaults` нет в домашнем каталоге, вы можете создать его любым текстовым редактором. Это даст вам возможность настраивать значения ресурсов по своему усмотрению. Пример файла со значениями по умолчанию под названием `Xdefaults.tmpl` находится в каталоге `/usr/lpp/X11/defaults`.

В следующем примере показана часть стандартного файла `.Xdefaults`:

```
*AutoRaise: on
*DeIconifyWarp: on
*warp:on
*TitleFont:andysans12
*scrollBar: true
*font: Rom10.500
Mwm*menu*foreground: black
Mwm*menu*background: CornflowerBlue
Mwm*menu*RootMenu*foreground: black
Mwm*menu*RootMenu*background: CornflowerBlue
Mwm*icon*foreground: grey25
Mwm*icon*background: LightGray
Mwm*foreground: black
Mwm*background: LightSkyBlue
Mwm*bottomShadowColor: Blue1
Mwm*topShadowColor: CornflowerBlue
Mwm*activeForeground: white
Mwm*activeBackground: Blue1
Mwm*activeBottomShadowColor: black
Mwm*activeTopShadowColor: LightSkyBlue
Mwm*border: black
Mwm*highlight:white

aixterm.foreground: green
aixterm.background: black
aixterm.fullcursor: true
```

```
aixterm.ScrollKey: on
aixterm.autoRaise: true
aixterm.autoRaiseDelay: 2
aixterm.boldFont:Rom10.500
aixterm.geometry: 80x25
aixterm.iconFont: Rom8.500
aixterm.iconStartup: false
aixterm.jumpScroll: true
aixterm.reverseWrap: true
aixterm.saveLines: 500
aixterm.scrollInput: true
aixterm.scrollKey: false
aixterm.title: AIX
```

Файл .mwmrc:

Большая часть изменяемых свойств может быть задана в ресурсах файла `.Xdefaults`. Однако раскладка клавиатуры, настройка клавиш мыши и определение меню для администратора окон указывается в дополнительном файле `.mwmrc`, на который есть ссылка в файле `.Xdefaults`.

Если файла `.mwmrc` нет в вашем домашнем каталоге, скопируйте его следующей командой:

```
cp /usr/lib/X11/system.mwmrc .mwmrc
```

Поскольку файл `.mwmrc` переопределяет системные значения файла `system.mwmrc`, заданные вами свойства не будут изменяться другими пользователи.

Ниже приведен фрагмент стандартного файла `system.mwmrc`:

```
#  ФАЙЛ ОПИСАНИЯ РЕСУРСОВ mwm ПО УМОЛЧАНИЮ (system.mwmrc)
#
# Описания панелей меню
#
# Описание корневого меню

Menu RootMenu
{ "Корневое меню"    f.title
  no-label                                      f.separator
  "Новое окно"       f.exec "aixterm &"
  "Поднять нижнее"   f.circle_up
  "Утопить верхнее"  f.circle_down
  "Обновить"         f.refresh
  no-label                                      f.separator
  "Перезапустить"    f.restart
  "Выход"            f.quit_mwm
}

# Описание меню окна по умолчанию

Menu DefaultWindowMenu MwmWindowMenu
{ "Восстановить"   _R   Alt<Key>F5              f.normalize
  "Переместить"      _M   Alt<Key>F7              f.move
  "Размер"      _S   Alt<Key>F8             f.resize
  "Свернуть"  _n   Alt<Key>F9             f.minimize
  "Развернуть"  _x   Alt<Key>F10            f.maximize
  "Вниз"     _L   Alt<Key>F3             f.lower
  no-label                                      f.separator
  "Закрыть"     _C   Alt<Key>F4             f.kill
}

# Меню окна без клавиш быстрого доступа
Menu NoAccWindowMenu
{
  "Восстановить"    _R     f.normalize
  "Переместить"     _M     f.move
  "Размер"          _S     f.resize
  "Свернуть"        _n     f.minimize
  "Развернуть"      _x     f.maximize
```

```
  "Вниз"            _L     f.lower
  no-label                                    f.separator
  "Закрыть"         _C     f.kill
}
Keys DefaultKeyBindings
{
  Shift<Key>Escape             icon|window        f.post_wmenu
  Meta<Key>space               icon|window        f.post_wmenu
  Meta<Key>Tab                 root|icon|window   f.next_key
  Meta Shift<Key>Tab           root|icon|window   f.prev_key
  Meta<Key>Escape              root|icon|window   f.next_key
  Meta Shift<Key>Escape        root|icon|window   f.prev_key
  Meta Ctrl Shift<Key>exclam   root|icon|window   f.set_behavior
}
#
# описание настройки клавиш мыши
#
Buttons DefaultButtonBindings
{
  <Btn1Down>        frame|icon             f.raise
  <Btn3Down>         frame|icon             f.post_wmenu
  <Btn1Down>        root                   f.menu  RootMenu
  <Btn3Down>         root                   f.menu  RootMenu
  Meta<Btn1Down>     icon|window            f.lower
  Meta<Btn2Down>    window|icon            f.resize
  Meta<Btn3Down>     window                 f.move
}
Buttons PointerButtonBindings
{
  <Btn1Down>        frame|icon             f.raise
  <Btn2Down>        frame|icon             f.post_wmenu
  <Btn3Down>        frame|icon             f.lower
  <Btn1Down>        root                   f.menu  RootMenu
  Meta<Btn2Down>    window|icon            f.resize
  Meta<Btn3Down>    window|icon            f.move
}
#
#  КОНЕЦ ФАЙЛА ОПИСАНИЯ РЕСУРСОВ mwm
#
```

Экспорт переменных оболочки (команда export):

Переменная оболочки называется *локальной*, если она известна только той оболочке, которая ее создала. Если вы запустите новую оболочку, то переменные старой оболочки будут ей неизвестны. Если вы хотите, чтобы переменные старой оболочки были известны во вновь открываемых оболочках, экспортируйте переменные, сделав их *глобальными*.

Сделать локальные переменные глобальными можно с помощью команды **export**. Для автоматического преобразования локальных переменных оболочки в глобальные экспортируйте их в свой файл `.profile`.

Примечание: Переменные можно экспортировать в дочерние, но не в родительские оболочки.

Примеры:

- Для того чтобы сделать локальную переменную *PATH* глобальной, введите:
  ```
  export PATH
  ```
- Для просмотра всех экспортированных переменных введите:
  ```
  export
  ```

  Появится приблизительно следующая информация:

```
DISPLAY=unix:0
EDITOR=vi
ENV=$HOME/.env
HISTFILE=/u/denise/.history
HISTSIZE=500
HOME=/u/denise
LANG=En_US
LOGNAME=denise
MAIL=/usr/mail/denise
MAILCHECK=0
MAILMSG=**YOU HAVE NEW MAIL.
USE THE mail COMMAND TO SEE YOUR MAILPATH=/usr/mail/denise?denise has mail !!!
MAILRECORD=/u/denise/.Outmail
PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/u/bin1
PWD=/u/denise
SHELL=/bin/ksh
```

Изменение шрифта по умолчанию (команда chfont):

Для того чтобы изменить шрифт, по умолчанию устанавливаемый при запуске системы, вызовите команду **chfont** или **smit**. Все доступные шрифты в системе определяются специальным файлом - *палитрой шрифтов*.

Примечание: Для запуска команды **chfont** у вас должны быть права доступа root.

Команда chfont

Ниже приведены примеры применения команды **chfont**:

- Для того чтобы сделать пятый шрифт палитры активным шрифтом, введите:
  ```
  chfont -a5
  ```
- Для изменения шрифта на полужирный курсив Roman того же размера введите:
  ```
  chfont -n /usr/lpp/fonts/It114.snf /usr/lpp/fonts/Bld14.snf  /usr/lpp/fonts/Rom14.snf
  ```

Сведения о синтаксисе приведены в описании команды **chfont** книги *Справочник по командам, том 1*.

команда smit

Команду **chfont** также можно выполнить с помощью **smit**.

Для выбора активного шрифта введите:

```
smit chfont
```

Для выбора палитры шрифта введите:

```
smit chfontpl
```

Изменение управляющих клавиш (команда stty):

С помощью команды **stty** можно изменить управляющие клавиши терминала.

Изменения будут действовать до завершения работы в системе. Если вы хотите сделать их постоянными, запишите их в файл `.profile`.

Примеры:

- Для того чтобы сделать Ctrl-Z клавишей прерывания, введите:
  ```
  stty intr ^Z
  ```
  Не забудьте указать пробел между `intr` и `^Z`.
- Для восстановления управляющих клавиш по умолчанию введите:
  ```
  stty sane
  ```
- Для просмотра текущих параметров введите:

```
stty -a
```

Сведения о синтаксисе приведены в описании команды **stty** книги *Справочник по командам, том 5*.

Изменение системного приглашения:

Вы можете изменить системное приглашение.

В оболочке предусмотрены следующие переменные приглашения:

Элемент	Описание
PS1	Приглашение, применяемое в обычном режиме
PS2	Приглашение, применяемое в случае, когда оболочка ожидает продолжения ввода
PS3	Приглашение пользователя root

Вы можете изменить любое приглашение, изменив значение соответствующей переменной оболочки. Изменение будет действовать до завершения работы в системе. Если вы хотите сделать их постоянными, запишите их в файл `.env`.

Примеры:

- Для просмотра текущего значения переменной *PS1* введите:
  ```
  echo "Приглашение $PS1"
  ```
  Появится приблизительно следующая информация:
  ```
  Приглашение: $
  ```
- Для изменения приглашения на `Ready>` введите:
  ```
  PS1="Ready> "
  ```
- Для изменения приглашения на продолжение ввода на строку `Enter more->` введите:
  ```
  PS2="Enter more->"
  ```
- Для изменения приглашения root на `Root->` введите:
  ```
  PS3="Root-> "
  ```

Справочник по системе BSD

Приведенная здесь информация предназначена для системных администраторов, знакомых с операционной системой BSD UNIX 4.3 или System V. Ниже перечислены различия и сходства этих систем с AIX.

В этом приложении обсуждаются следующие вопросы:

Основные сведения о BSD

Перед началом работы с BSD (Berkeley Software Distribution) необходимо ознакомиться с некоторым различием между BSD и AIX.

Введение в AIX системных администраторов BSD:

Ниже приведены советы для системных администраторов Berkeley Software Distribution (BSD), начинающих работать с AIX.

- Входите в систему под именем root с графической консоли.
- Если у вас нет навыков управления системой, выполняйте все задачи администрирования с системной консоли. На системной консоли работать проще, чем на удаленном терминале. Накопив достаточно опыта работы в системе, вы можете перейти к удаленным терминалам (xterm или текстовым).
- Используйте различные средства управления системой AIX. К ним относятся:

– Инструмент управления системой (SMIT). SMIT для системного администратора представляет собой интерфейс команд настройки и управления системой. SMIT позволяет выполнить большую часть задач по управлению системой.
– Администратор объектных данных (ODM). ODM позволяет выполнять процедуры работы с объектами базы данных ODM. В базе данных ODM хранится информация о конфигурации устройств
– Контроллер системных ресурсов (SRC). SRC предоставляет единый интерфейс для управления демонами и другими ресурсами системы.

Понятия, связанные с данным:

“Настройка большого числа устройств” на стр. 387
Под устройствами понимаются как аппаратные устройства, такие как принтеры, дисководы, адаптеры, шины и корпуса, так и псевдоустройства, такие как специальный файл ошибок и специальный пустой файл. Драйверы устройств расположены в каталоге `/usr/lib/drivers`.

“Контроллер системных ресурсов” на стр. 183
Контроллер системных ресурсов (SRC) предоставляет набор команд и подпрограмм, которые упрощают задачи создания и управления подсистемами для системного администратора и программиста.

Наиболее существенные различия между 4.3 BSD и AIX:

Ниже описаны основные различия между AIX и системами BSD 4.3.

В операционной системе AIX для запуска сетевых демонов применяется файл `/etc/rc.tcpip`, а не файл `/etc/rc.local`. Сценарий `/etc/rc.tcpip` запускается из файла `/etc/inittab`, а не из файла `/etc/rc`.

Если в системе запущен Контроллер системных ресурсов (SRC), то демоны TCP/IP работают под его управлением. Если вы не хотите, чтобы демоны TCP/IP работали под управлением SRC, то вызовите команду **smit setbootup_option** и выберите конфигурацию **rc**, применяемую в системах BSD.

Ниже перечислены функции управления сетью BSD 4.3, поддерживаемые операционной системой AIX:

- Средства ведения системного протокола уровня ядра (SYSLOG)
- Права доступа к сокетам домена UNIX.

Хранилище данных конфигурации

В системах BSD 4.3 большая часть данных конфигурации хранится в текстовых файлах. Связанная информация записывается на одной строке, причем обработка записей (сортировка и поиск) выполняется над самим текстовым файлом. Записи могут быть разной длины; они разделяются символом перевода строки. В системах BSD 4.3 есть средства преобразования больших текстовых файлов в формат базы данных (dbm). Соответствующие библиотечные функции выполняют поиск в файлах dbm, если они существуют, или в исходном текстовом файле, если файлы dbm не найдены.

В AIX часть информации о конфигурации хранится в обычных текстовых файлах, а часть - в файле с *разделами*. Раздел - это группа из нескольких строк, в которых хранится связанная информация. Все элементы снабжаются метками, упрощающими чтение файла.

Данная операционная поддерживает хранение информации о пользователях и паролях в формате dbm. Тем не менее, в некоторых файлах, таких как `/etc/passwd`, `/etc/group` и `/etc/inittab`, в AIX информация хранится в традиционном формате, а не в формате с несколькими разделами.

Остальные данные о конфигурации в данной операционной системе хранятся в файлах ODM. Информацию в файлах ODM можно просматривать и изменять с помощью Инструмента управления системой (SMIT). Кроме того, для просмотра этих файлов можно использовать непосредственно команды ODM. Для запроса информации из файлов ODM служат следующие команды:

- **odmget**
- **odmshow**.

Следующие команды ODM позволяют изменить файлы ODM:

- **odmadd**
- **odmcreate**
- **odmdrop**
- **odmchange**
- **odmdelete**.

Внимание: Неправильное изменение файлов ODM может привести к сбою системы, после которого вам не удастся ее перезапустить. Напрямую изменять файлы ODM с помощью команд ODM следует только в том случае, если это нельзя сделать с помощью SMIT.

Управление настройкой

При запуске системы, работающей под управлением данной операционной системы, администратор настройки вызывает ряд команд настройки. Эти команды называются *методами*. Методы определяют устройства, установленные в системе, и обновляют соответствующие файлы ODM, расположенные в каталоге `/etc/objrepos`.

Специальные файлы устройств не устанавливаются в каталог `/dev` заранее. Некоторые специальные файлы, например, для жестких дисков, создаются автоматически во время настройки при запуске системы. Другие специальные файлы, например для текстовых терминалов, создаются системным администратором с помощью меню **Устройства** Инструмента управления системой (SMIT). Эта информация также сохраняется в файлах ODM.

Управление диском

В AIX диски называются *физическими томами*. Разделы при этом называются *логическими томами*. Как и в BSD 4.3, один физический том может содержать несколько логических томов. Однако, в отличие от BSD 4.3, в данной операционной системе один логический том в AIX может занимать несколько физических томов. Для этого несколько физических томов должны быть объединены в *группу томов*, в которой можно создавать логические тома.

В данной операционной системе для управления файловыми системами и томами применяются следующие команды:

- **crfs**
- **varyonvg**
- **varyoffvg**
- **lsvg**
- **importvg**
- **exportvg**.

Кроме того, могут применяться следующие команды BSD 4.3:

- **mkfs**
- **fsck**
- **fsdb**
- **mount**
- **umount**.

Различия между командами BSD 4.3 и командами данной операционной системы AIX обсуждаются в разделе “Файловые системы - информация для системных администраторов BSD 4.3” на стр. 341.

BSD 4.3 хранит список файловых систем в файле `/etc/fstab`. В AIX каждой файловой системе соответствует раздел файла `/etc/filesystems`.

Команда tn3270

Команда **tn3270** представляет собой связь с командой **telnet**, но для определения раскладки клавиатуры 3270 она использует файл /etc/map3270 и текущее значение переменной среды *TERM*. Таким образом, команда **tn3270** полностью совпадает с аналогичной командой, применяемой в BSD.

Если вы хотите изменить Esc-последовательности, применяемые командами **tn3270**, **telnet** и **tn**, задайте переменную среды *TNESC* перед запуском этих команд.

Новые команды

Для управления конфигурацией и работы с дисками в данной операционной системе предусмотрено около 150 команд, не применявшихся в BSD 4.3.

Запуск Данная операционная система поддерживает автоматическую идентификацию и настройку устройств. В результате процесс запуска данной операционной системы существенно отличается от запуска систем BSD 4.3. Кроме ядра на диск RAM записывается также образ файловой системы загрузки и информация о предыдущей конфигурации базовых устройств. После этого загружается информация о текущей конфигурации и проверяется возможность доступа к физическим томам. Затем определяется устройство, в котором находится пространство подкачки. Выполняется проверка жесткого диска с корневой файловой системой. После этого операционная система заменяет корневую файловую систему из RAM на корневую файловую систему с жесткого диска и завершает процедуру запуска, включающую настройку устройств.

Проверка прав доступа пользователя

В операционных системах BSD версии 4.3 и AT&T UNIX версии младше SVR4 все идентификационные данные пользователей, в том числе зашифрованные пароли, хранятся в файле /etc/passwd. Как правило, права на чтение файла /etc/passwd есть у всех пользователей.

В системах SVR4 зашифрованные пароли хранятся не в файле /etc/passwd, а в файле /etc/shadow. Права на чтение файла /etc/shadow есть только у пользователей с правами доступа root и у отдельных программ (например, у программы /bin/login).

В AIX зашифрованные пароли хранятся в файле /etc/security/passwd. Кроме того, в каталоге /etc/security находятся файлы user и limits. Эти три файла хранят информацию о разрешенных пользователям способах доступа к системе (например, с помощью команд **rlogin** и **telnet**) и ограничениях на ресурсы, заданные для пользователей (например, ограничение на размер файлов и адресного пространства).

Печать

В AIX поддерживается большая часть команд печати BSD 4.3 с минимальными отличиями. Одно из отличий заключается в том, что в AIX применяется файл конфигурации /etc/qconfig.

Подсистема построчной печати, реализованная в данной операционной системе, может взаимодействовать с подсистемой построчной печати BSD 4.3, включая передачу заданий печати в системы BSD 4.3 и печать заданий, полученных из систем BSD 4.3.

Оболочки

Данная операционная система поддерживает оболочки Bourne, C и Korn. Полное имя оболочки Bourne - /bin/bsh. Файл /bin/sh - это жесткая связь с файлом /bin/ksh. Этот файл может изменяться администратором.

AIX не поддерживает применение команд **setuid** и **setgid** в сценариях оболочек.

Примечание:

1. В AIX нет сценариев оболочек, основанных на /bin/sh. Тем не менее, в других системах многие сценарии подразумевают, что /bin/sh - это оболочка Bourne.
2. Несмотря на то, что оболочки Bourne и Korn похожи, оболочка Korn не является точной копией оболочки Bourne.

Ссылки, связанные с данной:

“Команды для системного администрирования BSD 4.3” на стр. 337
В приведенном ниже списке перечислены команды операционной системы, предназначенные для управления средой AIX.

Таблица сравнения файлов 4.3 BSD, SVR4 и AIX:

В приведенной ниже таблице перечислены имена и функции файлов, применяемых в BSD 4.3, SVR4 и в AIX.

Таблица 61. Таблица сравнения файлов

Файл BSD 4.3	Файл SVR4	Файл AIX	База данных	Тип (odm/dbm)
L-Devices	Устройства	Устройства	нет	
L-dialcodes	Dialcodes	Dialcodes	нет	
L.cmds	Permissions	Permissions	нет	
L.sys	Systems	Systems	нет	
USERFILE	Permissions	Permissions	нет	
aliases	mail/namefiles	aliases	aliasesDB/DB	dbm
fstab	vfstab	filesystems	нет	
ftpusers	ftpusers	ftpusers	нет	
gettytab		нд		
group	group	group	нет	
hosts	hosts	hosts	нет	
hosts.equiv	hosts.equiv	hosts.equiv	нет	
inetd.conf	inetd.conf	inetd.conf	нет	
map3270	нд	map3270	нет	
motd	motd	motd	нет	
mtab	mnttab	нд	нет	
named.boot	named.boot	named.boot	нет	
named.ca		named.ca	нет	
named.hosts		named.data (см. примечание)	нет	
named.local		named.local	нет	
named.pid	named.pid	named.pid	нет	
named.rev		named.rev	нет	
networks	networks	networks	нет	
passwd	passwd	passwd	нет	
printcap	qconfig	qconfig		
protocols		protocols	нет	
remote	remote	remote	нет	
resolv.conf	resolv.conf	resolv.conf	нет	
sendmail.cf	sendmail.cf	sendmail.cf	sendmail.cfDB	другой
services		services	нет	
shells	shells	нд		
stab		нд		
syslog.conf		syslog.conf	нет	
syslog.pid		syslog.pid	нет	
diff, команда	terminfo	terminfo		
ttys	ttys	нд	да	odm
types		нд		
utmp	utmp	utmp		
vfont		нд		

Таблица 61. Таблица сравнения файлов (продолжение)

Файл BSD 4.3	Файл SVR4	Файл AIX	База данных	Тип (odm/dbm)
vgrindefs		vgrindefs		
wtmp	wtmp	wtmp		

Примечание: Имена файлов named.ca, named.hosts, named.local и named.rev можно изменить в файле named.boot. Однако в документации по данной операционной системе применяются именно эти имена файлов.

Преобразование имен и адресов:

Функции **gethostbyname** и **gethostbyaddr** из библиотеки libc обеспечивают поддержку DNS, NIS (бывшая служба Yellow Pages) и базы данных /etc/hosts.

Если файл /etc/resolv.conf существует, сначала выполняется обращение к серверу имен. Если имя не будет преобразовано, и работает служба NIS, то проверяются базы данных этой службы. Если служба NIS не запущена, просматривается файл /etc/hosts.

Электронная документация и описание команды man - информация для системных администраторов BSD 4.3:

AIX поддерживает команды **man-k**, **apropos** и **whatis**, но база данных, используемая этими командами, должна быть создана с помощью команды **catman-w**.

Команда **man** вначале выполняет текстовый поиск в файлах /usr/man/cat? . Затем она выполняет поиск информации в формате **nroff** в файлах /usr/man/man? . И к обычному тексту, и к информации в формате **nroff** можно добавлять собственные страницы.

Примечание:

- Текстовая справка для команды **man** не поставляется вместе с системой. Для создания базы данных необходимо запустить команду **catman**. В базе данных может быть как обычный текст в файлах /usr/man/cat? так и текст в формате **nroff** (в файлах /usr/man/man?) .
- Для того чтобы команда **man** могла работать с документацией в формате **nroff**, в системе должна быть установлена лицензионная программа форматирования текста.

Дополнительная информация об этих командах приведена в разделах **man**, **apropos**, **whatis** и **catman**.

NFS и NIS (бывшая служба Yellow Pages) - информация для системных администраторов BSD 4.3:

Ниже описаны NFS и NIS для системных администраторов BSD 4.3.

Демоны сетевой файловой системы (NFS) и служб информации о сети (NIS) запускаются из файла /etc/rc.nfs. Для запуска демонов NFS и NIS в файле /etc/rc.tcpip необходимо указать опцию запуска демона **portmap**. По умолчанию файл /etc/rc.nfs не вызывается файлом /etc/inittab. Если вы планируете применять сценарий /etc/rc.nfs, добавьте строку вызова этого сценария в файл /etc/inittab после сценария /etc/rc.tcpip.

Если в системе применяется служба NIS, то добавьте запись root перед записью +:: (плюс, двоеточие, двоеточие) в файл /etc/passwd и запись system перед записью +:: в файл /etc/group. Это позволит системному администратору входить в систему и изменять ее, даже если система не может установить соединение с сервером NIS.

Для настройки NFS воспользуйтесь командой SMIT **smit nfs**. В меню SMIT Служба информации о сети (ранее называвшаяся Yellow Pages) называется NIS. Большинство команд NFS и NIS находятся в каталогах /etc и /usr/etc.

Иногда для определения семейств и типов компьютеров в среде NFS применяется команда **arch**. Например, при работе с системой IBM® RS/6000 укажите в качестве идентификатора семейства (CPU) значение **power**, а в качестве типа компьютера - **ibm6000**.

Пароль пользователя - информация для системных администраторов BSD 4.3:

При вызове пользователем root команды **/bin/passwd** в AIX пользователю будет предложено ввести текущий пароль пользователя root.

Ниже приведен пример применения команды **/bin/passwd**:

```
# passwd cslater
Изменение пароля для "cslater"
Введите пароль root или
Старый пароль для cslater:
Новый пароль для cslater:
Повторно введите новый
пароль cslater:
#
```

В BSD 4.3 эта команда не запрашивает текущий пароль пользователя root. Пример работы версии этой команды в BSD 4.3 приведен ниже:

```
# passwd cslater
Новый пароль:
Введите новый пароль еще раз:
#
```

Управление BSD

Существует несколько команд для BSD, предназначенные для измерения производительности, печати и управления системой.

Учет ресурсов - информация для системных администраторов BSD 4.3:

В AIX файлы учета ресурсов, хранящиеся в каталоге `/usr/lib/acct`, а также утилиты составления отчетов, хранящиеся в каталоге `/usr/lib/sa`, включают файлы и утилиты, поставляемые в составе AT&T System V Release 4 (SVR4), а также дополнительные утилиты учета BSD 4.3.

Большинство команд учета ресурсов хранятся в каталоге `/usr/lib/acct`. Для включения функции учета ресурсов служит команда `/usr/lib/acct/startup`. Если учет ресурсов не ведется, то такие команды как **lastcomm(1)** не позволяют получить соответствующую информацию.

В данной операционной системе предусмотрены следующие функции учета BSD 4.3:

Элемент	Описание
last(1)	Позволяет узнать время последнего входа пользователей в систему и терминалы, с которыми работают пользователи
lastcomm(1)	Показывает последние выполненные команды в обратном хронологическом порядке
acct(3)	Включает и выключает учет ресурсов процессов
ac(8)	Учет информации о входе в систему
accton(8)	Включает или выключает учет ресурсов в системе
sa(8)	Выполняет общее обслуживание файлов учета ресурсов системы.

Данная операционная система поддерживает следующие команды и библиотечные функции учета ресурсов System V Interface Definition (SVID) Issue II:

Элемент	Описание
acctcms(1)	Создает краткий отчет об использовании команд на основании записей учета ресурсов
acctcom(1)	Показывает выбранные записи учета ресурсов процессов
acctcon1(1)	Преобразует записи о входе в систему и выходе из системы в записи о сеансах
acctcon2(1)	Преобразует записи о входе в систему и выходе из системы в итоговые записи учета ресурсов
acctdisk(1)	Создает итоговые записи учета ресурсов на основании вывода команды **diskusg(1)**
acctmerg(1)	Объединяет итоговые файлы учета ресурсов в промежуточный файл
accton(1)	Включает функцию учета ресурсов
acctprc1(1)	Обрабатывает информацию об учете ресурсов, полученную от команды **acct(3)**
acctprc2(1)	Обрабатывает вывод команды **acctprc1(1)** и формирует итоговые записи учета ресурсов
acctwtmp(1)	Обрабатывает записи учета времени соединения
chargefee(1)	Формирует счет на оплату для заданного имени пользователя
ckpacct(1)	Проверяет размер файла `/usr/adm/pacct`
diskusg(1)	Формирует информацию об учете ресурсов дисков
dodisk(1)	Выполняет функцию учета ресурсов дисков
fwtmp(1)	Преобразует двоичные записи (из файла `wtmp`) в формат ASCII. **Примечание:** Файл `wtmp` находится в каталоге `/var/adm`.
lastlogin(1)	Обновляет информацию о дате последнего входа в систему каждого пользователя
monacct(1)	Создает файлы с ежемесячным итоговым отчетом
prctmp(1)	Печатает файл с информацией о сеансе, созданный командой **acctcon1(1)**
prdaily(1)	Форматирует отчет со вчерашней информацией об учете ресурсов
prtacct(1)	Форматирует и печатает любой файл с итоговым отчетом
runacct(1)	Ежедневно собирает данные учета ресурсов
shutacct(1)	Вызывается программой завершения работы системы для остановки подсистемы учета ресурсов и занесения в протокол информации о причинах прекращения учета
startup(1)	Вызывается программой инициализации системы для запуска подсистемы учета ресурсов
turnacct(1)	Включает и выключает функцию учета ресурсов процессов
wtmpfix(1)	Исправляет системное время в файле в формате `wtmp`

Резервное копирование - информация для системных администраторов BSD 4.3:

Системные администраторы BSD 4.3 могут выполнять резервное копирование данных.

С помощью команд **tar** и **cpio** можно перемещать данные между системами. Команда **tar** для AIX не полностью совместима с командой 4.3 BSD **tar**. При чтении данных из конвейера в данной реализации команды **tar** необходимо указывать флаг **-B** (блоковый ввод). Команда **cpio** AT&T совместима с соответствующей командой данной операционной системы.

AIX может выполнять чтение и запись в формате команды **dump** и **restore**. Например, команда **backup** для AIX имеет следующий синтаксис:

```
backup -0uf Device файловая система
```

аналогична команде **dump** BSD 4.3 с синтаксисом:

```
dump 0uf Device файловая система
```

Аналогично, команда **restore** для AIX имеет следующий синтаксис:

```
restore -mivf устройство
```

аналогична команде **restore** BSD 4.3 в формате:

```
restore ivf устройство
```

В AIX также поддерживаются команды 4.3 BSD **rdump** и **rrestore**. Единственным различием между этими двумя версиями является то, что в AIX перед каждым аргументом следует указывать **-** (дефис). Например, следующая команда:

```
rdump -0 -f orca:/dev/rmt0 /dev/hd2
```

эквивалентна команде BSD 4.3, приведенной ниже:

```
rdump 0f orca:/dev/rmt0 /dev/hd2
```

Команда **backup** для AIX имеет следующий синтаксис:

```
backup -0f /dev/rmt0 /dev/hd2
```

аналогична команде 4.3 BSD **dump** с синтаксисом:

```
dump 0f /dev/rmt0 /dev/hd2
```

Поддержка накопителей на магнитной ленте IBM не SCSI

Данная операционная система не поддерживает накопители на магнитной ленте SCSI других фирм. Однако вы можете добавить свой собственный заголовок и интерфейс, который будет использоваться драйвером SCSI фирмы IBM.

Понятия, связанные с данным:

“Резервное копирование системы” на стр. 21
После завершения настройки системы рекомендуется создать резервную копию всех файловых систем, каталогов и файлов. При резервном копировании файловых систем можно восстановить файлы или файловые системы в случае сбоя жесткого диска. Есть несколько методов резервного копирования данных.

Информация, связанная с данной:

Добавление в систему неподдерживаемого устройства

Запуск системы - информация для системных администраторов BSD 4.3:

Ниже описан запуск системы AIX для системных администраторов BSD 4.3.

В системах BSD 4.3 программа **init** вызывается на последнем этапе процедуры запуска. Основная задача программы **init** заключается в создании процессов для всех портов терминалов. Список доступных портов терминалов считывается из файла `/etc/ttys`.

В System V программа **init** запускается при инициализации системы. Процесс **init** запускает другие процессы, указанные в файле `/etc/inittab`.

Данная операционная система использует ту же процедуру инициализации, что и System V. Вы можете изменить файл `/etc/inittab` вручную, с помощью команды **telinit** или с помощью следующих команд:

Элемент	Описание
chitab(1)	Изменяет записи в файле `/etc/inittab`
lsitab(1)	Показывает записи из файла `/etc/inittab`
mkitab(1)	Создает записи в файле `/etc/inittab`
rmitab(1)	Удаляет записи из файла `/etc/inittab`

Изменения, внесенные в файл `/etc/inittab`, вступают в силу при следующем запуске системы или при вызове команды **telinit q**.

Поиск и просмотр файлов - информация для системных администраторов BSD 4.3:

Ниже приведен список файловых команд BSD, поддерживаемых в AIX.

В данной операционной системе поддерживаются следующие команды BSD 4.3, предназначенные для работы с файлами:

- **which**
- **whereis**
- **what**

- **file**.

AIX не поддерживает формат **fast find** команды **find**, применяемый в BSD 4.3. В настоящее время в AIX отсутствует аналогичная функция. Вместо нее может применяться следующий сценарий оболочки **ffind**:

```
#!/bin/bsh
PATH=/bin
for dir in /bin /etc /lib /usr
do
find $dir -print | egrep $1
done
```

Формат вызова **ffind**:

`ffind` *имя-файла*

Пространство подкачки - информация для системных администраторов BSD 4.3:

Для работы с пространством подкачки предусмотрены следующие команды.

Элемент	Описание
chps(1)	Изменяет атрибуты пространства подкачки
lsps(1)	Показывает список атрибутов пространства подкачки
mkps(1)	Увеличивает объем системного пространства подкачки
rmps(1)	Удаляет пространство подкачки из системы
swapoff(1)	Отключает одно или несколько пространств подкачки
swapon(1)	Указывает дополнительные устройства для подкачки.

При создании пространства подкачки большого объема размещайте по одному логическому тому подкачки на каждом жестком диске. Это позволит распределить нагрузку между различными устройствами.

Изменение запуска по умолчанию для поддержки конфигурации 4.3 BSD ASCII:

Для управления сетевыми интерфейсами в данной операционной системе может применяться программа SMIT, файлы ODM или текстовые файлы конфигурации BSD 4.3.

Для управления сетевыми интерфейсами с помощью текстовых файлов конфигурации BSD 4.3 удалите символы комментария после следующего заголовка в файле `/etc/rc.net`:

```
# Part II - Traditional
Configuration
```

Если вы хотите работать с текстовыми файлами конфигурации и обеспечить поддержку SRC, то измените файл `/etc/rc.net`, удалив символ комментария перед командами **hostname**, **ifconfig** и **route** с соответствующими параметрами.

Если вы хотите работать с текстовыми файлами конфигурации без поддержки SRC, то с помощью команды **smit setbootup_option** перейдите к конфигурации **rc**, используемой в системах BSD. В этом случае при запуске системы параметры конфигурации считываются из файла `/etc/rc.bsdnet`. Измените файл `/etc/rc.bsdnet`, удалив символы комментария, стоящие перед командами **hostname**, **ifconfig** и **route** с соответствующими параметрами.

Дополнительные опции для команд ifconfig и netstat:

Ниже приведен список дополнительных опций для команд **ifconfig** и **netstat**.

Команда **ifconfig** для AIX имеет следующие дополнительные опции:

mtu Переменная *mtu* задает размер максимального блока передачи (MTU), применяемого в локальной

сети (и в локальных подсетях), а также MTU, применяемый в удаленных подсетях. Для обеспечения совместимости с Ethernet и другими сетями задайте для Token-Ring и Ethernet значение *mtu*, равное 1500.

allcast Флаг **allcast** управляет стратегией оповещения в сети Token-Ring. Установка флага **allcast** позволяет оптимизировать передачу данных через мосты Token-Ring. Сброс флага **allcast** (`-allcast`) минимизирует объем избыточных данных, передаваемых по сети Token-Ring.

Команда **netstat** для AIX имеет флаг **-v**. Команда **netstat -v** выводит статистическую информацию о драйвере, например, количество переданных и принятых байт, а также количество ошибок передачи и приема. Дополнительная информация о командах **ifconfig** и **netstat** приведена в разделах **ifconfig** и **netstat**.

Дополнительные команды управления сетью:

В AIX предусмотрены следующие дополнительные команды.

Элемент	Описание
securetcpip	Сценарий оболочки **securetcpip** включает режим управления доступом, улучшающий защиту сети. Он запрещает применение отдельных незащищенных программ TCP/IP, таких как **tftp**, **rcp**, **rlogin** и **rsh**. Кроме того, он ограничивает доступ к файлу `.netrc`.
gated	Команда **gated** обеспечивает поддержку MIB для SNMP.
no	Команда **no** задает опции сети, в том числе: **dogticks** Задает точность таймера для функций **ifwatchdog** **subnetsarelocal** Определяет, относится ли адрес пакета к локальной сети **ipsendredirects** Указывает, должно ли ядро отправлять сигналы перенаправления **ipforwarding** Указывает, должно ли ядро пересылать пакеты **tcp_ttl** Задает значение TTL для пакетов TCP **udp_ttl** Задает значение TTL для пакетов UDP **maxttl** Задает значение TTL для пакетов RIP **ipfragttl** Задает TTL для фрагментов IP **lowclust** Задает нижнее ограничение для пула **mbuf** кластера **lowmbuf** Задает нижнее ограничение для пула **mbuf** **thewall** Задает максимальный объем памяти, выделяемый для **mbuf** и пула **mbuf** кластера **arpt_killc** Задает время в минутах, по истечении которого удаляется неактивная запись протокола преобразования адресов (ARP)
iptrace	Команда **iptrace** обеспечивает трассировку пакетов протоколов Internet на уровне интерфейса.
ipreport	Команда **ipreport** преобразует данные трассировки в формат, удобный для чтения. Пример применения этой команды приведен ниже: `iptrace -i en0 /tmp/iptrace.log` `# удалить демон iptrace` `kill `ps ax | grep iptrace | awk '{ print $1 }'` `ipreport /tmp/iptrace.log | more`

Импорт файла паролей BSD 4.3:

Можно импортировать файл паролей BSD 4.3 в AIX.

Для импорта файла паролей BSD 4.3 выполните следующие действия:

1. Скопируйте файл паролей BSD 4.3 в файл `/etc/passwd` и введите:
   ```
   pwdck -y ALL
   ```

2. Обновите файл /etc/security/limits, добавив пустые разделы для всех новых пользователей. Эту операцию можно выполнить с помощью команды **usrck**, однако если вместе с файлом /etc/passwd не был импортирован файл /etc/group, то применение команды **usrck**может привести к возникновению ошибок. Дополнительная информация о команде **usrck** приведена в разделе **usrck**.

   Внимание: Если файл /etc/security/limits был изменен, размер стека не должен превышать 65536 байт. В противном случае вызов команды **usrck** может привести к возникновению ошибки. Для исправления ошибки уменьшите размер стека до 65536 байт и повторите вызов команды **usrck**.
3. Выполните команды **grpck** и **usrck** для проверки атрибутов пользователей и групп.

Изменение файла с паролями - информация для системных администраторов BSD 4.3:

Ниже описана процедура изменения записей в файле паролей и управление паролями AIX схожим с BSD 4.3 образом.

В AIX для управления паролями служат команды **lsuser**, **mkuser**, **chuser** и **rmuser**. Все эти функции могут быть также выполнены с помощью SMIT. Однако все эти команды за один вызов позволяют изменить информацию только для одного пользователя.

Дополнительные сведения об этих командах приведены в разделах **lsuser**, **mkuser**, **chuser** и **rmuser**.

Примечание: Для изменения информации о нескольких пользователях с помощью текстового редактора требуется одновременно отредактировать несколько файлов. Это связано с тем, что пароли хранятся в файле /etc/security/passwd, информация о правах доступа - в файле /etc/security/user, а остальные данные о пользователях - в файле /etc/passwd.

AIX не поддерживает команду **vipw**, но поддерживает команду **mkpasswd**. Тем не менее, для работы с паролями в AIX можно применять те же средства, что и в системе BSD 4.3. Для этого выполните следующую процедуру:

1. Поместите файл паролей BSD 4.3 в файл /etc/shadow.
2. Измените права доступа к файлу с помощью следующей команды:

   ```
   chmod 000 /etc/shadow
   ```
3. Поместите в каталог /etc следующий сценарий **vipw**:

```
-----------------------------------------------------
----
#!/bin/bsh
#
# vipw. Использует pwdck. В будущем может использовать usrck
#
PATH=/bin:/usr/bin:/etc:/usr/ucb # Добавьте это строку,
                                 # если ваш редактор
                                 # находится в другом каталоге
if [ -f /etc/ptmp ]; then
                echo "/etc/ptmp существует. Кто-то еще работает с vipw?"
        exit 1
fi
if [ ! -f /`which "$EDITOR" | awk '{ print $1 }'` ] ; then
        EDITOR=vi
fi
cp /etc/shadow /etc/ptmp
if (cmp /etc/shadow /etc/ptmp) ; then
        $EDITOR /etc/ptmp
else
        echo Невозможно скопировать shadow в ptmp
        exit 1
fi
if (egrep "^root:" /etc/ptmp >/dev/null) ; then
        cp /etc/ptmp /etc/shadow ; cp /etc/ptmp /etc/passwd
        chmod 000 /etc/passwd /etc/shadow
        pwdck -y ALL 2>1 >/dev/null # код возврата 114 может измениться
```

```
                rc=$?
        if [ $rc -eq 114 ]; then
                chmod 644 /etc/passwd
                        rm -f /etc/passwd.dir /etc/passwd.pag
                mkpasswd /etc/passwd
                        # обновление /etc/security/limits или ftp
                        # завершится неудачно
                else
                         pwdck -y ALL
                fi
else
        echo Неправильная запись для root в ptmp
fi
rm /etc/ptmp
------------------------------------------------------------
```

4. Если вы планируете применять сценарий **vipw** или команду **mkpasswd**, обратите внимание на то, что SMIT и команды **mkuser**, **chuser** и **rmuser** не используют команду **mkpasswd**. Введите:

   ```
   mkpasswd /etc/passwd
   ```

   При этом будут обновлены файлы `/etc/passwd.dir` и `/etc/passwd.pag`.

   Внимание: Инициализация переменной *IFS* и оператора `trap` повышает риск нарушения защиты, связанный с использованием функции **setuid**. Однако сценарии **vipw** и **passwd** предназначены для открытых сред, в которых основным является требование совместимости. Для реализации среды с более высокой степенью защиты рекомендуется применять только стандартные команды AIX.

5. Поместите в каталог `/usr/ucb` следующий сценарий **passwd**:

```
-----------------------------------------------------
#!/bin/ksh
#
# приводит файл /etc/security/passwd в соответствие
# с измененным файлом /etc/shadow
#
IFS=" "
PATH=/bin
trap "exit 2" 1 2 3 4 5 6 7 8 10 12 13 14 15 16 17 18 21 22 \
        23 24 25 27 28 29 30 31 32 33 34 35 36 60 61 62
if [ -n "$1" ]; then
        USERNAME=$1
else
        USERNAME=$LOGNAME
fi
if [ -f /etc/ptmp ]; then
        echo Файл паролей занят
        exit 1
fi
        trap "rm /etc/ptmp; exit 3" 1 2 3 4 5 6 7 8 10 12 13 \
                14 15 16 17 18 21 22 23 24 25 27 28 29 30 31 \
                32 33 34 35 36 60 61 62
if (cp /etc/security/passwd /etc/ptmp) ; then
        chmod 000 /etc/ptmp else
        rm -f /etc/ptmp exit 1
fi
if ( /bin/passwd $USERNAME ) ; then
        PW=` awk ' BEGIN { RS = "" }
                $1 == user { print $4 } ' user="$USERNAME:" \
/etc/security/passwd `
else
        rm -f /etc/ptmp
        exit 1
fi
rm -f /etc/ptmp
awk -F: '$1 == user { print $1":"pw":"$3 ":"$4":"$5":"$6":"$7 }
        $1 != user { print $0 }' user="$USERNAME" pw="$PW" \
                /etc/shadow > /etc/ptmp
```

```
chmod 000 /etc/ptmp
mv -f /etc/ptmp /etc/shadow
-------------------------------------------------------
```

6. Измените права доступа к сценарию **passwd** с помощью следующей команды:

   ```
   chmod 4711 /usr/ucb/passwd
   ```

7. Убедитесь, что в переменной среды *PATH* каждого пользователя каталог /usr/ucb указан раньше каталога /bin.

Измерение и настройка производительности - информация для системных администраторов BSD 4.3

Ниже описаны атрибуты устройства AIX, а также измерение и настройка производительности.

Со всеми устройствами данной операционной системы связаны определенные атрибуты. Для просмотра атрибутов устройства введите:

```
lsattr -E -l  устройство
```

Атрибуты, для которых указано значение True, могут быть изменены следующей командой:

```
chdev -l устройство -a атрибут=значение
```

Внимание: Неправильное изменение параметров устройства может привести к сбою системы.

По умолчанию максимальное число процессов для одного пользователя равно 40. Для пользователей, одновременно работающих с большим числом окон, это значение может быть недостаточным. Для глобального изменения этого атрибута введите следующую команду:

```
hdev -l sys0 -a maxuproc=100
```

В приведенном выше примере данному параметру присвоено значение 100. Новое значение вступит в силу при следующем запуске системы.

Для просмотра текущего значения этого или другого атрибута системы введите:

```
lsattr -E -l sys0
```

Атрибут **maxmbuf** в настоящее время не поддерживается службами mbuf.

В AIX поддерживаются команды **vmstat** и **iostat**, команда **systat** и средняя загрузка не поддерживается. Дополнительная информация о этих командах приведена в разделах **vmstat** и **iostat**.

Принтеры - информация для системных администраторов BSD 4.3

Операционная система AIX поддерживает две подсистемы печати: 4.3 BSD и System V.

Подсистемы печати System V используют команды, очереди, файлы и средства управления System V Release 4. Ниже приведена информация об управлении подсистемой печати BSD 4.3. С помощью SMIT можно выбрать тип применяемой подсистемы. В каждый момент времени может работать только одна подсистема.

Файлы и программы управления печатью хранятся в каталоге /usr/lpd. Архитектура, конфигурация, принципы управления очередями и демонами в BSD 4.3 и в данной операционной системе различаются. Тем не менее, обе подсистемы используют для удаленной печати протокол **lpd**. Обе подсистемы применяют файл /etc/hosts.lpd, а если он не существует, то файл /etc/host.equiv. В состав подсистемы печати данной операционной системы входит конвертер для работы с подсистемой печати BSD 4.3, поэтому данная операционная система может передавать задания печати в системы BSD 4.3, а также принимать задания печати, отправленные из систем BSD 4.3.

Файл /etc/printcap, применяемый в BSD 4.3, отсутствует в AIX. В этом файле хранится конфигурация программы буферизации и база данных функций принтера. Для правильной настройки принтера пользователи должны изучить формат и ключевые слова файла printcap.

Файл /etc/qconfig в AIX содержит только данные о настройке программы буферизации. Свойства принтеров хранятся в стандартной или настраиваемой базе данных ODM. Функции того или иного принтера можно определить в системе с помощью команды **mkvirprt** (создать виртуальный принтер).

Для того чтобы сделать доступным принтер lp0 на удаленном хосте viking, поместите следующие строки в файл /etc/printcap в системе BSD 4.3:

```
lp0|Удаленный принтер, подключенный к хосту
viking:Z
:lp=:rm=viking:rp=lp:st=/usr/spool/lp0d
```

В AIX для выполнения аналогичной операции необходимо поместить следующие строки в файл /etc/qconfig:

```
lp0:
        device = dlp0
        host = viking
        rq = lp
dlp0:
        backend = /usr/lib/lpd/rembak
```

В данной операционной системе для работы с принтерами поддерживаются следующие команды и библиотечные функции:

Элемент	Описание
cancel(1)	Отменяет запрос к построчному принтеру
chquedev(1)	Изменяет имя очереди принтера или графопостроителя
chvirprt(1)	Изменяет значения атрибутов виртуального принтера
disable(1)	Отключает очередь печати
enable(1)	Включает очередь печати
hplj(1)	Обрабатывает вывод **troff** для печати на принтере HP LaserJetII с картриджем типа K
ibm3812(1)	Обрабатывает вывод **troff** для печати на принтере IBM 3812 Mod 2 Pageprinter
ibm3816(1)	Обрабатывает вывод **troff** для печати на принтере IBM 3816 Pageprinter
ibm5587G(1)	Обрабатывает вывод **troff** для печати на принтере IBM 5587G с картриджем 32x32/24x24
lp(1)	Отправляет запрос построчному принтеру
lpr(1)	Помещает задания печати в очередь
lprm(1)	Удаляет задания печати из очереди построчного принтера
lpstat(1)	Показывает информацию о состоянии построчного принтера
lptest(1)	Формирует тестовый шаблон для построчного принтера
lsallqdev(1)	Показывает список всех настроенных принтеров, связанных с очередью
lsvirprt(1)	Показывает список значений атрибутов виртуального принтера
mkque(1)	Добавляет в систему очередь печати
mkquedev(1)	Добавляет в систему устройство очереди печати
mkvirprt(1)	Создает виртуальный принтер
pac(1)	Подготавливает записи учета ресурсов принтера/графопостроителя
piobe(1)	Администратор печати для базовых программ принтеров
pioburst(1)	Формирует начальную и конечную страницы, печатаемые до и после задания печати
piocmdout(3)	Подпрограмма, выводящая строку атрибутов форматирования
piodigest(1)	Создает и сохраняет список значений атрибутов определения виртуального принтера
pioexit(3)	Функция выхода из программы форматирования
pioformat(1)	Запускает программу форматирования
piofquote(1)	Преобразует некоторые символы управления печатью, предназначенные для принтеров PostScript.
piogetstr(3)	Функция, возвращающая строку атрибутов программы форматирования
piogetvals(3)	Функция инициализации переменных базы данных атрибутов принтера для программы форматирования
piomsgout(3)	Функция отправки сообщений из программы форматирования
pioout(1)	Интерфейс базового драйвера принтера
piopredef(1)	Создает определение потока данных для предопределенного принтера
proff(1)	Форматирует текст для принтеров с индивидуальными потоками данных
qadm(1)	Позволяет выполнять функции администрирования подсистемы буферизации печати
qconfig(4)	Настраивает систему постановки в очередь заданий печати

Элемент	Описание
qstatus(1)	Выводит информацию о состоянии подсистемы буферизации печати
restore(3)	Восстанавливает состояние принтера по умолчанию
rmque(1)	Удаляет из системы очередь печати
rmquedev(1)	Удаляет из системы устройство очереди принтера или графопостроителя
rmvirprt(1)	Удаляет виртуальный принтер
splp(1)	Позволяет просматривать и изменять параметры драйвера принтера
xpr(1)	Форматирует файл дампа окна для вывода на принтер

Информация, связанная с данной:

Обзор принтеров для управления системой

Команды для системного администрирования BSD 4.3

В приведенном ниже списке перечислены команды операционной системы, предназначенные для управления средой AIX.

Элемент	Описание
bosboot(1)	Инициализирует устройство загрузки.
bootlist(1)	Изменяет список загрузочных устройств системы или их порядок в списке.
cfgmgr(1)	Настраивает устройства путем вызова программ из каталога `/etc/methods`.
chcons(1)	Перенаправляет системную консоль на устройство или в файл, действует при следующем перезапуске.
chdev(1)	Изменяет параметры устройства.
chdisp(1)	Изменяет дисплей, применяемый подсистемой низкоуровневого терминала (LFT).
checkcw(1)	Подготавливает текст фиксированной ширины для команды **troff**.
checkeq(1)	Проверяет документы, отформатированные макрокомандой memorandum.
checkmm(1)	Проверяет документы, отформатированные макрокомандой memorandum.
checknr(1)	Проверяет файлы `nroff` и `troff`.
chfont(1)	Изменяет шрифт по умолчанию, выбранный при загрузке.
chfs(1)	Изменяет атрибуты файловой системы.
chgroup(1)	Изменяет атрибуты группы.
chgrpmem(1)	Изменяет администраторов и членов группы.
chhwkbd(1)	Изменяет атрибуты клавиатуры LFT (низкоуровневого терминала), хранящиеся в базе данных Администратора объектных данных ODM.
chitab(1)	Изменяет записи в файле `/etc/inittab`.
chkbd(1)	Изменяет раскладку клавиатуры, применяемую подсистемой низкоуровневого терминала (LFT) по умолчанию.
chkey(1)	Изменяет пользовательский ключ шифрования.
chlang	Изменяет переменную среды *LANG*, заданную в файле `/etc/environment`; изменение вступает в силу при следующем входе в систему.
chlicense(1)	Есть два типа пользовательских лицензий: фиксированная и нефиксированная. Применение фиксированных лицензий всегда разрешено, и число таких лицензий можно изменить с помощью флага **-u**. Применение нефиксированных лицензий можно разрешить или запретить с помощью флага **-f** (значения on и off).
chlv(1)	Изменяет параметры логического тома
chnamsv(1)	Изменяет конфигурацию службы имен TCP/IP на хосте
chprtsv(1)	Изменяет конфигурацию служб печати на клиенте или сервере
chps(1)	Изменяет атрибуты пространства подкачки.
chpv(1)	Изменяет свойства физического тома в группе томов.
chque(1)	Изменяет имя очереди.
chquedev(1)	Изменяет имя очереди принтера или графопостроителя.
chssys(1)	Изменяет определение подсистемы в объектном классе подсистем.
chtcb(1)	Изменяет или запрашивает атрибуты файла в Защищенной компьютерной базе.
chtz	Изменяет информацию о часовом поясе системы.
chuser(1)	Изменяет атрибуты указанного пользователя.
chvfs(1)	Изменяет записи в файле `/etc/vfs`.
chvg(1)	Задает характеристики группы томов.
chvirprt(1)	Изменяет значения атрибутов виртуального принтера.
crfs(1)	Добавляет файловую систему.
crvfs(1)	Создает записи в файле `/etc/vfs`.

Элемент	Описание
exportvg(1)	Экспортирует определение группы томов из набора физических томов.
extendvg(1)	Добавляет физические тома к группе томов.
grpck(1)	Проверяет правильность определения группы.
importvg(1)	Импортирует из набора физических томов определение новой группы томов.
lsallq(1)	Показывает список имен всех настроенных очередей.
lsallqdev(1)	Выводит полный список имен принтеров и графопостроителей указанной очереди.
lsdisp(1)	Показывает список дисплеев, доступных системе в настоящий момент.
lsfont(1)	Показывает список шрифтов для дисплея.
lsfs(1)	Показывает характеристики файловых систем.
lsgroup(1)	Показывает атрибуты групп.
lsitab(1)	Показывает записи файла `/etc/inittab`.
lskbd(1)	Показывает раскладки клавиатуры доступные в подсистеме низкоуровневого терминала (LFT).
lslicense(1)	Показывает число фиксированных лицензий и значение опции применения нефиксированных лицензий.
lslpp(1)	Показывает дополнительные программные продукты.
lsnamsv(1)	Показывает информацию службы имен, хранящуюся в базе данных.
lsprtsv(1)	Показывает информацию службы печати, хранящуюся в базе данных.
lsps	Показывает информацию и атрибуты пространства подкачки.
lsque(1)	Показывает имя раздела очереди.
lsquedev(1)	Показывает имя раздела устройства.
lssrc(1)	Показывает состояние подсистемы, группы подсистем или субсервера.
lsuser(1)	Показывает атрибуты учетных файлов пользователей.
lsvfs(1)	Показывает список записей файла `/etc/vfs`.
mkcatdefs(1)	Препроцессор для исходного файла сообщений.
runcat(1)	Перенаправляет вывод команды **mkcatdefs** на вход команде `gencat`.
mkdev(1)	Добавляет устройство в систему.
mkfont(1)	Добавляет код шрифта, связанный с дисплеем системы.
mkfontdir(1)	Создает файл `fonts.dir` по каталогу файлов шрифтов.
mkgroup(1)	Создает новую группу.
mkitab(1)	Создает записи в файле `/etc/inittab`.
mklv(1)	Создает логический том.
mklvcopy(1)	Создает копии логического тома.
mknamsv(1)	Настраивает для данного клиента службу имен хостов на основе TCP/IP.
mknotify(1)	Добавляет определение способа уведомления к объектному классу уведомлений.
mkprtsv(1)	Настраивает в системе хоста службу печати через TCP/IP.
mkps(1)	Добавляет в систему дополнительное пространство подкачки.
mkque(1)	Добавляет в систему очередь принтера.
mkquedev(1)	Добавляет в систему принтер.
mkserver(1)	Добавляет определение субсервера к объектному классу субсерверов.
mkssys(1)	Добавляет определение подсистемы к объектному классу подсистем.
mksysb	Создает резервную копию смонтированных файловых систем из группы томов rootvg для последующей повторной установки.
mkszfile	Записывает размер смонтированных файловых систем из группы томов rootvg для последующей переустановки.
mktcpip(1)	Задает значения, необходимые для запуска TCP/IP.
mkuser(1)	Создает новый учетный файл пользователя.
mkuser.sys(1)	Настраивает новый учетный файл пользователя.
mkvg(1)	Создает группу томов.
mkvirprt(1)	Создает виртуальный принтер.
odmadd(1)	Добавляет объекты в созданные объектные классы.
odmchange(1)	Изменяет содержимое выбранного объекта из указанного объектного класса.
odmcreate(1)	Создает исходные (`.c`) и заголовочные (`.h`) файлы, необходимые для разработки приложений ODM, и пустые объектные классы.
odmdelete(1)	Удаляет выбранные объекты из указанного объектного класса.
odmdrop(1)	Удаляет объектный класс.
odmget(1)	Записывает объекты из указанного объектного класса в файл `odmadd`.
odmshow(1)	Показывает определение объектного класса.
pwdck(1)	Проверяет правильность локальной информации идентификации.

Элемент	Описание
redefinevg	Переопределяет в базе данных конфигурации устройств набор физических томов для указанной группы томов.
reducevg(1)	Удаляет физические тома из группы томов. Удаление из группы томов всех физических томов приводит к удалению самой группы.
reorgvg(1)	Изменяет расположение физических разделов в группе томов.
restbase(1)	Восстанавливает информацию о настройке из образа загрузки.
rmdel(1)	Удаляет дельту из файла Системы управления исходным кодом (SCCS).
rmdev(1)	Удаляет устройство из системы.
rmf(1)	Удаляет каталоги и сообщения в них.
rmfs(1)	Удаляет файловую систему.
rmgroup(1)	Удаляет группу.
rmitab(1)	Удаляет записи из файла /etc/inittab.
rmlv(1)	Удаляет логические тома из группы томов.
rmlvcopy(1)	Удаляет копии логического тома.
rmm(1)	Удаляет сообщения.
rmnamsv(1)	Удаляет из хоста службу имен протокола TCP/IP.
rmnotify(1)	Удаляет определение способа уведомления из объектного класса уведомлений.
rmprtsv(1)	Удаляет конфигурацию службы печати из сервера или клиента.
rmps(1)	Удаляет пространство подкачки из системы.
rmque(1)	Удаляет очередь печати из системы.
rmquedev(1)	Удаляет очередь принтера или графопостроителя из системы.
rmserver(1)	Удаляет определение субсервера из объектного класса субсерверов.
rmssys(1)	Удаляет определение подсистемы из объектного класса подсистем.
rmuser(1)	Удаляет учетный файл пользователя.
rmvfs(1)	Удаляет записи из файла /etc/vfs
rmvirprt(1)	Удаляет виртуальный принтер.
savebase(1)	Сохраняет базовые данные о настройке устройств ODM на устройстве загрузки.
swapoff(1)	Deactivates one or more paging space.
swapon(1)	Указывает дополнительные устройства для подкачки.
syncvg(1)	Синхронизует различающиеся копии логического тома.
usrck(1)	Проверяет правильность определения пользователя.
varyoffvg(1)	Деактивирует группу томов.
varyonvg(1)	Активирует группу томов.

Понятия, связанные с данным:

“Наиболее существенные различия между 4.3 BSD и AIX” на стр. 323
Ниже описаны основные различия между AIX и системами BSD 4.3.

Cron для системных администратороа BSD 4.3

Демон **cron**, применяемый в данной операционной системе, аналогичен демону **cron** в System V Release 2.

Для запуска демона **cron** необходимо добавить запись в файл /etc/inittab.

Устройства - информация для системных администраторов BSD 4.3

Ниже описаны устройства для системных администраторов BSD 4.3.

В системе BSD 4.3 устройство доступно приложению при выполнении следующих условий:

- Устройство физически установлено и функционирует.
- В ядре есть драйвер устройства.
- В каталоге /dev находятся специальные файлы устройства.

В данной операционной системе устройство доступно приложению при выполнении следующих условий:

- Устройство физически установлено и функционирует.
- В ядре или загруженном расширении ядра есть драйвер устройства.
- В каталоге /dev находятся специальные файлы устройства.

- База данных объектов из каталога /etc/objrepos содержит запись об устройстве, соответствующую его физической конфигурации.

Для работы с базой данных объектов предназначены программы управления устройствами, называемые *методами*. Эти программы расположены в каталоге /etc/methods. Методы вызываются администратором настройки (его можно запустить с помощью команды **cfgmgr**) и другими командами.

Если прикладная программа не может получить доступ к устройству, то неисправно физическое устройство или повреждена база данных в каталоге /etc/objrepos.

Команда **cfgmgr** обрабатывает базу данных конфигурации, хранящуюся в каталоге **/etc/objrepos**. Эта команда вызывается при запуске администратора настройки (**cfgmgr**).

Схема действий Администратора настройки иллюстрируется следующим псевдокодом:

```
/* Главная процедура */
Пока есть правила в базе данных Правил_настройки
        {
        Получить и выполнить следующее правило
        Перехватить вывод (stdout) последней команды
        Проанализировать_вывод(stdout)
        }
/* Процедура анализа вывода */
/* stdout содержит список найденных устройств */
Проанализировать_вывод(stdout)
        {
        Пока есть устройства в списке
                {
                Попытаться найти устройство в базе данных
                если (! /* устройство не */ определено)
                        Получить из базы данных и выполнить метод
                        определения устройства
                если (! настроено)
                        {
                        Получить из базы данных и выполнить метод
                        настройки
                        Проанализировать_вывод(stdout)
                         }
                }
        }
```

UUCP - информация для системных администраторов BSD 4.3

В следующей таблице описаны команды и файлы UUCP.

Элемент	Описание
Dialers(4)	Показывает список модемов, применяемых средствами связи BNU
Maxuuxqts(4)	Ограничивает количество экземпляров демонов BNU **uuxqt**
Permissions(4)	Задает права доступа команд BNU к удаленным системам
Poll(4)	Указывает, когда программа BNU должна опрашивать удаленные системы
Systems(4)	Показывает список удаленных компьютеров, с которыми может устанавливать соединение локальная система
rmail(1)	Позволяет работать с почтой, полученной с помощью BNU
uucheck(1)	Проверяет наличие файлов и каталогов, необходимых для работы BNU
uuclean(1)	Удаляет файлы из буферного каталога BNU
uucleanup(1)	Удаляет выбранные файлы из буферного каталога BNU
uucpadm(1)	Позволяет ввести основную информацию о конфигурации BNU
uudemon.admin(1)	Периодически предоставляет информацию о состоянии передачи файлов BNU
uudemon.cleanu(1)	Очищает файлы протоколов и буферный каталог BNU
uudemon.hour(1)	Запускает процедуру обмена файлами с удаленной системой с помощью программы BNU
uudemon.poll(1)	Опрашивает системы, перечисленные в файле опроса BNU
uulog(1)	Предоставляет информацию о выполняемых в системе операциях передачи файлов BNU
uupoll(1)	Опрашивает удаленные системы BNU

Элемент	Описание
uuq(1)	Позволяет просмотреть очередь заданий BNU и удалить из нее некоторые задания
uusnap(1)	Показывает информацию о состоянии соединений BNU с удаленными системами
uustat(1)	Выдает информацию о состоянии и позволяет выполнить некоторые функции управления BNU

В AIX также поддерживаются команды 4.3 BSD **uuencode** и **uudecode**. Команда HDB **uugetty** не поддерживается. Информация об этих командах содержится в разделах **uuencode** и **uudecode**.

Информация, связанная с данной:

Файл BNU и структура каталогов

Файловые системы - информация для системных администраторов BSD 4.3

Аналогичные команды используются для монтирования и размонтирования файловых систем.

В операционной системе AIX список файловых систем хранится в файле `/etc/filesystem`. В ней предусмотрены команды для монтирования и размонтирования файловых систем.

Файл /etc/filesystems и /etc/fstab:

В системах BSD 4.3 список блоковых устройств и точек монтирования хранится в файле `/etc/fstab`. В системах SVR4 список блоковых устройств и информация о точках монтирования хранится в файле `/etc/vfstab`. В AIX информация о блоковых устройствах и точках монтирования хранится в файле `/etc/filesystems`.

Для обновления файла `/etc/filesystems` применяются команды **crfs**, **chfs** и **rmfs**.

Администраторы систем BSD 4.3 могут применять переменную *check*, заданную в файле `/etc/filesystems`. Переменная *check* может принимать значения `True`, `False` или числовое значение. Например, модно указать в файле `/etc/filesystems` значение `check=2`. Это число задает этап команды **fsck**, на котором будет проверена файловая система. Параметр *check* соответствует пятому полю в записи файла `/etc/fstab`.

В файле `/etc/filesystems` не предусмотрен параметр для задания частоты создания дампа.

Поддержка файловых систем в AIX:

AIX поддерживает различные файловые системы.

Данная операционная система поддерживает дисковые квоты.

Данная операционная система не позволяет монтировать дискеты в качестве файловых систем.

Синтаксис команд **mount** и **umount** для AIX отличается от версий этих команд для 4.3 BSD и SVR4. В приведенной ниже таблице указаны команды, применяемые для монтирования и размонтирования всех файловых систем в различных операционных системах:

Команды **mount** и **unmount**

Функция	Формат в данной операционной системе	Формат в BSD 4.3	Формат в SVR4
монтирование всех файловых систем	**mount all**	**mount -a**	**mountall**
размонтирование всех файловых систем	**umount all**	**umount -a**	**umountall**

Дополнительная информация приведена в разделе “Файловые системы” на стр. 425.

Терминалы - информация для системных администраторов BSD 4.3

Ниже описаны терминалы для системных администраторов BSD 4.3.

Как правило, администраторы систем BSD 4.3 разрешают или запрещают использование портов терминалов путем редактирования файла `/etc/ttys` и отправки сигнала **HUP** программе **init**.

В AIX информация о портах терминалов хранится в базе данных ODM, а запуск терминалов выполняется после того, как программа **init** прочитает содержимое файла `/etc/inittab`. В данной операционной системе настройка портов терминалов выполняется с помощью SMIT.

Порт терминала не связан с каким-либо определенным специальным файлом устройства из каталога `/dev`. В результате системным администраторам, не имеющим опыта работы с данной операционной системой, не всегда понятно, какой именно порт необходимо настроить. В меню программы SMIT первому последовательному порту системного планара (с меткой **s1**) соответствует расположение **00-00-S1**, адаптер **sa0** и порт **s1**. Второму последовательному порту системного планара (с меткой **s2**) соответствует расположение **00-00-S2**, адаптер **sa1** и порт **s2**.

Для включения и отключения порта воспользуйтесь командами **penable** и **pdisable**.

termcap и terminfo:

В данной операционной системе, как и в системе System V, применяются записи `terminfo` файла `/usr/lib/terminfo/?/*`.

Пользователям систем BSD 4.3 могут понадобиться следующие команды:

captoinfo(1)
Преобразует файл `termcap` в файл `terminfo`

tic(1) Компилирует файлы `terminfo`.

В составе данной операционной системы поставляется исходный текст многих записей `terminfo`. Некоторые из них необходимо скомпилировать с помощью команды **tic**. Полное имя исходного файла `termcap`: `/lib/libtermcap/termcap.src`.

Перенаправление ввода и вывода

Операционная система AIX позволяет управлять потоками ввода и вывода данных с помощью специальных символов и команд ввода-вывода.

При управлении вводом вы можете задать источник данных. Например, входные данные можно считывать с клавиатуры (стандартный ввод) или из файла. При управлении выводом вы можете указывать, куда следует направить полученные выходные данные. Например, вывод можно направить на экран (стандартный вывод) или в файл.

Как и любая другая многозадачная операционная система, AIX разработана с учетом возможности взаимодействия процессов.

Понятия, связанные с данным:

“Управление файлами” на стр. 191
В AIX предусмотрены разнообразные средства для работы с файлами. Обычно текстовые файлы создаются с помощью текстовых редакторов.

“Команды просмотра содержимого файла (pg, more, page и cat)” на стр. 195
Команды **pg**, **more** и **page** предназначены для просмотра содержимого файла и позволяют управлять скоростью просмотра файлов.

“Перенаправление ввода и вывода в оболочке Korn (POSIX)” на стр. 230
Перед выполнением команды оболочка Korn выполняет поиск символов перенаправления в командной

строке. Такие символы означают, что необходимо перенаправить ввод или вывод.

Файлы стандартного ввода, стандартного вывода и стандартный вывод сообщений об ошибках

Во время выполнения команды обычно предполагается, что следующие файлы уже открыты: стандартный ввод, стандартный вывод и стандартный вывод сообщений об ошибках (который иногда называется *выводом ошибок* или *диагностическим* выводом).

С каждым из этих файлов связан следующий *дескриптор файла* (число):

Элемент	Описание
Дескриптор файла 0	Стандартный ввод
Дескриптор файла 1	Стандартный вывод
Дескриптор файла 2	Стандартный файл для сообщений об ошибках (диагностических сообщений)

Дочерний процесс обычно наследует эти файлы от родительского процесса. Первоначально эти три файла связываются с рабочей станцией (0 - с клавиатурой, 1 и 2 с дисплеем). Перед выполнением команды из командной оболочки стандартные потоки ввода-вывода можно перенаправить.

Если в команде не задано имя файла, то *стандартный ввод* связывается с клавиатурой (иногда он обозначается *stdin*). После выполнения команды результат ее работы выводится в окно терминала.

Стандартный вывод связывается с окном терминала (иногда он обозначается *stdout*). По умолчанию команды получают входные данные из стандартного ввода, а затем записывают результат работы в стандартный вывод.

Информация об ошибках записывается в стандартный вывод сообщений об ошибках, который иногда называется *stderr*. По умолчанию он связан с окном терминала.

Значения по умолчанию для стандартного ввода и вывода могут быть переопределены. Входные данные могут быть получены из файла, а вывод команды также может быть записан в файл. Это называется *перенаправлением ввода-вывода*.

Вывод команды, который обычно отправляется на дисплейное устройство, можно перенаправить в файл. Это называется *перенаправлением вывода*. Такой прием применяется при выводе больших массивов данных, которые трудно прочитать в окне терминала, или когда несколько файлов объединяются в один большой файл.

Стандартный ввод, обычно связанный с вводом с клавиатуры, можно связать с файлом, хотя это применяется реже, чем перенаправление вывода. Это называется *перенаправлением ввода*. Перенаправление ввода позволяет заранее подготовить файл с входными данными команды.

Перенаправление стандартного вывода

Если в конец команды добавляется запись `>имя_файла`, то входные данные команды считываются из указанного файла. Символ `>` - это *оператор перенаправления вывода*.

Вывод любой команды может быть направлен не в окно терминала, а в файл.

Перенаправление вывода в файл

Вывод процесса можно перенаправить в файл, указав оператор перенаправления и имя файла после текста команды.

Например, для того чтобы перенаправить вывод команды **who** в файл с именем `users`, введите:

```
who > users
```

Примечание: Если файл `users` уже существует, он будет удален и замещен если только не указан опция **noclobber** встроенной команды **set** в оболочке **ksh** (Korn) или **csh** (в оболочке C).

Для просмотра содержимого файла `users` введите:

```
cat users
```

Будет показан список следующего вида:

```
denise    lft/0 13 мая 08:05
marta     pts/1 13 мая 08:10
endrica   pts/2 13 мая 09:33
```

Перенаправление вывода для добавления в файл

Когда в конце команды задается конструкция `>>` *имя-файла*, вывод команды перенаправляется в указанный файл, однако не заменяет его содержимое, а добавляется в конец файла. Символ `>>` - это *оператор перенаправления с добавлением*.

Например, для того чтобы добавить содержимое файла `file2` к файлу `file1`, введите:

```
cat file2 >> file1
```

Примечание: Если файл `file1` не существует, то он будет создан, если во встроенной команде **set** оболочки **ksh** (Korn) или оболочки **csh** (C) не задана опция **noclobber**.

Создание текстового файла путем перенаправления ввода с клавиатуры

Команда **cat** без параметров получает в качестве входных данных ввод с клавиатуры. Вы можете записать полученный ввод в файл.

Введите в новой строке Ctrl-D, чтобы обозначить конец текста.

Введите в командной строке:

```
cat > имя-файла
Это тест.
^D
```

Конкатенация текстовых файлов

Вы можете объединить несколько файлов в один. *Конкатенацией* называется объединение нескольких файлов в один.

В следующем примере будет создан файл `file4`, содержащий данные из файлов `file1`, `file2` и `file3` в указанном порядке.

Примеры:

- Введите в командной строке:

  ```
  cat file1 file2 file3 > file4
  ```

- В следующем примере демонстрируется типичная ошибка, которую допускают при конкатенации файлов:

  ```
  cat file1 file2 file3 > file1
  ```

  Внимание: Можно предположить, что в этом примере команда **cat** объединит содержимое файлов `file1`, `file2` и `file3` в файл `file1`. Однако команда **cat** вначале создает файл вывода, в результате чего содержимое файла `file1` будет удалено, а затем добавляет в него содержимое файлов `file2` и `file3`.

Перенаправление стандартного ввода

Если в конец команды добавляется запись `<` *имя_файла*, то входные данные команды считываются из указанного файла. Символ `<` - это *оператор перенаправления ввода*.

Примечание: Операция перенаправления ввода применима только для тех команд, которые обычно получают в качестве входных данных ввод с клавиатуры.

Например, для того чтобы отправить файл `letter1` в качестве сообщения пользователю `denise` с помощью команды **mail**, введите:

```
mail denise < letter1
```

Игнорирование вывода файла /dev/null

Файл `/dev/null` - специальный. У него есть уникальная особенность - он всегда пустой. Все данные, которые записаны в файл `/dev/null`, уничтожаются. Этим свойством можно воспользоваться, если вы хотите проигнорировать вывод команды или программы.

Например, пусть программа `myprog` получает ввод с клавиатуры и отправляет сообщения, которые необходимо игнорировать. Если программа должна читать входные данные из файла `myscript`, а выходные сообщения нужно игнорировать, то введите:

```
myprog < myscript >/dev/null
```

В данном примере программа `myprog` получает на входе файл `myscript`, а весь стандартный вывод отбрасывается.

Перенаправление стандартного вывода об ошибках и прочего вывода

Кроме стандартного ввода и вывода команды часто работают с другими потоками данных, например, выводом сообщений об ошибках или сообщений о состоянии (диагностических сообщений). Как и стандартный вывод, стандартный поток вывода сообщений об ошибок до перенаправления связан с окном терминала.

Для перенаправления вывода сообщений об ошибках и стандартного вывода применяются дескрипторы файлов. *Дескриптор файла* - это число, связанное с каждым из файлов ввода-вывода, которое обычно используется командой. Дескрипторы можно применять и для перенаправления стандартного ввода и вывода. Со стандартным вводом, выводом и выводом сообщений об ошибках связаны следующие дескрипторы файлов:

Элемент	Описание
0	Стандартный ввод (клавиатура)
1	Стандартный вывод (дисплей)
2	Стандартный вывод сообщений об ошибках (дисплей)

Для перенаправления стандартного вывода сообщений об ошибках введите значение `2` перед оператором перенаправления вывода или оператором перенаправления вывода путем добавления символов перенаправления (`>` или `>>`), а затем укажите имя файла. Например, для добавления стандартного вывода сообщений об ошибках команды **cc**, которая компилирует файл `testfile.c`, в конец файла `ERRORS`, введите:

```
cc testfile.c 2 >> ERRORS
```

Другой вывод можно перенаправить аналогичным образом, указав соответствующий дескриптор файла из диапазона от `0` до `9`. Например, если команда **cmd** записывает вывод в файл с дескриптором `9`, можно перенаправить вывод в файл `savedata` с помощью следующей команды:

```
cmd 9> savedata
```

Если во время выполнения команды данные записываются в несколько потоков вывода, то каждый из них можно перенаправить независимо от остальных. Предположим, что стандартный вывод команды записывается в файл с дескриптором 1, стандартный вывод сообщений об ошибках - в файл с дескриптором 2, а некоторые данные - в файл с дескриптором 9. Вы можете перенаправить каждый из этих потоков вывода в другие файлы с помощью следующей команды:

```
command > standard 2> error 9> data
```

Перенаправление вывода во ввод

Можно перенаправить вывод в документы ввода.

Если команда введена в следующем формате:

команда << конец_ввода

где *конец_ввода* - это любая строка, не содержащая символы подстановки, то оболочка будет рассматривать все последующие строки в качестве стандартного ввода для *команды*, пока не будет считана еще одна строка *конец_ввода* (перед ней могут быть указаны символы табуляции). Строки, введенные между первой и второй последовательностью *конец_ввода*, часто называются *внутренним вводом* или документом *ввода с консоли*. Если за символами перенаправления **<<** указан дефис (-), оболочка удаляет в каждой строке **документа ввода с консоли** начальные символы табуляции перед тем, как передать строку *команда*.

Оболочка создает временный файл, содержащий **документ ввода с консоли**, подставляет в тексте документа необходимые значения вместо переменных и команд, и только затем передает его на вход команде. Вместо шаблонов, указанных в командной строке вместе с командой, подставляются реальные имена файлов. Для того чтобы запретить все подстановки, укажите *конец_ввода* следующим образом:

```
команда << \конец_ввода
```

Документ ввода с консоли удобно применять тогда, когда входные данные занимают небольшой объем. В этом случае удобнее ввести эти данные с помощью процедуры оболочки, а не сохранять в отдельном файле (например, с помощью редактора). Например, введите в командной строке:

```
cat <<- xyz
      Это сообщение будет показано на
      экране без начальных пробелов.
   xyz
```

Понятия, связанные с данным:

“Перенаправление ввода и вывода в оболочке Korn (POSIX)” на стр. 230
Перед выполнением команды оболочка Korn выполняет поиск символов перенаправления в командной строке. Такие символы означают, что необходимо перенаправить ввод или вывод.

Перенаправление вывода с помощью конвейеров и фильтров

Вы можете объединить несколько команд таким образом, чтобы стандартный вывод одной команды применялся в качестве стандартного ввода другой команды. Такой набор команд называется *конвейером*.

Соединение между командами также называется *конвейером*. Конвейеры играют важную роль, поскольку они позволяют объединить несколько простых команд в одну сложную. С помощью конвейера вывод одной команды можно отправить на вход другой команде. Команды соединяются символом конвейера (|).

Фильтром называется процесс, который получает вывод команды, изменяет его, а затем записывает результат в стандартный вывод. Фильтры используются как отдельно, так и в сочетании с конвейером. Чаще всего применяются следующие фильтры:

- sort
- more
- pg

Примеры:

- Команда **ls** выводит на экран список файлов текущего каталога в виде непрерывного потока данных. Если объем данных превышает размер окна, то некоторые данные окажутся скрытыми от просмотра. Этого можно избежать, создав конвейер и направив вывод команды **ls** на вход команды **pg**, которая форматирует вывод для постраничного просмотра. Например, введите:

  ```
  ls | pg
  ```

В этом примере вывод команды **ls** отправляется на вход команды **pg**. Для перехода к следующей странице текста нажмите клавишу Enter.

Команды конвейера выполняются только в одном направлении (слева направо). Каждая команда конвейера выполняется как отдельный процесс, причем все процессы могут выполняться одновременно. Если нет входных данных или канал вывода переполнен, то процесс ожидает.

- Ниже приведен другой пример применения конвейеров вместе с командой **grep**. Команда **grep** выполняет поиск строк в файле по указанному шаблону. Для получения списка файлов, которые были созданы или изменены в июле, введите:

  ```
  ls -l | grep июл
  ```

  В этом примере вывод команды **ls** отправляется на вход команды **grep**.

Просмотр вывода программы и его копирование в файл (команда tee)

Команда **tee** применяется в конвейере. Она читает данные из стандартного ввода, затем записывает вывод программы в стандартный поток вывода и одновременно копирует его в указанный файл или файлы. Команда **tee** позволяет немедленно просмотреть вывод и одновременно сохранить его для использования в будущем.

Например, введите:

```
ps -ef | tee program.ps
```

В результате вывод команды **ps -ef** будет показан на дисплее и одновременно сохранен в файле `program.ps`. Если файл `program.ps` уже существует, то он будет заменен при условии, что не задана опция **noclobber** встроенной команды **set**.

Например, для того чтобы просмотреть вывод команды и сохранить его в существующем файле, введите:

```
ls -l | tee -a program.ls
```

Стандартный вывод команды **ls -l** будет показан на дисплее и добавлен в конец файла `program.ls`.

В окно терминала и файл `program.ls` будет выведена примерно следующая информация:

```
-rw-rw-rw-   1 jones   staff   2301   Sep 19    08:53 161414
-rw-rw-rw-   1 jones   staff   6317   Aug 31    13:17 def.rpt
-rw-rw-rw-   1 jones   staff   5550   Sep 10    14:13 try.doc
```

Сведения о синтаксисе приведены в описании команды **tee** в книге *Справочник по командам, том 5*.

Очистка окна терминала (команда clear)

Команда **clear** очищает окно терминала от сообщений и ввода с клавиатуры.

Введите в командной строке:

```
clear
```

Вся информация на экране будет удалена и появится командная строка.

Отправка сообщений в стандартный вывод

Вывод сообщений на экран с помощью команды **echo**.

Например, для вывода на экран сообщения введите следующую команду:

```
echo Вставьте дискету . . .
```

Будет показано следующее сообщение:

```
Вставьте дискету . . .
```

Ниже приведен пример применения команды **echo** с шаблоном, содержащим символы подстановки:

```
echo Файлы резервной копии: *.bak
```

Появится сообщение `Файлы резервной копии:`, за которым будет указан список файлов с расширением `.bak`.

Добавление к файлу одной строки текста (команда echo)

Можно добавить к файлу одну строку текста, указав в команде **echo** символ перенаправления с добавлением.

Например, введите в командной строке:

```
echo не забудьте создать резервные копии файлов почты в конце недели.>>notes
```

В результате сообщение `не забудьте создать резервные копии файлов почты в конце недели.` будет добавлено в конец файла `notes`.

Копирование содержимого окна в файл (команды capture и script)

Команда **capture** эмулирующая терминал VT100, позволяет скопировать содержимое окна терминала в произвольный файл. Команда **script** позволяет скопировать содержимое окна терминала в указанный файл, не запуская эмуляцию терминала VT100.

Обе команды применяются для печати диалога в сеансе связи.

Например, для копирования содержимого окна терминала VT100 введите в командной строке:

```
capture screen.01
```

Появится приблизительно следующая информация:

```
Команда capture запущена. Данные записаны в файл
screen.01.
Для копирования содержимого окна в файл screen.01 введите ^P.
Начат сеанс
эмуляции терминала vt100.
Для продолжения нажмите любую клавишу.
```

После ввода данных и копирования содержимого окна завершите работу команды **capture**, нажав клавиши Ctrl-D или введя `exit` и нажав Enter. Появится приблизительно следующая информация:

```
Команда capture выполнена. Данные записаны в файл
screen.01.
Сеанс эмуляции терминала vt100 завершен.
```

Для просмотра содержимого полученного файла введите команду **cat**.

Например, для копирования содержимого окна терминала введите в командной строке:

```
script
```

Появится приблизительно следующая информация:

```
Команда script запущена. Данные записаны в файл
typescript.
```

Все содержимое окна терминала копируется в файл `typescript`.

Для завершения команды **script** нажмите клавиши Ctrl-D или введите `exit` и нажмите Enter. Появится приблизительно следующая информация:

```
Выполнение команды script завершено. Данные записаны в
файл typescript.
```

Для просмотра содержимого полученного файла введите команду **cat**.

Дополнительные сведения о синтаксисе приведены в описании команд **capture** и **script** в книге в книге *Справочник по командам*.

Команда для вывода текста на экран большими буквами (команда banner)

Команда **banner** выводит в окно терминала заданный текст в формате ASCII большими буквами.

Каждая строка вывода может содержать не более 10 символов (строчных и прописных букв и цифр).

Например, введите в командной строке:

```
banner GOODBYE!
```

На экране появится слово GOODBYE!, написанное большими буквами.

Обзор команд для перенаправления ввода и вывода

Ниже описаны команды для перенаправления ввода и вывода.

Элемент	Описание
>	“Перенаправление стандартного вывода” на стр. 343
<	“Перенаправление стандартного ввода” на стр. 344
> >	“Перенаправление вывода для добавления в файл” на стр. 344
|	“Перенаправление вывода с помощью конвейеров и фильтров” на стр. 346
banner	Записывает строки символов ASCII в стандартный вывод большими буквами
capture	Позволяет записывать информацию, показанную на экране, в файл
clear	Очищает экран терминала
echo	Записывает символьные строки в стандартный вывод
script	Позволяет копировать ввод и вывод терминала в файл
tee	Показывает стандартный вывод программы и копирует его в файл

Восстановление ядра AIX

Начиная с AIX 6.1, ядро при необходимости можно восстановить без ошибок в выделенных процедурах, избегая непланового простоя системы.

По умолчанию восстановление ядра выключено. Если восстановление ядра включено, то во время восстановления ядра система может на некоторое время приостановиться. Обычно это время не превышает двух секунд. Сразу после восстановления ядра выполняются следующие действия:

- Появится следующее сообщение:

```
-------------------------------------------------------------------------
       Произошла ошибка при восстановлении ядра. Протокол восстановления.
        Все ошибки заносятся в системный протокол ошибок.
-------------------------------------------------------------------------
```

- AIX - в протокол ошибок заносится сообщение. Вы можете отправить данные протокола об ошибках в компанию IBM для обслуживания точно так же, как при отправке данных о полном завершении системы. Ниже приведен пример записи протокола об ошибке восстановления:

```
METKA:     RECOVERY
Дата/Время:       Пятница, 16 февраля/ 14:04:17 CST 2007
Тип:             INFO
Имя ресурса:      RMGR
Описание
Восстановление ядра
Подробные сведения
Базовое имя оперативного дампа
RECOV_20070216200417_0000
Имя функции
w_clear
Имя FRR
w_init_clear_frr
Строка признака
273
EEEE00009627A072
F10001001B18BBC0
w_clear+D0
```

```
wdog0030+288
test_index+4C
Данные протокола восстановления
0001 0000 0000 0000 F000 0000 2FFC AEB0 0000 0111 0000 0000 0000 0000 0021 25BC
8000 0000 0002 9032 EEEE 0000 9627 A072 F100 0100 1B18 BBC0 0000 0000 0000 0000
0000 0001 0000 0000 0006 0057 D2FF 8C00 0001 0148 0500 0000 8000 0000 0002 9032
.....
```

- AIX создает оперативный дамп. Данные оперативного дампа по умолчанию размещаются в каталоге `/var/adm/ras/livedump`, и файл получает имя **RECOV_*системное время_последовательность***, где *системное время* задает время, необходимое для восстановления ядра, а *последовательность* задает количество часов, затраченных на восстановление ядра. Вы можете отправить данные оперативного дампа в компанию IBM для обслуживания, точно так же, как при отправке данных о полном завершении системы. Дополнительная информация об оперативных дампах приведена в разделе оперативные дампы в книге *Основы программирования поддержки устройств и расширений ядра*.

Внимание: После восстановления ядра многие функции могут быть утеряны, но операционная система остается в стабильном состоянии. При необходимости отключите и перезагрузите систему для восстановления потерянных функций.

Присвоение ресурсов памяти и процессоров

AIX поддерживает данные о состоянии восстановления ядра в ходе выполнения основных операций. При включении восстановления ядра, требуются дополнительные команды процессора, чтобы поддержать данные и дополнительная память, чтобы сохранить их. Влияние на использование процессора минимальное. Дополнительный расход памяти может определяться следующим уравнением, где *maxthread* - это максимальное число нитей, запускаемых на системе, а *procnum* - число процессоров:

`требуемый объем памяти = 4 KB x maxthread + 128 KB x procnum`

Как показано в следующем примере, система с 16 процессорами и максимум 1000 нитями дополнительно расходует 6304 KB:

`4 x 1000 + 128 x 16 = 6304 KB`

Включение и отключение восстановления ядра

Восстановление ядра можно включить или отключить из интерфейса пути SMIT.

Для включения или отключения восстановления ядра воспользуйтесь следующим путем SMIT:

Выявление неполадки > Восстановление ядра > Изменить состояние восстановления ядра > Изменить состояние восстановления ядра при следующей загрузке

Текущее состояние восстановления ядра можно показать, используя следующий путь SMIT:

Выявление неполадки > Восстановление ядра > Показать состояние восстановления

Управление устройствами

Для управления различными устройствами в AIX используются команды. Примеры управляемых устройств: Администратор логических томов, файловые системы, накопители и принтеры.

Администратор логических томов

Администратор логических томов (LVM) представляет собой набор команд операционной системы, библиотечных подпрограмм, и других инструментов, предназначенных для управления логической памятью.

Администратор логических томов (LVM) управляет ресурсами дисковой памяти, устанавливая соответствие между *логической* структурой памяти и *физическими* дисками. Для выполнения этих функций LVM использует специальные драйверы устройств, работающие на более высоком уровне, чем стандартные драйверы дисков.

В состав LVM входит драйвер логических томов (LVDD) и библиотека подпрограмм LVM. *Драйвер логических томов* (LVDD) - это драйвер, управляющий всеми операциями ввода-вывода. Он выполняет преобразование логических адресов в физические и отправляет запросы на ввод-вывод драйверам конкретных устройств. *Библиотека функций LVM* содержит функции по управлению логическими и физическими томами; эти функции используются командами системы.

Информация, связанная с данной:

Администратор логических томов — Обзор

Описание драйвера устройств логических томов

Администратор логических томов — Основные сведения

Перед началом использования администратора логических томов необходимо ознакомиться с основными процессами и терминологией.

Процесс включения:

Процесс включения с помощью которого LVM проверяет пригодность группы томов к использованию и сохраняет на носителях этой группы текущей версии данных.

Для включения и выключения группы томов системы служат команды **varyonvg** и **varyoffvg**. Для того чтобы система могла обращаться к группе томов, эту группу необходимо подключить. В процессе подключения группы томов LVM считывает управляющую информацию с физических томов, входящих в состав группы. Эта информация, включающая область дескрипторов (VGDA) и область состояния группы томов (VGSA), хранится на каждом физическом томе группы.

В области VGDA хранится информация о соответствии между физическими и логическими разделами для каждого логического тома, входящего в состав группы томов, а также другая важная информация, например системное время. В области VGSA хранится информация о физических разделах и физических томах, оказавшихся недоступными во время загрузки системы.

Если в процессе подключения какие-либо физические тома оказываются недоступными, то эта команда перечисляет имена всех физических томов, входящих в группу, и показывает информацию о состоянии этих томов. Эта информация поможет определить, следует ли выключить эту группу томов.

Понятия, связанные с данным:

“Высокая готовность на случай сбоя диска” на стр. 403
Основной способ защиты от сбоев дисков заключается в особой настройке логических томов, например, создании зеркальных копий.

Целостность:

С помощью целостность LVM проверяет пригодность группы томов к использованию и сохраняет на носителях этой группы текущей версии данных.

Целостность определяется числом активных областей дескрипторов и состояния группы томов (VGDA/VGSA). Обеспечение целостности означает сохранение структуры данных областей VGDA/VGSA на диске в случае сбоя. На каждом физическом диске в группе томов присутствует по крайней мере одна область VGDA/VGSA. Если группа томов создается на одном диске, то первоначально на этом диске размещаются две области VGDA/VGSA. Если группа томов состоит из двух дисков, то на одном из них также хранятся две области VGDA/VGSA, а на втором - одна. При большем числе дисков в группе томов на каждом из них хранится по одной области VGDA/VGSA.

Потеря целостности наступает в том случае, когда не менее половины дисков (то есть их областей VGDA/VGSA) не могут быть прочитаны Администратором логических томов (LVM). Если в группе томов, состоящей из двух дисков, недоступен диск с одной областью VGDA/VGSA, то целостность сохраняется, так как доступными остаются две или три области VGDA/VGSA. Если же недоступен диск, на котором расположены две области VGDA/VGSA, то целостность считается нарушенной. Чем больше количество дисков в группе томов, тем менее вероятно нарушение целостности при сбое одного диска.

В случае нарушения целостности группа томов автоматически отключается и диски становятся недоступными для администратора логических томов (LVM). Таким образом, операции ввода-вывода, которые могли бы привести к потере данных, предназначенных для записи на диск, не выполняются. Кроме того, при отключении группы томов в протокол ошибок помещается сообщение об аппаратной ошибке и необходимости обслуживания.

В некоторых случаях бывает необходимо продолжить работу с группой томов даже при нарушении целостности. В этом случае проверку целостности для данной группы томов можно отключить. Такая группа томов называется *группой томов без контроля целостности*. Чаще всего такие группы томов применяются с зеркальным копированием. Если диск стал недоступным, но на одном из доступных дисков есть копия данного логического тома, то данные можно восстановить. Однако, возможна ситуация, когда все копии данных расположены на недоступном диске. В этом случае, независимо от того, применяется или нет зеркальное копирование, данные будут недоступны, хотя группа томов подключена.

Понятия, связанные с данным:

“Отключение контроля целостности для группы томов” на стр. 354
Вы можете отключить для группы томов контроль целостности, обеспечив постоянный доступ к данным даже в случае нарушения целостности.

Зеркальные пулы:

Зеркальные пулы позволяют разделить физические тома группы томов на отдельные пулы.

Зеркальный пул состоит из одного или нескольких физических томов. Каждый физический том одновременно может принадлежать только одному зеркальному пулу. При создании логического тома каждую создаваемую копию логического тома можно присвоить зеркальному пулу. Присвоенные зеркальному пулу копии логических томов могут выделять разделы только из физических томов этого зеркального пула. Таким образом появляется возможность ограничить для копии логического тома доступ к дискам. Без зеркальных пулов единственным способом задать ограничение, какой физический том использовать для размещения при создании или расширении логического тома, является использование файла размещения. Поэтому использование зеркальных пулов значительно упрощает этот процесс. Зеркальные пулы можно создать командой **extendvg** или командой **chpv**.

При создании нового зеркального пула необходимо указать имя зеркального пула. Имена зеркальных пулов должны отвечать следующим правилам:

- Должно содержать только буквы, цифры и символы **_** (подчеркивание), **-** (знак минуса) и **.** (точка).
- Должно быть не более 15 символов.
- Имя должно быть уникальным в группе томов.

Если в группе томов были использованы зеркальные пулы, эту группу томов больше нельзя импортировать в AIX версии, не поддерживающей зеркальные пулы. К ним относятся любые версии AIX ниже 6.1.1.0. Кроме того, для того чтобы можно было использовать зеркальные пулы с расширенным LVM с параллельным режимом, все узлы в кластере должны поддерживать зеркальные пулы.

Степень строгости зеркального пула

Степень строгости зеркального пула служит для принудительного ужесточения ограничений на использование зеркального пула. Степень строгости зеркального пула может принимать одно из следующих значений:

off Если для степени строгости зеркального пула задано значение `off`, никаких ограничений на использование зеркального пула не налагается. Это значение по умолчанию.

on Если для степени строгости зеркального пула задано значение `on`, каждая копия логического тома, созданная в группе томов, должна быть присвоена зеркальному пулу.

super Если для степени строгости зеркального пула задано значение `super`, применяются следующие ограничения:

- Локальный и удаленный физические тома не могут принадлежать одному и тому же зеркальному пулу.

  Примечание: Дополнительная информация о локальных и удаленных физических томах приведена в документации по HACMP/XD GLVM.
- В одной группе томов может быть не более трех зеркальных пулов.
- Каждый зеркальный пул должен содержать по крайней мере одну копию каждого логического тома в группе томов.

Администратор географических логических томов:

Администратор географических логических томов (GLVM) позволяет обслуживать зеркальную копию ваших данных в географически удаленном расположении.

GLVM может защитить ваше предприятие от сбоя путем создания зеркальной копии критических данных на удаленном сайте восстановления при сбое. Если в результате пожара или потопа происходит потеря данных на рабочем сайте, можно воспользоваться резервной копией данных на удаленном сайте восстановления при сбое.

Создание зеркальной копии данных выполняется по стандартной сети TCP/IP. Рабочий сайт и сайт восстановления при сбое не обязательно должны быть в одной физической сети. Между этими двумя сайтами допускаются маршрутизаторы и шлюзы. Вместо использования слишком длинных кабелей дисков для удаленного доступа к дискам используются сеть TCP/IP и драйвер устройств RPV (удаленный физический том).

Пользователь настраивает географически удаленные диски как удаленные физические тома, а затем объединяет эти удаленные физические тома с локальными физическими томами, создавая группы географически удаленных копий томов. Эти группы томов управляются Администратором логических томов (LVM) и работают аналогично обычным группам томов. GLVM поддерживает как синхронное, так и асинхронное создание удаленных зеркальных копий.

Группа томов без контроля целостности:

Администратор логических томов (LVM) автоматически отключает группу томов, если количество областей дескрипторов групп томов (VGDA) или областей состояния групп томов (VGSA). Тем не менее, существует опция, позволяющая сохранять группу томов активной до тех пор, пока она содержит хотя бы одну доступную область VGDA/VGSA. Если указана эта опция, создается *группа томов без контроля целостности.*

Перед тем как администратор логических томов разрешит подключение групп томов без контроля целостности, необходимо предоставить ему доступ ко всем дискам в этих группах. Это обеспечит актуальность данных VGDA и VGSA.

Создать группу томов без контроля целостности может потребоваться в системах, в которых у каждого логического тома есть хотя бы по две копии. При обнаружении сбоя группа томов остается активной, пока есть хотя бы один активный диск.

Примечание: Как пользовательские группы томов, так и группа томов **rootvg**, могут работать без контроля целостности, но процедуры настройки пользовательских групп томов без контроля целостности и

восстановления после сбоев отличаются от соответствующих процедур для группы томов **rootvg**. Для группы томов каждого типа необходимо применять процедуры, разработанные именно для данного типа группы томов.

Даже при использовании групп томов без контроля целостности *возможно* потерять целостность и увидеть следующее сообщение в выводе команды **errpt**:

```
ПОТЕРЯ ЦЕЛОСТНОСТИ, LVM ЗАКРЫВАЕТ ГРУППУ ТОМОВ.
```

Это сообщение появляется когда все физические тома находятся в состоянии `missing` и LVM автоматически выключает группу томов.

Сообщение содержит фразу `ПОТЕРЯ ЦЕЛОСТНОСТИ` так как отключение проверки целостности в группе том ов снижаются требования целостности к 1. С помощью команды **lsvg *vgname*** можно отобразить значение целостности в поле `ЦЕЛОСТНОСТЬ:`. Если физические тома находятся в состоянии `missing`, нарушаются даже эти минимальные требования к целостности, в результате чего появляется сообщение о целостности и автоматически отключается группа томов.

Информация, связанная с данной:

Администратор логических томов от А до Я: Введение и общие сведения

Отключение контроля целостности для группы томов:

Вы можете отключить для группы томов контроль целостности, обеспечив постоянный доступ к данным даже в случае нарушения целостности.

Данная процедура часто применяется в системах со следующей конфигурацией:

- Группа томов из двух дисков с зеркальным копированием логических томов
- Группа томов из трех дисков, в которой для каждого логического тома поддерживается одна или две зеркальные копии

В обеих описанных ситуациях группа томов с отключенным контролем целостности может продолжать работу даже в случае сбоя диска, пока в ней есть есть хотя бы одна работающая копия логического тома.

Для того чтобы обеспечить возможность восстановления группы томов с отключенным контролем целостности, должны быть выполнены следующие условия:

- Если применяются файловые системы JFS или JFS2, то необходимо создать зеркальную копию логического тома протокола JFS.
- Поместите зеркальные копии на разные диски. Для проверки физического расположения (PV1, PV2 и PV3) каждого из логических разделов введите следующую команду. (В случае размещения копий на различных дисках в столбцах PV1, PV2 и PV3 после hdisk будут указаны разные номера.)

  ```
  lslv -m логический-том
  ```

  Если копии логического тома находятся на одном диске, то в случае потери доступа к этому диску том также станет недоступным для пользователей, независимо от наличия или отсутствия контроля целостности в группе томов.

Отключить контроль целостности можно как для группы rootvg, так и для пользовательских групп томов, но способы настройки и восстановления после сбоев в этих двух случаях отличаются.

Для подключения пользовательской группы томов без контроля целостности должны быть доступны все физические тома группы. Поскольку группы томов без контроля целостности должны работать до тех пор, пока не будет утрачен доступ к последнему диску, во время подключения все диски должны быть доступны.

Внимание: Не включайте систему, если в группе томов rootvg отсутствуют какие-либо диски. Это можно делать только в том случае, когда восстановление диска невозможно. Администратор логических томов (LVM) всегда применяет флаг **-f** для принудительного подключения группы томов rootvg без контроля

целостности, хотя эта операция и связана с определенным риском. LVM должен принудительно подключить группу томов, поскольку запустить операционную систему при неактивной группе rootvg невозможно. Другими словами, LVM попытается подключить rootvg, даже если работает только один из дисков этой группы.

Понятия, связанные с данным:

“Высокая готовность на случай сбоя диска” на стр. 403
Основной способ защиты от сбоев дисков заключается в особой настройке логических томов, например, создании зеркальных копий.

“Высокая готовность в случае сбоя адаптера или блока питания” на стр. 404
Для защиты от отказов адаптера и источника питания выполните одно или несколько из перечисленных ниже действий, в зависимости от требуемого уровня надежности.

“Реализация стратегии группы томов” на стр. 415
После выбора стратегии создания группы томов проанализируйте текущую конфигурацию с помощью команды **lspv**.

“Целостность” на стр. 351
С помощью целостность LVM проверяет пригодность группы томов к использованию и сохраняет на носителях этой группы текущей версии данных.

Настройка Администратора логических томов

Администратор логических томов (LVM) устанавливается вместе с базовой операционной системой и требует дополнительной настройки. Однако прежде, чем LVM сможет работать с дисками, их необходимо настроить и определить в качестве физических томов.

Задачи, связанные с данной:

“Создание логического тома с прямым доступом для приложения” на стр. 396
Логический том с прямым доступом - это область физического и логического дискового пространства, которой приложение управляет непосредственно, минуя операционную систему или файловую систему. Это может быть, например, раздел или база данных.

Команды по обслуживанию объектов LVM:

В следующей таблице перечислены простейшие задачи по обслуживанию объектов LVM, к которым относятся физические и логические тома, группы томов и файловые системы.

Таблица 62. Задачи управления логическими томами и памятью

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Подключение группы томов	**smit varyonvg**	
Добавление в существующую группу томов жесткого диска без данных	**smit extendvg**	
Добавление в новую группу томов жесткого диска без данных	**smit mkvg**	
Добавление логического тома Прим. 1	**smit mklv**	
Создание группы томов	**smit mkvg**	
Создать и подключить группу томов	**smit mkvg**	
Изменение логического тома для применения функции распределения данных	**smit chlv1**	
Изменение имени группы томов Прим. 2	1. **smit varyoffvg** 2. **smit exportvg** 3. **smit importvg** 4. **smit mountfs**	1. **varyoffvg** *старое-имя-группы-томов* 2. **exportvg** *старое-имя-группы-томов* 3. **importvg** *новое-имя-группы-томов* 4. **mount all**
Изменение группы томов для применения функции автоматического подключения	**smit chvg**	
Изменение или настройка стратегий логического тома	**smit chlv1**	

Таблица 62. Задачи управления логическими томами и памятью (продолжение)

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Скопировать содержимое логического тома в новый логический том[Прим. 3]	**smit cplv**	
Копирование логического тома на существующий логический том того же размера[**Внимание. 1**]	**smit cplv**	
Копирование логического тома на существующий логический том меньшего размера[**Внимание. 1**] [Прим. 3]	Не применяйте SMIT[**Внимание. 2**]	1. Создайте логический том. Например: **mklv -y hdisk***N* **vg00 4** 2. Создайте на только что созданном логическом томе новую файловую систему. Например: **crfs -v jfs -d hdisk***N* **-m /doc -A yes** 3. Смонтируйте файловую систему. Например: **mount /doc** 4. Создайте каталог для точки монтирования. Например: **mkdir /doc/options** 5. Скопируйте файлы с исходного логического тома на целевой. Например: **cp -R /usr/adam/oldoptions/* \ /doc/options**
Копирование логического тома на существующий логический том большего размера[**Внимание. 1**]	**smit cplv**	
Отключить группу томов	**smit varyoffvg**	
Включение стратегии планирования изменений и проверки при записи	**smit chlv1**	
Увеличить максимальный размер логического тома	**smit chlv1**	
Увеличить размер логического тома	**smit extendlv**	
Просмотреть список логических томов, упорядоченный по группам томов	**smit lslv2**	
Просмотреть список физических томов	**smit lspv2**	
Просмотреть список групп томов	**smit lsvg2**	
Просмотр сведений о состоянии, логических томах и разделах физического тома	**smit lspv**	
Просмотр списка содержимого группы томов	**smit lsvg1**	
Просмотр сведений о состоянии и отображении логического тома	**smit lslv**	
Создание зеркальной копии логического тома с функцией распределения данных или без нее	**smit mklvcopy**	
Выключение питания съемного диска	**smit offdsk**	Доступна только при поддержке функции оперативного удаления
Включить питание съемного диска	**smit ondsk**	Доступна только при поддержке функции оперативного удаления
Удаление зеркальной копии из группы томов	**smit unmirrorvg**	
Удаление группы томов	**smit reducevg2**	
Реорганизация группы томов	**smit reorgvg**	
Удаление диска из конфигурации и отключение его питания	**smit rmvdsk1** или **smit rmvdsk** затем **smit opendoor**	

Внимание:

1. При копировании данных на существующий логический том все данные на этом томе будут заменены без запроса подтверждения.
2. Для копирования данных с логического тома большего размера на логический том меньшего размера не следует применять SMIT или команду **cplv**. Использование этих команд приведет к повреждению файловой системы, так как часть данных (включая главный блок) не будет перенесена на логический том меньшего размера.

Примечание:

1. После создания логический том находится в закрытом состоянии, поскольку этот логический том еще не используется в структурах LVM. Это состояние не изменится до тех пор, пока на логическом томе не будет смонтирована файловая система или пока он не будет открыт для прямого ввода-вывода.
2. Группу томов **rootvg** нельзя переименовать, импортировать и экспортировать.
3. Для дублирования логического тома требуется наличие достаточного количества памяти с прямым доступом.

Задачи, связанные с данной:

“Создание логического тома с прямым доступом для приложения” на стр. 396
Логический том с прямым доступом - это область физического и логического дискового пространства, которой приложение управляет непосредственно, минуя операционную систему или файловую систему. Это может быть, например, раздел или база данных.

Добавление дисков без выключения системы:

Приведенная ниже процедура описывает включение и настройку диска с помощью функции оперативного удаления, позволяющей добавлять диски без выключения системы.

Вы можете добавить диск для расширения дискового пространства или для устранения сбоя диска. Эта возможность реализована только в некоторых моделях систем.

1. Установите диск в свободную ячейку корпуса. Дополнительная информация о процедуре установки приведена в соответствующем разделе руководства по обслуживанию системы.
2. Включите новый диск с помощью следующей команды:
   ```
   smit ondsk
   ```

Теперь диск добавлен в систему, но работать с ним еще нельзя. Дальнейшие действия зависят от того, есть ли на новом диске данные.

- Если новый диск не содержит данных, то добавьте его в качестве физического тома в группу томов, воспользовавшись одним из следующих способов:
  - Для добавления диска в существующую группу томов введите следующую команду:
    ```
    smit extendvg
    ```
  - Для добавления диска в новую группу томов введите следующую команду:
    ```
    smit mkvg
    ```
- Если на диске есть данные, то импортируйте их.

Понятия, связанные с данным:

“Реализация стратегии группы томов” на стр. 415
После выбора стратегии создания группы томов проанализируйте текущую конфигурацию с помощью команды **lspv**.

Задачи, связанные с данной:

“Импорт и экспорт группы томов” на стр. 361
В следующей таблице описано применение процедур экспорта и импорта для перемещения пользовательской группы томов из одной системы в другую. (Группу томов rootvg нельзя экспортировать или импортировать.)

“Удаление диска без данных” на стр. 398
С помощью данной процедуры можно удалить диск, содержащий данные, без выключения системы.
“Удаление диска без данных” на стр. 400
Ниже описана процедура удаления диска без данных или с ненужными данными.

Изменение имени логического тома:

Описанная ниже процедура позволяет переименовать логический том с сохранением всех хранящихся на нем данных.

В следующих примерах имя логического тома изменяется с `lv00` на `lv33`.

1. Размонтируйте все файловые системы, связанные с логическим томом:
   ```
   unmount /файловая-система
   ```
   где *файловая-система* - полное имя файловой системы.

   Примечание:
   a. Если файловая система, которую вы пытаетесь размонтировать, используется, то команда **unmount** выдаст сообщение об ошибке. Команда **unmount** выполняется лишь в том случае, если в файловой системе нет открытых файлов и на ее устройстве нет текущих пользовательских каталогов.
   b. Другое название команды **unmount** - **umount**. Эти имена равноценны.
2. Переименуйте логический том с помощью следующей команды:
   ```
   chlv -n новое-имя старое-имя
   ```
   где флаг **-n** позволяет задать новое имя логического тома (*новое-имя*), а *старое-имя* - это изменяемое имя логического тома. Пример:
   ```
   chlv -n lv33 lv00
   ```

   Примечание: При переименовании протокола JFS или JFS2 система попросит вас выполнить команду **chfs** для всех файловых систем, использующих переименованное устройство протокола.
3. Смонтируйте файловые системы, размонтированные на шаге 1:
   ```
   mount /test1
   ```

Теперь логический том переименован и готов к работе.

Копирование логического тома в другой физический том:

В зависимости от конкретной ситуации, можно воспользоваться разными способами копирования логического тома на другой физический том с сохранением целостности файловой системы.

Есть несколько способов копирования логического тома или JFS в другой физический том. Выберите способ, наиболее подходящий в вашей ситуации.

Копирование логического тома:

Простейший способ заключается в применении команды **cplv** для копирования исходного логического тома и создании нового логического тома на целевом физическом томе.

1. Прекратите работу с логическим томом. При необходимости размонтируйте файловую систему и завершите работу всех приложений, обращающихся к логическому тому.
2. Выберите физический том с достаточным объемом свободного пространства для размещения всех данных исходного логического тома.

   Внимание: при копировании большего логического тома на меньший возможно повреждение файловой системы из-за потери части данных (включая главный блок).
3. С помощью следующей команды скопируйте исходный логический том (в данном примере это том с именем **lv00**) и создайте новый:

Примечание: Если при создании нового логического тома группа томов включена в параллельном режиме, то команда **cplv** выдаст сообщение об ошибке.

```
cplv lv00
```

4. Если необходимо, смонтируйте файловые системы и перезапустите приложения, работающие с логическим томом.

Копия логического тома готова к работе.

Копирование логического тома с поддержкой доступности исходного логического тома:

Если необходимо обеспечить постоянный доступ к исходному логическому тому, то для копирования содержимого этого тома вы можете воспользоваться командой **splitlvcopy**.

1. С помощью следующей команды SMIT создайте зеркальную копию логического тома:

   ```
   smit mklvcopy
   ```

2. Прекратите работу с логическим томом. При необходимости размонтируйте файловую систему, а также завершите работу всех приложений, обращающихся к логическому тому, либо переведите эти приложения в устойчивое состояние.

   Внимание: На следующем шаге применяется команда **splitlvcopy**. Обязательно закрывайте логические тома перед их разделением и размонтируйте все содержащиеся на них файловые системы перед запуском этой команды. Разделение открытого логического тома может привести к повреждению файловых систем и рассогласованию между исходным логическим томом и копией из-за одновременного обращения к логическому тому нескольких процессов.

3. Войдя в систему под именем пользователя с правами доступа root, скопируйте исходный логический том (`старый-том`) в новый (`новый-том`) с помощью следующей команды:

   ```
   splitlvcopy -y новый-том старый-том
   ```

   Флаг **-y** задает имя нового логического тома. Если `старый-том` не содержит управляющего блока, то команда **splitlvcopy** успешно завершит работу, но выдаст сообщение о том, что `новый-том` создан без управляющего блока логического тома.

4. Если необходимо, смонтируйте файловые системы и перезапустите приложения, работающие с логическим томом.

Копия логического тома готова к работе.

Копирование логического тома в другой физический том:

Для копирования логического тома с прямым доступом на другой физический том выполните следующие действия:

1. С помощью следующей команды создайте зеркальную копию логического тома на новом физическом томе группы томов:

   ```
   mklvcopy логический-том 2 новый-физ.-том
   ```

2. С помощью следующей команды синхронизируйте разделы в зеркальной копии:

   ```
   syncvg -l логический-том
   ```

3. С помощью следующей команды удалите с физического тома копию логического тома:

   ```
   rmlvcopy логический-том 1 старый-физический-том
   ```

Копия логического тома с прямым доступом готова к работе.

Создание протокола файловой системы на выделенном диске для заданной пользователем группы томов:

`Протокол файловой системы` JFS или JFS2 - это отформатированный список записей о транзакциях файловой системы. Протокол обеспечивает целостность файловой системы (не гарантируя целостности данных) в случае прерывания работы системы во время выполнения транзакции.

Во время установки системы на hd8 создается отдельный диск с корневой группой томов (rootvg). Следующая процедура позволяет создать протокол JFS для других групп томов на отдельном диске. При создании протокола JFS2 в процедуру необходимо внести следующие изменения:

- Тип устройства протокола: `jfs2log`.
- Для устройства протокола JFS2 команда **logform** требует указания опции `-V jfs2`.
- В командах **crfs** необходимо указать `jfs2` вместо `jfs`.

Примечание: Файл протокола JFS2 не обязательно должен находиться на отдельном диске как файловая система. Единственное требование - устройства ведения протоколов должны располагаться в одной и той же группе томов, что и файловая система. При выполнении этой процедуры протокол JFS2 должен находиться на отдельном диске в целях повышения производительности.

Создание файла протокола файловой системы для пользовательских групп томов в некоторых ситуациях может повысить производительность, например, если имеется сервер NFS и требуется чтобы транзакции для этого сервера обрабатывались без конкуренции с другими процессами.

Вы можете воспользоваться следующей процедурой, создающей группу томов (fsvg1) с двумя физическими томами (hdisk1 и hdisk2). Файловая система находится на hdisk2 (файловая система объемом 256 МБ, смонтированная в `/u/myfs`), а протокол - на hdisk1. По умолчанию размер протокола JFS равен 4 Мб. На один и тот же физический том с протоколом можно поместить редко используемые программы, например, `/blv`, - это не приведет к снижению производительности.

Для создания протокола JFS для пользовательской группы томов с использованием SMIT и интерфейса командной строки выполните следующие действия:

1. С помощью команды SMIT добавьте новую группу томов (в этом примере - fsvg1):
   ```
   smit mkvg
   ```
2. С помощью команды SMIT добавьте в эту группу томов новый логический том:
   ```
   smit mklv
   ```
3. В меню **Добавить логический том** заполните следующие поля. Пример:
   ```
   Имя логического тома                  fsvg1log

   Число логических разделов             1

   Имя физического тома                  hdisk1

   Тип логического тома                  jfslog

   Размещение на физическом томе         в центре
   ```
4. После заполнения полей нажмите Enter для применения внесенных изменений и выхода из SMIT.
5. Введите следующую команду в командной строке:
   ```
   /usr/sbin/logform /dev/fsvg1log
   ```
6. После появления следующего запроса, введите ответ **д** и нажмите Enter:
   ```
   Уничтожить /dev/fsvg1log ?
   ```
   Несмотря на грозное предупреждение, ничего на самом деле не уничтожается. При вводе ответа `д` система форматирует логический том для протокола JFS, обеспечивая возможность записи информации о транзакциях файловых систем.
7. С помощью команды SMIT добавьте еще один логический том:
   ```
   smit mklv
   ```
8. Введите то же имя группы томов, что и на шаге 2 (в данном примере `fsvg1`). В меню Логические тома укажите заполните следующие поля. Не забудьте указать для этого логического тома физический том, отличный от заданного на шаге 3. Пример:

```
Имя логического тома                fslv1

Число логических разделов           64

Имя физического тома                hdisk2

Тип логического тома                jfs
```

После заполнения полей нажмите Enter для применения внесенных изменений и выхода из SMIT.

9. С помощью следующих команд создайте на новом логическом томе файловую систему, укажите для нее протокол и смонтируйте файловую систему:

   ```
   crfs -v jfs -d группа-томов -m файловая-система -a logname=расположение-протокола

   mount файловая-система
   ```

   Здесь *логический-том* - это имя логического тома, созданного на шаге 2; *файловая-система* - имя монтируемой на этом логическом томе файловой системы; а *расположение-протокола* - имя логического тома, созданного на шаге 2. Пример:

   ```
   crfs -v jfs -d fslv1 -m /u/myfs -a logname=/dev/fsvg1log
   mount /u/myfs
   ```

10. Для проверки правильности настройки файловой системы и протокола введите следующую команду (подставив имя своей группы томов):

    ```
    lsvg -l fsvg1
    ```

    Будет выведена информация о созданных вами логических томах с указанием типов файловых систем, например:

    ```
    Имя лог. тома       Тип     ...
    /dev/fsvg1log       jfslog  ...
    fslv1               jfs     ...
    ```

Вы создали группу томов, содержащую по крайней мере два логических тома, расположенных на разных физических томах, причем на одном из этих логических томов хранится протокол файловой системы.

Примечание: Для обеспечения избыточности можно реализовать зеркальную защиту для устройства ведения протоколов JFS2 на уровне логического тома. Однако зеркальная защита не является общепринятой практикой и реализовывать ее не обязательно.

Импорт и экспорт группы томов:

В следующей таблице описано применение процедур экспорта и импорта для перемещения пользовательской группы томов из одной системы в другую. (Группу томов rootvg нельзя экспортировать или импортировать.)

При экспорте группы томов ее определение удаляется из конфигурации системы. При импорте группа томов регистрируется в новой системе.

Кроме того, с помощью процедуры импорта можно повторно зарегистрировать группу томов в той же системе, из которой она когда-то была экспортирована. Сочетание процедур импорта и экспорта позволяет также добавить диск с данными в отдельную группу томов.

Внимание: Команда **importvg** изменяет имя импортируемого логического тома, если в новой системе уже существует том с тем же исходным именем. При переименовании логического тома команда **importvg** направляет в стандартный поток вывода для ошибок сообщение об ошибке. При отсутствии конфликтов команда **importvg** также создайте точки монтирования и добавляет записи в файл `/etc/filesystems`.

Задачи импорта и экспорта группы томов		
Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Импорт группы томов	**smit importvg**	
Экспорт группы томов	1. Размонтируйте файловые системы в логических томах группы томов: **smit umntdsk** 2. Отключите группу томов: **smit varyoffvg** 3. Экспортируйте группу томов: **smit exportvg**	

Внимание: Группу томов, включающую пространство подкачки, нельзя экспортировать, если пространство подкачки активно. Перед экспортом такой группы томов отключите пространство подкачки:

```
chps -a n paging_space имя
```

Затем перезагрузите систему для отключения пространства подкачки.

Задачи, связанные с данной:

“Добавление дисков без выключения системы” на стр. 357
Приведенная ниже процедура описывает включение и настройку диска с помощью функции оперативного удаления, позволяющей добавлять диски без выключения системы.

“Удаление диска без данных” на стр. 398
С помощью данной процедуры можно удалить диск, содержащий данные, без выключения системы.

Перенос содержимого физического тома:
В этом разделе описана процедура перемещения физических разделов, соответствующих одному или нескольким логическим томам, с одного физического тома на другой или на несколько физических томов в группе томов. Эту процедуру можно также использовать для перемещения данных с неисправного диска перед его ремонтом или заменой. Эта процедура может применяться для физических томов, входящих как в группу томов rootvg, так и в пользовательские группы.

Внимание: При переносе загрузочного логического тома на другой физический том во избежание зависания системы необходимо очистить загрузочную запись на исходном томе. При выполнении команды **bosboot** необходимо также выполнить команду **chpv -c**, описание которой приведено в шаге 4 следующей процедуры.

1. Для переноса данных на новый диск выполните следующие действия. Либо перейдите к шагу 2.
   a. С помощью следующей команды убедитесь, что диск распознается системой и доступен:

      ```
      lsdev -Cc disk
      ```

      Вывод команды будет выглядеть примерно следующим образом:

      ```
      hdisk0 Доступно  10-60-00-8,0  16 Bit  LVD  Диск SCSI
      hdisk1 Доступно  10-60-00-9,0  16 Bit  LVD  Диск SCSI
      hdisk2 Доступно  10-60-00-11,0 16 Bit  LVD  Диск SCSI
      ```

   b. Если диск доступен, то выполните следующую команду, чтобы убедиться, что он не входит в состав другой группы томов:

      ```
      lspv
      ```

      Вывод команды будет выглядеть примерно следующим образом:

      ```
      hdisk0          0004234583aa7879       rootvg          активен
      hdisk1          00042345e05603c1       нет             активен
      hdisk2          00083772caa7896e       imagesvg        активен
      ```

      В приведенном ниже примере в качестве целевого можно использовать диск `hdisk1`.

      Если новый диск не указан в списке или недоступен, необходимо настроить диск или память логического тома.

c. Добавьте новый диск в группу томов:

```
extendvg группа-томов диск
```

где *группа-томов* - это имя группы томов, а *диск* - имя нового диска. В приведенном примере в качестве *диска* необходимо указать `hdisk1`.

2. Исходный и целевой тома должны входить в одну группу томов. Чтобы определить, относятся ли оба физических тома к одной группе, введите следующую команду:

```
lsvg -p группа-томов
```

где *группа-томов* - это имя группы томов. Вывод для группы томов rootvg будет выглядеть примерно следующим образом:

```
rootvg:
ИМЯ_PV     СОСТ. PV   ЧИСЛО PP    СВОБ. PP   РАСПРЕД. СВОБ. PP
hdisk0         активен        542         85          00..00..00..26..59
hdisk1         активен        542         306         00..00..00..00..06
```

Обратите внимание на число свободных физических разделов (PP).

3. Проверьте, достаточно ли на целевом диске свободного места для перемещения данных:
   a. Определите число физических разделов на исходном диске:

```
lspv исходный-диск | grep "USED PPs"
```

Здесь *исходный-диск* - это имя исходного диска, например, hdisk0. Вывод команды будет выглядеть примерно следующим образом:

```
ЗАНЯТЫЕ PP:      159 (636 МБ)
```

В данном примере для успешного перемещения данных на целевом диске должно быть не менее 159 свободных физических разделов (СВОБОДНЫХ PP).

   b. Сравните число свободных PP на исходном диске с числом свободных PP на целевом диске или дисках (шаг 2). Если число свободных PP больше, чем число занятых, значит на диске достаточно места для перемещения данных.

4. Этот шаг предназначен только для перемещения данных с диска, входящего в группу томов rootvg. Если данные перемещаются с диска в пользовательской группе томов, перейдите к шагу 5.

   Проверьте, находится ли загрузочный логический том (**hd5**) на исходном диске с помощью команды

```
lspv -l номер-исходного-диска | grep hd5
```

Если выходные данные отсутствуют, значит, загрузочный логический том находится другом диске. Перейдите к шагу 5.

Если вывод команды выглядит следующим образом:

```
hd5            2   2   02..00..00..00..00   /blv
```

то введите следующую команду:

```
migratepv -l hd5 исходный-диск целевой-диск
```

После этого будет показано предупреждение о необходимости выполнить команду **bosboot** для целевого диска. Кроме того, для очистки загрузочной записи на исходном диске необходимо запустить команду **mkboot -c** . Введите следующие команды:

```
bosboot -a -d /dev/целевой-диск
bootlist -m normal целевой-диск
mkboot -c -d /dev/исходный-диск
```

5. С помощью следующей команды SMIT переместите данные:

```
smit migratepv
```

6. Просмотрите список физических томов и выберите исходный том.

7. Перейдите в поле **Целевые** физические тома. Если вы сохраните значение по умолчанию, то для перемещения будут применяться все физические тома группы. В противном случае выберите один или несколько дисков, на которые достаточно свободного пространства для размещения перемещаемых физических разделов (см. шаг 4).
8. При необходимости в поле Переместить данные только этого **Логического тома** укажите логический том. В этом случае будут обработаны только те физические разделы, которые соответствуют заданному логическому тому и расположены на выбранном исходном физическом томе.
9. Для перемещения физических разделов нажмите Enter.

Теперь данные находятся на новом (целевом) диске. Исходный диск при этом остается в группе томов. Если диск работает надежно, то вы можете использовать его в качестве оперативного резерва. Если же диск работает со сбоями, то рекомендуется выполнить следующие действия:

1. Для удаления исходного диска из группы томов введите следующую команду:
   ```
   reducevg группа-томов исходный-диск
   ```
2. Для физического удаления исходного диска из системы введите следующую команду:
   ```
   rmdev -l исходный-диск -d
   ```

Понятия, связанные с данным:

“Структура логических разделах” на стр. 385
Логические тома - это организованные области данных на физических томах.

Задачи, связанные с данной:

“Настройка диска”
Настроить новый диск можно различными способами.

“Устранение неполадок дисков” на стр. 373
Приведенные ниже сведения касаются диагностики и устранения неполадок жестких дисков.

Настройка диска:

Настроить новый диск можно различными способами.

Вы можете настроить новый диск любым из следующих способов.

- Завершить работу системы и отключить ее можно с помощью первого способа. При наличии такой возможности следует всегда выключать питание системы при подключении к ней физических дисков.
- Если вы не можете завершить работу системы и вам известна вся информация о новом диске, включая подкласс, тип, имя родительского адаптера и подключение, то перейдите к разделу Способ 2.
- Если вы не можете завершить работу системы и вам известно только расположение диска, то перейдите к разделу Способ 3.

Несмотря на то, что после настройки с диском уже можно работать, администратор логических томов требует идентификации диска в качестве физического тома.

Способ 1

Если вы можете завершить работу системы и отключить ее питание перед подключением диска, то воспользуйтесь следующим способом:

1. Физически подключите новый диск к системе, а затем включите питание диска и системы в соответствии с документацией.
2. Во время загрузки системы разрешите администратору настройки (**cfgmgr**) автоматически настроить диск.
3. После загрузки войдите в систему под именем пользователя с правами доступа root и введите команду **lspv**. Будет показана примерно следующая информация:
   ```
   hdisk1    нет                 нет
   ```
   или:

```
hdisk1   00005264d21adb2e     нет
```

В первом поле указывается присвоенное системой имя диска. Во втором поле указан ИД физического тома (PVID), если он задан. Если новый диск не показан в выводе команды **lspv**, то обратитесь к книге *Установка и миграция*.

Теперь система может работать с диском, но для его применения администратором логических томов необходимо присвоить ИД физического тома. Если у нового диска нет ИД физического тома, то перейдите к разделу “Преобразование доступного диска в физический том” на стр. 366.

Способ 2

Данная процедура может применяться, если вы не можете завершить работу системы, но вам известна следующая информация о диске:

- Способ подключения диска (подкласс)
- Тип диска (тип)
- Место подключения диска к системе (родительское имя)
- Логический адрес диска (для подключения)

Выполните следующие действия:

1. Физически подключите новый диск к системе, а затем включите питание диска и системы в соответствии с документацией.
2. Настройте диск в качестве физического тома с помощью команды **mkdev** со следующими флагами:

   ```
   mkdev -c disk -s scsi -t 2200mb -p scsi3 \
   -w 6,0 -a pv=yes
   ```

   Ниже приведен пример добавления диска емкостью 2,2 ГБ с ИД SCSI 6 и номером логического устройства 0 на шине SCSI scsi3. Флаг **-c** определяет класс устройства. Флаг **-s** определяет подкласс. Флаг **-t** определяет тип устройства. Флаг **-p** задает присваиваемое имя родительского устройства. Флаг **-w** обозначает расположение диска по ИД SCSI и номеру логического устройства. Флаг **-a** задает пару атрибут-значение `pv=yes`, указывающую, что диск необходимо сделать физическим томом и записать на него загрузочную запись с уникальным идентификатором физического тома (если такая запись еще не существует).

Теперь диск определен и как доступное устройство и как физический том. Вы можете ввести команду **lspv** и просмотреть запись для нового диска. Если новый диск не показан в выводе команды **lspv**, то обратитесь к книге *Установка и миграция*.

Способ 3

Данная процедура может применяться, если вы не можете завершить работу системы и вам известно только расположение диска.

1. Физически подключите новый диск к системе, а затем включите питание диска и системы в соответствии с документацией.
2. Для проверки уже настроенных в системе физических дисков введите команду **lspv**. Дополнительная информация о команде **lspv** приведена в разделе Команда lspv. Вывод команды будет выглядеть примерно следующим образом:

   ```
   hdisk0       000005265ac63976    rootvg
   ```

3. Введите команду **cfgmgr** для запуска администратора настройки. Администратор настройки автоматически обнаруживает и настраивает все подключенные к системе новые устройства, включая новый диск. Дополнительная информация о команде **cfgmgr** приведена в разделе **cfgmgr**.
4. Для того чтобы проверить настройку нового диска, введите команду **lspv** еще раз. Ее вывод будет выглядеть примерно следующим образом:

```
hdisk1    нет                 нет
```

или

```
hdisk1   00005264d21adb2e     нет
```

В первом поле указывается присвоенное системой имя диска. Во втором поле указан ИД физического тома (PVID), если он задан. Если новый диск не показан в выводе команды **lspv**, то обратитесь к книге *Установка и миграция*.

Теперь система может работать с диском, но для его применения администратором логических томов необходимо присвоить ИД физического тома. Если у нового диска нет ИД физического тома, то перейдите к разделу “Преобразование доступного диска в физический том”.

Задачи, связанные с данной:

“Перенос содержимого физического тома” на стр. 362

Преобразование доступного диска в физический том:

Для того чтобы диск можно было включать в группы томов и работать с ним с помощью LVM, его необходимо настроить как физический том.

Для настройки физического тома выполните следующие действия:

1. Убедитесь, что диск, известный операционной системе, доступен и не используется самой операционной системой или другими приложениями. Введите в командной строке команду **lspv**. Вывод команды будет выглядеть примерно следующим образом:

   ```
   hdisk1    нет                 нет
   ```

   Проверьте:

   - Если в выводе команды не указано имя нового диска, то перейдите к разделу “Настройка диска” на стр. 364.
   - Если во втором поле вывода указан системный идентификатор физического тома (PVID) (например, `00005264d21adb2e`), значит диск уже настроен в качестве физического тома и выполнять эту процедуру не нужно.
   - Если в третьем поле вывода указано имя группы томов (например, `rootvg`), значит диск в настоящее время используется и эту процедуру для него выполнять нельзя.

   Если для диска не указан идентификатор физического тома и он не используется, то перейдите к следующему шагу.
2. Для преобразования доступного диска в физический том воспользуйтесь командой **chdev**. Пример:

   ```
   chdev -l hdisk3 -a pv=yes
   ```

   Флаг **-l** задает имя устройства диска. Флаг **-a** задает пару атрибут-значение `pv=yes`, указывающую, что диск необходимо сделать физическим томом и записать на него загрузочную запись с уникальным идентификатором физического тома (если такая запись еще не существует).

Теперь диск определен в качестве физического тома. Вы можете ввести команду **lspv** и просмотреть запись для нового диска.

Изменение PVID и VGID для rootvg:

Можно изменить идентификатор физического тома (PVID) и идентификатор группы томов (VGID) для группы томов rootvg на этапе загрузки системы.

Для того чтобы изменить PVID и VGID для rootvg, установите для атрибута *sys0 dev ghostdev* значение 2 и перезагрузите систему. Атрибут *sys0 device ghostdev* - это побитовый флаг.

- Для того чтобы установить атрибут *sys0 device ghostdev* для изменения PVID и VGID для группы томов rootvg, введите следующую команду:

  ```
  chdev -l sys0 -a ghostdev=2
  ```

Примечание: Значение2 для атрибута *sys0 device ghostdev* не установлено, после того как команда **ipl_varyon** изменяет PVID и VGID всех дисков в rootvg. Если команда **chdev** для изменения PVID любого диска rootvg не выполнена, то команда **ipl_varyon** отправляет предупреждающее сообщение и продолжает изменять rootvg. Если команда **chdev** для изменения PVID любого диска в rootvg не выполнена, и вы хотите изменить PVID и VGID при следующей перезагрузке, снова установите для атрибута *sys0 device ghostdev* значение 2.

- Для того чтобы вывести значение атрибута *ghostdev*, введите следующую команду:

  ```
  lsattr -E -l sys0 -a ghostdev
  ```

Замена физического тома в группе томов с зеркальной защитой:

Описанная ниже процедура служит для замены вышедшего из строя физического тома (PV) в группе томов с зеркальной защитой. Команда **replacepv** обеспечивает способ замены вышедшего из строя физического тома в большинстве конфигураций. Также предлагается альтернативная процедура для конфигураций, в которых невозможно использовать команду **replacepv** .

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

Предварительные требования

- Все логические тома, использующие вышедший из строя физический том, имеют допустимые копии в других доступных физических томах (возможным исключением является выделенный логический том дампа).

Замена вышедшего из строя физического тома с помощью команды replacepv

Предварительные требования

Если перечисленные ниже предварительные требования не могут быть выполнены, обратитесь к описанию альтернативной процедуры.

- Группа томов, содержащая вышедший из строя физический том, не является rootvg.
- Заменяющий физический том можно добавить в группу томов, содержащую вышедший из строя физический том (это может оказаться невозможным в зависимости от размера физического тома и параметров группы томов, содержащей физический том, например `Максимум PP на PV`).
- Заменяющий физический том должен предоставлять возможность настройки его в системе одновременно с вышедшим из строя физическим томом.
- Имя заменяющего физического тома может отличаться от имени вышедшего из строя физического тома.
- Размер заменяющего физического тома должен быть не меньше размера вышедшего из строя физического тома.
- Группа томов, содержащая вышедший из строя физический том, не должна быть группой томов мгновенных копий или содержать группу томов мгновенных копий.

Выполните следующие действия (предполагается, что вышедший из строя физический том - это `hdisk2`, а заменяющий его физический том - это `hdisk10`:

1. Если заменяющий физический том еще не установлен в системе, выполите необходимые для его установки действия. Для применения диспетчера настройки выполните следующую команду:

   ```
   cfgmgr
   ```

   С помощью команды **lspv** определите имя, присвоенное физическому тому. В данном примере предполагается, что физический том имеет имя `hdisk10`.
2. Для замены вышедшего из строя физического тома на том, заданный в первом шаге, выполните следующую команду:

   ```
   replacepv hdisk2 hdisk10
   ```

При выполнении команды hdisk2 будет заменен на hdisk10, а hdisk2 более не будет присвоен группе томов.

3. Для отмены определения вышедшего из строя физического тома выполните следующую команду:
   ```
   rmdev -dl hdisk2
   ```
4. Физически удалить вышедший из строя диск из системы.
5. Проверьте, что процедура выполнена успешно, запустив следующую команду:
   - Для проверки зеркальных копий всех логических томов с помощью нового физического тома выполните следующую команду:
     ```
     lslv группа-логических-томов
     ```
     Проверьте атрибут COPIES для всех логических томов, связанных с вышедшим из строя логическим томом, чтобы убедиться в наличии необходимого числа копий. Если число копий логического тома ниже меньше нужного числа, воспользуйтесь командой **mklvcopy** для создания дополнительных копий.
   - Для того, чтобы проверить, что все разделы логического тома синхронизированы, а устаревшие разделы отсутствуют, выполните следующую команду:
     ```
     lspv hdisk10
     ```
     Проверьте атрибут STALE PARTITIONS замененного физического тома, чтобы убедиться, что он имеет нулевое значение. При наличии устаревших разделов воспользуйтесь командой **syncvg** для синхронизации разделов.

В шаге 5 завершается процедура замены вышедшего из строя физического тома.

Замена вышедшего из строя физического тома в случае, когда конфигурация не допускает применения команды replacepv

Предположим, что вышедший из строя физический том hdisk0 и заменяющая его зеркальная копия hdisk1 являются частью группы томов *yourvg*.

1. Для удаления зеркальных копий из вышедшего из строя физического тома выполните следующую команду:
   ```
   unmirrorvg yourvg hdisk0
   ```
2. Если сбой физического тома произошел в rootvg, удалить hdisk0 из списка загрузки с помощью следующей команды:

   Примечание: Если ваша конфигурация использует загрузочные устройства, отличные от hdisk0 и hdisk1, добавьте их в синтаксис команды.
   ```
   bootlist -om normal hdisk1
   ```
   Этот шаг требует чтобы hdisk1 оставался загрузочным устройством в rootvg. Выполнив этот шаг убедитесь, что hdisk0 не упоминается в выводе.
3. Если сбой физического тома произошел в rootvg, воссоздайте выделенные устройства дампа из вышедшего из строя физического тома.

   При наличии выделенного устройства дампа, находящегося в вышедшем из строя физическом томе, для создания нового логического тома в имеющемся физическом томе можно использовать команду **mklv**. С помощью команды **sysdumpdev** можно задать новый логический том в качестве основного устройства дампа.
4. Для отмены определения вышедшего из строя физического тома выполните следующую команду:

   Примечание: Удаление записи диска также приведет к удалению жесткой ссылки /dev/ipldevice если вышедший из строя физический том использовался для загрузки системы.
   ```
   reducevg yourvg hdisk0
   rmdev -dl hdisk0
   ```

5. Если вышедший из строя физический том является последним использовавшимся загрузочным устройством, воссоздайте жесткую ссылку /dev/ipldevice, удаленную в шаге 4, выполнив следующую команду:

   ```
   ln /dev/rhdisk1 /dev/ipldevice
   ```

   Обратите внимание на букву r, предшествующую имени физического тома.

   Для проверки воссоздания жесткой ссылки /dev/ipldevice выполните следующую команду:

   ```
   ls /dev/ipldevice
   ```

6. Замена вышедшего из строя диска.
7. Для того чтобы задать новый физический том выполните следующую команду:

   ```
   cfgmgr
   ```

   Команда **cfgmgr** присваивает имя физического тома заменяющему вышедший из строя физическому тому. Присвоенное имя физического тома, скорее всего, будет идентично имени, ранее присвоенному вышедшему из строя тому. В данном примере предположим, что используемому для замены физическому тому присвоено имя hdisk0.
8. Для добавления нового физического тома в группу томов выполните следующую команду:

   ```
   extendvg yourvg hdisk0
   ```

   Вы можете увидеть следующее сообщение об ошибке:

   ```
   0516-050 Недостаточно дескрипторов в группе томов.
   Попытайтесь добавить меньший физический том или укажите другую группу томов.
   ```

   При появлении этого сообщения об ошибке и невозможности добавить физический том в группу томов вы можете создать зеркальные копии логических томов в другом физическом томе, уже существующем в группе томов или добавить физический том меньшего размера. Если ни тот, ни другой вариант не возможен, попробуйте обойти это ограничение, обновив группу томов до группы томов типа Большая или Масштабируемая с помощью команды **chvg**.
9. Зеркальное копирование группы томов.

   Примечание: При наличии всех следующих условий использовать команду **mirrorvg** невозможно:

   - Целевая система является логическим разделом (LPAR).
   - Копия загрузочного логического раздела (по умолчанию это hd5) расположена в вышедшем из строя физическом томе.
   - Адаптер заменяющего физического тома был динамически настроен в LPAR после последней холодной загрузки.

   При наличии всех перечисленных выше условий воспользуйтесь командой **mklvcopy** для воссоздания зеркальных копий для каждого логического тома следующим образом:

   a. Создайте копии загрузочного логического тома, чтобы гарантировать, что он выделен смежной серии физических разделов.
   b. Создайте копии оставшихся логических томов и синхронизируйте копии с помощью команды **syncvg**.
   c. Сделайте диск загрузочным, завершив LPAR и активировав его вместо перезагрузки с помощью команд shutdown или reboot. Это завершение системы необязательно выполнять немедленно, но оно требуется для загрузки системы с нового физического тома.

   В ином случае создайте новые копии логических томов в группе томов с помощью нового физического тома посредством следующей команды:

   Примечание: Команда **mirrorvg** по умолчанию отключает проверку целостности. Для rootvg вам понадобиться использовать опцию **-m** чтобы обеспечить, что копии новых логических томов привязаны к hdisk0 точно так же, как и работающий диск.

   ```
   mirrorvg yourvg hdisk0
   ```

10. Если в вашей конфигурации имеются копии одного логического тома, может потребоваться воссоздать эти копии с помощью следующей команды:

    ```
    mklvcopy -k
    ```

11. Если сбой физического тома произошел в rootvg, выполните инициализацию загрузочной записи с помощью следующей команды:

    ```
    bosboot -a
    ```

12. Если сбой физического тома произошел в rootvg, обновите список загрузки с помощью следующей команды:

    Примечание: Если ваша конфигурация использует загрузочные устройства, отличные от `hdisk0` и `hdisk1`, добавьте их в команду.

    ```
    bootlist -om normal hdisk0 hdisk1
    ```

13. Убедитесь в успешном выполнении процедуры.
    - Для проверки зеркальных копий всех логических томов в новом физическом томе выполните следующую команду:

      ```
      lslv группа-логических-томов
      ```

      Проверьте атрибут `COPIES` для всех логических томов, связанных с вышедшим из строя логическим томом, чтобы убедиться в наличии необходимого числа копий. Если число копий логического тома ниже меньше нужного числа, воспользуйтесь командой **mklvcopy** для создания дополнительных копий.
    - Для проверки синхронизации всех разделов логического тома убедитесь в отсутствии устаревших разделов с помощью следующей команды:

      ```
      lspv hdisk0
      ```

      Проверьте атрибут `STALE PARTITIONS` замененного физического тома, чтобы убедиться, что он имеет нулевое значение. При наличии устаревших разделов воспользуйтесь командой **syncvg** для синхронизации разделов.

    Если сбой физического тома произошел в rootvg, выполните следующие действия для проверки прочих аспектов этой процедуры:
    - Для проверки списка загрузки выполните следующую команду:

      ```
      bootlist -om normal
      ```

    - Для проверки устройства дампа выполните следующую команду:

      ```
      sysdumpdev -l
      ```

    - Для проверки списка загрузочных физических томов выполните следующую команду:

      ```
      ipl_varyon -i
      ```

    - Для проверки `/dev/ipl_device` выполните следующую команду:

      ```
      ls -i /dev/rhdisk1 /dev/ipldevice
      ```

      Убедитесь, что в выводе команды **ls** указано одинаковое число i-узла для обеих записей.

    Этот шаг завершает выполнение процедуры.

Информация, связанная с данной:

Администратор логических томов от А до Я: Введение и общие сведения

Уведомление администратора об отсутствующем физическом томе:

Несмотря на то, что при недоступности тома AIX заносит в протокол сообщение об ошибке, существуют ситуации, в которых такая ошибка может остаться незамеченной.

Например, если физический том входит в состав группы томов с зеркальной защитой, то пользователи не заметят ошибки до тех пор, пока будет доступна зеркальная копия данных. В таком случае средства автоматического уведомления могут отправить администратору сообщение об ошибке до того, как эта ошибка сможет оказать влияние на работу конечных пользователей.

Ниже описана настройке средств автоматического уведомления об отсутствии физического тома. Изменив эту процедуру, вы можете отслеживать другие важные ошибки.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Войдя в систему от имени пользователя root, создайте резервную копию файла ODM `/etc/objrepos/errnotify`. Резервной копии можно присвоить любое имя. В следующем примере при создании резервной копии к имени файла `errnotify` добавляется текущая дата:

   ```
   cd /etc/objrepos
   cp errnotify errnotifyдата
   ```

2. Создайте в текстовом редакторе файл `/tmp/pvmiss.add` со следующим содержимым:

   ```
   errnotify:
    en_pid = 0
    en_name = "LVM_SA_PVMISS"
    en_persistenceflg = 1
    en_label = "LVM_SA_PVMISS"
    en_crcid = 0
    en_type = "UNKN"
    en_alertflg = ""
    en_resource = "LVDD"
    en_rtype = "NONE"
    en_rclass = "NONE"
    en_method = "/usr/lib/ras/pvmiss.notify $1 $2 $3 $4 $5 $6 $7 $8 $9"
   ```

   После выполнения всех инструкций, описанных в этом разделе, демон уведомления об ошибках автоматически заменит переменные $1 - $9 в этом сценарии на подробную информацию из записи протокола ошибок.

3. Создайте в текстовом редакторе файл `/usr/lib/ras/pvmiss.notify` со следующим содержимым:

   ```
   #!/bin/ksh
   exec 3>/dev/console
   print -u3 "?"
   print -u3 - "-------------------------------------------"
   print -u3 "ALERT!  ALERT!  ALERT!  ALERT!  ALERT!  ALERT!"
   print -u3 ""
   print -u3 "Desc: PHYSICAL VOLUME IS MISSING. SEE ERRPT."
   print -u3 ""
   print -u3 "Error label: $9"
   print -u3 "Sequence number: $1"
   print -u3 "Error ID: $2"
   print -u3 "Error class: $3"
   print -u3 "Error type: $4"
   print -u3 "Resource name: $6"
   print -u3 "Resource type: $7"
   print -u3 "Resource class: $8"
   print -u3 - "-------------------------------------------"
   print -u3 "?"
   mail - "PHSYICAL VOLUME DECLARED MISSING" root <<-EOF
   -------------------------------------------
   ALERT!  ALERT!  ALERT!  ALERT!  ALERT!  ALERT!
   Desc: PHYSICAL VOLUME IS MISSING. SEE ERRPT.
   Error label: $9
   Sequence number: $1
   Error ID: $2
   Error class: $3
   Error type: $4
   Resource name: $6
   ```

```
Resource type: $7
Resource class: $8
--------------------------------------------
EOF
```

4. Сохраните файл и закройте текстовый редактор.
5. Задайте права доступа к созданному вами файлу. Например:
   ```
   chmod 755 /usr/lib/ras/pvmiss.notify
   ```
6. Для добавления в ODM определения LVM_SA_PVMISS, созданного на шаге 2, введите следующую команду:
   ```
   odmadd /tmp/pvmiss.add
   ```

Теперь система будет выполнять сценарий `/usr/lib/ras/pvmiss.notify` при каждом обнаружении ошибки LVM_SA_PVMISS. Этот сценарий выводит сообщение на консоль и отправляет пользователю root почтовое сообщение.

Понятия, связанные с данным:

“Структура логических разделах” на стр. 385
Логические тома - это организованные области данных на физических томах.

Информация, связанная с данной:

Команда odmadd

Выделение зеркального диска из группы томов:

Поддержка моментальных копий помогает защитить согласованность групп томов с зеркальной копией в случае возможного сбоя диска.

Функция моментальной копии позволяет выделить из группы с зеркальной защитой один или несколько дисков для их применения в качестве надежной (с точки зрения метаданных LVM) резервной копии группы томов, соответствующей заданному моменту времени, а также, при необходимости, снова интегрировать выделенные диски в группу томов. Следующая процедура позволяет выделить диск из группы томов с зеркальной защитой, а затем снова вернуть выделенный диск в исходную группу томов. Для дальнейшего повышения надежности моментальной копии необходимо размонтировать все файловые системы, а приложения, использующие логические тома с прямым доступом, необходимо перевести в известное состояние (состояние, начиная с которого приложение может выполнить восстановление в случае применения резервной копии).

При выполнении любого из следующих условий диск нельзя выделить из группы томов:

- Какой-либо диск уже отсутствует.
- В группе томов выделенного тома останется последний не устаревший раздел.
- В группе томов есть устаревшие разделы. Эту ситуацию можно обойти с помощью флага **-f** команды **splitvg**.

Функцию моментальной копии (особенно команду **splitvg**) нельзя применять в расширенном или обычном параллельном режиме. Группу томов выделенного диска нельзя перевести в параллельный или расширенный параллельный режим; кроме того, существует ряд ограничений на изменения, допустимые в исходной группе томов и в группе томов выделенного тома. Дополнительные сведения приведены в описании команды **chvg**.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Убедитесь, что для всех компонентов группы томов существуют зеркальные копии, причем они должны храниться на диске, не содержащем зеркальных копий, относящихся к другим наборам.
2. Для включения поддержки моментальной копии выделите из исходной группы томов (`исходная-группа`) отдельный диск или набор дисков:
   ```
   splitvg исходная-группа
   ```

Теперь у вас есть надежная резервная копия исходной группы томов, соответствующая определенному моменту времени. Однако, следует иметь в виду, что вы не можете изменять размещение группы томов с выделенным томом.

3. Активируйте выделенный диск и включите его в исходную группу томов с помощью следующей команды:

   ```
   joinvg origVG
   ```

   Теперь выделенный диск снова включен в исходную группу томов.

Понятия, связанные с данным:

“Структура логических разделах” на стр. 385
Логические тома - это организованные области данных на физических томах.

Информация, связанная с данной:

Команда chvg

Команда recreatevg

Команда splitvg

Администратор логических томов от А до Я: Введение и общие сведения

Устранение неполадок LVM

Вы можете столкнуться с несколькими распространенными типами неполадок в LVM и устранить их.

Устранение неполадок дисков:

Приведенные ниже сведения касаются диагностики и устранения неполадок жестких дисков.

Если вы подозреваете механическое повреждение, то выполните процедуру диагностики диска:

1. Войдите в систему как пользователь root и введите следующую команду:

   ```
   smit diag
   ```

2. Выберите **Диагностика текущей оболочки** для запуска инструмента диагностики AIX.
3. После прочтения инструкций по работе со средствами диагностики нажмите Enter.
4. Выберите **Процедуры диагностики**.
5. Выберите **Проверка системы**.
6. Пролистайте список и найдите в нем проверяемый диск.
7. Нажмите кнопку **Выполнить**.

В зависимости от полученных результатов, определите состояние диска:

- Если вы обнаружили неисправный диск, то прежде всего необходимо сохранить его данные. Миграция - лучший способ перенести данные с неисправного диска. Если выполнить миграцию не удалось, то воспользуйтесь описанными ниже процедурами восстановления данных с логических томов.
- Если диск неисправен, но вы можете устранить неполадку, не форматируя его, то данные потеряны не будут.
- Если диск необходимо отформатировать или заменить, то перед заменой следует по возможности создать резервную копию и удалить диск из группы томов и из системной конфигурации. Некоторые данные файловых систем без зеркальных копий могут быть потеряны.

Понятия, связанные с данным:

“Пространство на жестком диске” на стр. 374
Если объема свободной дисковой памяти системы стало недостаточно, вы можете увеличить ее несколькими способами. Можно автоматически найти и удалить ненужные файлы, запретить пользователям доступ к некоторым каталогам или добавить (путем монтирования) пространство из другого диска.

“Восстановление жесткого диска без повторного форматирования” на стр. 375
Если вы устранили неполадку диска и вернули его в систему, не выполняя форматирования, то система может автоматически активизировать этот диск и восстановить синхронизацию устаревших физических разделов диска во время загрузки. *Устаревшие физические разделы* содержат данные, которые система не может использовать.

Задачи, связанные с данной:

“Перенос содержимого физического тома” на стр. 362

“Восстановление с помощью переформатированного или замененного жесткого диска” на стр. 376
Вы можете восстановить данные с неисправного диска, когда необходимо повторно отформатировать или заменить неисправный диск.

“Восстановление неисправного диска при поддержании доступности системы” на стр. 380
Можно выполнить восстановление неисправного диска с помощью оперативной замены компонентов.

Пространство на жестком диске:

Если объема свободной дисковой памяти системы стало недостаточно, вы можете увеличить ее несколькими способами. Можно автоматически найти и удалить ненужные файлы, запретить пользователям доступ к некоторым каталогам или добавить (путем монтирования) пространство из другого диска.

Для выполнения этих задач необходимы права доступа root, права доступа группы system или административные права доступа.

Задачи, связанные с данной:

“Устранение неполадок дисков” на стр. 373
Приведенные ниже сведения касаются диагностики и устранения неполадок жестких дисков.

Команды для автоматической очистки файловой системы:

Для очистки файловой системы путем удаления ненужных файлов вызовите команду **skulker**.

Введите в командной строке следующее:

```
skulker -p
```

Команда **skulker** служит для периодического удаления устаревших или ненужных файлов из файловой системы. Такими файлами считаются файлы из каталога `/tmp`, созданные до указанной даты, файлы `a.out` и `ed.hup`, а также файлы дампа. Дополнительная информация о команде **skulker** приведена в разделе **skulker**.

Команда **skulker** обычно выполняется ежедневно как часть процедуры учета, запускаемой командой **cron** в часы наименьшей загрузки системы.

Понятия, связанные с данным:

“Переполнение диска” на стр. 449
Переполнение диска происходит при записи слишком большого числа файлов в выделенное пространство. Это может произойти в том случае, когда запущенные процессы создают много ненужных файлов.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Ограничение доступа пользователей к определенным каталогам:

Вы можете освободить пространство на диске и поддерживать его свободным с помощью ограничения доступа к каталогам и отслеживания использования диска.

1. Ограничьте доступ пользователей к некоторым каталогам, введя:

   ```
   chmod 755 каталог
   ```

В данном случае устанавливаются разрешения на чтение и запись для владельца (root) и разрешения только на чтение для группы и остальных пользователей. *Каталог* - это полное имя каталога. доступ к которому необходимо ограничить.

2. Отслеживайте использование диска отдельными пользователями, добавив следующую строку в файл **/var/spool/cron/crontabs/adm**:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   ```

   Эта строка в 2 часа утра (`0 2`) каждый четверг (`4`) запускает команду **dodisk**. Команда **dodisk** осуществляет учет использования диска. Чаще всего она вызывается как часть процедуры учета, запускаемой командой **cron** в часы наименьшей загрузки системы.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Монтирование пространства с другого диска:

Вы можете увеличить объем пространства на диске путем монтирования пространства с другого диска.

Существует два способа монтирования свободного пространства с другого диска:

- С помощью команды **smit mountfs**.
- Воспользуйтесь командой **mount**. Например:

  ```
  mount -n nodeA -vnfs /usr/spool /usr/myspool
  ```

  Команда **mount** позволяет монтировать файловые системы в конкретном расположении.

Ссылки, связанные с данной:

“Обслуживание файловых систем” на стр. 438
В следующей таблице перечислены простейшие задачи по обслуживанию файловых систем.

Восстановление жесткого диска без повторного форматирования:

Если вы устранили неполадку диска и вернули его в систему, не выполняя форматирования, то система может автоматически активизировать этот диск и восстановить синхронизацию устаревших физических разделов диска во время загрузки. *Устаревшие физические разделы* содержат данные, которые система не может использовать.

Если вы подозреваете наличие устаревшего физического раздела, то введите следующую команду:

```
lspv -M
физический-том
```

где *физический-том* - это имя физического тома. Вывод команды **lspv** будет содержать список всех разделов физического тома. Ниже приведен пример вывода:

```
hdisk16:112     lv01:4:2        устар.
hdisk16:113     lv01:5:2        устар.
hdisk16:114     lv01:6:2        устар.
hdisk16:115     lv01:7:2        устар.
hdisk16:116     lv01:8:2        устар.
hdisk16:117     lv01:9:2        устар.
hdisk16:118     lv01:10:2       устар.
```

В первом столбце показаны физические разделы, во втором - логические. Устаревшие физические разделы отмечены в третьем столбце.

Задачи, связанные с данной:

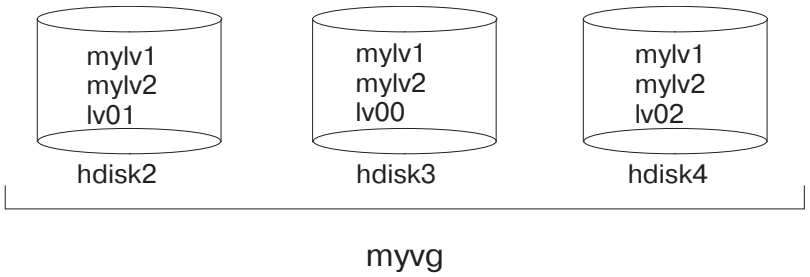
“Устранение неполадок дисков” на стр. 373
Приведенные ниже сведения касаются диагностики и устранения неполадок жестких дисков.

Восстановление с помощью переформатированного или замененного жесткого диска:

Вы можете восстановить данные с неисправного диска, когда необходимо повторно отформатировать или заменить неисправный диск.

Внимание: Перед форматированием или заменой неисправного диска следует удалить все ссылки на находящиеся на нем файловые системы без зеркальной защиты, а также удалить сам диск из группы томов и конфигурации системы. Если вы этого не сделаете, то могут возникнуть ошибки в базе данных ODM и базе данных конфигурации системы. Инструкции по выполнению этих действий приведены в описанной ниже процедуре под заголовком Перед заменой или переформатированием вышедшего из строя диска.

В следующей процедуре предполагается, что группа томов *myvg* содержит три диска: *hdisk2*, *hdisk3* и *hdisk4*. Неисправным считается диск *hdisk3*. На *hdisk2* хранится логический том *lv01* без зеркальной защиты и копия логического тома *mylv*. Логический том *mylv* с зеркальной защитой имеет три копии, каждая из которых занимает два физических раздела своего диска. Неисправный диск *hdisk3* содержит еще одну копию *mylv*, а также логический том *lv00* без зеркальной защиты. Наконец, *hdisk4* содержит третью копию *mylv* и логический том *lv02*. Этот сценарий показан на следующей схеме.



Данная процедура разделена на следующие основные части:

- Операции, которые необходимо выполнить для защиты данных перед заменой или форматированием неисправного диска
- Процедура форматирования или замены диска
- Операции, которые необходимо выполнить для восстановления данных после форматирования или замены диска

Перед заменой или форматированием неисправного диска

1. Войдите в систему как пользователь root.
2. Если вам неизвестно, какие логические тома хранятся на неисправном диске, то просмотрите его содержимое с помощью исправного диска. Например, для использования диска *hdisk4* с целью просмотра содержимого *hdisk3* введите следующую команду в командной строке:

   ```
   lspv -M -n hdisk4 hdisk3
   ```

   Команда **lspv** показывает информацию о физическом томе в группе. Вывод команды будет выглядеть примерно следующим образом:

   ```
   hdisk3:1        mylv:1
   hdisk3:2        mylv:2
   hdisk3:3        lv00:1
   hdisk3:4-50
   ```

   В первом столбце показаны физические разделы, во втором - логические. Разделы 4-50 не заняты.
3. Если это возможно, создайте резервную копию всех логических томов неисправного диска, у которых нет зеркальной копии. Инструкции приведены в разделе “Резервное копирование пользовательских файлов или файловых систем” на стр. 26.

4. Если вы работали с файловой системой без зеркальной копии, то размонтируйте ее. (Вы можете идентифицировать файловые системы без зеркальных копий по выводу команды **lspv**. У таких файловых систем число логических и физических разделов совпадает.) Для зеркальных файловых систем размонтирование не требуется.

   В нашем примере файловая система `lv00` на неисправном диске `hdisk3` не имеет зеркальной копии. Для ее размонтирования введите следующую команду:

   ```
   unmount /dev/lv00
   ```

   Если вам неизвестно имя файловой системы, то, предполагая, что файл `/etc/filesystems` расположен не на неисправном диске, введите команду `mount` для просмотра списка всех смонтированных файловых систем и найдите ту из них, которая связана с вашим логическим томом. Вы также можете воспользоваться командой **grep** для просмотра только тех записей файла `/etc/filesystems`, которые соответствуют нужному логическому тому. Например:

   ```
   grep lv00 /etc/filesystems
   ```

   Ниже приведен пример вывода этой команды:

   ```
   dev             = /dev/lv00
   log             = /dev/loglv00
   ```

   Примечания:

   a. Если файловая система, которую вы пытаетесь размонтировать, используется, то команда **unmount** выдаст сообщение об ошибке. Команда **unmount** выполняется лишь в том случае, если в файловой системе нет открытых файлов и на ее устройстве нет текущих пользовательских каталогов.

   b. Другое название команды **unmount** - **umount**. Эти имена равноценны.

5. Удалите все файловые системы без зеркальной копии с поврежденного физического тома с помощью команды **rmfs**.

   ```
   rmfs /файловая-система
   ```

6. Удалите все зеркальные логические тома поврежденного диска.

   Примечание: Не вызывайте команду **rmlvcopy** для логических томов `hd5` и `hd7`, относящихся к физическим томам группы rootvg. Система не позволит удалить эти логические тома, так как существует только одна их копия.

   Команда **rmlvcopy** удаляет копии из каждого логического раздела. Например, введите:

   ```
   rmlvcopy mylv 2 hdisk3
   ```

   Удалив копию на *hdisk3*, вы уменьшили число копий каждого логического раздела, принадлежащего логическому тому *mylv*, с трех до двух (один на *hdisk4* и один - на *hdisk2*).

7. Если неисправный диск входил в корневую группу томов и содержал логический том `hd7`, то удалите основное устройство дампа (`hd7`) с помощью следующей команды:

   ```
   sysdumpdev -P -p /dev/sysdumpnull
   ```

   Команда **sysdumpdev** служит для изменения расположения основного или дополнительного устройства дампа работающей системы. После перезагрузки устройство дампа будет снова находиться в исходном расположении.

   Примечание: Можно создавать дамп на DVD-приводе. Информация о настройке дисковода DVD в качестве устройства дампа приведена в описании команды **sysdumpdev**.

8. Удалите расположенные на диске пространства подкачки с помощью следующей команды:

   ```
   rmps имя-пространства-подкачки
   ```

   где *имя-пространства-подкачки* - это имя удаляемого пространства подкачки, фактически представляющее собой имя логического тома, на котором находится пространство подкачки.

Если при обработке команды **rmps** произошла ошибка, то с помощью команды **smit chps** необходимо отключить основное устройство подкачки и перезагрузить систему. Если в системе есть активные пространства подкачки, то во время обработки команды **reducevg** на шаге 10 может произойти ошибка.

9. Удалите из группы томов все остальные логические тома, в том числе те, которые не содержат файловых систем, с помощью команды **rmlv**. Например, введите:
   ```
   rmlv -f lv00
   ```
10. Удалите неисправный диск из группы томов с помощью команды **reducevg**. Например, введите:
    ```
    reducevg -df myvg hdisk3
    ```

    Если команду **reducevg** выполнить не удается, а также в том случае, если при выполнении команды возникает ошибка, обратитесь к шагу 13 и очистите информацию VGDA/ODM после форматирования или замены диска.

    Замена или форматирование неисправного диска:
11. Следующий шаг зависит от того, хотите ли вы отформатировать или заменить диск, а также от типа применяемого аппаратного обеспечения:
    - Если вы хотите отформатировать диск, выполните следующие действия:
      a. Войдите в систему как пользователь root и введите следующую команду:
         ```
         smit diag
         ```
      b. Выберите **Диагностика текущей оболочки** для запуска инструмента диагностики AIX.
      c. После прочтения **инструкций по работе со средствами диагностики** нажмите Enter.
      d. Выберите **Выбор задачи**.
      e. Пролистайте список задач и выберите пункт **Форматировать носитель**.
      f. Выберите диск для форматирования. После того, как вы подтвердите форматирование, все содержимое диска будет уничтожено.

      После форматирования диска перейдите к шагу 12.
    - Если система поддерживает оперативную замену дисков, то перейдите к процедуре “Восстановление неисправного диска при поддержании доступности системы” на стр. 380, а затем - к шагу 13.
    - Если система не поддерживает оперативную замену дисков, то выполните следующие действия:
      – Выключите старый диск с помощью команды SMIT **smit rmvdsk**. В поле Хранить определение в базе данных укажите `Нет`.
      – Для замены диска обратитесь в службу поддержки.

    После замены или форматирования неисправного диска
12. Выполните инструкции из разделов “Настройка диска” на стр. 364 и “Преобразование доступного диска в физический том” на стр. 366.
13. Если невозможно воспользоваться командой **reducevg** для удаления диска из старой группы томов перед его форматированием (шаг 10), то выполните следующие действия для очистки VGDA/ODM.
    a. Если группа томов состояла только из одного диска, который и был отформатирован, то введите:
       ```
       exportvg группа-томов
       ```

       где *группа-томов* - это имя группы томов.
    b. Если группа томов включает несколько дисков, то введите следующую команду:
       ```
       varyonvg группа-томов
       ```

       Система покажет сообщение об отсутствующем или недоступном диске, в котором будет указан новый или отформатированный диск. Запишите идентификатор физического тома (PVID) нового диска, указанный в сообщении **varyonvg**. Это строка из 16 символов, заключенная между именем отсутствующего диска и меткой PVNOTFND. Например:
       ```
       hdisk3 00083772caa7896e PVNOTFND
       ```

Введите:

```
varyonvg -f группа-томов
```

Отсутствующий диск показан с меткой PVREMOVED. Пример:

```
hdisk3 00083772caa7896e PVREMOVED
```

После этого введите команду:

```
reducevg -df группа-томов PVID
```

где PVID - идентификатор физического тома (в нашем примере - 00083772caa7896e).

14. Для добавления нового диска в группу томов введите команду **extendvg**. Например, введите:

    ```
    extendvg myvg hdisk3
    ```

15. Для создания на новом или отформатированном диске логических томов без зеркальной копии введите команду **mklv**. Например, введите:

    ```
    mklv -y lv00 myvg 1 hdisk3
    ```

    В данном примере логический том lv00 будет повторно создан на диске *hdisk3*. Число 1 означает, что у логического тома нет зеркальной копии.

16. Повторно создайте файловые системы на логическом томе с помощью команды **crfs**. Например, введите:

    ```
    crfs -v jfs -d lv00 -m /dev/lv00
    ```

17. Для восстановления данных в файловых системах без зеркальных копий перейдите к разделу “Восстановление пользовательских файлов из резервной копии” на стр. 31.

18. Для создания зеркальных копий логических томов воспользуйтесь командой **mklvcopy**. Например, введите:

    ```
    mklvcopy mylv 3 hdisk3
    ```

    В данном примере показано, как создать третий зеркальный раздел (логический том *mylv*) на *hdisk3*.

19. Для синхронизации новой зеркальной копии с данными других зеркальных копий (в данном примере - *hdisk2* и *hdisk4*) воспользуйтесь командой **syncvg**. Например, введите:

    ```
    syncvg -p hdisk3
    ```

В результате все файловые системы с зеркальной защитой должны быть восстановлены и синхронизированы. Если вам удалось создать резервные копии файловых систем без зеркальных копий, то они также готовы к работе. Нормальная работа системы должна быть восстановлена.

Задачи, связанные с данной:

“Устранение неполадок дисков” на стр. 373
Приведенные ниже сведения касаются диагностики и устранения неполадок жестких дисков.

Пример восстановления неисправного диска:

Для восстановления неисправного диска следует повторить все процедуры создания группы томов в обратном порядке.

В следующем примере показано, как был создан зеркальный логический том и как после сбоя диска он был изменен.

Примечание: Данный пример иллюстрирует лишь один из множества возможных случаев. Действия, выполненные в данном случае, нельзя считать универсальными.

1. Системный администратор, которого зовут Василий, создал группу томов **workvg** на диске hdisk1 с помощью следующей команды:

   ```
   mkvg -y workvg hdisk1
   ```

2. Затем он добавил в эту группу томов еще два диска:

```
extendvg workvg hdisk2

extendvg workvg hdisk3
```

3. После этого Василий создал логический том размером 40 МБ с тремя копиями. На каждом из трех дисков, составляющих группу **workvg**, находится одна копия. Он воспользовался следующей командой:

```
mklv -y testlv workvg 10

mklvcopy testlv 3
```

После того, как Василий создал группу томов workvg с зеркальной защитой, произошел сбой диска hdisk2. Таким образом, для восстановления потребовалось выполнить следующие действия:

1. Удалить копию логического тома с hdisk2:

```
rmlvcopy testlv 2 hdisk2
```

2. Отключить hdisk2 от системы для обновления ODM и VGDA:

```
reducevg workvg hdisk2
```

3. Удалить из конфигурации hdisk2 для подготовки к замене:

```
rmdev -l hdisk2 -d
```

4. Завершить работу системы:

```
shutdown -F
```

5. Заменить диск. Новый диск имеет тот же ИД SCSI, что и старый hdisk2.
6. Перезагрузить систему.

   Так как вы установили новый диск (система распознала новый PVID диска), то для него будет выбрано первое *свободное* имя hdisk. Поскольку на шаге 3 был указан флаг **-d**, то имя диска hdisk2 было освобождено, поэтому система выбрала в качестве нового имени имя hdisk2. Если флаг **-d** не был указан, то для нового диска будет выбрано имя hdisk4.

7. Василий добавил этот диск в группу томов **workvg**:

```
extendvg workvg hdisk2
```

8. Затем он создал две зеркальные копии логического тома:

```
mklvcopy testlv 3
```

   Администратор логических томов автоматически поместит третью копию логического тома на новый диск.

Восстановление неисправного диска при поддержании доступности системы:

Можно выполнить восстановление неисправного диска с помощью оперативной замены компонентов.

Процедура замены неисправного диска с помощью функции оперативной замены в целом совпадает с процедурой, описанной в разделе “Восстановление жесткого диска без повторного форматирования” на стр. 375, со следующими отличиями:

1. Для размонтирования файловых систем диска выполните процедуру Монтирование JFS или JFS2.
2. Для удаления диска из группы томов и операционной системы выполните процедуру “Удаление диска без данных” на стр. 400.
3. Для замены неисправного диска новым не требуется завершать работу системы. Выполните следующие действия:
   a. “Структура логических разделах” на стр. 385
   b. “Настройка диска” на стр. 364
   c. Перейдите к шагу 13 из “Восстановление с помощью переформатированного или замененного жесткого диска” на стр. 376.

Задачи, связанные с данной:

“Устранение неполадок дисков” на стр. 373
Приведенные ниже сведения касаются диагностики и устранения неполадок жестких дисков.

Замена диска в случае, когда группа томов состоит из одного диска:

Если диск, входящий в состав группы томов, почти вышел из строя, но остается доступным, то выполните одну из следующих процедур:

- “Перенос содержимого физического тома” на стр. 362

Если диск поврежден и недоступен, то выполните следующие действия:

1. Экспортируйте группу томов.
2. Замените диск.
3. Восстановите данные из резервной копии.

Ошибки физических и логических томов:

Вы можете столкнуться с некоторыми распространенными ошибками физических и логических томов и устранить их.

Неполадки активных областей:

Если вы заметили снижение производительности при обращении к логическим томам, то, возможно, на логических томах есть активные области, к которым происходит слишком много обращений в ходе операций ввода-вывода.

За дополнительной информацией обратитесь к разделу “Управление оперативной заменой в логических томах” на стр. 414.

Предупреждения LVCB:

Причиной возникновения предупреждений является наличие недопустимой информации в LVCB.

Блок управления логическим томом (LVCB) - это первый блок логического тома. Размер LVCB равен размеру блока физических томов в группе томов. В этой области хранится важная информация, например, дата создания логического тома, сведения о зеркальных копиях и возможных точках монтирования JFS. Некоторые команды LVM обновляют LVCB в соответствии с алгоритмами работы LVM. Старая копия LVCB считывается и анализируется на наличие ошибок. Если информация LVCB допустима, то LVCB обновляется. Если информация недопустима, то LVCB не обновляется и выводится следующее сообщение:

```
Внимание, невозможно записать
данные управляющего блока lv.
```

В большинстве случаев такое сообщение появляется, когда базы данных обращаются к логическим томам с прямым доступом непосредственно, минуя JFS. В таком случае информация базы данных записывается поверх LVCB. При использовании логических томов с прямым доступом это не опасно. После перезаписи LVCB пользователь может по-прежнему выполнить следующие операции:

- Увеличить размер логического тома
- Создать зеркальные копии логического тома
- Удалить логический том
- Создать журнализированную файловую систему

Однако существуют и ограничения на удаление LVCB. Логический том без LVCB нельзя импортировать в другие системы. Во время импорта команда LVM **importvg** просматривает LVCB всех определенных логических томов в группе томов. Если LVCB не существует, то в импортированной группе томов логический том будет по-прежнему определен в новой системе, а пользователи смогут обращаться к логическому тому с прямым доступом. Однако на практике чаще всего происходит следующее:

- Вся информация о JFS утрачивается и информация о точках монтирования не импортируется в новую систему. В этом случае необходимо создать новые точки монтирования, а доступность предыдущих данных, хранившихся в файловой системе, не гарантируется.
- Утрачивается часть информации логического тома, не относящаяся к JFS. В этом случае для заполнения ODM система использует информацию о логическом томе по умолчанию. Таким образом, вывод команды **lslv** может не соответствовать реальному логическому тому. Если на исходных дисках существуют копии логического тома, то информация не будет правильно отражена в базе данных ODM. С помощью команд **rmlvcopy** и **mklvcopy** восстановите копии логического тома и синхронизируйте ODM.

Ограничения физических разделов:

В соответствии с принципом построения Администратора логических томов (LVM), каждому логическому разделу соответствует физический раздел (PP). Каждый физический раздел, в свою очередь, соответствует нескольким секторам диска. LVM может отслеживать до 1016 физических разделов диска. В большинстве случаев диск использует не все 1016 отслеживаемых раздела.

При превышении данного ограничения может быть показано следующее сообщение:

```
0516-1162 extendvg: Внимание, размер физического
раздела размер-PP
требует создания число-PP разделов
для тома физический-том.  Для
группы томов
группа-томов установлено ограничение ограничение числа
физических разделов на том.  С помощью команды chvg
с флагом
-t измените максимальное число физических
разделов на
физический том для данной группы томов.
```

где

размер-PP
: Число от 1 МБ до 1 ГБ, степень 2.

число-PP
: Общее число физических разделов на диске, размер каждого раздела - *размер-PP*.

физический-том
: Имя физического тома, например, hdisk3.

группа-томов
: Имя группы томов.

ограничение
: Число, равное или кратное 1016.

Эти ограничения действуют в следующих ситуациях:

1. Если при создании группы томов с помощью команды **mkvg** указано число физических разделов диска в группе томов, превышающее 1016. Для обхода этого ограничения можно выбрать размер физического раздела из набора 1, 2, 4 (по умолчанию), 8, 16, 32, 64, 128, 256, 512, 1024 МБ и создать группу томов командой **mkvg -s**. Кроме того, вы можете указать коэффициент, позволяющий разместить на диске число физических разделов, кратное 1016. В этом случае группу томов следует создавать командой **mkvg -t**.
2. Если при добавлении диска в уже существующую группу томов с помощью команды **extendvg** для нового диска превышается ограничение 1016. В этой ситуации следует преобразовать существующую группу томов для размещения на диске числа физических разделов, кратного 1016. Это можно сделать командой **chvg -t**. Вы также можете заново создать группу томов с большим размером раздела, либо создать автономную группу томов, в которую будет включен только новый диск.

Ограничения разделов и rootvg

Если процедура установки обнаружит в корневой группе томов (rootvg) диск размером более 4 ГБ, то она изменит значение в команде **mkvg-s** так, чтобы разместить весь диск на доступных 1016 дорожках. Кроме того, при добавлении дисков в rootvg предполагается, что они определены с таким же размером физического раздела.

Ограничения разделов и системы RAID

В системах с RAID имя `/dev/hdiskX`, используемое LVM, может включать несколько дисков с размером, не равным 4 ГБ. В этом случае по-прежнему действует ограничение 1016. LVM ничего не известно об отдельных дисках, фактически входящих в состав `/dev/hdiskX`. LVM рассчитывает ограничение в 1016 исходя из распознаваемого размера `/dev/hdiskX`, а не фактического размера физических дисков, образующих `/dev/hdiskX`.

Синхронизации базы данных конфигураций устройств:

Из-за сбоя база данных конфигурации устройств может оказаться несогласованной с LVM. Синхронизировать базу данных конфигураций устройств можно с помощью сведений LVM.

Когда база данных конфигураций устройств становится несогласованной с LVM, команда работы с логическим томом может выдать, например, следующее сообщение об ошибке:

```
0516-322 База данных конфигурации устройств несогласованна...
```

ИЛИ

```
0516-306 В базе данных конфигурации устройств
не найден логический том имя-тома
.
```

(когда том с указанным *именем* обычно доступен).

Внимание: Не удаляйте записи `/dev`, соответствующие группам томов или логическим томам. Не изменяйте в базе данных записи о группах томов и логических томах с помощью Администратора объектных данных.

Для синхронизации базы данных конфигурации устройств с информацией LVM введите следующую команду (необходимы права доступа root):

```
synclvodm -v группа-томов
```

где *группа-томов* - это имя синхронизируемой группы томов.

Информация, связанная с данной:

“Исправление ошибок группы томов”
С помощью этих методов можно устранить ошибки групп томов.

Исправление ошибок группы томов:

С помощью этих методов можно устранить ошибки групп томов.

Если команда **importvg** работает неправильно, рекомендуется обновить базу данных конфигурации устройств.

Пропуск ошибок включения

Внимание: Пропуск ошибки включения не является стандартной операцией; перед продолжением проверьте все возможные источники ошибки: аппаратное обеспечение, кабели, адаптеры и источники питания. Пропуск ошибки целостности во время включения применяется только в качестве аварийной меры и только при отсутствии альтернатив (например, для копирования данных с неисправного диска). При пропуске ошибки целостности невозможно гарантировать целостность данных, хранящихся на выбранных копиях VGDA и VGSA.

Если вы решили принудительно включить группу томов, пропустив ошибку целостности, то для всех физических томов, отсутствующих во время включения, информация о состоянии физического тома будет изменена на Удален. Это значит, что все копии VGDA и VGSA будут удалены с этих физических томов. После этого такие физические тома больше не будут проверяться в ходе контроля целостности; их нельзя будет активировать в группе томов до тех пор, пока вы не вернете их в группу томов. Если целостность группы томов не нарушена, флаг -f команды varyonvg (используется для переопределения в случае потери целостности) будет игнорирован.

В следующих случаях вы можете пропустить сбой включения, чтобы получить доступ к данным в группе томов:

- Недоступные физические тома серьезно повреждены.
- Вы уверены, что при последнем включении группы томов был активен по крайней мере один из доступных сейчас физических томов, который должен также содержать действительную копию VGDA и VGSA. Удалите отсутствующие физические тома из конфигурации и выключите их питание до тех пор, пока не сможете выполнить их диагностику и ремонт.

Следующая процедура позволяет избежать потери целостности при отсутствии одного из дисков или при подозрении на возможность скорого сбоя диска:

1. Для временного удаления тома из группы томов введите:
   ```
   chpv -vr физический-том
   ```
   После завершения этой команды указанный *физический-том* больше не будет учитываться при контроле целостности. Однако, если вы попытаетесь выполнить команду **chpv** для дисков с двумя VGDA/VGSA в группе томов с двумя дисками, то эта команда выдаст сообщение об ошибке. Команда запрещает операции, приводящие к утрате целостности дисков.
2. При необходимости удалить диск с целью его ремонта выключите систему и удалите диск. (Инструкции приведены в разделе “Устранение неполадок дисков” на стр. 373.) После ремонта диска и его установки перейдите к следующему шагу.
3. Для того чтобы снова включить диск в группе томов в процедуру контроля целостности, введите:
   ```
   chpv -v a имя-PV
   ```
   Примечание: Команда **chpv** применяется только для изменения опции контроля целостности. Хранящиеся на диске данные по-прежнему находятся на нем, и их следует скопировать на другие диски, если вы не планируете возвращать данный диск в систему.

Предупреждение VGDA

В некоторых случаях неполадки возникают при добавлении нового диска в существующую группу томов или при создании новой группы томов. LVM выдает примерно следующее сообщение:

```
0516-1163 extendvg:
группа-томов уже содержит
максимальное число физических томов.  При максимальном числе
физических разделов на физический том, равном
```

ограничению, максимальное число
физических томов
в *группе-томов*
равно *максимальному-числу-дисков*.

Где:

группа-томов
Имя группы томов.

ограничение
Число, равное или кратное 1016.

максимальное-число-дисков
Максимальное число дисков в группе томов. Например, если на диске умещается 1016 физических разделов (PP), то *максимальное-число-дисков* равно 32; если 2032 - то *максимальное-число-дисков* равно 16.

Вы можете изменить файл `image.data` и воспользоваться измененными данными при установке, либо восстановить систему с помощью команды **mksysb** и заново создать группу томов большего размера. Дополнительная информация приведена в разделе *Установка и миграция*.

В прежних версиях AIX, когда ограничение было меньше 32 дисков, для **rootvg** делалось исключение из правила, определявшего максимальное число VGDA. Для предоставления большего объема свободного пространства при создании rootvg команда **mkvg -d** брала за основу число дисков, выбранное в меню установки. Это число **-d** было равно 7 для одного диска плюс один для каждого дополнительного диска. Например, при выборе двух дисков это число было равно 8, при выборе трех - 9, и т.д.

Понятия, связанные с данным:

“Синхронизации базы данных конфигураций устройств” на стр. 383
Из-за сбоя база данных конфигурации устройств может оказаться несогласованной с LVM. Синхронизировать базу данных конфигураций устройств можно с помощью сведений LVM.

Структура логических разделах

Логические тома - это организованные области данных на физических томах.

Для управления дисковой памятью применяется иерархическая структура. Каждому жесткому диску (*физическому тому*, PV) присвоено имя, например, `/dev/hdisk0`. Каждый физический диск, используемый в системе, входит в состав определенной *группы томов* (VG). Физические тома в составе группы разбиты на *физические разделы* (PP) одинакового размера. Для распределения данных на диске каждый физический том разделен на пять участков: **outer_edge**, **inner_edge**, **outer_middle**, **inner_middle** и **center**. Количество физических разделов в каждой области зависит от общего объема дисковой памяти.

Группа томов разбивается на один или несколько *логических томов* (LV). Пользователь рассматривает данные логического тома как непрерывную последовательность, но на физическом томе они могут располагаться в нескольких несмежных областях. Таким образом обеспечивается изменение размера и расположения файловых систем (пространств подкачки и пр.), а также распределение данных по физическим томам (возможно, с дублированием).

Каждый логический том состоит из одного или нескольких *логических разделов* (LP). Каждый логический раздел соответствует по крайней мере одному физическому разделу. Если для логического тома применяется зеркальное копирование, то для хранения дополнительных копий логического тома выделяются дополнительные физические разделы. Несмотря на то что логические разделы последовательно пронумерованы, связанные с ними физические разделы не обязательно располагаются последовательно и непрерывно.

Логические тома применяются для различных целей, например для организации подкачки, но каждый конкретный логический том может выполнять только одну функцию. Журнализированная файловая система

(JFS или JFS2) располагается на нескольких логических томах. Каждая JFS представляет собой пул блоков, размер которых совпадает с размером страницы (4 Кб). При записи данных в файл этому файлу выделяются дополнительные блоки. Не требуется, чтобы эти блоки были смежными или располагались рядом с другими блоками, выделенными файлу. Размер фрагмента для файловой системы должен быть меньше 4 Кб (512 байт, 1 Кб или 2 Кб).

После установки в системе создается одна группа томов (rootvg), в которую входит базовый набор логических томов, необходимых для запуска системы, а также томов, заданных в сценарии установки. Другие физические тома, подключенные к системе, можно добавить в группу томов с помощью команды **extendvg**. Физический том можно добавить либо в группу томов rootvg, либо в другую группу томов (группу томов можно создать командой **mkvg**). Характеристики логических томов можно изменять с помощью команд или Инструмента управления системой (SMIT).

Задачи, связанные с данной:

“Перенос содержимого физического тома” на стр. 362

“Уведомление администратора об отсутствующем физическом томе” на стр. 370
Несмотря на то, что при недоступности тома AIX заносит в протокол сообщение об ошибке, существуют ситуации, в которых такая ошибка может остаться незамеченной.

“Выделение зеркального диска из группы томов” на стр. 372
Поддержка моментальных копий помогает защитить согласованность групп томов с зеркальной копией в случае возможного сбоя диска.

“Уменьшение размера файловой системы в корневой группе томов” на стр. 443
Простейший способ сокращения размера *всех* файловых систем до их минимального размера заключается в указании в опции **Сократить** значения **да** при восстановлении базовой операционной системы из резервной копии.

Подготовка к установке устройства

Установка устройства в системе заключается в определении расположения устройства, в физическом подключении и настройке устройства с помощью Администратора конфигурации, или SMIT.

Примечание: Для установки устройства перед выполнением описанной ниже процедуры необходимо выключить питание системы. Это требование не относится к установке всех устройств. Обратитесь к документации, прилагаемой к конкретному устройству.

В этом разделе описаны задачи установки, общие для всех устройств. Так как в системе можно установить самые разнообразные устройства, здесь приведена только общая процедура. Дополнительная информация приведена в инструкциях, поставляемых с конкретным устройством.

1. Завершите работу всех приложений, запущенных в системе, и отключите питание системного блока с помощью команды **shutdown**.
2. Выключите системный блок и все подключенные устройства.
3. Отсоедините от сети системный блок и все подключенные устройства.
4. Подключите новое устройство к системе в соответствии с инструкциями, приведенным в руководстве по настройке и эксплуатации для данного устройства.
5. Включите в сеть системный блок и все подключенные устройства.
6. Включите все подключенные устройства, оставив системный блок выключенным.
7. После того как для всех устройств будет выполнена начальная проверка (POST), включите системный блок.

Администратор конфигурации автоматически просмотрит все подключенные устройства и настроит обнаруженные новые устройства. При настройке для новых устройств будут приняты значения атрибутов по умолчанию, конфигурация будет занесена в базу данных настраиваемых конфигураций, а устройству будет присвоено состояние **Доступно**.

Вы можете настроить устройство вручную с помощью команды быстрого доступа SMIT **smit dev**. При необходимости настройки атрибутов устройства, а также в том случае, когда автоматическая настройка устройства невозможна, обратитесь к документации по этому устройству.

Понятия, связанные с данным:

“База данных конфигураций устройств и управление устройствами” на стр. 527
Информация об устройствах содержится в базах данных описаний и настройки, которые составляют базу данных конфигурации устройств.

Настройка перезаписывающего оптического накопителя

Существует два способа настройки перезаписывающих оптических накопителей:

Перезаписывающий оптический накопитель следует подключить к системе и затем включить.

Способ 1

Первый способ является самым быстрым. При этом выполняется настройка только указанного перезаписывающего оптического накопителя. Необходимо ввести следующую информацию:

Элемент	Описание
Подкласс	Определяет способ подключения накопителя.
Тип	Задает тип накопителя.
Имя родительского устройства	Задает системное устройство, к которому подсоединен накопитель.
Расположение подключения	Задает логический адрес накопителя.

Для настройки перезаписывающего оптического накопителя введите следующую команду:

```
mkdev -c rwoptical -s Подкласс -t тип -p  Имя_родительского_устройства -w Расположение_подключения
```

Ниже приведен пример перезаписывающего оптического накопителя, подключенного к третьей шине SCSI (scsi3) с ИД SCSI, равным 6, и номером логического блока, равным нулю:

```
mkdev -c rwoptical -s scsi -t osomd -p scsi3 -w 6,0 -a pv=yes
```

Способ 2

Второй способ настройки предполагает применение программы Администратор настройки, которая анализирует текущую конфигурацию, обнаруживает новые устройства и автоматически настраивает их. Этот способ применяется в том случае, если известна не вся информация об перезаписывающем оптическом накопителе.

1. С помощью следующей команды запустите Администратор настройки и настройте все обнаруженные устройства (включая данный перезаписывающий оптический накопитель):

   ```
   cfgmgr
   ```

2. Введите следующую команду для просмотра списка имен, кодов расположения и типов всех настроенных в настоящее время перезаписывающих оптических накопителей:

   ```
   lsdev -C -c rwoptical
   ```

3. Определите имя нового настроенного перезаписывающего оптического накопителя по коду расположения, совпадающего с расположением добавленного накопителя.

Настройка большого числа устройств

Под устройствами понимаются как аппаратные устройства, такие как принтеры, дисководы, адаптеры, шины и корпуса, так и псевдоустройства, такие как специальный файл ошибок и специальный пустой файл. Драйверы устройств расположены в каталоге `/usr/lib/drivers`.

Число поддерживаемых AIX устройств зависит от нескольких параметров системы. На файловые системы, поддерживающие устройства, влияют следующие факторы:

- Для настройки большого числа устройств потребуется дополнительное пространство в базе данных ODM. Кроме того, может потребоваться больше специальных файлов устройств. В результате в файловой системе будет занято больше памяти и больше i-узлов.
- Информация о различных устройствах занимает различный объем в базе данных ODM. Число специальных файлов и i-узлов также зависит от типа устройства. Таким образом, объем памяти и число i-узлов в файловой системе зависит от типа установленных устройств.
- Устройства разветвленного ввода-вывода (MPIO) требуют больше памяти, чем прочие устройства, поскольку в базе данных ODM хранится информация не только об устройстве, но и обо всех путях к устройству. В общем случае, каждый путь требует в пять раз меньше памяти, чем устройство. Например, устройство MPIO с пятью путями требует столько же памяти, сколько два устройства без поддержки MPIO.
- В базе данных конфигурации ODM AIX хранится информация как о логических, так и о физических устройствах. К логическим устройствам относятся группы томов, логические тома, сетевые интерфейсы и т.п. В некоторых случаях связь между логическими и физическими устройствами может существенно повлиять на число поддерживаемых устройств. Например, если вы определите группу томов с двумя логическими дисками на каждом физическом диске, подключенном к системе, то в AIX будет создано четыре устройства для каждого диска. В то же время, если вы определите группу томов с шестью логическими дисками на каждом физическом диске, подключенном к системе, то в AIX будет создано восемь устройств для каждого диска. Таким образом, удастся подключить вдвое меньше устройств.
- Изменение атрибутов устройств по умолчанию увеличивает объем базы данных ODM, сокращая общее число поддерживаемых устройств.
- Для поддержки большего числа устройств требуется больше оперативной памяти.

Для поддержки устройств в AIX применяются две файловые системы:

- Файловая система RAM применяется при загрузке в среде без пространства подкачки и смонтированных дисковых файловых систем. Размер файловой системы RAM составляет 25% от объема памяти системы, но не более 128 МБ. Для размещения каждого килобайта файловой системы RAM требуется один i-узел. Минимальный объем памяти для операционной системы AIX равен 256 МБ, поэтому минимальный размер файловой системы RAM (с 65536 i-узлами) составляет 64 МБ. Если объем памяти системы превышает 512 МБ, то файловая система RAM будет занимать 128 МБ и 131072 i-узлов. Если для поддержки устройств требуется больший объем файловой системы RAM или большее число i-узлов, то система может не загрузиться. В этом случае вам придется удалить некоторые устройства.
- Размер и число i-узлов корневой файловой системы (rootvg) на диске можно увеличить, если в rootvg есть нераспределенные разделы. Максимальный объем файловой системы RAM позволяет настроить до 25000 устройств AIX. В это число включены как логические, так и физические устройства. В зависимости от других факторов, изложенных в этом разделе, ваша система может поддерживать как большее, так и меньшее число устройств.

Примечание: Увеличение числа устройств в системе ведет к увеличению времени настройки и, как следствие, времени загрузки системы.

Понятия, связанные с данным:

“Введение в AIX системных администраторов BSD” на стр. 322
Ниже приведены советы для системных администраторов Berkeley Software Distribution (BSD), начинающих работать с AIX.

Добавление съемного носителя

Вы можете добавить съемный носитель.

Следующая процедура позволяет добавить в систему дисковод CD-ROM с помощью SMIT. Другие типы устройств для съемных носителей применяют другие команды, однако следуют той же общей процедуре. Вы также можете добавить устройство для съемных носителей с помощью Администратора настройки или с помощью команды **mkdev**.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Для добавления в систему дисковода CD-ROM установите аппаратное обеспечение в соответствии с документацией по системе.
2. Войдя в систему под именем пользователя с правами доступа root, введите следующую команду SMIT:
   ```
   smit makcdr
   ```
3. На следующем экране выберите в списке поддерживаемых устройств тип дисковода.
4. На следующем устройстве выберите в списке родительский адаптер.
5. На следующем экране необходимо по-крайней мере выбрать в списке адрес для подключения. С помощью этого экрана можно также задать другие опции. После завершения процедуры нажмите Enter и SMIT добавит в систему дисковод CD-ROM.

Теперь новый дисковод CD-ROM распознан системой. Для добавления перезаписываемого оптического устройства воспользуйтесь командой **smit makomd**. Для добавления накопителя на магнитной ленте можно воспользоваться командой **smit maktpe**.

Дополнительные сведения приведены в описании команды **mkdev** в книге *Справочник по командам, том 3*.

Поддержка восстановления памяти для структуры логических томов

В AIX 7.2 с технологическим пакетом обслуживания 7200-01 и выше администратор логических томов (LVM) поддерживает освобождение дисковой памяти для физических томов, которые могут освободить память.

LVM информирует драйвер диска, который, в свою очередь, сообщает подсистеме хранения данных, что память раздела больше не используется и что подсистема хранения данных может вернуть выделенную память. Драйвер диска помогает LVM обнаружить возможность восстановления памяти физического тома. Команды конфигурации файловой системы и LVM, такие как `rmlv`, `rmlvcopy` и `chfs(shrink fs)` запускают восстановление памяти для разделов после их освобождения. LVM обнаруживает возможность восстановления памяти физического тома при открытии тома во время выполнения команды `varyonvg` или `extendvg`. Также LVM пытается определить эту возможность во время работы группы томов. Если для определения изменения состояния требуется повторно открыть физический том, администратор должен выполнить для группы томов команду `varyoffvg`, а затем `varyonvg`.

В группах томов, созданных до AIX 7.2 с технологическим пакетом обслуживания 1, свободная дисковая память раздела может быть недоступна для автоматического освобождения. Для восстановления этой памяти в таких свободных разделах администратор может создать и удалить фиктивный логический том. Но для разделов, освобожденных после установки AIX 7.2 с технологическим пакетом обслуживания 1, память будет восстановлена автоматически.

Процесс LVM, восстанавливающий дисковую память, будет запущен в фоновом режиме после выполнения команды `rmlv`. Если в системе возникнет сбой до завершения процессом LVM обработки восстановления для всех разделов, то разделы будут освобождены, но память не будет восстановлена для ожидающих разделов. В этом случае можно создать и удалить фиктивный логический том для восстановления памяти оставшихся разделов.

Процесс LVM не вызывает задержки при выполнении команды `varyoffvg` или `reducevg`, даже если процесс освобождения памяти находится в ожидающем режиме. Вместо ожидания завершения процесса освобождения памяти он будет аннулирован.

Примечание: Команды переходят в режим ожидания только для еще не обработанных запросов освобождения памяти, отправленных в драйвер диска.

Функция освобождения памяти доступна в подсистеме хранения данных для восстановления освобожденной памяти на физическом томе. Любая подсистема хранения поддерживает запросы на освобождение памяти, кратной определенному числу физических блоков, однако количество этих блоков зависит от типа

подсистемы хранения. Поэтому иногда восстановить блоки (все или некоторые) раздела не удается. так как не выполнено выравнивание размера восстанавливаемой памяти и физических блоков раздела. Некоторые подсистемы хранения данных поддерживают восстановление блоков, размер которых превышает размер раздела LVM. и частичное восстановление блока не поддерживается. В такой ситуации LVM не всегда сможет собрать достаточно последовательных свободных разделов для создания даже одного запроса на восстановление. Поэтому при удалении нескольких разделов LVM можно не запросить такой же объем памяти в подсистеме хранения данных. Для получения информации о созданных LVM запросах восстановления памяти можно использовать команду `lvmstat` с опцией **-r**.

Информация, связанная с данной:

команда varyoffvg

Определение структуры логических разделов

Логический том, распределенный между физическими томами, разбит на логические разделы; физические тома размещены в физических разделах.

Связи основных определений структуры логических разделов.

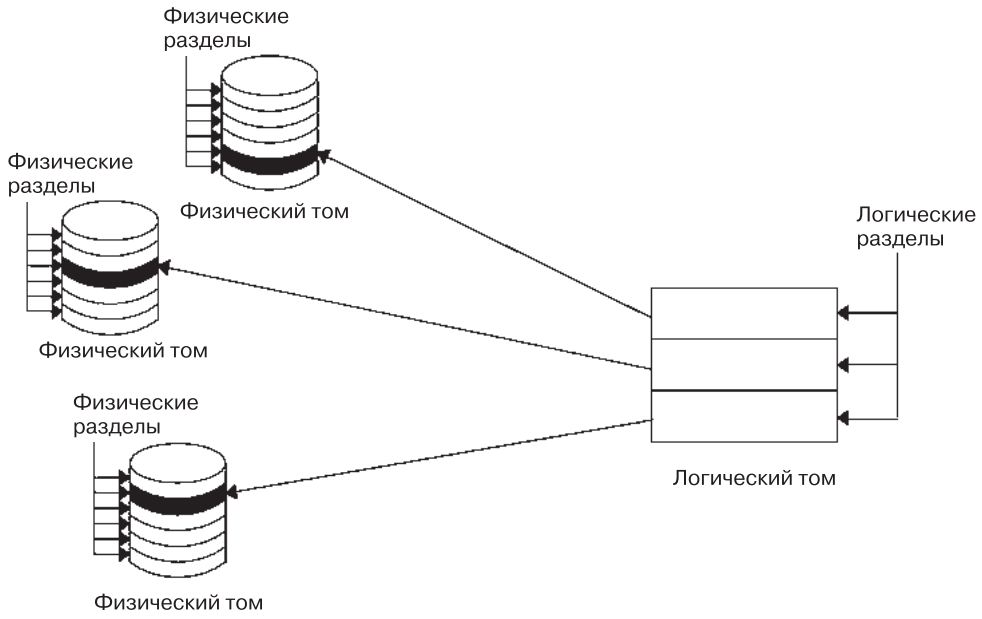


Рисунок 1. Группа томов. На данной схеме изображена группа томов, состоящая из трех физических томов, для которых указан максимальный диапазон. Логический том, распределенный между физическими томами, разбит на логические разделы; физические тома размещены в физических разделах.

Физические тома:

Перед тем как включить диск в группу томов, его необходимо настроить как физический том и сделать доступным для системы.

Каждому физическому тому соответствует определенная идентификационная информация и информация о настройке, в том числе идентификатор физического тома, уникальной в пределах данной системы.

LVM может использовать дополнительное пространство, добавляемое из массива RAID к LUN. Для этого необходимо добавить физические разделы к физическому тому, связанному с LUN.

Группы томов:

Группа томов - это набор, содержащий от 1 до 32 физических томов различной емкости и типа.

В большую группу томов может входить от 1 до 128 физических томов. Масштабируемая группа томов может содержать максимум 1024 физических тома. Физический том может принадлежать только одной группе томов. Всего допустимо до 255 активных групп томов.

Когда физический том включается в группу томов, физические блоки носителя объединяются в физические разделы, размер которых задается при создании группы томов.

После установки в системе создается одна группа томов (rootvg), в которую входит базовый набор логических томов, необходимых для запуска системы, а также томов, заданных в сценарии установки. В группу томов rootvg входят следующие компоненты: пространство подкачки, протокол журналов, данные загрузки и область для размещения дампа, - каждый из этих компонентов представляет собой отдельный логический том. Атрибуты rootvg отличаются от атрибутов пользовательских групп томов. Например, группу rootvg нельзя импортировать или экспортировать. При вызове команд для выполнения действий над rootvg необходимо учитывать особенности этой группы томов.

Новую группу томов можно создать командой **mkvg**. Добавление физического тома в группу томов выполняется командой **extendvg**, удаление - командой **reducevg**, а изменение размера - командой **chvg**. Над группами томов можно выполнять следующие операции: просмотр списка (**lsvg**), удаление (**exportvg**), установка (**importvg**), реорганизация (**reorgvg**), синхронизация (**syncvg**), разрешение использования (**varyonvg**) и запрещение использования (**varyoffvg**).

В небольших системах все подключенные физические тома могут размещаться в одной группе томов. Однако, в целях защиты рекомендуется создавать отдельные группы томов, поскольку можно задавать права доступа на уровне групп томов. Кроме того, использование нескольких групп томов упрощает обслуживание системы, так как во время обслуживания одной группы другие могут оставаться активными. Поскольку корневая группа томов должна быть активной постоянно, она содержит лишь минимальное число физических томов, необходимых для работы системы.

С помощью команды **migratepv** можно переносить данные с одного физического тома на другой *в пределах одной группы томов*. Кроме того, эта команда позволяет освобождать физические тома перед их удалением из группы томов. Например, можно переместить данные с физического тома, который необходимо заменить.

Группу томов, при создании которой использовался формат с небольшим максимальным числом физических и логических томов, можно преобразовать в формат, позволяющий добавить дополнительные физические и логические тома. Для выполнения этой операции необходимо, чтобы число разделов на каждом физическом томе, входящем в группу томов, было достаточным для расширения области дескрипторов группы томов (VGDA). Необходимое число свободных разделов зависит от текущего размера VGDA и физического раздела. Так как VGDA располагается на краю диска и должна занимать непрерывную область, то для выполнения этой операции необходимо наличие свободных разделов в этой части диска. Если эти разделы выделены пользователям, то данные из них будут перенесены в другие свободные разделы того же диска. Остальные физические разделы будут перенумерованы с учетом разделов, выделенных под VGDA. При этом изменится соответствие между логическими и физическими разделами на всех физических томах в составе данной группы томов. Если информация о соответствии между логическими и физическими разделами была сохранена (например, для восстановления после возможного сбоя), то после преобразования сохраненная информация устареет. Кроме того, если резервное копирование группы томов выполняется с опцией map (с сохранением информации о размещении) и восстановление выполняется с использованием сохраненной информации, то операция восстановления может завершиться неудачно, так как разделы с некоторыми номерами могут оказаться несуществующими (из-за сокращения числа разделов при преобразовании). При использовании опции map рекомендуется выполнять резервное копирование непосредственно перед преобразованием и сразу после него. Так как размер области VGDA существенно увеличился, то каждая операция, связанная с ее изменением (создание или изменение логического тома, добавление физического тома и т.п.), может занимать продолжительное время.

Понятия, связанные с данным:

“Физические разделы”
При добавлении физического тома в группу томов этот том разбивается на блоки одинакового размера, которые называются *физическими разделами*. Физический раздел - это наименьшая единица распределения дискового пространства, занимающая непрерывный участок физического тома.

Физические разделы:

При добавлении физического тома в группу томов этот том разбивается на блоки одинакового размера, которые называются *физическими разделами*. Физический раздел - это наименьшая единица распределения дискового пространства, занимающая непрерывный участок физического тома.

Размер раздела физического тома равен размеру раздела, указанному при создании группы томов (например, с помощью команды **mkvg -s**). На следующем рисунке показана связь физических разделов и групп томов.

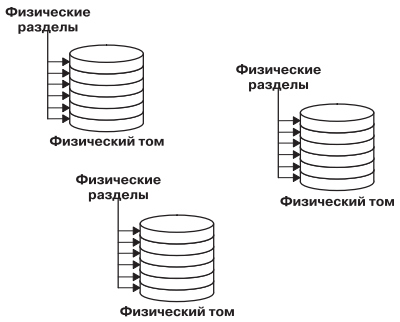


Рисунок 2. Группа томов, включающая три физических тома. На рисунке показана группа томов, в которую входит три физических тома, по шесть физических разделов в каждом.

Понятия, связанные с данным:

“Группы томов” на стр. 390
Группа томов - это набор, содержащий от 1 до 32 физических томов различной емкости и типа.

Логические тома:

После создания группы томов вы можете создать в ней логические тома.

Несмотря на то, что *логический том* может быть размещен не только в нескольких разных физических разделах, но даже в разных физических томах, с точки зрения пользователей он представляет собой единый, непрерывный том с возможностью расширения. С помощью команды **mklv** можно создавать дополнительные логические тома. Эта команда позволяет задавать имя логического тома и определять его характеристики, например количество и расположение логических разделов.

Для изменения имени и характеристик логического тома после его создания служит команда **chlv**, а для изменения количества выделенных логических разделов - команда **extendlv**. По умолчанию максимальный размер логического тома составляет 512 логических разделов, однако можно указать и большее значение. Это ограничение можно изменить командой **chlv**.

Примечание: После создания логического тома значение параметра Состояние LV, которое можно просмотреть командой **lslv**, изменится на Закрыт. Значение Открыт присваивается этому параметру, например, при создании и монтировании файловой системы на данном логическом томе.

Логические тома можно копировать командой **cplv**, просматривать командой **lslv**, удалять командой **rmlv**, а также уменьшать и увеличивать число поддерживаемых копий командами **mklvcopy** и **rmlvcopy** соответственно. Кроме того, при реорганизации группы томов можно изменять расположение логических томов.

В обычной группе томов может быть определено до 255 логических томов (в большой группе - до 511 томов, а в масштабируемой группе - до 4095 томов), но фактическое число томов в группе зависит от выделенного ей объема физической памяти и размера создаваемых логических томов.

Логические разделы:

При создании логического тома для него задается количество *логических разделов*.

Логический раздел соответствует одному, двум или трем физическим разделам в зависимости от количества экземпляров данных, которые вы хотите поддерживать. По умолчанию создается единственная экземпляр логического тома. В этом случае один логический раздел непосредственно соответствует одному физическому разделу. Каждый экземпляр, в том числе первый, называется *копией*. Расположение физических разделов на диске зависит от опций, указанных при создании логического тома.

Файловые системы:

Логический том определяет распределение дискового пространства вплоть до уровня физических разделов. Более точное управление данными достигается путем применения программных компонентов, например, администратора виртуальной памяти и файловых систем. Поэтому последний этап организации дисковой памяти - это создание *файловых систем*.

Для каждого логического тома можно создать отдельную файловую систему. Для создания файловой системы служит команда **crfs**.

Понятия, связанные с данным:

“Файловые системы” на стр. 425
Файловая система - это иерархическая структура (дерево) файлов и каталогов.

Ограничения на логическую память:

В приведенной ниже таблице описаны ограничения на логическую память.

По умолчанию максимальное число физических томов в группе равно 32 (в большой группе томов - 128, а в масштабируемой группе - 1024), но при создании пользовательских групп томов это число можно указать в команде **mkvg**. При создании группы rootvg во время установки системы этой переменной автоматически присваивается максимальное значение.

Ограничения на логическую память

Категория	Ограничение
Группы томов	• 255 групп томов для 32-битового ядра • 4096 групп томов для 64-битового ядра **Примечание:** В таблице устройств по 64-ому ядру числовое ограничение основных активных чисел равно 1024. В результате, число активных групп томов ограничено до 1024.
Физические тома	(MAXPVS / коэффициент для группы томов) на группу томов. MAXPVS равен 32 для обычной группы томов, 128 для большой группы томов и 1024 для масштабируемой группы томов.
Физические разделы	Обычные и большие группы томов: (1016 x коэффициент для группы томов) на физический том; размер каждого - до 1024 МБ. Масштабируемые группы томов: до 2097152 разделов размером до 126 ГБ. Для масштабируемых групп томов коэффициент не вычисляется.

Ограничения на логическую память

Категория	Ограничение
Логические тома	MAXLVS для группы томов: 255 для обычной группы, 511 для большой группы и 4095 для масштабируемой группы.

Если группа томов была создана до того, как вступило в силу ограничение, разрешающее создание не более 1016 физических разделов на физический том, и устаревшие (не содержащие текущие данные) разделы группы томов не преобразованы в поддерживаемый формат, то они будут обрабатываться неправильно. Для преобразования группы томов воспользуйтесь командой **chvg -t**. Значение коэффициента по умолчанию выбирается исходя из требования поддержки максимального по размеру диска в группе томов.

Например, если в состав созданной группы томов входит диск, объем которого равен 9 ГБ, а размер раздела составляет 4 МБ, то эта группа томов будет содержать примерно 2250 разделов. Для работы с 2250 разделами коэффициента преобразования должен быть равен 3 (1016 * 3 = 3048). Применение большего коэффициента при преобразовании обычной или большой группы томов позволяет включать в группу диски с числом разделов, равным 1016 * коэффициент. Кроме того, большее значение коэффициента можно задать для включения в группу томов дисков большего объема с разделами меньшего размера.

Выполнение этих операций приводит к сокращению общего числа дисков, которые можно включить в группу томов. Новое максимальное число дисков будет равно MAXPVS/коэффициент. Например, применение коэффициента 2 для обычной группы томов приводит к уменьшению максимального числа дисков в группе до 16 (32/2). Для большой группы томов применение коэффициента 2 приводит к уменьшению максимального числа дисков в группе томов до 64 (128/2).

Максимальный размер устройств LVM

Следующие ограничения обусловлены архитектурными ограничениями LVM. Если требуется перемещение поврежденных блоков LVM, значения PV не должны превышать 128 ГБ. Информация о минимальных размерах отдельных устройств дисковой памяти приведена в документации по устройствам дисковой памяти.

Следующие ограничения размеров относятся к 64–разрядному ядру:

Исходная VG

Ограничение для PV: 1 ГБ (PP) * 16256 (PPs/PV, коэффициент = 16) = 15.9 Тб

Ограничение для LV: 1 ГБ (PP) * 32512 (PPs/VG) = 31.8 Тб

Большая VG

Ограничение для PV: 1 ГБ (PP) * 65024 (PPs/PV, коэффициент = 64) = 63.5 Тб

Ограничение для LV: 1 ГБ (PP) * 130048 (PPs/VG) = 127 Тб

SVG Ограничение для PV & LV: 128 ГБ (PP) * 2048 K (PPs/PV) = 256 Пб

Следующие ограничения размеров относятся к 32–разрядному ядру:

Все типы VG

Ограничение для PV: < 1 Тб

Ограничение для LV: < 1 Тб

Настройка дисковой памяти логических томов

С помощью дисковой памяти логических томов, можно создавать зеркальные копии групп томов, определять логический том и удалять диск во время работы системы.

Зеркальная защита группы томов:

Эти сценарии иллюстрируют применение зеркальной защиты для обычной группы томов.

В приведенных инструкциях описано включение зеркальной защиты для группы томов rootvg с помощью SMIT.

(выделите в контейнере **Тома** группу томов, а затем выберите пункт **Зеркальная защита** в меню **Выбранные**). Опытные администраторы могут воспользоваться командой **mirrorvg**.

1. Войдя в систему под именем пользователя с правами доступа root, добавьте диск в группу томов с помощью следующей команды SMIT:

   ```
   smit extendvg
   ```

2. Создайте на новом диске зеркальную копию группы томов с помощью следующей команды SMIT:

   ```
   smit mirrorvg
   ```

3. На первой панели выберите группу томов для зеркальной защиты.
4. На второй панели вы можете задать опции зеркальной защиты или оставить значения по умолчанию. Для получения дополнительной информации обратитесь к электронной справке.

Примечание: Выполнение операции, начатое после ввода параметров в SMIT и нажатия кнопки OK или выхода, может занять довольно много времени. Продолжительность зависит от необходимости проверки ошибок, размера и числа логических томов в группе томов, а также от времени, необходимого для синхронизации зеркальных копий логических томов.

Теперь все изменения, вносимые в логические тома, будут дублироваться на зеркальных копиях в соответствии с опциями, указанными в SMIT.

Задачи, связанные с данной:

“Включение зеркальной защиты для корневой группы томов”
Ниже описано включение зеркальной защиты для группы томов rootvg.

Включение зеркальной защиты для корневой группы томов:

Ниже описано включение зеркальной защиты для группы томов rootvg.

Примечание: Включение зеркальной защиты для rootvg требует наличия опыта администрирования системы. При неправильном выполнении этой операции загрузка системы может стать невозможной.

В следующем сценарии группа томов rootvg размещается на диске `hdisk01`, а зеркальная копия создается на диске `hdisk11`:

1. Убедитесь, что `hdisk11` поддерживается в AIX в качестве загрузочного устройства:

   ```
   bootinfo -B hdisk11
   ```

   Если эта команда вернет значение 1, значит AIX может загружаться с указанного диска. Другое значение указывает. что `hdisk11` нельзя применять для зеркальной защиты rootvg.

2. Включите `hdisk11` в rootvg:

   ```
   extendvg rootvg hdisk11
   ```

   При получении сообщения об ошибке

   ```
   0516-050 Недостаточно дескрипторов в группе томов, попытайтесь
   добавить меньший физический том или укажите другую группу томов.
   ```

   или сообщения, аналогичного

   ```
   0516-1162 extendvg: Внимание, размер физического раздела 16 требует
   создания 1084 разделов для hdisk11. Для группы томов rootvg
   ограничение
   составляет в 1016 физических разделов на физический том. Укажите команду
   chvg c
   опцией -t, чтобы попытаться изменить максимальное число физических
   разделов
   на физический том для этой группы томов.
   ```

Возможны следующие варианты действий:

- Создайте зеркальную копию rootvg на пустом диске, уже входящем в rootvg.
- Воспользуйтесь диском меньшего объема.
- Измените максимальное число разделов, поддерживаемых rootvg:
  a. Посмотрите в сообщении, какое число физических разделов необходимо для целевого диска и каково текущее максимальное число, поддерживаемое rootvg.
  b. Введите команду **chvg -t**, указав в ней множитель текущего числа разделов, поддерживаемых в rootvg (в нашем примере 1016), позволяющий получить число, превышающее число физических разделов, необходимых для целевого диска (в нашем примере 1084). Например:

     ```
     chvg -t 2 rootvg
     ```

  c. Повторите ввод команды **extendvg**, указанной в начале шага 2.

3. Включите зеркальную защиту rootvg, воспользовавшись опцией точного преобразования, как в приведенной ниже команде:

   ```
   mirrorvg -m rootvg hdisk11
   ```

   Эта команда отключит контроль целостности для группы томов rootvg. Если вы не воспользуетесь опцией точного преобразования, то необходимо будет убедиться, что новая копия загрузочного логического тома, hd5, размещена в смежных разделах.
4. Инициализируйте все загрузочные записи и устройства:

   ```
   bosboot -a
   ```

5. Инициализируйте список загрузки:

   ```
   bootlist -m normal hdisk01 hdisk11
   ```

   Примечание:
   a. Даже в том случае, если в команде **bootlist** hdisk11 указан в качестве альтернативного загрузочного диска, невозможно гарантировать, что система использует `hdisk11` для загрузки в случае сбоя `hdisk01`. В этом случае вам может потребоваться загрузиться с носителя продукта, выбрать режим **обслуживания** и еще раз ввести команду **bootlist**, не указывая в ней диск, на котором произошел сбой.
   b. Если модель вашего аппаратного обеспечения не поддерживает команду **bootlist**, то вы все равно можете обеспечить зеркальную защиту rootvg, но при этом необходимо будет явно указать альтернативный загрузочный диск в случае сбоя исходного диска.

Задачи, связанные с данной:

“Зеркальная защита группы томов” на стр. 394
Эти сценарии иллюстрируют применение зеркальной защиты для обычной группы томов.

Создание логического тома с прямым доступом для приложения:

Логический том с прямым доступом - это область физического и логического дискового пространства, которой приложение управляет непосредственно, минуя операционную систему или файловую систему. Это может быть, например, раздел или база данных.

Обращение к диску, минуя файловую систему, позволяет приложению достичь более высокой производительности, особенно при работе с базами данных. Величина выигрыша при этом зависит от таких факторов, как размер базы данных или драйвер приложения.

Примечание: Для работы с логическим томом с прямым доступом приложению потребуется специальный символьный или блочный файл. При попытке открытия, чтения и записи данных приложение будет обращаться к этому файлу.

Внимание: Каждый логический том содержит управляющий блок (LVCB), расположенный в первом блоке. Размер LVCB равен размеру блока физических томов в группе томов. Данные расположены начиная со второго блока физического тома. В логическом томе с прямым доступом LVCB не защищен. Если приложение перезапишет LVCB, то команды, обычно обновляющие LVCB, будут выдавать сообщение об ошибке. Несмотря на то, что логический том может продолжать нормальную работу и перезапись LVCB является допустимой операцией, делать это не рекомендуется.

В приведенных ниже инструкциях с помощью SMIT и командной строки создается логический том с прямым доступом.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Войдите в систему от имени пользователя root и найдите свободные физические разделы, в которых вы можете создать логический том с прямым доступом:

   ```
   smit lspv
   ```

2. Выберите диск.
3. Примите значение по умолчанию во втором окне (состояние) и нажмите **OK**.
4. Умножьте значение, указанное в поле **Свободные PP** на значение поля **Размер PP**. Таким образом вы получите общий объем пространства доступного на этом диске для создания логического тома с прямым доступом. Если размер свободного пространства недостаточен, то выберите другой диск.
5. Завершите работу SMIT.
6. С помощью команды **mklv** создайте логический том с прямым доступом. Следующая команда создает логический том с прямым доступом `lvdb2003` в группе томов `db2vg`, используя `38` физических разделов по 4 МБ:

   ```
   mklv -y lvdb2003 db2vg 38
   ```

   Флаг **-y** позволяет указать пользовательское имя создаваемого логического тома.

Теперь логический том с прямым доступом создан. Если вы просмотрите содержимое группы томов, то логический том с прямым доступом будет указан с типом по умолчанию jfs. Этот тип записи для логического тома представляет собой просто метку. Он не указывает, что на логическом томе с прямым доступом смонтирована файловая система.

Инструкции по открытию файла `/dev/`*имя* логического тома с прямым доступом см. в документации по приложению.

Понятия, связанные с данным:

“Настройка Администратора логических томов” на стр. 355
Администратор логических томов (LVM) устанавливается вместе с базовой операционной системой и требует дополнительной настройки. Однако прежде, чем LVM сможет работать с дисками, их необходимо настроить и определить в качестве физических томов.

Ссылки, связанные с данной:

“Команды по обслуживанию объектов LVM” на стр. 355
В следующей таблице перечислены простейшие задачи по обслуживанию объектов LVM, к которым относятся физические и логические тома, группы томов и файловые системы.

Информация, связанная с данной:

Команда mklv

Администратор логических томов от А до Я: Введение и общие сведения

Снятие зеркальной защиты с корневой группы томов:

Вы можете снять защиту с корневой группы томов.

Внимание: Отключение зеркальной защиты для rootvg требует наличия опыта администрирования системы. Если эта процедура выполнена некорректно, могут возникнуть трудности с загрузкой системы.

В следующем сценарии корневой группой томов является hdisk01, для нее настроена зеркальная защита для hdisk11. В данном примере будет отключена зеркальная защита для hdisk11. Эта процедура является неизменной, вне зависимости от последнего загруженного диска.

1. Для отключения зеркальной защиты для корневой группы томов в hdisk11 выполните следующие действия:

   ```
   unmirrorvg rootvg hdisk11
   ```

   Команда **unmirrorvg** переключает целостность обратно на корневую группу томов.
2. Для исключения диска из корневой группы томов выполните следующие действия:

   ```
   reducevg rootvg hdisk11
   ```
3. С помощью следующей команды можно выполнить повторную инициализацию загрузочной записи оставшегося диска:

   ```
   bosboot -a -d /dev/hdisk01
   ```
4. С помощью следующей команды можно изменить список загрузки для удаления диска с отключенной зеркальной защитой из списка:

   ```
   bootlist -m normal hdisk01
   ```

Зеркальная защита диска отключена.

Удаление диска без выключения системы:

Приведенная ниже процедура описывает удаление диска с помощью функции оперативного удаления, позволяющей удалять диски без выключения системы. Эта возможность реализована только в некоторых моделях систем.

Функция оперативного удаления полезна в следующих ситуациях:

- Когда необходимо удалить диск с данными из отдельной группы, отличной от rootvg, с целью упрощения защиты или обслуживания.
- Когда необходимо удалить диск из группы томов на постоянной основе.
- Когда необходимо устранить неполадку диска.

Удаление диска без данных:

С помощью данной процедуры можно удалить диск, содержащий данные, без выключения системы.

Удаляемый диск должен входить в отдельную группу, отличную от группы rootvg. Эту процедуру можно применять для перемещения диска в другую систему.

1. Для просмотра информации о группе томов, связанной с удаляемым диском, введите следующую команду:

   ```
   smit lspv
   ```

   Будет показана примерно следующая информация:

   ```
   Физический том:     hdisk2                  Группа томов:      imagesvg
   ИД физ. тома:       00083772caa7896e     ИД VG:     0004234500004c00000000e9b5cac262
   Сост. физ. тома:    активен
   Устар. разделы:     0                        Выделяемый:       да
   Размер PP:          16 мб                    Логические тома:  5
   ```

```
Всего PP:           542 (8672 мб)           Дескрипторы VG:   2
Свободн. PP:        19 (304 мб)             Опер. резерв:     нет
Занятые PP:         523 (8368 мб)
Распр. свободных:   00..00..00..00..19
Распр. занятых:     109..108..108..108..90
```

Имя группы томов указано в поле Группа томов. В этом примере применяется группа томов imagesvg.

2. Для того чтобы убедиться, что диск входит в отдельную группу томов, отличную от rootvg, введите следующую команду:

   ```
   smit lsvg
   ```

   Затем выберите группу томов, связанную с диском (в данном примере - imagesvg). Будет показана примерно следующая информация:

   ```
   Группа томов:   imagesvg                ИД VG:          0004234500004c00000000e9b5cac262
   Сост. VG:       активно                 Размер PP:      16 мб
   Права дост.VG:  чтение/запись           Всего PP:       542 (8672 мб)
   Макс LV:        256                     Свободн. PP:    19 (304 мб)
   LV:             5                       Занят. PP:      523 (8368 мб)
   Откр. LV:       4                       Целостн:        2
   Всего PV:       1                       Дескрипторы VG: 2
   Устар. PV:      0                       Устар PP:       0
   Активн. PV:     1                       Автом. актив.:  да
   Макс. PP на PV: 1016                    Макс. PV:       32
   Разм. LTG:      128 кб                  Автом. синх.:   нет
   Опер. резерв:   нет
   ```

   В нашем примере в поле Всего PV указано, что с imagesvg связан только один физический том. Поскольку все данные этой группы томов хранятся на hdisk2, то эта процедура позволяет удалить hdisk2.

3. Для размонтирования всех файловых систем из логических томов диска введите следующую команду:

   ```
   smit umountfs
   ```

4. Для деактивации группы томов введите следующую команду:

   ```
   smit varyoffvg
   ```

5. Для экспорта группы томов введите следующую команду:

   ```
   smit exportvg
   ```

6. Для удаления диска введите следующую команду:

   ```
   smit rmvdsk
   ```

7. Посмотрите на индикатор удаляемого диска. Убедитесь, что желтый индикатор на корпусе диска не горит.
8. Извлеките диск. Информация о физическом удалении диска приведена в соответствующем разделе руководства по обслуживанию системы.

Теперь диск физически и логически удален из системы. Если вы удаляете диск насовсем, то процедура завершена.

Задачи, связанные с данной:

“Импорт и экспорт группы томов” на стр. 361
В следующей таблице описано применение процедур экспорта и импорта для перемещения пользовательской группы томов из одной системы в другую. (Группу томов rootvg нельзя экспортировать или импортировать.)

“Добавление дисков без выключения системы” на стр. 357
Приведенная ниже процедура описывает включение и настройку диска с помощью функции оперативного удаления, позволяющей добавлять диски без выключения системы.

Удаление диска без данных:

Ниже описана процедура удаления диска без данных или с ненужными данными.

Внимание: Эта процедура удаляет все хранящиеся на диске данные.

1. Для размонтирования всех файловых систем из логических томов диска введите следующую команду:
   ```
   smit umountfs
   ```
2. Для деактивации группы томов введите следующую команду:
   ```
   smit varyoffvg
   ```
3. Для экспорта группы томов введите следующую команду:
   ```
   smit exportvg
   ```
4. Для удаления диска введите следующую команду:
   ```
   smit rmvdsk
   ```
5. Посмотрите на индикатор удаляемого диска. Убедитесь, что желтый индикатор на корпусе диска не горит.
6. Извлеките диск. Информация о физическом удалении диска приведена в соответствующем разделе руководства по обслуживанию системы.

Теперь диск физически и логически удален из системы. Если вы удаляете диск насовсем, то процедура завершена.

Задачи, связанные с данной:

“Добавление дисков без выключения системы” на стр. 357
Приведенная ниже процедура описывает включение и настройку диска с помощью функции оперативного удаления, позволяющей добавлять диски без выключения системы.

Удаление логического тома с помощью удаления файловой системы:

Следующая процедура описывает удаление файловой системы JFS или JFS2, связанного с ней логического тома и соответствующего раздела файла `/etc/filesystems`, а также, дополнительно точки монтирования (каталога), в которой смонтирована файловая система.

Внимание: При удалении файловой системы удаляются все данные, хранящихся в этой файловой системе и на логическом томе.

Если требуется удалить логический том со смонтированной файловой системой другого типа или логический том без файловой системы, можно удалить только логический том.

Для удаления журнализированной файловой системы с помощью SMIT выполните следующие действия:

1. Размонтируйте файловую систему, расположенную на логическом томе, с помощью команды umount:
   ```
   umount /adam/usr/local
   ```

   Примечание: Применять команду **umount** к используемому устройству нельзя. Устройство считается используемым, если на нем находится открытый файл или текущий каталог.
2. Для удаления файловой системы введите следующую команду быстрого доступа:
   ```
   smit rmfs
   ```
3.
1. Выберите имя удаляемой файловой системы.
2. Перейдите в поле **Удалить точку монтирования** и укажите необходимое значение. Если вы выберете опцию **да**, то будет удален и каталог, в котором смонтирована файловая система (если этот каталог пуст).
3. Для удаления файловой системы нажмите Enter. SMIT попросит подтвердить запрос на удаление файловой системы.

4. Подтвердите удаление. После завершения операции SMIT покажет сообщение.

Теперь файловая система удалена вместе с данными и связанным с ней логическим томом.

Задачи, связанные с данной:

“Удаление только логического тома”
С помощью этой поцедуры можно удалить логический том с другим типом смонтированной файловой системы или логический том, не содержащий файловой системы.

Удаление только логического тома:

С помощью этой поцедуры можно удалить логический том с другим типом смонтированной файловой системы или логический том, не содержащий файловой системы.

Внимание: Выполнение этой процедуры приведет к удалению всех данных, хранящихся в удаляемой файловой системе и на логическом томе.

Следующая процедура описывает удаление логического тома и связанных с ним файловых систем. Вы можете воспользоваться данной процедурой для удаления файловой системы, отличной от JFS, а также для удаления логического тома, не содержащего файловых систем. После описания процедуры удаления логического тома ниже описано удаление раздела файла `/etc/filesystems` , относящегося к файловой системе, отличной от JFS.

Для удаления логического тома с помощью SMIT выполните следующие действия:

1. Если на логическом томе нет файловой системы, перейдите к шагу 4.
2. Размонтируйте все файловые системы, связанные с логическим томом:

   `unmount` *`/файловая-система`*

   где */файловая-система* - это полное имя файловой системы.

   Примечание:

   a. Если файловая система, которую вы пытаетесь размонтировать, используется, то команда **unmount** выдаст сообщение об ошибке. Команда **unmount** выполняется лишь в том случае, если в файловой системе нет открытых файлов и на ее устройстве нет текущих пользовательских каталогов.
   b. Другое название команды **unmount** - **umount**. Эти имена равноценны.
3. Для просмотра списка информации о файловых системах введите следующую команду:

   ```
   smit lsfs
   ```

   Ниже приведена часть показанной информации:

   ```
   Имя             Узел       Точка монт.      ...

   /dev/hd3        --         /tmp             ...

   /dev/locallv    --         /adam/usr/local  ...
   ```
4. Полагая, что для второго элемента соблюдаются стандартные соглашения о присвоении имен, можно определить, что `/adam/usr/local` - это имя файловой системы, а `locallv` - имя логического тома. Для проверки этого предположения введите следующую команду быстрого доступа:

   ```
   smit lslv2
   ```

   Ниже приведена часть показанной информации:

   ```
   imagesvg:
   Имя LV             Тип        LP    PP    PV   Сост. LV      Точка монтир.
   hd3                jfs        4     4     1    open/syncd    /tmp
   locallv            mine       4     4     1    closed/syncd  /adam/usr/local
   ```
5. Для удаления логического тома введите следующую команду:

   ```
   smit rmlv
   ```

6. Выберите имя удаляемого логического тома.
7. Перейдите в поле **Удалить точку монтирования** и укажите необходимое значение. Если вы выберете опцию **да**, то будет удален и каталог, в котором смонтирована файловая система (если этот каталог пуст).
8. Для удаления логического тома нажмите Enter. SMIT попросит подтвердить запрос на удаление логического тома.
9. Подтвердите удаление. После завершения операции SMIT покажет сообщение.
10. Если на логическом томе располагалась файловая система, отличная от JFS, удалите запись об этой файловой системе из файла `/etc/filesystems` с помощью следующей команды:

    ```
    rmfs /adam/usr/local
    ```

    Вы также можете указать имя файловой системы следующим образом:

    ```
    rmfs /dev/locallv
    ```

Теперь логический том удален. Если логический том содержал файловую систему, отличную от JFS, то необходимо удалить соответствующий раздел из файла `/etc/filesystems`

Задачи, связанные с данной:

“Удаление логического тома с помощью удаления файловой системы” на стр. 400
Следующая процедура описывает удаление файловой системы JFS или JFS2, связанного с ней логического тома и соответствующего раздела файла `/etc/filesystems`, а также, дополнительно точки монтирования (каталога), в которой смонтирована файловая система.

Изменение размера группы томов RAID:

В системах, использующих избыточный массив независимых дисков (RAID), предусмотрены опции команд **chvg** и **chpv**, позволяющие добавлять диск в группу RAID и увеличивать размер физического тома, применяемого LVM, без прерывания работы системы.

Примечание:

1. Данная функция недоступна, если группа томов активирована в обычном или расширенном параллельном режиме.
2. Данная процедура не позволяет изменить размер группы томов rootvg.
3. Данная процедура не позволяет изменить размер группы томов с активным пространством подкачки.

При активации (подключении) группы томов автоматически проверяется размер всех дисков в этой группе томов. Если размер увеличен, то система выдает информационное сообщение.

Ниже описана процедура увеличения размера группы томов в среде RAID:

1. Для проверки размера дисков и, при необходимости, изменения размера, введите следующую команду:

   ```
   chvg -g группа-томов
   ```

   где *группа-томов* - это имя группы томов. Данная команда проверяет все диски в группе томов. Если размер какого-либо диска увеличился, то она пытается добавить в физический том физические разделы. При необходимости команда определит необходимый множитель для ограничения 1016 и преобразует обычную группу томов в большую.
2. Для выключения в LVM перемещения поврежденных кластеров в группе томов введите следующую команду:

   ```
   chvg -b ny группа-томов
   ```

   где *группа-томов* - это имя группы томов.

Стратегии группы томов

Сбой диска - наиболее распространенная аппаратная ошибка системы хранения данных, вызываемая отказами адаптера или источника питания. Основной способ защиты от сбоев диска - создание логических томов.

Для защиты от отказов адаптера и источника питания необходимо создать для группы томов специальную аппаратную конфигурацию. Такая конфигурация включает два адаптера, работающих в режиме зеркального копирования, и по крайней мере по одному диску для каждого из адаптеров, на базе которых создается группа томов без контроля целостности. Однако высокая стоимость делает эту конфигурацию неприемлемой для обычных систем. Ее рекомендуется применять только для обеспечения особенно высокой надежности, когда доступ к данным после сбоя восстанавливается в течение нескольких секунд. Отказоустойчивая конфигурация позволяет снизить потери от аппаратных сбоев. Однако она не снижает вероятность случайного удаления файлов.

Понятия, связанные с данным:

“Стратегия логического тома” на стр. 405
Приведенная здесь информация поможет вам разработать стратегию использования логического тома, которая обеспечивала бы оптимальную надежность, производительность при приемлемом уровне затрат.

Зачем нужно создавать отдельные группы томов:

В некоторых случаях может потребоваться сгруппировать физические тома в группах томов, отличных от rootvg.

- Упрощение обслуживания.
  - Операции обновления, повторной установки и восстановления после сбоев становятся проще и безопаснее, поскольку вы можете отделить пользовательские файловые системы от файлов операционной системы и не подвергать риску пользовательские данные во время выполнения этих операций.
  - Вы можете обновлять или переустанавливать операционную систему без сохранения и восстановления пользовательских данных. Например, перед обновлением системы вы можете удалить пользовательскую группу томов из системы, размонтировав ее файловые системы. Отключите группу томов командой **varyoffvg**, а затем экспортируйте ее командой **exportvg**. После обновления системы вы можете снова импортировать пользовательскую группу томов командой **importvg** и смонтировать все ее файловые системы.
- Возможность выбора разных размеров физических разделов. Все физические тома в одной группы томов должны иметь одинаковый размер физических разделов. Физические тома с разным размером физических разделов должны находиться в различных группах томов.
- Возможность выбора разных параметров контроля целостности. Для создания файловой системы без контроля целостности поместите ее в отдельную группу томов; все остальные файловые системы могут оставаться в обычной группе томов с контролем целостности.
- Защита. Например, вы можете вынимать на ночь диски из компьютера.
- Возможность перемещения физических томов между системами. Если вы создадите отдельную группу томов для каждой системы, работающей с общим адаптером, то можно будет переключать физические тома между системами, не прерывая обычную работу систем (см. описание команд **varyoffvg**, **exportvg**, **importvg** и **varyonvg**).

Высокая готовность на случай сбоя диска:

Основной способ защиты от сбоев дисков заключается в особой настройке логических томов, например, создании зеркальных копий.

Хотя задача настройки групп томов с точки зрения защиты является вторичной, она важна с экономической точки зрения, поскольку от выбранной конфигурации зависит число физических томов в группе:

- В конфигурации с контролем целостности, применяемой по умолчанию, группа томов остается активной до тех пор, пока работает по крайней мере 51% входящих в нее дисков. В большинстве случаев для защиты от сбоев дисков в группе томов должно быть по крайней мере три диска с зеркальными копиями.
- В конфигурации без контроля целостности группа томов остается активной, пока на диске есть по крайней мере одна доступная область VGDA. Для защиты от сбоев дисков в такой группе томов необходимо два диска с зеркальными копиями.

При определении числа дисков в каждой группе томов необходимо учитывать пространство, которое будет занято зеркальными копиями данных. Помните, что вы можете создавать зеркальные копии и перемещать данные только между дисками, входящими в одну группу томов. Если на вашем компьютере есть большие файловые системы, то поиск дискового пространства для хранения зеркальных копий может оказаться непростой задачей. Ознакомьтесь с дополнительной информацией о настройке параметров дисков, содержащих копии логического тома, и размещении данных на диске.

Понятия, связанные с данным:

“Процесс включения” на стр. 351
Процесс включения с помощью которого LVM проверяет пригодность группы томов к использованию и сохраняет на носителях этой группы текущей версии данных.

“Отключение контроля целостности для группы томов” на стр. 354
Вы можете отключить для группы томов контроль целостности, обеспечив постоянный доступ к данным даже в случае нарушения целостности.

“Стратегия распределения для копий логического тома” на стр. 410
Размещение на диске единственной копии логического тома - довольно простая операция.

“Стратегии размещения данных на диске для логических томов” на стр. 411
При выборе стратегии размещения данных на диске рассматриваются пять областей диска, в которых могут располагаться физические разделы.

Высокая готовность в случае сбоя адаптера или блока питания:

Для защиты от отказов адаптера и источника питания выполните одно или несколько из перечисленных ниже действий, в зависимости от требуемого уровня надежности.

- Установите два адаптера, расположив их в одном или разных блоках. Второй вариант позволяет в случае отказа источника питания одного из блоков сохранить работоспособность второго адаптера.
- Подключите к каждому из адаптеров по крайней мере один диск. Если вы активизируете *перекрестное зеркальное копирование* (т.е. разместите копии логического раздела на разных физических томах) для логических томов диска A (подключенного к адаптеру A) и диска B (подключенного к адаптеру B), то такая конфигурация защитит группу томов от потери данных в случае сбоя одного из адаптеров или его источника питания. Для организации перекрестного зеркального копирования скопируйте логические тома дисков, подключенных к адаптеру A, на диски адаптера B, а логические тома адаптера B - на диски адаптера A.
- Объедините диски, подключенные к обоим адаптерам, в одну группу томов. При этом в случае отказа одного из адаптеров (или сбоя одного из источников питания) всегда будет доступна по крайней мере одна копия логического тома.
- Создайте группу томов без контроля целостности. Такая группа томов остается активной до тех пор, пока доступна хотя бы одна область дескрипторов группы томов (VGDA).
- Если группа томов состоит из двух дисков, то настройте перекрестное резервное копирование между адаптерами. Если к каждому из адаптеров подключено несколько дисков, используйте двойное резервное копирование. В этом случае одну зеркальную копию следует создать на диске, подключенном к тому же адаптеру, что и диск с исходным экземпляром данных, а другую - на диске, подключенном к другому адаптеру.

Понятия, связанные с данным:

“Отключение контроля целостности для группы томов” на стр. 354
Вы можете отключить для группы томов контроль целостности, обеспечив постоянный доступ к данным даже в случае нарушения целостности.

Стратегия логического тома

Приведенная здесь информация поможет вам разработать стратегию использования логического тома, которая обеспечивала бы оптимальную надежность, производительность при приемлемом уровне затрат.

Готовность - это свойство системы, при котором доступ к данным возможен даже в том случае, когда диск, на котором они хранятся, поврежден или недоступен. Обеспечение доступа к данным гарантируется с помощью копий, хранящихся на отдельных дисках; создание и обслуживание этих копий выполняется в процессе нормальной работы системы. Для обеспечения высокого коэффициента готовности применяются такие технологии, как зеркальное копирование и диски с возможностью оперативной замены.

Производительность - это средняя скорость доступа к данным. Применение в стратегии таких компонентов, как проверка записи и зеркальное копирование повышает надежность за счет дополнительной нагрузки на системные ресурсы, что приводит к снижению производительности. Применение зеркального копирования удваивает или даже утраивает фактический размер логического тома. Как правило, повышение надежности приводит к снижению производительности. Чередование данных на диске позволяет повысить производительность. Допускается одновременное применение чередования данных и зеркальной защиты. Можно выполнять поиск и устранение неполадок активных областей в случаях, когда количество ошибок ввода-вывода на некоторых логических разделах начинает влиять на производительность системы.

Управляя размещением данных на диске и их распределением между различными дисками, можно добиться максимальной производительности подсистемы памяти. Дополнительная информация о повышении производительности подсистемы управления памятью приведена в книге *Руководство по настройке производительности*.

Приведенная ниже информация поможет вам сбалансировать производительность, надежность и затраты. Следует иметь в виду, что повышение надежности приводит к снижению производительности. Однако при зеркальном копировании возможно и повышение производительности, если при чтении данных LVM всегда будет выбирать наименее загруженный диск.

> **Примечание:** Зеркальное копирование не защищает от потери отдельных файлов из-за случайного удаления или сбоев программного обеспечения. Эти файлы можно восстановить только с помощью обычных резервных копий на ленте или на дисках.

Понятия, связанные с данным:

“Стратегии группы томов” на стр. 403
Сбой диска - наиболее распространенная аппаратная ошибка системы хранения данных, вызываемая отказами адаптера или источника питания. Основной способ защиты от сбоев диска - создание логических томов.

Требования к зеркальной защите и чередованию:

Определите, являются ли данные, которые будут храниться на логическом томе, настолько ценными, чтобы тратить ресурсы процессора и дисковое пространство на поддержку зеркальных копий. При наличии файловой системы с последовательным доступом, для которой важно обеспечить высокую производительность, рекомендуется применять чередование данных на диске.

Высокая производительность и применение зеркального копирования не всегда исключают друг друга. Когда различные экземпляры (копии) логических разделов расположены на разных физических томах, которые, по возможности, должны быть подключены к различным адаптерам, LVM способен повысить скорость чтения, обращаясь за данными к наименее загруженному диску. Если диски подключены к одному адаптеру, то скорость выполнения операций записи не меняется, поскольку необходимо обновление всех копий данных. Однако, для чтения информации можно обратиться к любой из копий.

AIX LVM поддерживает следующие опции RAID:

Таблица 63. Поддержка RAID Администратором логических томов

Элемент	Описание
RAID 0	Чередование
RAID 1	Зеркальная защита
RAID 10 или 0+1	Зеркальная защита и чередование

Хотя применение зеркального копирования повышает надежность системы управления памятью, при этом не рекомендуется отказываться от обычных средств резервного копирования.

При зеркальной защите rootvg создайте отдельный логический том для дампа. Не указывайте логический том с зеркальным копированием в качестве устройства для дампа - он может быть записан неправильно. По умолчанию дамп создается в основном логическом томе подкачки, но, если для него включено зеркальное копирование, необходимо указать для дампа другой том.

Как правило, при изменении данных на логическом разделе автоматически обновляются все физические разделы, на которых хранятся данные этого логического раздела. Однако, данные физического раздела могут *устареть* из-за сбоев или недоступности физического тома во время изменения данных. LVM может обновить устаревшие данные, скопировав данные из физического раздела с текущими данными. Этот процесс называется *синхронизацией зеркальных копий*. Обновление может выполняться в момент запуска системы, во время активации физического тома или при обработке команды **syncvg**.

При изменении размещения загрузочного логического тома на физических разделах необходимо запустить команду **bosboot**. Это означает, что команду **bosboot** необходимо запускать при каждом изменении параметров зеркального копирования для загрузочного логического тома.

Стратегии планирования для зеркально защищенной записи на диск:

Если данные существуют в единственном экземпляре, то драйвер логических томов (LVDD) преобразует логический адрес, переданный в запросе на чтение или запись, в физический адрес и вызывает для обработки запроса соответствующий драйвер физического устройства. Стратегия одной копии, т.е. без применения зеркального копирования, предполагает перемещение неисправных блоков при обработке запросов на запись и возврат кодов ошибок при обработке запросов на чтение.

Для логического тома с зеркальным копированием существует следующие стратегии записи данных на диск:

Последовательная запись
: Запись различных копий выполняется последовательно. Физические разделы, соответствующие различным зеркальным копиям одного логического раздела, имеют номера: первый, второй и третий. При последовательном планировании физические разделы записываются в последовательность. Система не начинает запись в очередной физический раздел до окончания записи в предыдущий. Операция записи завершается только после завершения передачи данных во все физические разделы.

Параллельная запись
: Запись всех копий выполняется параллельно. Операция записи завершается только после завершения передачи данных во все физические разделы. Чтение также выполняется параллельно, с использованием наименее загруженного диска, что увеличивает производительность.

Параллельная запись с последовательным чтением
: Запись всех копий выполняется параллельно. При чтении сначала выполняется обращение к первой копии. При возникновении ошибки LVM пытается прочитать следующую копию. Затем LVM восстанавливает первую копию путем перераспределения физических блоков. После этого блок готов к дальнейшему использованию.

Параллельная запись с карусельным чтением
Запись всех копий выполняется параллельно. Чтение выполняется последовательно с разных зеркальных копий.

Стратегия перемещения поврежденных блоков
Указывает, разрешено ли в группе томов перемещать поврежденные блоки. Значение по умолчанию - *yes*. Когда для группы томов указано значение `yes`, поврежденные блоки перемещаются. Когда значение равно `no`, стратегия переопределяет параметры отдельных логических томов. После изменения этого значения снова станут применяться параметры, заданные для отдельных логических томов. Это значение указывает, следует ли перенаправлять операции ввода-вывода в перемещенный блок. Если значение равно `yes`, то в группе томов допускается перемещение поврежденного блока. Если значение равно `no`, то поврежденный блок не перемещается. Программное перемещение выполняется только в том случае, если аппаратное перемещение выполнить не удалось. В противном случае флаг перемещения поврежденного блока (BBR) неприменим.

Примечание: Перемещение поврежденного блока выполняется только в том случае, если стратегия перемещения поврежденных блоков включена как для группы томов, так и для логического тома.

Стратегия согласованности зеркальной записи для логического тома:

Если согласование зеркальных копий при записи (MWC) включено, то выполняется идентификация логических разделов, которые могут оказаться несогласованными из-за некорректного завершения работы системы или группы томов. При включении группы томов данная информация применяется для их согласования. Этот режим также называется *активным MWC*.

Если согласование MWC включено для логического тома, запросы к логическому тому блокируются до обновления кэша MWC целевых физических томов. После обновления кэш-памяти MWC запись данных продолжается. Обновление блоков кэша MWC выполняется только для тех дисков, на которые будут записаны данные.

MWC отрицательно влияет на производительность системы. Это связано с необходимостью регистрировать те запросы на запись, в которых активна Группа логического отслеживания (LTG). Для группы томов допустимы следующие размеры LTG: 128 КБ, 256 КБ, 512 КБ, 1024 КБ, 2 МБ, 4 МБ, 8 МБ и 16 МБ.

Примечание: Для работы с LTG размером больше 128 КБ диски группы томов должны поддерживать прием запросов ввода-вывода такого размера от процедур работы с диском. LTG - это единый блок данных на логическом томе, выровненный по размеру LTG. Снижение производительности происходит только при записи на тома с зеркальной защитой.

Гарантировать совпадение зеркальных копий необходимо только в случае сбоя, произошедшего до завершения записи всех зеркальных копий. Для всех логических томов в группе ведется один протокол MWC. Протокол MWC расположен на внешнем краю диска. Логические тома с MWC следует располагать как можно ближе внешнему краю.

Если включен Пассивный режим MWC выключено, в протоколе сохраняется только сообщение об открытии группы томов. При включении открытой группы томов после сбоя автоматически запускается синхронизация. При синхронизации блоки данных, считанные с учетом стратегии чтения, записываются в остальные зеркальные копии логического тома. Данная стратегия поддерживается только для больших групп томов.

Если согласование MWC выключено, то после сбоя системы или группы томов копии логического тома могут остаться несогласованными. Согласованность данных копий не обеспечивается автоматически. Сбой в момент записи данных может привести к тому, что при следующем подключении группы томов данные зеркальных копий будут различаться. Если для логического тома применяется зеркальное копирование, но стратегия MWC выключена, то для дальнейшей работы с этим томом необходима синхронизация. Например, введите:

```
syncvg -f -l LTV имя
```

Исключение составляют логические тома, содержимое которых действительно, только пока они открыты. Например, пространство подкачки.

С точки зрения записи логический том, для которого применяется зеркальное копирование, практически не отличается от обычного. LVM завершает запрос на запись только после полной передачи данных на более низкий уровень. Однако, до возврата сигнала **iodone** результат записи неизвестен. Если такой сигнал получен (запись завершена), восстановление не требуется. Все блоки, запись которых не была завершена (получен сигнал **iodone**) в момент сбоя должны быть проверены и перезаписаны, независимо режима MWC.

Поскольку логический том с зеркальной защитой с точки зрения системы не отличается от обычного, определить, какие именно данные были записаны, невозможно. Для обеспечения корректности данных любые приложения должны проверять, были ли выполнены все запрошенные перед сбоем операции записи.

Активная и пассивная стратегии MWC лишь обеспечивают идентичность зеркальных копий при подключении группы томов после сбоя, копируя данные из одной копии во все остальные. Эти стратегии MWC не отслеживают последние записанные данные. Активная стратегия MWC отслеживает только последнюю запись в LTG; MWC не обеспечивает восстановление данных. Пассивная стратегия обеспечивает согласование различных зеркальных копий путем перехода после сбоя в режим копирования при чтении. Правильность данных после сбоя должны определять приложения более высокого уровня, чем LVM. С точки зрения LVM, если приложение повторит все запросы на запись, которые обрабатывались в момент сбоя, то после выполнения этих запросов целостность данных будет восстановлена (при условии, что выполняется запись тех же блоков, запись которых не была завершена к моменту сбоя).

Примечание: Логические тома с зеркальным копированием, содержащие протоколы JFS или файловые системы, после сбоя должны быть синхронизированы принудительно, либо путем включения активной или пассивной стратегии MWC.

Стратегии распределения данных по дискам:

Стратегия распределения данных по дискам определяет число дисков, на которых будут расположены физические разделы логического тома.

Физические разделы для логического тома могут быть выделены на одном диске или быть распределены между всеми дисками группы томов. Стратегия распределения данных задается следующими параметрами команд **mklv** и **chlv**:

- Параметр **Range** задает число дисков, применяемых для размещения одной физической копии логического тома.
- Параметр **Strict** отменяет операцию **mklv**, если на одном физическом томе должно располагаться несколько копий.
- Параметр **Super Strict** указывает, что разделы одной зеркальной копии не могут находиться на одном физическом томе с разделами другой зеркальной копии.
- Для логических томов с чередованием данных можно выбрать только стратегию распределения с максимальным числом дисков и включенную опцию Super Strict.

Понятия, связанные с данным:

“Перемещение и уменьшение пространства подкачки hd6” на стр. 421
Уменьшение размера или перемещение пространства подкачки по умолчанию может потребоваться для повышения производительности системы путем переноса пространства подкачки на диски с меньшим уровнем нагрузки. Сокращение размера или перемещение подкачки по умолчанию также позволяет сохранить свободное пространство над диске hdisk0.

Стратегия распределения для одной копии логического тома:

Если выбрана стратегия распределения с минимальным числом дисков (`Range = minimum`), то все физические разделы одного логического тома будут расположены на одном диске для улучшения доступности. Если выбрана стратегия распределения с максимальным числом дисков (`Range = maximum`), то физические разделы будут расположены на различных дисках для повышения производительности.

Для максимальной надежности логических томов без зеркальной защиты установите значение minimum. Значение minimum указывает, что все исходные физические разделы логического тома будут по возможности располагаться на одном физическом томе. Если разместить все физические разделы на одном физическом томе оказывается невозможным, используется минимальное число томов, совместимое с другими параметрами.

Использование минимального числа физических томов снижает вероятность утраты данных в случае сбоя диска. Каждый дополнительный физический том, использованный для хранения копии данных, увеличивает эту вероятность. Если зеркальное копирование не применяется, то для логического тома, расположенного на четырех физических томах, вероятность потери данных из-за сбоя одного тома в четыре раза больше, чем для логического тома, расположенного на одном физическом томе.

На следующем рисунке показано действие стратегии распределения с минимальным числом дисков.

Рисунок 3. Стратегия распределения с минимальным числом дисков

На рисунке показано три диска. На одном диске - три физических раздела; на других физических разделов нет.

Если указано значение maximum, физические разделы будут распределены по физическому тому наиболее равномерным образом (с учетом других ограничений). Эта опция позволяет повысить производительность, так как размещение физических разделов на нескольких дисках сокращает среднее время доступа к логическому тому. Для поддержания приемлемой надежности значение maximum рекомендуется использовать только с зеркальной защитой.

На следующем рисунке показано действие стратегии распределения с максимальным числом дисков.

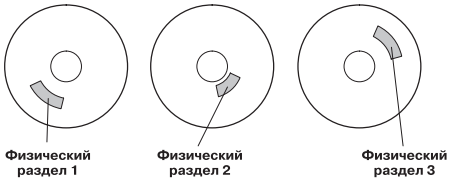


Рисунок 4. Стратегия распределения с максимальным числом дисков

На схеме показано три диска, по три физических раздела на каждом.

Приведенная информация применима и в случае расширения или копирования существующего логического тома. Распределение новых физических разделов зависит от текущей стратегии распределения и от расположения занятых физических разделов.

Понятия, связанные с данным:

“Стратегия распределения для копий логического тома”
Размещение на диске единственной копии логического тома - довольно простая операция.

Стратегия распределения для копий логического тома:

Размещение на диске единственной копии логического тома - довольно простая операция.

Однако при создании зеркальных копий задача усложняется. На следующих рисунках показан результат применения стратегии распределения с минимальным числом дисков и различными значениями параметра Strict.

При такой стратегии физические разделы первого экземпляра логического тома будут размещены на одном физическом томе (если это возможно). Будут ли дополнительные диски размещены на том же диске или на других зависит от значения параметра Strict. Другими словами, для размещения физических разделов используется минимальное число физических томов с учетом ограничений, налагаемых значениями других параметров, в том числе опции Strict.

Параметр `Strict = y` означает, что каждая копия логического тома должна находиться на отдельном физическом томе. `Strict = n` разрешает размещение копий на одном физическом томе. Параметр `Super Strict` не позволяет физическим разделам одной зеркальной копии находиться на том же диске, что и физические разделы другой зеркальной копии того же логического тома.

Примечание: Если число физических томов в группе меньше числа копий логического раздела, то параметру Strict необходимо присвоить значение **n**. Если число физических томов в группе меньше числа копий логического раздела, то параметру Strict необходимо присвоить значение **y**.

На следующих рисунках показано действие стратегии распределения с минимальным числом дисков и различными значениями параметра Strict.

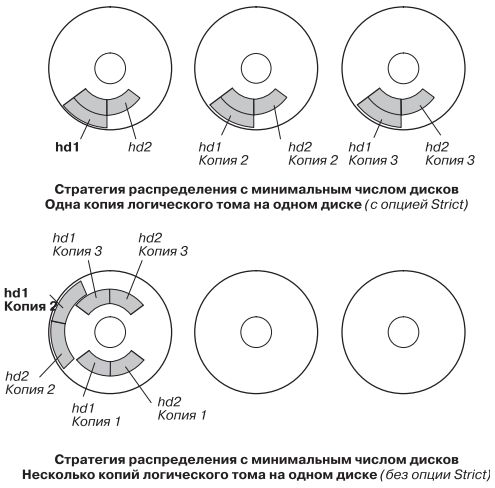


Рисунок 5. Стратегия распределения с минимальным числом дисков/Strict. Как показано на рисунке, если параметр Strict включен, то все копии логического раздела будут расположены на разных физических томах. Если параметр Strict выключен, все копии каждого логического раздела будут расположены на одном физическом томе.

На следующих рисунках показано действие стратегии распределения с максимальным числом дисков и различными значениями параметра *Strict*:

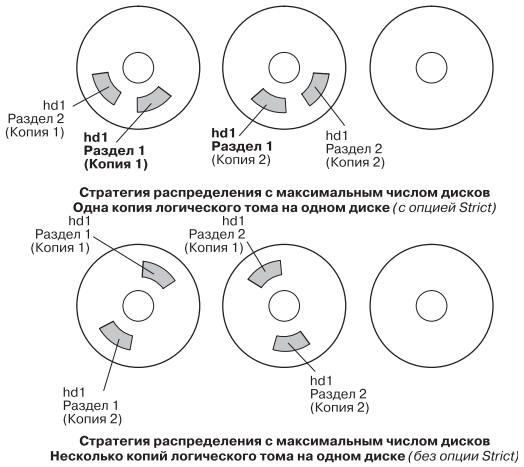


Рисунок 6. Стратегия распределения с максимальным числом дисков/Strict. Как показано на рисунке, если параметр Strict включен, то все копии раздела будут расположены на разных физических томах. Если параметр Strict выключен, все копии будут расположены на одном физическом томе.

Понятия, связанные с данным:

"Высокая готовность на случай сбоя диска" на стр. 403
Основной способ защиты от сбоев дисков заключается в особой настройке логических томов, например, создании зеркальных копий.

"Стратегия распределения для одной копии логического тома" на стр. 409
Если выбрана стратегия распределения с минимальным числом дисков (`Range = minimum`), то все физические разделы одного логического тома будут расположены на одном диске для улучшения доступности. Если выбрана стратегия распределения с максимальным числом дисков (`Range = maximum`), то физические разделы будут расположены на различных дисках для повышения производительности.

Стратегии размещения данных на диске для логических томов:

При выборе стратегии размещения данных на диске рассматриваются пять областей диска, в которых могут располагаться физические разделы.

Чем ближе физический раздел расположен к центру физического тома, тем меньше среднее время поиска, так как среднее расстояние при поиске между центром и любой другой областью диска минимально.

Поскольку чаще всего используется протокол файловой системы, то его можно разместить в центральной части физического тома. Реже всего используется загрузочный логический том, поэтому его лучше разместить ближе к краю физического тома.

Общее правило: чем больше частота операций ввода-вывода в логический том, тем ближе должны быть расположены его физические разделы к центру физического тома.

Из этого правила есть одно важное исключение: Логические тома с зеркальной защитой и включенным параметром MWC должны располагаться около внешнего края диска, так как именно там хранятся данные MWC. Если зеркальное копирование не используется, то MWC также не применяется и, следовательно, не влияет на производительность.

К пяти областям диска, в которых могут быть размещены физические разделы, относятся:

1. внешний край
2. внутренний край
3. внешняя область
4. внутренняя область

5. средняя область

Краевым разделам соответствует наибольшее среднее время поиска, что, как правило, означает большее время ответа для приложений, которые размещают свои данные на этом участке. Разделам в средней области соответствует минимальное среднее время поиска, что, как правило, означает наименьшее время ответа. Однако в средней области диска расположено меньшее число разделов, чем на других участках.

Понятия, связанные с данным:

“Высокая готовность на случай сбоя диска” на стр. 403
Основной способ защиты от сбоев дисков заключается в особой настройке логических томов, например, создании зеркальных копий.

Сочетание стратегий выделения ресурсов:

Если стратегии распределения данных между дисками и стратегия размещения данных на диске несовместимы, то результат будет непредсказуем.

Физические разделы будут распределены в соответствии с одной из выбранных стратегий. Например, если вы выберите размещение данных в средней части диска и использование минимального числа дисков, то приоритетной будет считаться стратегия распределения данных между дисками. Если это возможно, система поместит все разделы, соответствующие логическому тому, на одном диске, даже если не все разделы будут находиться в средней области диска. Перед реализацией стратегий необходимо изучить возможные последствия их совместного применения.

Применение файлов размещения для точного выделения ресурсов:

Если предлагаемые стратегии распределения данных между дисками и размещения данных на диске неприемлемы, то вы можете воспользоваться файлами размещения, которые позволяют точно задать порядок и расположение физических разделов для логического тома.

Для создания файлов размещения воспользуйтесь SMIT или командой **mklv -m**.

Например, для того чтобы создать в группе томов rootvg логический том lv06 с десятью разделами, которому соответствовали бы разделы с 1 по 3, с 41 по 45 и с 50 по 60 на диске hdisk1, необходимо выполнить следующую процедуру:

1. Убедитесь, что данные разделы не заняты:
   ```
   lspv -p hdisk1
   ```
2. Создайте файл, например `/tmp/mymap1`, содержащий следующие строки:
   ```
   hdisk1:1-3
   hdisk1:41-45
   hdisk1:50-60
   ```
   Команда **mklv** размещает физические разделы в том порядке, в котором они указаны в файле размещения. В файле размещения должно быть задано достаточно физических разделов для размещения всего логического тома. (Можно перечислить большее число разделов).
3. Вызовите команду:
   ```
   mklv -t jfs -y lv06 -m /tmp/mymap1 rootvg 10
   ```

Разработка стратегий логических томов с чередованием данных:

Логические тома с чередованием данных применяются для файловых систем, в которых хранятся часто используемые последовательные файлы большого размера, оказывающие существенное влияние на производительность. Чередование данных позволяет повысить производительность.

Примечание: Для дампа и загрузочных логических томов чередование данных не применяется. Загрузочный логический том должен состоять из последовательных физических разделов.

Для создания в группе VGName логического тома lv07 с 12 разделами и блоком чередования (размер блока чередования равен размеру раздела, умноженному на число дисков) в 16 КБ на дисках hdisk1, hdisk2 и hdisk3 введите следующую команду:

```
mklv -y lv07 -S 16K VGName 12 hdisk1 hdisk2 hdisk3
```

Для создания на любых трех дисках группы VGName логического тома lv08 с 12 разделами и блоком чередования размером 8 КБ введите следующую команду:

```
mklv -y lv08 -S 8K -u 3 VGName 12
```

Дополнительная информация о повышении производительности с помощью технологии чередования данных приведена в книге *Руководство по настройке производительности*.

Стратегия проверки записи:

Опция проверки записи позволяет проверять правильность каждой выполненной операции записи путем немедленного чтения записанных данных.

При обнаружении ошибки записи будет выдано соответствующее сообщение. Данная стратегия повышает надежность, но снижает производительность, так как на чтение затрачивается дополнительное время. Проверка записи для логического тома может быть включена при его создании командой **mklv**, либо позже командой **chlv**.

Стратегии оперативной замены диска:

Для группы томов с зеркальными логическими томами можно выбрать диски с возможностью оперативной замены.

Вы также можете задать стратегию замены дисков при сбое и параметры синхронизации данных.

Добавляемый в группу томов физический том с возможностью оперативной замены не должен быть меньше самого маленького диска в группе томов. Если в результате ошибок записи MWC какой-либо физический том будет помечен как отсутствующий, то все его данные будут перенесены на добавленный диск.

Команды **chvg** и **chpv**, управляющие возможностью оперативной замены, имеют следующие опции:

```
chvg -hстратегия_оперативной_замены
-sстратегия_синхронизации
группа_томов
```

где *стратегия_оперативной_замены* задает одну из следующих стратегий:

y Неисправные разделы будут автоматически перенесены на запасной диск. Из пула запасных дисков выбирается наименьший по объему диск, способный заменить поврежденный.

+ Неисправные разделы будут автоматически перенесены на запасной диск, при этом может быть задействован весь пул запасных дисков.

n Разделы не будут перенесены автоматически (по умолчанию).

r Удаляет все диски из пула дисков с возможностью оперативной замены для этой группы томов.

Параметр *стратегия_синхронизации* указывает, нужно ли автоматически синхронизировать разделы:

y Попытаться автоматически синхронизировать разделы.

n Не пытаться автоматически синхронизировать разделы. (По умолчанию.)

В параметре *группа_томов* указывается имя группы томов с зеркальной защитой.

Управление оперативной заменой в логических томах:

Некоторые неполадки логических томов можно устранить путем *оперативной замены*, без прерывания работы системы.

Такие неполадки возникают в случаях, когда количество ошибок ввода-вывода на некоторых логических разделах начинает влиять на производительность системы.

Первый шаг в решении любой проблемы - это ее определение. По умолчанию, система не собирает статистическую информацию об использовании логических томов. Если сбор такой информации включен, то при первом вызове команды **lvmstat** будут показаны статистические значения с момента предыдущей перезагрузки. При последующих вызовах команды **lvmstat** будет показана статистика, накопленная с момента предыдущего вызова этой команды.

Вывод команды **lvmstat** позволяет найти наиболее загруженные логические разделы. Если на одном физическом диске есть несколько сильно загруженных логических разделов, то вы можете перенести некоторые из них на другие диски с помощью команды **migratelp**.

В следующем примере сбор статистики включен и данные собираются путем последовательных вызовов команды **lvmstat**:

```
# lvmstat -v rootvg -e
# lvmstat -v rootvg -C
# lvmstat -v rootvg
```

Будет показана примерно следующая информация:

```
Логический том            iocnt      Kb_read     Kb_wrtn      КБ/с
  hd8                         4            0          16      0.00
  paging01                    0            0           0      0.00
  lv01                        0            0           0      0.00
  hd1                         0            0           0      0.00
  hd3                         0            0           0      0.00
  hd9var                      0            0           0      0.00
  hd2                         0            0           0      0.00
  hd4                         0            0           0      0.00
  hd6                         0            0           0      0.00
  hd5                         0            0           0      0.00
```

В предыдущем примере все счетчики были сброшены. Затем данные копируются из каталога `/unix` в каталог `/tmp`. Команда **lvmstat** отражает эти действия для группы томов rootvg:

```
# cp -p /unix /tmp
# lvmstat -v rootvg

Логический том            iocnt      Kb_read     Kb_wrtn      КБ/с
  hd3                       296            0        6916      0.04
  hd8                        47            0         188      0.00
  hd4                        29            0         128      0.00
  hd2                        16            0          72      0.00
  paging01                    0            0           0      0.00
  lv01                        0            0           0      0.00
  hd1                         0            0           0      0.00
  hd9var                      0            0           0      0.00
  hd6                         0            0           0      0.00
  hd5                         0            0           0      0.00
```

Вывод показывает занятость логического тома **hd3**, смонтированного в каталоге `/tmp` , тома **hd8**, являющегося логическим томом протокола JFS, корневого тома **hd4** (`/`), тома **hd2**, представляющего каталог `/usr`, и тома **hd9var**, представляющего каталог `/var`. Ниже показана подробная информация для **hd3** и **hd2**:

```
# lvmstat -l hd3

Log_part    mirror#    iocnt    Kb_read    Kb_wrtn     КБ/с
       1          1      299          0       6896     0.04
       3          1        4          0         52     0.00
       2          1        0          0          0     0.00
       4          1        0          0          0     0.00
# lvmstat -l hd2
Log_part    mirror#    iocnt    Kb_read    Kb_wrtn     Кб/с
       2          1        9          0         52     0.00
       3          1        9          0         36     0.00
       7          1        9          0         36     0.00
       4          1        4          0         16     0.00
       9          1        1          0          4     0.00
      14          1        1          0          4     0.00
       1          1        0          0          0     0.00
```

Вывод для группы томов содержит всю информацию об операциях ввода-вывода на логическом томе. Указывается число запросов ввода-вывода (`iocnt`), число прочитанных и записанных килобайт (`Kb_read` и `Kb_wrtn`), а также скорость передачи данных в килобайтах в секунду (`Kbps`). Эту информацию можно получить для каждого логического раздела на логическом томе в отдельности. Если применяются зеркальные копии, то статистика будет показана для каждого зеркального тома. В предыдущем примере некоторые строки, не представляющие интереса, были пропущены. Вывод всегда сортируется по убыванию значений в столбце iocnt.

Параметрами команды **migratelp** являются имя логического тома, номер логического раздела (как показано в выводе **lvmstat**) и необязательный номер конкретной зеркальной копии. Если последнее значение не указано, то применяется первая зеркальная копия. Для перемещения можно указать больший целевой физический том и номер целевого физического раздела. При успешном выполнении будет показана примерно следующая информация:

```
# migratelp hd3/1 hdisk1/109
  migratelp: Зеркальная копия 1 логического раздела 1 логического тома
        hd3 перенесена в логический раздел 109 на hdisk1.
```

Функция оперативной замены для логического тома или группы томов позволяет определять отчеты и статистику, просматривать статистику, выбирать логические разделы для переноса, указывать целевой физический раздел и проверять информацию перед подтверждением изменений.

Реализация стратегии группы томов

После выбора стратегии создания группы томов проанализируйте текущую конфигурацию с помощью команды **lspv**.

Стандартная конфигурация включает одну группу томов с несколькими физическими томами, подключенными к одному адаптеру, и другое необходимое оборудование. Обычно, чем больше дисков входит в группу томов с контролем целостности, тем выше вероятность сохранения целостности в случае сбоя диска. В группах без контроля целостности должно быть не менее двух дисков. Для изменения стратегии выполните следующие действия:

1. Просмотрите список занятых и свободных физических томов с помощью команды **lspv**.
2. Для обеспечения целостности добавьте один или несколько физических томов.
3. Отключите контроля целостности для группы томов
4. Установите другое аппаратное обеспечение, если это необходимо для обеспечения высокой готовности. См. инструкции в руководстве по обслуживанию системы.

Понятия, связанные с данным:

“Отключение контроля целостности для группы томов” на стр. 354
Вы можете отключить для группы томов контроль целостности, обеспечив постоянный доступ к данным даже в случае нарушения целостности.

Задачи, связанные с данной:

“Добавление дисков без выключения системы” на стр. 357
Приведенная ниже процедура описывает включение и настройку диска с помощью функции оперативного удаления, позволяющей добавлять диски без выключения системы.

Пространство подкачки и виртуальная память

В AIX для работы с объемом памяти, превышающем объем физической памяти, применяется виртуальная память.

Управлением страницами памяти в RAM или на диске занимается Администратор виртуальной памяти (VMM). Сегменты виртуальной памяти делятся на блоки, называемые *страницами*. *Пространство подкачки* - это часть логического тома, выделенная для хранения неиспользуемой информации, расположенной в виртуальной памяти. Такой логически том помечен как *пространство подкачки*. Если объем свободной памяти системы невелик, то редко используемые программы и данные перемещаются из оперативной памяти в пространство подкачки, освобождая таким образом оперативную память для других процессов.

Общие сведения о пространстве подкачки

Пространство подкачки - это часть логического тома с выделенной дисковой памятью, в которой хранится неиспользуемая в данный момент информация, расположенная в виртуальной памяти.

Такой логически том помечен как *пространство подкачки*. Если объем свободной памяти системы невелик, то редко используемые программы и данные перемещаются из оперативной памяти в пространство подкачки, освобождая таким образом оперативную память для других процессов.

Существует еще один тип пространства подкачки, в котором для хранения данных подкачки применяется сервер NFS. Для каждого клиента NFS, обращающегося к такому пространству подкачки, на сервере NFS создается и экспортируется специальный файл. Размер этого файла и определяет размер пространства подкачки данного клиента.

Необходимый объем пространства подкачки зависит от назначения системы. При нехватке пространства подкачки возможно аварийное завершение отдельных процессов, а переполнение пространства подкачки приводит к останову системы. При уменьшении свободного объема пространства подкачки можно определить дополнительное пространство подкачки.

Для изменения размера пространства подкачки логического тома можно создать новый логический том пространства подкачки или увеличить размер существующих логических томов пространства подкачки. Для увеличения размера пространства подкачки NFS необходимо увеличить размер файла, расположенного на сервере.

Общий объем пространства подкачки, который может использоваться системой, представляет собой сумму размеров всех активных логических томов пространства подкачки.

Понятия, связанные с данным:

“Устранение неполадок пространства подкачки” на стр. 423
Чаще всего неполадки пространства подкачки бывают связаны с нехваткой свободного места на диске.

Стратегии выделения пространства подкачки:

Переменная среды *PSALLOC* определяет используемый алгоритм выделения пространства подкачки: отложенный или статический.

В операционной системе AIX выделение пространства подкачки может выполняться в двух режимах. Значение по умолчанию: отложенный. Для применения алгоритма статического выделения пространства подкачки следует изменить переменную среды *PSALLOC*, предварительно рассмотрев следующие вопросы. При использовании статического алгоритма, заполнение пространства подкачки может привести к аварийному завершению работы системы.

Сравнение отложенного и статического выделения пространства подкачки:

Способ выделения памяти и пространства подкачки определяется значением переменной *PSALLOC*.

Если переменная *PSALLOC* не задана, ей присвоено значение `null`, или любое другое значение, отличное от `early`, то применяется алгоритм отложенного выделения (значение `deferred`).

Применяемый по умолчанию алгоритм (`deferred`) повышает эффективность использования дисковой памяти и поддерживает приложения, применяющие алгоритмы с резервированием ресурсов памяти. В соответствии с этим алгоритмом пространство подкачки не резервируется сразу после запроса. Дисковый блок выделяется только при необходимости выгрузить страницу из памяти. Некоторые программы запрашивают большой объем виртуальной памяти, а используют его лишь частично. К таким программам относятся технические приложения, применяющие в качестве структур данных разреженные векторы или матрицы. Алгоритм отложенного выделения памяти наиболее эффективен в системах реального времени с ядром, осуществляющим подкачку по запросу, например, при работе с ядром операционной системы.

Иногда пространство подкачки не используется, особенно в системах с большим объемом физической памяти, в которых подкачка применяется редко. Алгоритм позволяет отложить выделение пространства подкачки до момента необходимости выгрузки на диск неиспользуемой страницы памяти, что позволяет избежать неэффективного выделения пространства подкачки. Такой вариант может реальным, если алгоритм с отложенным выделением попытается выделить пространство подкачки большего размера, чем в системе доступно памяти. Такая ситуация называется фиксацией слишком большого пространства подкачки.

В случае фиксации слишком большого пространства подкачки. при котором пространство подкачки исчерпывается, попытка выделить дисковый блок пространства подкачки для выгрузки на диск неиспользуемой страницы приведет к возникновению ошибки. Операционная система в этой ситуации старается предотвратить сбой системы. аварийно завершая процессы, на которые влияет фиксация слишком большого пространства подкачки. Сигнал **SIGDANGER** отправляется для уведомления процессов о нехватке места в пространстве подкачки. При достижении критического уровня тем процессам, которые не получили сигнал **SIGDANGER**, отправляется сигнал **SIGKILL**.

Для применения статического алгоритма выделения памяти и пространства подкачки следует изменить значение переменной среды *PSALLOC* на `early`. Если в этот момент объем свободного пространства окажется недостаточным, то запрос на выделение памяти выполнен не будет.

Если переменной среды *PSALLOC* будет присвоено значение `early`, то все вновь запускаемые программы будут выполняться в режиме статического выделения памяти. Если при работе в этом режиме в момент получения запроса зарезервировать требуемый объем памяти не удастся, то подпрограммы **malloc** и **brk** вернут код ошибки.

Процессы. работающие в режиме среды со статическим выделением. не будут посылать сигнал **SIGKILL** при возникновении события, связанного с недостатком пространства подкачки.

В зависимости от ширины области применения, переменную среды *PSALLOC* можно изменить на `early` несколькими способами.

Переход в режим статического выделения памяти влияет на работу следующих подпрограмм:

- **malloc**
- **free**
- **calloc**
- **realloc**
- **brk**
- **sbrk**
- **shmget**

- **shmctl**

Задачи, связанные с данной:

“Настройка переменной среды PSALLOC для статического выделения” на стр. 420
Способ выделения памяти и пространства подкачки определяется значением переменной *PSALLOC*.

Статическое выделение:

Алгоритм статического выделения памяти гарантирует выделение запрошенного объема памяти. Для эффективной работы необходимо правильно выбрать режим выделения пространства подкачки.

Когда объем свободного места в пространстве подкачки достигает заданного порогового значения, новые процессы запускаться не будут, а уже запущенные процессы не смогут получить память. Все процессы, запущенные в режиме отложенного выделения памяти (по умолчанию), являются возможными получателями сигнала **SIGKILL**. Кроме того, поскольку память регулярно запрашивается ядром операционной системы, заполнение пространства подкачки может привести к аварийному завершению работы системы.

Перед тем, как перевести систему в режим статического выделения памяти, необходимо определить оптимальный объем пространства подкачки. Для режима статического выделения памяти требуется больший объем пространства подкачки, чем для отложенного. Объем пространства подкачки зависит от назначения системы и запускаемых в ней программ. В первом приближении можно выбрать объем пространства подкачки, превышающий объем физической памяти в четыре раза.

Некоторые приложения, запускаемые в режиме статического выделения памяти могут использовать очень большой объем пространства подкачки. Современному серверу AIXwindows в режиме статического выделения памяти необходимо более 250 МБ в пространстве подкачки. Объем пространства подкачки, необходимый для каждого приложения, зависит от алгоритма и режима работы приложения.

Все команды и подпрограммы, выдающие информацию об использовании процессами пространства подкачки и памяти, указывают также объем пространства подкачки, выделенный в статическом режиме. Команда **lsps** с флагом **-s** показывает общий объем выделенного пространства подкачки, включая объем, выделенный в статическом режиме.

Размер пространства подкачки по умолчанию:

Размер пространства подкачки по умолчанию определяется во время установки AIX по следующим правилам.

- Для пространства подкачки отводится не менее 16 МБ. Исключение составляет устройство `hd6`, которое может использовать минимум 64 МБ.
- Пространство подкачки может занимать не более 20% от общего объема дискового пространства.
- Если объем физической памяти меньше 256 МБ, то для пространства подкачки выделяется вдвое больший объем дисковой памяти.
- Если объем реальной памяти больше 256 МБ, то для пространства подкачки выделяется 512 МБ.

Файл, команды и опции пространства подкачки:

Файл `/etc/swapspaces` задает пространства подкачки и их атрибуты.

Запись о пространстве подкачки добавляется в файл `/etc/swapspaces` командой **mkps** и удаляется из файла `/etc/swapspaces` командой **rmps**. Атрибуты пространства подкачки в файле изменяются с помощью команды **chps -a** или **chps -c**. Файлы, для которых используется прежний формат (где отсутствуют атрибуты размера контрольной суммы и автоматической перекачки в файлы настройки), продолжают поддерживаться. Если пространство подкачки слишком велико, то удалите из него лишние логические разделы с помощью команды **chps-d** (перезагрузка не требуется).

Для управления пространством подкачки служат следующие команды:

Элемент	Описание
chps	Изменяет атрибуты пространства подкачки.
lsps	Показывает характеристики пространства подкачки.
mkps	Добавляет дополнительное пространство подкачки. При создании логического тома пространства подкачки команда **mkps** вызывает команду **mklv** с определенным набором опций. Для создания пространства подкачки NFS команда **mkps** вызывает команду **mkdev** с некоторым набором опций. Для пространств подкачки NFS в команде **mkps** необходимо указать имя хоста сервера NFS и путь к файлу, экспортируемому с сервера.
rmps	Удаляет неактивное пространство подкачки.
swapoff	Отключает одно или несколько пространств подкачки без перезагрузки системы. Информация из удаляемого пространства подкачки перемещается в другие активные пространства подкачки. Отключенное пространство подкачки можно удалить командой **rmps**.
swapon	Активизирует пространство подкачки. В начале инициализации системы вызывается команда **swapon** для активации начального устройства пространства подкачки. На последнем этапе инициализации, когда все устройства уже доступны, команда **swapon** вызывается для активации дополнительных пространств подкачки.

Опция **тип подкачки** является обязательной для для всех пространств подкачки логических томов.

Следующие опции применяются для повышения производительности при работе с логическим томом:

- Разместите его в середине диска для уменьшения среднего расстояния перемещения головки.
- Создайте несколько пространств подкачки на различных физических томах.

Настройка пространства подкачки

Многие задачи настройки можно выполнить с помощью SMIT. Пространство подкачки и выделение памяти контролируются посредством переменной среды **PSALLOC**.

Создание и активация пространства подкачки:

Для того чтобы операционная система могла работать с пространством подкачки, его необходимо создать и активировать.

Общий объем пространства подкачки часто определяется методом проб и ошибок. Однако в большинстве случаев объем оперативной памяти просто удваивается и полученное значение принимается за объем пространства подкачки.

Если вы предпочитаете работать со SMIT, то введите одну из следующих команд быстрого доступа:

- Для просмотра пространства подкачки: `smit lsps`
- Для добавления пространства подкачки: `smit mkps`
- Для активации пространства подкачки: `smit swapon`

Задачи, связанные с данной:

“Перемещение пространства подкачки hd6 внутри группы томов” на стр. 422
Перемещение пространства подкачки по умолчанию с диска hdisk0 на другой диск той же группы томов является достаточно простой процедурой, не требующей завершения работы и перезапуска системы.

Повышение производительности подкачки:

Для повышения производительности рекомендуется создавать несколько пространств подкачки и по возможности располагать их на разных физических томах.

Однако допускается и наличие нескольких пространств подкачки на одном физическом томе. При небольшом опыте работы с системой в случае выбора нескольких физических томов рекомендуется использовать только диски из группы томов rootvg.

Настройка переменной среды PSALLOC для статического выделения:

Способ выделения памяти и пространства подкачки определяется значением переменной *PSALLOC*.

Значение по умолчанию - `late`. В следующих примерах описаны различные способы изменения переменной среды *PSALLOC* для настройки режима статического выделения (значения `early`). Выбор конкретного способа зависит от того, в какой области должно действовать внесенное изменение.

- Введите следующую команду:
  ```
  PSALLOC=early;export PSALLOC
  ```
  Все последующие команды, вводимые в этом сеансе оболочки, будут применять режим статического выделения.
- Добавьте следующую команду в файл ресурса оболочки (`.shrc` или `.kshrc`):
  ```
  PSALLOC=early;export PSALLOC
  ```
  Все процессы, запускаемые в вашем сеансе работы в системе, за исключением оболочки входа в систему, будут применять режим статического выделения. Этот способ также обеспечивает защиту процессов от сигнала **SIGKILL**.
- С помощью функции **putenv** в программе присвойте переменной среды *PSALLOC* значение `early`. Статический режим выделения начнет применяться, начиная со следующего вызова функции **exec**.

Понятия, связанные с данным:

“Сравнение отложенного и статического выделения пространства подкачки” на стр. 417
Способ выделения памяти и пространства подкачки определяется значением переменной *PSALLOC*.

Изменение или удаление пространства подкачки:

Изменение пространства подкачки легко выполнить с помощью SMIT, но удаление пространства подкачки более рискованно.

Изменить параметры пространства подкачки можно с помощью команды SMIT `smit chps`.

При удалении пространства подкачки следует соблюдать особую осторожность, особенно, если вы удаляете пространство подкачки по умолчанию, например, hd6. Для удаления пространства подкачки по умолчанию необходима специальная процедура, поскольку эти пространства активируются сценариями настройки системы во время загрузки. Для удаления одного из пространств подкачки по умолчанию необходимо изменить эти сценарии и создать новый загрузочный образ.

Внимание: Неправильно выполненное удаление пространств подкачки по умолчанию может привести к возникновению ошибок при перезагрузке системы. Описанная ниже процедура требует наличия большого опыта управления системой.

Для удаления существующего пространства подкачки выполните следующие действия:

1. Войдя в систему под именем пользователя с правами доступа root, введите следующую команду SMIT:
   ```
   smit swapoff
   ```
2. Если удаляемое пространство подкачки является устройством дампа по умолчанию, то перед его удалением следует выбрать в качестве устройства дампа другое пространство подкачки или логический том. Для изменения устройства дампа по умолчанию введите следующую команду:
   ```
   sysdumpdev -P -p /dev/новое-устройство-дампа
   ```
3. Удалите пространство подкачки с помощью команды:
   ```
   smit rmps
   ```

Понятия, связанные с данным:

“Устранение неполадок пространства подкачки” на стр. 423
Чаще всего неполадки пространства подкачки бывают связаны с нехваткой свободного места на диске.

Применение интерфейса программирования режима выделения пространства подкачки:

Программный интерфейс, управляющий режимами выделения памяти в пространстве подкачки, использует переменную среды *PSALLOC*.

Для того чтобы проверить, в каком режиме работает приложение (со статическим или динамическим выделением пространства подкачки), выполните следующие действия:

1. Для того чтобы узнать текущее значение переменной среды *PSALLOC*, вызовите функцию **getenv**.
2. Если это необходимо, измените значение переменной среды *PSALLOC* функцией **setenv**. Так как значение переменной *PSALLOC* проверяет только функция **execve**, вызовите функцию **execve**, передав ей все параметры и переменные среды приложения. При повторном запуске программа проверит значение переменной среды *PSALLOC*, и, обнаружив правильное значение, продолжит работу.
3. Если функция **getenv** вернет верное значение переменной среды *PSALLOC*, ничего изменять не нужно. Выполнение приложения продолжится в обычном режиме.

Перемещение и уменьшение пространства подкачки hd6:

Уменьшение размера или перемещение пространства подкачки по умолчанию может потребоваться для повышения производительности системы путем переноса пространства подкачки на диски с меньшим уровнем нагрузки. Сокращение размера или перемещение подкачки по умолчанию также позволяет сохранить свободное пространство над диске hdisk0.

Цель перемещения пространства подкачки и уменьшения его размера одна: использовать для подкачки наименее загруженные диски. При установке системы на диске hdisk0 по умолчанию создается логический том пространства подкачки (hd6), в котором частично или полностью хранятся файловые системы / (корневая) и /usr. Если вы выбрали стратегию распределения с минимальным числом дисков, при которой вся файловая система / и большая часть файловой системы /usr хранятся на диске hdisk0, то перемещение пространства подкачки на менее загруженный диск может существенно повысить производительность. Но даже если вы выбрали стратегию распределения с максимальным числом дисков и обе файловые системы / и /usr распределены по нескольким физическим томам, то на диске hdisk2 (предположим, что вы работаете с тремя дисками) скорее всего будет меньше логических разделов, относящихся к наиболее загруженным файловым системам.

Приведенные процедуры позволяют сокращать объем пространства подкачки hd6 и перемещать его в пределах группы томов.

Понятия, связанные с данным:

“Стратегии распределения данных по дискам” на стр. 408
Стратегия распределения данных по дискам определяет число дисков, на которых будут расположены физические разделы логического тома.

“Устранение неполадок пространства подкачки” на стр. 423
Чаще всего неполадки пространства подкачки бывают связаны с нехваткой свободного места на диске.

Уменьшение пространства подкачки hd6:

В следующей процедуре для уменьшения размера существующих пространств подкачки, включая основное пространство подкачки, а также основное и дополнительное устройство дампа, применяется команда **chps**.

Команда **chps** вызывает сценарий **shrinkps**, уменьшающий пространство подкачки, не вызывая ошибок при загрузке системы. Этот сценарий выполняет следующие операции:

1. Создает временное пространство подкачки на том же томе
2. Перемещает информацию во временное пространство
3. Создает на том же томе новое пространство подкачки меньшего размера
4. Удаляет старое пространство подкачки

Для успешного выполнения команды **chps** на диске должно быть достаточно свободного места (не присвоенного ни одному логическому тому) для создания временного пространства подкачки. Размер временного пространства подкачки равен размеру пространства, необходимого для хранения всех страниц из старого пространства подкачки. Минимальный размер основного пространства подкачки - 32 МБ. Минимальный размер других пространств подкачки - 16 МБ.

Примечание: Если при выполнении следующей процедуры будет обнаружена ошибка ввода-вывода, то система может потребовать немедленного перезапуска.

1. Для просмотра информации о распределении логических томов и файловых систем по физическим томам воспользуйтесь следующей командой:

   ```
   lspv -l hdiskX
   ```

   где *hdiskX* - имя физического тома.
2. Для уменьшения размера пространства подкачки введите следующую команду:

   ```
   smit chps
   ```

Примечание: Основное пространство подкачки задано в загрузочной записи. Поэтому при перезагрузке системы основное пространство подкачки всегда активируется. Команда **chps** не позволяет отключить основное пространство подкачки.

Приоритет отдается управлению оперативной конфигурацией. Проверка системы может привести к невозможности уменьшения пространства подкачки. Ошибки, возникшие в процессе создания временного пространства подкачки приведут к выходу из процедуры, в результате чего буду восстановлены первоначальные параметры конфигурации системы. Другие неполадки могут обусловить необходимость вмешательства системного администратора и немедленной перезагрузки системы. Некоторые ошибки могут сделать невозможным удаление временного пространства подкачки. Для исправления такой неполадки понадобится вмешательство системного администратора.

Внимание: В случае обнаружения командой **swapoff** в сценарии **shrinkps** ошибки ввода-вывода в системной или пользовательской резервной странице рекомендуется немедленно перезагрузить систему во избежание сбоя. После перезагрузки временное пространство подкачки активизируется, и можно попытаться закрыть и перезапустить приложения, обнаружившие ошибки ввода-вывода. Если попытка будет успешной и команда **swapoff** отключит пространство подкачки, завершить процедуру уменьшения можно вручную, создав пространство подкачки необходимого размера и удалив временное пространство подкачки с помощью команд **mkps**, **swapoff** и **rmps**.

До перезапуска системы не пытайтесь удалить (с помощью **rmps**) или активировать (с помощью **chps**) отключенное пространство подкачки, в котором была обнаружена ошибка ввода-вывода. Дисковое пространство может быть использовано повторно, что приведет к возникновению новых неполадок.

Перемещение пространства подкачки hd6 внутри группы томов:

Перемещение пространства подкачки по умолчанию с диска hdisk0 на другой диск той же группы томов является достаточно простой процедурой, не требующей завершения работы и перезапуска системы.

Войдя в систему под именем пользователя с правами доступа root введите следующую команду для перемещения пространства подкачки по умолчанию (hd6) с диска hdisk0 на диск hdisk2:

```
migratepv -l hd6 hdisk0 hdisk2
```

Внимание: Не рекомендуется перемещать пространство подкачки с именем hd6 из rootvg в другую группу томов, так как это имя применяется в некоторых операциях, настройка которых затруднена, включая второй этап загрузки и обращение к группе томов rootvg при загрузке со съемного носителя. На втором этапе загрузки могут быть активны только пространства подкачки, находящиеся в группе томов rootvg, и, если в этой группе не окажется ни одного пространства подкачки, то это может оказать существенное влияние на процесс загрузки. Если вы хотите разместить большинство пространств подкачки в других группах томов, то рекомендуется уменьшить hd6 до объема физической памяти, а затем создать пространства подкачки необходимого размера в других группах томов.

Понятия, связанные с данным:

“Создание и активация пространства подкачки” на стр. 419
Для того чтобы операционная система могла работать с пространством подкачки, его необходимо создать и активировать.

Устранение неполадок пространства подкачки

Чаще всего неполадки пространства подкачки бывают связаны с нехваткой свободного места на диске.

Общий объем пространства подкачки часто определяется методом проб и ошибок. Однако в большинстве случаев объем оперативной памяти просто удваивается и полученное значение принимается за объем пространства подкачки. При нехватке пространства подкачки возможно аварийное завершение отдельных процессов, а переполнение пространства подкачки приводит к останову системы. Приведенная ниже информация о сигналах и ошибках поможет вам отслеживать и устранять или предупреждать неполадки, связанные с пространством подкачки.

Операционная система контролирует число свободных блоков пространства подкачки и обнаруживает ситуации, когда их становится слишком мало. Если число свободных блоков пространства подкачки опускается ниже порога, называемого *уровнем предупреждения пространства подкачки*, то система информирует об этом все процессы (за исключением **kprocs**) с помощью сигнала **SIGDANGER**. Если заполнение пространства подкачки продолжается и число свободных блоков опускается ниже *критического уровня пространства подкачки*, то система отправляет сигнал **SIGKILL** процессам, наиболее интенсивно использующим пространство подкачки, и не имеющим обработчика сигнала **SIGDANGER**. (По умолчанию процессы игнорируют сигнал **SIGDANGER**.) Система продолжает отправлять сигналы **SIGKILL** до тех пор, пока число свободных блоков пространства подкачки не поднимется выше критического уровня.

Примечание: Если параметру **low_ps_handling** присвоено значение 2 (в команде **vmo**) и не найдены процессы для уничтожения (без обработчика **SIGDANGER**), то система отправит сигнал **SIGKILL** самому молодому процессу, у которого есть обработчик сигнала **SIGDANGER**.

Процессы, выделяющие память динамически, могут контролировать объем доступного пространства подкачки, отслеживая его с помощью функции **psdanger** или с помощью специальных функций выделения. С помощью функции **disclaim** вы можете избежать завершения процессов при достижении критического уровня пространства подачки. Для этого необходимо определить обработчик сигнала **SIGDANGER** и освободить в нем ресурсы памяти и пространства подкачки, выделенные в областях данных и стека, а также в общих сегментах памяти.

Если на экране появляется следующее сообщение, то это значит, что необходимо увеличить пространство подкачки:

```
INIT: Нехватка пространства подкачки!
```

ИЛИ

```
Пространство подкачки скоро будет исчерпано.
Рекомендуется сохранить документы, поскольку работа
этой программа (и, возможно, всей операционной системы) может
быть завершена без дополнительных предупреждений, как только
пространство подкачки будет заполнено.
```

Понятия, связанные с данным:

“Перемещение и уменьшение пространства подкачки hd6” на стр. 421
Уменьшение размера или перемещение пространства подкачки по умолчанию может потребоваться для повышения производительности системы путем переноса пространства подкачки на диски с меньшим уровнем нагрузки. Сокращение размера или перемещение подкачки по умолчанию также позволяет сохранить свободное пространство над диске hdisk0.

“Общие сведения о пространстве подкачки” на стр. 416
Пространство подкачки - это часть логического тома с выделенной дисковой памятью, в которой хранится неиспользуемая в данный момент информация, расположенная в виртуальной памяти.

Задачи, связанные с данной:

“Изменение или удаление пространства подкачки” на стр. 420
Изменение пространства подкачки легко выполнить с помощью SMIT, но удаление пространства подкачки более рискованно.

Администратор виртуальной памяти

Администратор виртуальной памяти (VMM) обслуживает запросы к памяти, поступающие от системы и ее приложений.

Сегменты виртуальной памяти разбиты на блоки, называемые *страницами*, каждая из которых может находиться либо в оперативной памяти (RAM), либо на диске. В AIX для работы с объемом памяти, превышающем объем физической памяти, применяется виртуальная память. Управлением страницами памяти в RAM или на диске занимается Администратор виртуальной памяти (VMM).

Управление физической память в Администраторе виртуальной памяти:

В AIX сегменты виртуальной памяти разбиты на блоки по 4096 байт, называемые страницами. Оперативная память также разбита на страницы по 4096 байт.

VMM выполняет две основные функции:

- Управляет размещением страниц памяти
- Обслуживает обращения к страницам памяти, которые находятся в пространстве подкачки или еще не существуют.

Для выполнения этих функций VMM поддерживает *список свободных* страниц памяти. При определении того, какие страницы виртуальной памяти будут занесены в список свободных, VMM использует алгоритм замены страниц. Этот алгоритм учитывает существование постоянных и рабочих сегментов, интенсивность подкачки и пороги VMM.

Список свободных страниц Администратора виртуальной памят:

VMM поддерживает список свободных страниц памяти, используемых для удовлетворения запросов на страницы.

AIX занимает всю свободную оперативную память, за исключением небольшого объема, управляемого списком свободных страниц. При этом VMM *перераспределяет* и *выгружает* страницы для освобождения пространства и занесении информации о свободных страницах в список. При этом перераспределяемые страницы виртуальной памяти выбираются алгоритмом замены страниц VMM.

Постоянные или рабочие сегменты в администраторе виртуальной памяти:

AIX различает два типа сегментов памяти. Для понимания принципов работы VMM необходимо понимать разницу между рабочими и постоянными сегментами.

Постоянный сегмент расположен на жестком диске. Файлы, содержащие данные и исполняемые программы, связаны с постоянными сегментами. При обращении к файлам JFS или JFS2 их данные копируются в RAM.

Параметры VMM позволяют указать, когда виртуальная память, выделенная для постоянных страниц, должна быть перераспределена для размещения других данных.

Рабочие сегменты существуют только в процессе их использования программой. С рабочими сегментами на связана дисковая память. Стек процессов и области данных связаны с рабочими сегментами и текстовыми сегментами общих библиотек. Если хранить в оперативной памяти все страницы рабочих сегментов невозможно, то они переносятся на диск. Для этого применяется пространство подкачки. При завершении программы все ее рабочие сегменты заносятся в список свободных.

Рабочие сегменты и пространство подкачки в администраторе виртуальной памяти:

За рабочими сегментами в оперативной памяти закреплено место в пространстве подкачки.

Оно используется только в том случае, если страница должна быть выгружена. Страница, выделенная для одного сегмента памяти, не может применяться для другого. Она остается зарезервированной до тех пор, пока соответствующий сегмент существует в виртуальной памяти. Это позволяет не тратить время на выделение памяти в пространстве подкачки при каждой операции выгрузки страницы.

VMM использует два режима выделения пространства подкачки: *статический* и *динамический*. При статическом выделении пространство подкачки выделяется при запросе на рабочую страницу. При динамическом выделении пространство выделяется при фактической выгрузке страницы из оперативной памяти. Этот тип позволяет существенно снизить требования к объему пространства подкачки.

Средство управления загрузкой памяти Администратор виртуальной памяти:

Когда процесс обращается к странице виртуальной памяти, которая находится на диске (потому что она была выгружена или еще не считывалась), то для размещения этой страницы в оперативной памяти из нее придется выгрузить одну или несколько других страниц, если список свободных страниц пуст. VMM пытается выгрузить те страницы, которые не использовались дольше всех и, скорее всего, не будут использоваться в ближайшем будущем.

Эффективный алгоритм распределения хранит в оперативной памяти страницы работающих процессов, а страницы простаивающих выгружает на диск. Однако при интенсивном использовании оперативной памяти всеми процессами выбрать такие страницы достаточной сложно. В результате на диск могут выгружаться и страницы активных процессов. При этом может может начаться постоянное *бесполезное* перемещение страниц. В таких случаях система тратит большую часть времени на загрузку и выгрузку страниц, а не на выполнение инструкций. Алгоритм управления загрузкой памяти VMM позволяет выявлять и исправлять такие ситуации.

Файловые системы

Файловая система - это иерархическая структура (дерево) файлов и каталогов.

Эта структура аналогична перевернутому дереву, корень которого расположен наверху, а ветви направлены вниз. В таком дереве файлов данные и программы объединены в группы, называемые каталогами. Это позволяет одновременно выполнять операции для всего содержимого каталога.

Файловая система располагается на одном логическом томе. К файловой системе будут относится все файлы и каталоги, расположенные на логическом томе. Некоторые задачи удобнее выполнять сразу для всей файловой системы, а не для каждого каталога в отдельности. Например, можно создать резервную копию всей файловой системы, переместить файловую систему или установить ее защиту. Можно создать *моментальную копию* файловой системы JFS или JFS2, то есть образ файловой системы в данный момент времени.

Примечание: Максимальное число логических разделов на логическом томе равно 32 512. Дополнительная информация о характеристиках логических томов файловых систем приведена в описании команды **chlv**.

Для создания файловой системы на логическом томе можно воспользоваться командой **mkfs** (создать файловую систему) или Инструментом управления системой (команда **smit**).

Для предоставления доступа к файловой системе ее необходимо смонтировать в каталоге, называемом точкой монтирования. После монтирования нескольких файловых систем создается структура каталогов, образующая единую файловую систему. Она представляет собой иерархическую систему с одним корнем. Эта структура образуется из всех базовых и создаваемых вами файловых систем. Для предоставления доступа к локальным и удаленным файловым системам применяется команда **mount**. В результате файловая система становится доступной на чтение и запись с вашего компьютера. Для монтирования и размонтирования файловой системы обычно требуется принадлежность пользователя к группе system. Файловые системы, указанные в файле `/etc/filesystems`, монтируются автоматически. Для размонтирования локальной или удаленной файловой системы, с которой не работают никакие пользователи или процессы, можно воспользоваться командой **umount**. Дополнительная информация о монтировании файловых систем приведена в разделе “Монтирование” на стр. 453.

Основной тип файловой системы AIX называется *журнализированной файловой системой (JFS)*. Как и в базах данных, для поддержания целостности таких файловых систем применяется журнализация. Это позволяет предотвратить повреждение файловой системы при аварийном завершении работы системы.

Операционная система AIX поддерживает несколько типов файловых систем, включая журнализированную файловую систему (JFS) и расширенную журнализированную файловую систему (JFS2). Более подробная информация о типах и характеристиках файловых систем приведена в разделе “Типы файловых систем” на стр. 459.

С операциями над файловыми системами связаны некоторые наиболее важные задачи управления системой, в том числе:

- Выделение областей памяти на логических томах для файловых систем
- Создание файловых систем
- Предоставление пользователям доступа к файловой системе
- Отслеживание использования памяти, выделенной файловой системе
- Создание резервной копии файловой системы на случай потери данных в результате сбоя системы
- Поддержание файловых систем в согласованном состоянии

Эти задачи выполняются системным администратором.

Понятия, связанные с данным:

“Файловые системы” на стр. 393
Логический том определяет распределение дискового пространства вплоть до уровня физических разделов. Более точное управление данными достигается путем применения программных компонентов, например, администратора виртуальной памяти и файловых систем. Поэтому последний этап организации дисковой памяти - это создание *файловых систем*.

Задачи, связанные с данной:

“Создание и резервное копирование моментальной копии JFS2” на стр. 36
Можно создавать моментальные копии смонтированных файловых систем JFS2, создавая соответствующий определенному моменту времени образ файловой системы на уровне блоков.

“Создание и резервное копирование внешней моментальной копии JFS2” на стр. 36
Можно создавать моментальные копии смонтированных файловых систем JFS2, создавая соответствующий определенному моменту времени образ файловой системы на уровне блоков.

“Создание и резервное копирование внутренней моментальной копии JFS2” на стр. 37
Можно создавать моментальные копии смонтированных файловых систем JFS2, создавая соответствующий определенному моменту времени образ файловой системы на уровне блоков.

Основные сведения о файловой системе

Перед тем как приступить к управлению и настройке файловой системы, необходимо ознакомиться с основами организации и содержимым дерева файлов.

Организация и содержимое файлового дерева:

Дерево файлов - это иерархическая структура, позволяющая объединять файлы со сходной информацией в каталоги. Такая структура дерева файлов позволяет монтировать удаленные каталоги и файлы.

Системный администратор применяет эти каталоги в качестве компонентов для создания уникального дерева файлов на каждом компьютере-клиенте, монтируя каталоги одного или нескольких серверов. По сравнению с хранением информации на локальном диске, у удаленного монтирования файлов и каталогов есть следующие преимущества:

- Экономия дискового пространства
- Более простое, централизованное управление.
- Обеспечение более надежной защиты данных.

У дерева файлов есть следующие особенности:

- Файлы, общие для компьютеров с одинаковой аппаратной архитектурой, расположены в файловой системе `/usr`.
- Файлы, создаваемые отдельными клиентам (например, буферные файлы и файлы почты) находятся в файловой системе `/var`.
- Текстовые файлы, применяемые системами с различной архитектурой (например, руководства), хранятся в каталоге `/usr/share`.
- Корневая файловая система (`/`) содержит файлы и каталоги, необходимые для работы системы. Например, в ней находится каталог устройств, программы, применяемые при запуске системы, а также точки монтирования файловых систем.
- Файловая система `/home` служит точкой монтирования пользовательских каталогов.

Структура файловой системы:

Важно понимать разницу между файловой системой и каталогом. Файловая система - это раздел жесткого диска, выделенный для хранения файлов. Доступ к данному разделу жесткого диска осуществляется путем монтирования файловой системы в каталог. После монтирования файловая система выглядит для пользователя как обычный каталог.

Однако, поскольку файловые системы и каталоги имеют разную структуру, их данными можно управлять независимо.

После установки операционной системы создается структура каталогов, показанная на следующем рисунке.

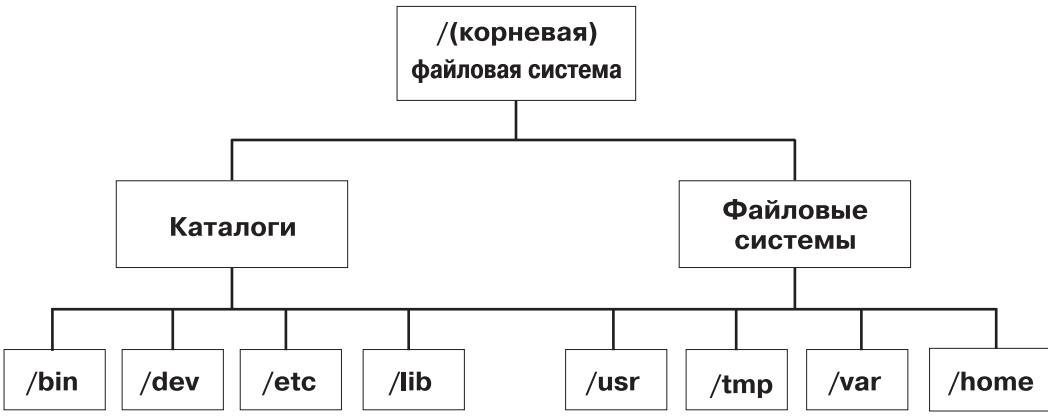


Рисунок 7. Дерево файловой системы /root. На этом рисунке показана структура каталогов, начинающаяся с файловой системы / (root), и включающая каталоги и файловые системы. В число каталогов входят /bin , /dev, /etc и /lib. К числу файловых систем относятся /usr, /tmp, /var и /home.

Каталоги в правой части (/usr, /tmp, /var и /home) представляют собой файловые системы, то есть для них выделены отдельные области дискового пространства. Эти файловые системы автоматически монтируются при запуске системы, поэтому с точки зрения пользователя эти файловые системы неотличимы от каталогов, показанных в левой части рисунка (/bin, /dev, /etc и /lib).

Ниже описаны файловые системы, которые по умолчанию располагаются на связанном устройстве автономного компьютера:

/устройство	/файловая система
/dev/hd1	/home
/dev/hd2	/usr
/dev/hd3	/tmp
/dev/hd4	/(корневой каталог)
/dev/hd9var	/var
/proc	/proc
/dev/hd10opt	/opt

У дерева файлов есть следующие особенности:

- Файлы, общие для компьютеров с одинаковой аппаратной архитектурой, расположены в файловой системе /usr.
- Файлы отдельных клиентов, например, буферные файлы или файлы почты, расположены в файловой системе /var.
- Корневая файловая система The / содержит файлы и каталоги, необходимые для работы системы. В частности, она содержит:
  - Каталог устройств (/dev)
  - Точки монтирования других файловых систем к корневой файловой системе, например, /mnt
- Файловая система /home представляет собой точку монтирования домашних каталогов пользователей.
- На серверах каталог /export содержит файлы пространства подкачки, корневые файловые системы клиентов (не предназначенные для совместного использования), каталог дампа, домашний каталог и каталог /usr/share для бездисковых клиентов, а также экспортированные каталоги /usr.
- Файловая система /proc содержит информацию о состоянии процессов и нитей в системе.
- Файловая система /opt содержит дополнительное программное обеспечение, в частности, приложения.

Ниже приведено описание некоторых подкаталогов корневой файловой системы /.

Элемент	Описание
/bin	Символьная ссылка на каталог /usr/bin.
/dev	Содержит каталоги со специальными файлами локальных устройств. Каталог /dev содержит специальные файлы накопителей на магнитной ленте, принтеров, разделов дисков и терминалов.
/etc	Содержит файлы с информацией о конфигурации, уникальные для каждого компьютера. В приведенных примерах этот каталог содержит: • /etc/hosts • /etc/passwd
/export	Содержит каталоги и файлы сервера, предназначенные для удаленных клиентов.
/home	Точка монтирования файловых систем, содержащих домашние каталоги пользователей. Файловая система /home содержит файлы и каталоги отдельных пользователей. На автономном компьютере каталог /home служит точкой монтирования для локальной файловой системы. В сетевой среде на сервере могут храниться файлы пользователей, которые должны быть доступны нескольким компьютерам. В этом случае выполняется удаленное монтирование копии каталога /home, хранящейся на сервере, в локальную файловую систему /home.
/lib	Символьная связь с каталогом /usr/lib, содержащим независимые от архитектуры библиотеки с именами вида lib*.a.
/sbin	Содержит файлы, предназначенные для загрузки компьютера и монтирования файловой системы /usr. Большинство команд, выполняемых при загрузке, расположены в дисковой файловой системе RAM загрузочного образа, поэтому каталог /sbin содержит небольшое число команд.
/tmp	Точка монтирования файловых систем, содержащих временные файлы, созданные системой.
/u	Символьная связь с каталогом /home.
/usr	Точка монтирования файловых систем, содержащих неизменяемые файлы, которые предназначены для совместного использования различными компьютерами (например, исполняемые файлы и текстовые файлы ASCII). На автономном компьютере каталог /usr служит точкой монтирования отдельной локальной файловой системы. В бездисковых компьютерах и компьютерах с небольшим объемом дисковой памяти каталог /usr служит для монтирования файловой системы удаленного сервера.
/var	Точка монтирования файлов, уникальных для каждой машины. Каталог /var рассматривается в качестве файловой системы, так как число содержащихся в нем файлов постоянно увеличивается. Например, он содержит символьную связь с каталогом /usr/tmp, в котором расположены временные рабочие файлы.

Корневая файловая система:

Корневая файловая система расположена в основе иерархического дерева файлов. Она включает файлы и каталоги, необходимые для работы системы, в том числе каталог устройств и программы загрузки системы. Кроме того, в корневую файловую систему входят точки монтирования, позволяющие добавлять в иерархию другие файловые системы.

На следующем рисунке показаны основные каталоги корневой файловой системы.

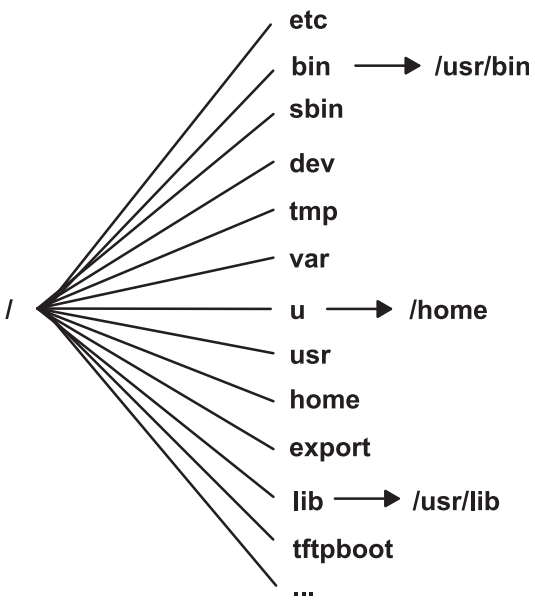


Рисунок 8. Корневая файловая система. На схеме показана корневая файловая система и ее каталоги. Каталог `/bin` указывает на каталог `/usr/bin`. Каталог `/lib` указывает на каталог `/usr/lib`. Каталог `/u` указывает на каталог `/home`.

В следующем списке приведена информация о содержимом каталогов корневой файловой системы (/).

Элемент	Описание
`/etc`	Содержит файлы с информацией о конфигурации, уникальные для каждого компьютера. В приведенных примерах этот каталог содержит: • `/etc/hosts` • `/etc/passwd` Каталог `/etc` содержит административные файлы. Большая часть команд, ранее хранившихся в каталоге `/etc`, теперь находится в каталоге `/usr/sbin`. Однако в целях совместимости в каталоге `/etc` сохранены символьные связи с некоторыми исполняемыми файлами. В приведенных примерах этот каталог содержит следующие файлы: • `/etc/chown` - символьная связь с каталогом `/usr/bin/chown`. • `/etc/exportvg` - символьная связь с каталогом `/usr/sbin/exportvg`.
`/bin`	Символьная связь с каталогом `/usr/bin`. В предыдущих файловых системах UNIX в каталоге `/bin` хранились пользовательские команды, которые теперь находятся в каталоге `/usr/bin`.
`/sbin`	Содержит файлы, предназначенные для загрузки компьютера и монтирования файловой системы `/usr`. Большинство команд, выполняемых при загрузке, расположены в дисковой файловой системе RAM загрузочного образа, поэтому каталог `/sbin` содержит небольшое число команд.
`/dev`	Содержит каталоги со специальными файлами локальных устройств. Каталог `/dev` содержит специальные файлы накопителей на магнитной ленте, принтеров, разделов дисков и терминалов.
`/tmp`	Точка монтирования файловых систем, содержащих временные файлы, создаваемые системой. Файловая система `/tmp` - это пустой каталог.
`/var`	Точка монтирования файлов, уникальных для каждой машины. Каталог `/var` настроен как независимая файловая система, так как число содержащихся в нем файлов постоянно увеличивается. Дополнительная информация приведена в разделе “Файловая система /var” на стр. 433.
`/u`	Символьная связь с каталогом `/home`.
`/usr`	Содержит неизменяемые файлы, которые могут использоваться совместно несколькими системами, например, исполняемые файлы и документация в формате ASCII. Автономные системы монтируют корневую часть локальной файловой системы в каталог `/usr`. На бездисковых компьютерах и компьютерах с небольшим объемом дисковой памяти каталог `/usr` служит для монтирования файловой системы удаленного сервера. Дополнительная информация о монтировании файловых систем в каталог “Файловая система /usr” на стр. 431 приведена в разделе `/usr`.

Элемент	Описание
/home	Точка монтирования файловых систем, содержащих домашние каталоги пользователей. Файловая система /home содержит файлы и каталоги отдельных пользователей. В автономных системах каталог /home представляет собой отдельную файловую систему. В сетевой среде на сервере могут храниться файлы пользователей, которые должны быть доступны нескольким компьютерам. В этом случае копия каталога /home сервера, монтируется в локальную файловую систему /home.
/export	Содержит каталоги и файлы сервера, предназначенные для удаленных клиентов. Дополнительная информация о дереве каталогов, расположенном в каталоге /export, приведена в разделе “Каталог /export” на стр. 434.
/lib	Символьная связь с каталогом /usr/lib. Дополнительная информация приведена в разделе “Файловая система /usr”.
/tftpboot	Содержит загрузочные образы и информацию о загрузке для бездисковых клиентов.

Файловая система /usr:

В файловой системе /usr хранятся исполняемые файлы, с которыми могут работать различные компьютеры.

Основные подкаталоги каталога /usr показаны на следующем рисунке.

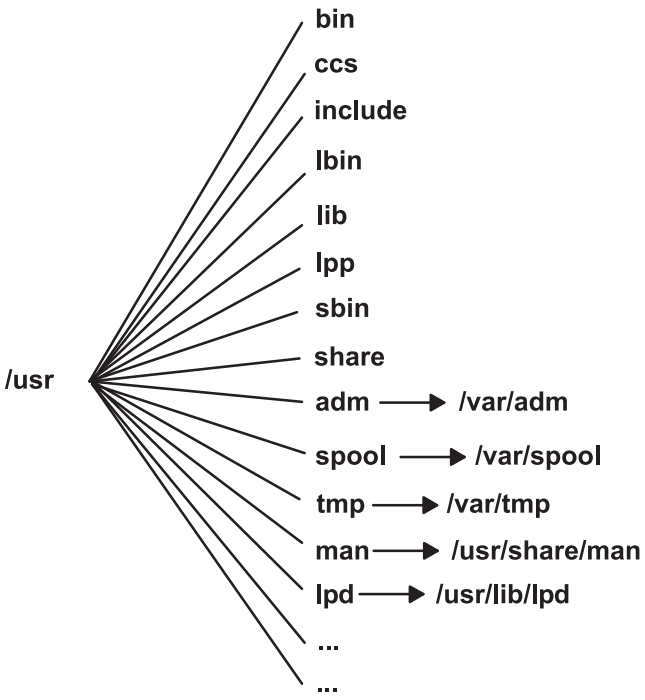


Рисунок 9. Файловая система /usr. На рисунке показаны основные подкаталоги каталога /usr: /bin, /ccs, /lib, /lpp, /adm с подкаталогом /var/adm и /man с подкаталогом /usr/share/man.

В автономных системах /usr представляет собой отдельную файловую систему, расположенную на логическом томе /dev/hd2. На компьютерах без дисков или с дисками небольшого объема поверх локальной файловой системы /usr монтируется каталог удаленного сервера, доступ к которому предоставляется только для чтения. В файловой системе /usr хранятся команды, библиотеки и данные, предназначенные только для чтения.

Все файлы и каталоги файловой системы /usr, за исключением /usr/share, могут использоваться любыми компьютерами с той же аппаратной архитектурой.

Файловая система /usr содержит следующие каталоги:

Элемент	Описание
/usr/bin	Содержит обычные команды и сценарии оболочки. Например, в каталоге /usr/bin хранятся команды **ls**, **cat** и **mkdir**.
/usr/ccs	Содержит двоичные пакеты для разработки программ.
/usr/include	Содержит включаемые файлы, или файлы заголовков.
/usr/lbin	Содержит исполняемые файлы команд.
/usr/lib	Содержит независимые от архитектуры библиотеки в формате lib*.a. Каталог /lib в корневом каталоге / - это символьная связь с каталогом /usr/lib, поэтому все файлы, помещаемые в каталог /lib, фактически хранятся в каталоге /usr/lib. В целях совместимости в этом каталоге находятся несколько не библиотечных файлов.
/usr/lpp	Содержит дополнительные программные продукты.
/usr/sbin	Содержит утилиты управления системой, включая команды Инструмента управления системой (SMIT). Большая часть команд, раннее хранившихся в каталоге /etc, теперь находится в каталоге /usr/sbin.
/usr/share	Содержит файлы, применяемые компьютерами с различной архитектурой. Дополнительная информация приведена в разделе “Каталог /usr/share”.

Ниже описаны символьные связи с каталогом /var:

Элемент	Описание
/usr/adm	Символьная связь с каталогом /var/adm
/usr/mail	Символьная связь с каталогом /var/spool/mail
/usr/news	Символьная связь с каталогом /var/news
/usr/preserve	Символьная связь с каталогом /var/preserve
/usr/spool	Символьная связь с каталогом /var/spool
/usr/tmp	Символьная связь с каталогом /var/tmp; каталог /usr может совместно использоваться несколькими узлами и монтироваться в режиме только для чтения.

Ниже описаны символьные связи с каталогами /usr/share и /usr/lib:

Элемент	Описание
/usr/dict	Символьная связь с каталогом /usr/share/dict
/usr/man	Символьная связь с каталогом /usr/share/man
/usr/lpd	Символьная связь с каталогом /usr/lib/lpd.

Каталог /usr/share:

В каталоге /usr/share хранятся текстовые файлы, общие для компьютеров с различной архитектурой. К файлам этого каталога могут обращаться компьютеры с различной аппаратной архитектурой.

В среде со смешанной архитектурой бездисковые клиенты обычно монтируют один каталог сервера в своем каталоге /usr, а другой - в каталоге /usr/share. Файлы в каталоге /usr/share входят в состав одного или нескольких пакетов, устанавливаемых отдельно. Таким образом, узлу могут соответствовать разные части каталога /usr: это зависит от того, были ли они установлены локально при монтировании каталога сервера в каталоге /usr/share.

На следующем рисунке показаны некоторые файлы и подкаталоги каталога /usr/share.

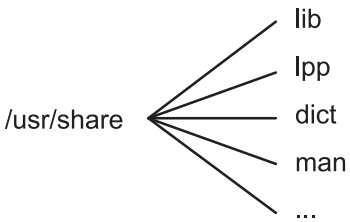


Рисунок 10. Каталог /usr/share.

На данном рисунке показаны следующие подкаталоги каталога /usr/share: /lib, /lpp, /dict и /man.

Каталог /usr/share содержит следующие файлы и подкаталоги:

Элемент	Описание
/usr/share/man	Содержит справочные руководства, если они установлены в системе
/usr/share/dict	Содержит орфографический словарь и индексные файлы
/usr/share/lib	Содержит файлы данных, не зависящие от архитектуры, такие как terminfo, learn,tmac, me и macros
/usr/share/lpp	Содержит данные и информацию о программных продуктах, которые можно дополнительно установить в системе.

Файловая система /var:

Размер файловой системы /var может существенно увеличиваться в процессе работы системы, поскольку в ней хранятся подкаталоги и файлы данных, применяемые активно работающими приложениями, такими как система учета ресурсов, программа обработки почты и программа буферизации печати.

Внимание: Если приложения, работающие в системе, активно применяют файловую систему /var, то увеличьте ее размер или регулярно вызывайте команду **skulker**. По умолчанию размер файловой системы /var составляет 4 МБ.

Некоторые файлы из каталога /var, например, /var/adm/wtmp и /var/adm/ras/errlog, необходимо регулярно просматривать.

Ниже перечислены другие файлы из каталога /var, которые необходимо просматривать:

Элемент	Описание
/var/adm/ras/trcfile	Если включена трассировка
/var/tmp/snmpd.log	Если в системе работает команда **snmpd**.

На рисунке "Каталог /var" показаны некоторые каталоги файловой системы /var.

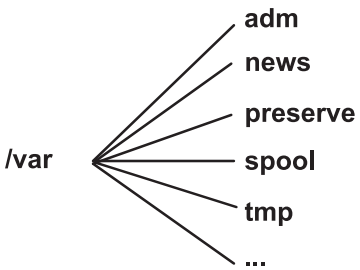


Рисунок 11. Каталог /var. На рисунке показаны основные подкаталоги каталога /var: /adm, /news , /preserve, /spool и /tmp.

Элемент	Описание
/var/adm	Содержит файлы протоколов и учета ресурсов
/var/news	Содержат сообщения конференций
/var/preserve	Содержит данные, сохраненные после прерванных сеансов редактирования; выполняет те же функции, что и каталог /usr/preserve в предыдущих выпусках
/var/spool	Содержит файлы таких программ, как программа обработки электронной почты; выполняет те же функции, что и каталог /usr/spool в предыдущих выпусках
задания	Содержит временные файлы; выполняет те же функции, что и каталог /usr/tmp в предыдущих выпусках. Каталог /usr/tmp теперь представляет собой символьную связь с каталогом /var/tmp.

Каталог /export:

Каталог /export содержит файлы сервера, импортируемые клиентами, например, бездисковыми компьютерами, компьютерами без данных или с недостаточным объемом памяти.

Сервер может экспортировать различные типы данных, хранящихся на диске, включая пакеты исполняемых программ, пространство подкачки и корневые файловые системы для бездисковых клиентов и клиентов с недостаточным объемом дискового пространства. Обычно все подобные каталоги и файлы находятся в каталоге /export. В следующем списке перечислены некоторые подкаталоги каталога /export:

/exec Содержит каталоги, монтируемые бездисковыми клиентами в файловых системах /usr.

/swap Содержит файлы подкачки бездисковых клиентов.

/share Содержит каталоги, монтируемые бездисковыми клиентами в каталоге /usr/share.

/root Содержит каталоги, монтируемые бездисковыми клиентами в корневых файловых системах /.

/home Содержит каталоги, монтируемые бездисковыми клиентами в файловых системах /home.

В каталоге /export по умолчанию хранятся ресурсы, которые предоставляются бездисковым клиентам для выполнения команд. Каталог /export - это единственный каталог сервера, в котором хранятся ресурсы клиентов. Поскольку клиенты монтируют эти ресурсы в свое собственное дерево каталогов, эти ресурсы с точки зрения клиентов выглядят как обычные компоненты локального дерева каталогов. Ниже перечислены основные подкаталоги каталога /export и соответствующие точки монтирования в дереве файлов клиента:

/export/root
Этот каталог монтируется в корневой файловой системе клиента (/). По умолчанию корневые каталоги клиентов расположены в каталоге /export/root и носят имена хостов клиентов.

/export/exec
Другое название - каталог общего дерева продуктов (SPOT). Этот каталог монтируется в файловой системе /usr клиента. SPOT - это версии файловой системы /usr, хранящиеся в каталоге /export/exec под именами, соответствующими их номеру выпуска. По умолчанию применяется имя RISCAIX.

/export/share
Этот каталог монтируется в файловой системе /usr/share клиента. Он содержит данные, которые могут применяться компьютерами с различной архитектурой. Расположение каталога по умолчанию - /export/share/AIX/usr/share.

/export/home
Этот каталог монтируется в файловой системе /home клиента. Он содержит пользовательские каталоги с именами хостов клиентов. По умолчанию расположение домашних каталогов клиентов - /export/home.

/export/swap
Другое название этого каталога - каталог подкачки. В автономных системах или системах без данных пространство подкачки расположено на локальном диске; для бездисковых клиентов пространство подкачки представляет собой файл сервера. Этот файл носит имя хоста клиента и по умолчанию расположен в каталоге /export/swap.

/export/dump
В автономных системах в качестве устройства дампа применяется локальный диск; в бездисковых клиентах эту роль выполняет файл сервера. По умолчанию файл находится в подкаталоге каталога /export/dump, причем имя этого подкаталога совпадает с именем хоста клиента.

microcode
Этот каталог содержит микрокод для физических устройств. Расположение каталога по умолчанию - /export/exec/RISCAIX/usr/lib/microcode.

Шифрование файловых систем JFS2:

Начиная с AIX версии 6.1, Зашифрованная файловая система (EFS) поддерживается на файловых системах JFS2. EFS позволяет зашифровывать данные и управлять доступом к ним посредством защиты по ключу.

Ключ предоставляется каждому пользователю и сохраняется в защищенном хранилище ключей в зашифрованном виде. Благодаря удачному паролю, ключи пользователя загружаются в ядро и связываются с разрешениями процесса. Для открытия файла, защищенного EFS, сначала проверяются разрешения процесса. Если процесс обнаруживает ключ, соотвествующий файловой защите, то процесс расшифровывает ключ и содержимое файла.

Файловые системы JFS2 по умолчанию не включаются с помощью EFS. Файловая система JFS2 должна включаться с помощью EFS до начала шифрования данных. Дополнительная информация о команде подключения с помощью EFS **efsenable** приведена в разделе *Справочник по командам, том 2*.

Настройка файловых систем

При добавлении или настройке файловых систем вы можете выбрать необходимые опции в контейнере Файловые системы команд быстрого доступа SMIT.

Команды быстрого доступа SMIT приведены в следующей таблице:

Таблица 64. Задачи управления логическими томами и файловыми системами

Задача	*Команда быстрого доступа SMIT*
Добавить JFS или JFS2	**smit crfs**
Добавить JFS2 к существующему логическому тому	**smit crjfs2lvstd**
Меню добавления JFS к уже определенному логическому тому	Создайте логический том и введите команду **smit crjfslv**
Изменить атрибуты JFS или JFS2[Примечание 1]	**smit chfs**
Проверить размер файловой системы	**smit fs**
Увеличить размер файловой системы	JFS: **smit chjfs** JFS2: **smit chjfs2**
Уменьшить размер файловой системы	JFS2: **smit chjfs2**

Примечание: Команда SMIT для **уменьшения размера файловой системы** применима только к JFS2.

Управление файловой системой

Файловая система представляет собой полную структуру каталогов, содержащую корневой каталог и, возможно, подкаталоги и файлы, расположенные ниже в иерархической структуре.

Каждая файловая система расположена на отдельном логическом томе. Перечисленные ниже задачи управление системой имеют отношение к файловым системам:

- Выделение областей памяти на логических томах для файловых систем
- Создание файловых систем
- Предоставление пользователям доступа к файловой системе
- Отслеживание использования памяти, выделенной файловой системе
- Резервное копирование файловых систем для защиты от системных сбоев и потери данных

- Создание моментальной копии файловой системы.
- Поддержание файловых систем в согласованном состоянии.

Ниже приведен список наиболее часто применяемых команд работы с файловыми системами:

Элемент	Описание
backup	Выполняет полное или дополняющее резервное копирование файловой системы
chfs -a splitcopy	Создает оперативную резервную копию смонтированной файловой системы JFS
дд	Копирует данные непосредственно с одного устройства на другое; применяется для создания резервных копий файловых систем
df	Показывает отчет об использованном и свободном пространстве в файловых системах
fsck	Проверяет файловые системы и устраняет ошибки
mkfs	Создает файловую систему указанного размера на заданном логическом томе
mount	Подключает файловую систему к дереву файлов системы, обеспечивая доступ к файлам и каталогам этой файловой системы
restore	Восстанавливает файлы из резервной копии
snapshot	Создает моментальную копию файловой системы JFS2
umount	Удаляет файловую систему из дерева файлов системы. В результате файлы и каталоги файловой системы становятся недоступными.

Просмотр объема свободной памяти в файловой системе (команда df):

С помощью команды **df** можно узнать общий объем памяти, выделенный файловой системе, и объем свободной памяти. В параметре **Файловая_система** задается имя устройства, на котором расположена файловая система, точка монтирования файловой системы или относительный путь к файловой системе.

Если параметр **Файловая_система** не задан, то команда **df** выводит информацию обо всех файловых системах, смонтированных в настоящее время. Если в этом параметре задан файл или каталог, то команда **df** показывает информацию о файловой системе, к которой он относится.

Как правило, команда **df** использует использует счетчики, расположенные в главном блоке. В некоторых исключительных случаях счетчики могут содержать ошибочные данные. Например, если во время выполнения команды **df** файловая система активно изменялась, то счетчик свободной памяти может содержать неверную информацию.

Сведения о синтаксисе приведены в описании команды **df** в книге *Справочник по командам, том 2*.

Примечание: В некоторых удаленных файловых системах, например, Сетевой файловой системе (NFS), столбцы с информацией об объеме свободной памяти будут пустыми, если соответствующая информация не была предоставлена сервером.

Ниже приведены примеры применения команды **df**:

- Для просмотра информации обо всех смонтированных файловых системах введите:
  ```
  df
  ```
  Если каталоги /, /usr, /site и /usr/venus находятся в разных файловых системах, то вывод команды **df** будет выглядеть примерно следующим образом:
  ```
  Filesystem 512-blocks  free   %used   Iused %Iused   Mounted on
  /dev/hd4    20480     13780    32%      805   13%     /
  /dev/hd2   385024     15772    95%    27715   28%     /usr
  /dev/hd9var 40960     38988     4%      115    1%     /var
  /dev/hd3    20480     18972     7%       81    1%     /tmp
  /dev/hd1     4096      3724     9%       44    4%     /home
  ```
- Для просмотра сведений об объеме свободной памяти в файловой системе текущего каталога введите:
  ```
  df .
  ```

Команды файловой системы:

Существует набор команд для работы с файловой системой, не зависящий от ее типа.

Файл `/etc/filesystems` контролирует список файловых систем, которыми можно управлять с помощью следующих команд:

Элемент	Описание
chfs	Изменяет параметры файловой системы
crfs	Создает файловую систему
lsfs	Показывает параметры файловой системы
rmfs	Удаляет файловую систему
mount	Предоставляет доступ к файловой системе

Следующие четыре команды предназначены для работы с виртуальными файловыми системами. Файл `/etc/vfs` содержит информацию о типах файловой системы, которыми можно управлять с помощью следующих команд:

Элемент	Описание
chvfs	Изменяет параметры типа файловых систем
crvfs	Создает новый тип файловых систем
lsvfs	Показывает параметры типа файловых систем
rmvfs	Удаляет тип файловых систем

Сравнение файловых систем в различных компьютерах:

Если файловые системы, находящиеся на двух разных компьютерах, должны совпадать, но вы подозреваете, что одна из них повреждена, то можно сравнить эти файловые системы.

В этом разделе описана процедура сравнения атрибутов одноименных файловых систем, одна их которых находится на локальном хосте (*исходный-хост*), а другая - на удаленном хосте.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Войдите в систему удаленного хоста под именем root. Например:

   ```
   tn juniper.mycompany.com


   AIX Версии 6.1
   (C) Copyrights by IBM and by others 1982, 2002.
   Имя: root
   Пароль root:
   ```

2. Откройте в текстовом редакторе файл `.rhosts` на удаленном хосте и добавьте в него раздел, разрешающий пользователю root запускать удаленные команды в защищенном режиме. Формат этого раздела:

   исходный-хост root

   Полученный файл `.rhosts` может выглядеть, например, следующим образом:

   ```
   NIM.mycompany.com root
   nim.mycompany.com root
   host.othernetwork.com root
   исходный-хост.mycompany.com root
   ```

3. Сохраните файл и закройте соединение.
4. Войдите в систему *исходного-хоста* под именем пользователя с правами доступа root и создайте с помощью текстового редактора еще один файл. В нашем случае этот файл называется `compareFS`. Например:

```
vi compareFS
```

5. Добавьте в этот файл следующий текст, где *имя-FS* - это имя сравниваемой файловой системы, а *удаленный-хост* - имя хоста, на котором находится сравниваемая файловая система:

   ```
   имя-FS -> удаленный-хост
   install -v ;
   ```

   Примечание: В строке с командой **install** в этом файле между параметром `-v` и точкой с запятой (`;`) должен быть пробел.

   Например:

   ```
   /home/jane/* -> juniper.mycompany.com
   install -v ;
   ```

6. Сохраните файл и закройте текстовый редактор. Файл `compareFS` будет применяться в качестве *distfile* в команде **rdist** на следующем шаге.
7. Введите в командной строке:

   ```
   /usr/bin/rdist -f compareFS
   ```

   Если вы ожидаете большой объем вывода, означающий существенное различие файловых систем, то укажите имя файла для сохранения вывода. Например:

   ```
   /usr/bin/rdist -f compareFS > compareFS_output
   ```

В выводе будут указаны различия между файловыми системами.

Информация, связанная с данной:

Команда rdist

Файловый формат rhosts для TCP/IP

Описание защиты удаленных команд

Обслуживание файловых систем

В следующей таблице перечислены простейшие задачи по обслуживанию файловых систем.

Таблица 65. Задачи обслуживания файловых систем

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Резервное копирование файлов и каталогов по имени	**smit backfile**	**backup** Прим. 1
Создание и резервное копирование моментальной копии JFS2	**smit backsnap**	**backsnap** Прим. 1
Просмотр списка всех файловых систем на диске	**smit lsmntdsk**	
Просмотр списка файловых систем на съемном диске	**smit lsmntdsk**	
Просмотр списка смонтированных файловых систем	**smit fs**	
Монтирование групп файловых систем Прим. 5	**smit mountg**	**mount -t** *группа*
Монтирование JFS или JFS2 Прим. 3	**smit mountfs**	**mount**
Монтирование моментальной копии JFS2	**smit mntsnap**	**mount -v jfs2 -o snapshot** Устройство *Точка-монтирования*
Удаление JFS или JFS2	**smit rmfs**	
Удаление моментальной копии JFS2	**smit rmsnap**	**snapshot -d** *устройство-моментальной-копии*
Восстановление файловой системы JFS2 с помощью моментальной копии	**smit rollbacksnap**	**rollback** [**-s**] [**-v**] [**-c**] *моментальная-копия-FS объект-моментальной-копии*
Размонтирование файловой системы Прим. 4	**smit umountfs**	
Размонтирование файловой системы на съемном диске Прим. 4	**smit umntdsk**	

Таблица 65. Задачи обслуживания файловых систем (продолжение)

Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Размонтирование группы файловых систем[Прим. 5]	**smit umountg**	**umount -t** *группа*
Работа с квотами для расширенных JFS	**smit j2fsquotas**	
Включение или выключение функции управления квотами	**smit j2enablequotas**	
Прекращение или возобновление действия квот	**smit j2enforcequotas**	**quotaon|off -v**
Просмотр информации об использовании квот	**smit j2repquota**	**repquota -v**
Обновление статистики использования блоков дисков и файлов	**smit j2quotacheck**	**quotacheck -v**
Добавление класса ограничений	**smit j2addlimit**	**j2edlimit -e**
Изменение или просмотр свойств класса ограничений	**smit j2changelimit**	
Настройка класса ограничений в качестве ограничений по умолчанию для файловой системы	**smit j2defaultlimit**	
Назначение класса ограничений пользователю или группе	**smit j2assignlimit**	
Просмотр классов ограничений, связанных с файловой системой	**smit j2listlimits**	**j2edlimit -l '-u'**
Удаление класса ограничений	**smit j2removelimit**	

Примечание:

1. Информация об опциях приведена в описании команд.
2. Не следует изменять имена наиболее важных файловых систем, а именно, / (корневая) на логическом томе (hd4), `/usr` на hd2, `/var` на hd9var, `/tmp` на hd3 и `/blv` на hd5. Если вы следуете правилу присвоения имен hd*n*, то начните нумерацию с hd10.
3. Перед монтированием проверьте файловую систему с помощью процедуры “Проверка файловой системы” на стр. 442 или команды **fsck**.
4. Ошибки при размонтировании файловых систем чаще всего вызваны тем, что какие-либо файлы этих файловых систем заняты пользователями или процессами. Для того чтобы узнать имя пользователя или процесса, открывшего файл, выполните команду **fuser**.
5. Группа файловых систем - это набор файловых систем, у которых в файле `/etc/filesystems` задано одинаковое значение параметра **type=**.

Задачи, связанные с данной:

“Монтирование пространства с другого диска” на стр. 375
Вы можете увеличить объем пространства на диске путем монтирования пространства с другого диска.

Восстановление одного или нескольких файлов из активной моментальной копии JFS2:

При повреждении файла вы можете заменить его копией из активной моментальной копии JFS2.

Для восстановления одного или нескольких файлов из изображения активной внешней моментальной копии JFS2 воспользуйтесь следующей процедурой.

В данном случае следует предположить, что `/home/aaa/myfile` является поврежденным файлом в файловой системе `/home`.

1. Смонтируйте моментальную копию с помощью следующей командыMount the snapshot with a command similar to the following:

```
mount -v jfs2 -o snapshot /dev/mysnaplv /tmp/mysnap
```

2. Перейдите в каталог моментальной копии с помощью следующей команды:
   ```
   cd /tmp/mysnap
   ```
3. Скопируйте правильный файл из моментальной копии для перезаписи поврежденного файла с помощью следующей команды:
   ```
   cp aaa/myfile /home/aaa/myfile
   ```

   Эта команда копирует только файл с именем `myfile`. При копировании всех файлов из моментальной копии в каталог `aaa` воспользуйтесь следующей командой:
   ```
   cp -R aaa /home/aaa
   ```

   Дополнительные примеры по замене поврежденных файлов с изображениями моментальных копий приведены в описаниях команд **cp**, **cpio** в книге *Справочник по командам, том 1*.

Восстановление одного или нескольких файлов из активной внутренней моментальной копии JFS2:

При повреждении файла вы можете заменить его копией из активной внутренней моментальной копии JFS2.

Для восстановления одного или нескольких файлов из изображения активной внутренней моментальной копии JFS2 воспользуйтесь следующей процедурой.

В данном случае следует предположить, что `/home/aaa/myfile` является поврежденным файлом в файловой системе `/home`.

1. Перейдите в каталог моментальной копии с помощью следующей команды:
   ```
   cd /home/.snapshot/mysnap
   ```
2. Скопируйте правильный файл из моментальной копии для перезаписи поврежденного файла с помощью следующей команды:
   ```
   cp aaa/myfile /home/aaa/myfile
   ```

   Эта команда копирует только файл с именем `myfile`. При копировании всех файлов из моментальной копии в каталог `aaa` воспользуйтесь следующей командой:
   ```
   cp -R aaa /home/aaa
   ```

   Дополнительные примеры по замене поврежденных файлов с изображениями моментальных копий приведены в описаниях команд **cp**, **cpio** в книге *Справочник по командам, том 1*.

Файловые системы на компакт-дисках и DVD:

Компакт-диски и диски DVD не монтируются автоматически, если не включена соответствующая функция.

Для включения этой функции смонтируйте файловую систему CDRFS или UDFS с помощью команды **cdmount**, например:
```
cdmount cd0
```

Файловую систему UDFS, доступную для чтения и записи, можно смонтировать вручную с помощью следующей команды:
```
mount -V udfs устройство точка-монтирования
```

Здесь *устройство* - это имя устройства DVD, а *точка-монтирования* - это точка монтирования файловой системы.

Применение файловых систем на перезаписываемых компакт-дисках:

Файловые системы CDRFS и JFS можно использовать с перезаписываемых оптических носителей.

Файловая система CD-ROM (CDRFS) может применяться на перезаписываемом оптическом носителе, защищенном от записи, так же, как на диске CD-ROM. В следующей таблице описаны процедуры добавления, монтирования и размонтирования файловой системы CDRFS на перезаписываемом оптическом носителе. Для монтирования файловой системы нужно задать следующие параметры:

Элемент	Описание
Имя устройства	Определяет имя устройства, в котором размещен носитель.
Точка монтирования	Каталог монтирования файловой системы.
Автоматическое монтирование	Указывает, должна ли файловая система монтироваться автоматически при перезапуске системы.

Задачи работы с CDRFS на оптическом носителе		
Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Добавление файловой системы CDRFS[1]	**smit crcdrfs**	1. Добавление файловой системы: **crfs -v cdrfs -p ro -d***Имя-устройства* **-m** *Точка-монтирования* **-A** *Автоматическое-монтирование* 2. Монтирование файловой системы: **mount** *Точка-монтирования*
Удаление файловой системы CDRFS[2]	1. Размонтирование файловой системы: **smit umountfs** 2. Удалите файловую систему:**smit rmcdrfs**	1. Размонтировать файловую систему: **umount** *Файловая-система* 2. Удаление файловой системы: **rmfs** *Точка-монтирования*

Примечание:

- Убедитесь, что перезаписываемый оптический носитель защищен от записи.
- Перед извлечением перезаписываемого оптического носителя необходимо размонтировать файловую систему CDRFS.

JFS - это файловая система для чтения и записи, применяемая как на перезаписываемых оптических носителях, так и на жестких дисках. Для создания и импорта файловой системы, расположенной на оптическом носителе, у вас должны быть системные права доступа (вы должны входить в группу system). Кроме того, вам потребуется следующая информация:

Имя группы томов
Задает имя группы томов

Имя устройства
Задает логическое имя оптического диска

Точка монтирования
Задает каталог, в котором будет смонтирована файловая система.

Автоматическое монтирование
Указывает, должна ли файловая система автоматически монтироваться при запуске системы

Примечание:

- Любая группа томов, создаваемая на оптическом носителе, должна целиком располагаться на этом носителе. Группа томов не может размещаться на нескольких перезаписываемых оптических дисках.
- При обращении к журнализированной файловой системе не обязательно указывать имя группы томов, которое задавалось при создании этой группы томов.

Задачи работы с JFS на оптических носителях		
Задача	*Команда быстрого доступа SMIT*	*Команда или файл*
Добавить JFS	1. Поместите оптический диск в дисковод. 2. Создать группу томов (при необходимости): **smit mkvg** 3. Создать журнализированную файловую систему: **smit crfs**	1. Поместите оптический диск в дисковод. 2. Создать группу томов (при необходимости): **mkvg -f -y** *Имя-группы-томов* **-d 1** *Имя-устройства* 3. Создать журнализированную группу томов: **crfs -v jfs -g** *Имя-группы-томов* **-a size=***Размер-файловой-системы* **-m** *Точка-монтирования* **-A** *Автоматическое-монтирование* **-p rw** 4. Монтирование файловой системы: **mount** *Точка-монтирования*
Обращение к существующей JFS[Прим. 1]	1. Поместите оптический диск в дисковод. 2. Импортировать группу томов: **smit importvg**	1. Поместите оптический диск в дисковод. 2. Импортировать группу томов: **importvg -y** *Имя-группы-томов Имя-устройства* 3. Смонтировать файловую систему: **mount** *Точка-монтирования*
Удаление JFS[Прим. 2]	1. Размонтирование файловой системы: **smit umountfs** 2. Удаление файловой системы: **smit rmjfs**	1. Размонтировать файловую систему: **umount** *Файловая-система* 2. Удаление файловой системы: **rmfs** *Точка-монтирования*

Примечание:

- Эту процедуру необходимо выполнять каждый раз при помещении носителя с журнализированной файловой системой в дисковод.
- Удаление журнализированной файловой системы приведет к уничтожению всех данных, содержащихся в этой файловой системе и на оптическом носителе.

Проверка файловой системы:

При сбое диска или остановке работающей системы со смонтированными файловыми системами возможно появление несоответствий в файловых системах. В этом случае, перед монтированием файловых систем их необходимо проверить.

Кроме того, проверка файловых систем требуется в следующих ситуациях:

- После сбоя; например, если пользователь не может перейти в доступный для него каталог.
- Перед резервным копированием файловой системы, чтобы во время ее восстановления не возникали ошибки и неполадки.
- Во время установки и загрузки системы, чтобы убедиться, что в файлах операционной системы нет ошибок.

Проверка пользовательской файловой системы:

Для проверки пользовательской файловой системы выполните следующие действия:

1. Перед проверкой размонтируйте пользовательскую файловую систему.
2. Убедитесь, что у вас есть права на запись в файловую систему. В противном случае команде **fsck** не удастся восстановить поврежденные файлы, даже если вы выберете опцию восстановления таких файлов.
3. Выполните команду **smit fsck** для перехода к меню **Проверить файловую систему**.
4. Выполните одно из следующих действий:
   - В поле **Имя файловой системы для проверки** укажите имя файловой системы, которую необходимо проверить, либо

- В поле **Тип файловой системы** укажите тип проверяемой файловой системы, например, журнализированная файловая система (jfs).

5. Если вы хотите ограничиться лишь наиболее вероятными повреждениями, то укажите значение Да в поле **Быстрая проверка?**. При выборе опции быстрой проверки просматриваются лишь те файловые системы, в которых повреждения наиболее вероятны, например, файловые системы, которые были смонтированы в момент прерывания работы системы.
6. В поле **Рабочий файл** укажите имя временного файла из файловой системы, которая не будет проверяться.
7. Запустите процедуру проверки.

Проверка файловых систем root и /usr:

Для запуска команды **fsck** с целью проверки файловой системы / или /usr необходимо завершить работу системы и загрузиться со съемного носителя, поскольку файловые системы / и /usr нельзя размонтировать на работающей системе.

В приведенной ниже процедуре описан запуск команды **fsck** для файловых систем / и /usr из служебной оболочки.

1. Завершите работу системы. (Необходим доступ Root)
2. Загрузка с установочного носителя.
3. В меню **Приветствие** выберите опцию **Обслуживание**.
4. В меню **Обслуживание** выберите опцию доступа к группе томов.
5. Выберите группу томов rootvg. Появится список логических томов, относящихся к выбранной группе томов.
6. Выберите опцию **2** для доступа к группе томов и запуска оболочки перед монтированием файловых систем. Далее вы запустите команду **fsck** с соответствующими опциями и именами устройств файловых систем. Команда **fsck** проверит согласованность файловой системы и динамически исправит в ней ошибки. С файловой системой / (корневая) связано устройство /dev/hd4, а с файловой системой /usr - устройство /dev/hd2.
7. Для проверки файловой системы / введите следующую команду:

   ```
   $ fsck -y /dev/hd4
   ```

   Если вы не слишком опытный пользователь, то укажите флаг **-y** (см. описание команды **fsck**).
8. Для проверки файловой системы /usr введите следующую команду:

   ```
   $ fsck -y /dev/hd2
   ```
9. Для проверки других файловых систем в rootvg введите команду **fsck** с соответствующим именем устройства. С файловой системой /tmp связано устройство /dev/hd3, а с файловой системой /var - устройство /dev/hd9var.
10. По окончании проверки файловых систем перезагрузите систему.

Уменьшение размера файловой системы в корневой группе томов:

Простейший способ сокращения размера *всех* файловых систем до их минимального размера заключается в указании в опции **Сократить** значения **да** при восстановлении базовой операционной системы из резервной копии.

Простейший способ сокращения размера *всех* файловых систем до их минимального размера заключается в указании в опции **Сократить** значения **да** при восстановлении базовой операционной системы из резервной копии. Опцию **Сократить** нельзя применять совместно с описанной ниже процедурой. Если после выполнения описанной ниже процедуры вы укажете в опции **Сократить** значение **да**, то изменения, внесенные в файл /image.data, будут переопределены при установке.

Данная процедура позволяет вручную сократить размер выбранной файловой системы rootvg. Вы должны указать файловую систему, не полностью использующую выделенное ей дисковое пространство, а затем изменить объем выделенного пространства, освободив часть дискового пространства в корневой группе томов. В ходе процедуры необходимо будет создать резервную копию групп томов и переустановить операционную систему с учетом внесенных изменений.

Внимание: Данная процедура требует завершения работы системы и переустановки операционной системы. При переустановке операционной системы операцию следует планировать на время, соответствующее минимальной загрузке системы. Перед переустановкой операционной системы необходимо создать надежные резервные копии данных, а также всех пользовательских приложений и групп томов.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Создайте отдельную резервную копию всех файловых систем, *не* входящих в rootvg. Отдельные резервные копии позволяют гарантировать целостность файловых систем.
2. Войдя в систему под именем пользователя с правами доступа root, проверьте, какие файловые системы в корневой группе томов не используют все выделенное им дисковое пространство:

   ```
   df -k
   ```

   Флаг **-k** показывает размер файловых систем в килобайтах. Вывод команды будет выглядеть примерно следующим образом:

   ```
   Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
   /dev/hd4           196608      4976   98%     1944     2% /
   /dev/hd2          1769472    623988   65%    36984     9% /usr
   /dev/hd9var        163840     65116   61%      676     2% /var
   /dev/hd3            65536     63024    4%      115     1% /tmp
   /dev/hd1            49152      8536   83%      832     7% /home
   /proc                   -         -    -        -     -  /proc
   /dev/hd10opt        32768     26340   20%      293     4% /opt
   ```

   По показанной информации видно, что в файловой системе, смонтированной в /usr, много свободных блоков и низок процент использования дискового пространства. Вы можете существенно увеличить число свободных блоков, сократив число разделов, выделенных файловой системе /usr.
3. Проверьте содержимое файла /etc/filesystems и убедитесь, что смонтированы все файловые системы rootvg. Если это не так, то они не будут включены в повторно установленную систему.
4. Создайте файл /image.data со списком всех активных файловых систем rootvg, которые должны быть включены в процедуру установки:

   ```
   mkszfile
   ```
5. Откройте файл /image.data в текстовом редакторе.
6. Найдите строку usr в разделе lv_data, относящемуся к файловой системе /usr. С помощью значений, указанных в этом разделе, определите, насколько вы можете сократить число логических разделов в файловой системе /usr. Размер дополнительных логических разделов по умолчанию определен в записи PP_SIZE файла /image.data. Файл /image.data будет выглядеть примерно следующим образом:

   ```
   lv_data:
    VOLUME_GROUP= rootvg
    LV_SOURCE_DISK_LIST= hdisk0
    LV_IDENTIFIER= 00042345d300bf15.5
    LOGICAL_VOLUME= hd2
    VG_STAT= active/complete
    TYPE= jfs
    MAX_LPS= 32512
    COPIES= 1
    LPs= 108
    STALE_PPs= 0
    INTER_POLICY= minimum
    INTRA_POLICY= center
    MOUNT_POINT= /usr
    MIRROR_WRITE_CONSISTENCY= on/ACTIVE
    LV_SEPARATE_PV= yes
   ```

```
PERMISSION= read/write
LV_STATE= opened/syncd
WRITE_VERIFY= off
PP_SIZE= 16
SCHED_POLICY= parallel
PP= 108
BB_POLICY= relocatable
RELOCATABLE= yes
UPPER_BOUND= 32
LABEL= /usr
MAPFILE=
LV_MIN_LPS= 70
STRIPE_WIDTH=
STRIP_SIZE=
```

Число логических разделов, выделенных данному логическому тому, равно 108 (LPs=108).

7. На основании результатов, полученных на шаге 2, определите число логических разделов, необходимых для размещения существующих данных в файловой системе /usr. С помощью следующей команды вы можете определить размер существующих файлов файловой системы /usr:

   ```
   df -k /usr
   ```

   Эта команда покажет те же значения (в килобайтах), что были указаны для файловой системы /usr на шаге 2.

   ```
   Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
   /dev/hd2          1769472    623988   65%    36984     9% /usr
   ```

   a. Вычтите число свободных блоков из общего числа выделенных блоков 1024.

      ```
      1769472 - 623988 = 1145484
      ```

   b. Добавьте оценку дополнительного пространства, учитывающего дальнейшее рост системы. Например, добавьте к результату 200000.

      ```
      1145484 + 200000 = 1345484
      ```

   c. Разделите результат на размер логического раздела в байтах (16*1024), получив в результате минимальное число необходимых логических разделов.

      ```
      1345484 / 16384 = 82.121826171875
      ```

      Округлив результат вверх, вы получите новое число логических разделов (LPs=83).

8. В файле image.data измените значение поля LPs с 108 на 83.
9. Найдите раздел fs_data, относящийся к файловой системе /usr. Раздел fs_data будет выглядеть примерно следующим образом:

   ```
   fs_data:
    FS_NAME= /usr
    FS_SIZE= 3538944
    FS_MIN_SIZE= 2290968
    FS_LV= /dev/hd2
    FS_FS= 4096
    FS_NBPI= 4096
    FS_COMPRESS= no
    FS_BF= false
    FS_AGSIZE= 8
   ```

10. Вычислите размер файловой системы (FS_SIZE), умножив размер физического раздела (PP_SIZE) на 2 (число используемых физическим разделом блоков по 512 байт) и на число логических разделов LPs). Используя значения из приведенного примера получаем следующий результат:

    ```
    PP_SIZE * блоки-512 * LPs = FS_SIZE
    16384 * 2 * 83 = 2719744
    ```

11. В файле image.data измените значение поля FS_SIZE с 3538944 на 2719744.
12. Вычислите минимальный размер файловой системы (FS_MIN_SIZE) на основе фактически используемого в настоящее время объема файловой системы /usr:

a. Вычислите минимальное число необходимых разделов. Используя значения из приведенного примера получаем следующий результат:

   ```
   size_in_use (см. шаг 7a) / PP_SIZE =
   число разделов
   1145484 / 16384 = 69.914794921875
   ```

b. Вычислите минимальный размер для полученного числа разделов. Округлив полученный на предыдущем шаге результат вверх до 70, получаем:

   ```
   PP_SIZE * блоки-512 * число разделов = FS_MIN_SIZE
   16384 * 2 * 70 = 2293760
   ```

13. В файле `image.data` измените значение поля `FS_MIN_SIZE` с `2290968` на `2293760`.
14. Сохраните файл и закройте текстовый редактор.
15. Размонтируйте все файловые системы, не входящие в группу томов rootvg.
16. Если существуют пользовательские группы томов, то введите следующие команды для их отключения и экспорта:

    ```
    varyoffvg VGName
    exportvg группа-томов
    ```

17. Загрузите магнитную ленту в накопитель на магнитной ленте и введите следующую команду создания полной резервной копии системы:

    ```
    mksysb /dev/rmt0
    ```

    В резервную копию такого типа включается информация о размере файловых систем из файла `/image.data`; в дальнейшем эта информация потребуется для переустановки системы с новыми значениями размера файловых систем.

    Примечание: Для инициализации резервной копии используйте только команду **mksysb**. В случае применения SMIT будет создан новый файл `image.data`, который переопределит внесенные вами изменения.

18. С помощью созданной резервной копии повторно установите операционную систему с опцией **Использовать текущие параметры системы**. Во время установки проверьте правильность следующих опций:
    - Для параметра **Восстановить исходное размещение** должно быть указано значение **нет**
    - Для параметра **Сократить файловые системы** должно быть указано значение **нет**

    Дополнительные сведения о процедуре установки приведены в разделе Установка резервных копий системы.
19. После установки операционной системы перезапустите систему в обычном режиме. Теперь размер файловой системы `/usr` изменен, но пользовательские файловые системы еще недоступны.
20. С помощью следующей команды смонтируйте все файловые системы:

    ```
    mount all
    ```

    Для уже смонтированных файловых систем могут быть показаны сообщения `Устройство занято`.

Теперь размер файловой системы `/usr` изменен, в корневой группе томов стало больше свободного дискового пространства и все файловые системы доступны.

Понятия, связанные с данным:

“Структура логических разделах” на стр. 385
Логические тома - это организованные области данных на физических томах.

Информация, связанная с данной:

Создание резервной копии корневой группы томов в файле или на магнитной ленте

Описание файла /image.data

Команда mkszfile

Команда mksysb

Устранение неполадок файловых систем

Следующие приемы устранения неполадок предназначены для разрешения некоторых наиболее распространенных неполадок, которые могут возникнуть в файловой системе. Если эта информация не помогла устранить неполадку, обратитесь в службу поддержки.

Устранение переполнения пользовательской файловой системы:

При переполнении пользовательской файловой системы выполните следующие действия.

1. Удалите старые резервные копии и файлы дампа. В приведенном ниже примере удаляются все файлы `*.bak`, `.*.bak`, `a.out`, `core`, `*` или `ed.hup`.

   ```
   find / \( -name "*.bak" -o -name core -o -name a.out -o \
           -name "...*" -o -name ".*.bak" -o -name ed.hup \) \
           -atime +1 -mtime +1 -type f -print | xargs -e rm -f
   ```

2. Для предотвращения переполнения диска запускайте команду **skulker** как часть процесса **cron** для удаления ненужных и временных файлов.

   Команда **skulker** удаляет файлы из каталога `/tmp`, файлы, созданные до указанной даты, файлы `a.out`, файлы `ed.hup` и файлы дампа. Эту команду можно выполнять ежедневно как часть процедуры учета, запускаемой командой **cron** в часы наименьшей загруженности системы (если система учета запущена).

   Программа-демон **cron** запускает команды оболочки в указанное время. В соответствии с инструкциями, приведенными в файлах `crontab`, можно настроить периодический запуск любых команд, например, команды **skulker**. Отправить файлы `crontab` можно с помощью команды **crontab**. Для изменения файла `crontab` необходимы права доступа root.

Задачи, связанные с данной:

“Настройка системы учета ресурсов” на стр. 167
Вы можете настроить систему учета ресурсов.

Устранение повреждений файловой системы:

Файловая система повреждается, если искажается информация об i-узлах или главных блоках в структуре каталогов.

Повреждение может быть вызвано аппаратной ошибкой или программой, которая неправильно работает с информацией об i-узлах или главных блоках. (Программы, написанные на ассемблере или C, могут без вызова системных функций напрямую обращаться к аппаратуре.) Одним из признаков поврежденной файловой системы является то, что в ней не удается выделить память или считать либо записать данные.

Для восстановления поврежденной файловой системы необходимо выполнить диагностику и устранить все найденные ошибки. Команда **fsck** выполняет диагностику низкого уровня и восстановление.

Ниже описана процедура устранения повреждений файловой системы:

1. Войдя в систему под именем пользователя с правами доступа root, размонтируйте поврежденную файловую команду с помощью одной из следующих команд SMIT: **smit unmountfs** (для файловой системы жесткого диска) или **smit unmntdsk** (для файловой системы съемного диска).
2. Определите повреждение файловой системы с помощью команды **fsck**. В приведенном ниже примере команда **fsck** выполняет проверку размонтированной файловой системы, расположенной в устройстве `/dev/myfilelv`:

   ```
   fsck /dev/myfilelv
   ```

   Команда **fsck** находит и восстанавливает поврежденные файловые системы. Обычно файловая система согласована, и команда **fsck** просто сообщает о количестве файлов и занятых и свободных блоков в ней. Если целостность файловой системы нарушена, то команда **fsck** выдает информацию об обнаруженных несоответствиях и запрашивает у вас разрешения исправить их. Команда **fsck** прежде всего пытается всеми возможными способами исправить ситуацию и избежать действий, которые могут привести к утере данных. Тем не менее, в некоторых случаях команда **fsck** вынуждена рекомендовать вам

уничтожить поврежденный файл. Список несоответствий, проверяемых командой **fsck** , приведен в описании этой команды в книге *Справочник по командам, том 2*.

3. Если повреждение файловой системы невозможно устранить, то восстановите ее с резервной копии.

   Внимание: Восстановление файловой системы из резервной копии уничтожает и заменяет любую файловую систему, хранившуюся ранее на данном диске.

   Для восстановления файловой системы из резервной копии воспользуйтесь командой SMIT **smit restfilesys** или следующей последовательностью команд:

   ```
   mkfs /dev/myfilelv
   mount /dev/myfilelv /myfilesys
   cd /myfilesys
   restore -r
   ```

   В данном примере команда **mkfs** создает новую файловую систему на устройстве `/dev/myfilelv`, а также инициализирует метку тома, метку файловой системы и начальный блок. Команда **mount** устанавливает точку монтирования `/dev/myfilelv` для файловой системы **myfilesys**, а команда **restore** восстанавливает файловую систему из резервной копии.

   Если резервная копию была создана с помощью дополняющего резервного копирования, то восстановление необходимо выполнять по мере возрастанию уровней резервных копий (например, 0, 1, 2). При выполнении процедуры **smit restfilesys** для восстановления полной файловой системы введите целевой каталог, устройство для восстановления (отличное от `/dev/rfd0`) и число блоков, считываемых за одну операцию ввода.

Задачи, связанные с данной:

“Восстановление пользовательских файлов из резервной копии” на стр. 31
При восстановлении данных из резервной копии наибольшую трудность представляет определение магнитной ленты, на которой записан нужный файл. С помощью команды **restore -T** можно просмотреть список содержимого архива. Рекомендуется восстанавливать файлы в каталог `/tmp`, чтобы случайно не перезаписать другие пользовательские файлы.

Исправление неправильной сигнатуры в главном блоке файловой системы:

В случае повреждения главного блока файловой системы доступ к этой файловой системе становится невозможен. Неправильную сигнатуру в главном блоке файловой системы можно исправить.

В большинстве случае восстановить поврежденный главный блок невозможно. Следующая процедура описывает восстановление главного блока в файловой системе JFS, если это повреждение связано с неправильной сигнатурой. В случае повреждения основного главного блока файловой системы JFS2 воспользуйтесь командой **fsck** для автоматического копирования дополнительного главного блока и восстановления основного главного блока.

В следующей процедуре предполагается, что `/home/myfs` - это файловая система JFS, находящаяся на физическом томе `/dev/lv02`.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

1. Размонтируйте файловую системе `/home/myfs`, в которой вы подозреваете повреждение:

   ```
   umount /home/myfs
   ```

2. Для того чтобы убедиться, что файловая система действительно повреждена, вызовите команду **fsck** для этой файловой системы. Пример:

   ```
   fsck -p /dev/lv02
   ```

   Если неполадка заключается в повреждении главного блока, то **fsck** вернет одно из следующих сообщений:

   ```
   fsck: Not an AIXV5 file system
   ```

   ИЛИ

```
Файловая система неизвестного типа
```

3. От имени пользователя с правами доступа root введите команду **od** для просмотра главного блока файловой системы:

   ```
   od -x -N 64 /dev/lv02 +0x1000
   ```

   Здесь флаг **-x** задает просмотр информации в шестнадцатеричном формате, а флаг **-N** указывает, что система должна форматировать не более 64 байт, начиная от параметра смещения (+), задающего точку начала вывода. Ниже приведен пример вывода:

   ```
   0001000  1234 0234 0000 0000 0000 4000 0000 000a
   0001010  0001 8000 1000 0000 2f6c 7633 0000 6c76
   0001020  3300 0000 000a 0003 0100 0000 2f28 0383
   0001030  0000 0001 0000 0200 0000 2000 0000 0000
   0001040
   ```

   В данном примере обратите внимание на поврежденную сигнатуру со смещением 0x1000 (1234 0234). Если при создании файловой системы применялись только значения по умолчанию, то должна быть показана сигнатура 0x43218765. В случае переопределения каких-либо значений по умолчанию должна быть показана сигнатура 0x65872143.

4. С помощью команды **od** проверьте правильность дополнительной сигнатуры главного блока. Пример команды и ее вывода:

   ```
   $ od -x -N 64 /dev/lv02 +0x1f000
   001f000  6587 2143 0000 0000 0000 4000 0000 000a
   001f010  0001 8000 1000 0000 2f6c 7633 0000 6c76
   001f020  3300 0000 000a 0003 0100 0000 2f28 0383
   001f030  0000 0001 0000 0200 0000 2000 0000 0000
   001f040
   ```

   Обратите внимание на правильную сигнатуру со смещением 0x1f000.

5. Скопируйте дополнительный главный блок в основной. Пример команды и ее вывода:

   ```
   $ dd count=1 bs=4k skip=31 seek=1 if=/dev/lv02 of=/dev/lv02
   dd: считано записей: 1+0
   dd: записано записей: 1+0
   ```

6. С помощью команды **fsck** удалите несогласованные файлы, вызванные применением дополнительного главного блока. Пример:

   ```
   fsck /dev/lv02 2>&1 | tee /tmp/fsck.errs
   ```

Информация, связанная с данной:

команда fsck

Команда od

Переполнение диска

Переполнение диска происходит при записи слишком большого числа файлов в выделенное пространство. Это может произойти в том случае, когда запущенные процессы создают много ненужных файлов.

Устранить неполадку можно с помощью следующих процедур:

Примечание: Для удаления процессов, принадлежащих другим пользователям, необходимы права доступа root.

Понятия, связанные с данным:

“Команды для автоматической очистки файловой системы” на стр. 374
Для очистки файловой системы путем удаления ненужных файлов вызовите команду **skulker**.

Поиск вышедших из строя процессов:

С помощью данной процедуры можно обнаружить проблемный процесс.

1. Для определения состояния процессов и выявления процессов, вызвавших ошибку, введите следующую команду:

   ```
   ps -ef | pg
   ```

   Команда **ps** показывает сведения о состоянии процесса. Флаг **-e** означает, что должна быть показана информация о всех процессах (кроме процессов ядра), а флаг **-f** указывает, что в списке процессов должны быть показаны имена команд и значения параметров, с помощью которых был запущен процесс. Команда **pg** показывает информацию по одной странице, чтобы большой объем информации не прокручивался на экране слишком быстро.

   Найдите системные или пользовательские процессы, использующие слишком большой объем системных ресурсов, таких как время CPU. Среди системных процессов это могут быть процессы **sendmail**, **routed** и **lpd**.
2. Для определения пользовательских процессов, интенсивно использующих CPU, введите следующую команду:

   ```
   ps -u
   ```
3. Запишите ИД процессов (PID) для всех процессов, вызывающих неполадки.

Завершение процесса:

Процес, вызвавший неполадку, можно завершить.

Следующая процедура позволяет прервать работу процесса, вызвавшего неполадку:

1. Для прерывания работы процесса введите следующую команду:

   ```
   kill -9 PID
   ```

   Где *PID* - ИД вызвавшего неполадку процесса.
2. Удалите файлы, созданные процессом, с помощью следующей команды:

   ```
   rm  file1 file2 file3
   ```

   где *file1 file2 file3* - имена файлов, связанных с процессом.

Освобождение дискового пространства без прерывания процесса:

Для освобождения дискового пространства, выделенного активному файлу, без завершения процесса, перенаправьте в это файл вывод другой команды. При перенаправлении файл будет усечен, а блоки памяти освободятся.

При удалении активного файла из файловой системы выделенные ему блоки не освобождаются до тех пор, пока не будет удален последний указатель open (в результате закрытия файла процессом или в результате завершения самого процесса, открывшего файл). Если процесс записывает данные в файл, то при удалении этого файла выделенное ему дисковое пространство не будет освобождено до завершения процесса.

Например:

```
$ ls -l
всего 1248
-rwxrwxr-x      1 web   staff   1274770 Июнь l 20 11:19 datafile
$ date > datafile
$ ls -l
всего 4
-rwxrwxr-x      1 web   staff        29 Июнь 20 11:20 datafile
```

Вывод команды **date** заменяет предыдущее содержимое файла `datafile`. Указанное число блоков усеченного файла отражает изменение размера файла с 1248> до 4. Если процесс продолжит запись данных в новый файл, то при следующем вызове команды **ls** может быть показана такая информация:

```
$ ls -l
всего 8
-rxrwxr-x       1 web   staff   1278866 Июль 20 11:21 datafile
```

Размер файла datafile отражает объем данных, добавленных процессом, но число выделенных блоков невелико. Это значит, что в файле datafile есть свободное пространство. Свободное пространство файла представляет собой область файла, для которой не выделены блоки дискового пространства.

Переполнение корневого каталога /:

При заполнении корневой файловой системы (/) выполните следующие действия.

- С помощью следующей команды просмотрите содержимое файла /etc/security/failedlogin

  ```
  who /etc/security/failedlogin
  ```

  Слишком быстрое создание терминалов может привести к появлению записей о неудавшемся входе в систему. Для очистки файла после просмотра или сохранения его содержимого выполните следующую команду:

  ```
  cp /dev/null /etc/security/failedlogin
  ```

- Просмотрите, нет ли в каталоге /dev неправильно указанного имени устройства. Если имя устройства задано неправильно, например, rmto вместо rmt0, то в каталоге /dev будет создан файл rmto. Обычно команда продолжает работу до заполнения всей корневой файловой системы. Каталог /dev является частью корневой (/) файловой системы. Просмотрите записи, не относящиеся к устройствам, т.е. не имеющие основного и дополнительного номеров. Для поиска таких записей проверки введите следующую команду:

  ```
  cd /dev
  ls -l | pg
  ```

  В колонке, в которой для обычного файла указывается размер, для файла устройств указывается два числа, разделенных запятой. Например:

  ```
  crw-rw-rw-  1 root     system    12,0 окт 25 10:19 rmt0
  ```

  Если имя файла или его размер указывают на недопустимость файла устройства, как в следующем примере, то удалите такой файл:

  ```
  crw-rw-rw-   1 root     system  9375473 окт 25 10:19 rmto
  ```

  Примечание:

  – Не удаляйте из каталога /dev файлы, соответствующие допустимым именам устройств. Одним из индикаторов недопустимости файла устройства является размер, превышающий 500 байт.

  – При работе в системе средств контроля возможно быстрое заполнение каталога /audit.

- С помощью команды **find** найдите слишком большие файлы, которые можно было бы удалить. Например, для поиска в корневом каталоге (/) всех файлов размером более 1 МБ введите следующую команду:

  ```
  find / -xdev -size  +2048 -ls |sort -r -n +6
  ```

  Эта команда находит все файлы размером более 1 МБ и упорядочивает их по размеру, начиная с самых больших. При таком поиске можно также применять другие флаги команды find, например, **-newer**. Подробные сведения приведены в описании команды **find**.

  Примечание: При проверке корневой файловой системы файлы устройств с основными и дополнительными номерами в каталоге /dev будут перечислены вместе с обычными файлами, для которых указывается размер. Основные и дополнительные номера, разделенные запятой, можно игнорировать.

  Перед удалением файлов воспользуйтесь следующей командой, позволяющей убедиться, что файл не используется другим процессом:

  ```
  fuser файл
  ```

где *файл* - имя проверяемого большого файла. Если файл открыт в момент удаления, то он будет удален только из списка содержимого каталога. Блоки, выделенные этому файлу, будут освобождены только после завершения процесса, использующего файл.

Устранение переполнения в файловой системе /var:

При заполнении корневой файловой системы /var выполните следующие действия.

- С помощью команды find найдите в каталоге /var самые большие файлы. Например:

  ```
  find /var -xdev -size  +2048 -ls| sort -r  +6
  ```

  Подробные сведения приведены в описании команды **find**.
- Найдите в каталоге /var/tmp устаревшие и ненужные файлы.
- Проверьте размер файла /var/adm/wtmp, в который заносится информация о всех операциях локального и удаленного входа в систему, а также о сеансах telnet. Во время работы системы учета размер этого протокола может расти неограниченно. Система учета очищает протокол каждую ночь. Файл /var/adm/wtmp можно очистить вручную или отредактировать его, удалив из этого файла старую или ненужную информации. Для очистки файла введите следующую команду:

  ```
  cp /dev/null  /var/adm/wtmp
  ```

  При редактировании файла /var/adm/wtmp сначала с помощью следующей команды создайте его временную копию:

  ```
  /usr/sbin/acct/fwtmp < /var/adm/wtmp >/tmp/out
  ```

  Отредактируйте файл /tmp/out, удалив из него ненужные записи, а затем замените исходный файл с помощью команды

  ```
  /usr/sbin/acct/fwtmp -ic < /tmp/out > /var/adm/wtmp
  ```
- Очистите протокол ошибок в каталоге /var/adm/ras. Протокол ошибок можно полностью очистить только вручную.

  Примечание: Не используйте команду **cp /dev/null** для очистки протокола ошибок. Нулевая длина файла errlog отключает функцию ведения протокола ошибок операционной системы; такой файл нужно заменить сохраненной копией.

  1. Остановите демон протокола ошибок с помощью следующей команды:

     ```
     /usr/lib/errstop
     ```
  2. Удалите или переместите в другую файловую систему файл протокола ошибок с помощью одной из следующих команд:

     ```
     rm /var/adm/ras/errlog
     ```

     или

     ```
     mv /var/adm/ras/errlog файл
     ```

     где *файл* - имя перемещаемого файла errlog.

     Примечание: При удалении файла протокола ошибок удаляются и хронологические данные.
  3. Перезапустите демон протокола ошибок с помощью следующей команды:

     ```
     /usr/lib/errdemon
     ```

  Примечание: Вы можете ограничить размер размер протокола ошибок с помощью следующих записей **cron**:

  ```
  0 11 * * * /usr/bin/errclear -d S,O 30
  0 12 * * * /usr/bin/errclear -d H 90
  ```
- Проверьте размер файла trcfile в этом каталоге. Если он очень большой и трассировка в данный момент не выполняется, то вы можете удалить этот файл с помощью следующей команды:

  ```
  rm /var/adm/ras/trcfile
  ```

- Если в качестве устройства дампа применяется hd6 (это значение по умолчанию), то в каталоге `/var/adm/ras` возможно наличие множества файлов `vmcore*`. Если это старые или ненужные файлы, то вы можете удалить их с помощью команды **rm**.
- Проверьте каталог `/var/spool`, содержащий файлы подсистемы очередей. Для очистки подсистемы очередей воспользуйтесь следующей командой:

  ```
  stopsrc -s qdaemon
  rm /var/spool/lpd/qdir/*
  rm /var/spool/lpd/stat/*
  rm /var/spool/qdaemon/*
  startsrc -s qdaemon
  ```
- Проверьте каталог `/var/adm/acct`, содержащий записи системы учета. Во время работы системы учета этот каталог может содержать несколько больших файлов.
- Проверьте, не сохранены ли в каталоге `/var/preserve` прерванные сеансы **vi**. Как правило, эти файлы можно удалить. Если пользователь хочет восстановить сеанс, то он может воспользоваться командой **vi -r** для просмотра списка восстанавливаемых сеансов. Для восстановления выбранного сеанса применяется команда **vi -r** *файл*.
- Измените файл `/var/adm/sulog` , в котором сохраняется информация о числе обращений к командам **su** и о результатах этих попыток. Этот обычный текстовый файл, который можно просматривать и изменять с помощью текстового редактора. При удалении этого файла он будет заново создан при следующем обращении к команде **su**. Измените файл `/var/tmp/snmpd.log`, в который заносятся события демона **snmpd**. При удалении этот файл создается демоном **snmpd**.

  Примечание: Размер файла `/var/tmp/snmpd.log` можно ограничить. Для этого измените файл `/etc/snmpd.conf`, указав в соответствующем размере максимальный размер в байтах.

Понятия, связанные с данным:

“Учет ресурсов системы” на стр. 156
Утилита учета ресурсов системы позволяет собирать данные и создавать отчеты об индивидуальном и групповом использовании различных ресурсов системы.

Фиксация прочих файловых систем и общие приемы поиска:

С помощью команды **find** с флагом **-size** найдите большие файлы, либо, в случае переполнения файловой системы, с помощью флага **-newer** найдите файлы с самой поздней датой последнего изменения.

Для создания базового файла, который можно указать с флагом **-newer**, воспользуйтесь следующей командой:

```
touch
ММддччмм файл
```

где *ММ* - это месяц, *дд* - день, *чч* - часы в формате 24 часов, *мм* - минуты, а *файл* - имя файла, создаваемого командой **touch**.

После создания базового файла вы можете воспользоваться командой find для поиска больших файлов, более новых, чем указанный:

```
find /файловая-система -xdev -newer базовый-файл -ls
```

Вы также можете воспользоваться командой **find** для писка файлов, измененных в течение последних 24 часов, как показано в следующем примере:

```
find /файловая-система -xdev -mtime 0 -ls
```

Монтирование

Монтирование позволяет получить доступ к файловым системам, файлам, каталогам, устройствам и специальным файлам, расположенным в определенном узле. Это единственный способ предоставить доступ к файловой системе.

Команда **mount** служит для монтирования файловой системы в заданном каталоге.

Вы можете смонтировать файловую систему при наличии прав доступа к монтируемому файлу или каталогу и прав на запись для точки монтирования. Члены группы system могут монтировать устройства (в этом случае устройства или файловые системы монтируются в каталоги) и файловые системы, описанные в файле `/etc/filesystems`. Пользователь с правами доступа root может смонтировать любую файловую систему, указав в командной строке устройство и каталог. В файле `/etc/filesystems` перечисляются файловые системы, которые должны автоматически монтироваться при загрузке системы. Команда **mount** предназначена для монтирования файловых систем после запуска системы.

Точки монтирования:

Точка монтирования - это каталог или файл, с помощью которого обеспечивается доступ к новой файловой системе, каталогу или файлу. Для монтирования файловой системы или каталога в качестве точки монтирования следует использовать каталог, а для монтирования файла точкой монтирования должен быть файл.

Как правило, файловая система, каталог или файл при монтируются к свободному узлу, но это не является обязательным требованием. Если файл или каталог, применяемый в качестве точки монтирования, содержит данные, то эти данные становятся недоступными на то время, пока поверх точки монтирования смонтирован другой файл или каталог. Таким образом, смонтированный файл или каталог перекрывает первоначальное содержимое точки монтирования. Исходный каталог или файл становится доступен после размонтирования.

Если файловая система монтируется в каталоге, то права доступа к корневому каталогу смонтированной файловой системы имеют более высокий приоритет, чем права доступа к точке монтирования. Исключением является ссылка на родительский каталог .. (две точки) в точке монтирования. Для того чтобы новая файловая система была доступна операционной системе, должен быть доступен родительский каталог точки монтирования.

Например, если в каталоге `/home/frank` ввести команду **cd ..**, , то текущим станет каталог `/home`. Если каталог `/home/frank` - это корневой каталог смонтированной файловой системы, то для выполнения команды `cd ..` операционная система должна получить информацию о родительском каталоге каталога `/home/frank`.

Для выполнения команд, обращающихся к информации о родительском каталоге, у пользователя должны быть права доступа на поиск в каталоге точки монтирования. Отсутствие прав на поиск в каталоге точки монтирования может привести к непредсказуемым результатам. Например, может перестать работать команда **pwd**. Если поиск в каталоге точки монтирования запрещен, то команда **pwd** показывает следующее сообщение:

```
pwd: Доступ запрещен
```

Для устранения этой ошибки нужно задать для каталога точки монтирования как минимум режим доступа 111.

Монтирование файловых систем, каталогов и файлов:

Монтирование бывает двух типов - удаленное и локальное. *Удаленное монтирование* выполняется для систем, обмен информацией с которыми осуществляется с помощью каналов связи. В удаленных файловых системах, например в сетевой файловой системе (NFS), необходимо экспортировать файлы перед монтированием. *Локальное монтирование* выполняется в локальной системе.

Каждая файловая система связана с отдельным устройством (логическим томом). Для того чтобы файловая система стала доступной, она должна быть подключена к существующей структуре каталогов (к корневой файловой системе или другой уже подключенной файловой системе). Это подключение выполняется командой **mount**.

Для доступа к одной и той же файловой системе, каталогу или файлу может использоваться несколько путей. Например, если с одной базой данных работают несколько пользователей, иногда бывает удобно смонтировать одну и ту же базу данных несколько раз, причем каждый раз в целях мониторинга и разделения заданий можно указывать свое имя и пароль. Это достигается путем монтирования одной файловой системой в нескольких точках. Например, файловую систему `/home/server/database` можно смонтировать в `/home/user1`, `/home/user2` и `/home/user3`:

```
/home/server/database        /home/user1
/home/server/database        /home/user2
/home/server/database        /home/user3
```

Доступ нескольких пользователей к файловой системе, каталогу или файлу можно организовать с помощью символьной связи. Для создания ссылок применяется команда **ln -s**. Если несколько пользователей связаны с одним файлом, то при каждом обращении к этому файлу каждый из них будет видеть все внесенные изменения.

Автоматическое управление монтированием:

Монтирование может выполняться автоматически при запуске системы.

Автоматическое монтирование бывает двух типов. Первый тип - это монтирование файловых систем, необходимых для загрузки и запуска. Монтирование таких файловых систем выполняется непосредственно в процессе загрузки. Для таких файловых систем в файле `/etc/filesystems` указано `mount = automatic`. Второй тип - это автоматическое монтирование под управлением пользователя. Монтирование таких файловых систем выполняется командой **mount all** сценария `/etc/rc`. Для таких файловых систем в файле `/etc/filesystems` указано `mount = true`.

Параметры автоматического монтирования хранятся в файле `/etc/filesystems`. Монтирование выполняется последовательно, одновременно выполняется не больше одной операции. Порядок монтирования можно изменять. Дополнительная информация о каталоге `/etc/filesystems` приведена в разделе `/etc/filesystems`.

Файл `/etc/filesystems` состоит из разделов, по одному для каждой точки монтирования. В каждом разделе указаны атрибуты и параметры монтирования соответствующей файловой системы. Монтирование файловых систем выполняется в том порядке, в котором они перечислены в файле `/etc/filesystems`. Ниже приведен пример разделов файла `/etc/filesystems file`:

```
/:
 dev=/dev/hd4
 vol="root"
 mount=automatic
 check=false
 free=true
 vfs=jfs
 log=/dev/hd8
 type-bootfs

/home:
 dev=/dev/hd1
 vfs=jfs
 log=/dev/hd8
 mount=true
 check=true
 vol="/home"
 free=false

/usr:
 /dev=/dev/hd2
 vfs=jfs
 log=/dev/hd8
 mount=automatic
```

```
check=false
type=bootfs
vol="/usr"
free=false
```

Для изменения порядка монтирования измените его в файле /etc/filesystems. Если какую-либо файловую систему смонтировать не удалось, то монтирование других файловых систем, указанных в файле /etc/filesystems, будет продолжено. Например, невозможность смонтировать файловую систему /home не помешает монтированию файловой системы /usr. Монтирование может завершиться неудачно из-за синтаксических ошибок, невыполнения определенных условий или системных неполадок.

Монтирование защиты для бездисковых рабочих станций:

Бездисковым рабочим станциям необходима возможность создания и доступа к специальным файлам устройств в удаленных системах для монтирования каталогов /dev. Поскольку серверы не отличают собственные файлы устройств и файлы устройств клиентов, у пользователю сервера предоставляется доступ к физическим устройствам сервера через файлы устройств клиента.

Например, владельцем устройства **tty** автоматически становится пользователь, работающий с соответствующим **терминалом**. Если идентификаторы пользователей в системах клиента и сервера не совпадают, пользователь клиента может получить несанкционированный доступ к устройству **tty**, с которым работает другой пользователь сервера.

Пользователь, у которого есть необходимые права доступа на компьютере-клиенте, может создавать специальные файлы устройств, связанные с физическими устройствами сервера, даже если у него нет прав доступа к этим устройствам. Воспользовавшись затем своей непривилегированной учетной записью на сервере, пользователь сможет получить доступ к защищенным устройствам с помощью новых специальных файлов устройств.

Аналогичная проблема с защитой возникает при работе с программами в системах клиента и сервера, пользующимися функциями **setuid** и **setgid**. Для нормальной работы необходимо предоставить бездисковым клиентам возможность создавать и запускать на сервере программы **setuid** и **setgid**. Как и в предыдущем случае, сервер не может определить, для какого компьютера предназначена программа - для клиента или для сервера.

Кроме того, ИД пользователей и групп на сервере и на клиенте могут отличаться. Таким образом, пользователи могут запускать на сервере программы с правами доступа, не предоставленными этим пользователям.

Проблема состоит в том, что программы **setuid**, **setgid** и специальные файлы устройств предназначены для использования на том же компьютере, на котором они были созданы.

Решение состоит в применении опций команды **mount**, ограничивающих доступ к таким объектам. Эти опции можно также указать в файле /etc/filesystems.

Опция nosuid команды mount запрещает выполнение программ **setuid** и **setgid** из смонтированной файловой системы. Эту опцию следует указывать для всех файловых систем, которые монтируются на данном хосте, но предназначены исключительно для использования с другого хоста (например, для файловых систем, экспортируемых бездисковым клиентам).

Опция nodev команды **mount** запрещает доступ к специальным файлам устройств через смонтированную файловую систему. Эту опцию также следует указывать для всех файловых систем, которые предназначены для использования другим хостом (например, для файловых систем, экспортируемых для бездисковых клиентов).

Как правило, пользователям сервера доступ к каталогу /export запрещен.

Экспорт каталога /export/root

Каталог `/export/root` необходимо экспортировать с правами доступа на чтение и запись; кроме того, должен быть предоставлен доступ пользователю root. Однако, вы можете смонтировать этот каталог с помощью команды **mount** со следующими опциями:

Элемент	Описание
nosuid	Запрещает запускать на сервере программы **setuid** клиента.
nodev	Запрещает пользователю доступ к устройствам сервера с помощью специальных файлов устройств клиента.

Вместо указания соответствующих опций при монтировании каталога `/export/root` можно просто запретить пользователям сервера доступ к этому каталогу.

Экспорт каталога /export/exec

Каталог `/export/exec` следует экспортировать с правами доступа только на чтение; кроме того, должен быть предоставлен доступ пользователю root. Однако, вы можете смонтировать этот каталог с помощью команды **mount** со следующими опциями:

Элемент	Описание
nosuid	Запрещает пользователю запускать на сервере программы **setuid** клиента. При экспорте каталога `/usr` сервера опцию **nousid** указывать нельзя.
nodev	Запрещает пользователю доступ к устройствам сервера с помощью специальных файлов устройств клиента.

Экспорт каталога /export/share

Каталог `/export/share` следует экспортировать с правами доступа только для чтения; кроме того, должен быть предоставлен доступ пользователю root. Так как в этом каталоге, как правило, хранятся только данные (в нем нет исполняемых файлов и файлов устройств), то указывать в команде mount опции защиты необязательно.

Экспорт каталога /export/home

Существует несколько способов монтирования каталога `/home`:

- Можно смонтировать каталог `/export/home/`*имя-клиента* в каталоге `/home` клиента. В этом случае клиент получает права доступа на чтение и запись, а пользователь root - полные права доступа. Для обеспечения защиты системы смонтируйте каталог `/export/home` со следующими опциями команды **mount**:

Элемент	Описание
nosuid	Запрещает пользователю запускать на сервере программы **setuid** клиента.
nodev	Запрещает пользователю доступ к устройствам сервера с помощью специальных файлов устройств клиента.

- Можно смонтировать каталог `/home` сервера в каталоге `/home` клиента. В этом случае каталог `/home` необходимо экспортировать с правами доступа на чтение и запись, но без предоставления доступа пользователю root. Для обеспечения защиты системы смонтируйте каталог `/home` на сервере и на клиенте, указав в команде **mount** опции `nosuid` и `nodev`.
- Другой способ заключается в монтировании каждого каталога `/home/`*имя-пользователя* в соответствующем каталоге клиента `/home/`*имя-пользователя*, чтобы у пользователей была возможность работать со своими домашними каталогами на различных компьютерах. При этом каталоги `/home/`*имя-пользователя* на сервере и на клиентах должны быть смонтированы командой **mount** с опциями `nousid` и `nodev`.

Экспорт каталога /export/swap

Экспортируйте файл `/export/swap/`*имя-клиента* с правами на чтение и запись, и предоставьте права доступа пользователю root. Никаких действий по обеспечению защиты предпринимать не требуется. Пользователям сервера доступ к файлам `/export/swap/`*имя-клиента* должен быть запрещен.

Бездисковое монтирование:

Несмотря на то, что файловые системы бездисковых рабочих станций монтируются из каталога /exports сервера, они ничем не отличаются от файловых систем автономных систем.

Ниже приведен список экспортируемых каталогов сервера и соответствующих точек монтирования на бездисковой рабочей станции:

Экспортируемые каталоги на сервере	Импортируемые каталоги на бездисковом компьютере
/export/root/*имя_хоста*	/ (корневой каталог)
/export/exec/*имя_SPOT*	/usr
/export/home/*имя_хоста*	/home
/export/share	/usr/share
/export/dump	Применяется бездисковыми клиентами для дампа
/export/swap	Применяется на бездисковых клиентах в качестве удаленного пространства подкачки

Дополнительная информация о каталоге /export приведена в разделе “Каталог /export” на стр. 434.

Как правило, пользователям сервера доступ к каталогу /export запрещен.

Экспорт каталога /export/root

Каталог /export/root необходимо экспортировать с правами доступа на чтение и запись; кроме того, должен быть предоставлен доступ пользователю root. Однако, вы можете смонтировать этот каталог с помощью команды **mount** со следующими опциями:

Элемент	Описание
nosuid	Запрещает запускать на сервере программы **setuid** клиента.
nodev	Запрещает пользователю доступ к устройствам сервера с помощью специальных файлов устройств клиента.

Вместо указания соответствующих опций при монтировании каталога /export/root можно просто запретить пользователям сервера доступ к этому каталогу.

Экспорт каталога /export/exec

Каталог /export/exec следует экспортировать с правами доступа только на чтение; кроме того, должен быть предоставлен доступ пользователю root. Однако, вы можете смонтировать этот каталог с помощью команды **mount** со следующими опциями:

Элемент	Описание
nosuid	Запрещает пользователю запускать на сервере программы **setuid** клиента. При экспорте каталога /usr сервера опцию **nousid** указывать нельзя.
nodev	Запрещает пользователю доступ к устройствам сервера с помощью специальных файлов устройств клиента.

Экспорт каталога /export/share

Каталог /export/share следует экспортировать с правами доступа только для чтения; кроме того, должен быть предоставлен доступ пользователю root. Так как в этом каталоге, как правило, хранятся только данные (в нем нет исполняемых файлов и файлов устройств), то указывать в команде mount опции защиты необязательно.

Экспорт каталога /export/home

Существует несколько способов монтирования каталога /home:

- Можно смонтировать каталог /export/home/*имя-клиента* в каталоге /home клиента. В этом случае клиент получает права доступа на чтение и запись, а пользователь root - полные права доступа. Для обеспечения защиты системы смонтируйте каталог /export/home со следующими опциями команды **mount**:

Элемент	Описание
nosuid	Запрещает пользователю запускать на сервере программы **setuid** клиента.
nodev	Запрещает пользователю доступ к устройствам сервера с помощью специальных файлов устройств клиента.

- Можно смонтировать каталог /home сервера в каталоге /home клиента. В этом случае каталог /home необходимо экспортировать с правами доступа на чтение и запись, но без предоставления доступа пользователю root. Для обеспечения защиты системы смонтируйте каталог /home на сервере и на клиенте, указав в команде **mount** опции **nosuid** и **nodev**.
- Другой способ заключается в монтировании каждого каталога /home/*имя-пользователя* в соответствующем каталоге клиента /home/*имя-пользователя*, чтобы у пользователей была возможность работать со своими домашними каталогами на различных компьютерах. При этом каталоги /home/*имя-пользователя* на сервере и на клиентах должны быть смонтированы командой **mount** с опциями **nousid** и **nodev**.

Экспорт каталога /export/dump

Экспортируйте каталог /export/dump/*имя-клиента* с правами на чтение и запись, и предоставьте права доступа пользователю root. Пользователям сервера доступ к файлам /export/dump/*имя-клиента* должен быть запрещен.

Экспорт каталога /export/swap

Экспортируйте файл /export/swap/*имя-клиента* с правами на чтение и запись, и предоставьте права доступа пользователю root. Никаких действий по обеспечению защиты предпринимать не требуется. Пользователям сервера доступ к файлам /export/swap/*имя-клиента* должен быть запрещен.

Типы файловых систем

AIX поддерживает файловые системы нескольких типов.

Это следующие типы:

Журнализированная файловая система (JFS) или расширенная журнализированная файловая система (JFS2)

Поддерживает полный набор команд работы с файловыми системами. Как и в базах данных, для поддержания целостности таких файловых систем применяется журнализация. Это позволяет предотвратить повреждение файловой системы при аварийном завершении работы системы.

Каждая файловая система JFS и JFS2 располагается на отдельном логическом томе. Операционная система монтирует файловые системы во время инициализации. Разделение всего дерева файлов на несколько файловых систем повышает эффективность выполнения таких операций управления системой, как резервное копирование, восстановление и исправление, так как вы можете работать только с одной частью дерева файлов.

JFS - это базовый тип файловой системы. В файловых системах такого типа поддерживается полный набор команд.

JFS2 - это базовый тип файловой системы. В файловых системах такого типа поддерживается полный набор команд.

Разница между JFS и JFS2 заключается в том, что JFS2 поддерживает большие файлы и файловые системы.

сетевая файловая система (NFS)

NFS - это распределенная файловая система, позволяющая работать с удаленными файлами и каталогами так же, как с локальными. Например, пользователь с помощью команд операционной системы может выполнить операции создания, удаления, чтения и записи удаленного файла или каталога, а также настройки его атрибутов.

файловая система на компакт-диске (CDRFS)

Позволяет работать с файлами на компакт-диске через обычный интерфейс файловой системы (выполнять операции открытия, чтения и закрытия).

Файловая система на диске DVD (UDFS)

Позволяет работать с файлами на DVD через обычный интерфейс файловой системы.

Информация, связанная с данной:

Сетевая файловая система

JFS и JFS2:

Поддержка журнализированной файловой системы (JFS) и расширенной журнализированной файловой системы (JFS2) встроена в операционную систему. Обе файловые системы связывают данные о файлах и каталогах со структурой Администратора логических томов AIX.

Различие между ними заключается в том, что в JFS2 реализовано 64-разрядное ядро и поддержка больших файлов.

Эти файловые системы описаны в следующих разделах. Если явно не указано иное, то вся приведенная информация относится к обеим файловым системам.

Функции JFS и JFS2:

Расширенная журнализированная файловая система (JFS2) позволяет хранить файлы значительно большего размера, чем журнализированная файловая система (JFS).

Для выполнения можно выбрать JFS или JFS2. JFS2 - это файловая система по умолчанию в AIX 6.1.

Примечание: В отличие от файловой системы JFS, JFS2 не позволяет применять API **link()** для каталогов. Это ограничение может вызвать ошибку на некоторых приложениях, которые правильно работали в JFS.

Ниже перечислены основные функции JFS и JFS2:

Функции	**JFS2**	**JFS**
Размеры фрагментов и блоков	Размеры блоков: 512, 1024, 2048, 4096 байт Максимальный размер файловой системы: 4, 8, 16, 32 Тб	Размеры фрагментов: 512, 1024, 2048, 4096 байт Максимальный размер файловой системы: 128, 256, 512, 1024 ГБ
Максимальный размер файловой системы	32 Тб	1 Тб
Минимальный размер файловой системы	16 МБ	Неприменимо
Максимальный размер файла	16 Тб	Приблизительно 63,876 ГБ
Число i-узлов	Динамическое, ограничено дисковым пространством	Фиксировано; устанавливается при создании файловой системы
Структура каталогов	Двоичное дерево	Линейная
Сжатие	Нет	Да
Квоты	Да	Да
Ведение протоколов ошибок	Да	Да

Примечание:

1. Максимальный размер файла и файловой системы для 32-разрядного ядра равен 1 Тб (размер физического раздела). Например, если размер физического раздела для группы томов равен 64 МБ, то максимальный размер файловой системы составит (1 Тб - 64 МБ) = (1048576 МБ - 64 МБ) = 1048512 МБ. Это обусловлено ограничением максимального размера логического тома при использовании 32-разрядного ядра.

2. JFS2 поддерживает стандартную схему ведения протоколов ошибок AIX. Дополнительная информация о ведении протокола ошибок AIX приведена в разделе Error-Logging Overview книги *Программирование: Разработка и отладка программ*.

Сегментация дискового пространства JFS и JFS2:

В большинстве файловых систем UNIX дисковое пространство разделено на блоки одинакового размера, в каждом из которых расположен логический блок файла или каталога. Такие физические блоки часто называются *блоками диска*. Один блок диска хранит данные только одного логического блока файла или каталога.

Использование логических блоков сравнительно большого размера (например, по 4096 байт) и выделение дисковых блоков, равных по размеру логическим блокам, позволяет снизить число операций дискового ввода-вывода, выполняемых для каждой операции с файловой системой. В этом случае данные файлов и каталогов хранятся на диске в небольшом количестве крупных блоков, а не в большом количестве маленьких блоков. Например, если размер логического блока равен 4096 байт, то для хранения файла размером 4096 байт или менее будет выделен один блок диска размером 4096 байт. Следовательно, для выполнения операции чтения или записи потребуется выполнить только одну операцию дискового ввода-вывода. При меньшем размере логического блока для размещения того же объема данных потребуется несколько блоков диска, что приведет к увеличению числа операций дискового ввода-вывода, необходимых для обращения к файлу. Применение больших логических блоков, равных по размеру дисковым блокам, имеет еще одно преимущество - такой подход позволяет снизить количество операций, связанных с выделением дополнительного дискового пространства при добавлении данных в файлы и каталоги. Это объясняется тем, что большие дисковые блоки позволяют разместить больший объем данных.

Однако если размер блока диска будет равен размеру логического блока, а в файловой системе содержится большое число небольших файлов и каталогов, то значительный объем памяти будет использоваться впустую. Это связано с тем, что для размещения неполного логического блока файла или каталога будет выделяться блок диска, равный по размеру логическому блоку. Следовательно, часть блока диска, выделенного для неполного логического блока, будет расходоваться впустую. Незанятая часть блока диска не может применяться для размещения данных другого файла или каталога, поскольку этот блок уже выделен другому объекту. Если файловая система содержит значительное число мелких файлов и каталогов, то общий объем потерянной впустую дисковой памяти может быть очень большим.

Журнализированная файловая система (JFS) разбивает дисковое пространство на блоки, называемые *фрагментами*. Расширенная файловая система (JFS2) разбивает пространство на *блоки*. Однако цель остается прежней: эффективное распределение дискового пространства.

Фрагменты JFS имеют меньший размер, чем выделяемый по умолчанию блок памяти (4096 байт). Фрагменты позволяют более эффективно использовать дисковое пространство при сохранении неполных логических блоков файлов и каталогов. Поддержка фрагментов JFS основана на технологии поддержки фрагментов Berkeley Software Distribution (BSD).

JFS2 поддерживает размер блока 512, 1024, 2048 и 4096. Блоки диска небольшого размера позволяют более эффективно использовать дисковое пространство при сохранении неполных логических блоков файлов и каталогов. С другой стороны, для управления блоками небольшого размера требуется выполнять большее число операций. Размер блока JFS2 задается при создании файловой системы. В разных файловых системах могут применяться блоки разного размера, но в пределах каждой файловой системы могут применяться только блоки одного размера.

Понятия, связанные с данным:

“Сжатие данных в JFS” на стр. 468
JFS поддерживает фрагментированные и сжатые файловые системы, в которых экономия дисковой памяти достигается за счет того, что размер минимального логического блока (фрагмента) файла на диске меньше, чем размер физического блока (4 КБ).

Фрагменты JFS:

В JFS дисковая память может выделяться блоками (или *фрагментами*), размер которых меньше размера логического блока, составляющего 4096 байт.

Если размер фрагмента меньше 4096 байт, то данные отдельного логического блока можно разместить более эффективно, выделив ровно столько фрагментов, сколько необходимо для хранения данных. Например, если размер логического блока равен 500 байт, то для него можно выделить только один фрагмент размером 512 байт (если предположить, что размер фрагментов равен 512 байт). В этом случае объем неиспользуемой дисковой памяти будет значительно меньше. При добавлении данных в неполный логический блок ему будут выделены дополнительные фрагменты.

Размер фрагмента файловой системы задается во время ее создания. В журнализированной файловой системе (JFS) поддерживаются фрагменты размером 512, 1024, 2048 и 4096 байт. В различных файловых системах могут применяться фрагменты разных размеров, но внутри одной файловой системы размер фрагмента постоянен. В одной и той же системе (компьютере) могут применяться фрагменты разных размеров, то есть пользователь может выбрать для каждой файловой системы наиболее подходящий размер фрагмента.

Поддержка фрагментов JFS рассматривает файловую систему как набор смежных фрагментов, а не как набор смежных блоков диска. Однако для повышения эффективности дисковых операций дисковое пространство часто выделяется в виде блоков размером 4096 байт, то есть блоки диска остаются равными логическим блокам. В этом случае выделение блока диска можно рассматривать как выделение диапазона смежных фрагментов, общий размер которого равен 4096 байт.

При уменьшении размера фрагмента увеличиваются затраты на выполнение операции (дополнительные операции подвода считывающих головок, передача данных и операции выделения памяти), но одновременно повышается эффективность использования дискового пространства. Для поиска оптимального баланса между ростом затрат и повышением рациональности использования дискового пространства в поддержке фрагментов JFS применяются следующие правила:

- По возможности для файлов или логических блоков каталогов дисковое пространство выделяется в виде набора смежных фрагментов размером 4096 байт.
- Дисковое пространство размером менее 4096 байт выделяется только для размещения неполных логических блоков файлов и каталогов размером менее 32 КБ.

Когда размер файлов и каталогов файловой системы становится больше 32 КБ, становится невыгодно выделять дисковое пространство блоками размером менее 4096 байт. Сэкономленная дисковая память составляет незначительный процент от общего объема памяти файловой системы, при этом стоимость обслуживания небольших блоков диска остается очень высокой. Поскольку выделение блоков диска размером менее 4096 байт позволяет повысить эффективность использования дискового пространства только при работе с небольшими файлами и каталогами, для логических блоков файлов и каталогов размером 32 КБ и более всегда выделяются диапазоны фрагментов размером 4096 байт. Для любого неполного логического блока большого файла или каталога также выделяется диапазон фрагментов размером 4096 байт.

Блоки JFS2:

Расширенная файловая система (JFS2) разбивает пространство на *блоки*. JFS2 поддерживает размер блока 512, 1024, 2048 и 4096.

В разных файловых системах могут применяться блоки разного размера, но в пределах каждой файловой системы могут применяться только блоки одного размера.

Блоки диска небольшого размера позволяют более эффективно использовать дисковое пространство при сохранении неполных логических блоков файлов и каталогов. С другой стороны, для управления блоками

небольшого размера может потребоваться выполнить большее число операций. Кроме того, драйвер устройства должен уметь обращаться к блокам диска того же или меньшего размера, чем размер блока файловой системы.

Поскольку в файловых системах с размером блока менее 4096 байт дисковое пространство выделяется блоками небольшого размера, то при увеличении размера файлов и каталогов операции выделения памяти выполняются чаще. Например, в файловой системе с размером блока 512 байт при записи данных размером 512 байт в файл нулевого размера будет выделен один блок. Если при следующей операции записи размер файла увеличится еще на 512 байт, то файлу потребуется выделить еще один блок. В файловой системе с блоками по 4096 байт дисковое пространство было бы выделено только один раз, при первой операции записи. Во время второй операции записи дополнительное пространство не выделялось бы, поскольку первоначально выделенный блок размером 4096 байт достаточен для размещения всех данных файла.

Размер блока файловой системы задается при ее создании с помощью Инструмента управления системой (SMIT) или команд **crfs** и **mkfs**. При выборе размера блока файловой системы следует учесть предполагаемый размер файлов, которые будут храниться в этой файловой системе.

Размер блока файловой системы можно узнать с помощью Инструмента управления системой (SMIT) или команды **lsfs**. Для прикладных программ определить размер блока файловой системы можно с помощью функции **statfs**.

Блок - это минимальная единица дискового пространства, которая может быть выделена системой. Состояние блоков файловой системы (занят или свободен) хранится в таблице использования блоков файловой системы. Если размер блока файловой системы меньше 4096 байт, то для хранения таблицы использования блоков этой файловой системы требуется больше виртуальной памяти и дискового пространства.

Переменное число i-узлов:

Так как применение сегментации дискового пространства повышает эффективность использования дисковой памяти, возрастает число небольших файлов и каталогов, которые могут храниться в файловой системе.

Однако дисковое пространство - это только один из ресурсов файловой системы, необходимый для файлов и каталогов: каждому файлу и каталогу также требуется дисковый i-узел.

JFS и i-узлы:

В JFS допустимо, чтобы число i-узлов диска, создаваемых в файловой системе, отличалось от значения по умолчанию.

Число i-узлов диска задается во время создания файловой системы в виде *числа байт на i-узел* (*NBPI*). Например, если значение NBPI равно 1024, то i-узел диска будет создаваться для каждых 1024 байт дискового пространства файловой системы. Если значение NBPI будет маленьким (например, 512 байт), то будет создано большое число i-узлов, а если значение NBPI будет большим (например, 16384), то будет создано небольшое число i-узлов.

В файловых системах JFS для каждого блока группы размещения размером NBPI байт создается один i-узел. Число i-узлов файловой системы ограничивает общее число файлов и максимальный размер файловой системы. Если группа размещения выделяется только частично, то для нее создается столько же i-узлов, сколько необходимо для полной группы. Значение NBPI обратно пропорционально общему числу i-узлов файловой системы.

В JFS общее число i-узлов всех файловых систем не должно превосходить 16 M (2^{24}).

Набор допустимых значений NBPI зависит от размера группы размещения (*agsize*). Значение по умолчанию - 8 Мб. При значении *agsize* 8 Мб допускаются следующие значения NBPI: 512, 1024, 2048, 4096, 8192 и 16384.

Допустимы и большие значения *agsize*. Для *agsize* допустимы значения 8, 16, 32 и 64. Интервал допустимых значений NBPI увеличивается в соответствии с увеличением размера *agsize*. При увеличении значения *agsize* в два раза (до 16 Мб) допустимые значения NBPI также удваиваются: 1024, 2048, 4096, 8193, 16384 и 32768.

Размер фрагмента и значение NBPI задаются при создании файловой системы с помощью Инструмента управления системой (SMIT) или команд **crfs** и **mkfs**. При выборе размера фрагмента и числа i-узлов файловой системы нужно оценить планируемое число файлов, которое будет храниться в файловой системе.

Размер фрагмента и значение NBPI можно узнать с помощью Инструмента управления системой (SMIT) или команды **lsfs**. В прикладных программах значение размера фрагмента можно получить с помощью функции **statfs**.

JFS2 и i-узлы:

В JFS2 i-узлы выделяются по мере необходимости.

Если в файловой системе есть место для дополнительных i-узлов, они автоматически выделяются. Таким образом, число доступных i-узлов ограничено лишь размером самой файловой системы.

Ограничения размера JFS и JFS2:

При создании файловой системы можно указать максимальный размер JFS. Выбор правильного значения должен основываться на нескольких факторах.

Рекомендуемый максимальный размер JFS2 равен 16 терабайт. Минималльным размером файловой системы JFS2 является 16 МБ.

Несмотря на то, что применение файловых систем с размером фрагмента менее 4096 байт позволяет более эффективно использовать дисковое пространство, создание фрагментов небольшого размера может привести к некоторому снижению производительности системы.

Сведения о состоянии фрагментов (JFS) и блоков (JFS2) файловой системы хранятся в таблице использования блоков файловой системы. Если размер блока файловой системы меньше 4096 байт, то для хранения таблицы использования блоков этой файловой системы требуется больше виртуальной памяти и дискового пространства.

Поскольку в файловых системах с размером блока или фрагмента менее 4096 байт дисковое пространство выделяется блоками небольшого размера, то при увеличении размера файлов и каталогов операции выделения памяти выполняются чаще. Например, для записи в файл нулевой длины 512 байт данных нужно выделить один фрагмент, если размер фрагмента или блока равен 512 байт. Для добавления в файл следующих 512 байт нужно выделить еще один фрагмент или блок. В файловой системе с размером фрагмента или блока 4096 байт дисковое пространство было бы выделено только один раз - при выполнении первой операции записи. Во время выполнения второй операции записи не требуется выделять дополнительную память, так как первый фрагмент или блок размером 4096 байт достаточен для хранения всех данных. Число операций выделения памяти можно минимизировать, если размер файлов каждый раз будет увеличиваться на 4096 байт.

Размер протокола файловой системы - это еще один параметр, который необходимо задать.

Обычно все журнализированные файловые системы применяют общий протокол размером 4 МБ. Например, после начальной установки все файловые системы корневой группы томов применяют в качестве общего протокола JFS логический том hd8. По умолчанию размер раздела в логическом томе равен 4 МБ, а размер протокола по умолчанию равен одному разделу; таким образом, в корневой группе томов обычно содержится протокол JFS размером 4 МБ. Когда размер файловой системы достигает 2 ГБ или когда суммарный размер файловых систем, использующих общий протокол, достигает 2 ГБ, то размер протокола, применяемого по умолчанию, может оказаться недостаточным. В обоих случаях размер протокола

увеличивается по мере увеличения размера файловой системы. После изменения размера логического тома протокола необходимо вызвать команду **logform**, позволяющую повторно инициализировать протокол с учетом нового размера логического тома. Максимальный размер протокола JFS составляет 256 МБ.

Общий размер файловых систем, использующих один протокол JFS, ограничен. В общем случае один протокол может обслуживать один триллион байт. Если этот размер превышен или скоро будет превышен, а также если команда **logredo** (вызываемая командой **fsck**) сообщает о нехватке памяти, то добавьте еще один протокол JFS.

Обычно файловые системы JFS2 применяют общий протокол. Когда размер файловой системы достигает 2 ГБ или когда суммарный размер файловых систем, использующих общий протокол, достигает 2 ГБ, то размер протокола, применяемого по умолчанию, может оказаться недостаточным. В любом случае вы можете либо увеличить размер протокола, либо добавить еще один протокол JFS2.

Максимальный размер JFS:

Максимальный размер JFS задается во время ее создания. При этом следует учитывать NBPI, размер фрагмента, размер группы размещения и прочие факторы.

Размер файловой системы ограничен наименьшим из следующих значений:
$NBPI * 2^{24}$

или
$Размер\_фрагмента * 2^{28}$

Например, если NBPI равен 512, то размер файловой системы будет ограничен 8 ГБ ($512 * 2^{24} = 8$ ГБ). JFS поддерживает следующие значения NBPI: 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536 и 131072.

В JFS общее число i-узлов всех файловых систем не должно превосходить 16 М (2^{24}).

Для каждого блока группы размещения размером *NBPI* байт создается один i-узел. Если группа размещения выделяется только частично, то для нее создается столько же i-узлов, сколько необходимо для полной группы. Значение NBPI обратно пропорционально общему числу i-узлов файловой системы.

В JFS вся память разделена на группы, состоящие из i-узлов и блоков диска с пользовательскими данными. Эти группы называются группами размещения. Размер группы размещения задается во время создания файловой системы. Поддерживаются группы размещения размером 8 МБ, 16 МБ, 32 МБ и 64 МБ. С каждым размером группы размещения связан диапазон значений NBPI. Эти значения приведены в следующей таблице:

Размер группы размещения в мегабайтах	**Допустимые значения NBPI**
8	512, 1024, 2048, 4096, 8192, 16384
16	1024, 2048, 4096, 8192, 16384, 32768
32	2048, 4096, 8192, 16384, 32768, 65536
64	4096, 8192, 16384, 32768, 65536, 131072

JFS поддерживает фрагменты размером 512, 1024, 2048 и 4096 байт. Адреса фрагментов JFS указываются в i-узлах и промежуточных блоках в виде 28-разрядных чисел. Адрес фрагмента представляет собой число от 0 до 2^{28}.

Максимальный размер JFS2:

Очень большие файловые системы JFS2, содержащие очень большие файлы, более практичны, чем содержащие большое число маленьких файлов. Выполнение команды **fsck** для больших файловых систем с небольшими файлами занимает существенное время.

Рекомендуется придерживаться следующих ограничений:

Элемент	Описание
Максимальный размер JFS2:	32 Тб
Максимальный размер файла JFS2:	16 Тб
Минимальный размер JFS2:	16 МБ

Фрагментация свободного пространства JFS:

Применение в файловой системе JFS фрагментов размером менее 4096 байт может привести к сильной фрагментации свободного дискового пространства.

Например, предположим, что диск разделен на восемь фрагментов размером 512 байт. Пусть первый, четвертый, пятый и седьмой фрагменты диска были выделены для файлов размером 512 байт каждый, а второй, третий, шестой и восьмой фрагменты остались свободными. Несмотря на то, что свободно четыре фрагмента общим объемом 2048 байт, они не будут выделены для неполного логического блока, требующего наличия четырех свободных фрагментов (2048 байт), поскольку все эти фрагменты должны располагаться последовательно.

Поскольку фрагменты, выделенные для размещения файла или каталога, должны располагаться последовательно, фрагментация свободного дискового пространства может привести к тому, что операция над файловой системой не будет выполнена даже в том случае, если общий объем свободного дискового пространства достаточен для выполнения операции. Например, для записи в файл нулевой длины одного логического блока требуется выделить один блок памяти размером 4096 байт. Если свободное пространство файловой системы фрагментировано и состоит из 32 несмежных фрагментов размером по 512 байт (всего 16 КБ), то операция записи выполнена не будет, поскольку система не сможет найти восемь последовательных фрагментов (4096 байт дискового пространства), необходимых для выполнения операции.

Для дефрагментации свободного дискового пространства файловой системы JFS можно воспользоваться командой **defragfs**. Выполнение команды **defrags** позволяет повысить производительность системы.

Файл с зарезервированным пространством:

Файл представляет собой последовательность индексированных блоков. Блоки размечаются от i-узла до логического смещения файла.

Файл с индексами, не указывающими на блоки данных, называется *файлами с зарезервированным пространством*. Файл с зарезервированным пространством имеет определенный размер, однако сумма длин всех его блоков данных меньше размера самого файла. Для определения файлов с зарезервированным пространством служит команда **fileplace**. Она показывает все незанятые блоки файла.

Примечание: В большинстве случаев определить, соответствует ли число блоков данных файла его размеру, позволяет также команда **du**. В файловой системе со сжатием файл, число блоков данных которого не соответствует его размеру, может не являться файлом с зарезервированным пространством.

Файл с зарезервированным пространством создается, когда приложение расширяет файл за пределы уже занятых индексов, однако записанные данные не занимают все новые индексы. Новый размер файла отражает самую дальнюю запись в файл.

При чтении данных из незаполненной области файла возвращаются нулевые байты. При записи в эту область файла выделяются необходимые блоки.

Такие свойства могут оказать влияние на операции с файлом или команды архивации. Например, следующие команды не сохраняют зарезервированное пространство файла:

- **cp**

- **mv**
- **tar**
- **cpio**

Примечание: В случае команды **mv** это верно лишь в отношении перемещения файла в другую файловую систему. При перемещении файла в рамках одной файловой системы зарезервированное пространство сохраняется.

В результате копирования или восстановления файлов с помощью названных выше команд зарезервированное пространство будет утеряно. Однако следующие команды позволяют сохранить или восстановить зарезервированное пространство в файле:

- **backup**
- **restore**
- **pax**

При работе с файлами с зарезервированными пространством следует соблюдать осторожность, так как возможно исчерпание ресурсов файловой системы.

Большие файлы в JFS:

В файловой системе можно создавать большие файлы.

Большие файлы поддерживаются всеми файловыми системами JFS2.

Для создания файловых систем с поддержкой больших файлов служат команды **crfs** и **mkfs**. У обеих команд есть опция (`bf=true`), позволяющая включить поддержку больших файлов. Для создания таких файловых систем можно также воспользоваться SMIT.

В файловых системах с поддержкой больших файлов первые 4 Мб данных файла хранятся в блоках по 4096 байт. Все остальные данные размещаются в больших блоках диска размером 128 Кб. На самом деле большие блоки диска представляют собой 32 смежных блока размером 4096 байт.

Например, в обычной файловой системе для размещения файла размером 132 Мб необходимо 33 промежуточных адресных блока (каждый из которых содержит 1024 адреса блоков размером по 4 Кб). Файл размером 132 Мб в файловой системе с поддержкой больших файлов займет 1024 дисковых блока по 4 Кб и 1024 блока по 128 Кб. Поддержка больших файлов позволяет использовать для файла размером 132 Мб только два адресных блока. В файловых системах обоих типов потребуется один двойной косвенный блок.

Для большого блока диска требуется 32 смежных блока размером 4 Кб. Если после записи данных в файл его размер становится больше 4 Мб, то операция может завершится неудачно с кодом ошибки ENOSPC из-за того, что в файловой системе нет 32 смежных блоков размером 4 Кб.

Примечание: В файловой системе может быть несколько тысяч свободных блоков, но если среди них нет 32 блоков, расположенных последовательно, то дисковое пространство выделено не будет.

Команда **defragfs** реорганизует дисковое пространство и объединяет свободные блоки в смежные области.

JFS применяется для инициализации новых дисков. При монтировании первой файловой системы, поддерживающей большие файлы, JFS запускает процедуру ядра kproc, обнуляющую начальные расположения файлов. После размонтирования файловой системы, поддерживающей большие файлы, процедура kproc повторяется.

Сжатие данных в JFS:

JFS поддерживает фрагментированные и сжатые файловые системы, в которых экономия дисковой памяти достигается за счет того, что размер минимального логического блока (фрагмента) файла на диске меньше, чем размер физического блока (4 КБ).

Сжатие данных в JFS2 не поддерживается.

Во фрагментированной файловой системе в таких фрагментах хранятся только последние логические блоки файлов размером не более 32 КБ. В сжатой файловой системе в таких фрагментах могут храниться все блоки файлов любого размера. В среднем за счет сжатия данных экономится примерно 50% дисковой памяти.

Однако применение фрагментов и сжатия данных повышает риск фрагментации дискового пространства. Фрагменты, выделенные логическому блоку, должны занимать непрерывную область памяти. Если свободная память файловой системы сильно фрагментирована, то сложно выбрать достаточное количество свободных смежных сегментов, необходимых для размещения логического блока, даже если общее число свободных фрагментов велико. В JFS для решения проблемы фрагментации применяется программа **defragfs**, объединяющая свободные блоки в последовательные цепочки. С помощью этой утилиты можно увеличить объем непрерывной свободной памяти в фрагментированных и сжатых файловых системах. Экономия дисковой памяти за счет применения фрагментов и сжатия данных достаточно велика, чтобы можно было смириться с фрагментированностью файловой системы, которую всегда можно устранить.

Алгоритмы сжатия данных в текущей версии JFS совместимы с предыдущими версиями данной операционной системы. API, в который включены все системные вызовы, связанные со сжатием данных, остался в новой версии JFS без изменений.

Понятия, связанные с данным:

“Сегментация дискового пространства JFS и JFS2” на стр. 461
В большинстве файловых систем UNIX дисковое пространство разделено на блоки одинакового размера, в каждом из которых расположен логический блок файла или каталога. Такие физические блоки часто называются *блоками диска*. Один блок диска хранит данные только одного логического блока файла или каталога.

Реализация сжатия данных в JFS:

Сжатие данных - это атрибут файловой системы, задаваемый при ее создании с помощью команды **crfs** или **mkfs**. Для определения сжатия данных можно также воспользоваться SMIT.

Внимание: Корневая файловая система (/) не может быть сжата. Кроме того, не рекомендуется сжимать файловую систему `/usr`, поскольку программа **installp** должна иметь возможность точно определить размер этой файловой системы при установке обновлений и новых компонентов программного обеспечения.

Сжатие данных применяется только для обычных файлов и длинных символьных связей. Фрагменты поддерживаются и при работе с каталогами и метаданными, сжатие которых не выполняется. Каждый логический блок файла сжимается независимо от других перед записью на диск. Такой подход обеспечивает максимальную скорость выборки и обновления данных, хотя эффективность сжатия несколько снижается (по сравнению со сжатием больших блоков данных).

После сжатия данных для размещения логического блока обычно требуется менее 4096 байт дискового пространства. Сжатый блок записывается на диск, и для него выделяется непрерывный блок памяти, состоящий из минимального достаточного числа фрагментов. Если логический блок не удается сжать, то он записывается на диск без сжатия и для него выделяется непрерывный блок памяти размером 4096 байт.

Текущее значение атрибута сжатия данных можно узнать с помощью команды **lsfs -q**. Для определения сжатия данных можно также воспользоваться SMIT.

Понятия, связанные с данным:

“Косвенное поведение сжатия данных в JFS”
Поскольку после успешного завершения операции записи файла (или сохранения - при использовании файлов размещения) программа не ожидает возникновения ошибки, связанной с недостатком свободного места на диске (ENOSPC), необходимо гарантировать, что на диске хватит места для записи логических блоков.

Косвенное поведение сжатия данных в JFS:

Поскольку после успешного завершения операции записи файла (или сохранения - при использовании файлов размещения) программа не ожидает возникновения ошибки, связанной с недостатком свободного места на диске (ENOSPC), необходимо гарантировать, что на диске хватит места для записи логических блоков.

Для этого при первом изменении логического блока ему выделяется 4096 байт дискового пространства, что достаточно для записи даже несжатого блока. Если такой объем памяти выделить нельзя, система возвращает код ошибки ENOSPC или EDQUOT, даже в том случае, если объем свободного дискового пространства достаточен для размещения сжатого логического блока. Поэтому в случаях, когда в файловой системе очень мало свободной памяти, а также когда пользователь израсходовал почти всю выделенную ему память, могут выдаваться ошибочные сообщения о том, что недостаточно дисковой памяти.

Кроме того, сжатые файловые системы могут работать следующим образом:

- Поскольку изначально для логического блока выделяется 4096 байт, некоторые системные вызовы могут вернуть код ошибки ENOSPC или EDQUOT. Например, если старый файл размещен в памяти с помощью системного вызова **mmap**, то операция сохранения в предыдущее расположение может привести к получению сообщения об ошибке ENOSPC.
- При использовании сжатия данных блок диска целиком остается зарезервированным за измененным логическим блоком вплоть до момента записи на диск. Если ранее измененный блок занимал меньше памяти, чем полный физический блок, то сначала ему дополнительно выделяется полный физический блок, затем измененный блок сжимается и записывается на диск, и только после этого освобождается блок, занятый ранее. Такая ситуация характерна для обычных фрагментов. В каждом файле может содержаться только один логический блок, занимающий менее одного физического блока. В сжатой файловой системе такими могут быть все логические блоки.
- Ни один из ресурсов, ранее выделенных логическому блоку, не освобождается до тех пор, пока приложение не обратиться к системному вызову **fsync** или **sync**.
- Системный вызов **stat** позволяет узнать число фрагментов, выделенных для файла. Это число вычисляется по следующей схеме: к числу фрагментов, занятых блоками, добавляется число фрагментов дисковой памяти, выделенной измененным и еще не записанным блокам. Каждому такому блоку выделяется полный физический блок размером 4096 байт. Системный вызов **stat** не учитываются ресурсы, выделенные файлу ранее. После фиксации i-узла вызов **stat** возвращает правильное количество выделенных фрагментов в том случае, если ни один из измененных блоков не был сжат. Обратите внимание: проверка ограничений на дисковую память выполняется без учета того, что некоторые фрагменты заняты только на время и будут освобождены. При записи сжатых логических блоков файла на диск число выделенных им фрагментов уменьшается, что приводит к изменению дисковых квот и значения, выдаваемого системным вызовом **stat**.

Понятия, связанные с данным:

“Реализация сжатия данных в JFS” на стр. 468
Сжатие данных - это атрибут файловой системы, задаваемый при ее создании с помощью команды **crfs** или **mkfs**. Для определения сжатия данных можно также воспользоваться SMIT.

Алгоритм сжатия данных в JFS:

В AIX применяется разработанный в IBM алгоритм сжатия на базе стандартного алгоритма LZ. Алгоритм LZ сжимает данные путем замены второго и всех последующих вхождений строки на указатель, задающий первое вхождение строки, и ее длину.

В начале процедуры сжатия в базе данных повторяющихся строк нет ни одной строки, поэтому как минимум первый байт данных не будет "сжат" и будет записан в выходную строку в исходном виде (при этом к нему добавляется сигнальный бит и символ занимает уже 9 бит - 0 и исходный байт). После сжатия определенного объема данных, например, *N* байт, программа сжатия начинает искать самую длинную строку, которая встречается в этих *N* байтах, совпадающую со строкой, с которой начинается следующий необработанный байт. Если длина найденной строки равна 0 или 1, то следующий байт не кодируется и записывается в исходном виде (с сигнальным битом). В противном случае совпадающая строка заменяется на пару (указатель,длина), а число обработанных байт увеличивается на длину этой строки. В модификации алгоритма LZ фирмы IBM для *N* поддерживаются значения 512, 1024 и 2048. В алгоритме IBM LZ зафиксировано кодирование пар (указатель, длина) и обычных символов. Указатель представляет собой поле фиксированной длины размером log2 *N*, а длина - это поле переменной длины.

Влияние на производительность сжатия данных в JFS:

Поскольку сжатие данных основано на применении фрагментов, все выводы о влиянии на производительность, относящиеся к фрагментам, применимы и к сжатию данных.

Помимо этого, сжатие данных оказывает следующее влияние на производительность файловой системы:

- Сжатие и извлечение данных занимает значительное время, поэтому в некоторых случаях применять сжатые файловые системы нецелесообразно.
- Большинство обычных файлов UNIX записываются только один раз, но некоторые файлы обновляются. При изменении файла для каждого обновляемого логического блока дополнительно выделяется 4096 байт физической дисковой памяти, независимо от того, сколько памяти этот блок будет занимать после сжатия. Перераспределение памяти происходит после окончательной записи сжатого логического блока на диск. Все эти действия не нужно выполнять для несжатых файловых систем.
- Применение сжатия данных повышает нагрузку на процессор. Программное сжатие данных требует примерно 50 тактов процессора на байт, а развертывание - примерно 10 тактов на байт.

Оперативные резервные копии JFS и моментальные копии JFS2:

Можно создать моментальный образ файловой системы JFS или файловой системы JFS2, который в дальнейшем можно использовать для восстановления системы. Однако требования и особенности поведения этого образа зависят от типа файловой системы.

Файловая система JFS позволяет создать статическую копию зеркальной копии файловой системы, предназначенную только для чтения. Обычно зеркальная копия изменяется при изменении исходной файловой системы, однако сделанная таким образом моментальная копия изменена не будет. Она будет соответствовать тому моменту времени, когда было выполнено копирование. При создании резервной копии в ней не будут отражены изменения, внесенные после создания моментальной копии. Поэтому копирование рекомендуется выполнять в то время, когда активность файловой системы минимальна. Все изменения, внесенные после создания копии, не будут в ней отражены.

В файловой системе JFS2 такой моментальный образ называется *моментальной копией*. Моментальная копия остается статической и сохраняет те же права доступа, которые были в исходной файловой системе (называемой *snappedFS*). Моментальную копию JFS2 можно создать без размонтирования или приостановки файловой системы. Можете использовать моментальную копию JFS2 для оперативного копирования файловой системы, для доступа к файлам и каталогом, существовавшим в момент создания моментальной копии, или для сохранения резервной копии на съемном носителе. При работе с моментальными копиями JFS2 необходимо учесть следующее:

- Образ моментальной копии файловой системы корня (/) или /usr заменяется при перезагрузке системы. Для сохранения моментальных копий других файловых систем можно размонтировать файловую систему перед выполнением перезагрузки. Моментальные копии, созданные в AIX 5.2 с 5200-01, можно восстановить. Если команды **fsck** или **logredo** выполняются в файловой системе JFS2 с моментальной копией, созданной в AIX 5.2 с 5200-01, эта моментальная копия будет сохранена. Также можно восстановить файловую систему, размонтированную с очисткой, с моментальной копией, созданной в AIX 5.2, если она монтируется в системе AIX 5.2 с 5200-01.
- Для файловой системы с моментальными копиями не рекомендуется выполнять команду **defragfs**. Каждый блок, перемещенный при дефрагментации, необходимо внести в моментальную копию, что занимает достаточно много времени и требует дополнительного пространства в логическом томе моментальной копии.
- При нехватке пространства для моментальной копии все моментальные копии данной snappedFS удаляются. При этом в протокол ошибок заносится сообщение.
- Если при записи в моментальную копию происходит сбой, все моментальные копии данной snappedFS удаляются. При этом в протокол ошибок заносится сообщение.
- Если моментальная копия была создана в AIX 5.2 с 5200-01, или к ней обращались из этой системы, то из AIX 5.2 она будет недоступна. Перед монтированием файловой системы такие моментальные копии следует удалить.
- Файловая система JFS2, у которой есть моментальная копия в AIX 5.3, будет доступна только системам AIX 5.2 с 5200-01 и системам более поздних версий. Если в системе планируется восстановить старую версию, то для обеспечения доступа к файловой системе необходимо предварительно удалить моментальные копии.

Оперативные резервные копии JFS:

Можно создать синхронное изображение файловой системы JFS с ее последующим использованием для целей резервного копирования.

Файловая система JFS позволяет создать статическую копию зеркальной копии файловой системы, предназначенную только для чтения. Обычно зеркальная копия изменяется при изменении исходной файловой системы, однако сделанная таким образом моментальная копия изменена не будет. Она будет соответствовать тому моменту времени, когда было выполнено копирование. При создании резервной копии в ней не будут отражены изменения, внесенные после создания моментальной копии. Поэтому копирование рекомендуется выполнять в то время, когда активность файловой системы минимальна. Все изменения, внесенные после создания копии, не будут в ней отражены.

Моментальные копии JFS2:

Можно создать синхронное изображение файловой системы JFS2 с ее последующим использованием для целей резервного копирования.

В файловой системе JFS2 такая копия называется *моментальной*. Моментальная копия не изменяется и имеет те же права доступа, что и исходная файловая система (называемая *snappedFS*). Моментальную копию JFS2 можно создать без размонтирования или приостановки файловой системы. Моментальную копию JFS2 можно использовать в следующих случаях:

- Доступ к файлам или каталогам в том виде, которому они соответствовали в момент создания моментальной копии.
- Резервное копирование на съемный носитель.

Существует два типа моментальных копий JFS2: внутренняя и внешняя. Внешняя моментальная копия JFS2 создается в отдельном логическом томе файловой системы. Внешнюю моментальную копию можно смонтировать отдельно от файловой системы на ее собственной уникальной точке монтирования.

Внутренняя моментальная копия JFS2 создается в том же логическом томе, что и файловая система, и размещает блоки из файловой системы. Доступ к внутренней моментальной копии можно получить из скрытого каталога `.snapshot` в корневом каталоге файловой системы JFS2 с моментальной копией. Файловую систему JFS2 необходимо включать для поддержки внутренних моментальных копий в момент создания файловой системы.

Моментальные копии JFS2 не поддерживают проверку квот файловой системы. Для определения состояния квот нельзя пользоваться командой моментальной копии **repquota**. При отзыве изображения файловой системы к изображению моментальной копии синхронно сохраняется информация о квоте. Запомните следующие замечания, касающиеся внешних и внутренних моментальных копий JFS2:

- Если моментальная копия была создана в AIX 5.2 с 5200-01, или к ней обращались из этой системы, то из AIX 5.2 она будет недоступна. Перед монтированием файловой системы такие моментальные копии следует удалить.
- Файловая система JFS2, у которой есть моментальная копия в AIX 5.3, будет доступна только системам AIX 5.2 с 5200-01 и системам более поздних версий. Если в системе планируется восстановить старую версию, то для обеспечения доступа к файловой системе необходимо предварительно удалить моментальные копии.
- Не рекомендуется проводить запуск команды **defragfs** в файловой системе JFS2 с внешними моментальными копиями, поскольку каждый блок, перемещенный при дефрагментации, необходимо внести в моментальную копию, что занимает достаточно много времени и требует дополнительного пространства в логическом томе моментальной копии.
- При нехватке памяти для внешней моментальной копии или ее сбое все копии данной snappedFS помечаются как недействительные. Таким образом, произойдет сбой доступа к моментальной копии. При этом в протокол ошибок заносится сообщение.

Замечания к внутренней моментальной копии JFS2:

- Внутренние моментальные копии сохраняются при запуске команды **logredo** на файловой системе JFS2 с внутренней моментальной копией.
- Внутренние моментальные копии удаляются, если для наладки файловой системы JFS2 необходимо было ее изменить с помощью команды **fsck**.
- При нехватке памяти для внутренней моментальной копии или ее сбое все копии данной snappedFS помечаются как недействительные. Таким образом, произойдет сбой доступа к внутренней моментальной копии. При этом в протокол ошибок заносится сообщение.
- Внутренние моментальные копии отдельно не монтируются. Доступ к внутренним моментальным копиям в корневом каталоге `.snapshot` файловой системы можно получить сразу после их создания. В результате, доступ к внутренним моментальным копиям можно получить через сервер NFS без экспорта отдельной точки монтирования для моментальной копии.
- Внутренние моментальные копии несовместимы с выпусками AIX, появившимися ранее AIX 6.1. Файловую систему JFS2, созданную для поддержки внутренних моментальных копий, нельзя модифицировать на более раннем выпуске AIX.
- Файловая система JFS2, созданная для поддержки внутренних моментальных копий, также включается для поддержки Расширенных атрибутов версии 2.
- Файловую систему JFS2 с внутренними моментальными копиями нельзя использовать с Интерфейсом программирования приложений управления данными (DMAPI).
- Для файловой системы JFS2 с внутренними моментальными копиями не рекомендуется выполнять команду **defragfs**.
- Каталог `.snapshot` нельзя вернуть с помощью системного вызова **readdir()**. Это избавит Вас от непредвиденного появления моментальных копий. Все вызовы или команды системы зависят от возможного сбоя вызова системы **readdir()** с каталогом `.snapshot`, (например, системная команда **/bin/pwd** и системный вызов **getcwd()** каталога `.snapshot` не могут найти родительский каталог).

Совместимость и перенос:

Файловые системы JFS полностью совместимы между AIX 5.1 и AIX 5.2. Предыдущие версии операционной системы полностью совместимы с текущей версией JFS, однако при переходе к предыдущей версии следует обратить особое внимание на файловые системы, в которых размер фрагмента, значение NBPI или размер группы размещения отличаются от принятых по умолчанию.

Примечание: Файловые системы JFS не поддерживается на диска с размером сектора 4 КБ. Таким образом, при создании файловой системы и выполнении операций резервного копирования не следует использовать диски с размером сектора 4 КБ.

Файловые системы JFS2, за исключением моментальных копий, совместимы между AIX 5.1 и AIX 5.2, но не совместимы с более ранними версиями операционной системы. Файловые системы JFS2 с моментальными копиями не поддерживаются в AIX 5.1. Перед возвратом к более ранним версиям AIX все файловые системы JFS2 должны быть размонтированы без ошибок, поскольку команда **logredo** может не работать для файловых систем, созданных в более поздних выпусках.

Примечание: Файловые системы JFS2, созданные в формате v2 или преобразованные в этот формат, недоступны в старых выпусках AIX.

Ниже описаны проблемы, которые могут возникнуть при использовании файловых систем из более ранних версий операционной системы:

Образы файловой системы JFS

Любые образы файловых систем JFS, в которых применяется принятый по умолчанию размер фрагмента, NBPI, равный 4096 байт и размер группы размещения (agsize), равный 8, могут переноситься из текущей в версию AIX 4.3 (и наоборот), не требуя каких-либо специальных действий по преобразованию.

Примечание: мгновенные копии JFS2: Моментальные копии JFS2, которые были созданы в AIX 5L версии 5.2 с рекомендуемым пакетом обслуживания 5200-01, или к которым обращались из этой операционной системы, будут недоступны для более ранних выпусков. Перед монтированием файловой системы такие моментальные копии следует удалить.

Резервное копирование и восстановление между файловыми системами JFS

Операции резервного копирования и восстановления можно выполнять в файловых системах JFS с разным размером блоков, однако из-за более эффективного использования дискового пространства в файловых системах с меньшим размером блоков операция восстановления в файловой системе с блоками большего размера может завершиться неудачно. Это необходимо учесть при сохранении и восстановлении всей файловой системы, поскольку такая ошибка может возникнуть даже в том случае, когда общий размер целевой файловой системы больше размера исходной файловой системы.

Если при восстановлении резервной копии сжатой файловой системы она преобразуется в обычную файловую систему или сжатую файловую систему с другим размером фрагмента, то операция восстановления может завершиться неудачно из-за нехватки дискового пространства. Это необходимо учитывать при сохранении и восстановлении всей файловой системы, поскольку такая ситуация может возникнуть даже в том случае, если размер целевой файловой системы больше размера исходной файловой системы.

Ограничения драйвер устройств JFS и JFS2

Драйвер устройства должен уметь обращаться к блокам диска того же или меньшего размера, чем размер блока или фрагмента файловой системы. Например, если файловая система JFS была создана на компьютере с драйвером RAM, то этот драйвер должен поддерживать блоки размером 512 байт для хранения файловой системы с размером фрагмента 512 байт. Если драйвер поддерживает адресацию только на уровне страниц, то может применяться только JFS с размером фрагмента 4096.

Копирование JFS в другой физический том:

Файловую систему JFS можно скопировать на другой физический том с сохранением целостности файловой системы.

В следующем сценарии описано копирование файловой системы JFS на другой физический том с сохранением целостности файловой системы.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

Для копирования JFS на другой физический том с сохранением целостности файловой системы выполните следующие действия:

1. прекратите работу в копируемой файловой системой. Если приложение, работающее с файловой системой, не находится в стабильном состоянии или состояние файловой системы вам неизвестно, то вы не сможете точно определить, какие именно данные включены в резервную копию.
2. С помощью следующей команды SMIT создайте зеркальную копию логического тома:
   ```
   smit mklvcopy
   ```
3. Скопируйте файловую систему с помощью следующей команды:
   ```
   chfs -a splitcopy=/backup -a copy=2 /testfs
   ```

   Параметр **splitcopy** флага **-a** указывает, что команда должна выделить зеркальную копию файловой системы и смонтировать ее только для чтения в новой точке монтирования. Это позволяет создать копию файловой системы с согласованными метаданными журналов и использовать эту копию в качестве резервной копии.
4. Для перемещения зеркальной копии в другую точку монтирования введите следующую команду SMIT:
   ```
   smit cplv
   ```

Теперь вы можете начать работу с копией файловой системы.

Информация, связанная с данной:

Команда mklv

Файловая система на компакт-диске и файловая система UDF:

Файловая система на компакт-диске (CDRFS) - это локальная файловая система, предназначенная только для чтения, которая может быть сохранена на носителе CD-ROM, CD-RW (если защищена от записи) и DVD-ROM. Максимальный размер файла CDRFS - 2 Гб, независимо от используемого типа носителя. Файловая система UDF является реализацией локальной файловой системы, позволяющей запись, которая может быть сохранена только для чтения на носителе DVD-ROM или с возможностью записи на носителе DVD-RAM.

По умолчанию компакт-диски монтируются автоматически, однако эту функцию можно отключить. В последнем случае для монтирования файловой системы CDRFS применяется команда **cdmount**.

AIX поддерживает следующие форматы томов и файловых структур CDRFS:

Тип	Описание
Стандарт ISO 9660:1988(E)	CDRFS поддерживает уровень 3 обмена данными и уровень 1 реализации ISO 9660.
Спецификация High Sierra Group	Предшественник стандарта ISO 9660, обеспечивающий обратную совместимость со старыми дисками CD-ROM.
Протокол Rock Ridge Group	Расширение стандарта ISO 9660, полностью совместимое с этим стандартом. Этот протокол поддерживает полный набор функций файловой системы POSIX и основан на протоколах System Use Sharing Protocol (SUSP) и Rock Ridge Interchange Protocol (RRIP), позволяя получить доступ к CD-ROM путем монтирования, как и к любой другой файловой системе UNIX.
Формат файлов CD-ROM eXtended Architecture File Format (только режим 2, форма 1)	Формат Расширенная архитектура CD-ROM (XA) - это развитие стандарта ISO 9660, которое применяется в мультимедийных приложениях, основанных на CD-ROM, например, Photo CD.

Для всех форматов томов и структур файлов применяются следующие ограничения:

- Наборы, состоящие из нескольких томов, не поддерживаются
- Поддерживаются только файлы без чередования

CDRFS зависит от низкоуровневого драйвера CD-ROM, обеспечивая прозрачность формата физического сектора (CD-ROM режима 1 и CD-ROM XA режима 2 формы 1) и многосеансовый формат диска (преобразует набор дескрипторов томов из области распознавания томов последнего сеанса).

Примечание: Перед извлечением компакт-диска необходимо размонтировать CDRFS.

Поддерживается файловая система другого типа под названием UDFS, представляющая собой файловую систему только для чтения, хранящуюся на носителе DVD-ROM. Перед извлечением носителя необходимо размонтировать UDFS. AIX поддерживает форматы UDFS версий 1.50, 2.00 и 2.01.

UDFS следует экспортировать с помощью NFS в режиме только для чтения. Запись в UDFS, смонтированную как NFS, не поддерживается.

Для применения команды **cdmount** для автоматического монтирования UDFS, доступной для чтения и записи, отредактируйте файл `cdromd.conf`. Можно также вручную смонтировать доступную для записи и чтения UDFS с помощью команды **mount**.

Каталоги

Каталог - это особый тип файла, предназначенный для хранения информации, которая необходима для доступа к файлам или другим каталогам. Каталоги обычно занимают меньше памяти, чем файлы других типов.

Файловая система состоит из групп каталогов и файлов, хранящихся в этих каталогах. Обычно файловые системы представляются в виде перевернутого дерева. Корневой каталог, обозначаемый косой чертой (/), находится в основании и является вершиной дерева каталогов.

Ветви этого дерева представляют собой подкаталоги корневого каталога, которые, в свою очередь, могут содержать файлы и подкаталоги. Каждая ветвь задает однозначный путь к объекту системы в структуре каталогов.

Наборы файлов хранятся в каталогах. Такие наборы файлов часто связаны друг с другом. Структура каталогов позволяет упорядочить их.

Файл - это набор данных, который можно прочитать или записать. В файле может храниться программа, текст, данные или описание устройства. Команды, принтеры, терминалы, соответствия и прикладные программы хранятся именно в файлах. Такой подход позволяет работать с различными элементами системы единым образом, обеспечивая гибкость файловой системы.

Каталоги позволяют объединять файлы и другие каталоги в группы, организовывая файловую систему по иерархическому принципу. Такая многоуровневая структура файловой системы обеспечивает особую гибкость.

Каталог состоит из записей. В каждой записи хранится имя файла или подкаталога и ссылочный номер узла в индексе (номер *i-узла*). Для повышения быстродействия и экономии дискового пространства данные файла хранятся в разных областях памяти компьютера. Запись i-узла содержит адреса, позволяющие найти все распределенные блоки данных, связанные с файлом. В записи i-узла также хранится такая информация о файле, как время последнего изменения и обращения, режимы доступа, число ссылок, имя владельца и тип файла.

Для управления каталогами применяется специальная группа команд. Например, с помощью команды **ln** можно связать с одним i-узлом несколько записей каталогов с различными именами.

Поскольку в каталогах часто хранится информация, которая не должна быть доступна всем пользователям системы, в системе предусмотрены средства ограничения доступа к каталогам. Устанавливая права доступа к каталогу, вы можете выбирать, кто может работать с каталогом и кто может изменять информацию в каталоге.

Понятия, связанные с данным:

“Режимы доступа к файлам и каталогам” на стр. 297
У каждого файла есть владелец. Владельцем нового файла становится пользователь, создавший этот файл. *Права доступа* к файлу определяет его владелец. Права доступа указывают для других пользователей системы, могут ли они читать, изменять и исполнять файл. Права доступа к файлу могут быть изменены только владельцем файла или пользователем root.

Типы каталогов:

Каталоги могут создаваться операционной системой, системным администратором и пользователями.

Каталоги, созданные системой, обычно содержат файлы специальных типов, например команды. На вершине иерархии файловой системы находится созданный системой каталог `/ (корневой)`. Каталог `/` обычно содержит следующие стандартные каталоги, также созданные системой:

Элемент	Описание
`/dev`	Содержит специальные файлы для устройств ввода-вывода.
`/etc`	Содержит файлы инициализации и управления системой.
`/home`	Содержит домашние каталоги пользователей системы.
`/tmp`	Содержит временные файлы, которые могут быть удалены через определенное время.
`/usr`	Содержит системные каталоги `lpp`, `include` и прочие.
`/usr/bin`	Содержит пользовательские исполняемые программы.

Некоторые каталоги, такие как ваш начальный (или домашний) каталог (`$HOME`), создаются и изменяются системным администратором. В начальный каталог вы попадаете сразу после входа в систему.

Каталоги, создаваемые вами, называются *пользовательскими* каталогами. Такие каталоги применяются для систематизации файлов.

Структура каталогов:

Каталоги могут содержать файлы и подкаталоги. *Подкаталогом* называется каталог внутри другого каталога. Каталог, содержащий этот подкаталог, называется *родительским каталогом*.

Каждый каталог содержит запись для родительского каталога, в котором он был создан, с именем `..` (dot dot), and an entry for the directory itself(две точки), и запись для самого себя с именем `.` (точка). В большей части списков содержимого каталогов эти записи скрыты.

Дерево каталогов
Обычно структура файловой системы со временем усложняется. Постарайтесь поддерживать структуру файлов и каталогов настолько простой, насколько это возможно. Давайте файлам и каталогам простые осмысленные имена. Это упростит работу с ними.

Родительский каталог
У каждого каталога, за исключением / (корневого), есть один родительский каталог и может быть произвольное число подкаталогов.

Домашний каталог
В *домашнем*, или начальном, каталоге вы оказываетесь сразу после входа в систему. Этот каталог создается системным администратором для каждого пользователя. В домашнем каталоге хранятся личные файлы пользователя. Каталоги пользователя, предназначенные для личных целей, обычно создаются в домашнем каталоге. В любой момент вы можете вернуться в свой домашний каталог, введя в командной строке **cd** и нажав Enter.

Рабочий каталог
Во время работы с файловой системой вы всегда находитесь в каком-либо каталоге. Этот каталог называется *текущим* или *рабочим*. Имя рабочего каталога можно узнать с помощью команды **pwd**. Для изменения рабочего каталога служит команда **cd**.

Соглашения об именах каталогов:

Имя каждого каталога должно быть уникальным в его родительском каталоге. Это обеспечивает единственность полного пути к каталогу в файловой системе.

Имена каталогов должны соответствовать тем же соглашениям о присвоении имен, что и имена файлов. Эти соглашения описаны в разделе “Соглашения об именах файлов” на стр. 189.

Сокращения каталогов:

Для задания некоторых каталогов удобно использовать сокращения.

Ниже приведен список сокращений:

Сокращение	Значение
.	Текущий каталог.
..	Каталог, родительский по отношению к текущему.
~	Домашний каталог. (Кроме оболочки Bourne. Дополнительная информация приведена в разделе “Оболочка Bourne” на стр. 258).
$HOME	Домашний каталог. (Для всех оболочек).

Пути к каталогам:

К каждому файлу или каталогу можно перейти по единственному *пути* в структуре каталогов файловой системы. Путь указывает расположение каталога или файла в файловой системе.

Примечание: Длина пути не должна превышать 1023 символов.

В файловой системе предусмотрены следующие виды путей:

Элемент	Описание
полный путь	Полный путь начинается с каталога / (корневого). Имя полного пути начинается с косой черты (/).
относительный путь	Относительный путь начинается с текущего каталога и включает его родительский каталог или один из подкаталогов.

Полный путь представляет собой полное имя каталога или файла, начиная с каталога / (корневого). Полный путь не зависит от того, в какой точке файловой системы вы находитесь. Имена полных путей начинаются с косой черты (/), обозначающей корневой каталог. /A/D/9 - это полный путь для 9. Первая косая черта (/) обозначает каталог / (корневой), с которого начинается поиск. Оставшаяся часть пути говорит о том, что нужно перейти в каталог A, найти в нем каталог D а в нем - 9.

В системе могут существовать несколько файлов с именем 9, если полные имена этих файлов в файловой системе различаются. Пути /A/D/9 и /C/E/G/9 соответствуют двум разным файлам с именем 9.

В отличие от полных путей, относительные пути указывают на каталог или файл по отношению к текущему рабочему каталогу. Для перехода вверх по иерархии файловой системы в относительных путях применяются подкаталоги с именем две точки (..). Две точки (..) представляют родительский каталог. Поскольку относительные пути не начинаются от корневого каталога, их имена не начинаются с косой черты (/). Относительные пути чаще всего применяются для указания файла или каталога в текущем каталоге или в его подкаталогах. Если D - текущий каталог, то относительным путем к 10 будет F/10 (полным - по прежнему /A/D/F/10). Относительным путем к 3 будет ../../B/3.

На текущий каталог ссылается имя точка (**.**). Точка (**.**) обычно применяется в программах для указания текущего каталога.

Создание каталогов (команда mkdir):

Команда **mkdir** позволяет создать один или несколько каталогов с именами, указанными в параметре *Каталог*.

Новые каталоги будут содержать стандартные записи: точку (**.**) и две точки (**..**). С помощью флага **-m** *Режим* можно задать режим доступа к новым каталогам.

Новый каталог создается в текущем рабочем каталоге, если не указан полный путь к другому объекту файловой системы.

Ниже приведены примеры применения команды **mkdir**:

- Для создания в текущем рабочем каталоге каталога Test с правами доступа по умолчанию введите:
  ```
  mkdir Test
  ```
- Для создания в ранее созданном каталоге /home/demo/sub1 каталога Test с правами доступа rwxr-xr-x введите:
  ```
  mkdir -m 755 /home/demo/sub1/Test
  ```
- Для создания в каталоге /home/demo/sub2 каталога Test с правами доступа по умолчанию введите:
  ```
  mkdir -p /home/demo/sub2/Test
  ```

  Флаг **-p** означает, что при необходимости следует создать также каталоги /home, /home/demo и /home/demo/sub2.

Сведения о синтаксисе приведены в описании команды **mkdir** в книге *Справочник по командам, том 3*.

Перемещение или переименование каталогов (команда mvdir):

Команда **mvdir** предназначена для перемещения и переименования каталогов.

Ниже приведены примеры применения команды **mvdir**:

- Для перемещения каталога введите:
  ```
  mvdir book manual
  ```
  Каталог `book` будет перемещен в каталог `manual`, если последний существует. В противном случае каталог `book` будет переименован в `manual`.
- Для перемещения и переименования каталога введите:
  ```
  mvdir book3 proj4/manual
  ```
  Если каталог `manual` уже существует, то каталог `book3` будет перемещен в каталог `proj4/manual`. Другими словами, каталог `book3` станет подкаталогом `proj4/manual`. Если каталог manual не существует, то каталог `book3` будет переименован в `proj4/manual`.

Сведения о синтаксисе приведены в описании команды **mvdir** в книге *Справочник по командам, том 3*.

Просмотр текущего каталога (команда pwd):

Команда **pwd** записывает в стандартный вывод полное имя текущего каталога (имя, начинающееся с корневого каталога `/`).

Каталоги разделяются косой чертой (`/`). Первая косая черта (`/`) обозначает корневой каталог `/`, последним указан текущий каталог.

Например, для того чтобы узнать текущий каталог, введите:

```
pwd
```

Появится полное имя текущего каталога, например:

```
/home/thomas
```

Переход в другой каталог (команда cd):

Команда **cd** изменяет текущий каталог. У вас должны быть права доступа на выполнение (на поиск) к новому каталогу.

Если параметр *Каталог* не указан, команда **cd** делает текущим домашний каталог (`$HOME` в оболочках **ksh** и **bsh**, `$home` в оболочке **csh**). Если указан полный путь, он становится текущим каталогом. Полный путь начинается с косой черты (`/`), обозначающей корневой каталог `/(root)`, точки (`.`), обозначающей текущий каталог, или двух точек (`..`) обозначающих родительский каталог. Если указан неполный путь, команда **cd** ищет указанный подкаталог в каталогах, перечисленных в переменной оболочки *$CDPATH* (*$cdpath* в оболочке **csh**). Синтаксис и семантика этой переменной такие же, как у переменной оболочки *$PATH* (*$path* в переменной **csh**).

Ниже приведены примеры применения команды **cd**:

- Для перехода в домашний каталог введите:
  ```
  cd
  ```
- Для перехода в каталог `/usr/include` введите:
  ```
  cd /usr/include
  ```
- Для перехода в подкаталог `sys` введите:
  ```
  cd sys
  ```

Если текущий каталог `/usr/include` содержит подкаталог `sys`, то текущим станет каталог `/usr/include/sys`.

- Для перехода в родительский каталог введите:

  ```
  cd ..
  ```

  Специальное имя файла - две точки (**..**) обозначают переход на один уровень вверх, в родительский каталог.

Сведения о синтаксисе приведены в описании команды **cd** в книге *Справочник по командам, том 1*.

Копирование каталогов (команда cp):

Команда **cp** копирует файл или каталог, указанный в параметре *Исходный_файл* или *Исходный_каталог*, в файл или каталог, заданный в параметре *Целевой_файл* или *Целевой_каталог*.

Если *Целевой_файл* существует, команда заменяет старое содержимое файла. При копировании нескольких *Исходных_файлов* должен быть указан целевой каталог.

Для помещения копии файла *Исходный_файл* в определенный каталог укажите путь к этому каталогу в параметре *Целевой_каталог*. При копировании в другой каталог файлы сохраняют свои имена, если только в конце пути не было указано новое имя файла. Команда **cp** также копирует полные каталоги в другие каталоги. Для этого нужно указать флаг **-r** или **-R**.

Ниже приведены примеры применения команды **cp**:

- Для того чтобы скопировать все файлы из каталога `/home/accounts/customers/orders` в каталог `/home/accounts/customers/shipments`, введите:

  ```
  cp /home/accounts/customers/orders/* /home/accounts/customers/shipments
  ```

  Эта команда копирует файлы (без каталогов) из каталога `orders` в каталог `shipments`.
- Для того чтобы скопировать каталог вместе со всеми файлами и подкаталогами в другой каталог, введите:

  ```
  cp -R /home/accounts/customers /home/accounts/vendors
  ```

  Каталог `customers` вместе со всеми содержащимися в нем файлами, подкаталогами и файлами в этих подкаталогах будет скопирован в каталог `vendors`.

Сведения о синтаксисе приведены в описании команды **cp** в книге *Справочник по командам, том 1*.

Просмотра содержимого каталога (команда ls):

Для просмотра содержимого каталога введите команду **ls**.

Команда **ls** записывает в стандартный вывод содержимое каждого из указанных *Каталогов* или имя каждого из указанных *Файлов*, а также дополнительную информацию, определяемую флагами. Если ни *Файл*, ни *Каталог* не указан, команда **ls** показывает содержимое текущего каталога.

По умолчанию команда **ls** сортирует всю информацию по именам файлов в алфавитном порядке. Если команда выполняется администратором, то по умолчанию применяется флаг **-A** и выводятся все записи, за исключением точки (**.**) и двух точек (**..**). Для просмотра всех файлов, включая те, имена которых начинаются с точки (**.**)(точки), введите команду **ls -a**.

Вы можете отформатировать вывод одним из следующих образов:

- Для размещения по одной записи на каждой строке укажите флаг **-l**.

- Для вывода списка в виде нескольких столбцов укажите флаг **-C** или **-x**. При выводе на терминал флаг **-C** установлен по умолчанию.
- Для разделения записей запятыми укажите флаг **-m**.

Число символов в строке вывода команда **ls** получает из переменной среды *$COLUMNS*. Если эта переменная не задана, команда читает файл `terminfo`. Если команда **ls** не может определить число символов в строке ни одним из этих способов, принимается значение по умолчанию, равное 80.

Информация, показанная с флагом **-e** или **-l**, означает следующее:

Первый символ равен:

Элемент	Описание
d	Запись является каталогом.
b	Запись является специальным блочным файлом.
c	Запись является специальным символьным файлом.
l	Запись является символьной связью.
p	Запись является специальным файлом конвейера (FIFO).
s	Запись является локальным сокетом.
-	Запись является обычным файлом.

Следующие девять символов разделены на три набора по три символа в каждом. Первые три символа показывают права доступа владельца файла или каталога. Следующие три символа показывают права доступа прочих пользователей группы. Последние три символа показывают права доступа всех остальных пользователей. Символы каждого набора показывают права доступа на чтение, запись и выполнение файла. Права доступа на выполнение для каталога разрешают поиск файла в каталоге.

Права доступа обозначаются следующим образом:

Элемент	Описание
r	Предоставлены права доступа для чтения
t	Только владелец каталога или файла может удалять или переименовывать файлы в каталоге, даже если у других пользователей есть права доступа для записи в каталог.
w	Предоставлены права доступа для записи (редактирования).
x	Предоставлены права доступа для выполнения (поиска).
-	Соответствующие права доступа не предоставлены.

С флагом **-e** выводится та же информация, что и с флагом **-l**, но добавляется 11-й символ, означающий следующее:

Элемент	Описание
+	Показывает, что для файла есть расширенная информация о защите. Например, для файла может существовать расширенный **ACL**, атрибуты режима **TCB** или **TP**.
-	Показывает, что для файла нет расширенной информации о защите.

Команда **ls** также показывает полное число блоков в файле, включая те, что заняты файлом косвенно.

Примеры:

- Для просмотра всех файлов, находящихся в текущем каталоге, введите:
  ```
  ls -a
  ```
  При этом будут показаны все файлы, включая
  - точка (`.`)
  - две точки (`..`)
  - Прочие файлы, имена которых могут начинаться или нет с точки (`.`)

- Для просмотра подробной информации введите:
  ```
  ls -l chap1 .profile
  ```

  Будет показан расширенный список с подробной информацией о файлах `chap1` и `.profile`.
- Для просмотра подробной информации о каталоге введите:
  ```
  ls -d -l . manual manual/chap1
  ```

  Будет показана подробная информация о содержимом каталогов `.` (точка) и `manual`, а также о файле `manual/chap1`. Без флага **-d** был бы показан список с подробной информацией о файлах в каталогах `.` и `manual`, а не подробная информация о самих каталогах.

Сведения о синтаксисе приведены в описании команды **ls** книги *Справочник по командам, том 3*.

Удаление каталогов (команда rmdir):

Команда **rmdir** удаляет из системы указанный *каталог*.

Каталог должен быть пуст (может содержать только `.` и две точки (`..`). Кроме того, у вас должны быть права на запись в его родительский каталог. Для того чтобы убедиться, что каталог пуст, вызовите команду **ls -a** *каталог*.

Ниже приведены примеры применения команды **rmdir**:

- Для удаления каталога со всем его содержимым введите:
  ```
  rm mydir/* mydir/.*
  rmdir mydir
  ```

  Сначала будет удалено содержимое каталога `mydir`, а затем и сам каталог. Команда **rm** выдаст сообщение об ошибке при попытке удалить каталоги точка (`.`) и две точки (`..`), а затем команда **rmdir** удалит эти записи и сам каталог.

  Примечание: Команда **rm mydir/* mydir/.*** удаляет файлы, имена которых начинаются с точки, в последнюю очередь. Помните, что в каталоге всегда могут находиться файлы, имена которых начинаются с точки, однако команда **ls** показывает их только с флагом **-a**.
- Для удаления каталога `/tmp/jones/demo/mydir` и всех его подкаталогов введите:
  ```
  cd /tmp
  rmdir -p jones/demo/mydir
  ```

  Каталог `jones/demo/mydir` будет удален из каталога `/tmp`. Если удаляемый каталог не пуст или если у вас нет прав на запись в него, команда будет завершена с соответствующим сообщением об ошибке.

Сведения о синтаксисе приведены в описании команды **rmdir** в книге *Справочник по командам, том 4*.

Сравнение содержимого каталогов (команда dircmp):

Команда **dircmp** сравнивает два указанных каталога, *Каталог1* и *Каталог2*, и записывает результат в стандартный вывод.

Сначала команда **dircmp** сравнивает имена файлов в обоих каталогах. При обнаружении файлов с одинаковыми именами, команда **dircmp** сравнивает содержимое этих файлов.

Сначала команда **dircmp** показывает список файлов, уникальных для каждого из каталогов. Затем она перечисляет файлы с одинаковыми именами в обоих каталогах, но с разным содержимым. Если никакие флаги не указаны, команда также показывает файлы с одинаковым содержимым и именами в обоих каталогах.

Ниже приведены примеры применения команды **dircmp**:

- Для создания отчета о различиях между файлами в каталогах `proj.ver1` и `proj.ver2` введите:

  ```
  dircmp proj.ver1 proj.ver2
  ```

  Будет показан отчет о различиях между каталогами `proj.ver1` и `proj.ver2`. В полученном отчете отдельно будут перечислены файлы, найденные только в одном из каталогов, и файлы, существующие в обоих. Если файл содержится в обоих каталогах, команда **dircmp** указывает, совпадают ли его копии.
- Для просмотра подробных сведений о различиях между файлами в каталогах `proj.ver1` и `proj.ver2` введите:

  ```
  dircmp -d -s proj.ver1 proj.ver2
  ```

  Флаг **-s** отменяет вывод информации об одинаковых файлах. Флаг **-d** указывает на необходимость распечатки результатов сравнения файлов командой **diff**.

Сведения о синтаксисе приведены в описании команды **dircmp** в книге *Справочник по командам, том 2*.

Обзор команд для работы с файловыми системами и каталогами:

Ниже описаны команды для работы с файловыми системами и каталогами, команды для выполнения процедур по обработке каталогов и список сокращений каталогов.

Таблица 66. Обзор команд для работы с файловыми системами

Элемент	Описание
df	Сообщает информацию о распределении памяти в файловых системах.

Таблица 67. Сокращения каталогов

Элемент	Описание
`.`	Текущий каталог.
`..`	Каталог, родительский по отношению к текущему.
`~`	Домашний каталог. (Кроме оболочки Bourne. Дополнительная информация приведена в разделе “Оболочка Bourne” на стр. 258).
`$HOME`	Домашний каталог. (Для всех оболочек).

Таблица 68. Обзор команд для процедур обработки каталогов

Элемент	Описание
cd	Изменяет текущий каталог.
cp	Копирует файлы или каталоги.
dircmp	Сравнивает два каталога и содержимое их общих файлов.
ls	Показывает содержимое каталога.
mkdir	Создает один или несколько каталогов.
mvdir	Перемещает (переименовывает) каталог.
pwd	Показывает путь к текущему каталогу.
rmdir	Удаляет каталог.

Управление рабочей схемой

Управление рабочей схемой (WLM) предоставляет системному администратору возможность более точно управлять распределением ресурсов между процессами, выполняемым планировщиком и Администратором виртуальной памяти (VMM).

С помощью WLM можно предупредить возникновение конфликтов между процессами и выделять ресурсы в зависимости от задач, решаемых разными группами пользователей.

Внимание: Для эффективного использования WLM необходимы глубокие знания о текущей производительности и процессах системы. Неправильная настройка WLM может существенно снизить производительность.

Основная область применения WLM - большие системы. В таких системах одновременно действует большое число различных серверов (например, серверов печати, баз данных, систем управления пользователями и систем обработки транзакций), что позволяет снизить стоимость эксплуатации системы. Рабочие схемы этих серверов могут конфликтовать, причем каждый из этих серверов выполняет свои задачи и предъявляет свои собственные требования к объему ресурсов.

Кроме перечисленных задач, WLM позволяет разделять группы пользователей с разным уровнем использования системных ресурсов. Это позволяет обеспечить нормальную работу задач определенного класса (например, интерактивных задач или задач с низким коэффициентом использования CPU) при наличии в системе других задач (например, пакетных или задач, использующих большой объем памяти).

Кроме того, взаимодействие WLM с подсистемой учета позволяет вести учет ресурсов для классов WLM, а не только для пользователей и групп.

Понятия, связанные с данным:

“Учет ресурсов системы” на стр. 156
Утилита учета ресурсов системы позволяет собирать данные и создавать отчеты об индивидуальном и групповом использовании различных ресурсов системы.

Концепции управления рабочей схемой

WLM позволяет определять различные классы обслуживания заданий, а также задавать атрибуты этих классов.

В число этих атрибутов входят минимальные и максимальные объемы ресурсов CPU, физической памяти и дискового ввода-вывода, которые выделяются классу. WLM автоматически классифицирует задания в соответствии с правилами, заданными системным администратором. Правила классификации основаны на значениях набора атрибутов процесса. Системный администратор или пользователь с соответствующими правами доступа может вручную изменить класс задания, назначенный автоматически.

Понятия, связанные с данным:

“Управление WLM” на стр. 490
Администратор рабочей схемы (WLM) позволяет системным администраторам управлять тем, как планировщик и администратор виртуальной памяти (VMM) выделяют ресурсы процессам. С помощью WLM можно предупредить возникновение конфликтов между процессами и выделять ресурсы в зависимости от задач, решаемых разными группами пользователей.

Терминология управления рабочей схемой:

В данной таблице приведены сведения об общих терминах, связанных с управлением нагрузкой.

Элемент	Описание
класс	*Класс* - это группа процессов и связанных с ними нитей. Для всех процессов класса задаются одинаковые ограничения на использование ресурсов и число общих ресурсов. Термин *класс* относится как к подклассам, так и к суперклассам.
суперкласс	*Суперкласс* - это класс, на основе которого созданы подклассы. Любой процесс, входящий в суперкласс, относится к одному из его подклассов. С суперклассом связан набор правил классификации, применяемых для выбора подкласса процесса. Кроме того, с суперклассом связан набор ограничений и целевых значений, определяющих объем ресурсов, доступных процессам суперкласса. Эти ресурсы распределяются по подклассам в соответствии с ограничениями и целевыми значениями, заданными для различных подклассов.
подкласс	*Подкласс* - это класс, связанный ровно с одним суперклассом. Все процессы подкласса также являются элементами суперкласса. Подклассам доступны только те ресурсы, которые выделены суперклассу. С подклассом связан набор правил классификации, описывающих относящиеся к нему процессы суперкласса. Кроме того, с подклассом связан набор ограничений и целевых значений, определяющих объем ресурсов, доступных процессам подкласса. Ограничения и целевые значения указывают, какая доля ресурсов, выделенных суперклассу, будет предоставлена процессам подкласса. Для управления WLM можно воспользоваться SMIT или командной строкой.

Элемент	Описание
механизм классификации	*Механизм классификации* - это набор правил классификации, в соответствии с которыми определяются суперкласс и подкласс каждого процесса.
правило классификации	*Правило классификации* задает набор значений атрибутов процесса, при которых процесс относится к указанному классу.
значение атрибута процесса	С каждым процессом связан некоторый набор *значений атрибутов*. К атрибутам процесса относятся идентификатор пользователя, идентификатор группы и имя исполняемого файла.
ограничения на использование ресурсов	*Ограничения на использование ресурсов* задают диапазоны, в которых должны находиться показатели использования ресурсов. Эти значения никак не связаны с ограничениями, задаваемыми функцией **setrlimit**.
доля общих ресурсов	*Доля общих ресурсов* определяет долю ресурсов, доступных классу (подклассу или суперклассу). Это значение применяется для оптимального распределения ресурсов между классами одного уровня с одинаковым приоритетом.
показатель использования ресурса	*Показатель использования ресурса* задает долю ресурса, выделенную процессу или группе процессов системы. Данное значение подсчитывается для отдельных процессов или для групп процессов в зависимости от текущего уровня сбора информации о ресурсах процессов.
уровень сбора информации о ресурсах	*Уровень сбора информации о ресурсах* определяет уровень объектов, для которых собирается информация об использовании ресурсов, и на которые накладываются ограничения. Это могут быть отдельные процессы класса, процессы, принадлежащие определенному пользователю или все процессы класса. В настоящее время информация об использовании ресурсов собирается только на уровне классов.
свойства класса процесса	*Свойства класса процесса* - это набор свойств, которые присваиваются процессу в зависимости от его класса.
права доступа к классу	*Права доступа к классу* - это набор правил, по которым определяется, какие пользователи и группы могут выполнять операции над классом, его процессами и нитями. В их число входят права на перенос процессов в класс вручную и права на создание подклассов в суперклассе.
приоритет класса	*Приоритет класса* определяет преимущество класса перед другими классами при получении ресурсов. Вначале ресурсы распределяются (с учетом заданных ограничений) между классами с наиболее высоким приоритетом. Приоритеты можно задавать как для суперклассов, так и для подклассов. Ресурсы распределяются между суперклассами в соответствии с их приоритетами. Затем ресурсы, полученные каждым суперклассом, распределяются между его подклассами в соответствии с приоритетами этих подклассов. Таким образом, распределение ресурсов определяется в основном приоритетами суперклассов, и в меньшей степени - приоритетами подклассов.

Классы для управления рабочей схемой:

WLM позволяет системным администраторам определить классы и задать для каждого из них набор атрибутов и ограничений на использование ресурсов.

Класс, к которому относится процесс, определяется в соответствии с критериями, заданными системным администратором. Ограничения на использование ресурсов и целевые объемы ресурсов действуют для всех процессов класса. Создание классов обслуживания и управление распределением ресурсов на уровне классов приложений позволяет предотвратить конфликты приложений с различными режимами доступа к ресурсам.

WLM поддерживает двухуровневую иерархию классов:

- Сначала ресурсы системы распределяются между суперклассами в соответствии с параметрами каждого суперкласса. Параметры использования ресурсов задаются системным администратором.
- В суперклассе, в свою очередь, можно определить несколько подклассов. Ресурсы, полученные суперклассом, распределяются между его подклассами в соответствии с параметрами этих подклассов.
- Системный администратор может передать права на управление подклассами суперкласса *администратору суперкласса* или группе администраторов.
- WLM поддерживает до 69 суперклассов (64 пользовательских) и 64 подклассов в суперклассе (61 пользовательский).
- В зависимости от назначения системы, системный администратор может определить только суперклассы, или суперклассы, содержащие подклассы.

Примечание: В этом разделе термин *класс* относится как к подклассам, так и к суперклассам. При необходимости применяется более точный термин.

Классификация процессов для управления рабочей схемой:

Классификация процессов выполняется в соответствии с правилами, заданными системным администратором. Критерием классификации является набор атрибутов процесса, таких как идентификатор пользователя и группы, имя исполняемого файла, тип процесса и тег приложения.

Суперкласс процесса выбирается по заданным правилам классификации. Если в суперклассе есть подклассы, то с ним связан еще один набор правил для выбора подкласса, к которому относится процесс. В процессе автоматической классификации учитываются значения атрибутов **наследования** подкласса и суперкласса.

Автоматическая классификация выполняется при вызове функции **exec**. Кроме того, класс процесса повторно определяется после вызова функции, которая может повлиять на атрибуты классификации процесса. Примерами таких функций могут служить **setuid**, **setgid**, **setpri** и **plock**.

Пользователь с соответствующими правами доступа может вручную изменить суперкласс или подкласс процесса (группы процессов), назначенный автоматически.

Понятия, связанные с данным:

“Атрибуты классов” на стр. 504
Перечень всех атрибутов класса WLM.

Управление ресурсами:

WLM управляет ресурсами двумя способами: с помощью абсолютных и относительных (в процентах) значений.

На основании относительных значений можно управлять следующим ресурсами:

- Использование процессора для обработки в классе нитей типа SCHED_OTHER. Подсчитывается общее число тактов процессора, затраченных на выполнение всех нитей класса. Настройка нитей с фиксированным приоритетом невозможна. Таким образом, их нельзя изменить и они могут превысить ограничения на использование процессора.
- Физической памятью. Подсчитывается суммарное число страниц памяти, принадлежащих всем процессам класса.
- Интенсивностью дискового ввода-вывода. Под интенсивностью понимается общее число блоков размером 512 байт, считываемых с диска и записываемых на диск всеми нитями класса в секунду.

Ресурсами можно управлять на основании двух типов абсолютных значений: для класса и для процесса. В число абсолютных значений для класса входят:

Число процессов в классе
Число активных процессов в классе.

Число нитей в классе
Число активных нитей в классе.

Число входов в класс
Число активных сеансов пользователей в классе.

В число абсолютных значений для процесса входят:

Общее время CPU
Общее время, затраченное CPU на обработку одного процесса.

Общий объем ввода-вывода
Общее число блоков ввода-вывода при обработке одного процесса.

Общее время подключения
Общее время, в течение которого активен сеанс входа в систему.

Параметры использования ресурсов:

WLM позволяет системным администраторам задавать параметры использования ресурсов различных типов отдельно для каждого класса.

Эти параметры включают:

- Целевой объем используемых ресурсов различных типов. Целевой объем ресурсов задается в виде доли общих ресурсов. Доля общих ресурсов представляет собой относительный объем ресурсов, выделяемых разным классам. Например, если доля времени CPU двух классов равна 1 и 3, и в системе нет других активных классов, то этим процессам будет выделено 25% и 75% процессорного времени соответственно. Совокупное потребление ресурсов зависит от числа активных общих ресурсов и объема ресурсов, выделенного для данного приоритета.
- Минимальное и максимальное ограничение. Эти ограничения задаются в процентах от общего объема ресурсов. WLM поддерживает максимальные ограничения двух типов:
  - Гибкий максимум задает максимальный объем ресурсов, который может быть предоставлен при наличии запросов от других классов. Если другие запросы отсутствуют, этот максимум может быть превышен.
  - Жесткий максимум задает максимальный объем ресурсов, который не может быть превышен ни при каких условиях. Обратите внимание, что нити с фиксированным приоритетом не подчиняются этим правилам и могут превышать установленные ограничения.
- Общие ограничения. Общие ограничения соблюдаются в точности. Процесс, превысивший какое-либо ограничение, прерывается. Если достигнуто общее ограничение для класса, то новые ресурсы, которые превысили бы это ограничение, не создаются.

Как правило, гибких ограничений достаточно для управления ресурсами. Жесткое ограничение максимума может привести к появлению незанятых ресурсов без обоснованной причины. При задании жесткого максимума следует проявлять особую осторожность, поскольку неверное значение может привести к существенному снижению производительности. Кроме того, общие ограничения также следует использовать обоснованно, так как они могут вызывать прерывание процессов.

В активном состоянии WLM пытается обеспечивать максимальный объем ресурсов для каждого класса. Поскольку на некоторые предельные значения накладываются дополнительные ограничения, сумма этих значений может превышать 100 процентов. В этом случае, если активны все классы, то достигнуть предельного значения для каждого класса невозможно. WLM управляет потреблением ресурсов процессора путем изменения приоритетов планирования нитей, приоритет которых не фиксирован, в зависимости от параметров производительности классов, которым эти нити принадлежат. Такой подход позволяет обеспечить равномерную загрузку процессора в течение длительного периода времени.

Например, если единственный активный класс - это класс A, минимальное значение для процессора равно 0%, а целевое - 60, то этот класс получит 100% процессор. Если становится активен класс B с минимумом 0% и целевым значением 40, то классу A будет выделено 60 процентов процессорного времени, а классу B - 40%. Система достигнет стабильного распределения 60 на 40 процентов в течение нескольких секунд.

В этом примере предполагается, что конфликтов между классами нет. В реальных условиях ограничения для процессора и памяти взаимозависимы. Например, класс может не достигнуть даже минимального ограничения процессора, если для него заданы слишком низкие ограничения на память.

Для настройки определения и ограничения классов для некоторого набора приложений WLM содержит инструмент **wlmstat**, позволяющий узнать объем ресурсов, занятый каждым классом. Для отслеживания динамических параметров системы предназначен графический инструмент **wlmmon**.

Ограничения виртуальной памяти Администратора рабочей схемы:

Ограничение виртуальной памяти Администратора рабочей схемы (WLM) позволяет администраторам предотвратить системный сбой или падение производительности из-за чрезмерной подкачки с помощью применения ограничения виртуальной памяти к классу или процессу.

При превышении ограничения WLM принимает меры, выполняя одно из следующих действий:

- убиение всех процессов в классе WLM, превышающих данное ограничение
- убиение только процессов, из-за которых применение классов WLM превысило ограничение
- убиение процесса, превысившего ограничение

Ограничения виртуальной памяти можно указать для любого заданного пользователем класса, любого подкласса по умолчанию в заданном пользователем суперклассе и для суперклассе по ум олчанию.

Для ведения учета WLM рассматривает только следующие источники в качестве виртуальной памяти при определении использования класса или процесса WLM:

- куча
- инициализированные данные загрузчика, BSS, общая библиотека и частные загружаемые сегменты
- UBLOCK и области mmap
- большие и закрепленные страницы пользовательского пространства

Администратор может указать ограничение виртуальной памяти WLM для класса или для каждого процесса в классе. При превышении ограничения класса WLM может убить все процессы, присвоенные классу, или убить только процессы, которые привели к превышению ограничения, в зависимости от того, имеет ли атрибут **vmenforce** класса значение `class` или `proc`, соответственно. По умолчанию убиваются только процесс, приведший к переполнению ограничения. Процесс убивается если использования виртуальной памяти процессом превышает ограничение.

Режимы работы для WLM:

WLM позволяет управлять общим и процентным потреблением ресурсов классами, а также общим потреблением ресурсов процессами. WLM в активном режиме позволяет включить регулирование потребления всех типов ресурсов.

В WLM есть режим, позволяющий отслеживать и классифицировать новые и существующие процессы, не предпринимая попыток управлять потреблением ресурсов. Этот режим называется *пассивным*.

Пассивный режим отслеживает потребление ресурсов *без вмешательства* WLM и применяется, например, при определении будущих ограничений для новой системы. Полученные значения позволяют задать правильные ограничения, обеспечивающее приоритетное выполнение важных приложений.

Если вы планируете управлять только доступом к ресурсам процессора, то можно запустить WLM в активном режиме для процессора и в пассивном режиме для всех остальных режимов. Такой режим называется *только cpu*. Если необходимо управлять только процентными значениями для каждого класса, то учет и настройку общих значений для классов и процессов можно выключить. Все режимы WLM позволяют отключить связывание набора ресурсов.

Динамическое управление WLM:

Во время работы WLM можно изменить любой параметр текущей конфигурации. Изменения могут включать в себя изменение параметров любого класса, добавление новых классов или удаление существующих.

Изменения могут вноситься следующими способами:

- Путем изменения файлов свойств активной конфигурации (в каталоге, на который указывает символьная связь `/etc/wlm/current`) и обновления WLM командой **wlmcntrl**.
- Путем создания новой конфигурации и загрузки этой конфигурации WLM.
- С помощью интерфейса командной строки WLM (командами **mkclass**, **chclass** и **rmclass**).
- С помощью API WLM.

Для автоматического переключения на новую конфигурацию в заданное время дня можно применять *наборы конфигурации*. Эти наборы позволяют администраторам задать наборы конфигурации и диапазоны времени, когда они должны применяться.

Инструменты мониторинга:

С помощью этих команд WLM можно просмотреть статистику WLM и отследить операции WLM.

- Команда **wlmstat** выводит статистику (доля ресурсов, используемых различными классами) в текстовом виде.
- Команда **wlmmon** показывает распределение ресурсов по классам в графическом виде.
- Команда **wlmperf** - это дополнительное средство, входящее в состав программы Performance Toolbox. Оно предоставляет дополнительные возможности, такие как ведение хронологии операции и последующий просмотр хронологии.

Совместимость с предыдущими версиями в Администраторе рабочих схем:

В этом случае вывод команды **wlmstat** по умолчанию содержит только суперклассы и похож на вывод этой команды в предыдущих версиях.

Например:

```
# wlmstat
               CLASS  CPU  MEM  DKIO
        Unclassified    0    0   0
           Unmanaged    0    0   0
             Default    0    0   0
              Shared    0    2   0
              System    2   12   0
              class1    0    0   0
              class2    0    0   0
#
```

Если для некоторых суперклассов администратор WLM определит подклассы, они будут добавлены в вывод команды. Например:

```
# wlmstat
               CLASS  CPU  MEM  DKIO
        Unclassified    0    0   0
           Unmanaged    0    0   0
             Default    0    0   0
              Shared    0    2   0
              System    3   11   7
              class1   46    0   0
    class1.Default     28    0   0
     class1.Shared      0    0   0
        class1.sub1    18    0   0
              class2   48    0   0
#
```

Вывод этой команды аналогичен выводу команды **ps**. Для процессов, суперкласс которых не содержит дополнительных подклассов, команда **ps** показывает в поле класса только имя суперкласса.

Учет ресурсов для классов:

Система учета ресурсов AIX позволяет собирать статистическую информацию о ресурсах конкретного пользователя, группы или класса WLM.

Если функция учета ресурсов процессов включена, операционная система собирает информацию о ресурсах, используемых процессом, и записывает ее в файл при завершении процесса. Запись учета ресурсов содержит 64-значный ключ, идентифицирующий класс WLM, к которому относится процесс.

64-разрядное число используется вместо полного 34-символьного имени класса для экономии дискового пространства (в противном случае, размер записи учета ресурсов увеличился бы вдвое). При создании отчета о ресурсах процесса команда учета ресурсов преобразует ключ в имя класса. При преобразовании применяются определения классов, заданные в файлах конфигурации WLM. Таким образом, если класс был удален после создания записи, но до завершения процесса, его имя не будет найдено, и класс будет показан в отчете как `Неизвестный`.

Для того чтобы в записях учета ресурсов правильно указывалось имя удаленного класса, выполните одно из следующих действий:

- Не удаляйте класс из файла классов; вместо этого удалите ссылки на него из файлов правил. Само определение класса можно удалить после создания отчета.
- Удалите класс из действующего файла конфигурации и добавьте его определение в файл classes фиктивной конфигурации (конфигурации, которая никогда не активируется). После создания отчета это определение можно будет удалить.

Понятия, связанные с данным:

“Учет ресурсов системы” на стр. 156
Утилита учета ресурсов системы позволяет собирать данные и создавать отчеты об индивидуальном и групповом использовании различных ресурсов системы.

Управление WLM

Администратор рабочей схемы (WLM) позволяет системным администраторам управлять тем, как планировщик и администратор виртуальной памяти (VMM) выделяют ресурсы процессам. С помощью WLM можно предупредить возникновение конфликтов между процессами и выделять ресурсы в зависимости от задач, решаемых разными группами пользователей.

WLM позволяет определять различные классы обслуживания заданий, а также задавать атрибуты этих классов. В число этих атрибутов входят минимальные и максимальные объемы ресурсов CPU, физической памяти и дискового ввода-вывода, которые выделяются классу. WLM автоматически классифицирует задания в соответствии с правилами, заданными системным администратором. Правила классификации основаны на значениях набора атрибутов процесса. Автоматическая классификация может быть изменена вручную системным администратором или пользователем с соответствующими правами доступа.

WLM - это часть базовой операционной системы, которая устанавливается вместе с операционной системой, но представляет собой дополнительную службу. Она должна быть настроена в соответствии с системной средой, причем ее следует запускать только когда вы планируете применять управление рабочей схемой, и останавливать, когда вы хотите приостановить или завершить работу со службой WLM.

В этом разделе описаны процедуры настройки классов и правил WLM, соответствующих вашей среде, а также приведены рекомендации по устранению неполадок, связанных с использованием ресурсов.

Внимание: Предполагается, что пользователь, выполняющий задачи этого раздела, знаком с принципами управления рабочей схемой. Для эффективного использования WLM необходимы глубокие знания о текущей производительности и процессах системы. Неправильная настройка WLM может существенно снизить производительность.

Понятия, связанные с данным:

“Концепции управления рабочей схемой” на стр. 484
WLM позволяет определять различные классы обслуживания заданий, а также задавать атрибуты этих классов.

Запуск и завершение управления рабочей схемой:

WLM - это одна из дополнительных служб, которую можно запускать и останавливать.

Для запуска и остановки WLM рекомендуется применять один из интерфейсов управления системой, т.е. SMIT.

- Для запуска и завершения работы WLM с помощью SMIT введите команду `smit wlmmanage`.

Основное различие между этими опциями заключается в том, на какой срок вносится изменение. Существует три способа запуска или остановки WLM с помощью SMIT:

текущий сеанс
: Если вы запросили остановку WLM с этой опцией, то служба WLM будет остановлена в текущем сеансе и снова запущена при следующей загрузке. Если вы запросили запуск WLM с этой опцией, то служба будет запущена в текущем сеансе, но не будет запускаться при следующей загрузке.

следующая загрузка
: Если вы запросили остановку WLM с этой опцией, то служба WLM продолжит работу в текущем сеансе, и *не будет* запущена при следующей загрузке. Если вы запросили запуск WLM с этой опцией, то служба не будет доступна в текущем сеансе, однако будет запущена при следующей загрузке.

оба варианта
: При остановке WLM с такой опцией служба WLM будет остановлена в текущем сеансе и *не будет* запущена при следующей загрузке. Если вы запросили запуск WLM с этой опцией, то служба будет доступна как в текущем сеансе, *так и* после следующей загрузки.

Вы также можете воспользоваться командой **wlmcntrl**, однако команда **wlmcntrl** позволяет запускать и останавливать WLM только в текущем сеансе. Если вы хотите воспользоваться интерфейсом командной строки и хотите, чтобы изменения действовали после перезагрузки системы, то необходимо внести изменения в файл `/etc/inittab`

WLM позволяет управлять общим и процентным потреблением ресурсов классами, а также общим потреблением ресурсов процессами. Для управления всеми типами ресурсов необходимо запустить WLM в *активном* режиме. В WLM есть режим, позволяющий отслеживать и классифицировать новые и существующие процессы, не предпринимая попыток управлять потреблением ресурсов. Этот режим называется *пассивным*. Если вы планируете управлять только доступом к ресурсам процессора, то можно запустить WLM в активном режиме для процессора и в пассивном режиме для всех остальных режимов. Такой режим называется *только cpu*.

Все процессы, которые были запущены до начала работы WLM, классифицируются по загруженным правилам присвоения, и WLM отслеживает их работу.

Свойства WLM:

Свойства WLM можно задать с помощью SMIT или интерфейса командной строки, а также путем создания текстовых файлов конфигурации. Интерфейсы SMIT с помощью команд WLM сохраняют конфигурацию в этих текстовых файлах, называемых *файлами свойств*.

Совокупность файлов свойств WLM составляет конфигурацию WLM. Можно создать несколько наборов файлов свойств, определив тем самым несколько различных конфигураций управления рабочей схемой. Эти наборы обычно хранятся в подкаталогах каталога **/etc/wlm**. Суперклассы конфигурации *Config* описаны в файлах свойств WLM *classes*, *description*, *limits*, *shares* и *rules* в каталоге `/etc/wlm/Config`. Подклассы

суперкласса *Super* данной конфигурации описаны в файлах *classes*, *limits*, *shares* и *rules* в каталоге `/etc/wlm/Config/Super`. Запустить или остановить WLM, а также изменить конфигурацию может только пользователь root.

Ниже перечислены имена файлов свойств:

Элемент	Описание
classes	Определения классов
description	Описания конфигурации
groupings	Группы значений атрибутов
limits	Ограничения для классов
shares	Описания целевых общих ресурсов для класса
rules	Правила назначения классов

Команда **wlmcntrl**, которая загружает файл свойств WLM, а также другие команды WLM позволяют указывать имя каталога, из которого будут считываться файлы свойств. Это позволяет изменять конфигурацию WLM, не меняя значений по умолчанию.

Символьная связь `/etc/wlm/current` указывает на каталог, содержащий текущие файлы конфигурации. При запуске команды **wlmcntrl** с заданной конфигурацией эта связь обновляется. Примеры файлов конфигурации, поставляемые с операционной системой, располагаются в каталоге `/etc/wlm/standard`.

Создание групп значений атрибутов:

Вы можете группировать значения атрибутов и задавать для таких групп отдельное значение в файле `rules`. Такие *группы значений атрибутов* определяются в файле `groupings` в структуре каталогов конфигурации WLM.

По умолчанию в конфигурации нет файла `groupings`. Специальные команды или утилиты для создания этого файла не предусмотрены. Для создания и применения групп значений атрибутов выполните следующие действия:

1. Войдя в систему под именем пользователя с правами доступа root, перейдите в нужный каталог конфигурации, например:
   ```
   cd /etc/wlm/MyConfig
   ```
2. С помощью текстового редактора создайте и отредактируйте файл `groupings`. Например:
   ```
   vi groupings
   ```
3. Атрибуты и значения определяются следующим образом:

   атрибут = *значение*, *значение*, ...

   Все значения должны быть разделены запятыми. Пробелы не учитываются. Допускается задание диапазонов и символов подстановки. Например:
   ```
   trusted = user[0-9][0-9], admin*
   nottrusted = user23, user45
   shell = /bin/?sh, \
           /bin/sh, \
           /bin/tcsh
   rootgroup=system,bin,sys,security,cron,audit
   ```
4. Сохраните файл.
5. Для применения групп атрибутов в условиях выбора класса измените файл `rules`. Перед именами групп атрибутов необходимо указывать символ доллара ($) для включения соответствующих значений или восклицательный знак (!) для их исключения. Восклицательный знак нельзя применять при определении элементов групп (шаг 3). Он является единственным модификатором, допустимым перед группами в файле правил (rules). В следующем примере звездочка (*) указывает на то, что данная строка является комментарием.

```
* класс зарез. пользователь группа приложение         тип         тег
classA  -      $trusted,!$nottrusted  -           -                 -     -
classB  -      -                      -           $shell,!/bin/zsh  -     -
classC  -      -                      $rootgroup  -                 -     -
```

6. Сохраните файл.

Теперь в правилах классификации применяются группы значений атрибутов. При анализе правил система разворачивает элементы, начинающиеся с $, подставляя вместо них соответствующие значения из файла `groupings`. Если элемент синтаксически недопустим или не указан в файле `groupings`, то система выдает предупреждающее сообщение и продолжает обработку остальных правил.

Создание временной конфигурации:

Вы можете создать набор конфигураций и указать для каждой конфигурации набора дни и интервалы времени, когда должна применяться данная конфигурация.

Такие наборы, называемые *набором временных конфигураций*, независимы от обычной конфигурации, но совместимы с ней. Для переключения между обычной конфигурацией и набором конфигураций можно применять команду **wlmcntrl -u**.

Как правило, при использовании набора конфигураций уже существующие конфигурации связываются с определенным интервалом времени . Поскольку в каждый момент времени может применяться только одна конфигурация, то каждый интервал времени должен быть уникальным; перекрытие или совпадение интервалов не допускается.

Демон **wlmd** уведомляет WLM об окончании интервала времени текущей конфигурации и о необходимости применения другой конфигурации из набора. Управлять интервалами времени может только пользователь root, задавая их в текстовом файле `.times` в каталоге набора конфигураций.

Для создания наборов временных конфигураций выполните следующие действия:

1. Войдя в систему под именем пользователя с правами доступа root, создайте каталог набора конфигураций и перейдите в этот каталог. Например:

   ```
   mkdir /etc/wlm/MyConfigSet
   cd /etc/wlm/MyConfigSet
   ```

2. С помощью текстового редактора создайте файл `.times` и укажите в нем имена конфигураций и интервалы времени в следующем формате:

   ```
   имя-конфигурации:
               time = "N-N,чч:мм-чч:мм"
   ```

   или

   ```
   имя-конфигурации:
               time = -
   ```

   (без указания значения времени) здесь *N* - это число, задающее день недели в диапазоне от 0 (воскресенье) до 6 (суббота), *чч* - часы в диапазоне от 00 (полночь) до 23,а *мм* - минуты в диапазоне от 00 до 59. Вы можете указать только день или вообще не указывать значения. Значение часов 24 допустимо только со значением минут 00, указывая на конец выбранных суток. Если вместо диапазона для какой-либо конфигурации указан дефис (-), то эта конфигурация применяется в то время, когда не действует ни одна из оставшихся конфигураций. Без задания интервала времени можно указать только одну конфигурацию.

   Пример:

   ```
   conf1:
       time =
   conf2:
       time = "1-5,8:00-17:00"
   ```

```
conf2
    time = "6-0,14:00-17:00"
conf3
    time = "22:00-6:00"
```

3. Команда **wlmcntrl -u** позволяет обновить WLM с учетом изменений, внесенных в набор конфигураций. Например:

   ```
   wlmcntrl -u /etc/wlm/MyConfigSet
   ```

Теперь WLM применяет новый набор временных конфигураций.

Для создания наборов конфигураций и управления ими можно также применять команды **confsetcntrl** и **lswlmconf**. Пример:

Для создания набора конфигураций `confset1` с конфигурацией по умолчанию `conf1` введите следующую команду:

```
confsetcntrl -C confset1 conf1
```

Для добавления `conf2` в `confset1` и использования ее в качестве активной конфигурации ежедневно с 8:00 утра до 5:00 вечера введите следующую команду:

```
confsetcntrl -d confset1 -a conf2 "0-6,08:00-17:00"
```

Для Того чтобы сделать данный набор конфигураций активным, введите команду

```
wlmcntrl -d confset1
```

Создание набора ресурсов:

Применение наборов ресурсов позволяет эффективно разделить процессы на основании используемых ими процессоров. Разделив рабочие схемы на два класса и выделив каждому классу отдельное подмножество процессоров, можно устранить конкуренцию за процессоры между этими рабочими схемами, даже если они конкурируют за физическую память и ресурсы ввода-вывода.

Простейший способ создания набора ресурсов заключается в использовании команды SMIT (**smit addrsetcntl**) или команды **mkrset**.

В следующем примере описано создание набора ресурсов в четырехпроцессорной системе. Задача заключается в создании набора процессоров, включающего процессоры 0 - 2, и использовании этого набора в конфигурации WLM для предоставления всем процессам суперкласса только этих трех процессоров.

1. Войдя в систему под именем пользователя с правами доступа root, просмотрите доступные ресурсы, из которых можно будет создавать наборы, с помощью следующей команды:

   ```
   lsrset -av
   ```

   Будет показана примерно следующая информация:

   ```
   T  Имя               Влад.   Группа  Режим   CPU  Память  Ресурсы
   r  sys/sys0          root    system  r-----    4   98298  sys/sys0
   r  sys/node.00000    root    system  r-----    4   98298  sys/sys0
   r  sys/mem.00000     root    system  r-----    0   98298  sys/mem.00000
   r  sys/cpu.00003     root    system  r-----    1       0  sys/cpu.00003
   r  sys/cpu.00002     root    system  r-----    1       0  sys/cpu.00002
   r  sys/cpu.00001     root    system  r-----    1       0  sys/cpu.00001
   r  sys/cpu.00000     root    system  r-----    1       0  sys/cpu.00000
   ```

   В показанном списке **sys/sys0** соответствует всей системе (в данном случае это четырехпроцессорная система SMP). Если в классе WLM не указан атрибут **rset**, то по умолчанию применяется именно этот набор, к которому могут обращаться все процессы.

2. С помощью следующей команды SMIT создайте набор ресурсов с выбранным именем:

   ```
   smit addrsetcntl
   ```

В данном примере необходимо заполнить следующие поля:

Пространство имен
admin

Имя набора ресурсов
proc0_2

Ресурсы
Выберите в списке строки, соответствующие ресурсам памяти и процессорам 0 - 2 (sys/cpu.00000 - sys.cpu.00002).

Остальные поля
Выберите в списках необходимые значения.

После ввода значений и выхода из SMIT в `/etc/rsets` будет создан набор ресурсов `admin/proc0_2`

3. Для применения нового набора ресурсов добавьте его в структуры данных ядра с помощью следующей команды SMIT:

   ```
   smit reloadrsetcntl
   ```

   Это меню позволяет обновить базу данных `немедленно`, `при следующей загрузке` или `использовать оба варианта`. Поскольку это первое обращение к новому набору ресурсов, выберите опцию `оба варианта`, чтобы набор ресурсов был загружен немедленно и автоматически загружался при каждом запуске системы. (Если вы изменили уже существующий набор, то можно выбрать опцию `немедленно`.)

4. Добавьте новый набор ресурсов в класс WLM:

   ```
   smit wlmclass_gal
   ```

   Укажите класс (в нашем примере - **super1**), а затем выберите в списке **Набор ресурсов** значение **admin/proc0_2**. После выбора значений и выхода из SMIT файл `classes` на диске будет изменен.

5. Выполните одно из следующих действий:
   - Если WLM работает, то обновите конфигурацию с помощью следующей команды SMIT:

     ```
     smit wlmupdate
     ```

   - Если WLM не работает, то запустите его с помощью следующей команды SMIT:

     ```
     smit wlmstart
     ```

6. Проверьте эффективность нового набора ресурсов, созданного для класса. Например:
   a. Запустите в классе **super1** 90 зацикленных программ (программ, выполняющих бесконечный цикл).
   b. Введите команду `wlmstat`. Будет показана примерно следующая информация:

      ```
                    Класс  CPU Пам.B-B
            Unclassified    0    0   0
               Unmanaged    0    0   0
                 Default   8   0   0
                  Shared   0   0   0
                  System   0   0   0
                  super1  75   0   0
                  super2   0   0   0
        super2.Default     0   0   0
         super2.Shared     0   0   0
           super2.sub1     0   0   0
           super2.sub2     0   0   0
      ```

      Показанные данные свидетельствуют о том, что 90 процессов, связанных с выбранными процессорами, которые в иной ситуации могли бы использовать до 100% ресурсов процессоров, используют только 75%, поскольку набор ресурсов разрешает этим процессам работу только на процессорах 0 - 2.

   c. С помощью следующей команды SMIT проверьте, к какому набору ресурсов имеет доступ процесс (по ИД процесса).

      ```
      smit lsrsetproc
      ```

Укажите ИД проверяемого процесса или выберите его в списке. Ниже приведен пример вывода для одного из зацикленных процессов:

```
CPU  Память  Ресурсы
  3   98298  sys/mem.00000 sys/cpu.00002 sys/cpu.00001 sys/cpu.00000
```

Однако процесс, поступающий от класса без указанного атрибута **rset**, использует набор ресурсов `Default`, в который включены все процессоры за исключением тех, которые входят в исключительный набор ресурсов. Процесс, не принадлежащий ни одному классу, использует класс `System` (если это процесс пользователя root) или класс `Default` (если это не процесс пользователя root). Для любого из этих классов может быть определен набор ресурсов.

Следующие сведения относятся к процессу **init**, в классе которого не задан набор ресурсов:

```
CPU  Память  Ресурсы
  4   98298  sys/sys0
```

Вы создали набор ресурсов и включили его в один их классов WLM.

Примечание: WLM не применяет набор ресурсов для связывания процессов, уже связанных с помощью функции **bindprocessor** с другим набором ресурсов. После удаления связи с другим набором ресурсов, WLM автоматически активизирует связь со своим набором ресурсов.

Примечание: Наборы ресурсов можно создавать для каждого класса WLM, который содержит классы `Default` и `System`.

Понятия, связанные с данным:

“Реестр набора ресурсов в WLM” на стр. 517
Реестр **rset** позволяет системным администраторам определять наборы ресурсов и присваивать им имена.

Информация, связанная с данной:

lsrset, команда

Настройка WLM для объединения ресурсов:

Средства управления рабочей схемой (WLM) позволяют управлять ресурсами, которые применяются в различных заданиях системы.

Шаблон конфигурации WLM по умолчанию создается во всех устанавливаемых операционных системах AIX. Следующая процедура позволяет обновить шаблон конфигурации WLM и реализовать на общем сервере стратегию управления ресурсами. Полученная в результате конфигурация может применяться в качестве исходной точки тестирования. Окончательная точная настройка WLM будет определяться требованиями к стратегии и рабочей схемой, характерной для вашей системы.

Примечание:

1. Для эффективного использования WLM необходимы глубокие знания о текущей производительности и процессах системы. Для создания конфигурации, обеспечивающей требуемые результаты, может потребоваться несколько этапов тестирования и настройки. Настройка WLM с применением крайних или неточных значений может привести к существенному снижению производительности системы.
2. Настройка WLM упрощается, когда вам уже известно один или несколько атрибутов классификации процесса (например, пользователь, группа или имя приложения). Если вы не знакомы с текущими данными об использовании ресурсов, то с помощью таких инструментов, как **topas**, выявите процессы, использующие максимальный объем ресурсов, и считайте полученную информацию исходной точкой для определения классов и правил.
3. В следующем сценарии предполагается, что вы знакомы с основными принципами WLM, изложенными в книге “Концепции управления рабочей схемой” на стр. 484.

Файлы конфигурации WLM хранятся в каталоге `/etc/wlm/`*имя-конфигурации*. Каждый подкласс суперкласса определяется в файле конфигурации `/etc/wlm/`*имя-конфигурации/имя-суперкласса*. Дополнительная информация об этих файлах приведена в *Справочник по файлам* .

В следующей процедуре вы объедините рабочие схемы двух серверов разных отделов в одном более мощном сервере. В данном примере редактируются файлы конфигурации, но вы также можете создать конфигурацию с помощью команды быстрого доступа SMIT **smit wlmconfig_create**. В данной процедуре вы выполните следующие операция:

1. Определите требования объединяемых приложений к ресурсам. Это поможет вам определить, сколько приложений можно переместить на новый сервер.
2. Определите уровни для начала тестирования объединенной рабочей схемы, например, общие ресурсы и ограничения.
3. Выполните окончательную настройку для достижения требуемых результатов.

Описанная ниже процедура была протестирована в отдельных версиях AIX. Результаты, которые вы можете получить, в значительной степени зависят от конкретных версии и уровня AIX.

Шаг 1. Идентификация требований приложения

В данном сценарии предполагается, что вы работаете с рабочей схемой, типичной для сервера баз данных. Предполагается, что задания можно объединить в следующие категории:

Получатели запросов
Это процессы, большую часть времени простаивающие в ожидании запроса, и активирующиеся только в ответ на полученный запрос. Несмотря на то, что такие процессы не потребляют большой объем ресурсов, для них важно время отклика.

Обработчики запросов
Это процессы, выполняющие непосредственную обработку запросов, как локальных, так и удаленных. Такие процессы могут использовать большой объем ресурсов процессора и памяти.

Программы создания отчетов
Это процессы, выполняющие автоматизированные задачи. Они могут требовать существенного объема ресурсов процессора и памяти, но требования к времени отклика у них невысоки.

Мониторы
Это процессы, периодически запускаемые для проверки состояния системы или приложений. Такие процессы могут требовать существенного объема ресурсов памяти и процессора, но лишь на протяжении небольшого времени.

Команды
Это команды и другие приложения, которые могут запускаться пользователями в любой момент времени. Их требования к ресурсам предсказать невозможно.

Помимо перечисленных задач, существуют следующие категории запланированных заданий:

SysTools
Это процессы, выполняющие автоматизированные задачи. Такие задания не критичны для работы системы, но их необходимо периодически запускать в определенные интервалы времени.

SysBatch
Это редко запускаемые процессы, не критичные для работы системы, и не требующие обязательного завершения в определенный момент времени.

Начинать создание конфигурации следует с определения классов и правил. Ниже вы определите собственные классы с помощью перечисленных выше общих категорий. Для этого выполните следующую процедуру:

1. Создайте в каталоге `/etc/wlm` новую конфигурацию с именем `MyConfig`:
   ```
   mkdir /etc/wlm/MyConfig
   ```
2. Скопируйте файлы шаблонов в каталог `/etc/wlm/MyConfig`:
   ```
   cp -pr /etc/wlm/template/* /etc/wlm/MyConfig
   ```
3. Для создания суперклассов включите в файл `/etc/wlm/MyConfig/classes` следующие записи:

```
System:

Default:

DeptA:

DeptB:

SysTools:

SysBatch:
```

В качестве начальной точки вы определите для каждого отдела свой суперкласс (поскольку сервер будет совместно использоваться двумя отделами). Суперклассы SysTool и SysBatch будут обеспечивать обработку запланированных заданий, описанных выше. Суперклассы System и Default должны быть определены всегда.

4. В каталоге `MyConfig` создайте каталоги для суперклассов DeptA и DeptB. (При создании конфигурации необходимо создать каталоги для всех суперклассов, содержащих подклассы.) На следующем шаге вы определите пять подклассов (под одному для каждой категории заданий) для суперкласса каждого отдела.
5. Для создания подклассов каждой из категории заданий отредактируйте файлы `/etc/wlm/MyConfig/DeptA/classes` и `/etc/wlm/MyConfig/DeptB/classes`, включив в них следующие записи:

```
Listen:

Work:

Monitor:

Report:

Command:
```

Примечание: Содержимое файлов `classes` в разных суперклассах может быть различным.

После идентификации классов необходимо создать правила классификации, которые будут применяться для классификации процессов на уровне суперклассов и подклассов. Для простоты предположим, что расположение всех запускаемых приложений известно, и все процессы одного отдела выполняются в группе UNIX deptA, а все процессы второго отдела - в группе UNIX deptB.

6. Для создания правил суперклассов отредактируйте файл `/etc/wlm/MyConfig/rules`, включив в него следующие записи:

```
DeptA - - deptA - -
DeptB - - deptB - -
SysTools - root,bin - /usr/sbin/tools/* -
SysBatch - root,bin - /usr/sbin/batch/* -
System - root - - -
Default - - - - -
```

Примечание: Если может работать несколько экземпляров приложения, причем все атрибуты классификации, отличные от метки, совпадают, то для различия таких приложений и включения их в разные классы, воспользуйтесь командой **wlmassign** или процедурой **wlm_set_tag**.

7. Для задания более точных правил подклассов создайте файлы `/etc/wlm/MyConfig/DeptA/rules` и `/etc/wlm/MyConfig/DeptB/rules` со следующим содержимым:

```
Listen - - - /opt/myapp/bin/listen* -
Work - - - /opt/myapp/bin/work* -
Monitor - - - /opt/bin/myapp/bin/monitor -
Report - - - /opt/bin/myapp/report* -
Command - - - /opt/commands/* -
```

8. Для определения уровня использования ресурсов каждым классом запустите WLM в пассивном режиме с помощью следующей команды:

```
wlmcntrl -p -d MyConfig
```

После запуска WLM в пассивном режиме вы сможете сначала запустить каждое приложение в отдельности и получить более точную информацию о его требованиях к объему ресурсов. Затем вы можете запустить все приложения одновременно для определения взаимодействия между классами.

Другой способ определения требований приложений к объему ресурсов заключается в запуске WLM в пассивном режиме на каждом из объединяемых серверов. Недостаток такого подхода заключается в том, что вам придется заново создавать конфигурации в объединенной системе, а требуемая доля ресурсов новой системы скорей всего будет отличаться от значения, полученного в исходных системах.

Шаг 2. Определение уровней, общих ресурсов и ограничений

Конфигурация WLM реализует стратегию управления ресурсами. Запуск WLM в пассивном режиме позволяет получить информацию, которая поможет вам определить применимость выбранной стратегии к данной рабочей схеме. После этого вы можете определить уровни, общие ресурсы и ограничения, обеспечивающие управление рабочей схемой на основе выбранной стратегии управления ресурсами.

В данном примере предъявляются следующие требования:

- Класс System должен иметь наивысший приоритет и гарантированный доступ к необходимой ему доле системных ресурсов в любой момент времени.
- Класс SysTools должен иметь доступ к определенной доле ресурсов в любой момент времени, но его приоритет несколько ниже, чтобы не оказывать сильного влияния на работу приложений в DeptA и DeptB.
- Класс SysBatch не должен мешать выполнению любых других задач в системе.
- DeptA получит 60% общего объема ресурсов (имеются в виду общие ресурсы классов), а DeptB - 40%. В пределах DeptA и DeptB:
  - Процессы класса Listen должны отвечать на запросы с минимальной задержкой, но не должны потреблять большого объема ресурсов.
  - Процессы класса Work должны иметь доступ к большей части ресурсов. Классы Monitor и Command также должны иметь доступ к определенной части ресурсов, меньшей, чем для класса Work.
  - Класс Report не должен мешать выполнению любых других задач в системе.

Следующая процедура определяет уровни, общие ресурсы и ограничения:

1. Для создания уровней суперкласса включите в файл `/etc/wlm/MyConfig/classes` следующие записи:

```
System:

Default:

DeptA:
        localshm = yes
        adminuser = adminA
        authuser = adminA
        inheritance = yes

DeptB:
        localshm = yes
        adminuser = adminB
        authuser = adminB
        inheritance = yes

SysTools:
        localshm = yes

SysBatch:
        tier = 1
        localshm = yes
```

Суперкласс SysBatch помещен на уровень 1, поскольку он содержит низкоприоритетные задания, которые не должны мешать работе остальных заданий в системе. (Если уровень не задан, то по умолчанию применяется уровень 0.) Управление суперклассами каждого отдела определяется атрибутами adminuser и authuser. Для DeptA и DeptB задан атрибут inheritance (наследование). Все новые процессы, запускаемые в классе с наследованием, будут относиться к этому классу.

2. Для создания уровней подкласса для каждой группы заданий измените файлы `/etc/wlm/MyConfig/DeptA/classes` и `/etc/wlm/MyConfig/DeptB/classes`, включив в них следующие записи:

```
Listen:

Work:

Monitor:

Report:
        tier = 1
Command:
```

3. Для первоначального задания общих ресурсов для суперклассов укажите в файле `/etc/wlm/MyConfig/shares` следующие записи:

```
DeptA:
        CPU = 3
        memory = 3

DeptB:
        CPU      =   2
        memory = 2
```

Поскольку всего определено 5 общих ресурсов процессора, процессам DeptA будет доступно три из пяти ресурсов (или 60% от общего объема ресурсов процессора), а процессам DeptB - два из пяти ресурсов (или 40%). Поскольку вы не указали общие ресурсы для классов SysTools, System и Default, то их уровень использования ресурсов не будет зависеть от числа активных общих ресурсов, что обеспечит этим классам более высокий приоритет доступа к ресурсам, чем у DeptA и DeptB (до достижения ограничения). Для класса SysBatch общие ресурсы не заданы, поскольку это единственный суперкласс уровня 1 и задавать для него долю общих ресурсов не имеет смысла. Задания класса SysBatch могут получать только те ресурсы, которые не используются классами уровня 0.

4. Для первоначального задания общих ресурсов подклассов отредактируйте файлы `/etc/wlm/MyConfig/DeptA/shares` и `/etc/wlm/MyConfig/DeptB/shares`, включив в них следующую информацию:

```
Work:
        CPU = 5
        memory = 5

Monitor:
        CPU = 4
        memory = 1

Command:
        CPU = 1
        memory = 1
```

Поскольку вы не указали общие ресурсы для класса Listen, он будет иметь в суперклассе наиболее высокий приоритет доступа к запрашиваемым ресурсам. Наибольшая доля общих ресурсов выделена классу Work, поскольку у него самые высокие требования к объему ресурсов. Классам Monitor и Command доли общих ресурсов выделены в соответствии с особенностями и относительным уровнем важности этих классов. Для класса Report общие ресурсы не заданы, поскольку это единственный подкласс уровня 1 и задавать для него долю общих ресурсов не требуется. Задания класса Report могут получать только те ресурсы, которые не используются подклассами уровня 0.

На следующем шаге вы укажете ограничения для классов, для которых не задана доля общих ресурсов. (Вы также можете задать ограничения для классов с выделенной долей общих ресурсов. Дополнительные сведения приведены в разделе Управление ресурсами с помощью WLM).

5. Для первоначального задания ограничений для суперклассов укажите в файле /etc/wlm/MyConfig/limits следующие записи:

```
Default:
        CPU = 0%-10%;100%
        memory = 0%-10%;100%

SysTools:
        CPU = 0%-10%;100%
        memory = 0%-5%;100%

System:
        CPU = 5%-50%;100%
        memory = 5%-50%;100%
```

   Для классов System, SysTools и Default заданы мягкие ограничения сверху, чтобы эти классы не влияли существенно на работу других заданий системы. Для класса System заданы ограничения снизу на ресурсы процессора и памяти, поскольку этот класс включает процессы, критически важные для работы системы, и должен иметь гарантированный доступ к ресурсам.

6. Для задания ограничений подклассов отредактируйте файлы /etc/wlm/MyConfig/DeptA/limits и /etc/wlm/MyConfig/DeptB/limits, включив в них следующие записи:

```
Listen:
        CPU = 10%-30%;100%
        memory = 10%-20%;100%

Monitor:
        CPU = 0%-30%;100%
        memory = 0%-30%;100%
```

   Примечание: В каждом файле подкласса можно задать свои ограничения.

   Для классов Listen и Monitor заданы мягкие ограничения сверху, чтобы эти классы не влияли существенно на работу других подклассов того же суперкласса. В частности, система не должна продолжать прием запросов на запуск заданий класса Work, если у этого класса нет ресурсов для обработки запросов. Для класса Listen задано мягкое ограничение снизу, обеспечивающее небольшое время отклика. Ограничение снизу по объему памяти гарантирует, что используемые этим классом страницы памяти не будут удалены при замене страниц, что позволит ускорить обработку. Ограничение снизу на объем ресурсов процессора гарантирует, что во время работы эти процессы будут иметь максимальный приоритет доступа к ресурсам процессора в пределах своего суперкласса.

Шаг 3. Тонкая настройка конфигурации WLM

1. Контролируйте работу системы с помощью команды **wlmstat**, чтобы убедиться, что WLM правильно распределяет ресурсы, то есть некоторые процессы не получают слишком много ресурсов, тогда как объем ресурсов других процессов необоснованно ограничивается. Если ресурсы распределяются неправильно измените долю общих ресурсов, выделяемых различным процессам, и обновите WLM.
2. В процессе отслеживания и настройки долей, ограничений и приоритетов определите, следует ли передать права на управление подклассами другим администраторам. Затем администратор может собрать дополнительную информацию и настроить параметры по своему усмотрению.

Администратор каждого суперкласса может выполнить эту операцию для всех подклассов своего суперкласса. Единственное отличие заключается в том, что WLM не может работать в пассивном режиме на уровне подклассов. Настройку подклассов нужно выполнять в активном режиме WLM. Для того чтобы настройка не повлияла на работу пользователей и приложений, вначале укажите значения по умолчанию для параметров приоритета, ограничений (0% для минимума и 100% для обоих максимумов) и доли общих ресурсов ('-'). В этом случае WLM не будет управлять распределением ресурсов между подклассами.

См. дополнительную информацию

- Администратор рабочей схемы.
- Управление рабочей схемой.

- Диагностика управления рабочей схемой в книге *Руководство по настройке производительности*.
- Описания файлов `classes`, `limits`, `rules` и `shares` в книге *Справочник по файлам* .
- Команды **topas**, **wlmassign**, **wlmcheck**, **wlmcntrl** и **wlmstat**.
- Описания процедур WLM, в частности **wlm_set_tag**.

Понятия, связанные с данным:

“Настройка управления рабочей схемой” на стр. 518
Определения и атрибуты классов, долю общих ресурсов, ограничения и правила классификации можно задать с помощью SMIT или из командной строки. Все определения и правила хранятся в обычных текстовых файлах, которые можно создать и изменить с помощью текстового редактора.

Классы

WLM позволяет управлять распределением ресурсов системы путем определения классов обслуживания и присвоения ресурсов этим классам.

У каждого класса есть атрибуты, задающие его поведение и привилегии. Каждый процесс в системе принадлежит какому-либо классу обслуживания, что позволяет управлять выделением ресурсов этому процессу. Процессы распределяются по классам вручную или с помощью правил классификации.

WLM поддерживает классы двух уровней: *суперклассы* и *подклассы*. Суперклассам ресурсы выделяются исходя из объема доступных ресурсов в системе, а подклассам - исходя из объема ресурсов в суперклассе. Подклассы позволяют более точно управлять распределением ресурсов. Ответственность за создание подклассов можно передать специальному административному пользователю или административной группе суперкласса.

Для определения параметров подклассов и суперклассов можно использовать SMIT или командную строку. Приложения могут пользоваться API WLM. Все определения хранятся в текстовых файлах, называемых *файлами свойств* WLM.

Длина имени не должна превышать 16 символов; при этом имя может содержать прописные и строчные буквы английского алфавита, цифры и символы подчеркивания (_). В некоторых конфигурациях WLM имена суперклассов не должны повторяться. Имена подклассов одного суперкласса также не должны повторяться, однако это ограничение не распространяется на подклассы разных суперклассов. В этом случае для точного указания подкласса необходимо задать имя суперкласса и имя подкласса, разделенные точками, например: *Super.Sub*.

Суперкласс:

Системный администратор может определить не более 64 суперклассов.

Кроме того, автоматически создаются еще пять суперклассов:

Суперкласс *Default*
: Суперкласс по умолчанию. Он всегда определен в системе. Все процессы пользователя root, которым не был автоматически назначен суперкласс, заносятся в суперкласс Default. В суперкласс *Default* можно добавить и другие процессы, задав соответствующие правила классификации.

Суперкласс *System*
: В этот класс входят все привилегированные процессы (процессы пользователя root), для которых не заданы правила классификации. Кроме того, к этому суперклассу относятся страницы памяти из сегментов памяти ядра и нитей ядра. Этот суперкласс можно назначить другим процессам, задав соответствующие правила классификации. По умолчанию минимальный объем памяти этого суперкласса составляет 1%.

Суперкласс *Shared*
: Этому суперклассу принадлежат все страницы памяти, совместно используемые процессами из различных суперклассов. В их число входят страницы общих сегментов памяти и страницы файлов,

используемые процессами различных суперклассов. Общая память и файлы, применяемые различными процессами одного суперкласса, относятся к этому суперклассу. Страницы передаются суперклассу Shared только при обращении к ним процесса другого суперкласса. Этот суперкласс включает только общие страницы физической памяти и относящиеся к ней ограничения. Для него нельзя задать общие ресурсы и ограничения на использование ресурсов других типов. Кроме того, нельзя определить подклассы и правила классификации. Сегмент памяти, используемый совместно процессами из разных подклассов одного суперкласса, может быть помещен в подкласс *Shared* или оставлен в текущем подклассе, в зависимости от значения атрибута **localshm** исходного подкласса.

Суперкласс *Unclassified*

К этому суперклассу относятся страницы памяти процессов, которым не назначен ни один класс. При запуске WLM все существующие процессы классифицируются в соответствии с правилами классификации загруженной конфигурации WLM. При начальной классификации процессов все страницы памяти каждого процесса передаются либо суперклассу этого процесса (если они не используются совместно или используются процессами из одного подкласса), либо суперклассу *Shared*, если они используются процессами различных суперклассов.

Однако существует небольшое количество страниц, которые не могут быть связаны с определенным процессом во время начальной классификации. Эти страницы передаются суперклассу *Unclassified*. Большая часть из них позднее распределяется по другим классам при обращении к странице, изменении ее размера или ее освобождении каким-либо процессом. Суперкласс *Unclassified* не включает в себя процессы. Этот суперкласс содержит только общие страницы физической памяти и относящиеся к ней ограничения. Для него нельзя задать общие ресурсы и ограничения на использование ресурсов других типов. Кроме того, нельзя определить подклассы и правила классификации.

Суперкласс *Unmanaged*

Всегда определяется специальный суперкласс с именем *Неуправляемый*. Он не включает в себя процессы. К этому классу относятся все закрепленные страницы памяти в системе, не управляемые WLM. Процессорное время, затраченное ожидающими процессами waitproc, не принадлежит никакому классу во избежание 100% использования системы. Для него нельзя задать общие ресурсы и ограничения на использование ресурсов других типов. Кроме того, нельзя определить подклассы и правила классификации.

Подклассы:

Системный администратор или администратор суперкласса может определить до 61 подклассов каждого суперкласса.

Кроме того, всегда определены два специальных подкласса - *Default* (подкласс по умолчанию) и *Shared* (подкласс общих ресурсов).

Подкласс *Default*

Подкласс по умолчанию, который всегда определен в классе. К подклассу *Default* относятся все процессы суперкласса, которым не был автоматически назначен другой подкласс. Подкласс *Default* можно назначить и другим процессам, задав соответствующие правила классификации.

Подкласс *Shared*

К этому подклассу относятся все страницы памяти, применяемые процессами из различных подклассов суперкласса. В их число входят страницы из общих сегментов памяти и страницы файлов, применяемых процессами различных подклассов одного суперкласса. Общая память и файлы, открытые различными процессами одного подкласса, принадлежат этому подклассу. Страницы файла или общего сегмента памяти передаются подклассу *Shared*, только когда к ним обращается процесс из другого подкласса того же суперкласса. Подкласс *Shared* не включает в себя процессы. Этот подкласс может содержать только общие сегменты физической памяти и связанные с ними ограничения; общие ресурсы других типов, ограничения на использование ресурсов других типов, а также правила классификации для этого подкласса задать нельзя. Сегмент памяти,

используемый совместно процессами из разных подклассов одного суперкласса, может быть помещен в подкласс *Shared* или оставлен в текущем подклассе, в зависимости от значения атрибута **localshm** исходного подкласса.

Атрибуты классов:

Перечень всех атрибутов класса WLM.

Имя класса
Строка, длина которой не превышает 16 символов, состоящая только из строчных и прописных букв английского алфавита, цифр и символов подчеркивания (_).

Приоритет
Число от 0 до 9, задающее приоритет доступа к ресурсам по сравнению с другими классами.

Наследование
Указывает, наследуют ли дочерние процессы класс родительского процесса.

localshm
Позволяет запретить перемещение страниц этого класса в общую память.

Администратор (adminuser, admingroup, authgroup) (только для суперклассов)
Позволяет передать права на управление суперклассом.

Права доступа (authuser, authgroup)
Указывает, каким пользователям разрешено вручную назначать данный класс процессам.

Набор ресурсов (rset)
Задает набор ресурсов CPU, доступных классу (набор процессоров).

delshm
Удаляет сегменты общей памяти если последний ссылающийся на них процесс был убит из-за превышения ограничения виртуальной памяти.

vmeforce
Указывает, следует ли убить все процессы в классе или только избранные процессы при достижении классом ограничения виртуальной памяти.

io_priority
Приоритет, присвоенный запросам ввода-вывода, которые были отправлены нитями, отнесенными к этому классу. Этот приоритет используется для определения важности буферов ввода-вывода на уровне устройства. Если запоминающее устройство не поддерживает приоритеты ввода-вывода, приоритет будет игнорирован. Допустимые приоритеты ввода-вывода лежат в диапазоне от 0 до 15.

Понятия, связанные с данным:

“Классификация процессов для управления рабочей схемой” на стр. 486
Классификация процессов выполняется в соответствии с правилами, заданными системным администратором. Критерием классификации является набор атрибутов процесса, таких как идентификатор пользователя и группы, имя исполняемого файла, тип процесса и тег приложения.

Атрибут приоритета:

Приоритеты позволяют задать порядок выделения ресурсов классам WLM.

Классы могут иметь приоритет от 0 до 9, где 0 - самый высокий. Классу с приоритетом 0 доступны все ресурсы системы. Классам с более низким приоритетами доступны все ресурсы, не занятые классами с более высоким приоритетом. Совокупное потребление ресурсов зависит от числа активных общих ресурсов и объема ресурсов, выделенного для данного приоритета. Поскольку все системные ресурсы доступны только для приоритета 0, именно этот приоритет рекомендуется присваивать классам с жизненно важными для системы процессами. Классы, приоритет которых не задан, считаются классами с приоритетом 0.

Приоритеты действуют как на уровне суперклассов, так и на уровне подклассов. От приоритетов суперклассов зависит распределение ресурсов между суперклассами. От приоритетов подклассов зависит распределение ресурсов, полученных суперклассом, между его подклассами. Приоритеты подклассов разных суперклассов не связаны между собой.

Атрибут наследования:

Атрибут **inheritance** указывает, должны ли процессы автоматически изменять класс при изменении атрибутов классификации процессов.

При создании нового процесса процедурой **fork** он всегда наследует класс родительского процесса, вне зависимости от значения этого атрибута. Единственным исключением является ситуация, когда у родительного процесса есть тег, тег **наследовать теги при выполнении fork** выключен, и наследование классов для родительского класса также выключено. В этом случае класс дочернего процесса присваивается на основании правил классификации.

Если наследование класса выключено, то класс присваивается процессу автоматически на основании правил классификации, после вызова всех процедур, изменяющих атрибуты, которые используются правилами. Обычно это процедуры **exec** однако классификацию могут изменять и другие процедуры: **setuid**, **setgid**, **plock**, **setpri** и **wlm_set_tag** . Если наследование включено, то процесс остается в текущем классе. Для процессов из классов с включенным наследованием может применяться классификация вручную.

Допустимые значения атрибута **inheritance** - да и нет. Если значение не указано, наследование выключено.

Этот атрибут может быть задан как для суперкласса, так и для подкласса. Для подкласса заданного суперкласса:

- Если атрибут **inheritance** установлен как на уровне суперкласса, так и на уровне подкласса, дочернему процессу будет присвоен подкласс родительского процесса.
- Если атрибут **inheritance** установлен для суперкласса и не установлен для подкласса, дочернему процессу будет присвоен один из подклассов суперкласса родительского процесса в соответствии с правилами автоматической классификации.
- Если атрибут **inheritance** не установлен для суперкласса и установлен для подкласса, то суперкласс дочернего процесса будет выбран в соответствии с правилами автоматической классификации.
  - Если в результате дочерний процесс окажется в том же классе, что и родительский процесс, ему будет назначен подкласс родительского процесса.
  - Если же суперкласс дочернего процесса будет отличаться от суперкласса родительского процесса, подкласс дочернего процесса будет определен в соответствии с правилами автоматической классификации.
- Если атрибут **inheritance** не установлен ни для суперкласса, ни для подкласса, суперкласс и подкласс дочернего процесса будут определены в соответствии с правилами автоматической классификации.

Атрибут localshm:

Атрибут **localshm** может быть задан как для суперкласса, так и для подкласса.

Атрибут **localshm** применяется для предотвращения переноса сегментов памяти, выделенных классу, в подкласс или суперкласс *Shared* при обращении к ним процессов из других классов. Допустимые значения этого атрибута: yes и no. Если указано значение `yes`, то сегменты общей памяти в этом классе останутся локальными и не будут перенесены в соответствующий класс *Shared*. По умолчанию применяется значение `no`.

Сегменты памяти классифицируются на основании числа станичных ошибок. При создании сегмент помечается как принадлежащий суперклассу *Unclassified*. После первой страничной ошибки сегмент переносится в класс того процесса, в котором возникла ошибка. Если позднее страничная ошибка возникнет при работе с этим сегментом процесса другого класса, то WLM решает, нужно ли переносить сегмент в

соответствующий подкласс или суперкласс *Shared*. Если сегмент памяти и процесс, в котором возникла страничная ошибка, принадлежат разным суперклассам, то выполняется одно из следующих действий:

- Если атрибут **localshm** суперкласса сегмента равен `yes`, то сегмент остается в текущем суперклассе. Если атрибут **localshm** подкласса сегмента равен `yes`, то сегмент остается в текущем подклассе. Если в суперклассе атрибут **localshm** равен `yes`, а в подклассе - `no`, то сегмент переносится в подкласс *Shared* этого суперкласса.
- Если атрибут **localshm** суперкласса сегмента равен `no`, то сегмент переносится в суперкласс *Shared*. Такая операция выполняется по умолчанию.

Если процесс и сегмент принадлежат разным подклассам одного суперкласса, и атрибут **localshm** подкласса сегмента равен `yes`, то сегмент остается в текущем подклассе или суперклассе. В противном случае сегмент перемещается в подкласс *Shared* суперкласса.

Если сегмент памяти и процесс, вызвавший страничную ошибку, находятся в одном классе (суперклассе и подклассе), то никакие действия не выполняются.

Атрибут администратора:

Атрибуты **adminuser** и **admingroup** предоставляют отдельному пользователю или группе пользователей права на управление суперклассом.

Примечание: Эти атрибуты действительны только для суперклассов.

Атрибут **adminuser** задает имя пользователя (из указанных в файле `/etc/passwd`), которому предоставлены права на управление суперклассом. Атрибут **admingroup** указывает имя группы пользователей (из указанных в файле `/etc/group`), которым предоставляются права на управление суперклассом.

Каждому из атрибутов может быть присвоено только одно значение. Атрибуты могут задаваться в любой комбинации. У указанного пользователя или группы пользователей будут права на выполнение следующих операций:

- Создание и удаление подклассов.
- Изменение атрибутов подклассов, а также режима общего доступа к ресурсам и ограничений на использование ресурсов.
- Задание, удаление и изменение правил автоматической классификации.
- Обновление текущей конфигурации WLM для суперкласса.

Атрибут проверки прав доступа:

Атрибуты **authuser** и **authgroup** можно задать в любом классе. Они задают пользователя и группу пользователей, у которых есть права на назначение класса (суперкласса или подкласса) процессам вручную.

Когда процессу (или группе процессов) вручную назначается суперкласс, подкласс для этого процесса (или группы) выбирается в соответствии с правилами классификации, заданными для суперкласса.

Каждому из атрибутов может быть присвоено только одно значение. Атрибуты могут задаваться в любой комбинации.

Атрибут набора ресурсов:

Атрибут набора ресурсов (*rset*) можно задать в любом классе. Он задает имя набора ресурсов, определенного системным администратором.

Атрибут *rset* определяет подмножество ресурсов CPU системы. Значение атрибута по умолчанию, "system", предоставляет классу доступ ко всем ресурсам CPU системы. Если для подкласса указано *rset*, то набор CPU в наборе должен быть подмножеством CPU, доступных для суперкласса. (Подробные сведения приведены в описании команды **mkrset**.

Примечание: При присвоении наборов ресурсов классам с приоритетом 0 нужно соблюдать особую осторожность. Поскольку классам с более низкими приоритетами доступно только некоторое подмножество CPU, то при высокой загрузке этих CPU может возникнуть нехватка процессорного времени.

Классификация процессов в WLM

В WLM предусмотрено два способа классификации процессов.

- Класс может присваиваться автоматически на основании правил классификации при изменении атрибута классификации. Автоматическая классификация выполняется всегда, когда активен WLM (ее нельзя выключить). Это наиболее распространенный способ классификации процессов.
- Пользователь может вручную назначить класс для процесса или группы процессов при условии, что у него есть права доступа к этим процессам и целевому классу. Пользователь может выполнить классификацию вручную с помощью команды WLM, вызванной из командной строки или интерфейса SMIT, а приложение - с помощью соответствующего API WLM. Класс, назначенный вручную, переопределяет класс, назначенный автоматически или при наследовании.

Автоматическая привязка классов в WLM:

Для автоматической классификации процессов применяется набор правил классификации, заданных администратором WLM.

Можно задать два уровня правил:

- Правила классификации, заданные на уровне конфигурации WLM, применяются для определения суперкласса процесса.
- Правила классификации, заданные в суперклассе, применяются для определения подкласса процесса.

Эти правила основаны на значениях атрибутов процесса. К ним относятся:

- Идентификатор пользователя процесса
- Идентификатор группы процесса
- Полное имя исполняемого файла
- Тип процесса (например, `32bit` или `64bit`)
- Тег процесса.

Тег - это атрибут процесса, который представляет собой строку символов, заданную приложением с помощью API WLM.

Процесс классификации состоит в последовательном сравнении значений атрибутов из правил классификации с атрибутами процесса до нахождения правила, содержащего текущее значение атрибута процесса. Список правил классификации хранится в файле с именем `rules`.

Правило присвоения класса представляет собой текстовую строку со следующими полями, разделенными пробелами:

Элемент	Описание
Имя	Это поле содержит имя класса, определенного в файле классов (`rules`) соответствующего уровня (суперкласса или подкласса). Имена файлов могут содержать не более 16 символов. Допустимы только прописные и строчные буквы английского алфавита, цифры и символы подчеркивания. Правила классификации для системных классов Unclassified, Unmanaged и Shared определить нельзя.
Зарезервировано	Это поле зарезервировано для будущих версий. В этом поле должен быть указан дефис (-); его пропускать нельзя.
Пользователь	Это поле может содержать дефис (-) или хотя бы одно имя пользователя (определенное в файле `/etc/passwd`). Элементы списка разделяются запятыми (,). Если перед именем указать восклицательный знак (!), то соответствующий пользователь будет исключен из класса. Можно также указывать шаблоны имен пользователей в формате шаблонов оболочки Korn. Если допустимых имен пользователей нет, то правило игнорируется.
Группа	Это поле может содержать дефис (-) или хотя бы одно имя группы (определенное в файле `/etc/group`). Элементы списка разделяются запятыми (,). Если перед именем указать восклицательный знак (!), то соответствующая группа будет исключена из класса. Можно также указывать шаблоны имен групп в формате шаблонов оболочки Korn. Если допустимых имен групп нет, то правило игнорируется.
Приложение	Это поле может содержать дефис (-) или список имен исполняемых файлов. Все процессы приложений, указанных в списке, будут относится к описываемому классу. В качестве имени можно указать полное имя исполняемого файла или шаблон имени в формате оболочки Korn. Элементы списка разделяются запятыми (,). Если перед именем указать восклицательный знак (!), то соответствующее приложение будет исключено из класса. Если при загрузке не найдено ни одно приложение, то правило игнорируется. Если в последствии в результате монтирования файловой системы такие приложения появятся, то правило будет применяться.
Тип	Это поле может содержать дефис (-) или список атрибутов процесса. Для этих атрибутов допустимы следующие значения: • **32bit**: 32-разрядный процесс • **64bit**: 64-разрядный процесс • **plock**: процесс вызвал системную функцию **plock** для закрепления памяти • **fixed**: приоритет процесса фиксирован (SCHED_FIFO или SCHED_RR) Тип **fixed** предназначен только для целей классификации. WLM не управляет использованием процессора для обработки нитей и процессов с фиксированным приоритетом. Этот атрибут классификации позволяет изолировать процессы с фиксированным приоритетом, поскольку они могут вызвать ошибки других процессов класса. Кроме того, с помощью этого атрибута можно получить информацию об обработке таких процессов. В поле **type** может быть указано несколько перечисленных выше атрибутов, соединенных знаком плюс (+). Атрибуты **32bit** и **64bit** нельзя указывать одновременно.
Тег	Это поле может содержать дефис (-) или список тегов приложения. Тег приложения - это строка длиной до 30 символов. Элементы списка разделяются запятыми (,).

Атрибуты Пользователь, Группа, Приложение и Тег могут применяться для групп.

При создании процесса вызовом fork ему присваивается класс родительского процесса. Процесс может быть перемещен в другой класс, когда он изменит один из своих атрибутов, применяемых для классификации, посредством системного вызова, например, *exec*, *setuid* (и связанные вызовы), *setgid* (и связанные вызовы), *setpri* и *plock*.

Для классификации нового процесса WLM проверяет файл `rules` с правилами верхнего уровня для определения его суперкласса. WLM последовательно просматривает правила из этого файла и сравнивает указанные в них значения со значениями атрибутов процесса. Правила просматриваются в том порядке, в котором они указаны в файле. Если все значения, указанные в правиле, совпадают со значениями атрибутов процесса, то процессу назначается суперкласс, указанный в первом поле правила. После этого аналогичным образом просматривается файл правил суперкласса и определяется подкласс, к которому относится процесс.

Процесс относится к классу, описанному в правиле, только в том случае, если все значения его атрибутов соответствуют значениям полей правила. Ниже приведен список критериев, по которым определяется, соответствует ли атрибут значению поля из файла `rules`:

- Если в поле файла правил указан дефис (-), ему соответствует любое значение атрибута процесса.

- Для всех атрибутов, кроме атрибута *тип*, соответствием считается совпадение атрибута процесса с любым из значений списка, перед которыми не стоит восклицательный знак (!).
- Если в поле *тип* указано несколько атрибутов процесса, разделенных знаками (+), то соответствием считается наличие всех указанных атрибутов.

Как на уровне суперкласса, так и на уровне подкласса WLM просматривает правила в том порядке, в котором они перечислены в файле `rules`, и присваивает процессу класс первого подходящего правила. В связи с этим порядок правил в файле крайне важен для правильной классификации. Его необходимо учитывать при создании и изменении правил.

Обычная привязка классов в WLM:

Процессу или группе процессов можно вручную назначить определенный суперкласс или подкласс с помощью SMIT или команды **wlmassign**.

Дополнительная информация приведена в описании команды **wlmassign**. Приложения могут назначать классы процессам с помощью API **wlm_assign**.

Для назначения класса процессу или отмены этого назначения пользователь должен обладать определенными права доступа. Классификация вручную может быть выполнена или отменена независимо на уровне суперкласса и на уровне подкласса. Способ классификации задается в флагах функции API или параметрах команды, вызываемой инструментами управления WLM. Таким образом, процессу может быть назначен только суперкласс, только подкласс, или одновременно и суперкласс, и подкласс этого суперкласса. В последнем случае операция может быть выполнена как за один вызов команды или функции API, так и за два вызова, сделанных различными пользователями.

Эта гибкая схема, однако она может показаться чрезмерно сложной. Ниже приведены два поясняющих примера.

Пример 1: первичное присваивание процессов

Системный администратор вручную изменяет суперкласс процесса *Process1* с *superclassA* на *superclassB* (назначение только суперкласса). Подкласс суперкласса *superclassB* будет определен WLM согласно правилам автоматической классификации. Процесс *Process1* будет отнесен к классу *superclassB.subclassA* и помечен как имеющий суперкласс, назначенный вручную.

Пользователь с соответствующими правами доступа изменяет класс процесса *Process2* с *superclassA.subclassA* на *superclassA.subclassB* (то есть изменяет подкласс, к которому относится процесс). Процессу *Process2* будет назначен новый подкласс, после чего он будет помечен как имеющий подкласс.

Администратор суперкласса *superclassB* может вручную изменить класс процесса *Process1* на *subclassC*, который является одним из подклассов суперкласса *superclassB*. Процесс *Process1* будет классифицирован как относящийся к классу *superclassB.subclassC*, после чего он будет помечен как имеющий суперкласс и подкласс.

Пример 2: Изменение или отмена классификации процесса вручную

Изменение или отмену классификации процесса проще выполнять на уровне подкласса.

Допустим, системный администратор хочет увеличить объем ресурсов процесса *Process2*, для чего он вручную назначает процессу *Process2* суперкласс *superclassC*. В примере 1 процессу *Process2* был вручную назначен подкласс *subclassB* суперкласса *superclassA*, в результате чего он был помечен как имеющий подкласс. После назначения процессу *Process2* другого суперкласса предыдущее назначение теряет смысл и отменяется. В результате, при отсутствии наследования, процессу *Process2* будет вручную назначен суперкласс *superclassC* и автоматически (в соответствии с правилами классификации) - подкласс этого суперкласса.

Теперь предположим, что системному администратору потребовалось отменить назначение процессу *Process1* суперкласса *superclassB*. В результате, при отсутствии наследования, будет

отменена операция присвоения суперкласса вручную, а суперкласс процесса *Process1* будет выбран в соответствии с правилами автоматической классификации.

Если правила не изменились, процессу *Process1* будет назначен суперкласс *superclassA*, а его связь с подклассом *superclassB.subclassC* будет отменена.

Если бы в соответствии с правилами автоматической классификации процесс *Process1* был отнесен к суперклассу *superclassB*, то назначение ему подкласса *superclassB.subclassC* осталось бы в силе. В результате *Process1* был бы помечен, как имеющий подкласс, назначенный вручную.

Обновления WLM:

При обновлении WLM (командой **wlmcntrl -u**) может быть загружен новый набор правил классификации.

В этом случае, процессы часто переклассифицируются на основании новых правил. WLM не изменяет классификацию процессов, выполненную вручную или полученную в результате наследования, за исключением случаев, когда класс не существует в новой конфигурации.

Замечания о безопасности для WLM:

Для того чтобы назначить класс процессу или отменить такое назначение, у пользователя должны быть права доступа к процессу и целевому классу.

Из этого условия можно сделать следующие выводы:

- Пользователь root может назначить любой класс любому процессу.
- Администратор подкласса заданного суперкласса (пользователь, имя или группа которого указаны в атрибуте **adminuser** или **admingroup** суперкласса) может вручную изменить подкласс процесса на другой подкласс, относящийся к тому же суперклассу.
- Пользователь может назначить своему процессу (процессу, связанному с действующим ИД этого пользователя) подкласс, если у него есть права на назначение этого подкласса вручную (то есть если имя пользователя или его группа указаны в атрибуте **authuser** или **authgroup** суперкласса или подкласса).

Для изменения класса, назначенного вручную, или отмены этого назначения у пользователя должны быть по крайней мере те же права доступа, что и у пользователя, назначившего класс.

Управление ресурсами с помощью WLM

WLM отслеживает и управляет распределением ресурсов между всеми нитями и процессами, работающими в системе, путем разделения их на классы. Для каждого класса WLM можно задать минимальное и максимальное ограничение на использование ресурсов различных типов, а также целевой объем ресурсов.

Целевой объем ресурсов представляет собой оптимальный объем ресурсов для заданий данного класса. Доля общих ресурсов и ограничения на использование ресурсов на уровне суперклассов вычисляются относительно объема всех ресурсов системы. На уровне подклассов эти значения вычисляются относительно того объема ресурсов, который был предоставлен суперклассу. Иерархия классов позволяет разделить ресурсы системы между группами пользователей (суперклассами) и предоставить управление дальнейшим распределением администраторам этих суперклассов. Для распределения ресурсов внутри суперкласса его администратор может создавать и настраивать подклассы.

Типы ресурсов в WLM:

WLM управляет процентным потреблением ресурсов трех типов.

Элемент	Описание
Время CPU, затраченное нитями класса	Сумма всех тактов CPU, затраченных на выполнение нитей класса.
Физическая память, выделенная процессам класса	Общее число страниц памяти, принадлежащих процессам класса.
Интенсивность дискового ввода-вывода, выполняемого заданиями класса	Интенсивность обмена данными с дисками (число блоков размером 512 байт в секунду), выполняемого нитями класса.

Каждую секунду WLM подсчитывает долю ресурса, использованную классом.

- Общее количество процессорного времени, доступное в системе каждую секунду, равно число CPU в системе, умноженному на 1 секунду. Например, если за последнюю секунду в 8-процессорной системе на выполнение всех нитей класса было затрачено 2 секунды времени CPU, этот класс получил 2/8 = 25% от полного объема ресурсов CPU. Для управления распределением ресурсов WLM применяет затухающее среднее значение, подсчитанное за несколько секунд.
- Общий объем физической памяти, доступной процессам системы, равен общему число страниц памяти в системе за вычетом числа закрепленных страниц. Закрепленные страницы не распределяются WLM, поскольку их невозможно отнять у одного класса и передать другому. Доля памяти, занятая классом, вычисляется как отношение числа незакрепленных страниц памяти, принадлежащих всем процессам класса, к общему числу незакрепленных страниц памяти в системе.
- Основная проблема управления дисковым вводом-выводом состоит в определении пропускной способности устройства. Если диск занят на 100%, то число блоков, считываемых и записываемых в секунду, будет сильно зависеть от способа доступа к диску. Интенсивность последовательной записи данных на диск одним приложением будет значительно отличаться от интенсивности ввода-вывода в случае произвольного доступа к диску нескольких приложений. Если за пропускную способность устройства принять максимальную интенсивность последовательного ввода-вывода, то при оценке загруженности устройства в случае выполнения ввода-вывода с произвольным доступом может получиться значение 20%, хотя на самом деле устройство загружено на 100%.

Для получения более точной оценки использования диска различными классами WLM применяет информацию о загруженности диска за последнюю секунду, предоставляемую драйверами дисков (ее можно просмотреть с помощью команды **iostat**). WLM подсчитывает общее число блоков, прочитанных или записанных на диск за последнюю секунду всеми классами и каждым классом в отдельности, и получает от драйвера оценку загруженности устройства. Затем WLM пропорционально делит последнюю величину между классами.

Например, если за последнюю секунду было записано (или прочитано) 1000 блоков, и устройство было загружено на 70%, то класс, записавший 100 блоков, занимает 7% пропускной способности устройства. Как и в случае процессорного времени, для управления дисковым вводом-выводом WLM применяет затухающее среднее значение, подсчитанное за несколько секунд.

Доля общих ресурсов и ограничения на использование ресурсов независимо подсчитываются для каждого диска, с которым работали нити класса. WLM регулирует интенсивность дискового ввода-вывода для каждого диска в отдельности. Это означает, что если ограничение на число операций ввода-вывода, выполняемых процессами класса, было достигнуто для некоторого диска, то будет ограничен обмен данными только с этим диском - другие устройства не будут затронуты.

WLM поддерживает учет и управление ресурсами на основе общего потребления. Таким образом можно управлять двумя типами значений: общими значениями для класса и для процесса.

общие значения для класса

Ограничения на число процессов, нитей и сеансов можно задать для каждого класса. Эти значения указывают абсолютный объем соответствующего ресурса, допустимый для класса в каждый момент времени. Ограничения строго соблюдаются; если ограничение достигнуто, то новые экземпляры этого ресурса создать не удастся. Это будет продолжаться до тех пор, пока фактические значения не станут меньше ограничений.

общие значения для процесса

Для каждого процесса можно задать ограничения на использование процессора, блоков дискового ввода-вывода и время подключения сеанса. Эти ограничения указываются на уровне классов, но применяются для каждого процесса в классе (указанное значение может потребить каждый процесс). Полученные значения отражают общий объем каждого конкретного ресурса, потребленного процессом за время его существования. Процесс, превысивший ограничение, прерывается. Процессу отправляется сигнал SIGTERM, и если через 5 секунд процесс не будет завершен, ему будет отправлен сигнал SIGKILL. При достижении сеансом 90% времени подключения, на управляющий терминал отправляется предупреждающее сообщение.

Целевые доли в WLM:

Целевое (или желаемое) процентное значение потребления ресурса классом определяется числом выделенных ему долей.

Доля определяет относительный объем выделенного классу ресурса по сравнению с другими классами того же приоритета. Фактически это число выделенных классу долей, поделенное на общее число долей для данного приоритета. Если применяются ограничения, то целевое значение будет также ограничено некоторым диапазоном [минимум, гибкий максимум]. Если вычисленное значение окажется за пределами этого диапазона, то оно устанавливается равным соответствующей верхней или нижней границе. Число активных долей - это общее число долей во всех классах, в которых есть хотя бы один активный процесс. Поскольку число активных классов - динамическая величина, то и целевое значение также является динамическим. Если для данного приоритета есть только один активный класс, то целевое значение будет равно 100% общего объема ресурсов, выделенного для данного приоритета.

Допустим, что есть три активных суперкласса с приоритетом 0—A, B и C с долями 15, 10 и 5, соответственно. Тогда целевыми значениями будут:

target(A) = 15/30 = 50%
target(B) = 10/30 = 33%
target(C) = 5/30 = 17%

Если через некоторое время класс B станет неактивным (в нем не будет активных процессов), то целевые значения для классов A и C автоматически изменятся:

target(A) = 15/20 = 75%
target(C) = 5/20 = 25%

Таким образом, доли представляют собой саморегулирующийся инструмент настройки процентного использования ресурсов, выделенных классу или переданных из других классов при их завершении.

Общее число долей для класса может быть любым числом от 1 до 65535. Доли можно указать как для суперклассов, так и для подклассов. В первом случае они представляют относительный объем ресурса среди всех активных суперклассов с заданным приоритетом. Во втором случае они представляют относительный объем ресурса среди всех активных подклассов с заданным приоритетом в суперклассе. Доли, заданные для подклассов разных суперклассов, не связаны между собой.

В некоторых случаях, целевое значение для класса должно быть независимым от числа активных долей. В этом случае в качестве числа долей необходимо указать "-". Объем потребления данного ресурса в этом классе регулироваться не будет. Целевое значение будет равно общему объему ресурсов, выделенному для данного приоритета, минус сумма минимальных значений для других классов с данным приоритетом. Полученное целевое или фактическое значение (наименьшее из них) вычитается из объема ресурсов, доступных другим классам с данным приоритетом.

Предположим, что классам A, B, C и D присвоены доли "-", 200, 150 и 100, соответственно. Все классы активны, и класс A потребляет 50% ресурса:

target(A) = не регулируется = 100%
target(B) = 200/450 * доступно = 44% * 50% = 22%
target(C) = 150/450 * доступно = 33% * 50% = 17%
target(D) = 100/450 * доступно = 22% * 50% = 11%

Поскольку класс A не регулируется и потребляет 50 процентов ресурсов, то доли потребления ресурсов остальными классами вычисляются исходя из оставшихся 50 процентов. Таким образом, класс A всегда будет иметь больший приоритет, чем остальные классы (см. “Приоритет классов в WLM” на стр. 516).

Примечание: Нерегулируемый класс - это не то же самое, что класс с более высоким приоритетом. Следующее поведение описывает нерегулируемые классы с одним приоритетом, но не классы с разными приоритетами:

- Поскольку регулируется потребление конкретного ресурса, класс может быть регулируемым для одного типа ресурсов, и нерегулируемым - для другого.
- Учитываются минимальные ограничения для других классов с тем же приоритетом. Классы с более высоким приоритетом не учитывают минимальные ограничения, заданные для классов с более низким приоритетом.
- Даже при отсутствии минимальных ограничений для других классов с тем же приоритетом, потребление нерегулируемого ресурса зависит от поведения этих классов, поскольку классы конкурируют за другие ресурсы. При задании значений в каждой конкретной ситуации необходимо тщательное тестирование.

Если число долей не указано, то будет применяться значение по умолчанию "-". Обратите внимание, что в первой версии WLM значение по умолчанию было равно 1.

Для каждого класса указывается доля общих ресурсов различных типов. Эти значения задаются в разделах файла **shares**. Например:

```
доли
имя-класса:
   CPU      =   2
   memory   =   4
   diskIO   =   3
```

Спецификация ограничения ресурса WLM:

Помимо указания долей для определения относительного потребления ресурсов, WLM позволяет задавать ограничения на ресурсы для класса. С их помощью администратор может более точно управлять выделением ресурсов. Эти значения указываются в процентах относительно общего объема ресурсов, доступного приоритету класса.

Допустимы три типа процентных ограничений:

Минимум
Это минимальный объем ресурса, доступный классу. Если фактическое потребление меньше этого значения, то классу будет предоставлен больший приоритет при обращении к ресурсу. Допустимы значения от 0 до 100, где 0 - значение по умолчанию.

Гибкий максимум
Это максимальный объем ресурса, который может потребить класс при наличии конкуренции за этот ресурс. Если фактическое потребление больше этого значения, то классу будет предоставлен меньший приоритет при обращении к ресурсу. Если среди классов с тем же приоритетом нет конкуренции за ресурс, то класс сможет потребить любой объем ресурса. Допустимы значения от 1 до 100, где 100 - значение по умолчанию.

Жесткий максимум
Это максимальный объем ресурса, который может потребить класс, даже при отсутствии конкуренции. Если это ограничение достигнуто, то дополнительный объем ресурса потребить не удастся. Допустимы значения от 1 до 100, где 100 - значение по умолчанию.

Ограничения на объем выделяемых ресурсов задаются в специальном файле, каждый раздел которого соответствует определенному классу. Вначале указывается гибкий минимум, а затем, через дефис (-), гибкий максимум. Жесткий максимум можно указать после гибкого максимума через точку с запятой (;). После каждого значения следует символ процента (%).

Ниже приведены примеры применения файлов правил:

- Если пользователь joe из группы acct3 запустит `/bin/vi`, процесс попадет в суперкласс acctg.
- Если пользователь sue из группы dev запустит `/bin/emacs`, процесс попадет в суперкласс devlt (по идентификатору группы), но не будет помещен в подкласс editors, поскольку пользователь sue исключен из этого класса. В результате процессу будет присвоен класс devlt. Default.
- Непосредственно после запуска программы `/usr/sbin/oracle` с идентификатором пользователя `oracle` и идентификатором группы `dbm` для обслуживания базы данных `DB1` новый процесс будет помещен в суперкласс default. Только после того, как процесс изменит свой тег на _DB1, ему будет присвоен суперкласс db1.

Для каждого класса в файле `limits` создается раздел, в котором перечислены ограничения на ресурсы всех типов. Пример:

```
доли
имя-класса:
   CPU     =   0%-50%;80%
   memory  =   10%-30%;50%
```

В этом примере ограничения на дисковый ввод-вывод не заданы, поэтому будут применяться следующие значения по умолчанию:

```
diskIO   =   0%-100%;100%
```

В приведенных выше примерах предполагается, что ни для одного класса не установлен атрибут наследования. В противном случае новые процессы относились бы к тем же классам, что и их родительские процессы.

При выборе значений ограничений должны соблюдаться следующие правила:

- Минимум не должен превышать максимум.
- Гибкий максимум не должен превышать жесткий максимум.
- Сумма минимальных значений всех суперклассов с одинаковым приоритетом не должна превышать 100.
- Сумма минимальных значений всех подклассов одного суперкласса не должна превышать 100.

Если класс достигает жесткого максимума и пытается получить доступ к дополнительным страницам памяти, запускается Алгоритм замены страниц VMM (LRU), который отбирает страницы у класса перед предоставлением новых, обеспечивая выполнение указанного жесткого ограничения. Недостатком такого алгоритма является то, что даже при наличии большого числа свободных страниц в системе будет выполняться подкачка, которая приводит к снижению производительности системы. Перед указанием жесткого максимума для какого-либо класса рекомендуется задать значения минимального объема памяти для других классов.

Классы с минимальным потреблением имеют самый высокий приоритет, поэтому сумма минимальных значений не должна быть очень большой. Обязательно учитывайте требования других классов.

Условие о том, что сумма минимумов для классов с одинаковым приоритетом не должна превышать 100, означает, что класс с максимальным приоритетом всегда получает хотя бы минимальный объем ресурсов. WLM не гарантирует, что минимум действительно будет достигнут. Это зависит от поведения других процессов и других ограничений. Например, класс может не достичь заданного минимального уровня использования CPU из-за отсутствия необходимого объема памяти.

Определение минимального объема физической памяти в некоторой степени защищает страницы процессов класса (по крайней мере процессов с максимальным приоритетом). Если число страниц класса меньше минимального значения, то эти страницы могут быть отобраны только в том случае, если всем остальным активным классам также не выделено минимальное число страниц, и одному из классов требуются дополнительные страницы. У класса с максимальным приоритетом страницы никогда не отбираются, если их число меньше минимального. Например, определение минимального объема физической памяти для класса интерактивных заданий гарантирует, что у этого класса не будут отобраны все страницы памяти между последовательными запросами пользователей (даже если памяти недостаточно). Это позволяет сократить время ответа системы.

Внимание: Неправильное использование жестких максимумов может существенно понизить производительность системы и программ. Жесткие ограничения могут приводить к появлению неиспользуемых ресурсов, поэтому в большинстве ситуаций более применимы гибкие ограничения. Например, жесткие максимумы можно применять для ограничения потребления ресурсов классами с более высоким приоритетом, чтобы обеспечить ресурсы для более низкого, хотя лучшим решением может быть присвоение программе более высокого приоритета.

Общие ограничения можно указывать в файле ограничений в следующем формате:

Таблица 69. Ограничения ресурсов для WLM

Ресурс	Единицы измерения	Единица измерения по умолчанию	Максимум	Минимум
totalCPU	s, m, h, d, w	s	$2^{30} – 1$s	10 s
totalDiskIO	KB, MB, TB, PB, EB	KB	$(2^{63} – 1)$ * 512/1024 KB	1 MB
totalConnectTime	s, m, h, d, w	s	$2^{63} – 1$ s	5 m
totalProcesses	–	–	$2^{63} – 1$	2
totalThreads	–	–	$2^{63} – 1$	2
totalLogins	–	–	$2^{63} – 1$	1

Примечание: Регистр единиц измерения не учитывается. s = секунды, m = минуты, h = часы, d = дни, w = недели, KB = килобайты, MK = мегабайты, ... и т.д.

Ниже приведен пример строки с ограничениями:

```
BadUserClass:
     totalCPU = 1m
     totalConnectTime = 1h
```

Общие ограничения можно задать с помощью значений из предыдущей таблицы, но со следующими условиями:

- Если указано значение totalThreads, то оно не должно быть меньше totalProcesses.
- Если указано totalThreads, но не указано totalProcesses, то ограничение totalProcesses будет равно значению totalThreads.

Общие ограничения можно задать как для суперкласса, так и для подкласса. При проверке ограничений сначала проверяется значения для подкласса. Если указаны оба ограничения, то применяется наименьшее из них. Если значение для подкласса больше значения для суперкласса, то при загрузке конфигурации будет отправлено предупреждение, но конфигурация будет загружена. Это очень важно с точки зрения класса, поскольку общие ограничения абсолютны (не связаны с суперклассом), и один подкласс может потребить все ресурсы суперкласса. Если значение не указано, то применяется значение по умолчанию "-", указывающее, что ограничений нет. По умолчанию учет ресурсов для классов и процессов запускается при запуске WLM. Для отключения учета и настройки общих значений предназначена опция **-T [класс|процесс]** команды **wlmcntrl**.

Приоритет классов в WLM:

WLM распределяет ресурсы по классам путем присвоения приоритета каждого класса по отношению к каждому ресурсу.

Приоритет класса является динамической величиной и основан на приоритете ресурса, долях, ограничениях и текущем потреблении ресурсов этим классом. Классам с более высоким приоритетом предоставляется преимущественный доступ к ресурсу. На самом высоком уровне приоритеты классов также упорядочены. Классы с приоритетом 0 всегда будут иметь преимущество перед классами с приоритетом 1 (если не достигнут жесткий максимум).

При определении приоритетов, WLM применяет ограничения в следующем порядке (сверху вниз):

жесткие ограничения
Если потребление ресурсов классом превысит жесткий максимум, то классу будет предоставлен самый низкий приоритет, а выделение дополнительного объема ресурсов будет запрещено.

приоритет
При отсутствии жестких ограничений, приоритет класса ограничен минимумом и максимумом, допустимыми для данного приоритета.

гибкие ограничения
Если потребление ресурсов классом меньше гибкого максимума, то классу предоставляется более высокий приоритет. Если потребление ресурсов классом больше гибкого максимума, то классу предоставляется меньший приоритет.

доли Доли применяются для вычисления целевого потребления ресурсов классом. Приоритет класса повышается при снижении потребления ресурса и понижается при его увеличении. Учтите, что предпочтение отдается гибким ограничениям, и приоритет класса определяется на их основе.

Хотя для каждого класса и ресурса можно указать как долю, так и приоритет, результат будет более предсказуем в случае, если указано только одно значение.

Наборы ресурсов процессоров специального назначения:

Наборы ресурсов процессоров специального назначения (XRSET) позволяют администраторам выделить отдельные ресурсы для выполнения важных заданий. XRSET представляет собой именованный набор ресурсов, изменяющий способ работы всех входящих в его состав CPU. CPU специального назначения обрабатывает только программы, переданные ему на выполнение явным образом.

Создание XRSET

Для создания XRSET необходимы права доступа пользователя root. Создайте набор ресурсов в пространстве имен **sysxrset**с помощью команды **mkrset**. Например, команда **mkrset -c 1-3 sysxrset/set1** создает XRSET, содержащий CPU 1, 2 и 3. Кроме того, XRSET можно создать с помощью функции **rs_registername()**.

Просмотр XRSET, определенных в системе

Команда **lsrset -v -n sysxrset** позволяет просмотреть список всех XRSET, созданных в системе. (API, предназначенный для этой цели, в настоящее время отсутствует.)

Удаление XRSET

Для удаления XRSET необходимы права доступа пользователя root. Команда **rmrset** позволяет удалить XRSET. Кроме того, XRSET можно удалить с помощью функции **rs_discardname()**.

Перезапуск системы

В ходе перезагрузки системы все заданные XRSET удаляются из реестра.

Определение заданий для XRSET

Задания, подлежащие обработке с помощью процессоров специального назначения, можно указать различными способами. Наборы ресурсов, содержащие процессоры специального назначения, можно задать с помощью команд **attachrset** и **execrset**. Наборы ресурсов, содержащие процессоры специального назначения, можно связать с классами WLM. Задания, принадлежащие таким классам WLM, могут использовать процессоры специального назначения, указанные в наборах ресурсов.

Применение XRSETs совместно с Bindprocessor и _system_configuration.ncpus

Команда **bindprocessor** не позволяет передать задания на выполнение процессорам специального назначения. Процессоры специального назначения обрабатывают только задания, связанные с набором ресурсов.

При создании XRSET число процессоров, указанное в конфигурации системы (поле **_system_configuration.ncpus**), не изменяется. В системе остаются NCPU.

Программы, вызывающие NCPU с помощью **bindprocessor**, при обращении к CPU из XRSET получат сообщение об ошибке `EINVAL`. Связывание допустимо для все ИД, возвращаемых командой **bindprocessor**, в которой указана опция запроса. Опция запроса (**bindprocessor -q**) возвращает только допустимые ИД связывания, за исключением связанных с процессорами специального назначения.

Например, если в системе включены 10 процессоров, три из которых принадлежат XRSET, команда **bindprocessor -q** будет выполнена успешно, если в ней указать ИД связывания в диапазоне от 0 до 6. Если указать в команде **bindprocessor** ИД связывания в диапазоне от 7 до 9, будет выдано сообщение об ошибке `EINVAL`.

Применение XRSET с операциями динамического изменения конфигурации CPU

Как правило, процессоры с исключительным доступом не ограничивают операции динамического изменения конфигурации CPU. Однако после создания XRSET и назначения процессорам заданий вы не можете удалить отдельные CPU. В динамическом режиме CPU можно добавить в систему процессоров как общего, так и специального назначения. CPU добавляется в качестве процессора специального назначения, если в XRSET содержится соответствующий логический ИД CPU.

Наборы ресурсов в WLM:

WLM использует наборы ресурсов (или *rset*) для указания конкретного подмножества ресурсов системы, доступных классу. WLM управляет физическими ресурсами двух типов - оперативной памятью и процессорами. Набор ресурсов состоит из памяти и хотя бы одного процессора.

Определить набор ресурсов и присвоить ему имя можно с помощью SMIT. Затем с помощью интерфейсов администрирования WLM пользователь root или администратор суперкласса могут указать этой имя в качестве атрибута **rset** класса WLM. После этого каждый процесс из данного класса WLM сможет использовать только ресурсы из указанного набора.

В данный момент системы поддерживают только один домен памяти для всех наборов ресурсов, поэтому этот способ не разделяет физически рабочую нагрузку на .

Реестр набора ресурсов в WLM:

Реестр **rset** позволяет системным администраторам определять наборы ресурсов и присваивать им имена.

Для снижения риска появления повторяющихся имен в реестре поддерживается двухуровневая схема именования. Имя набора ресурсов представляется в форме *пространство_имен*/*имя_rset*. Как *пространство_имен*, так и *имя_rset* могут быть длиной до 255 символов, включая прописные и строчные буквы, цифры, символы подчеркивания и точки (**.**). Прописные и строчные буквы различаются. *Пространство_имен* **sys** зарезервировано системой и используется для определения наборов **rset** системных ресурсов.

Имена определений **rset** не могут повторяться в одном пространстве имен. При создании нового определения **rset** с тем же именем, что и существующий **rset**, новое определение заменяет старое, включая все права доступа и привилегии. Создавать, изменять и удалять наборы ресурсов и обновлять встроенную базу данных **rset** с помощью SMIT может только пользователь root.

С каждым определением **rset** связаны ИД владельца, ИД группы и права доступа. Эти атрибуты задаются при создании определения **rset** и позволяют контролировать доступ к ресурсам. Как и в случае обычных файлов, применяются отдельные права доступа на чтение и запись для пользователей и групп. Права на чтение позволяют просматривать определение **rset**, а права на запись - изменять и удалять это определение.

Определения **rset**, заданные системными администраторами, хранятся в файле настройки `/etc/rsets`. Формат этого файла не документирован; пользователи должны управлять **rset** с помощью SMIT. Как и определения классов WLM, определения **rset** должны быть сначала загружены в структуры данных ядра.

Задачи, связанные с данной:

“Создание набора ресурсов” на стр. 494
Применение наборов ресурсов позволяет эффективно разделить процессы на основании используемых ими процессоров. Разделив рабочие схемы на два класса и выделив каждому классу отдельное подмножество процессоров, можно устранить конкуренцию за процессоры между этими рабочими схемами, даже если они конкурируют за физическую память и ресурсы ввода-вывода.

Настройка управления рабочей схемой

Определения и атрибуты классов, долю общих ресурсов, ограничения и правила классификации можно задать с помощью SMIT или из командной строки. Все определения и правила хранятся в обычных текстовых файлах, которые можно создать и изменить с помощью текстового редактора.

Эти файлы (называемые *файлами свойств WLM*) хранятся в подкаталогах `/etc/wlm`. Набор файлов, описывающий все суперклассы и связанные с ними подклассы, составляет конфигурацию WLM. Файлы конфигурации WLM с именем **Config** хранятся в подкаталогах `/etc/wlm/Config`. В этом каталоге хранятся определения параметров суперклассов WLM. Каталог содержит файлы `description`, `classes`, `shares`, `limits` и `rules`. Кроме того, указанный каталог может содержать подкаталоги, имена которых совпадают с именами определенных суперклассов, хранящих определения подклассов. Например, для суперкласса *Super* из конфигурации **Config** в каталоге `/etc/wlm/Config/Super` будут храниться файлы свойств подклассов суперкласса *Super*. Каталог содержит файлы `description`, `classes`, `shares`, `limits` и `rules`.

После создания конфигурации WLM ее можно активизировать с помощью команды быстрого доступа **smit wlmmanage** или команды **wlmcntrl**.

Определив несколько конфигураций подсистемы управления рабочей схемой, вы можете создать несколько наборов файлов свойств. Такие конфигурации обычно хранятся в подкаталогах каталога `/etc/wlm`. Символьная связь `/etc/wlm/current` указывает на каталог с текущим набором файлов конфигурации. Эта связь обновляется командой **wlmcntrl** при запуске WLM с тем или иным набором фалов конфигурации.

Информация, связанная с данной:

“Настройка WLM для объединения ресурсов” на стр. 496
Средства управления рабочей схемой (WLM) позволяют управлять ресурсами, которые применяются в различных заданиях системы.

Требования к приложению для конфигурации WLM:

Для того чтобы определить, какие классы потребуются, вы должны знать требования к вычислительной мощности со стороны всех пользователей и приложений, выполняемых в системе. Необходимо также знать, какие приложения наиболее важны, а для каких можно ограничить объем ресурсов. На основе полученных знаний вы сможете определить суперклассы и подклассы.

Выбор приоритетов зависит от того, какую функцию выполняет WLM в вашей организации. Если необходимо оптимизировать нагрузку на сервер, то вам уже должны быть известны приложения, пользователи и требования к ресурсам, поэтому некоторые шаги настройки можно пропустить.

WLM позволяет классифицировать процессы на основании пользователя, группы, приложения, типа, тега или любой комбинации этих атрибутов. WLM распределяет ресурсы между классами, поэтому системный администратор должен сгруппировать приложения и пользователей по классам на основе требований к ресурсам. Например, вы можете отделить интерактивные процессы, практически не использующие процессор, но требующие быстрого ответа от системы, от пакетных заданий, интенсивно работающих с процессором и памятью. Эта задача аналогична отделению потока OLTP от информации, полученной в результате выполнения запросов к базе данных.

Эту задачу можно выполнить из SMIT или командной строки. Для создания первой конфигурации WLM, включая определение суперклассов и задание их атрибутов, рекомендуется воспользоваться SMIT. Вначале вы можете задать лишь некоторые атрибуты, оставив для остальных атрибутов значения по умолчанию. Это касается как доли общих ресурсов, так и ограничений на использование ресурсов. Эти свойства класса можно будет динамически изменить позднее.

Затем запустите WLM в пассивном режиме, проверьте, что классификация процессов выполняется правильно, и начните изучать требования приложений к ресурсам системы.

Проверьте конфигурацию с помощью команды **wlmcheck** или соответствующих меню SMIT. Запустите WLM в пассивном режиме. WLM классифицирует все существующие процессы (и процессы, созданные после запуска WLM) и начнет собирать статистику об использовании процессора и памяти, а также о числе обращений к дискам для различных классов. WLM не будет перераспределять ресурсы.

Убедитесь, что процессы разбиты на классы в точности так, как планировал системный администратор (укажите флаг **-o** в команде **ps**). Если некоторые классы созданы неправильно, то измените для них правила классификации или установите бит наследования (если новые процессы должны оставаться в том же классе, что и родительский процесс) и обновите WLM. Повторяйте эту процедуру до тех пор, пока все процессы не будут правильно классифицированы на первом уровне (уровне суперклассов).

Запуск и обновление WLM в пассивное режиме относительно безопасно, не увеличивает нагрузку на систему и не влияет на обычную работу системы. Для запуска и обновления WLM применяется команда **wlmcntrl**, которая может быть вызвана из командной строки или из SMIT.

Запустите WLM в пассивном режиме и соберите статистическую информацию с помощью команды **wlmstat**. Команду **wlmstat** можно запускать через равные промежутки времени и просматривать долю ресурсов, выделяемую различным суперклассам (относительно общего объема ресурсов). Таким образом вы сможете отслеживать использование ресурсов основными приложениями в течение длительных периодов времени.

Приоритеты, доли общих ресурсов и ограничения в WLM:

На основании собранной статистики и целей определите приоритеты и ограничения на использование ресурсов для всех классов.

В некоторых случаях необходимо задать максимальные и минимальные ограничения. Вначале попробуйте достичь поставленной цели путем настройки приоритетов и доли общих ресурсов. Ограничения следует применять только в случаях, когда решить задачу с помощью указанных выше параметров невозможно. В некоторых ситуациях вам потребуется создать подклассы.

- Укажите минимальные значения для приложений, которые занимают мало ресурсов, но требуют быстрого отклика на внешние события. Например, если интерактивное приложение простаивает в течение длительного периода времени, его страницы памяти будут отобраны другими заданиями. Для того чтобы зарезервировать за приложением из класса с нулевым приоритетом некоторое количество страниц, укажите минимальный объем памяти.

- Максимальные значения обычно указываются для заданий с низким приоритетом, активно использующих ресурсы. Если ресурсы системы не распределены по разделам, жесткий максимум следует указывать только для невозобновляемых ресурсов, например, памяти. Это связано с тем, что для перераспределения страниц памяти между процессами требуется значительное время на выгрузку данных в пространство подкачки. В случае ресурсов процессора выбор правильного сочетания значений гибкого максимума и приоритета приведет к тому, что процессам с более высоким приоритетом немедленно выделяется необходимый объем ресурсов.

При создании и настройке параметров подклассов вы можете обновлять только ту часть конфигурации WLM, которая связана с подклассами заданного суперкласса, не затрагивая процессы из других суперклассов.

При необходимости создайте другие конфигурации с другими значениями параметров параметрами. Вы можете значительно сэкономить свое время, создав новую конфигурацию путем копирования существующей конфигурации и изменения ее параметров.

Точная настройка конфигурации WLM:

Контролируйте работу системы с помощью команды **wlmstat**, чтобы убедиться, что WLM правильно распределяет ресурсы, то есть некоторые процессы не получают слишком много ресурсов, тогда как объем ресурсов других процессов необоснованно ограничивается. Если ресурсы распределяются неправильно измените долю общих ресурсов, выделяемых различным процессам, и обновите WLM.

В процессе отслеживания и настройки долей, ограничений и приоритетов определите, следует ли передать права на управление подклассами другим администраторам. Затем администратор может собрать дополнительную информацию и настроить параметры по своему усмотрению.

Администратор каждого суперкласса может выполнить эту операцию для всех подклассов своего суперкласса. Единственное отличие заключается в том, что WLM не может работать в пассивном режиме на уровне подклассов. Настройку подклассов нужно выполнять в активном режиме WLM. Для того чтобы настройка не повлияла на работу пользователей и приложений, вначале укажите значения по умолчанию для параметров приоритета, ограничений (0% для минимума и 100% для обоих максимумов) и доли общих ресурсов ('-'). В этом случае WLM не будет управлять распределением ресурсов между подклассами.

Устранение неполадок управления рабочей схемой

Если система работает не так, как ожидалось, вам может потребоваться изменить настройку WLM.

Значения процентного потребления для каждого класса можно просмотреть с помощью команды **wlmstat**. Собрав и проанализировав эти данные, вы можете определить, какие изменения необходимо внести в конфигурацию. После обновления конфигурации WLM необходимо активировать ее с помощью команды **wlmcntrl -u**.

Следующие рекомендации помогут вам выбрать способ обновления конфигурации:

- Если число активных общих ресурсов для какого-либо уровня сильно изменяется с течением времени, то вы можете не выделять явно ресурсы классу, чтобы он получал необходимый объем ресурсов независимо от числа активных общих ресурсов. Такой подход можно применять для важных классов, требующих приоритетного доступа к ресурсам.
- Если необходимо обеспечить гарантированный доступ к определенному объему ресурсов, то укажите для этих ресурсов ограничение снизу. Такой подход может применяться для интерактивных заданий, которые не потребляют большого объема ресурсов, но должны быстро реагировать на внешние события.
- Если необходимо ограничить доступ к ресурсам, но это невозможно достаточно эффективно сделать с помощью общих ресурсов, то задайте ограничения сверху. В большинстве случаев достаточно мягкого ограничения сверху, однако для строгого управления доступом можно применять жесткие ограничения. Поскольку жесткие ограничения сверху могут привести к неэффективному использованию системных

ресурсов, а при использовании для управления ресурсами памяти - и к увеличению числа операций подкачки, то перед заданием жестких ограничений необходимо задать для остальных классов ограничения сверху.

- Если менее важные задания мешают работе более важных, то переместите менее важные задания на более низкий уровень. Такой подход обеспечит более низкий приоритет менее важных заданий, не позволяя им конкурировать за доступные ресурсы с работающими более важными заданиями.
- Если класс не может получить необходимый объем ресурсов, то выясните, не может ли эта ситуация быть вызвана конкуренцией за другой ресурс. Если это так, то измените для класса опции выделения ресурса, за который идет конкуренция.
- Если процессы в пределах класса сильно различаются по своей работе и уровню использования ресурсов, то увеличьте число классов для более тонкого управления ресурсами. Кроме того, вы можете создать отдельный класс для каждого важного приложения.
- Если анализ показал, что ресурсы, необходимые одному классу, зависят от уровня использования ресурсов другим классом, то измените выделение ресурсов в соответствии с полученными сведениями. Например, если объем ресурсов, необходимый классу ClassZ, зависит от числа запросов, которые может обработать класс ClassA, то у класса ClassA должен быть гарантированный доступ к объему ресурсов, позволяющему обеспечить потребности класса ClassZ.
- Если одному или нескольким приложениям постоянно не хватает ресурсов, необходимых для нормальной работы, то единственным выходом является снижение нагрузки на систему.

Примечание: Для снижения нагрузки на администратора WLM вы можете определить пользователя *adminuser* для суперкласса. После выполнения основной настройки вы можете предоставить внесение последующих изменений (включая создание и настройку подклассов) администраторам суперкласса, т.е. пользователям adminuser.

API управления рабочей схемой

Приложения могут использовать API WLM, представляющие собой набор функций в библиотеке `/usr/lib/libwlm.a` для запуска задач, которые администратор WLM может выполнять с помощью интерфейса командной строки.

В том числе:

- Создание, изменение и удаление классов
- Изменение атрибутов и параметров класса
- Удаление классов
- Классификация процессов вручную
- Получение статистической информации о WLM

Этот API позволяет приложениям задать специальный атрибут классификации - тег. Формат тега приложения описан в документации по приложению; тег позволяет различать экземпляры одного приложения при распределении ресурсов. Таким образом, различные экземпляры приложения можно отнести к различным классам.

Кроме того, функция **wlm_set_tag** позволяет приложению задать тег приложения и режим наследования этого тега дочерними процессами при вызовах функций **fork** и **exec**. Нитям также можно присвоить теги приложений с помощью тега **wlm_set_thread**. Тег приложения нити может наследоваться функциями **fork**, **exec** или **pthread_create**. Эта библиотека поддерживает многонитевые 32- и 64-разрядные приложения.

Тег приложения:

Тег приложения - это строка символов, применяемая в качестве одного из критериев для автоматической классификации процессов (согласно файлу `правила`). Обычно этот критерий задается приложением в дополнение к системным критериям, таким как пользователь пользователь (*user*), группа (*group*), приложение (*application*) и тип (*type*).

Сразу после того как процесс или нить задает тег приложения, выполняется его повторная классификация в соответствии с действующими правилами выбора суперкласса и подклассов, определенными для текущей конфигурации WLM. При просмотре правил классификации учитываются все атрибуты процесса, включая новый тег.

Тег влияет на класс процесса, если он указан хотя бы в одном правиле классификации. Формат и значение тегов должны быть подробно описаны в документации по приложению, чтобы администраторы WLM могли применять различные значения тегов в правилах классификации, действующих для разных экземпляров приложения.

Поскольку у разных пользователей могут быть разные требования к набору характеристик процессов приложения, применяемых для их классификации, разработчикам приложений рекомендуется предусматривать возможности по настройке компонентов тега с помощью файлов конфигурации или атрибутов времени выполнения. Администратор приложения может задать формат тега приложения. Применяемые при этом атрибуты и формат тега WLM зависят от приложения и определяются его разработчиком.

Например, экземпляр сервера базы данных может сообщить информацию о текущей базе данных (*db_name*) и порте TCP, к которому подключен пользователь (*port_num*). Администраторы могут использовать эту информацию следующим образом:

- Создать разные классы для процессов, работающих с разными базами данных, и указать для этих классов различные параметры доступа к ресурсам.
- Разделить процессы, обрабатывающие запросы различных клиентов, по номеру порта.
- Создать один суперкласс для каждой базы данных и подкласс для каждого номера порта.

Для того чтобы администратор мог решить эти задачи, у него должна быть возможность задать содержимое и формат тега. Предположим, что формат тега передается приложению в файле конфигурации или параметре времени выполнения, например, `WLM_TAG=`*`$db_name`* или `WLM_TAG=`*`$db_name_$port_num`*.

При задании тега приложение может указать, должен ли этот тег наследоваться дочерними процессами; при этом дочерние процессы будут относиться к тому же классу, что и родительский процесс. Обычно дочерним процессам разрешается наследовать тег.

Ниже приведен пример применения тегов приложения. В этом примере тег базы данных совпадает с ее именем. Два экземпляра сервера, работающие с разными базами данных, задают два разных тега, например, `db1` и `db2`.

Системный администратор может создать два класса, `dbserv1` и `dbserv2`, и определить классы серверов баз данных и их дочерних процессов (если включено наследование тега) с помощью тега. Затем с этими классами могут быть связаны различные параметры доступа к ресурсам.

Для реализации этой схемы можно задать следующие правила классификации:

```
* класс   resvd польз. группа    приложение      тип   тэг
*
dbserv1     -    -     dbadm     /usr/sbin/dbserv  -   db1
dbserv2     -    -     dbadm     /usr/sbin/dbserv  -   db2
```

Типы интерфейсов программирования приложения:

Ниже описаны типы интерфейсов программирования приложений WLM (WLM).

API управления классами

С помощью API WLM приложения могут выполнять следующие действия:

- Запрашивать имена и свойства существующих классов из указанной конфигурации WLM (**wlm_read_classes**).

- Создавать в указанной конфигурации WLM новые классы, задавать значения различных атрибутов классов (приоритет, наследование и пр.), а также настраивать режимы общего доступа и ограничения на использование для ресурсов, находящихся под управлением WLM, таких как CPU, физическая память и блочный ввод-вывод (**wlm_create_class**).
- Изменять свойства существующих классов из заданной конфигурации WLM, в том числе атрибуты классов, а также ограничения на использование и режимы общего доступа для ресурсов (**wlm_change_class**).
- Удалять существующие классы из указанной конфигурации (**wlm_delete_class**).

Все изменения вносятся только в файлы свойств указанной конфигурации WLM. Однако если в качестве имени конфигурации будет указана пустая строка, то изменения будут внесены в классы, загруженные в память, что приведет к немедленному обновлению текущей конфигурации.

Для вызова указанных API требуются те же права доступа, что и для выполнения аналогичных действий с помощью командной строки или интерфейса SMIT:

- Просмотреть имя и свойства класса может любой пользователь
- Создать, изменить или удалить суперкласс может только пользователь root
- Создать, изменить или удалить подкласс заданного суперкласса может только пользователь root или администратор суперкласса (задается в атрибуте **adminuser** или **admingroup** суперкласса).

Если управление WLM осуществляется одновременно администраторами WLM из командной строки и приложениями с помощью API, следует принять дополнительные меры предосторожности. Оба интерфейса работают в одном пространстве имен и объектов.

Кроме того, если приложение с помощью API изменит текущую конфигурацию WLM (например, создаст новые классы), то администратор WLM узнает об этом изменении только увидев новые классы в выводе команды **wlmstat** или аналогичной команды. Для того чтобы избежать ошибок в приложениях, которые могут возникнуть при вызове API во время обновления WLM администратором, классы, созданные с помощью этого API, но не определенные в файле свойств WLM, не удаляются автоматически из текущей конфигурации. Они продолжают существовать до явного удаления функцией **wlm_delete_class** или командой **rmclass** (вызванной системным администратором из командной строки или интерфейса SMIT).

Кроме того, API WLM позволяют приложениям выполнять следующие действия:

- Просматривать и изменять режим работы WLM с помощью функции **wlm_set**
- Узнавать текущее состояние WLM
- Завершать работу WLM
- Изменять активный режим на пассивный, и наоборот
- Включать и выключать связывание **rset**
- Запускать и обновлять текущую или указанную конфигурацию WLM функцией **wlm_load**
- Назначать класс процессу или группе процессов функцией **wlm_assign**

Для вызова указанных API требуются те же права доступа, что и для выполнения аналогичных действий командами **wlmcntrl** и **wlmassign**:

- Просмотреть информацию о состоянии WLM может любой пользователь
- Изменять режим работы WLM может только пользователь root
- Обновлять всю конфигурацию может только пользователь root
- Обновлять подкласс заданного суперкласса может только пользователь root или администратор суперкласса (указанный в атрибуте **adminuser** или **admingroup**)
- Добавлять процессы в суперкласс или подкласс может только пользователь root, пользователь со специальными правами доступа (указанный в параметре **authuser** или **authgroup**), либо администратор суперкласса (указанный в атрибуте **adminuser** или **admingroup**).

API статистики WLM

Функции API WLM и **wlm_get_bio_stats** предоставляют приложениям доступ ко всей статистической информации WLM, которую можно получить командой **wlmstat**.

API классификации WLM

Функция **wlm_check** позволяет пользователю просматривать определения классов и правила классификации указанной конфигурации WLM. Функция API **wlm_classify** позволяет приложению выяснить, к какому классу был бы отнесен процесс с указанными атрибутами.

Информация, связанная с данной:

Команда wlmassign

Двоичная совместимость:

Для обеспечения двоичной совместимости с будущими версиями структур данных во всех вызовах API указывается номер версии.

Это позволит библиотеке определить формат структур данных, ожидаемый приложением.

Примеры классификации, правил и ограничений WLM

Существует несколько одновременно применяемых способов классификации процессов.

Максимальная гибкость настройки достигается при использовании алгоритма последовательного поиска первого совпадения. Вы можете группировать процессы следующим образом: по имени пользователя, предусмотрев специальные правила для некоторых программ; по имени программы, предусмотрев специальные правила для некоторых пользователей; другими способами.

Пример правил назначений WLM:

В данном примере показан файл `rules` верхнего уровня для конфигурации *Config* (файл `/etc/wlm/Config/rules`).

```
* Этот файл содержит правила, применяемые WLM
* для определения суперклассов процессов
*
* класс зарез. пользователь группа приложение         тип         тег
db1      -      -       -    /usr/bin/oracle*      _DB1
db2      -      -       -    /usr/bin/oracle*      _DB2
devlt    -      -      dev        -            -    -
VPs      -   bob,ted    -         -            -    -
acctg    -      -     acct*      -            -    -
System   -    root      -         -            -    -
Default   -       -        -       -                -          -
```

Ниже приведен пример файла `rules` для суперкласса **devlt** с полным именем `/etc/wlm/Config/devlt/rules`:

```
* Этот файл содержит правила, применяемые WLM
* для определения подкласса суперкласса devlt,
* к которому относится процесс
*
* класс зарез. пользователь группа приложение         тип         тег
hackers   -    jim,liz     -       -                -         -
hogs      -       -        -       -            64bit+plock    -
editors   -     !sue       -   /bin/vi,/bin/emacs    -         -
build     -       -        -   /bin/make,/bin/cc     -         -
Default   -       -        -       -                -         -
```

Примечание: Звездочкой (*) помечены строки файла `rules`, содержащие комментарий.

Ниже приведены примеры применения файлов правил. В приведенных примерах предполагается, что ни для одного класса не установлен атрибут наследования. В противном случае новые процессы относились бы к тем же классам, что и их родительские процессы.

- Если пользователь joe из группы acct3 запустит **/bin/vi**, процесс попадет в суперкласс acctg.
- Если пользователь sue из группы dev запустит **/bin/emacs**, процесс попадет в суперкласс devlt (по идентификатору группы), но не будет помещен в подкласс editors, поскольку пользователь исключен из этого класса. По умолчанию процессу будет присвоен класс devlt.
- Непосредственно после запуска программы **/usr/sbin/oracle** с идентификатором пользователя oracle и идентификатором группы dbm для обслуживания базы данных DB1 новый процесс будет помещен в суперкласс Default. Только после того, как процесс изменит свой тег на _DB1, ему будет присвоен суперкласс db1.

Пример классов WLM с общими долями и ограничениями:

Например, предположим, что классам A, B, C и D присвоены доли 3, 2, 1, и 1, соответственно.

Если активны классы A, C и D, то целевые значения будут выключены следующим образом:

```
target(A) = 3/5 = 60%
target(C) = 1/5 = 20%
target(D) = 1/5 = 20%
```

Если во время тестирования будет обнаружено, что приложения из класса A работают нормально при использовании 50% ресурса, то оставшиеся 50% можно предоставить другим классам. Для этого классу A можно присвоить гибкий максимум в 50% ресурса. Поскольку текущее целевое значение равно 60%, то оно будет изменено для удовлетворения заданному гибкому максимуму. В этом случае целевое значение или реальное потребление (наименьшее из этих значений) для класса A вычитается из общего объема ресурсов. Поскольку целевое значение для класса A было вычтено из ограничения (а не из доли ресурса), то доля ресурса также вычитается из общего числа активных долей. Если текущее потребление ресурса классом A равно 48%, то будут применяться следующие целевые значения:

```
target(A) = 3/5 = 60%, softmax = 50, = 50%
target(C) = 1/2 * (100 - 48)          = 26%
target(D) = 1/2 * (100 - 48)          = 26%
```

Если через некоторое время снова станут активными все классы, то целевые значения снова будут изменены:

```
target(A) = 3/7 = 42%
target(B) = 2/7 = 28%
target(C) = 1/7 = 14%
target(D) = 1/7 = 14%
```

Пример классов WLM с ограничениями CPU:

В данном примере рассматривается распределение ресурсов CPU, причем предполагается, что каждый класс может использовать все предоставленные ему ресурсы.

Существует два класса с одинаковым приоритетом - A и B. Для класса A задано ограничение на объем ресурсов CPU [30% - 100%]. Для класса B задано ограничение [20% - 100%]. Если оба класса работают одновременно и не требуют выделения дополнительных ресурсов CPU, то WLM сначала обеспечит выделение каждому из этих классов минимальной доли процессорного времени в секунду (с усреднением за несколько секунд). Остальное процессорное время будет распределено WLM в соответствии с целевыми значениями ресурсов CPU.

Если целевые значения для классов A и B равны 60% и 40% соответственно, то значения коэффициента использования CPU для A и B стабилизируется на уровне 60% и 40%.

Предположим, что был добавлен класс C. Этот класс представляет собой группу заданий, связанных с CPU, которым должно быть предоставлено не менее половины процессорного времени. Для класса C заданы

ограничения [20% - 100%] и целевое значение ресурсов CPU 100%. Если приоритет классов A, B и C одинаков, то при запуске задания класса C доля ресурсов CPU, предоставляемая классам A и B будет постепенно снижаться до тех пор, пока коэффициенты использования CPU не стабилизируются на уровне 30%, 20% и 50% для классов A,B и C соответственно. Целевые значения для A и B в этом случае также будут минимальными.

Системный администратор может запретить выделение пакетным заданиям более 50% ресурсов CPU при наличии в системе других заданий, возможно, с более высоким приоритетом. Допустим, что в приведенном выше примере классу C соответствует более низкий приоритет. В этом случае заданию класса C будут предоставлены ресурсы CPU, не использованные заданиями классов A и B. В приведенном примере ресурсы CPU не будут предоставлены заданию класса C, поскольку предполагается, что задания класса A и B могут полностью загрузить процессор. Однако в большинстве случаев классы с более высоким приоритетом (как классы A и B) состоят из интерактивных заданий и заданий обработки транзакций, не использующих постоянно все ресурсы CPU. В такой ситуации заданию класса C будет предоставлен некоторый объем ресурсов CPU, за которые оно будет конкурировать с заданиями других классов, имеющих тот же или более низкий приоритет.

Пример классов WLM с ограничениями памяти:

В этом примере рассматривается распределение памяти между группами процессов, для работы которых требуется различный объем памяти.

Необходимо обеспечить работу трех групп процессов: группы интерактивных процессов, которые должны выполняться по мере необходимости (PEOPLE); группы пакетных заданий, постоянно работающих в фоновом режиме (BATCH1); группы более важных пакетных заданий, выполняемых каждую ночь (BATCH0).

Для группы PEOPLE указан минимальный объем памяти 20%, целевой объем общей памяти 50 и приоритет 1. Ограничение в 20% обеспечивает быстрый отклик приложений рабочего стола на действия пользователей после перерывов в работе.

Для группы BATCH1 указан минимальный объем памяти 50%, целевой объем общей памяти 50 и приоритет 3.

Для группы BATCH0 указан минимальный объем памяти 80%, целевой объем общей памяти 50 и приоритет 2.

Суммарный минимальный объем памяти для классов PEOPLE и BATCH1 составляет 70. При обычной работе (когда процессы BATCH0 не выполняются) оба класса могут получить всю необходимую память. Остальная часть системной памяти разделяется между этими процессами поровну, несмотря на различие их приоритетов. При запуске BATCH0 минимальное ограничение на объем памяти достигает 150. До завершения работы процессов с более высоким приоритетом WLM будет игнорировать минимальные требования процессов с более низким приоритетом. BATCH0 будет использовать память из 50% доли BATCH1, но не из 20% доли PEOPLE. После завершения работы BATCH0 память будет предоставлена процессам с приоритетом 3, их работа будет продолжена, а прежнее распределение памяти будет восстановлено.

Команды управления рабочей схемой

Команды WLM позволяют системным администраторам выполнять следующие действия.

В том числе:

- Создавать, изменять и удалять суперклассы и подклассы с помощью команд **mkclass**, **chclass** и **rmclass**. Эти команды обновляют *классы*, *общие ресурсы* и *ограничения* файлов.
- Запускать, останавливать и обновлять WLM с помощью команды **wlmcntrl**.
- Находить в файлах параметров WLM требуемую конфигурацию и определять класс процесса с помощью команды **wlmcheck**.

- Отслеживать использование ресурсов классами с помощью команды **wlmstat** (в формате ASCII). Большинство средств управления производительностью, такие как **svmon** или **topas**), могут выдавать статистику, разбитую на классы WLM.
- Определять класс процесса и приложения с помощью команды **ps**. Команда **ps** также позволяет найти все процессы, принадлежащие подклассу или суперклассу.
- Команды для управления правилами присвоения не предусмотрены. Это можно сделать с помощью SMIT или текстового редактора.

Узлы устройств

Устройства объединены в кластеры, называемые *узлами*. Каждый узел представляет собой логическую подсистему устройств, в которой устройства нижнего уровня находятся в зависимости типа родитель-потомок от устройств более высокого уровня.

Например, системный узел - самый верхний и содержит все физические устройства системы. Устройство Система - это узел самого верхнего уровня; на следующем уровне находятся шины и адаптеры, зависящие от устройства верхнего уровня. На самом нижнем уровне иерархии находятся устройства, к которым не подключены никакие другие устройства. Такие устройства зависят от всех устройств, находящихся на более высоких уровнях иерархии.

При запуске анализ зависимостей устройств позволяет настроить все устройства, составляющие узел. Настройка начинается с узла верхнего уровня. Все устройства более низкого уровня не настраиваются до тех пор, пока не будет настроено устройство верхнего уровня.

Операционная система AIX поддерживает функцию разветвленного ввода-вывода (MPIO). Если драйвер устройства поддерживает MPIO, то у устройства может быть несколько предков в иерархии. У такого устройства существует несколько путей к системе или логическому разделу.

Классы устройств

При управлении устройствами операционная система должна знать, подключение каких устройств допустимо. В операционной системе устройства разделены на три иерархические группы.

К этим группам относятся:

- Функциональные классы
- Функциональные подклассы
- Типы устройств

Функциональные классы состоят из устройств, выполняющих одинаковые функции. Например, один из функциональных классов составляют принтеры как устройства для печати. Функциональные классы разделены на подклассы в зависимости от более тонких различий между устройствами. Например, интерфейсы принтеров бывают последовательные или параллельные. Последовательные принтеры составляют один подкласс, параллельные - другой. Различные типы устройств определяются по модели и изготовителю.

Классы устройств образуют в операционной системе допустимые связи родитель-потомок. Для этого применяются подклассы, которые могут быть подключены только к одному из допустимых родительских устройств. Например, словосочетание "8-портовый адаптер RS-232" означает, что к любому из 8 портов этого адаптера можно подключать только устройства, относящиеся к подклассу RS-232.

Классы и иерархические зависимости устройств хранятся в базе данных конфигурации устройств Администратора объектных данных (ODM).

База данных конфигураций устройств и управление устройствами

Информация об устройствах содержится в базах данных описаний и настройки, которые составляют базу данных конфигурации устройств.

База данных описаний содержит данные о конфигурации всех возможных устройств, поддерживаемых системой. В этой базе данных хранится информация об иерархическом классе каждого устройства. База данных настройки содержит данные о конфигурации всех определенных и настроенных устройств системы. Каждому из подключенных к системе устройств в этой базе данных соответствует определенная запись.

Администратор настройки - это программа, автоматически настраивающая устройства при запуске или во время работы системы. Администратор настройки считывает информацию из баз данных описаний и настройки и обновляет информацию в базе данных настройки.

Функция многоканального ввода-вывода (MPIO) использует уникальный идентификатор устройства (UDID) для идентификации каждого устройства MPIO, вне зависимости от применяемого пути. UDID хранится в базе данных конфигурации устройств. При обнаружении устройства система с помощью UDID определяет, обнаружено ли новое устройство или путь к уже существующему устройству. При наличии нескольких путей к устройству драйвер устройства или расширение ядра Path Control Manager определяет, какой путь будет применяться для обработки конкретного запроса.

Для выполнения задач по управлению устройствами, таких как их удаление, добавление или настройка, вы можете применять SMIT и командную строку.

Понятия, связанные с данным:

“Управление совместимыми с MPIO устройствами” на стр. 540
Технология разветвленного ввода-вывода (MPIO) позволяет определить альтернативные пути к устройству для восстановления после сбоя.

Задачи, связанные с данной:

“Подготовка к установке устройства” на стр. 386
Установка устройства в системе заключается в определении расположения устройства, в физическом подключении и настройке устройства с помощью Администратора конфигурации, или SMIT.

Состояния устройства

Подключенные к системе устройства могут находиться в одном из следующих состояний.

Подключенные к системе устройства могут находиться в одном из следующих состояний:

Элемент	Описание
Не определено	Устройство неизвестно системе.
Определено	В базу данных настройки занесена информация об устройстве, но оно недоступно системе.
Доступно	Устройство определено, подключено к системе и настроено.
Остановлено	Устройство недоступно, но контролируется драйвером.

Если терминал и принтер вместе используют одно соединение, оба эти устройства будут иметь одного родителя и порт в базе данных конфигурации. В каждый момент времени может быть настроено только одно из этих устройств. При настройке соединения с терминалом информация об установках принтера сохраняется для будущего применения. Устройство не удаляется; оно находится в состоянии Определено. В этом состоянии находятся устройства, которые в данный момент не используется или временно отключены от системы; для них сохраняется вся информация о настройке.

Устройство может быть сделано доступным, если для него существует драйвер.

Некоторые устройства, особенно псевдоустройства TCP/IP, могут находиться также в остановленном состоянии.

Задачи, связанные с данной:

“Удаление конфигурации асинхронного абзаца” на стр. 556
Можно удалить конфигурацию асинхронного адаптера.

Коды расположения устройств

Код расположения - это совокупность названия блока CPU или системного блока, адаптера, кабелей и асинхронного блока распределения (если он есть), к которым подключено устройство или рабочая станция. Этот код представляет собой один из способов идентификации физических устройств.

Код расположения содержит не более четырех информационных полей. Число полей зависит от типа устройства. Поля задают блок, разъем, соединитель и порт. В каждом из полей указывается два символа.

Код расположения блока состоит только из поля блока, содержащего два символа. Код расположения адаптера состоит из полей блока и разъема в формате *AA-BB*, где *AA* относится к расположению блока, а *BB* задает шину и разъем адаптера. Код расположения остальных устройств задается в формате *AA-BB-CC* или *AA-BB-CC-DD*, где*AA-BB* - это код расположения адаптера, к которому подключено устройство, *CC* задает соединитель адаптера, а *DD* задает номер порта или адрес устройства SCSI.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Коды расположения адаптера

Код расположения адаптера представляет собой две пары цифр в формате *AA-BB*, где *AA* задает код расположения блока адаптера, а *BB* - шину ввода-вывода и разъем карты адаптера.

Значение `00` поля **AA** означает, что адаптер расположен в блоке CPU или системном блоке, в зависимости от типа системы. Если в поле **AA** указано другое значение, значит карта адаптера расположена в блоке расширения ввода-вывода. В этом случае значение *AA* задает шину ввода-вывода и номер разъема асинхронного адаптера. Первая цифра задает шину ввода-вывода. Значение `0` соответствует стандартной шине ввода-вывода, а `1` - дополнительной. Вторая цифра задает номер разъема указанной шины ввода-вывода.

Первая цифра поля **BB** задает плату ввода-вывода карты адаптера. Если карта находится в блоке CPU или в системном блоке, то для стандартной шины ввода-вывода эта цифра будет равна `0`, а для дополнительной - 1. Если карта адаптера расположена в блоке расширения ввода-вывода, то эта цифра будет равна `0`. Вторая цифра задает номер разъема указанной шины ввода-вывода (или номер разъема блока расширения ввода-вывода), к которому подключен адаптер.

Код расположения `00-00` соответствует стандартной плате ввода-вывода.

Примеры:

Элемент	Описание
`00-05`	Задает карту адаптера в разъеме номер 5 стандартной платы ввода-вывода, расположенную в блоке CPU или системном блоке, в зависимости от типа системы.
`00-12`	Задает адаптер в разъеме номер 2 дополнительной шины ввода-вывода, расположенный в блоке CPU.
`18-05`	Задает адаптер в разъеме номер 5 блока расширения ввода-вывода. Блок подключен к асинхронному адаптеру расширения, установленному в разъеме 8 дополнительной шины ввода-вывода блока CPU.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Коды расположения принтера и графопостроителя

Код расположения `00-00-S1-00` или `00-00-S2-00` задает принтер, графопостроитель или терминал, подключенный к последовательному порту стандартной платы ввода-вывода `s1` или `s2`. Код расположения `00-00-0P-00` задает параллельный принтер, подключенный к параллельному порту стандартной платы ввода-вывода.

Остальные коды расположения задают принтеры, графопостроители и терминалы, подключенные к карте адаптера, отличной от стандартной платы ввода-вывода. Код расположения таких принтеров, графопостроителей и терминалов задается в формате *AA-BB-CC-DD*, где *AA-BB* соответствует коду

расположения контроллера.

Элемент	Описание
AA	Значение `00` поля **AA** соответствует карте адаптера, расположенной в блоке CPU или в системном блоке, в зависимости от типа устройства. Остальные значения поля **AA** задают адаптер, расположенный в блоке расширения ввода-вывода. Первая цифра задает шину ввода-вывода, вторая - номер разъема шины в блоке CPU, к которому подключен асинхронный адаптер блока расширения ввода-вывода.
BB	Первая цифра поля **BB** задает шину ввода-вывода карты адаптера. Если карта находится в блоке CPU или в системном блоке, то для стандартной шины ввода-вывода эта цифра будет равна `0`, а для дополнительной - `1`. Если карта адаптера расположена в блоке расширения ввода-вывода, то эта цифра будет равна `0`. Вторая цифра задает номер разъема шины ввода-вывода (или номер разъема блока расширения ввода-вывода), к которой подключен адаптер.
CC	Поле **CC** задает разъем карты адаптера, к которому подключен асинхронный блок распределения. Допустимы значения `01`, `02`, `03` и `04`.
DD	Поле **DD** задает номер порта асинхронного блока распределения, к которому подключен принтер, графопостроитель или терминал.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Коды расположения терминалов

Коды расположения `00-00-S1-00` и `00-00-S2-00` задают терминал, подключенный к стандартному последовательному порту ввода-вывода `s1` или `s2`.

Все остальные коды расположения задают терминал, подключенный к карте адаптера, отличной от стандартной платы ввода-вывода. Код расположения таких устройств указывается в формате *AA-BB-CC-DD*, где *AA-BB* задает код расположения управляющего адаптера.

Элемент	Описание
AA	Значение `00` поля **AA** соответствует карте адаптера, расположенной в блоке CPU или в системном блоке, в зависимости от типа устройства. Если в поле **AA** указано другое значение, то карта адаптера расположена в блоке расширения ввода-вывода. В этом случае первая цифра задает шину ввода-вывода, а вторая - номер разъема шины в блоке CPU, к которому подключен асинхронный адаптер блока расширения ввода-вывода.
BB	Первая цифра поля **BB** задает шину ввода-вывода карты адаптера. Если карта расположена в блоке CPU или в системном блоке, то для стандартной шины ввода-вывода эта цифра будет равна `0`, а для дополнительной - `1`. Если карта адаптера расположена в блоке расширения ввода-вывода, то эта цифра будет равна `0`. Вторая цифра задает номер разъема шины ввода-вывода (или номер разъема блока расширения ввода-вывода), к которой подключен адаптер.
CC	Поле **CC** задает разъем карты адаптера, к которому подключен асинхронный блок распределения. Допустимы значения `01`, `02`, `03` и `04`.
DD	Поле **DD** задает номер порта асинхронного блока распределения, к которому подключен асинхронный терминал.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Коды расположения дисков SCSI

Ниже приведены коды расположений устройств SCSI.

Эти коды расположения относятся ко всем устройствам SCSI, включая:

- CD-ROM
- Дисковые накопители
- Устройства-инициаторы
- Перезаписываемые оптические устройства
- Накопители на магнитной ленте
- Целевые устройства

Для таких устройств код расположения задается в формате `AA-BB-CC-S,L`. Поля **AA**-**BB** задают код расположения контроллера SCSI.

Элемент	Описание
AA	Значение `00` поля **AA** соответствует карте управляющего адаптера, расположенной в блоке CPU или в системном блоке, в зависимости от типа устройства.
BB	Поле **BB** задает шину ввода-вывода и разъем карты адаптера. Первая цифра задает шину ввода-вывода. Стандартной шине ввода-вывода соответствует значение `0`, а дополнительной - `1`. Вторая цифра задает разъем шины ввода-вывода карты адаптера. Для стандартного контроллера SCSI значение поля `BB` равно **00**.
CC	Поле **CC** задает шину адаптера SCSI, к которому подключено устройство. Если в адаптере предусмотрена только одна шина SCSI, то значение этого поля равно `00`. В противном случае значение `00` задает устройство, подключенное к внутренней шине SCSI карты адаптера, а значение `01` задает устройство, подключенное к внешней шине SCSI карты адаптера.
S,L	Поле **S,L** задает ИД SCSI и номер логического устройства (LUN), связанные с устройством SCSI. Значение `S` задает идентификатор SCSI, а значение `L` задает LUN.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Коды расположения Dials/LPFKeys

Код расположения дополнительной клавиатуры или клавиатуры LPF, подключенной к адаптеру графического ввода, задается в формате *AA-BB-CC*.

Значения отдельных полей интерпретируются следующим образом:

Элемент	Описание
AA	Значение `00` поля **AA** соответствует карте управляющего адаптера, расположенной в блоке CPU или в системном блоке, в зависимости от типа устройства.
BB	Поле **BB** задает шину ввода-вывода и разъем карты адаптера. Первая цифра задает шину ввода-вывода. Стандартной шине ввода-вывода соответствует значение `0`, а дополнительной - `1`. Вторая цифра задает номер разъема указанной шины ввода-вывода, к которой подключен адаптер.
CC	Поле **CC** задает разъем адаптера, к которому подключено устройство. Если устройство подключено к порту `1` карты адаптера, то значение поля равно `01`, а если к порту `2` - то `02`.

Примечание: Для устройств ввода и клавиатур LPF, подключенных по последовательной линии связи, код расположения не указывается. Предполагается, что устройство подключено к асинхронному терминалу. Этот асинхронный терминал задается пользователем в определении устройства ввода/клавиатуры LPF.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Коды расположений многопротокольного порта

Код расположения многопротокольного порта задается в формате *AA-BB-CC-DD*, где *AA-BB* - код расположения многопротокольной карты адаптера.

Значения отдельных полей интерпретируются следующим образом:

Элемент	Описание
AA	Значение `00` поля **AA** соответствует карте многопротокольного адаптера, расположенной в блоке CPU или в системном блоке, в зависимости от типа устройства.
BB	Поле **BB** задает шину ввода-вывода и разъем карты адаптера. Первая цифра задает шину ввода-вывода. Стандартной шине ввода-вывода соответствует значение `0`, а дополнительной - `1`. Вторая цифра задает номер разъема указанной шины ввода-вывода, к которой подключен адаптер.
CC	Поле **CC** задает разъем карты адаптера, к которому подключен многопротокольный блок распределения. Значение этого поля всегда равно `01`.
DD	Поле **DD** задает номер физического порта многопротокольного блока распределения. Допустимые значения - `00`, `01`, `02` и `03`.

Информация, связанная с данной:

Расположения и коды расположений компонентов

Настройка iSCSI

Настройка iSCSI подразумевает настройку адаптера и добавление или обновление адресатов.

Настройка адаптера iSCSI в AIX

Настройка адаптера iSCSI является простой задачей.

1. Введите **smit iscsi** в командной строке AIX. Появится окно iSCSI.
2. Выберите в окне iSCSI пункт **Адаптер iSCSI**. Появится окно Адаптер iSCSI.
3. Выберите пункт **Изменить / Показать параметры адаптера iSCSI** в окне адаптера iSCSI. Появится окно изменения/просмотра параметров адаптера iSCSI.
4. Выберите из списка настраиваемый адаптер iSCSI. Появится окно настройки, схожее с приведенным в следующем примере.

```
                                                [Поля ввода]

  Адаптер iSCSI                                 ics0

  Описание                                      Адаптер iSCSI

  Состояние                                     Доступен

  Расположение                                  10-60

  Макс. число команд в очереди адаптера        [200]      +                    +#

  Макс. размер передачи                         [0x100000]     +

  Файл поиска                                   [/etc/iscsi/targetshw]
  Стратегия поиска                              [file]
  Файл автоматического поиска секретов          [/etc/iscsi/autosecret]
  IP-адрес адаптера                             [10.1.4.187]
  Маска подсети адаптера                        [255.255.255.0]

  Адрес шлюза адаптера                          [10.1.4.1]

  Применить изменения только к базе данных       нет       +
```

Примечание: При возникновении вопросов, связанных с предназначением определенного поля, наведите курсор на поле и нажмите клавишу **F1** для получения справки.

Для применения поиска простого файла введите `file` в поле **Стратегия поиска**. Для применения поиска ODM введите `odm` в поле **Стратегия поиска**. Для применения поиска DHCP адресатов iSCSI введите `slp` в поле **Стратегия поиска**.

Обновление простого файла целевого назначения iSCSI

Простой файл является статическим файлом конфигурации, используемым для целевых адресатов iSCSI. По умолчанию он имеет имя `/etc/iscsi/targetshw`.

Необходимо явно указать в этом файле все соответствующие свойства поиска целевых адресатов iSCSI.

Информация, связанная с данной:

Файл targets

Добавление статически обнаруживаемых адресатов iSCSI в ODM

Если автоматическое обнаружение отключено, адаптер iSCSI получает описания адресатов iSCSI из простого фа йла или ODM.

Для управления данными адресата iSCSI в ODM можно использовать команды AIX или SMIT. С помощью команд **chiscsi**, **lsiscsi**, **mkiscsi** и **rmiscsi** можно изменить, просмотреть, доабвить и удалить сведения адресата iSCSI из ODM.

Для добавления одного статически обнаруживаемого адресата iSCSI в ODM с помощью SMIT выполните следующие действия:

1. Введите **smit iscsi** в командной строке AIX. Появится окно iSCSI.

2. Выберите **Параметры целевого устройства iSCSI в ODM** в окне iSCSI. В окне ODM появятся параметры целевого устройства iSCSI.
3. Выберите в окне iSCSI пункт **Добавить целевое устройство iSCSI в ODM**. Появится окно добавления целевого устройства iSCSI в ODM.
4. Выберите в окне iSCSI пункт **Добавить целевое устройство iSCSI в ODM**. Появится окно добавления целевого устройства iSCSI в ODM.
5. Выберите из списка настраиваемый адаптер iSCSI. Появится окно добавления статически обнаруживаемого целевого устройства iSCSI для выбранного адаптера.
6. Введите в соответствующих полях нужные данные. Ниже приведен пример.

```
                                                [Поля ввода]

Адаптер iSCSI                                    ics0

Имя адресата iSCSI                               [iqn.mds9216.emc.sym]      +

Группа iSCSI                                      static      +

IP-адрес адресата iSCSI                         [10.1.4.25]      +

Номер порта адресата iSCSI                        [3260]     +

Пароль                                          [пароль]      +
```

Примечание: При возникновении вопросов, связанных с предназначением определенного поля, наведите курсор на поле и нажмите клавишу **F1** для получения справки.

Добавление статически обнаруживаемых адресатов iSCSI из простого файла в ODM

С помощью SMIT можно импортировать данные из простого файла в ODM.

1. Введите **smit iscsi** в командной строке AIX. Появится окно iSCSI.
2. Выберите **Параметры целевого устройства iSCSI в ODM** в окне iSCSI. В окне ODM появятся параметры целевого устройства iSCSI.
3. Выберите в окне iSCSI пункт **Добавить целевое устройство iSCSI в ODM**. Появится окно добавления целевого устройства iSCSI в ODM.
4. Выберите пункт **Добавить данные целевого устройства iSCSI в ODM из файла**. Появится окно добавления данных целевого устройства iSCSI в ODM из файла.
5. Выберите из списка настраиваемый адаптер iSCSI. Появится окно добавления данны целевого устройства iSCSI в ODM из файла для выбранного адаптера iSCSI.
6. Введите в соответствующих полях нужные данные. Ниже приведен пример.

```
                                                [Поля ввода]

Устройство протокола iSCSI                        iscsi3

Группа iSCSI                                    [static]      +

Имя файла адресата iSCSI                      [/etc/iscsi/targetshw]      +
```

Примечание: При возникновении вопросов, связанных с предназначением определенного поля, наведите курсор на поле и нажмите клавишу **F1** для получения справки.

Управление оперативной заменой PCI

Вы можете установить новый адаптер PCI с возможностью оперативной замены в разъем PCI во время работы системы.

Этот адаптер может быть установлен вместо адаптера того же или другого типа. Для того чтобы новые ресурсы стали доступными для операционной системы и приложений, не требуется перезапускать операционную систему. Ниже перечислены некоторые из возможных причин установки адаптеров:

- Расширение функциональных возможностей или ресурсов существующего аппаратного или встроенного программного обеспечения.
- Перенос адаптеров PCI из системы, в которой эти адаптеры больше не нужны.
- Установка новой системы, в которой карты адаптеров (в том числе адаптеров PCI) устанавливаются после начальной настройки вспомогательных аппаратных подсистем, а также установки и запуска операционной системы.

Примечание: После добавления адаптера путем оперативной замены или добавления адаптера PCI или путем динамического распределения ресурсов между разделами этот адаптер и подключенные к нему дочерние устройства будут недоступны в качестве загрузочных устройств, поддерживаемых командой **bootlist**. Для того чтобы добавить их в список загрузочных устройств, необходимо перезапустить систему. Уже указанный в списке загрузочных устройств адаптер, который впоследствии будет заменен на правильный тип адаптера, по-прежнему является допустимым загрузочным устройством.

Вы можете заменить неисправный адаптер PCI с поддержкой оперативной замены, установленный в блоке ввода-вывода или в основном блоке системы, без завершения работы операционной системы или отключения питания компьютера. После замены адаптера будет применяться старый драйвер, поскольку тип устройства не изменился. Информация о конфигурации устройств, подключенных к заменяемому адаптеру, используется при настройке нового адаптера. Ниже перечислены некоторые их возможных причин замены адаптеров:

- Временная замена карты для определения причины неполадки и поиска неисправного FRU.
- Замена неисправного или ненадежно работающего адаптера на исправный.
- Замена неисправного резервного адаптера в конфигурации HACMP.

После удаления адаптера предоставляемые им ресурсы станут недоступными для операционной системы и приложений. Ниже перечислены некоторые их возможных причин удаления адаптеров:

- Удаление существующих подсистем ввода-вывода.
- Удаление адаптера, который больше не нужен, а также удаление неисправного адаптера при отсутствии нового адаптера для замены.
- Перенос адаптера в другую систему из системы, в которой он больше не нужен.

Перед удалением или заменой устройства с поддержкой оперативной замены необходимо удалить конфигурацию этого устройства. Драйвер устройства должен освободить все выделенные устройству системные ресурсы. При этом освобождается выделенная память, удаляется определение обработчиков прерываний и EPOW, освобождается канал DMA и ресурсы таймера и выполняются другие необходимые операции. Кроме того, драйвер устройства должен проверить, что отключены прерывания устройства, средства работы с памятью шины и каналом ввода-вывода шины.

Перед удалением и после установки адаптера системный администратор должен выполнить следующие задачи:

- Прервать, а затем возобновить работу приложений, демонов и процессов, использующих устройство.
- Размонтировать и вновь смонтировать файловые системы.
- Удалить и вновь создать определения устройств и выполнить все необходимые операции, связанные с освобождением используемого устройства.
- Перевести систему в состояние обслуживания.
- Получить и установить все необходимые драйверы устройств.

Перед удалением или заменой устройства необходимо удалить устройство из конфигурации системы и убедиться, что оно находится в состоянии Определено. Сделать это можно с помощью команды **rmdev**. Перед переключением адаптера в состояние Определено необходимо закрыть все приложения, работающие с адаптером, поскольку в противном случае операция выполнена не будет. Дополнительная информация о команде **rmdev** приведена в разделе **rmdev**.

В некоторых случаях вам может потребоваться выполнить следующие задачи:

- Подготовить адаптер PCI с поддержкой оперативной замены к установке, удалению или замене.
- Определить разъемы и адаптеры PCI, участвующие в операции оперативной замены.
- Удалить или установить адаптеры PCI с поддержкой оперативной замены.

Примечание: Во время операций оперативной замены Администратор объектных данных (ODM) остается заблокированным. Это может привести к зависанию или сбою других задач, которым требуется ODM. Может произойти зависание или сбой изменений конфигурации на уровне кластера, инициируемые другими узлами кластера. Поэтому убедитесь, что такие задачи не выполняются, пока не будет завершена операция оперативной замены.
Внимание: Средства управления оперативной заменой адаптеров PCI позволяют добавлять, удалять и заменять адаптеры PCI без выключения компьютера или перезапуска операционной системы, однако эти средства нельзя применять для работы с некоторыми устройствами, установленными в разъемах с поддержкой оперативной замены. Например, без выключения системы нельзя удалить или заменить жесткий диск, входящий в группу томов rootvg, а также адаптер ввода-вывода, к которому подключен этот диск. Это связано с тем, что данные устройства необходимы для работы операционной системы. Если для группы томов rootvg включена зеркальная защита, то с помощью команды **chpv** диски можно деактивизировать. Если группа томов rootvg располагается на дисках MPIO, подключенных к нескольким контроллерам ввода-вывода, то один из этих контроллеров можно удалить (или заменить), не перезагружая систему. В этом случае все пути от удаляемого (или заменяемого) контроллера ввода-вывода необходимо удалить из конфигурации, вызвав команду **rmdev -R** для адаптера. Эта команда удалит из конфигурации пути и адаптер. После этого можно приступить к работе с Управлением оперативной заменой. Перед удалением или добавлением адаптера PCI убедитесь, что этот адаптер поддерживает оперативную замену. Необходимая информация приведена в книге *Справочник по установке адаптеров PCI*, которая поставляется вместе с компонентами системы, поддерживающими оперативную замену. Инструкции по установке и удалению адаптеров приведены в документации по системному блоку.

Понятия, связанные с данным:

“Удаление адаптера связи” на стр. 549
Перед удалением или заменой адаптер с поддержкой оперативной замены необходимо отключить.

Задачи, связанные с данной:

“Удаление настройки адаптеров памяти” на стр. 555
Перед удалением или заменой адаптер памяти необходимо отключить.

“Удаление конфигурации асинхронного абзаца” на стр. 556
Можно удалить конфигурацию асинхронного адаптера.

Просмотр сведений о разъеме оперативной замены PCI

Перед добавлением, удалением или заменой адаптера с возможностью оперативной замены можно просмотреть информацию об имеющихся в системе разъемах PCI с поддержкой оперативной замены.

Можно просмотреть следующую информацию:

- Список всех имеющихся в системе разъемов PCI с поддержкой оперативной замены
- Свободен разъем или занят
- Список занятых разъемов
- Характеристики разъема, такие как его имя, описание, тип соединения и имя подключенного устройства

Кроме того, это можно сделать с помощью SMIT или команд системы. Для выполнения этих задач необходимо войти в систему как пользователь root.

Процедура быстрого доступа SMIT

1. Введите в командной строке `smit devdrpci` и нажмите Enter.
2. Выполните задачи с помощью программы SMIT.

Дополнительную информацию по выполнению этих задач можно получить, нажав клавишу справки F1 в окне диалога SMIT.

Процедура команд

Следующие команды позволяют просмотреть информацию о разъемах с поддержкой оперативной замены и подключенных устройствах:

- Команда lsslot выводит список всех разъемов PCI с поддержкой оперативной замены с указанием их характеристик.
- Команда **lsdev** показывает текущее состояние всех устройств, установленных в системе.

Удаление конфигурации адаптеров связи PCI

Ниже приведен обзор процедуры удаления адаптера связи PCI из конфигурации системы. Приведенная информация относится к адаптерам Ethernet, Token-ring, FDDI и ATM.

Если ваше приложение использует протокол TCP/IP, то перед переключением адаптера в состояние Определен необходимо удалить интерфейс TCP/IP этого адаптера из списка сетевых интерфейсов. Для того чтобы узнать, настроен ли адаптер для работы с TCP/IP и активен ли сетевой интерфейс этого адаптера, вызовите команду **netstat**. Информация о команде **netstat** приведена в разделе **netstat**.

Для адаптера Ethernet может быть создано два интерфейса: стандартный Ethernet (en *X*) и IEEE 802.3 (et*X*). *X* совпадает с номером, указанным в имени адаптера `ent`*X*. С TCP/IP может работать только один из этих интерфейсов. Например, с адаптером Ethernet `ent0` могут быть связаны интерфейсы en0 и et0.

Адаптеру Token-Ring может соответствовать только один интерфейс Token-ring (tr*X*). *X* совпадает с номером, указанным в имени адаптера `tok`*X*. Например, адаптеру Token-ring `tok0` может соответствовать интерфейс tr0.

Адаптеру ATM может соответствовать только один интерфейс ATM (at*X*). *X* совпадает с номером, указанным в имени адаптера `atm`*X*. Например, адаптеру ATM `atm0` может соответствовать интерфейс at0. Тем не менее, с одним адаптером ATM может работать несколько эмулируемых клиентов.

Для удаления интерфейса из сети служит команда **ifconfig**. Команда **rmdev** удаляет устройство PCI из конфигурации, оставляя при этом определение устройства в классе настроенных устройств. После того как адаптер будет переключен в состояние Определен, вы можете удалить его с помощью команды **drslot**.

Для того чтобы удалить все дочерние устройства шины PCI `pci1` и все их дочерние устройства, сохранив в объектном классе Настроенные устройства их определения, введите:

```
rmdev  -p pci1
```

Будет показана примерно следующее сообщение:

```
rmt0 Определено
hdisk1 Определено
scsi1 Определено
ent0 Определено
```

Понятия, связанные с данным:

“Удаление адаптера связи” на стр. 549
Перед удалением или заменой адаптер с поддержкой оперативной замены необходимо отключить.

Удаление или замена адаптера PCI оперативной замены

Удаление или замену адаптера PCI оперативной замены можно выполнить, не завершая работу операционной системы и не отключая питание системы. После удаления адаптера его ресурсы станут недоступны для операционной системы и приложений.

Перед удалением адаптера его необходимо удалить из конфигурации.

Ниже описаны процедуры удаления адаптера PCI оперативной замены. Теперь эти задачи можно выполнить с помощью SMIT или команд системы. Для выполнения этих задач необходимо войти в систему как пользователь root.

При замене адаптера на другой адаптер того же типа информация о конфигурации заменяемого адаптера сохраняется и сравнивается с конфигурацией нового адаптера. Существующий драйвер устройства заменяемого адаптера должен поддерживать новый адаптер.

Процедура быстрого доступа SMIT

1. Введите в командной строке `smit devdrpci` и нажмите Enter.
2. Выполните задачи с помощью программы SMIT.

Дополнительную информацию по выполнению этих задач можно получить, нажав клавишу справки F1 в окне диалога SMIT.

Процедура команд

Следующие команды позволяют просмотреть информацию о разъемах и устройствах с поддержкой оперативной замены, а также адаптер PCI с поддержкой оперативной замены:

- Команда **lsslot** выводит список всех разъемов PCI с поддержкой оперативной замены с указанием их характеристик. Дополнительная информация приведена в описании команды **lsslot** в книге *Справочник по командам, том 3*.
- Команда **lsdev** показывает текущее состояние всех устройств, установленных в системе. Дополнительная информация приведена в описании команды **lsdev** в книге *Справочник по командам, том 3*.
- Команда **drslot** подготавливает разъем к оперативному удалению адаптера. Дополнительная информация приведена в описании команды **drslot** в книге *Справочник по командам, том 2*.

Понятия, связанные с данным:

“Удаление адаптера связи” на стр. 549
Перед удалением или заменой адаптер с поддержкой оперативной замены необходимо отключить.

Задачи, связанные с данной:

“Удаление настройки адаптеров памяти” на стр. 555
Перед удалением или заменой адаптер памяти необходимо отключить.

“Удаление конфигурации асинхронного абзаца” на стр. 556
Можно удалить конфигурацию асинхронного адаптера.

Добавление адаптера PCI оперативной замены

Адаптер PCI с поддержкой оперативной замены можно установить в свободный разъем системного блока, предоставив его ресурсы операционной системе и приложениям без перезапуска операционной системы. Тип устанавливаемого адаптера может как совпадать, так и отличаться от типа установленного адаптера.

Ниже описана процедура добавления нового адаптера PCI оперативной замены.

Внимание: Перед оперативным добавлением адаптеров PCI ознакомьтесь с содержанием книги *Справочник по установке адаптеров PCI*, чтобы определить, поддерживает ли адаптер оперативную замену. Инструкции по установке и удалению адаптеров приведены в документации по системному блоку.

Для оперативного добавления нового адаптера PCI необходимо выполнить следующие задачи:

- Поиск и идентификация свободного разъема в системе
- Подготовка разъема к настройке адаптера
- Установка драйвера устройства, если это необходимо
- Настройка нового адаптера

Кроме того, это можно сделать с помощью SMIT или команд системы. Для выполнения этих задач необходимо войти в систему как пользователь root.

Примечание: При оперативном добавлении адаптера в систему этот адаптер и подключенные к нему устройства нельзя будет указать в качестве загрузочного устройства с помощью команды **bootlist**. Для обнаружения в системе всех возможных загрузочных устройств может потребоваться перезапуск операционной системы.

Процедура быстрого доступа SMIT

1. Введите в командной строке `smit devdrpci` и нажмите Enter.
2. Выполните задачи с помощью программы SMIT.

Дополнительную информацию по выполнению этих задач можно получить, нажав клавишу справки F1 в окне диалога SMIT.

Процедура команд

Следующие команды позволяют просмотреть информацию о разъемах и устройствах PCI с поддержкой оперативной замены, а также оперативно добавить адаптер PCI:

- Команда **lsslot** выводит список всех разъемов с поддержкой оперативной замены с указанием их характеристик. Дополнительная информация приведена в описании команды **lsslot** в книге *Справочник по командам, том 3*.
- Команда **lsdev** показывает текущее состояние всех устройств, установленных в системе. Дополнительная информация приведена в описании команды **lsdev** в книге *Справочник по командам, том 3*.
- Команда **drslot** подготавливает разъем к оперативному добавлению или удалению адаптера. Дополнительная информация приведена в описании команды **drslot** в книге *Справочник по командам, том 2*.

Разветвленный ввод-вывод

Функция разветвленного ввода-вывода (MPIO) позволяет однозначно диагностировать устройство через одно или несколько физических соединений, называемых также *путями*.

За управление путями отвечает модуль управления путями (PCM).

Драйвер устройства MPIO может управлять целевыми устройствами нескольких типов. PCM может поддерживать несколько конкретных устройств. Таким образом, один драйвер устройства может взаимодействовать с несколькими PCM, которые управляют вводом-выводом на целевые устройства по нескольким путям.

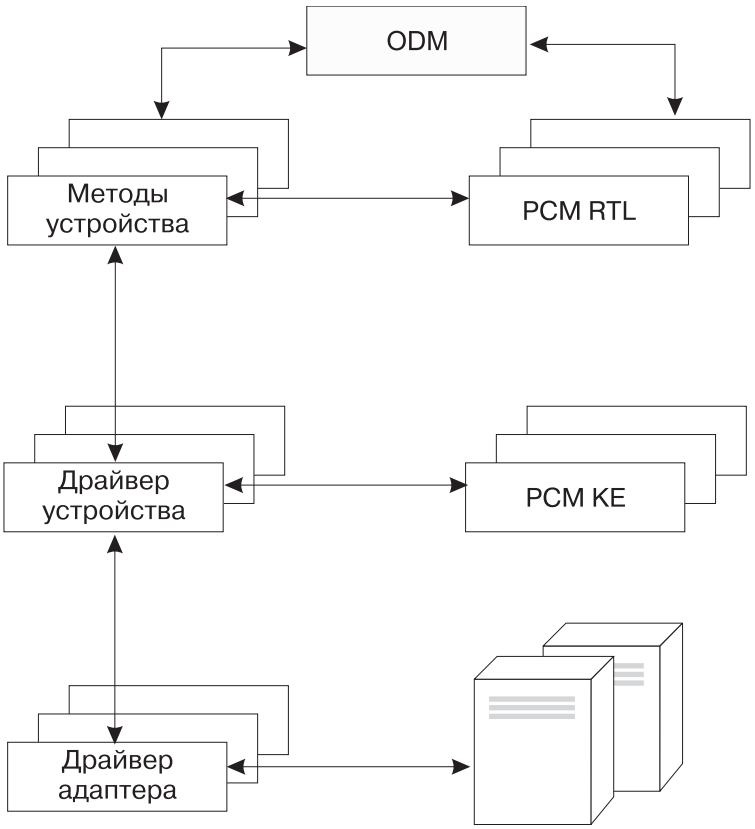


Рисунок 12. Взаимодействие компонентов MPIO. На рисунке показаны принципы взаимодействия между различными компонентами MPIO. Драйвер устройства MPIO управляет несколькими типами целевых устройств, для каждого из которых требуется отдельный PCM. (KE=расширение ядра, RTL=загружаемый модуль).

Перед началом работы с MPIO необходимо соответствующим образом настроить драйвер, методы и предопределенные атрибуты устройства в Администраторе объектных данных (ODM), чтобы обеспечить поддержку определения, настройки и управления устройствами MPIO. Драйверы дисков параллельного SCSI и Fibre Channel, а также соответствующие методы поддерживают диски MPIO. Диски iSCSI поддерживаются как устройства MPIO. Драйвер накопителя на магнитной ленте Fibre Channel и его методы поддерживают лентопротяжные устройства MPIO. Кроме того, для поддержки MPIO были изменены предопределенные атрибуты некоторых устройств в базе данных ODM.

AIX PCM состоит из модуля настройки PCM RTL и расширения ядра PCM KE. PCM RTL - это загружаемый во время выполнения модуль, который содержит методы определения дополнительных атрибутов PCM KE устройства и ODM пути, необходимые PCM KE. PCM RTL загружается с помощью специального метода. Затем происходит обращение к одной или нескольким процедурам в PCM RTL, которые выполняют операции по инициализации или изменению переменных PM KE.

PCM KE обеспечивает функции управления путями для всех драйверов устройств, поддерживающих интерфейс MPIO. Способность PCM KE обнаружить путь и передать информацию о нем драйверу устройства зависит от конфигурации устройства. Каждый драйвер устройства с поддержкой MPIO добавляет путь к устройству от драйверов более высокого уровня. Обслуживание и распределение ввода-вывода по различным путям выполняется PCM KE и не зависит от драйвера устройства MPIO.

PCM KE может обеспечивать несколько алгоритмов маршрутизации, выбираемых пользователем. Кроме того, PCM KE позволяет собрать информацию, применяемую при определении и выборе наилучшего пути для запроса ввода-вывода. PCM KE может выбирать наилучший путь на основании нескольких критериев, включая баланс загрузки, скорость соединения, число ошибок соединения и т.п.

В операционной системе AIX PCM обладает дополнительными возможностями по проверке работоспособности, которые могут применяться в следующих целях:

- Проверка путей и определение доступного пути для операции ввода-вывода.
- Включение пути, который ранее был отмечен как поврежденный из-за временной ошибки (например, временного отключения кабеля от устройства).
- Проверка неиспользуемых путей, которые могут применяться при необходимости отката (например, если атрибут алгоритма равен `failover`, функция проверки работоспособности может проверить альтернативные пути).

Не все диски можно определить и настроить с помощью PCM, предусмотренных в AIX по умолчанию. PCM, применяемые в AIX по умолчанию, состоят из двух модулей управления путями, один из которых предназначен для работы с дисковыми накопителями, а другой - с накопителями на магнитной ленте. Если установленное у вас устройство определить не удалось, то свяжитесь с его производителем и узнайте, доступно ли для него PCM.

Ссылки, связанные с данной:

“Атрибуты прочих типов накопителей на магнитной ленте SCSI (тип ost)” на стр. 573
Ниже описаны атрибуты прочих накопителей SCSI на магнитной ленте (тип ost).

Управление совместимыми с MPIO устройствами

Технология разветвленного ввода-вывода (MPIO) позволяет определить альтернативные пути к устройству для восстановления после сбоя.

Восстановление после сбоя - это алгоритм управления путями доступа к устройствам, повышающий степень надежности и готовности устройств за счет автоматического обнаружения сбоев путей ввода-вывода и перенаправления операций ввода-вывода на альтернативные пути. Все диски SCSI SCSD автоматически настраиваются как устройства MPIO. Некоторые диски Fibre Channel можно настроить как Прочие диски MPIO. Могут поддерживаться и другие устройства, если их драйвер совместим с реализацией MPIO, предусмотренной в AIX.

MPIO устанавливается и настраивается при установке BOS. Никакая специальная настройка не требуется, однако вы можете добавлять, удалять, перенастраивать, а также включать и отключать устройства и пути доступа к ним с помощью SMIT или интерфейса командной строки. Управлять путями доступа MPIO можно с помощью следующих команд:

mkpath
: Добавляет путь доступа к устройству.

rmpath
: Удаляет путь доступа к устройству.

chpath Изменяет атрибут или состояние пути доступа к устройству.

lsmpio Показывает информацию об устройствах хранения с поддержкой разветвленного ввода-вывода (MPIO), в том числе состояние, конфигурацию и статистику.

lspath Показывает информацию о пути доступа к устройству.

Понятия, связанные с данным:

“База данных конфигураций устройств и управление устройствами” на стр. 527
Информация об устройствах содержится в базах данных описаний и настройки, которые составляют базу данных конфигурации устройств.

“Настройка устройства MPIO” на стр. 542
При настройке устройства MPIO применяются те же команды, что и при настройке обычных устройств.

Подключение устройства SCSI к устройству MPIO:

Если устройство SCSI настроено в качестве устройства MPIO, то его могут поддерживать не более двух адаптеров.

При подключении параллельного устройства SCSI в качестве устройства MPIO воспользуйтесь приведенным ниже примером. Здесь описан минимальный набор необходимых настроек; для подключения вашего устройства могут потребоваться и другие операции.

1. Выключите питание и установите два адаптера SCSI.
2. Подключите устройство к обоим адаптерам SCSI.
3. Включите питание системы.
4. Измените ИД SCSI на одном из адаптеров. По умолчанию ИД адаптера SCSI ID равен 7. Поскольку ИД адаптеров SCSI не должны повторяться, укажите для одного из адаптеров другое значение, например, 6.
5. Запустите команду **cfgmgr**.
6. Для проверки конфигурации введите следующую команду:

   ```
   lspath -l hdiskX
   ```

   где *X* - логический номер нового устройства. Команда должна показать два пути и их состояние.

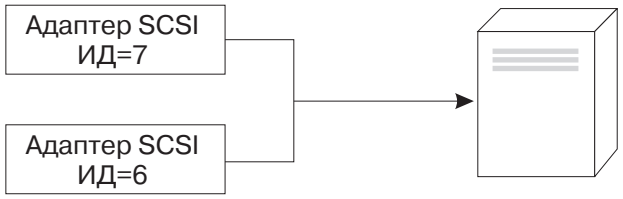


Рисунок 13. Подключение устройства MPIO SCSI

На этом рисунка показано подключение двух адаптеров SCSI к одному устройству.

Подключение устройства Fibre Channel в качестве устройства MPIO:

Устройство Fibre Channel можно подключить к нескольким адаптерам. Программные ограничения для них не установлены.

При подключении устройства Fibre Channel в качестве устройства MPIO воспользуйтесь приведенным ниже примером. Здесь описан минимальный набор необходимых настроек; для подключения вашего устройства могут потребоваться и другие операции.

1. Выключите питание и установите два адаптера Fibre Channel.
2. Подключите адаптеры к коммутатору или концентратору.
3. Подключите устройство к коммутатору или концентратору.
4. Включите питание системы.
5. Для проверки конфигурации введите следующую команду:

   ```
   lspath -l hdiskX
   ```

   где *X* - логический номер нового устройства. Будет показано по одному пути для каждого установленного адаптера, а также состояние этих путей.

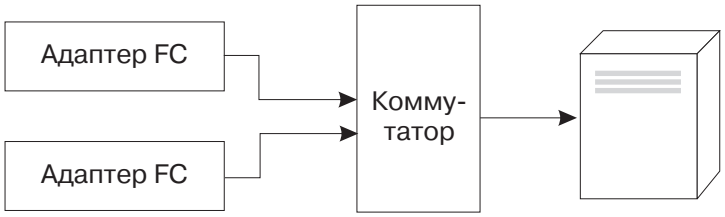


Рисунок 14. Подключение устройства MPIO Fibre Channel. На рисунке показано подключение двух адаптеров Fibre Channel к коммутатору, который подключен к устройству.

Настройка устройства MPIO

При настройке устройства MPIO применяются те же команды, что и при настройке обычных устройств.

Команды **cfgmgr**, **mkdev**, **chdev**, **rmdev** и **lsdev** поддерживают управление экземплярами устройства MPIO и просмотр их атрибутов. С каждым экземпляром устройства MPIO связан экземпляр пути. С помощью команд **mkpath**, **chpath**, **rmpath** и **lspath** можно управлять экземплярами путей и просматривать их атрибуты.

Экземпляр пути можно добавить или удалить из устройства MPIO, не отключая это устройство.

У устройства MPIO могут быть и дополнительные атрибуты, однако все устройства MPIO должны поддерживать атрибуты **reserve_policy** и **algorithm**. Атрибут **reserve_policy** задает способ резервирования, который будет применяться драйвером при открытии устройства, и может применяться для ограничения доступа к устройству со стороны других адаптеров этой или другой системы. Атрибут **algorithm** задает способ управления вводом-выводом по путям, настроенным для устройства.

Понятия, связанные с данным:

“Управление совместимыми с MPIO устройствами” на стр. 540
Технология разветвленного ввода-вывода (MPIO) позволяет определить альтернативные пути к устройству для восстановления после сбоя.

“Атрибуты устройства MPIO” на стр. 543
Следующие атрибуты поддерживаются только разветвленными устройствами. Атрибуты можно просматривать и изменять с помощью SMIT или команд (в частности, команд **lsattr** и **chdev**).

“Атрибуты модуля управления путем” на стр. 545
В дополнение к PCM, предусмотренным в AIX по умолчанию, производитель может предоставить PCM для конкретного устройства. Набор атрибутов, которые может изменять пользователь, определяется производителем устройства. У такого PCM могут быть атрибуты устройства и атрибуты пути.

Поддерживаемые альтернативные устройства

PCM, предусмотренные в AIX по умолчанию, поддерживают те устройства, которые определены в наборе файлов `devices.common.IBM.mpio.rte`.

Для того чтобы устройства, не поддерживаемые различными PCM AIX, идентифицировались как поддерживающие MPIO, необходимо, чтобы производитель устройства предоставил атрибуты, определенные в ODM, PCM, или другом поддерживаемом коде.

Для получения списка устройств, поддерживаемых AIX PCM, запустите следующий сценарий:

```
odmget -qDvDr=aixdiskpcmke PdDv | grep uniquetype | while read line
do
      utype=`echo $line | cut -d'"' -f2`
      dvc=`odmget -q"uniquetype=$utype AND attribute=dvc_support" PdAt`
      echo $dvc | grep values | cut -d'"' -f2
done
```

Для получения списка накопителей на магнитной ленте, поддерживаемых AIX PCM, запустите следующий сценарий:

```
odmget -qDvDr=aixtapepcmke PdDv | grep uniquetype | while read line
do
      utype=`echo $line | cut -d'"' -f2`
      dvc=`odmget -q"uniquetype=$utype AND attribute=dvc_support" PdAt`
      echo $dvc | grep values | cut -d'"' -f2
done
```

Вывод сценария будет содержать список типов устройств, которые поддерживаются PCM по умолчанию. PCM дисков AIX поддерживает устройства трех типов: *самонастраивающийся параллельный диск SCSI* (`disk/scsi/scsd`), *другой диск MPIO FC*(`disk/fcp/mpioosdisk`) и *другой MPIO iSCI* (`disk/iscsi/mpioosdisk`). PCM магнитных лент AIX поддерживает *другие магнитные ленты MPIO FC* (`tape/fcp/mpioost`).

Другие диски MPIO FC и *другие магнитные ленты MPIO FC* являются подмножеством других дисков и магнитных лент Fibre Channel, соответственно. Устройство распознается как *другой диск FC MPIO*, если у него нет определенных производителем атрибутов ODM, и оно сертифицировано для работы с PCM AIX по умолчанию. Сертификация устройства не гарантирует, что будут поддерживаться все возможности устройства, если это устройство было настроено как *другой диск FC MPIO*.

Другие диски MPIO iSCSI являются подмножеством других дисков iSCSI. Устройство распознается как *другой диск MPIO iSCSI*, если у него нет определенных производителем атрибутов ODM, и оно сертифицировано для работы с PCM AIX по умолчанию. Сертификация устройства не гарантирует, что будут поддерживаться все возможности устройства, если это устройство было настроено как *другой диск FC iSCSI*.

Для того чтобы определить устройства, которые поддерживаются как *другие диски MPIO FC*, запустите следующий сценарий:

```
odmget -quniquetype=disk/fcp/mpioosdisk PdAt | grep deflt | cut -d'"' -f2
```

Для определения того, какие устройства поддерживаются в качестве *других магнитных лент FC MPIO*, запустите следующий сценарий:

```
odmget -q "uniquetype=tape/fcp/mpioosdisk AND attribute=mpio_model_map PdAt | grep deflt | cut -d'"' -f2
```

Для того чтобы определить устройства, которые поддерживаются как *другие диски MPIO iSCSI*, запустите следующий сценарий:

```
odmget -quniquetype=disk/iscsi/mpioosdisk PdAt | grep deflt | cut -d'"' -f2
```

Будет показан список производителей и моделей устройств.

Для просмотра списка всех установленных в системе устройств MPIO введите:

```
odmget -qattribute=unique_id PdAt | grep uniquetype | cut -d'"' -f2
```

Будет показан список уникальных типов устройств MPIO, поддерживаемых PCM AIX по умолчанию и другими PCM.

Атрибуты устройства MPIO

Следующие атрибуты поддерживаются только разветвленными устройствами. Атрибуты можно просматривать и изменять с помощью SMIT или команд (в частности, команд **lsattr** и **chdev**).

Некоторые атрибуты устройства многомаршрутного ввода-вывода (MPIO) поддерживают одновременное обновление атрибутов. Значения атрибутов можно изменять, когда диск открыт и используется, и новые значения применяются незамедлительно. Для некоторых атрибутов, в частности, **reserve_policy**, могут существовать ограничения относительно того, когда можно их изменять или какие новые значения может принимать атрибут. Например, если диск открыт и в данный момент используется как диск хранилища

кластера, то операционная система AIX блокирует любые попытки задать резервную стратегию для диска, потому что другие узлы кластера потеряют доступ к хранилищу.

Все устройства MPIO должны поддерживать атрибут **reserve_policy**. Обычно поддерживается также атрибут **PR_key**. У устройства MPIO могут быть и другие атрибуты. Список таких атрибутов приведен ниже.

FC3_REC
Указывает, требуется ли устройству включать восстановление после ошибок с использованием волоконно-оптического канала класса 3. Включение этой возможности повышает эффективность обнаружения и восстановления после определенных типов ошибок для фабрики, которые относятся к волоконно-оптическому каналу. Этот атрибут доступен только в ограниченном ассортименте устройств. Атрибут может принимать следующие значения:

true Включает восстановление после ошибок с использованием волоконно-оптического канала класса 3.

false Выключает восстановление после ошибок с использованием волоконно-оптического канала класса 3.

reserve_policy
Определяет, будет при при открытии устройства применяться резервирование. Допустимы следующие значения:

no_reserve
Резервирование применяться не будет. К устройству могут обращаться другие инициаторы, включая находящихся в других системах.

single_path_reserve
Для устройства будет применяться способ резервирования SCSI2, то есть к устройству сможет обращаться только инициатор резервирования. Другие инициаторы обращаться к устройству не смогут. С помощью технологии SCSI2 устройство будет назначено только одному инициатору (пути), все остальные запросы вызовут конфликт.

Если выбрано значение **single_path_reserve**, то алгоритмы выбора пути могут привести к неэффективному использованию ресурсов системы. Например, допустим, что у PCM есть атрибут, которому присвоено значение, распределяющее ввод-вывод по нескольким путям. Если включено **single_path_reserve**, драйвер диска может вызвать сигнал BDR, а затем зарезервировать диск для нового пути, отправив команду отмены предыдущего резервирования. При каждом выборе нового пути ресурсы начинают использоваться неэффективно, а производительность снижается, так как происходит отправка BDR и резервирование целевого устройства. (AIX PCM не позволяет выбирать алгоритмы, которые могут вызвать неэффективное использование ресурсов.)

PR_exclusive
Применяет при открытии постоянное резервирование SCSI3 и технологию единственного хоста. Значение **PR_key** не должно повторяться в пределах одной системы. Атрибут **PR_key** применяется для предотвращения доступа к устройству со стороны инициаторов из других систем.

PR_shared
Применяет при открытии постоянное резервирование SCSI3 и технологию нескольких хостов. Значение **PR_key** не должно повторяться в пределах одной системы. Инициаторы из других систем должны зарегистрироваться перед обращением к устройству.

PR_key
Требуется только в том случае, если устройство поддерживает любую из постоянных стратегий резервирования (**PR_exclusive** или **PR_shared**).

Понятия, связанные с данным:

“Настройка устройства MPIO” на стр. 542
При настройке устройства MPIO применяются те же команды, что и при настройке обычных устройств.

Информация, связанная с данной:

lsattr, команда

chdev, команда

Атрибуты модуля управления путем

В дополнение к PCM, предусмотренным в AIX по умолчанию, производитель может предоставить PCM для конкретного устройства. Набор атрибутов, которые может изменять пользователь, определяется производителем устройства. У такого PCM могут быть атрибуты устройства и атрибуты пути.

Ниже перечислены атрибуты устройства, которые поддерживаются PCM, предусмотренными в AIX по умолчанию:

algorithm

Задает способ распределения ввода-вывода устройства по путям. Для атрибута algorithm допустимы следующие значения:

Примечание: Некоторые устройства поддерживают только подмножество этих значений.

failover

Отправляет все операции ввода-вывода по единственному маршруту. Если маршрут помечен как давший сбой или выключенный, то для отправки всех операций ввода-вывода выбирается следующий доступный маршрут. Этот алгоритм хранит все включенные пути в упорядоченном списке в зависимости от значения атрибута **приоритет маршрута** в порядке возрастания. Для каждой операции ввода-вывода выбирается допустимый маршрут с наименьшим значением приоритета.

round_robin

Распределяет операции ввода-вывода по нескольким включенным маршрутам. Для устройств, имеющих активные и пассивные маршруты, или маршруты с разным уровнем предпочтения, для операций ввода-вывода используется только подмножество маршрутов. Если маршрут помечен как давший сбой или выключенный, то он больше не используется для отправки операций ввода-вывода. Операции ввода-вывода распределяются в зависимости от атрибута **приоритет маршрута**. Маршруты с большими значениями приоритета получают больше операций ввода-вывода.

shortest_queue

Распределяет операции ввода-вывода по нескольким включенным маршрутам. Для устройств, имеющих активные и пассивные маршруты, или маршруты с разным уровнем предпочтения, для операций ввода-вывода используется только подмножество маршрутов. Этот алгоритм подобен алгоритму **round_robin**. Однако алгоритм **shortest_queue** распределяет операции ввода-вывода в зависимости от числа ожидающих операций ввода-вывода для каждого маршрута. Для очередной операции выбирается маршрут, имеющий наименьшее число ожидающих операций ввода-вывода. Если выбран алгоритм **shortest_queue**, то атрибут **приоритет маршрута** игнорируется.

hcheck_mode

Указывает, какие пути должны проверяться во время проверки работоспособности. Этот атрибут поддерживает следующие режимы:

разрешено

Команда **healthcheck** отправляется по всем доступным маршрутам. Этот режим не отправляет команду **healthcheck** по маршрутам, которые выключены или отсутствуют.

failed Команда **healthcheck** отправляется по всем маршрутам, давшим сбой. Этот режим не отправляет команду **healthcheck** по маршрутам, которые включены, выключены или отсутствуют.

nonactive

(По умолчанию) Отправляет команду **healthcheck** по маршрутам, по которым не

выполняются операции ввода-вывода, в том числе по маршрутам, давшим сбой и по включенным маршрутам. Этот режим не отправляет команду **healthcheck** по маршрутам, которые выключены или отсутствуют.

hcheck_interval
Задает интервал проверки работоспособности путей устройства. Допустимы значения от 0 до 3600 секунд. Если указано значение 0, то проверка работоспособности не выполняется.

Примечание: Проверка работоспособности выполняется только в том случае, если диск открыт каким-либо процессом и пока не закрыт. Если диск не открыт, то модуль управления путем не проверяет пути, даже если атрибуту **hcheck_interval** этого устройства присвоено ненулевое значение.

dist_tw_width
Задает "временное окно". Это интервал времени, в течение которого распределенный алгоритм обнаружения ошибок собирает данные об операциях ввода-вывода, в которых возникла ошибка. Значение атрибута **dist_tw_width** измеряется в миллисекундах. Чем меньше это значение, тем меньше интервал времени для взятия каждого образца, и тем ниже чувствительность алгоритма к коротким последовательностям ошибок ввода-вывода. Чем больше значение атрибута, тем выше чувствительность алгоритма к коротким последовательностям ошибок и вероятность обнаружения неисправного пути.

dist_err_percent
Определяет процентную долю "временных окон", в которых была обнаружена ошибка. При достижении этого значения путь помечается как неисправный из-за своей низкой производительности. Значение атрибута **dist_err_percent** должно лежать в диапазоне от 0 до 100. Когда этот атрибут равен 0, распределенный алгоритм обнаружения ошибок выключен. Значение по умолчанию - 0. Распределенный алгоритм обнаружения ошибок берет образцы в пути от устройства к адаптеру и проверяет их на наличие ошибок. Алгоритм подсчитывает процентную долю образцов с ошибками. Если эта доля больше значения атрибута **dist_err_percent**, то путь помечается как неисправный.

Ниже перечислены атрибуты пути AIX PCM:

path priority
Изменяет поведение алгоритма выбора путей из списка.

Если значение атрибута algorithm - failover, то информация о путях хранится в виде списка. Порядок следования записей в этом списке задает последовательность выбора путей. Записи в списке упорядочены на основании значения приоритета. Наивысший приоритет - 1. Одно и то же значение приоритета может быть присвоено нескольким путям, однако если приоритеты всех путей совпадают, то при выборе учитывается время настройки путей.

Если используется значение атрибута алгоритма **round_robin**, то алгоритм **приоритет маршрута** назначает каждому маршруту приоритет. Маршруты выбираются для операций ввода-вывода в пропорциональной зависимости от приоритетов маршрутов. Поэтому пути с более высоким приоритетом выбираются для большего количества операций ввода-вывода. Если все приоритеты одинаковы, то маршруты выбираются в равных пропорциях.

cntl_hcheck_int

Последовательность проверки работоспособности контроллера запускается при обнаружении сбоя транспорта фабрики памяти. Атрибут **cntl_delay_time** определяет максимальную продолжительность активности последовательности проверки работоспособности контроллера, в секундах. Если команда проверки работоспособности контроллера выполняется успешно и обнаруживает доступный путь, то последовательность проверки работоспособности контроллера завершит работу, сделав возможным продолжение ввода-вывода. Если по окончании последовательности проверки работоспособности контроллера пригодных путей не будет обнаружено, то все ожидающие и последующие запросы на ввод-вывод к устройству будут отклоняться, пока служба проверки работоспособности устройства не обнаружит, что возвращен неисправный путь.

Пока последовательность проверки работоспособности контроллера активна, атрибут **cntl_hcheck_interval** задает интервал в секундах, с которым выдаются команды проверки работоспособности контроллера. **cntl_hcheck_interval** должен быть меньше, чем **cntl_delay_time**, если только он не равен нулю или отключен. Если **cntl_delay_time** или **cntl_hcheck_interval** равен нулю, то эта функция отключена.

cntl_delay_time

Последовательность проверки работоспособности контроллера запускается при обнаружении сбоя транспорта фабрики памяти. Атрибут **cntl_delay_time** определяет максимальную продолжительность активности последовательности проверки работоспособности контроллера, в секундах. Если команда проверки работоспособности контроллера выполняется успешно и обнаруживает доступный путь, то последовательность проверки работоспособности контроллера завершит работу, сделав возможным продолжение ввода-вывода. Если по окончании последовательности проверки работоспособности контроллера пригодных путей не будет обнаружено, то все ожидающие и последующие запросы на ввод-вывод к устройству будут отклоняться, пока служба проверки работоспособности устройства не обнаружит, что возвращен неисправный путь.

Пока последовательность проверки работоспособности контроллера активна, атрибут **cntl_hcheck_interval** задает интервал в секундах, с которым выдаются команды проверки работоспособности контроллера. **cntl_hcheck_interval** должен быть меньше, чем **cntl_delay_time**, если только он не равен нулю или отключен. Если **cntl_delay_time** или **cntl_hcheck_interval** равен нулю, то эта функция отключена.

timeout_policy

Настраивает поведение PCM для тайм-аутов команды и ошибок транспорта. Когда для **timeout_policy** установлено значение **fail_path** или **disable_path**, может снизится производительность, если устройство многоканального ввода/вывода (MPIO) встретит перемежающиеся неполадки фабрики сети хранения данных (SAN) на некоторых путях устройства. Атрибут **timeout_policy** может иметь следующие значения:

retry_path

Первое вхождение тайм-аута команды на пути не вызывает немедленной ошибки пути. Если путь, в котором возникла неполадка из-за проблем транспорта, восстановлен проверкой работоспособности, то восстановленный путь может быть использован немедленно.

fail_path

Возникает ошибка пути при первом вхождении тайм-аута команды, если он не является последним путем в группе путей. Если путь, в котором возникла ошибка из-за проблем транспорта, восстановлен, то он не используется для операций ввода/вывода записи и чтения, пока не пройдет определенный промежуток времени без ошибок в этом пути. Когда эта функция включена, может возникнуть ошибка, перед тем как операция чтения или записи будет перенаправлена в пути, восстановленные после ошибки транспорта.

disable_path

Возникает ошибка пути при первом вхождении тайм-аута команды, если он не является последним путем в группе путей. Если путь, в котором возникла ошибка из-за проблем транспорта, восстановлен, то он не используется для операций ввода/вывода записи и чтения, пока не пройдет определенный промежуток времени без ошибок в этом пути. Если в этом пути продолжают возникать множественные тайм-ауты команд в течение определенного промежутка времени, он может быть выключен. Выключенные пути остаются выключенными (и не используются), пока не выполнено одно из следующий действий: выполнена команда **chpath** для включения выключенного пути, перенастроен затронутый диск или перезагружена система.

Понятия, связанные с данным:

“Настройка устройства MPIO” на стр. 542
При настройке устройства MPIO применяются те же команды, что и при настройке обычных устройств.

Атрибуты репликации SAN

Должен быть установлен Многоканальный ввод-вывод (MPIO) AIX и устройство должно использовать модуль управления путями (PCM) AIX. Эти атрибуты имеют зависимости от настроек и компонентов, предоставленных подсистемой дискового пространства.

Следующие атрибуты AIX относятся к поведению номеров логических устройств (LUN), которые реплицируются через подсистемы дискового пространства. Все подсистемы дискового пространства или уровни микрокода могут не поддерживать эти атрибуты. Программное обеспечение кластеризации или обеспечения высокой готовности, такое как PowerHA SystemMirror, устанавливается для координации управления репликацией сети хранения данных (SAN) среди узлов в кластере. Следующие атрибуты не могут быть изменены на Сервере виртуального ввода-вывода (VIOS).

san_rep_cfg

Определяет, как синхронное устройство, которое использует Копирование на удаленный равноправный сервер (PPRC), определено и настроено в операционной системе AIX. Этот атрибут влияет на значение `unique_id` экземпляра диска, и оно может измениться при изменении значения атрибута. Атрибут **san_rep_cfg** не изменяет состояния устройства PPRC в подсистеме дискового пространства. Атрибут имеет следующие значения:

none [Default]

Настраивает LUN в синхронном устройстве, которое использует PPRC в качестве отдельных логических экземпляров дисков.

new Определяет и настраивает синхронное устройство, которое использует PPRC как одиночный логический экземпляр. Устройство определяется и настраивается, только когда ни один из существующих экземпляров диска не соответствует LUN в устройстве PPRC.

new_and_existing

Определяет и настраивает синхронное устройство, которое использует PPRC как одиночный экземпляр логического диска. Если ни один экземпляр логического диска не представляет устройство PPRC, определяется новый экземпляр диска.

migrate_disk

Определяет и настраивает синхронное устройство, которое использует PPRC как одиночный экземпляр физического диска и использует выбранный экземпляр логического диска для устройства. Операция поддерживается только на устройствах, на которых атрибут **san_rep_device** установлен как supported или detected. Когда на целевом устройстве атрибут **san_rep_device** установлен как supported, операция выполняется, только если репликация SAN была установлена в запоминающем устройстве после последней настройки диска. Эта операция поддерживается на дисках, которые открыты и используются операционными системами AIX, если устройство не применяется в качестве диска хранилища в кластере. Значение `unique_id` для затронутого диска обновляется в соответствии с ИД устройства PPRC.

revert_disk

Настраивает существующее синхронное устройство, которое использует экземпляр логического диска PPRC для экземпляра физического диска устройства, отличного от PPRC. Эта операция поддерживается только на устройствах, на которых атрибут **san_rep_device** установлен как yes. Имя логического устройства и специальный файл экземпляра целевого диска остаются неизменными. Первичный (исходный) LUN для устройства репликации SAN используется для возвращенного экземпляра физического диска. Если первичный (исходный) LUN не найден или неизвестен хосту, вторичный (целевой) LUN используется для возвращенного экземпляра физического диска. Эта операция поддерживается на дисках, которые открыты и используются операционными системами AIX, если устройство не применяется в качестве диска хранилища в кластере. Значение `unique_id` для затронутого диска обновляется в соответствии с ИД LUN.

Значения **none**, **new** и **new_and_existing** предназначены для изменения поведения для всех устройств того же уникального типа Администратора объектных данных (ODM). Команда **chdef** используется

для установки значений **none**, **new** и **new_and_existing**. Команда **chdef** изменяет значение по умолчанию для атрибута всех устройств указанного уникального типа ODM. Команда **chdef** также изменяет значение атрибута для существующих экземпляров устройств, которые уже определены. Для устройств, которые уже настроены при выполнении команды **chdef**, изменение не вступает в силу до перезагрузки системы, или пока эти устройства не будут перенастроены. Команда **chdef** требует класс, подкласс и тип атрибута. Для того чтобы определить уникальный тип ODM для атрибута **san_rep_cfg**, выполните команду `lsattr -l hdisk# -F class,subclass,type`. Пример:

```
lsdev -l hdisk0 -F class,subclass,type
disk,fcp,aixmpiods8k
chdef -a san_rep_cfg=none -c disk -s fcp -t aixmpiods8k
chdef -a san_rep_cfg=new -c disk -s fcp -t aixmpiods8k
chdef -a san_rep_cfg=new_and_existing -c disk -s fcp -t aixmpiods8k
```

Значения **migrate_disk** и **revert_disk** предназначены для изменения поведения одиночного и заданного экземпляра устройства. Команда **chdev** должна быть использована для установки значения **migrate_disk** или **revert_disk** заданного устройства. Команда **chdev** изменяет значение только для заданного устройства. Пример:

```
chdev -a san_rep_cfg=migrate_disk -l hdisk0
chdev -a san_rep_cfg=revert_disk -l hdisk0
```

san_rep_device

Указывает, что экземпляр логического диска определен как устройство репликации SAN. Этот атрибут устанавливается в процессе настройки диска и может устареть, если состояние устройства изменено после настройки диска. Атрибут имеет следующие значения:

no Устройство не настроено в операционной системе AIX как устройство репликации SAN. Это устройство не удовлетворяет требованиям для настройки в качестве устройства репликации SAN.

supported

Устройство не настроено в операционной системе AIX как устройство репликации SAN. Это устройство удовлетворяет требованиям для настройки в качестве устройства репликации SAN. Однако оно в данный момент не настроено как устройство репликации SAN в подсистеме дискового пространства.

detected

Устройство не настроено в операционной системе AIX как устройство репликации SAN. Операционная система AIX обнаружила, что это устройство удовлетворяет требованиям и может быть настроено в качестве устройства репликации SAN, и оно в данный момент настроено как устройство репликации SAN в подсистеме дискового пространства.

yes Устройство настроено в операционной системе AIX как устройство репликации SAN.

Информация, связанная с данной:

chdef, команда

chdev, команда

lsdev, команда

Удаление адаптера связи

Перед удалением или заменой адаптер с поддержкой оперативной замены необходимо отключить.

Для отключения адаптера связи необходимо выполнить следующие задачи:

- Закрыть все приложения, работающие с удаляемым или заменяемым адаптером.
- Идентифицировать и выключить все устройства, подключенные к адаптеру
- Составить список всех занятых разъемов или определить, в какой разъем установлен нужный адаптер
- Определить расположение разъема адаптера
- Просмотреть и удалить информацию об интерфейсе из списка сетевых интерфейсов
- Отключить адаптер

Для того чтобы выключить адаптер связи в конфигурации с помощью следующих процедур, необходимо войти в систему как пользователь **root**:

Понятия, связанные с данным:

“Управление оперативной заменой PCI” на стр. 533
Вы можете установить новый адаптер PCI с возможностью оперативной замены в разъем PCI во время работы системы.

Задачи, связанные с данной:

“Удаление конфигурации адаптеров связи PCI” на стр. 536
Ниже приведен обзор процедуры удаления адаптера связи PCI из конфигурации системы. Приведенная информация относится к адаптерам Ethernet, Token-ring, FDDI и ATM.

“Удаление или замена адаптера PCI оперативной замены” на стр. 536
Удаление или замену адаптера PCI оперативной замены можно выполнить, не завершая работу операционной системы и не отключая питание системы. После удаления адаптера его ресурсы станут недоступны для операционной системы и приложений.

Удаление конфигурации адаптеров Ethernet, Token-ring, FDDI и ATM:

Для удаления конфигурации адаптеров Ethernet, Token-ring, FDDI или ATM выполните следующие действия:

1. Введите команду `lsslot -c pci`, чтобы составить список всех установленных в системном блоке разъемов с поддержкой оперативной замены.
2. Введите соответствующую команду SMIT, показанную в данных примерах, чтобы составить список установленных адаптеров и просмотреть текущее состояние всех устройств в системном блоке:

Элемент	Описание
smit lsdenet	Чтобы составить список адаптеров Ethernet
smit lsdtok	Чтобы составить список адаптеров Token-Ring
smit ls_atm	Чтобы составить список адаптеров ATM

Для различных типов адаптеров применяются следующие соглашения о присвоении имен:

Элемент	Описание
Имя	**Тип адаптера**
atm0, atm1, ...	Адаптер ATM
ent0, ent1, ...	Адаптер Ethernet
tok0, tok1, ...	Адаптер Token-Ring

3. Закройте все приложения, работающие с адаптером, который необходимо отключить. Для продолжения этой процедуры необходимо отключить все устройства для дампа, расположенные в сети. Для этого выполните следующие действия:
   a. Введите в командной строке следующую команду:
      ```
      smit dump
      ```
   b. Выберите **Показать текущие устройства дампа**.
   c. Проверьте, не находятся ли некоторые настроенные устройства дампа в сети. Если таких устройств нет, то выйдите из SMIT и перейдите к шагу 4. Для того чтобы изменить расположение устройства дампа, нажмите **Отмена** или и перейдите к следующему шагу.
   d. Если в сети находится основное устройство дампа, то переместите его на локальный дик, выбрав **Изменить основное устройство дампа** и введя локальное расположение с поле **Основное устройство дампа**.
   e. Если в сети находится дополнительное устройство дампа, то переместите его на локальный дик, выбрав **Изменить дополнительное устройство дампа** и введя локальное расположение с поле **Дополнительное устройство дампа**.
   f. Нажмите **OK** или Enter.

4. Введите команду `netstat -i`, чтобы просмотреть список всех настроенных интерфейсов и определить, настроен ли адаптер для работы с TCP/IP. Вывод команды будет выглядеть примерно следующим образом:

```
Имя   Mtu    Сеть      Адрес          Ipkts  Ierrs  Opkts Oerrs Coll
lo0   16896  link#1                     076      0    118     0    0
lo0   16896  127       127.0.0.1        076      0    118     0    0
lo0   16896  ::1                        076      0    118     0    0
tr0    1492  link#2    8.0.5a.b8.b.ec   151      0    405    11    0
tr0    1492  19.13.97  19.13.97.106     151      0    405    11    0
at0    9180  link#3    0.4.ac.ad.e0.ad    0      0      0     0    0
at0    9180  6.6.6     6.6.6.5            0      0      0     0    0
en0    1500  link#5    0.11.0.66.11.1   212      0      1     0    0
en0    1500  8.8.8     8.8.8.106        212      0      1     0    0
```

   У адаптеров Token-Ring может быть только один интерфейс. У адаптеров Ethernet может быть два интерфейса. У адаптеров ATM может быть несколько интерфейсов.

5. Введите соответствующую команду **ifconfig** , как это показано в приведенных ниже примерах, чтобы удалить интерфейс из списка сетевых интерфейсов.

Элемент	Описание
`ifconfig en0 detach`	Чтобы удалить стандартный интерфейс Ethernet
`ifconfig et0 detach`	Чтобы удалить интерфейс Ethernet IEEE 802.3
`ifconfig tr0 detach`	Чтобы удалить интерфейс Token-Ring
`ifconfig at0 detach`	Чтобы удалить интерфейс ATM

6. Введите соответствующую команду **rmdev** , как это показано в приведенных ниже примерах, чтобы отключить адаптер и *сохранить* соответствующее ему определение устройства в классе объектов настраиваемых устройств:

Элемент	Описание
`rmdev -l ent0`	Чтобы отключить адаптер Ethernet
`rmdev -l tok1`	Чтобы отключить адаптер Token-Ring
`rmdev -l atm1`	Чтобы отключить адаптер ATM
`rmdev -p pci1`	Чтобы удалить все дочерние устройства шины PCI и все их дочерние устройства, сохранив в объектном классе Настроенные устройства их определения.

   Примечание: Для того чтобы отключить адаптер и *удалить* определение устройства из класса объектов настраиваемых устройств, укажите с командой **rmdev** флаг **-d**.
   Внимание: *Не* указывайте флаг **-d** с командой **rmdev** при оперативной замене адаптера.

Задачи, связанные с данной:

“Удаление конфигурации адаптеров ATM” на стр. 553
Перед удалением адаптера необходимо отключить все устройства эмуляции LAN.

Удаление конфигурации адаптеров WAN:

Вы можете удалить конфигурацию адаптера WAN.

Для отключения адаптера WAN выполните следующие действия:

1. Введите команду `lsslot -c pci`, чтобы составить список всех установленных в системном блоке разъемов с поддержкой оперативной замены.
2. Введите соответствующую команду SMIT, показанную в данных примерах, чтобы составить список установленных адаптеров и просмотреть текущее состояние всех устройств в системном блоке:

Элемент	Описание
`smit 331121b9_ls`	Чтобы составить список 2-портовых многопротокольных адаптеров WAN
`smit riciophx_ls`	Чтобы составить список адаптеров WAN ARTIC

Для различных типов адаптеров применяются следующие соглашения о присвоении имен:

Имя	Тип адаптера
dpmpa	2-портовый многопротокольный адаптер
riciop	Адаптер ARTIC960

3. Введите команду `lsdev -C -c port`, чтобы просмотреть список портов X.25 хоста. Вывод команды будет выглядеть примерно следующим образом:

```
sx25a0  Доступно 00-05-01-00     Порт X.25
x25s0   Доступно 00-05-01-00-00  Эмулятор V.3 X.25
```

4. Закройте все приложения, работающие с адаптером, который необходимо отключить. Для продолжения этой процедуры необходимо отключить все устройства для дампа, расположенные в сети. Для этого выполните следующие действия:
   a. Введите в командной строке следующую команду:

      ```
      smit dump
      ```

   b. Выберите **Показать текущие устройства дампа**.
   c. Проверьте, не находятся ли некоторые настроенные устройства дампа в сети. Если таких устройств нет, то выйдите из SMIT и перейдите к шагу 5, описанному ниже. Для того чтобы изменить расположение устройства дампа, нажмите **Отмена** или и перейдите к следующему шагу.
   d. Если в сети находится основное устройство дампа, то переместите его на локальный дик, выбрав **Изменить основное устройство дампа** и введя локальное расположение с поле **Основное устройство дампа**.
   e. Если в сети находится дополнительное устройство дампа, то переместите его на локальный дик, выбрав **Изменить дополнительное устройство дампа** и введя локальное расположение с поле **Дополнительное устройство дампа**.
   f. Нажмите **OK** или Enter.
5. В приведенной ниже таблице указаны команды, позволяющие отключить и удалить драйверы для следующих адаптеров:

Имя	2-портовый многопротокольный адаптер
`smit rmhdlcdpmpdd`	Чтобы отключить устройство
`smit rmsdlcscied`	Чтобы отключить эмулятор SDLC COMIO

Имя	Адаптер ARTIC960Hx PCI
`smit rmtsdd`	Чтобы отключить драйвер устройства
`smit rmtsdports`	Чтобы удалить порт эмуляции MPQP COMIO

Удаление конфигурации 4-портового адаптера PCI 10/100 Base-TX Ethernet IBM:

В 4-портовом адаптере PCI 10/100 Base-TX Ethernet четыре порта, каждый из которых необходимо отключить перед удалением адаптера.

1. Введите команду `lsslot -c pci`, чтобы составить список всех установленных в системном блоке разъемов с поддержкой оперативной замены.
2. Введите команду `smit lsdenet`, чтобы составить список всех устройств подкласса PCI. Вывод команды будет выглядеть примерно следующим образом:

```
ent1  Доступно  1N-00 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Порт 1)
ent2  Доступно  1N-08 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Порт 2)
ent3  Доступно  1N-10 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Порт 3)
ent4  Доступно  1N-18 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Порт 4)
```

3. Закройте все приложения, работающие с адаптером, который необходимо отключить. Для продолжения этой процедуры необходимо отключить все устройства для дампа, расположенные в сети. Для этого выполните следующие действия:
   a. Введите в командной строке следующую команду:
      ```
      smit dump
      ```
   b. Выберите **Показать текущие устройства дампа**.
   c. Проверьте, не находятся ли некоторые настроенные устройства дампа в сети. Если таких устройств нет, то выйдите из SMIT и перейдите к шагу 4. Для того чтобы изменить расположение устройства дампа, нажмите **Отмена** или и перейдите к следующему шагу.
   d. Если в сети находится основное устройство дампа, то переместите его на локальный дик, выбрав **Изменить основное устройство дампа** и введя локальное расположение с поле **Основное устройство дампа**.
   e. Если в сети находится дополнительное устройство дампа, то переместите его на локальный дик, выбрав **Изменить дополнительное устройство дампа** и введя локальное расположение с поле **Дополнительное устройство дампа**.
   f. Нажмите **OK** или Enter.
4. Введите команду `netstat -i`, чтобы просмотреть список всех настроенных интерфейсов и определить, настроен ли адаптер для работы с TCP/IP. Вывод команды будет выглядеть примерно следующим образом:

   ```
   Имя   Mtu    Сеть      Адрес          Ipkts  Ierrs  Opkts Oerrs Coll
   lo0   16896  link#1                     076      0    118     0    0
   lo0   16896  127       127.0.0.1        076      0    118     0    0
   lo0   16896  ::1                        076      0    118     0    0
   tr0    1492  link#2    8.0.5a.b8.b.ec   151      0    405    11    0
   tr0    1492  19.13.97  19.13.97.106     151      0    405    11    0
   at0    9180  link#3    0.4.ac.ad.e0.ad    0      0      0     0    0
   at0    9180  6.6.6     6.6.6.5            0      0      0     0    0
   en0    1500  link#5    0.11.0.66.11.1   212      0      1     0    0
   en0    1500  8.8.8     8.8.8.106        212      0      1     0    0
   ```

   У адаптеров Ethernet может быть два интерфейса, например, `et0` и `en0`.
5. С помощью команды **ifconfig** удалите интерфейсы из списка сетевых интерфейсов. Например, введите команду `iconfig en0 detach`, чтобы удалить стандартный интерфейс Ethernet, а затем введите команду `iconfig et0` для удаления интерфейса IEEE 802.3.
6. С помощью команды **rmdev** можно удалить конфигурацию адаптера и сохранить определение его драйвера в классе настроенных устройств. Например, введите `rmdev -l ent0`.

   Примечание: Для того чтобы отключить адаптер и *удалить* определение устройства из класса объектов настраиваемых устройств, укажите с командой **rmdev** флаг **-d**.
   Внимание: *Не* указывайте флаг **-d** с командой **rmdev** при оперативной замене адаптера.

Удаление конфигурации адаптеров ATM:

Перед удалением адаптера необходимо отключить все устройства эмуляции LAN.

Адаптеры ATM могут применяться с протоколами эмуляции обычных IP и LAN. Протокол эмуляции LAN позволяет реализовать эмуляцию LAN в сети ATM. Существуют следующие типы эмуляции LAN: Ethernet/IEEE 802.3, Token-ring/IEEE 802.5 и MPOA (MultiProtocol Over ATM).

Выполните следующие действия для удаления интерфейса LAN:

1. Введите команду `lsslot -c pci`, чтобы составить список всех установленных в системном блоке разъемов с поддержкой оперативной замены.
2. Введите команду `smit ls_atm`, чтобы составить список всех адаптеров ATM. Вывод команды будет выглядеть примерно следующим образом:

   ```
   .
   .
   atm0 Доступно  04-04 IBM PCI 155 Mbps ATM Adapter (14107c00)
   atm1 Доступно  04-06 IBM PCI 155 Mbps ATM Adapter (14104e00)
   ```
3. Введите команду `smit listall_atmle`, чтобы составить список всех клиентов эмуляции LAN на этих адаптерах. Вывод команды будет выглядеть примерно следующим образом:

   ```
   ent1 Доступно   Клиент эмуляции LAN ATM  (Ethernet)
   ent2 Доступно   Клиент эмуляции LAN ATM  (Ethernet)
   ent3 Доступно   Клиент эмуляции LAN ATM  (Ethernet)
   tok1 Доступно   Клиент эмуляции LAN ATM  (Token Ring)
   tok2 Доступно   Клиент эмуляции LAN ATM  (Token Ring)
   ```

   Каждый адаптер ATM может использоваться несколькими клиентами эмуляции.
4. Введите команду `smit listall_mpoa`, чтобы составить список всех клиентов эмуляции LAN на этих адаптерах. Вывод команды будет выглядеть примерно следующим образом:

   ```
   mpc0 Доступно      Клиент эмуляции MPOA LAN ATM
   ```

   `atm0` и `atm1` - физические адаптеры ATM. `mpc0` - клиент эмуляции MPOA. `ent1`, `ent2`, `ent3`, `tok1` и `tok2` - клиенты эмуляции LAN.
5. Введите `entstat`, чтобы определить, какой адаптер использует клиент. Вывод команды будет выглядеть примерно следующим образом:

   ```
   ------------------------------------------------------------
   СТАТИСТИКА ETHERNET (ent1) :
   Тип устройства: ATM LAN EmulationATM Адрес: 00:04:ac:ad:e0:ad
   .
   .
   .
   Статистика эмуляции LAN ATM:
   -------------------------------------
   Имя эмуляции LAN: ETHelan3
   Имя локального устройства ATM: atm0
   Локальный MAC-адрес LAN:
   .
   .
   ```
6. Закройте все приложения, работающие с адаптером, который необходимо отключить. Для продолжения этой процедуры необходимо отключить все устройства для дампа, расположенные в сети. Для этого выполните следующие действия:
   a. Введите в командной строке следующую команду:

      ```
      smit dump
      ```
   b. Выберите **Показать текущие устройства дампа**.
   c. Проверьте, не находятся ли некоторые настроенные устройства дампа в сети. Если таких устройств нет, то выйдите из SMIT и перейдите к шагу 7. Для того чтобы изменить расположение устройства дампа, нажмите **Отмена** или и перейдите к следующему шагу.
   d. Если в сети находится основное устройство дампа, то переместите его на локальный дик, выбрав **Изменить основное устройство дампа** и введя локальное расположение с поле **Основное устройство дампа**.
   e. Если в сети находится дополнительное устройство дампа, то переместите его на локальный дик, выбрав **Изменить дополнительное устройство дампа** и введя локальное расположение с поле **Дополнительное устройство дампа**.
   f. Нажмите **OK** или Enter.

7. С помощью команды **rmdev -l** *устройство* можно удалить конфигурацию интерфейса в следующем порядке:
   - Интерфейс эмуляции = en1, et1, en2, et2, tr1, tr2 ...
   - Интерфейс эмуляции = ent1, ent2, tok1, tok2 ...
   - Многопротокольный интерфейс ATM (MPOA) = mpc0
   - Адаптер ATM = atm0
8. Для того чтобы удалить адаптер SCSI `scsi1` и все его дочерние устройства, сохранив в объектном классе Настроенные устройства их определения, введите:

   ```
   rmdev  -R scsi1
   ```

   Будет показана примерно следующее сообщение:

   ```
   rmt0 Определено
   hdisk1 Определено
   scsi1 Определено
   ```
9. Для того чтобы удалить только дочерние устройства адаптера SCSI `scsi1`, сохранив в объектном классе Настроенные устройства их определения, введите:

   ```
   rmdev  -p scsi1
   ```

   Будет показана примерно следующее сообщение:

   ```
   rmt0 Определено
   hdisk1 Определено
   ```
10. Для того чтобы удалить все дочерние устройства шины PCI `pci1` и все их дочерние устройства, сохранив в объектном классе Настроенные устройства их определения, введите:

    ```
    rmdev  -p pci1
    ```

    Будет показана примерно следующее сообщение:

    ```
    rmt0 Определено
    hdisk1 Определено
    scsi1 Определено
    ent0 Определено
    ```

Задачи, связанные с данной:

“Удаление конфигурации адаптеров Ethernet, Token-ring, FDDI и ATM” на стр. 550
Для удаления конфигурации адаптеров Ethernet, Token-ring, FDDI или ATM выполните следующие действия:

Удаление настройки адаптеров памяти

Перед удалением или заменой адаптер памяти необходимо отключить.

Для выполнения этих задач необходимо войти в систему как пользователь root.

Следующая процедура удаляет конфигурацию адаптеров устройств хранения SCSI и Fibre Channel.

Для отключения адаптера памяти необходимо выполнить следующие задачи:

- Закрыть все приложения, работающие с удаляемым, заменяемым или перемещаемым адаптером.
- Размонтировать файловые системы
- Идентифицировать и выключить все устройства, подключенные к адаптеру
- Составить список всех занятых разъемов или определить, в какой разъем установлен нужный адаптер
- Определить расположение разъема адаптера
- Отключить родительские и дочерние устройства
- Отключить адаптер

Удаление адаптеров SAS, SCSI и Fibre Channel из конфигурации

К адаптерам памяти обычно подключаются носители, например диски и накопители на магнитной ленте. Для удаления родительского устройства необходимо удалить или перевести в состояние "определено" все подключенные к ним устройствам.

Для того чтобы удалить конфигурацию адаптеров SCSI и Fibre Channel, выполните следующие действия:

1. Закройте все приложения, работающие с адаптером, который необходимо отключить.
2. Введите команду `lsslot-c pci` чтобы составить список всех установленных в системном блоке разъемов с поддержкой оперативной замены.
3. Введите `lsdev -C` для просмотра текущего состояния всех устройств в системном блоке.
4. Введите команду `umount`, чтобы размонтировать файловые системы, каталоги или файлы, для которых используется данный адаптер. Дополнительная информация приведена в разделе Монтирование JFS или JFS2.
5. Введите `rmdev -l adapter -R` чтобы сделать адаптер недоступным.

   Внимание: *He* указывайте флаг **-d** с командой **rmdev** для выполнения оперативной замены, так как при этом будет удалена текущая конфигурация.

Понятия, связанные с данным:

“Управление оперативной заменой PCI” на стр. 533
Вы можете установить новый адаптер PCI с возможностью оперативной замены в разъем PCI во время работы системы.

Задачи, связанные с данной:

“Удаление или замена адаптера PCI оперативной замены” на стр. 536
Удаление или замену адаптера PCI оперативной замены можно выполнить, не завершая работу операционной системы и не отключая питание системы. После удаления адаптера его ресурсы станут недоступны для операционной системы и приложений.

Удаление конфигурации асинхронного абзаца

Можно удалить конфигурацию асинхронного адаптера.

Для выполнения этих задач необходимо войти в систему как пользователь root.

Ниже описаны действия по удалению конфигурации асинхронного адаптера.

Перед удалением или заменой асинхронный адаптер необходимо отключить. Для отключения асинхронного адаптера необходимо выполнить следующие задачи:

- Закрыть все приложения, работающие с удаляемым, заменяемым или перемещаемым адаптером.
- Идентифицировать и выключить все устройства, подключенные к адаптеру
- Составить список всех занятых разъемов или определить, в какой разъем установлен нужный адаптер
- Определить расположение разъема адаптера
- Отключить родительские и дочерние устройства
- Отключить адаптер

Процедура

Перед удалением или заменой асинхронного адаптера необходимо отключить сам адаптер и все подключенные к нему устройства. Для отключения этих устройств необходимо завершить работу всех работающих с ними процессов. Выполните следующие действия:

1. Закройте все приложения, работающие с адаптером, который необходимо отключить.
2. Введите команду `lsslot-c pci` чтобы составить список всех установленных в системном блоке разъемов с поддержкой оперативной замены.
3. Введите `lsdev -C -c tty` для просмотра текущего состояния всех устройств в системном блоке.

4. Введите команду `lsdev -C -c printer`, чтобы составить список всех принтеров и графопостроителей, подключенных к адаптеру.
5. Чтобы сделать адаптер недоступным, выполните команду **rmdev**.

   Внимание: *Не* указывайте флаг **-d** с командой **rmdev** для выполнения оперативной замены, так как при этом будет удалена текущая конфигурация.

Понятия, связанные с данным:

“Управление оперативной заменой PCI” на стр. 533
Вы можете установить новый адаптер PCI с возможностью оперативной замены в разъем PCI во время работы системы.

“Состояния устройства” на стр. 528
Подключенные к системе устройства могут находиться в одном из следующих состояний.

Задачи, связанные с данной:

“Удаление или замена адаптера PCI оперативной замены” на стр. 536
Удаление или замену адаптера PCI оперативной замены можно выполнить, не завершая работу операционной системы и не отключая питание системы. После удаления адаптера его ресурсы станут недоступны для операционной системы и приложений.

Информация, связанная с данной:

Управление печатью

Устранение неполадок устройств ввода-вывода

Вы можете определить причину возникновения неисправности устройства.

Проверка программного обеспечения устройства

Ниже перечислены операции, позволяющие устранить неполадки, связанные с программным обеспечением устройств:

- Проверка протокола ошибок
- Просмотр всех устройств
- Проверка состояния устройства
- Проверка атрибутов устройства
- Изменение атрибутов устройства
- Применение устройства с другими приложениями
- Указание нового устройства

Проверка протокола ошибок

Проверьте, не было ли занесено в протокол ошибок сообщений об ошибках устройства, его адаптера или использующего устройство приложения. Сведения о выполнении этой проверки приведены в разделе Средство протоколирования ошибок. Выполнив указанные действия, вернитесь к этому шагу.

Была ли устранена неполадка?

Если предыдущий метод не позволил устранить неполадку, перейдите к следующему шагу(**Просмотр устройств**) для просмотра списка всех устройств.

Просмотр устройств

Составьте список всех определенных и доступных устройств с помощью команды **lsdev -C**. Это команда позволяет просмотреть характеристики всех устройств системы.

Если нужное устройство есть в списке, перейдите к следующему шагу (**Проверка состояния устройства**) для проверки состояния устройства.

Если устройство отсутствует в списке устройств, задайте новое устройство (см. ниже раздел **Определение нового устройства**).

Проверка состояния устройства

Найдите нужное устройство в списке, созданном с помощью команды **lsdev -C**. Проверьте, доступно ли устройство.

Если устройство находится в состоянии Доступно, перейдите к следующему шагу (**Проверка атрибутов устройства**) для проверки атрибутов устройства.

Если состояние устройства отличается от Доступно, задайте новое устройство (см. ниже раздел **Определение нового устройства**).

Проверка атрибутов устройства

С помощью команды **lsattr -E -l** *Имя-устройства* просмотрите атрибуты устройства.

Команда **lsattr** позволяет просмотреть атрибуты и значения атрибутов для устройств системы. Информация о правильной настройке атрибутов конкретного устройства приведена в его документации.

Если атрибуты устройства настроены правильно, см. ниже раздел **Применение устройства с другим приложением**.

Если атрибуты устройства заданы неправильно, перейдите к следующему шагу, **Изменение атрибутов устройства**.

Изменение атрибутов устройства

Измените атрибуты устройства с помощью команды **chdev -l** *Имя* **-a** *Атрибут***=***Значение*. Перед выполнением команды обратитесь к разделу *Справочник по командам, том 1*.

Команда **chdev** изменяет характеристики устройства, обозначенного флагом **-l** *имя*.

Если изменение атрибутов не устранило неполадку с устройством, перейдите к следующему шагу, **Применение устройства с другим приложением**.

Применение устройства с другим приложением

Попробуйте использовать это устройство в другом приложении. Если с другим приложением устройство работает нормально, проблема может заключаться в первом приложении.

Если с другим приложением устройство работает нормально, проблема может заключаться в первом приложении. Сообщите о неполадке в сервисное представительство.

Если устройство не работает с другим приложением, перейдите к следующему шагу, **Определение нового устройства**.

Определение нового устройства

Примечание: Для работы с командой **mkdev** у вас должны быть права доступа root или вы должны входить в группу security.

Добавьте устройство в систему с помощью команды **mkdev**.

Команда mkdev позволяет определить и сделать доступным новое устройство или сделать доступным устройство, определенное ранее. Уникально идентифицировать предопределенное устройство позволяют различные комбинации флагов **-c**, **-s** и **-t**. Перед выполнением команды обратитесь к разделу *Справочник по командам, том 3*.

Если после создания определения устройства неполадка не была устранена, вы можете завершить процедуру и сообщить о неполадке в сервисное представительство или проверить устройство с помощью диагностической программы.

Проверка соединения с устройством:

Для проверки соединения с устройством выполите следующие действия:

1. Проверьте, работает ли электрическая розетка, к которой подключено устройство.
2. Проверьте правильность подключения силового кабеля устройства к электрической розетке.
3. Проверьте правильность соединения кабеля передачи данных с устройством и с системным блоком.
4. В случае устройства SCSI проверьте правильность установки терминатора SCSI и настройки адреса SCSI.
5. В случае устройств связи проверьте правильность подключения к линии связи.
6. Убедитесь, что питание устройства включено.

Инструкции по подключению кабелей, настройке и устранению неполадок для конкретных устройств приведены в документации по этим устройствам.

Если неполадку устранить не удалось, то перейдите к следующему шагу.

Устранение неполадок при удалении адаптера:

Если устройство открыто при применении команды **rmdev** для удаления конфигурации адаптера, может появится сообщение об ошибке.

Если при вводе команды **rmdev** для отключения адаптера появляется сообщение следующего типа, это означает, что устройство открыто. Это может быть вызвано тем, что приложения продолжают обращаться к адаптеру, который вы собрались удалить или заменить.

```
#rmdev -l ent0
Ошибка метода (/usr/lib/methods/ucfgent):
        0514-062
 Невозможно выполнить указанную функцию, так как
 устройство занято.
```

Для устранения неполадки необходимо определить, какие приложения продолжают использовать адаптер, и закрыть их. Среди этих приложений могут быть:

- TCP/IP
- SNA
- OSI
- IPX/SPX
- Novell NetWare
- Streams
- Общий интерфейс управления каналом передачи данных (GDLC)
  - IEEE Ethernet DLC
  - Token-ring DLC
  - FDDI DLC

Приложения Системной сетевой архитектуры

Адаптер могут использовать следующие приложения SNA:
- DB2
- TXSeries (CICS & Encina)
- DirectTalk
- MQSeries
- HCON
- ADSM

Приложения Streams

Адаптер могут использовать следующие приложения streams:
- IPX/SPX
- Novell NetWare V4 и Novell NetWare Services 4.1
- Соединения и NetBios данной операционной системы

Приложения, выполняемые на адаптерах WAN

Адаптер WAN могут использовать следующие приложения:
- SDLC
- Bisync
- X.25
- ISDN
- QLLC для X.25

Приложения TCP/IP

Отключить приложения TCP/IP, применяющие уровень интерфейса позволяет команда **ifconfig**. При этом в приложении, применяющем TCP/IP, истекает время ожидания, и приложение выводит сообщение о том, что интерфейс отключен. После того, как адаптер будет добавлен или заменен, а интерфейс подключен с помощью команды **ifconfig**, работа приложения продолжится.

Проверка состояния готовности устройства:

Вы можете проверить, находится ли устройство в состоянии готовности.

Для определения готовности устройства выполните следующие действия:
1. Убедитесь, что индикатор готовности устройства горит.
2. Убедитесь, что съемные носители, такие как магнитные ленты, дискеты и компакт-диски, вставлены правильно.
3. В случае принтера или графопостроителя проверьте красящую ленту, наличие бумаги и красящего порошка.
4. При попытке записи на устройство убедитесь, что носитель не защищен от записи.

Была ли устранена неполадка после проверки? Если после проверки готовности устройств неполадка не была устранена, перейдите к следующему шагу.

Диагностика устройства:

Для определения вышедшего из строя устройства выполните диагностику аппаратного обеспечения.

Если в ходе выполнения аппаратной диагностики определить неполадку устройства не удалось, проверьте программное обеспечение устройства. Если устройство успешно прошло проверку, проблема может

заключаться в конфликте устройства с программным обеспечением системы. В этом случае сообщите о неполадке в организацию, обслуживающую аппаратное обеспечение системы.

Направленная настройка устройств

Команду **cfgmgr** можно использовать с флагом **-c** в качестве опции соединения для ограниченного диапазона направленных настроек устройств ввода-вывода.

Информация, связанная с данной:

команда cfgmgr

Направленная настройка устройств FC и FCoE

Опция **cfgmgr -c** используется с адаптерами волоконно-оптических каналов (FC) и волоконно-оптических каналов с использованием Ethernet (FCoE) для направленной настройки.

Команду **cfgmgr** можно использовать с флагом **-c** в качестве опции соединения для ограниченного диапазона настроек устройств. Для адаптеров FC и FCoE синтаксис таков:

cfgmgr -l fscsi0 **-c** "*параметр=значение*[,*параметр=значение*,...]"

С помощью строки соединения можно ограничить область обнаружения устройств, используя один из следующих параметров:

Таблица 70. Параметры для флага **cfgmgr -c**

Имя параметра	Описание
ww_name	Глобальное имя порта целевого устройства
node_name	Глобальное имя узла целевого устройства
scsi_id	ИД N_Port целевого устройства, который преобразуется в ИД SCSI для устройств хранения, использующих волоконно-оптический протокол (FCP)
lun_id	Номер логического накопителя (LUN)

Например, следующая команда настраивает единственный LUN на целевом порту хранилища **lun_id** 0x1000000000000 с глобальным именем порта 0x5001738000330191:

```
# cfgmgr -l fscsi0 -c "ww_name=0x5001738000330191,lun_id=0x1000000000000"
```

Это сканирование проводится только для порта адаптера хоста `fscsi0`.

Примечания:

- Начальные символы `0x` в значении параметра являются необязательными.
- Все параметры должны быть шестнадцатеричными числами.

В следующем примере указан только один параметр:

```
# cfgmgr -l fscsi0 -c "lun_id=0x1000000000000"
```

Эта команда сканирует все порты устройства хранения в сети области хранения (SAN) и настраивает этот единственный логический накопитель для каждого целевого порта SAN, где существует данный LUN.

Рекомендации и правила для параметров фильтрации соединений

При выборе параметров фильтрации обратите внимание на следующее:

- Направленная настройка для устройств FC и FCoE применяется только к средам с подключением коммутатора. Если указать строку соединения, которая непосредственно связана с целевым портом, то произойдет сбой соединения и будет выдано сообщение о том, что невозможно найти дочерние устройства.

- Флаг **-c** поддерживается только в сочетании с флагом **-l** команды **cfgmgr**, который ограничивает область действия команды одним устройством `fscsiX` в каждый конкретный момент.
- Если указать **-?** в качестве строки соединения для флага **-c** команды **cfgmgr** наряду с флагом **-v**, то будет выведена информация об использовании.
- Если параметр указан дважды (например, два параметра **lun_id**), произойдет ошибка. Устройства не будут обнаружены.
- Разрешены любые комбинации параметров **lun_id**, **scsi_id**, **ww_name** и **node_name** за исключением дубликатов. Для уникальной идентификации настраиваемого узла LUN, цели или хранилища следует указать один или, предпочтительно, два параметра, хотя можно указать и больше. В приведенном ниже списке указаны параметры или сочетания параметров, требуемые для уникальной идентификации узла LUN, цели или хранилища:
  - Параметры **ww_name** и **lun_id** обеспечивают уникальную идентификацию настраиваемого LUN на целевом порту.
  - Параметры **scsi_id** и **lun_id** обеспечивают уникальную идентификацию настраиваемого LUN на целевом порту.
  - Параметры **node_name** и **lun_id** настраивают LUN для всех целевых портов для определенного узла хранилища. С помощью этих параметров можно настроить целевые порты только в том случае, если у всех целевых портов совпадают значения параметра **node_name**, что может оказаться справедливым для некоторых устройств хранения.
  - Параметр **ww_name** настраивает все LUN для определенного целевого устройства.
  - Параметр **node_name** настраивает все целевые порты для определенного узла хранилища (только если у всех целевых портов совпадают значения параметра **node_name**, что может оказаться справедливым для некоторых устройств хранения).
  - Параметр **lun_id** настраивает LUN на всех целевых портах, которые видимы с этого устройства `fscsi`.
- Если указано более двух параметров, то код конфигурации устройства использует эту дополнительную информацию для проверки расположения устройства. Если значение какого-либо из указанных параметров конфликтует со значениями, указанными в SAN, то происходит сбой команды и устройства не настраиваются.

Информация, связанная с данной:

команда cfgmgr

Накопители на магнитной ленте

Описанные здесь функции управления системой относятся к накопителям на магнитной ленте.

Многие из задач включают получение или занесение информации в базу данных конфигурации устройств. База данных конфигурации устройств состоит из основной базы данных конфигурации, которая содержит информацию о всех возможных типах устройств, поддерживаемых системой, и базы данных пользовательской конфигурации, которая содержит информацию о конкретных устройствах, установленных в системе. Для применения в операционной системе устройство должно быть определено в базе данных пользовательской конфигурации, а его тип должен быть определен в основной базе данных конфигурации.

Атрибуты накопителей на магнитной ленте

Вы можете настроить атрибуты накопителей на магнитной ленте исходя из требований вашей системы.

Атрибуты можно просматривать и изменять с помощью SMIT или команд (в частности, команд **lsattr** и **chdev**).

Каждый накопитель на магнитной ленте использует только часть всех возможных атрибутов.

Общая информация о каждом атрибуте

Размер блока

Этот атрибут задает размер блока, используемого при работе с магнитной лентой (чтении или

записи). Данные записываются на ленту в виде блоков, с интервалами между соседними блоками. На неотформатированную магнитную ленту обычно записываются более длинные блоки, что позволяет сократить число межблоковых интервалов и записать больший объем данных. Значение **0** указывает на блоки переменной длины. Допустимые значения и значения, используемые по умолчанию, зависят от типа магнитной ленты.

Буферы устройств

Если атрибуту Буферы устройств присвоено значение `mode=yes` (командой **chdev**), то приложение уведомляется о завершении записи после передачи данных в буфер накопителя на магнитной ленте; в этот момент данные могут быть еще не записаны на ленту. Если указано значение `mode=no`, то приложение уведомляется о завершении записи только после того, как данные действительно записаны на ленту. Значение `mode=no` не позволяет использовать потоковый режим при чтении с ленты или записи на нее. Значение по умолчанию - `mode=yes`.

Если указан параметр `mode=no`, то накопитель на магнитной ленте работает медленнее, однако повышается надежность записи данных при сбое питания или неисправности системы. Кроме того, появляется возможность более точно контролировать ошибку конца ленты.

Расширенные метки файлов

Если для этого атрибута указано значение `no` (атрибут **extfm** команды **chdev**), то при записи метки файла всегда будут применяться стандартные метки. При указании значения `yes` записывается расширенная метка файла. Этот атрибут может применяться накопителями на магнитной ленте. Значение по умолчанию: `no`. Например, расширенные метки файлов на 8-мм ленте занимают 2,2 Мб памяти на ленте, а их запись может занять до 8,5 секунд. Стандартные метки файлов занимают 184 Кб и записываются примерно 1,5 с.

Для 8-мм ленты в режиме добавления используйте расширенные метки файлов. Это позволяет добиться более точного позиционирования после операций преобразования меток файлов и снизить количество ошибок.

Перемотка

Если для этого атрибута указано значение `ret=yes` (команда **chdev**, атрибут **ret**), то накопитель автоматически перемотает магнитную ленту после ее установки или после перезагрузки устройства. Под *перемоткой* ленты понимается ее перемотка до конца, а затем обратная перемотка к началу. Эта операция позволяет добиться равномерного натяжения магнитной ленты по всей ее длине. *Перемотка* ленты может уменьшить количество ошибок, однако на нее тратится несколько минут. Если указать значение `ret=no`, то устройство не будет автоматически перематывать ленту. Значение по умолчанию - `yes`.

Параметр плотности #1 и параметр плотности #2

Параметр плотности #1 (команда **chdev**, атрибут **density_set_1**) задает значение плотности, которую накопитель применяет при записи специальных файлов `/dev/rmt*`, `/dev/rmt*.1`, `/dev/rmt*.2` и `/dev/rmt*.3`. Параметр плотности #2 (команда **chdev**, атрибут **density_set_2**) задает значение плотности, которую накопитель применяет при записи специальных файлов `/dev/rmt*.4`, `/dev/rmt*.5`, `/dev/rmt*.6` и `/dev/rmt*.7`. Дополнительная информация приведена в разделе “Специальные файлы для накопителей на магнитной ленте” на стр. 574.

Значение плотности представляется десятичным числом в интервале от **0** до **255**. Нулевое (**0**) значение соответствует плотности, используемой по умолчанию для данного накопителя; как правило это высокая плотность. Конкретные допустимые значения плотности зависят от типа накопителя. Эти атрибуты не оказывают влияния на способность устройства читать ленты с различной поддерживаемой плотностью записи. Обычно в атрибуте Параметр плотности #1 указывают самую большую плотность записи для накопителя, а в атрибуте Параметр плотности #2 - предыдущий уровень плотности.

Поддержка резервирования

Для накопителей на магнитной ленте, поддерживающих атрибут резервирования (команда **chdev**, атрибут **res_support**) значение `res_support=yes` позволяет зарезервировать накопитель на шине SCSI, пока он открыт. Если накопитель используется несколькими адаптерами SCSI, это позволяет обеспечить доступ к устройству одного адаптера, пока устройство открыто. Некоторые накопители

SCSI не поддерживают команды резервирования/разблокирования. Для некоторых накопителей SCSI заданы предопределенные значения этого атрибута, поэтому команды резервирования/разблокирования поддерживаются всегда.

Размер блока переменной длины
Данный атрибут (команда **chdev** атрибут **var_block_size**) задает размер блока, необходимый накопителю при сохранении записей переменной длины. Для некоторых накопителей SCSI необходимо задавать ненулевой размер блока данных даже в том случае, если записи на ленте будут иметь переменную длину. Для выбора записей переменной длины присвойте атрибуту **Размер блока** значение **0**. Описание допустимых значений этого атрибута приведено в документации по конкретному накопителю SCSI.

Сжатие данных
Укажите значение `compress=yes` для атрибута **compress** команды **chdev**), чтобы включить на накопителе режим сжатия (если устройство его поддерживает). В этом случае устройство записывает данные на магнитную ленту в сжатом формате, поэтому на одну ленту может поместиться больший объем данных. Если для этого атрибута указано значение `no`, то накопитель записывает данные в исходном виде (без сжатия). Значение этого атрибута не влияет операцию чтения. Значение по умолчанию равно `yes`.

Автозагрузчик
Укажите для этого атрибута значение `autoload=yes` (команда **chdev** атрибут **autoload**), чтобы подключить автозагрузчик, если он присутствует. В этом случае, если загрузчик получает следующую ленту, выполнение любых операций чтения или записи, в результате которых был достигнут конец предыдущей ленты, автоматически продолжается со следующей ленты. Команды накопителя, действие которых ограничено одной единственной кассетой, будут работать по-прежнему. Значение по умолчанию равно `yes`.

Время между повторами
Этот атрибут задает период времени (в секундах) между моментом сбоя при выполнении команды и ее повторным запуском. Система может повторно выполнять команду, вызывающую сбой, до четырех раз. Этот атрибут допустим только для накопителей на магнитной ленте типа OST. По умолчанию устанавливается значение `45`.

Тайм-аут чтения/записи
Атрибут Тайм-аут чтения/записи или Максимальная задержка для **чтения/записи** задает максимальный интервал времени (в секундах), который система выделяет для завершения команды чтения или записи. Этот атрибут допустим только для накопителей на магнитной ленте типа OST. По умолчанию применяется значение `144`.

Ошибка возврата при смене ленты
Атрибут Сообщение об ошибке при смене ленты и сбросе (если он установлен) приводит к выдаче сообщения об ошибке, если накопитель на магнитной ленте был перезагружен, или была заменена магнитная лента. Перед закрытием ленты на накопителе необходимо выполнить операцию, после которой лента будет находиться не в начальном положении. Возвращается ошибка `-1`, `errno` равно `EIO`. При переходе в приложение код ошибки сбрасывается. Кроме того, код ошибки сбрасывается при изменении конфигурации накопителя на магнитной ленте.

Атрибуты накопителей на магнитной ленте для 4-мм ленты емкостью 2,0 ГБ (4mm2gb):

Ниже приведены атрибуты накопителей на магнитной ленте для 4-мм ленты емкостью 2,0 ГБ (4mm2gb)

Размер блока
Значение по умолчанию - 1024.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 4-мм магнитной ленты емкостью 4,0 ГБ, то атрибуты Перемотка, Размер блока переменной длины, Поддержка резервирования,

Параметр плотности #1 и Параметр плотности #2 имеют предопределенные значения, которые нельзя изменить. Это связано с тем, что накопитель на магнитной ленте всегда выполняет запись в режиме 2.0.

Атрибуты накопителей на магнитной ленте для 4-мм ленты емкостью 4,0 ГБ (4mm4gb):

Ниже приведены атрибуты накопителей на магнитной ленте для 4-мм ленты емкостью 4,0 ГБ (тип 4mm4gb).

Размер блока
Значение по умолчанию - 1024.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Пользователь не может изменять значение плотности записи на данном устройстве; устройство само автоматически настраивается в зависимости от установленного носителя:

Тип носителя	Конфигурация устройства
DDS	Только чтение.
DDS ||||	Чтение/запись только в режиме 2,0 ГБ.
DDS2	Чтение данных любой плотности; запись только в режиме 4,0 ГБ.
не-DDS	Не поддерживается; кассета будет выгружена.

Сжатие данных
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 4-мм магнитной ленты емкостью 4,0 ГБ, то атрибуты Перемотка, Размер блока переменной длины, Поддержка резервирования, Параметр плотности #1 и Параметр плотности #2 имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на 8-мм магнитной ленте емкостью 2,3 ГБ (8mm):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 2,3 ГБ размером 8 мм (тип 8mm).

Размер блока
Значение по умолчанию - 1024. Меньшая величина приводит к уменьшению объема данных, сохраняемых на магнитной ленте.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 8-мм магнитной ленты емкостью 2,3 ГБ, то атрибуты Перемотка, Поддержка резервирования, Размер блока переменной длины, Сжатие данных, Параметр плотности #1 и Параметр плотности #2 имеют предопределенные значения, которые нельзя изменить. Это связано с тем, что накопитель на магнитной ленте всегда выполняет запись в режиме 2,3.

Атрибуты накопителей на магнитной ленте емкостью 5,0 ГБ (тип 8mm5gb):

Ниже приведены атрибуты накопителей на магнитной ленте для 8-мм ленты емкостью 5,0 ГБ (8mm5gb)

Размер блока
Значение по умолчанию - 1024. Если запись на ленту выполняется в режиме 2.3 ГБ, то меньшее значение атрибута приводит к уменьшению количества данных, которые могут быть записаны на ленту.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Допустимы следующие значения:

Значение	Значение
140	Режим 5 ГБ (со сжатием)
21	Режим 5 ГБ без сжатия
20	Режим 2.3 ГБ
0	По умолчанию (режим 5,0 ГБ)

По умолчанию применяется значение 140 для Параметра плотности #1 и 20 для Параметра плотности #2. Значение 21 для Параметра плотности #1 или #2 позволяет пользователю читать или записывать данные на магнитную ленту в режиме 5 без сжатия.

Сжатие данных
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 8-мм магнитной ленты емкостью 5,0 ГБ, то атрибуты Перемотка, Поддержка резервирования и Размер блока переменной длины имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на магнитной ленте емкостью 20000 МБ размером 8 мм (самонастраивающиеся):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 20000 МБ размером 8 мм (самонастраивающиеся).

Размер блока
Значение по умолчанию - 1024.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Устройство может читать и записывать данные в формате 20,0 ГБ. При выполнении команды чтения накопитель на магнитной ленте автоматически определяет, в каком формате записаны данные на ленте. При выполнении команды записи формат данных определяется Параметром плотности.

Допустимы следующие значения:

Значение	Значение
39	Режим 20 ГБ (со сжатием)
0	По умолчанию (режим 20.0 ГБ)

Значение по умолчанию для параметров плотности #1 и #2 - 39.

Сжатие данных
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 8-мм магнитной ленты емкостью 20,0 ГБ, то атрибуты Перемотка, Поддержка резервирования и Размер блока переменной длины имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на магнитной ленте емкостью 35 ГБ (тип 35gb):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 35 ГБ (тип 35gb).

Размер блока
Производительность устройства IBM 7205 Model 311 зависит от размера блока. Рекомендуемый минимальный размер блока для данного устройства составляет 32 КБ. Любой блок размером меньше 32 КБ ограничивает скорость передачи данных (время резервного копирования или восстановления). В следующей таблице приведены рекомендуемые размеры блока для различных команд:

Поддерживаемая команда	Размер блока по умолчанию (в байтах)	Рекомендация
BACKUP	32K или 51.2K (по умолчанию)	Выбирается 32 КБ или 51,2 КБ в зависимости от того, выполняется ли `Backup` по имени или нет. Изменять этот параметр не нужно.
TAR	10 Кб	В Руководстве указан ошибочный размер блока, равный 512 КБ. Присвойте параметру **объединение записей в блоки** значение **-N64**.
MKSYSB	См. **BACKUP**	Команда **MKSYSB** использует команду **BACKUP**. Изменять этот параметр не нужно.
DD	н/д	Присвойте параметру **объединение записей в блоки** значение **bs=32K**.
CPIO	н/д	Присвойте параметру **объединение записей в блоки** значение **-C64**.

Примечание: Задавая размер блока, вы должны иметь представление о емкости и производительности устройства. Использование маленьких блоков сильно влияет на быстродействие и слабо - на емкость носителя. Емкость в форматах 2,6 ГБ (плотность) и 6,0 ГБ (плотность) зависит от размера блока, если он меньше рекомендуемого. Например, при использовании 1024-байтовых блоков для резервного копирования 32 ГБ данных требуется приблизительно 22 часа. Резервное копирование тех же 32 ГБ с использованием блоков с размером 32 КБ займет всего 2 часа.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
На диаграмме показан тип кассеты с данными и Параметр плотности (в десятичном и шестнадцатеричном виде) для накопителя на магнитной ленте IBM 7205-311. Во время восстановления (чтения) устройство автоматически выбирает плотность, соответствующую плотности записи данных. Выполняя операцию резервного копирования (записи) необходимо задать

плотность, соответствующую используемой кассете.

Поддерживаемые кассеты	Стандартная емкость	Емкость при сжатии данных	SMIT	HEX
DLTtape III	2.6 ГБ	2,6 ГБ (без сжатия)	23	17h
	6.0 ГБ	6,0 ГБ (без сжатия)	24	18h
	10.0 ГБ	20,0 ГБ (по умолчанию для устройства)	25	19h
DLTtapeIIIxt	15.0 ГБ	30.6 ГБ (по умолчанию для устройства)	25	19h
DLTtapeIV	20.0 ГБ	40.0 ГБ	26	1Ah
	35.0 ГБ	70,0 ГБ (по умолчанию для устройства)	27	1Bh

Примечание: Если стандартная емкость для кассеты не поддерживается, то устройство устанавливает для нее наивысшую поддерживаемую емкость.

Сжатие данных
Обычно степень сжатия зависит от типа записываемых данных (см. таблицу, приведенную выше). Для данного значения емкости в режиме сжатия данных предполагается степень сжатия 2:1.

Атрибуты с фиксированными значениями
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты накопителей на магнитной ленте емкостью 150 МБ размером 1/4 дюйма (тип 150mb):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 150 МБ размером 1/4 дюйма (тип 150mb).

Размер блока
Размер блока по умолчанию - 512. Допустимо еще одно значение размера блока, равное 0. Оно соответствует блокам переменной длины.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Запись на 1/4-дюймовую ленту допустима только с начала ленты (BOT) или только после того, как будет установлено, что на ленте нет данных. Если на ленте есть данные, вы не сможете заменить их, за исключением случая начала с BOT. Если вы хотите добавить данные на установленную на начало ленту, на которой уже есть данные, вам необходимо перемотать ее вперед до следующей метки файла, что приведет к выдаче системного сообщения об ошибке. Только после этого вы сможете вновь начать запись на ленту.

Перемотка
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Допустимы следующие значения:

Значение	Значение
16	QIC-150
15	QIC-120
0	По умолчанию (QIC-150) или последнее значение плотности, установленное в системе.

По умолчанию применяется значение 16 для Параметра плотности #1 и 15 для Параметра плотности #2.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 1/4-дюймовой магнитной ленты

емкостью 150 МБ, то атрибуты Расширенные метки файлов, Поддержка резервирования, Размер блока переменной длины и Сжатие данных имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на магнитной ленте емкостью 525 МБ размером 1/4 дюйма (тип 525mb):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 525 МБ размером 1/4 дюйма (тип 525mb).

Размер блока
Размер блока по умолчанию - 512. Другие допустимые значения размера блока: 0 (для блоков переменной длины) и 1024.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Запись на 1/4-дюймовую ленту допустима только с начала ленты (BOT) или только после того, как будет установлено, что на ленте нет данных. Если на ленте есть данные, вы не сможете заменить их, за исключением случая начала с BOT. Если вы хотите добавить данные на установленную на начало ленту, на которой уже есть данные, вам необходимо перемотать ее вперед до следующей метки файла, что приведет к выдаче системного сообщения об ошибке. Только после этого вы сможете вновь начать запись на ленту.

Перемотка
Запись на 1/4-дюймовую ленту допустима только с начала ленты (BOT) или только после того, как будет установлено, что на ленте нет данных. Если на ленте есть данные, вы не сможете заменить их, за исключением случая начала с BOT. Если вы хотите добавить данные на установленную на начало ленту, на которой уже есть данные, вам необходимо перемотать ее вперед до следующей метки файла, что приведет к выдаче системного сообщения об ошибке. Только после этого вы сможете вновь начать запись на ленту.

Параметр плотности #1 и параметр плотности #2
Допустимы следующие значения:

Значение	Значение
17	QIC-525*
16	QIC-150
15	QIC-120
0	По умолчанию (QIC-525) или последнее значение плотности, установленное в системе.

* QIC-525 - единственный режим, поддерживающий размер блока 1024.

По умолчанию применяется значение 17 для Параметра плотности #1 и 16 для Параметра плотности #2.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 1/4-дюймовой магнитной ленты емкостью 525 МБ, то атрибуты Расширенные метки файлов, Поддержка резервирования, Размер блока переменной длины и Сжатие данных имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на магнитной ленте емкостью 1200 МБ размером 1/4 дюйма (тип 1200mb-c):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 1200 МБ размером 1/4 дюйма (тип 1200mb-c).

Размер блока
Размер блока по умолчанию - 512. Другие допустимые значения размера блока: 0 (для блоков переменной длины) и 1024.

Буферы устройств

Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов

Запись на 1/4-дюймовую ленту допустима только с начала ленты (BOT) или только после того, как будет установлено, что на ленте нет данных. Если на ленте есть данные, вы не сможете заменить их, за исключением случая начала с BOT. Если вы хотите добавить данные на установленную на начало ленту, на которой уже есть данные, вам необходимо перемотать ее вперед до следующей метки файла, что приведет к выдаче системного сообщения об ошибке. Только после этого вы сможете вновь начать запись на ленту.

Перемотка

Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2

Допустимы следующие значения:

Значение	Значение
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	По умолчанию (QIC-1000) или последнее значение плотности, установленное в системе.

* Размер блока 1024 поддерживается только в режимах QIC-525 и QIC-1000.

По умолчанию применяется значение 21 для Параметра плотности #1 и 17 для Параметра плотности #2.

Атрибуты с фиксированными значениями

Если накопитель на магнитной ленте настроен как устройство для 1/4-дюймовой магнитной ленты емкостью 1200 МБ, то атрибуты Расширенные метки файлов, Поддержка резервирования, Размер блока переменной длины и Сжатие данных имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на магнитной ленте емкостью 12000 Мб размером 4 мм (самонастраивающиеся):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 12000 Мб размером 4 мм (самонастраивающиеся).

Размер блока

Производительность накопителя на магнитной ленте IBM емкостью 12000 Мб и размером 4 мм зависит от размера блока. Рекомендуемый минимальный размер блока для данного устройства составляет 32 КБ. Любой блок размером меньше 32 КБ ограничивает скорость передачи данных (время резервного копирования или восстановления). В следующей таблице приведены рекомендуемые размеры блока для различных команд:

Поддерживаемая команда	Размер блока по умолчанию (в байтах)	Рекомендация
BACKUP	32K или 51.2K (по умолчанию)	Выбирается 32 КБ или 51,2 КБ в зависимости от того, выполняется ли `Backup` по имени или нет. Изменять этот параметр не нужно.
TAR	10 Кб	В Руководстве указан ошибочный размер блока, равный 512 КБ. Присвойте параметру **объединение записей в блоки** значение **-N64**.
MKSYSB	См. BACKUP	Команда **MKSYSB** использует команду **BACKUP**. Изменять этот параметр не нужно.
DD		Установите для параметра **объединения записей в блоки** значение `bs=32K`.

Поддерживаемая команда CPIO	Размер блока по умолчанию (в байтах)	Рекомендация
		Установите для параметра **объединения записей в блоки** значение `-C64`.

Примечание: Задавая размер блока, вы должны иметь представление о емкости и производительности устройства. Использование маленьких блоков сильно влияет на быстродействие и слабо - на емкость носителя.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
На рисунке показан тип кассеты с данными и Параметр плотности (в десятичном и шестнадцатеричном виде) для накопителя IBM 12000 MB 4 mm. Во время восстановления (чтения) устройство автоматически выбирает плотность, соответствующую плотности записи данных. Во время резервного копирования (записи) вы должны задать значение плотности, соответствующее применяемой кассете.

Поддер- живаемые кассеты	Стан- дартная емкость	Емкость при сжатии данных	SMIT	HEX
DDS III	2.0 ГБ	4.0 ГБ	19	13h
DDS2	4.0 ГБ	8.0 ГБ	36	24h
DDS3	12.0 ГБ	24.0 ГБ	37	25h

Примечание: Если стандартная емкость для кассеты не поддерживается, то устройство устанавливает для нее наивысшую поддерживаемую емкость.

Сжатие данных
Обычно степень сжатия зависит от типа записываемых данных (см. таблицу, приведенную выше). Для данного значения емкости в режиме сжатия данных предполагается степень сжатия 2:1.

Атрибуты с фиксированными значениями
Для этого атрибута справедливо общее описание, которое было приведено выше.

Атрибуты накопителей на магнитной ленте емкостью 13000 МБ размером 1/4 дюйма (самонастраивающиеся):

Ниже описаны атрибуты накопителей на магнитной ленте емкостью 13000 МБ размером 1/4 дюйма (самонастраивающиеся).

Размер блока
Размер блока по умолчанию - 512. Другие допустимые значения размера блока: 0 (для блоков переменной длины) и 1024.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Запись на 1/4-дюймовую ленту допустима только с начала ленты (BOT) или только после того, как будет установлено, что на ленте нет данных. Если на ленте есть данные, вы не сможете заменить их, за исключением случая начала с BOT. Если вы хотите добавить данные на установленную на начало ленту, на которой уже есть данные, вам необходимо перемотать ее вперед до следующей метки файла, что приведет к выдаче системного сообщения об ошибке. Только после этого вы сможете вновь начать запись на ленту.

Перемотка
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Допустимы следующие значения:

Значение	Значение
33	QIC-5010-DC*
34	QIC-2GB*
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	По умолчанию (QIC-5010-DC)*

* Размер блока 1024 поддерживается только в режимах QIC-525, QIC-1000, QIC-5010-DC и QIC-2GB.

По умолчанию значение параметра плотности #1 равно 33, а параметра плотности #2 - 34.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 1/4-дюймовой магнитной ленты емкостью 13000 МБ, то атрибуты **Расширенные метки файлов**, **Поддержка резервирования** и **Размер блока переменной длины** имеют предопределенные значения, которые нельзя изменить.

Атрибуты 1/2-дюймовых 9-дорожечных накопителей на магнитной ленте (тип 9trk):

Ниже описаны атрибуты 1/2-дюймовых 9-дорожечных накопителей на магнитной ленте (тип 9trk).

Размер блока
По умолчанию применяется размер блока, равный 1024.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Допустимы следующие значения:

Значение	Значение
3	6250 бит на дюйм (bpi)
2	1600 bpi
0	Указанная ранее плотность записи

По умолчанию для Параметра плотности #1 применяется значение 3, а для Параметра плотности #2 - значение 2.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как устройство для 1/2-дюймовой магнитной ленты с 9 дорожками, то атрибуты Расширенные метки файлов, Перемотка, Поддержка резервирования, Размер блока переменной длины и Сжатие данных имеют предопределенные значения, которые не могут быть изменены.

Атрибуты картриджа 3490e 1/2 дюйма (тип 3490e):

Ниже описаны атрибуты картриджа 3490e 1/2 дюйма (тип 3490e).

Размер блока
Размер блока по умолчанию - 1024. Данное устройство обладает высоким быстродействием, поэтому размер блока сильно влияет на эффективность работы. Использование блоков большего размера может значительно увеличить быстродействие, поэтому необходимо использовать наибольший возможный размер блока.

Примечание: Увеличение размера блока может привести к несовместимости с другими программами в вашей системе. Если это произошло, то при выполнении таких программ вы получите следующее сообщение об ошибке:

```
При обращении к системе получен недопустимый параметр.
```

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Сжатие
Для этого атрибута справедливо общее описание, которое было приведено выше.

Автозагрузчик
Данное устройство имеет автоматический механизм (автозагрузчик) для последовательной загрузки и удаления кассет из кассетной стойки. Для того чтобы эти операции выполнялись правильно, переключатель на передней панели должен быть установлен в положение AUTO, а для атрибута Автозагрузчик задано значение `yes`.

Атрибуты прочих типов накопителей на магнитной ленте SCSI (тип ost):

Ниже описаны атрибуты прочих накопителей SCSI на магнитной ленте (тип ost).

Размер блока
По умолчанию система устанавливает значение 512, но оно должно быть заменено на значение по умолчанию для вашего накопителя. Стандартные значения - 512 и 1024. Устройства для 8-мм и 4-мм лент обычно используют значение 1024; если для атрибута размера блока оставить значение 512, то пространство на ленте будет расходоваться нерационально. Значение 0 указывает на переменный размер блока.

Буферы устройств
Для этого атрибута справедливо общее описание, которое было приведено выше.

Расширенные метки файлов
Для этого атрибута справедливо общее описание, которое было приведено выше.

Параметр плотности #1 и параметр плотности #2
Для обоих параметров значение по умолчанию равно 0. Другие значения и их интерпретация зависят от типа накопителя.

Поддержка резервирования
Значение по умолчанию — no. Если устройство поддерживает команды резервирования и освобождения, то значение атрибута можно изменить на yes. Если вы не уверены, то лучше указать no.

Размер блока постоянной длины
По умолчанию размер блока переменной длины равен 0. Ненулевые значения используются главным образом для 1/4-дюймовой кассеты (QIC). Информация об оптимальном размере блока для конкретного накопителя приведена в спецификации интерфейса SCSI.

Время между повторами
Этот атрибут допустим только для накопителей на магнитной ленте устройств типа ost.

Тайм-аут чтения/записи
Этот атрибут допустим только для накопителей на магнитной ленте устройств типа ost.

Атрибуты с фиксированными значениями
Если накопитель на магнитной ленте настроен как Другой накопитель на магнитной ленте SCSI, то атрибуты Расширенные метки файлов, Перемотка и Сжатие данных имеют предопределенные значения, которые не могут быть изменены.

Атрибуты накопителей на магнитной ленте MPIO
Накопители на магнитной ленте с поддержкой MPIO имеют ряд дополнительных атрибутов, находящихся в группе атрибутов устройств MPIO.

Понятия, связанные с данным:

“Разветвленный ввод-вывод” на стр. 538
Функция разветвленного ввода-вывода (MPIO) позволяет однозначно диагностировать устройство через одно или несколько физических соединений, называемых также *путями*.

Специальные файлы для накопителей на магнитной ленте

С каждым накопителем на магнитной ленте, известным операционной системе, связано несколько специальных файлов.

Запись в файл и чтение из файла на магнитной ленте выполняется с помощью специальных файлов `rmt`. Это файлы `/dev/rmt*`, `/dev/rmt*.1`, `/dev/rmt*.2`, ... `/dev/rmt*.7`. `rmt*` - это логическое имя накопителя, например, `rmt0`, `rmt1` и т.д.

Выбирая один из специальных файлов, связанных с накопителем, вы тем самым определяете, как будут выполняться операции ввода/вывода для этого устройства.

Элемент	Описание
Плотность	Вы можете задать для накопителя на магнитной ленте два параметра плотности записи: Параметр плотности #1 и Параметр плотности #2. Эти параметры представляют собой часть атрибутов накопителя. Поскольку достаточно удобно, если Параметр плотности #1 имеет значение наибольшей возможной плотности записи для накопителя, а Параметр плотности #2 - следующее возможное значение плотности, то иногда специальные файлы, использующие Параметр плотности #1, рассматриваются как файлы высокой плотности, а специальные файлы, использующие Параметр плотности #2 - файлы низкой плотности, но это не всегда правильно. При чтении с ленты параметр плотности игнорируется.
Перемотка-при-закрытии	Вы можете указать, нужно ли перематывать магнитную ленту к началу, если специальный файл, обращающийся к ленте, закрывается. Если выбрана опция перемотки, то при закрытии файла лента перематывается к началу.
Перемотка-при-открытии	Вы можете указать, нужно ли перематывать магнитную ленту при открытии файла. В данном случае под перемоткой ленты понимается ее перемотка до конца, а затем обратная перемотка к началу. Эта операция позволяет добиться большей равномерности натяжения магнитной ленты и снижения количества ошибок. Если выбрана опция перемотки при открытии, то при открытии файла лента перематывается к началу.

В следующей таблице приведены имена специальных файлов `rmt` и их характеристики.

Специальный файл	Перемотка при закрытии	Перемотка при открытии	Параметр плотности
`/dev/rmt*`	Да	Нет	#1
`/dev/rmt*.1`	Нет	Нет	#1
`/dev/rmt*.2`	Да	Да	#1
`/dev/rmt*.3`	Нет	Да	#1
`/dev/rmt*.4`	Да	Нет	#2
`/dev/rmt*.5`	Нет	Нет	#2
`/dev/rmt*.6`	Да	Да	#2
`/dev/rmt*.7`	Нет	Да	#2

Предположим, что вы хотите записать три файла на магнитную ленту, загруженную в накопитель rmt2. Первый файл будет располагаться в начале ленты, за ним - второй файл, за вторым - третий. Далее, предположим, что для накопителя вы хотите использовать параметр плотности #1. Для записи на ленту можно использовать следующий список специальных файлов, в приведенном ниже порядке.

1. `/dev/rmt2.3`
2. `/dev/rmt2.1`
3. `/dev/rmt2`

Почему выбраны именно эти специальные файлы?

- Для первого файла выбрано устройство `/dev/rmt2.3`, так как оно перематывает ленту при открытии файла, гарантируя, что первый файл будет записан в начало ленты. Опция перемотки при закрытии не используется, поскольку следующая операция ввода/вывода начинается в том месте, где завершается этот файл. Если лента уже перемотана к началу, когда открывается первый файл, быстрее будет использовать в качестве первого файла `/dev/rmt2.1`, так как в при этом экономится время на перемотку ленты.

- Для второго файла выбрано устройство `/dev/rmt2.1`, так как оно не перематывает ленту ни при открытии, ни при закрытии файла. Нет никакой необходимости в переходе к началу ленты ни когда файл открывается, ни когда он закрывается.
- Для третьего файла выбрано устройство `/dev/rmt2`, так как третий файл следует за вторым и перемотка ленты при открытии файла не требуется. Поскольку после записи на ленту третьего файла больше никаких операций не планируется, выбирается опция перемотки к началу и закрытия. При следующем обращении к магнитной ленте ее нужно будет начать читать с начала.

При работе с магнитной лентой вместо специальных файлов `rmt` можно использовать команду **tctl**.

Поддержка USB-устройств

Операционная система AIX поддерживает USB-устройства.

Операционная система AIX поддерживает следующие типы USB-устройств компании IBM и сторонних производителей:

- Флэш-накопитель
- Дисковый накопитель
- Оптический накопитель (Blu-ray, DVD или CD-ROM)
- Ленточный накопитель
- Клавиатура
- Мышь
- Колонки

Примечание: Операционная система AIX не поддерживает USB-принтеры.

Некоторые USB-устройства сторонних производителей могут не распознаваться в операционной системе AIX. Например, порт USB в системах Power Systems является слаботочным. Из-за этого операционная система AIX поддерживает не все имеющиеся на рынке USB-устройства.

Поддержка накопителей USB

Начиная с версии AIX 5.3 с технологическим пакетом обслуживания 5300-09 и AIX 6.1 с технологическим пакетом обслуживания 6100-02, реализована поддержка флэш-накопителей с интерфейсом универсальной последовательной шины (USB).

Поддержка этих устройств входит в состав следующего пакета:

```
devices.usbif.08025002
```

Поддержка флэш-накопителей USB в AIX проверяется с использованием выборки флэш-накопителей USB, которые производятся поставщиками, соблюдающими отраслевые стандарты. Драйверы устройств для хост-контроллера USB в AIX поддерживают стандарт USB 2.0. Накопителям USB присваиваются логические имена (например, `usbms0` и `usbms1`), представляющие как символьные, так и блочные специальные файлы. Например, специальным сырым файлом для `usbms0` является `/dev/rusbms0`, а специальным блочным файлом — `/dev/usbms0`. До AIX версии 5.3 с технологическим пакетом обслуживания 5300-11 и AIX версии 6.1 с технологическим пакетом обслуживания 6100-04 флэш-накопители USB настраивались как `/dev/flashdrive0`.

На єтих накопителях поддерживается файловая система Международная организация по стандартизации (ISO) (ISO 9660 только для чтения). Для создания резервной копии системы на приводах используется команда **tar**, команда **cpio** или архивы резервного копирования или восстановления. Можно также с помощью команды **dd** добавить образы ISO к дискам.

Операционная система AIX не поддерживает технологию plug-and-play для USB-накопителей. Для того чтобы предоставить пользователям AIX доступ к накопителю USB, пользователь root должен подключить его к разъему USB и выполнить следующую команду:

```
cfgmgr -l usb0
```

Внимание: Будьте осторожны при удалении флэш-накопителей из портов. Если диски не закрыты правильно или не размонтированы перед их удалением, данные на них могут быть повреждены.

После удаления накопителей они остаются в администраторе объектных данных (ODM) в доступном состоянии, пока пользователь root не запустит следующую команду:

```
rmdev -l usbms<i>n</i>
```

Пока накопитель находится в доступном состоянии, его можно подключить заново и смонтировать или открыть. Если в момент извлечения накопитель был открыт пользователем, то он будет доступен для повторного использования только после того, как пользователь закроет его и откроет заново.

Поддержка накопителя USB Blu-ray только для чтения

Версии AIX версии 6.1 с технологическим пакетом обслуживания 6100-06 и выше распознают и настраивают приводы Blu-ray, подключенные через USB.

Эта функциональная возможность включена в следующий пакет устройств:

```
devices.usbif.08025002
```

Способность операционной системы AIX читать носитель Blu-ray проверяется на образце отраслевых стандартных накопителей производителя оборудования (OEM) USB Blu-ray.

Накопители USB Blu-ray настраиваются с использованием логических имен, например `cd0` и `cd1`. Эти накопители содержат специальные как сырые, так и блочные файлы. Например, необработанный обычный файл для `cd0` - `/dev/rcd0`, а специальный блочный файл - `/dev/cd0`.

Опция только для чтения предусматривается для файловой системы Международной организации по стандартизации (ISO) (ISO 9660 только для чтения), файловой системы UDF (версия 2.01 или ниже) и стандартных команд доступа к оптическим носителям, например **dd** и **tar**.

Операционная система AIX не поддерживает операцию записи на носители CD, DVD или Blu-ray, находящиеся в приводе USB Blu-ray. Хотя операция записи и не запрещена (если накопитель имеет такую возможность), IBM не предоставляет поддержки в случае неполадок при выполнении операции записи.

Операционная система AIX не поддерживает технологию plug-and-play для USB-накопителей Blu-ray. Для того чтобы сделать накопитель USB Blu-ray доступным пользователям AIX, пользователь root должен подключить накопитель к порту USB системы и выполнить следующую команду:

```
cfgmgr -l usb0
```

После удаления накопителя он остается в доступном состоянии в базе данных администратора объектных данных (ODM), пока пользователь root не выполнит следующую команду:

```
rmdev -l cd<i>n</i>
```

Пока накопитель находится в доступном состоянии, его можно заново подключить к системе. Если в момент извлечения накопитель был открыт пользователем, то он будет доступен только после того, как пользователь закроет его и откроет заново.

Кэширование данных памяти

Устройства кэширования поддерживают кэширование данных памяти на уровне сервера.

Устройствами кэша могут быть устройства следующих типов:

- Подключенные к серверу флэш-устройства, такие как встроенные в сервер SSD.
- Флэш-устройства, напрямую подключенные к серверу с помощью контроллеров SAS.
- Флэш-ресурсы в сети хранения данных (SAN).

Концепция кэширования данных памяти

Кэшировать данные памяти можно динамически (включать или останавливать кэширование) во время выполнения задачи. Для выполнения операции кэширования не требуется переводить задачу в неактивное состояние.

Для объяснения концепции кэширования применяются следующие термины:

Устройство кэша
Устройство кэша - это твердотельный накопитель (SSD) или флэш-диск, используемый для кэширования.

Пул кэша
Пул кэша - это группа устройств кэша, используемое только для кэширования памяти.

Раздел кэша
Раздел кэша - это логическое устройство кэша, создаваемое из пула кэша.

Целевое устройство
Целевое устройство - это кэшируемое запоминающее устройство.

Один раздел кэша можно применять для кэширования одного или нескольких целевых устройств. После создания кэша целевого устройства все запросы на чтение блоков этого устройства перенаправляются в кэширующее программное обеспечение. Если конкретный блок найден в кэше, то запрос ввода-вывода обрабатывается для устройства кэша. Если блок в кэше не найден, либо если выполнялся запрос на запись, запрос ввода-вывода возвращается в целевое устройство.

Преимущества кэширования данных памяти

Кэширование данных памяти на сервере может повысить плотность виртуализации, особенно для перегруженной подсистемы хранения данных.

Кэширование данных памяти имеет следующие преимущества:

Задержка
Аналитические и транзакционные задачи имеют уменьшенное время запрос-ответ из-за меньших задержек хранилища SSD. При использовании кэширования на сервере средняя задержка транзакционной задачи может уменьшиться вдвое.

Производительность
Для задач OLTP повышается скорость транзакций из-за более высокой производительности дисков SSD.

Производительность операций записи
В среде с перегруженной сетью хранения данных (SAN) использование флэш-диска в качестве кэша может значительно снизить нагрузку запросов чтения. После снижения нагрузки запросов записи может повыситься производительность записи SAN. и она сможет более эффективно обслуживать большое число клиентов и хостов.

Меньшие потребности в памяти
После настройки флэш-диска в качестве кэша некоторые задачи могут работать с меньшими потребностями в памяти.

Ограничения кэширования данных памяти

Убедитесь в понимании ограничений и дополнительных требованиях к конфигурации функции кэширования. Необходимо также учесть ограничения приложения для кэшируемых целевых устройств.

Примите к сведению следующие ограничения при кэшировании данных памяти:

- Кэширующее программное обеспечение настроено как кэш, доступный только для чтения. Это означает, обрабатываться будут только запросы на чтение флэш-накопителя SSD. Все запросы на запись будут обрабатываться исходным запоминающим устройством.
- Данные, записанные в запоминающее устройство, не записываются в кэш автоматически. Если операция записи выполняется для блока, находящегося в кэше, существующие данные в кэше помечаются как недопустимые. Время, через которое этот блок будет снова добавлен в кэш, зависит от частоты его использования.
- В каждом логическом разделе (LPAR) AIX требуется дополнительная память, так как кэширующее программное обеспечение управляет метаданными каждого блока чтения. Для любого LPAR с включенным кэшированием требуется не меньше 4 ГБ памяти.
- Кэширующее программное обеспечение загружает данные в кэш на основе локальных шаблонов чтения т помечает записи кэша как недопустимые тоже локально. Целевые устройства не должны использоваться совместно больше чем одним LPAR одновременно. Целевые устройства не должны входить в состав кластерного хранилища, такого как Oracle RAC, DB2 pureScale и GPFS. Целевые устройства из кластера с высокой готовностью могут кэшироваться только при условии, что доступ к ним организован таким образом, что в каждый момент времени только один хост считывает и записывает данные на целевом устройстве, и кэширование включено только на активном узле.
- Диск кэша можно назначить либо LPAR AIX, либо LPAR виртуального сервера ввода-вывода (VIOS). Устройства кэша нельзя использовать совместно.
- Кэширующее программное обеспечение должно открывать целевые устройства для перехвата всех запросов ввода-вывода к ним. Если задаче требуется открыть целевое устройство с исключительными правами доступа после запуска кэширования, операция исключительного открытия не будет выполнена. В таких случаях необходимо остановить кэширование и запустить его после запуска задачи.
- Если диски используются в качестве целевых устройств, то атрибуту диска **reserve_policy** не должно быть присвоено значение `single_path`.
- После запуска операции кэширования для целевого устройства логика модуля кэша начинает запись данных в кэш только после тайм-аута. Эта задержка необходима для гарантированного завершения всех ожидающих операций ввода-вывода. запущенных до операции кэширования. Точное время задержки вычисляется системой на основе числа доступных путей и атрибута **rw_timeout** (если он задан) целевого диска. Если вычисленное системой время требуется изменить на пользовательское. можно в файле `/etc/environment` присвоить переменной окружения *DEFAULT_IO_DRAIN_TIMEOUT_PD* пользовательское значение тайм-аута в секундах.

Компоненты кэширования данных памяти

Кэширующее программное обеспечение состоит из компонентов управления кэшем и модуля кэша.

Управление кэшем

Кэшированием данных памяти можно управлять с помощью команды **cache_mgt**, доступной в операционной системе AIX и в виртуальном сервере ввода-вывода (VIOS). Команда **cache_mgt** позволяет выполнить следующие задачи:

- Создать пул кэша и разделы на нем.
- Назначить раздел кэша целевому устройству или логическому разделу AIX (LPAR) в качестве виртуального устройства vSCSI.
- Запустить и остановить операцию кэширования.

Модуль кэша

Модуль кэша - это основная часть кэширующего программного обеспечения. Модуль кэша определяет, какие блоки памяти необходимо занести в кэш, а также из какого расположения извлекать данные - из кэша или из основной памяти.

Алгоритм кэширования основан на механизме заполнение-при-чтении, заносящий данные в кэш по пространственному признаку (находящиеся рядом с блоками, для которых выполнялись последние операции чтения). Алгоритм кэширования заносит данные в кэш быстрее, чем кэш освобождается.

Для всех блоков в кэше постоянно выполняется проверка частоты их чтения, на основе результатов этой проверки создается карта горячих зон. Эта карта учитывает и частоту, и давность обращений. Если кэш заполнен, то новые записи будут добавляться в него только в том случае, если значимость нового блока выше блока с самой низкой значимостью в кэше. Блок с самой низкой значимостью будет удален из кэша, а новая запись добавлена.

Агрессивное заполнение обеспечивает короткое время повышения значимости, что гарантирует эффективность включенного кэша. Основанная на карте горячих зон стратегия удаления обеспечивает динамичность кэша и постоянную адаптацию к изменяющимся требованиям задач.

Настройка кэширования данных памяти

В операционной системе AIX кэширование сервера устройств с флэш-памятью поддерживается в нескольких разных конфигурациях. Эти конфигурации различаются способом назначения устройства кэша логическому разделу (LPAR) AIX.

Кэширование сервера поддерживает в операционной системе AIX следующие режимы:

- Выделенный режим
- Виртуальный режим
- Режим виртуализации ИД N_Port (NPIV)

Кэширование данных памяти в выделенном режиме:

В выделенном режиме устройство кэша назначено непосредственно логическому разделу AIX (LPAR).

Необходимо создать пул кэша, а затем раздел кэша на устройстве кэша. В выделенном устройстве кэша можно создать только один раздел кэша. Раздел кэша можно использовать для любого числа целевых устройств в LPAR AIX. LPAR не является мобильным, так как этому LPAR выделено устройство кэша. Если требуется перенести LPAR на другой сервер, сначала необходимо вручную остановить кэширование и отменить настройку устройства кэша.

На следующем рисунке показан пример конфигурации кэширования в LPAR AIX для выделенного устройства кэша.

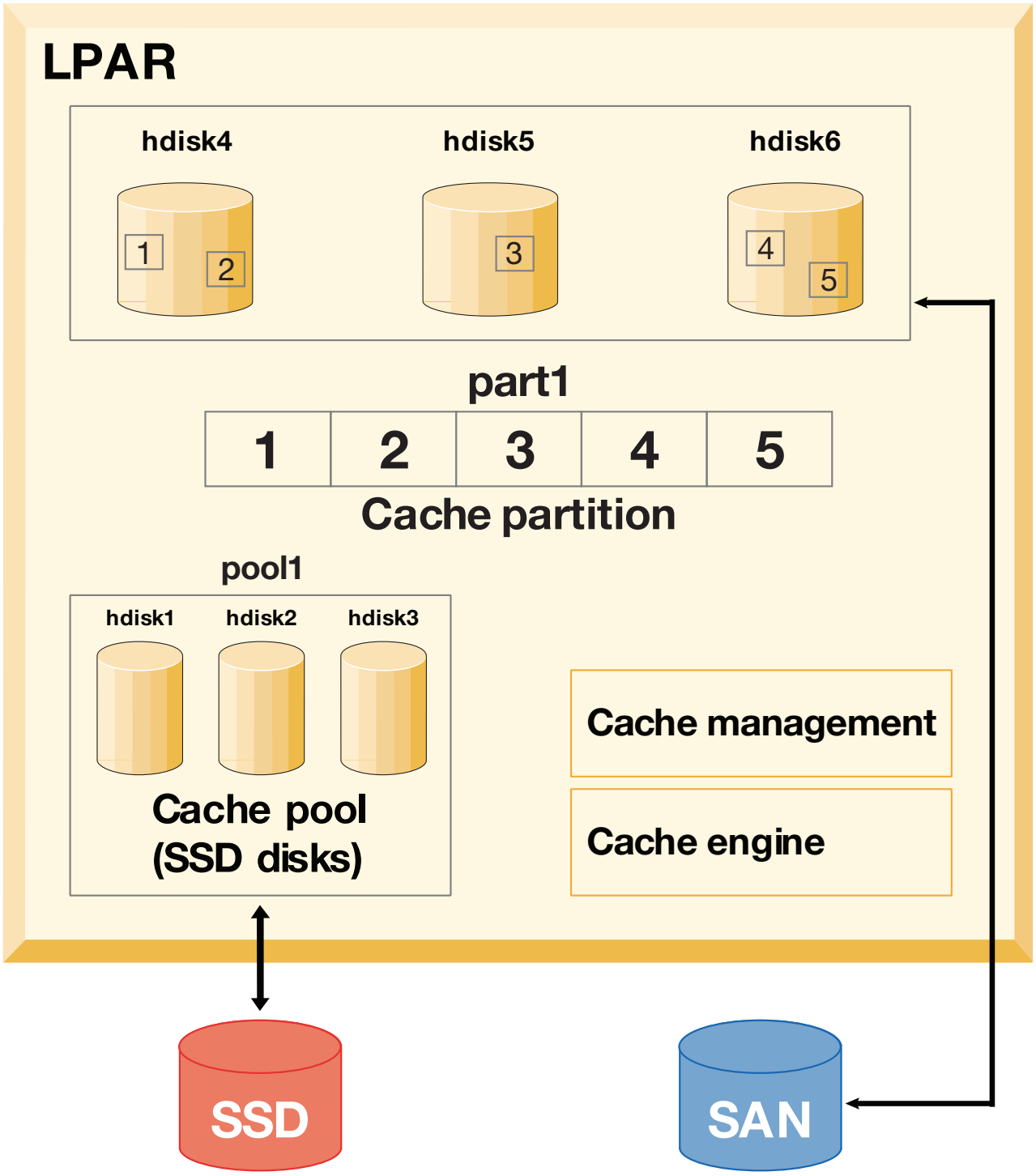


Рисунок 15. Кэширование данных памяти: конфигурация для выделенного устройства кэша

Устройства кэша: `hdisk1`, `hdisk2` и `hdisk3`, целевые устройства: `hdisk4`, `hdisk5` и `hdisk6`. Для запуска и отслеживания кэширования на целевых устройствах выполните следующие действия:

1. Создайте на устройстве SSD пул кэша.

   ```
   # cache_mgt pool create -d hdisk1,hdisk2,hdisk3 -p cmpool0
   ```

2. Создайте в пуле кэша раздел кэша размером 80 МБ.

   ```
   # cache_mgt partition create -p cmpool0 -s 80M -P part1
   ```

3. Назначьте раздел кэша целевым дискам, которые требуется кэшировать.

```
# cache_mgt partition assign -t hdisk4 -P part1
# cache_mgt partition assign -t hdisk5 -P part1
# cache_mgt partition assign -t hdisk6 -P part1
```

4. Запустите кэширование целевого устройства.

```
# cache_mgt cache start -t hdisk4
# cache_mgt cache start -t hdisk5
# cache_mgt cache start -t hdisk6
```

5. Запустите мониторинг статистики попаданий в кэш.

```
# cache_mgt monitor get -h -s
```

Информация, связанная с данной:

команда cache_mgt

Кэширование данных памяти в виртуальном режиме:

В виртуальном режиме устройство кэша назначается виртуальному серверу ввода-вывода (VIOS).

В виртуальном режиме пул кэша создается в VIOS. Затем пул кэша разделяется на разделы в VIOS. Каждый раздел кэша можно назначить адаптеру виртуального хоста (vhost). Если раздел кэша находится на логическом разделе (LPAR) AIX, то раздел кэша можно использовать для кэширования целевого устройства. Раздел кэша можно перенести на другой сервер, так как устройство кэша является виртуальным. Перед миграцией процесс кэширования в исходном LPAR автоматически останавливается. В рамках операции миграции на целевом VIOS динамически создается раздел кэша такого же размера (если на целевом VIOS установлено кэширующее ПО и доступен пул кэша). Во время миграции раздел становится доступным для LPAR. После завершения миграции процесс кэширования автоматически будет запущен в целевом LPAR. В этом случае кэширование запускается в пустом (незаполненном) состоянии.

На следующем рисунке показан пример конфигурации кэширования в виртуальном режиме. Здесь устройство кэша расположено в LPAR VIOS, а целевое устройство - в LPAR AIX.

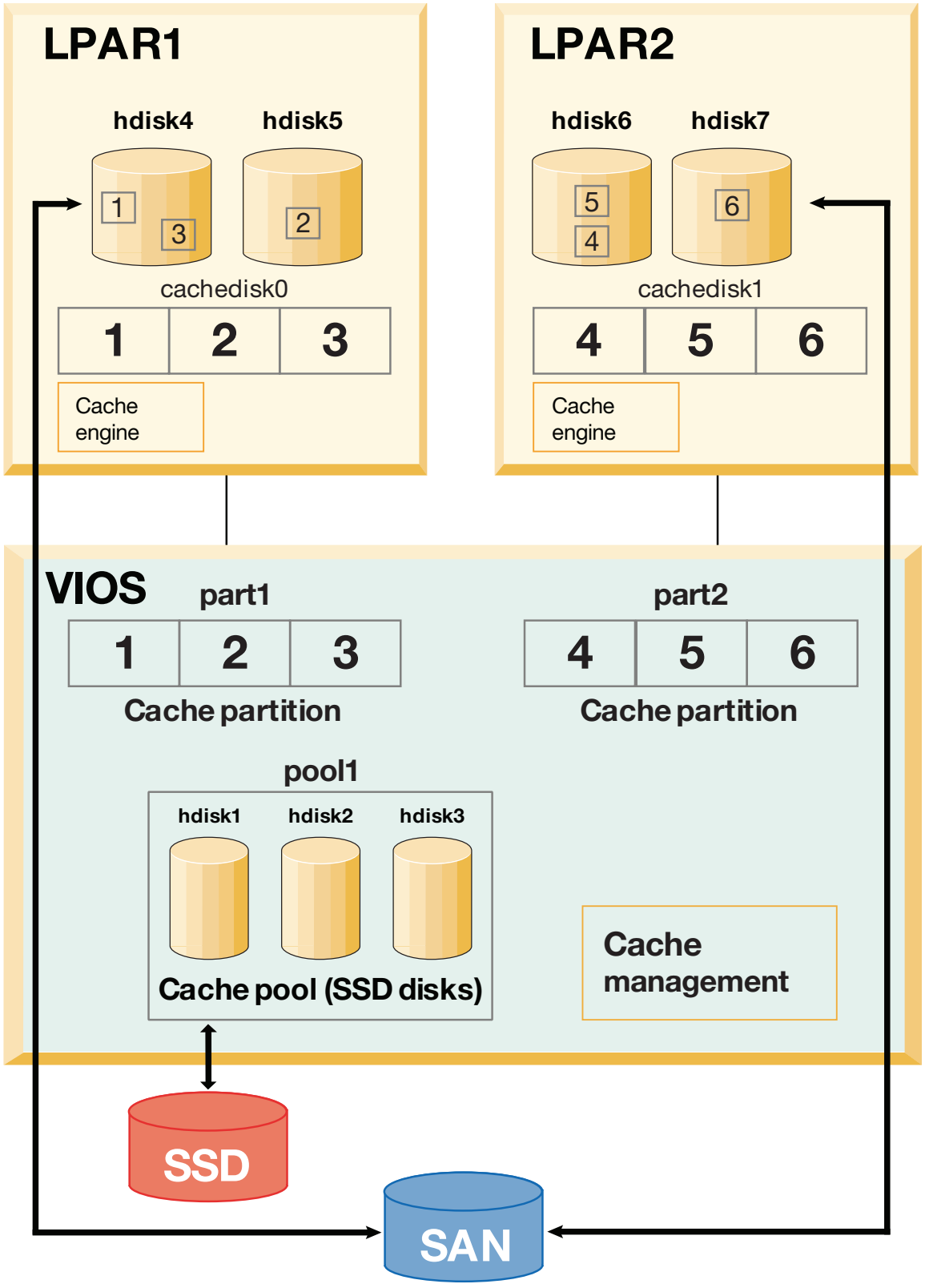


Рисунок 16. Кэширование данных памяти: конфигурация для виртуального устройства кэша

Устройства кэша: `hdisk1`, `hdisk2` и `hdisk3` (в LPAR VIOS), целевые устройства: `hdisk4` и `hdisk5` (в LPAR AIX). Для запуска и отслеживания кэширования на целевых устройствах выполните следующие действия:

1. В LPAR VIOS создайте пул кэша на устройстве SSD.

```
# cache_mgt pool create -d hdisk1,hdisk2,hdisk3 -p cmpool0
```

2. В LPAR VIOS создайте на основе пула кэша раздел кэша размером 80 МБ.
```
# cache_mgt partition create -p cmpool0 -s 80M -P part1
```
3. В LPAR VIOS назначьте раздел адаптеру виртуального хоста.
```
# cache_mgt partition assign -P part1 -v vhost0
```
4. В LPAR AIX назначьте раздел кэша целевым устройствам.
```
# cache_mgt partition assign -t hdisk4 -P cachedisk0
# cache_mgt partition assign -t hdisk5 -P cachedisk0
```
5. В LPAR AIX запустите кэширование целевых устройств.
```
# cache_mgt cache start -t hdisk4
# cache_mgt cache start -t hdisk5
```
6. В LPAR AIX отслеживайте статистику попаданий в кэш.
```
# cache_mgt monitor get -h -s
```

Информация, связанная с данной:

команда cache_mgt

команда cache_mgt в VIOS

Кэширование данных памяти в режиме NPIV:

В этом режиме устройство кэша доступно в качестве виртуального устройства Fibre Channel (виртуализация ИД N_Port - N_Port ID Virtualization) в логическом разделе AIX (LPAR).

Необходимо создать пул кэша, а затем раздел кэша в LPAR AIX. В LPAR AIX можно создать только один раздел кэша. Раздел кэша можно использовать для любого числа целевых устройств в LPAR AIX. LPAR можно перенести на другой сервер, так как устройство кэша доступно в сети хранения данных (SAN). Устройство кэша должно быть видимым в целевой системе. Операция кэширования может продолжать выполнение во время миграции, и кэш будет заполнен после окончания миграции.

Кэширование целевого устройства в режиме NPIV аналогично кэшированию данных памяти в выделенном режиме, за исключением того, что устройство кэша доступно в SAN. Однако процедура кэширования целевого устройства аналогична кэшированию данных памяти в выделенном режиме.

Управление кэшем данных памяти

Хотя кэш настроен, требования к кэшированию со временем могут измениться. Может потребоваться добавить для кэширования новые задачи. Для реализации изменившихся требований можно расширить пул кэша с помощью дополнительных устройств кэша, создать раздел кэша в существующем пуле или увеличить размер существующего раздела.

Для управления конфигурацией кэширования можно использовать следующие примеры:

- Для добавления устройства кэша в пул введите следующую команду:
```
# cache_mgt pool extend -p pool1 -d hdisk4 -f
```
  Флаг **-f** заменяет любой существующий формат диска (hdisk4). если он содержит существующую группу томов.
- Чтобы создать для новой задачи раздел размером 100 МБ, введите следующую команду:
```
# cache_mgt partition create -p pool1 -s 100M -P part2
```
- Для увеличения существующего раздела на 20 МБ введите следующую команду:
```
# cache_mgt  partition extend -p part1 -s 20M
```

Замечания о высокой готовности:

Если целевые устройства, для которых выполняется кэширование, входят в состав группы ресурсов, управляемой в кластере высокой готовности, то требуется должным образом запланировать операцию аварийного переключения.

В один момент времени можно активировать кэширование только на одном узле. Перед инициацией любого события аварийного переключения необходимо убедиться, что кэширование отключено в исходной системе. После завершения аварийного переключения на альтернативную систему необходимо вручную включить кэширующее программное обеспечение.

Для включения кэширующего программного обеспечения выполните следующие действия:

1. Остановите кэширование исходной системы.
   ```
   # cache_mgt cache stop -t hdisk2
   ```
2. Запустите кэширование в новой системе после завершения аварийного восстановления.
   ```
   # cache_mgt cache start -t hdisk2
   ```

Мониторинг статистики кэша

Для просмотра статистики кэша отдельного целевого устройства применяется команда **cache_mgt monitor get**.

Например, если кэшируется только целевое устройство `hdisk1`, вывод команды **cache_mgt** аналогичен показанному в следующем примере:

```
# cache_mgt monitor get -h -s
Статистика ввода-вывода устройства ETS -- hdisk1
Время начала сбора статистики -- Пнд, 27 мар 2017, 07:10:41
------------------------------------------------------
 Число операций чтения:                    152125803
 Число операций записи:                     79353626
 Число попаданий при чтении:                     871
 Число частичных попаданий при чтении:            63
 Передано байт при чтении:            10963365477376
 Передано байт при записи:             4506245999616
 Передано байт попаданий при чтении:       48398336
 Передано байт частичных попаданий:         5768192
 Число операций чтения с заполнением:    2033078104
 Передано байт чтения с заполнением:532959226494976
```

Команда **cache_mgt** позволяет узнать следующие показатели кэширования:

Число операций чтения
: Общее число всех операций чтения с целевого устройства, инициированных всеми приложениями. Это значение учитывает операции чтения с устройства кэша, выполняемые, когда данные доступны на устройстве кэша. Данное значение отражает общее количество отдельных запросов на чтение, а не их размер (кол-во байт).

Число операций записи
: Общее количество операций записи на целевое устройство. Данное значение отражает общее количество отдельных запросов на запись, а не их размер (кол-во байт).

Число попаданий при чтении
: Общее число операций чтения с целевого устройства, которые можно считать *полными попаданиями. Полное попадание* - это экземпляр операции чтения, в которой запрос на чтение полностью выполняется с устройства кэша. Значение **Число попаданий при чтении** отражает общее количество отдельных запросов на чтение с попаданием в кэш, а не размер этих запросов на чтение (кол-во байт). Данное значение включается в **Число операций чтения**.

Число частичных попаданий при чтении
: Общее число операций чтения с целевого устройства, которые можно считать *частичными*

попаданиями. *Частичное попадание* - это экземпляр операции чтения, в которой запрос на чтение частично выполняется с устройства кэша. Те данные, которые недоступны на устройстве кэша, запрашиваются с целевого устройства. **Число частичных попаданий при чтении** отражает общее количество отдельных запросов на чтение, а не их размер (кол-во байт). Данное значение включается в **Число операций чтения**.

Передано байт при чтении
Общее число байт, переданных при обработке всех запросов на чтение с целевого устройства, инициированных приложениями. Это значение включает в себя общий размер в байтах *полных попаданий*, *частичных попаданий* и всех данных, запрошенных с целевого устройства.

Передано байт при записи
Общее число байт, переданных при обработке всех запросов на запись на целевое устройство, инициированных приложениями.

Передано байт попаданий при чтении
Общее число байт, считанных во всех случаях *полного попадания*.

Передано байт частичных попаданий
Общее число байт, считанных во всех случаях *частичного попадания*.

Число операций чтения с заполнением
Общее число команд чтения с целевого устройства, при выполнении которых данные добавлялись на устройство кэша. Это значение не отражает число случаев, когда данные добавлялись на устройство кэша, так как один запрос на запись данных в устройство кэша может быть разделен на несколько операций чтения, если максимальный размер данных, передаваемых на целевой диск, меньше размера запроса.

Передано байт чтения с заполнением
Общее число байт, прочитанных с целевого устройства, когда данные добавлялись на устройство кэша.

Имена пользователей, системные идентификаторы и пароли

Для правильной настройки вашей рабочей среды операционной системе необходимо знать, кто вы.

Для идентификации себя в системе введите *ИД пользователя* (имя пользователя) и *пароль*. Пароли - это один из способов защиты данных. Другие пользователи не смогут войти в систему под вашим именем, если они не знают ваш пароль.

Если в системе разрешена работа нескольких пользователей, то для каждого пользователя создается учетная запись, пароль и идентификатор. Операционная система отслеживает ресурсы, применяемые каждым пользователем. Эта функция называется *системным учетом*. Каждому пользователю выделяется отдельная область в памяти, называемая *файловой системой*. С точки зрения пользователя, вошедшего в систему, файловая система содержит только файлы этого пользователя, хотя на самом деле в системе хранятся тысячи других файлов.

Вы можете создать для себя несколько идентификаторов. Для того чтобы перейти от одного имени (идентификатора) к другому, не нужно завершать работу в системе. Достаточно запустить различные оболочки или изменить свое имя в текущей оболочке, не выходя из системы. Кроме того, если ваша система объединена в сеть с другими системами, вы можете входить в любую другую систему, в которой для вас есть учетная запись. Такая операция называется *удаленным входом в систему*.

Закончив работу в операционной системе, выйдите из нее. Это позволит гарантировать, что ваши файлы и данные не будут повреждены.

Вход в операционные системы

Перед тем как начать работу с операционной системой, необходимо включить компьютер и войти в систему. При входе в систему вы указываете свой идентификатор, и система создает для вас рабочую среду.

Конфигурация системы может быть задана таким образом, что вход в нее будет разрешен только в определенные часы и дни недели. Если вы попытаетесь войти в систему в другое время, то попытка окажется неудачной. Выяснить, в какие часы разрешен вход в систему, можно у системного администратора.

При входе в систему вы вводите свой идентификатор и пароль. После входа в систему текущим каталогом автоматически становится ваш домашний каталог (называемый также *начальным каталогом*).

Если компьютер включен, просто войдите в систему - сеанс будет запущен.

1. В приглашении **пользователь:** введите свой идентификатор пользователя:
   ```
   пользователь: ИД_пользователя
   ```
   Пример:
   ```
   login: denise
   ```
2. Если появится приглашение **пароль:**, введите пароль. (При вводе пароль не будет показан на экране.)
   ```
   пароль: [пароль]
   ```

Отсутствие приглашения на ввод пароля означает, что пароль не нужен; вы можете приступить к работе в операционной системе.

Если компьютер не включен, то перед входом в систему выполните следующие действия:

1. Включите питание всех подключенных устройств.
2. Включите системный блок, нажав на кнопку питания (**I**).
3. Посмотрите на трехсимвольный индикатор. Если при выполнении процедуры начального тестирования не возникнет ошибок, индикатор будет пустым.

Если произойдет ошибка, требующая вмешательства оператора, то на индикаторе будет продолжать высвечиваться трехзначный код, а работа системного блока прервется. Информацию об ошибках с различными кодами и действиях по их исправлению можно получить у системного администратора.

После завершения процедуры начального тестирования на экране появится примерно следующее приглашение для входа в систему:

```
пользователь:
```

После входа в систему будет запущен интерфейс командной строки (оболочка) или графический интерфейс (например, AIXwindows или Общая среда рабочего стола (CDE)), в зависимости от конфигурации системы.

За информацией о настройке пароля и имени пользователя обратитесь к системному администратору.

Запуск нескольких сеансов (команда login)

Если для работы над несколькими проектами были созданы разные учетные записи, вы можете запустить сразу несколько сеансов. При этом для входа в систему может применяться одно и то же имя или разные имена пользователей.

Примечание: В каждой системе задано ограничение на количество одновременно работающих пользователей. Это ограничение определяется лицензионным соглашением и зависит от конкретной конфигурации системы.

Например, если вы уже вошли в систему как пользователь `denise1`, и хотите запустить второй сеанс от имени `denise2`, введите в командной строке:

```
login denise2
```

Если появится приглашение **пароль:**, введите пароль. (При вводе пароль не будет показан на экране.) Теперь у вас запущено два сеанса.

Сведения о синтаксисе приведены в описании команды **login** книги *Справочник по командам, том 3*.

Переключение на другого пользователя (команда su)

Команда **su** позволяет изменить ИД пользователя, связанный с сеансом.

Например, для того чтобы переключиться на пользователя `joyce`, введите в командной строке:

```
su joyce
```

Если появится приглашение **пароль:**, введите пароль пользователя `joyce`. Теперь вы работаете в системе под именем `joyce`. Если вы не знаете пароль, запрос будет отклонен.

Вы можете убедиться, что текущий ИД пользователя - `joyce`, введя команду **id**.

Понятия, связанные с данным:

"Отображение ИД пользователей"
Для просмотра системного идентификатора (ИД) указанного пользователя введите команду **id**. Системные идентификаторы - это номера, идентифицирующие пользователей и группы пользователей в системе.

Информация, связанная с данной:

Синтаксис команды su

Подавление сообщений при входе в систему

После входа в систему команда **login** выдаст сообщение дня, дату и время последней удачной и неудачной попыток входа в систему с этим ИД пользователя, а также общее число неудачных попыток входа в систему с этим ИД пользователя с момента последнего изменения идентификационной информации (т.е. пароля). Для того чтобы эта информация не выводилась на экран, поместите в свой домашний каталог файл `.hushlogin`.

Введите в командной строке домашнего каталога следующую команду:

```
touch .hushlogin
```

Команда **touch** создаст пустой файл с именем `.hushlogin`, если он не существует. При следующем входе в систему сообщения выдаваться не будут. Вы можете оставить только сообщение дня и отменить выдачу остальных сообщений.

Информация, связанная с данной:

touch, команда

Выход из системы (команды exit и logout)

Для того чтобы выйти из системы, выполните одно из следующих действий в командной строке.

Нажмите управляющие клавиши для ввода символа конца файла (Ctrl-D).

ИЛИ

Введите `exit`.

ИЛИ

Введите `logout`.

После выхода из системы появится приглашение **пользователь:**.

Отображение ИД пользователей

Для просмотра системного идентификатора (ИД) указанного пользователя введите команду **id**. Системные идентификаторы - это номера, идентифицирующие пользователей и группы пользователей в системе.

Команда **id** выдает следующую информацию (если она задана):

- Имя пользователя и фактический ИД пользователя
- Имя группы пользователей и фактические ИД группы
- Имена и идентификаторы дополнительных групп пользователя, если они есть

Например, введите в командной строке:

```
id
```

Появится приблизительно следующая информация:

```
uid=1544(sah) gid=300(build) euid=0(root) egid=9(printq) groups=0(system),10(audit)
```

В этом примере имя пользователя - `sah`, идентификатор - `1544`; имя основной группы - `build`, идентификатор - `300`; действующее имя пользователя - `root`, идентификатор - `0`; действующее имя группы - `printq`, идентификатор - `9`; имена дополнительных групп - `system` и `audit`, идентификаторы - `0` и `10`, соответственно.

Например, введите в командной строке:

```
id denise
```

Появится приблизительно следующая информация:

```
uid=2988(denise) gid=1(staff)
```

В этом примере идентификатор пользователя `denise` - `2988`, имя его основной группы - `staff`, идентификатор группы - `1`.

Сведения о синтаксисе приведены в описании команды **id** книги *Справочник по командам, том 3*.

Задачи, связанные с данной:

“Переключение на другого пользователя (команда su)” на стр. 587
Команда **su** позволяет изменить ИД пользователя, связанный с сеансом.

Просмотр имени пользователя (команды whoami и logname):

Если вы одновременно запустили несколько сеансов, то вполне можете забыть, под каким именем работаете в данный момент. Для отображения данной информации можно использовать команды **whoami** и **logname**.

Применение команды whoami

Для просмотра текущего имени пользователя введите следующую команду:

```
whoami
```

Появится приблизительно следующая информация:

```
denise
```

В данном примере ИД пользователя равно `denise`.

Сведения о синтаксисе приведены в описании команды **whoami** книги *Справочник по командам, том 6*.

Применение команды who am i

Команда **who am i**, представляющая собой разновидность команды **who**, позволяет просмотреть ИД пользователя, имя терминала и время входа в систему. Введите в командной строке:

```
who am i
```

Появится приблизительно следующая информация:

```
denise   pts/0   21 июня 07:53
```

В этом примере ИД пользователя - `denise`, имя терминала - `pts/0`, дата и время входа пользователя в систему - 21 июня, 7:53.

Сведения о синтаксисе приведены в описании команды **who** книги *Справочник по командам, том 6*.

Применение команды logname

Другая разновидность команды **who**, команда **logname**, выдает ту же информацию, что и команда **who**.

Введите в командной строке:

```
logname
```

Появится приблизительно следующая информация:

```
denise
```

В этом примере ИД пользователя - `denise`.

Просмотр имени операционной системы (команда uname):

Для того чтобы просмотреть название операционной системы, введите команду **uname**.

Например, введите в командной строке:

```
uname
```

Появится приблизительно следующая информация:

```
AIX
```

В этом примере название операционной системы - AIX.

Сведения о синтаксисе приведены в описании команды **uname** книги *Справочник по командам, том 5*.

Просмотр имени системы (команда uname):

Для того чтобы просмотреть имя системы, подключенной к сети, введите команду **uname** с флагом **-n**. Имя системы применяется для идентификации системы в сети. Это не то же самое, что ИД пользователя.

Например, введите в командной строке:

```
uname -n
```

Появится приблизительно следующая информация:

```
barnard
```

В этом примере имя системы - `barnard`.

Сведения о синтаксисе приведены в описании команды **uname** в книге *Справочник по командам, том 5*.

Просмотр всех пользователей, работающих в системе:

Для просмотра списка всех пользователей, работающих в локальной системе, введите команду **who**.

Будет показана следующая информация: ИД пользователя, имя системы, дата и время входа в систему.

Примечание: Эта команда выдает информацию только о пользователях локального узла.

Для просмотра информации о пользователях локальной системы введите:

```
who
```

Появится приблизительно следующая информация:

```
joe   lft/0 8 июня 08:34
denise   pts/1 8 июня 07:07
```

В этом примере пользователь `joe` на терминале `lft/0` вошел в систему 8 июня в 8:34.

См. описание команды **who**.

Пароль

Пароль позволяет защитить файлы от несанкционированного доступа.

С каждой учетной записью в системе связан пароль. Защита играет важную роль в вычислительных системах, так как она предотвращает несанкционированный доступ к системе. Кроме того, с помощью функции защиты некоторым пользователям можно предоставить исключительные права на применение тех или иных команд и файлов. Иногда системные администраторы ограничивают набор команд и файлов, которые будут доступны пользователям, с целью повысить защищенность системы.

Советы по выбору пароля:

Ваш пароль должен быть уникальным. *Пароли не должны использоваться несколькими пользователями.* Пароли следует оберегать не менее тщательно, чем обычное имущество компании. Создаваемый пароль должен быть одновременно трудным для угадывания и легким для запоминания.

Чем сложнее пароль, тем надежнее защищен ваш ИД пользователя. Не рекомендуется выбирать пароли, совпадающие с вашим именем или днем рождения, а также с наиболее употребительными словами - такие пароли легко угадать.

Рекомендуется указывать пароли, состоящие из не менее чем шести символов и содержащие небуквенные символы. Кроме того, удачными паролями считаются необычные словосочетания и слова, намеренно написанные с ошибкой.

Примечание: Если пароль слишком труден для запоминания и вам приходится его записывать, выберите другой пароль.

При выборе пароля руководствуйтесь следующими инструкциями:

- Не указывайте свой ИД пользователя, а также всевозможные его модификации (в обратном порядке, удвоенный) в качестве пароля.
- Не используйте пароли повторно. Повторное использование паролей может быть запрещено конфигурацией системы.
- Не указывайте в качестве паролей личные имена.
- Не указывайте в качестве пароля слова, хранящиеся в электронных орфографических словарях.
- Длина пароля должна составлять не менее шести символов.
- Не указывайте в качестве паролей ругательства; при угадывании паролей их пробуют прежде всего.
- Выбирайте легко запоминающиеся пароли, чтобы вам не пришлось их записывать.
- Выбирайте пароли, содержащие цифры, а также строчные и прописные буквы.
- Рекомендуется задавать пароли, состоящие из двух слов, разделенных цифрами.
- Выбирайте легко произносимые пароли. Их легче запомнить.
- Не записывайте пароль. Если все же возникает необходимость записать его, поместите запись в надежное место, например в сейф.

Изменение паролей (команда passwd):

Свой пароль можно изменить с помощью команды **passwd**.

1. Введите в командной строке:

   ```
   passwd
   ```

   Если у вас нет пароля, перейдите к шагу 2.
2. Появится следующее приглашение:

   ```
   Изменение пароля для  ИД-пользователя
   Старый пароль для ИД-пользователя:
   ```

   Этот запрос не позволяет другому пользователю изменить ваш пароль. Введите текущий пароль и нажмите Enter.
3. Появится следующее приглашение:

   ```
   Новый пароль ИД-пользователя:
   ```

   Введите новый пароль и нажмите Enter.
4. Будет показано следующее приглашение для подтверждения пароля.

   ```
   Введите новый пароль еще раз:
   ```

   Этот запрос позволяет избежать случайных ошибок при вводе пароля.

Сведения о синтаксисе приведены в описании команды **passwd** в книге *Справочник по командам, том 4*.

Обнуление пароля (команда passwd):

Если вы не хотите вводить пароль при каждом входе в систему, установите пустой пароль.

Для установки пустого пароля текущего пользователя введите:

```
passwd
```

В приглашении на ввод нового пароля нажмите Enter или Ctrl-D.

Команда **passwd** не будет запрашивать пароль еще раз. Появится сообщение, подтверждающее сброс пароля.

Дополнительные сведения и описание синтаксиса приведены в описании команды **passwd**.

Обзор команд работы с именами пользователей, системными идентификаторами и паролями для входа в систему

Имеются команды для работы с именами пользователей, ИД систем и паролями.

Команды входа и выхода из системы

Элемент	Описание
login	Запускает сеанс
logout	Завершает все процессы
shutdown	Завершает работу системы
su	Изменяет ИД пользователя, связанный с сеансом
touch	Обновляет время изменения файла и время обращения к файлу, либо создает пустой файл

Команды идентификации пользователя и системы

Элемент	Описание
id	Показывает системную идентификационную информацию об указанном пользователе
logname	Показывает имя пользователя, под которым он вошел в систему.
uname	Показывает имя текущей операционной системы
who	Показывает список пользователей, работающих в системе
whoami	Показывает текущее имя пользователя, под которым вы работаете

Команда работы с паролем

Элемент	Описание
passwd	Изменяет пароль пользователя

Common Desktop Environment

С помощью Общая среда рабочего стола (CDE) можно получить доступ к сетевым устройствам и средствам без необходимости знать о их расположении. Можно обмениваться данными между приложениями, просто перенося объекты мышью.

Многие задачи, которые раньше требовали сложного синтаксиса командной строки, можно выполнить более легко и единообразно на разных платформах. Например, можно централизованно настраивать и распространять приложения среди пользователей. Можно также централизованно управлять защитой, доступностью и функциональной совместимостью приложений у поддерживаемых вами пользователей.

Примечание: Справка Общая среда рабочего стола (CDE) 1.0, электронная документация в Интернет и печатные копии руководств могут ссылаться на настольную систему как на Общую настольную среду, настольную систему AIXwindows, CDE 1.0 или просто настольную систему.

Включение и отключение автоматического запуска рабочего стола

Рекомендуется настроить систему таким образом, чтобы Общая среда рабочего стола автоматически запускалась при включении системы.

Это можно сделать с помощью Инструмента управления системой (программы SMIT) или командной строки.

`Предварительные требования`

Включать и выключать режим автоматического запуска рабочего стола может только пользователь root.

Сведения о включении и отключении автоматического запуска рабочего стола приведены в следующей таблице.

Автоматический запуск и завершение работы Общей среды рабочего стола

Задача	Команда SMIT	Команда или файл
Включение режима автоматического запуска рабочего стола[1]	**smit dtconfig**	**/usr/dt/bin/dtconfig -e**
Выключение режима автоматического запуска рабочего стола[1]	**smit dtconfig**	**/usr/dt/bin/dtconfig -d**

Примечание: После выполнения этой задачи перезагрузите систему.

Запуск общей среды рабочего стола вручную

С помощью этой процедуры можно запусти общую среду рабочего стола вручную.

1. Войдите в систему в качестве пользователя root.
2. Введите в командной строке:
   ```
   /usr/dt/bin/dtlogin -daemon
   ```

Появится окно **Вход в систему рабочего стола**. После входа в систему запускается сеанс рабочего стола.

Завершение работы общей среды рабочего стола вручную

Когда работа Администратора входа в систему завершается вручную, то завершается работа всех серверов X и сеансов рабочего стола, запущенных этим администратором.

1. Откройте окно эмулятора терминала и войдите в систему как пользователь root.
2. Узнайте ИД процесса Администратора входа в систему с помощью следующей команды:
   ```
   cat /var/dt/Xpid
   ```
3. Завершите работы Администратора входа в систему с помощью следующей команды:
   ```
   kill -term ИД-процесса
   ```

Изменение профайла рабочего стола

При входе пользователя в систему рабочего стола файл с параметрами среды оболочки (`.profile` или `.login`) не считывается. Перед входом пользователя в систему рабочий стол запускает X-сервер, поэтому функции файла `.profile` или `.login` должен выполнять Администратор входа в систему рабочего стола.

Пользовательские переменные среды задаются в файле */домашний-каталог/* `.dtprofile`. Шаблон этого файла хранится в файле `/usr/dt/config/sys.dtprofile`. Задайте в файле `.dtprofile` те переменные среды и команды оболочки, которые относятся к рабочему столу. Для того чтобы включить в файл `.dtprofile` переменные среды оболочки, добавьте строки в конец файла.

Глобальные переменные среды можно задать в файлах конфигурации Администратора входа в систему. За информацией о настройке переменных среды обратитесь к руководству *Общая среда рабочего стола 1.0: Расширенный справочник пользователя и системного администратора*.

Добавление и удаление дисплеев и терминалов для Common Desktop Environment

Можно добавить и удалить дисплеи и терминалы для Common Desktop Environment.

Администратор входа в систему можно запустить в системе с одной локальной растровой или графической консолью. Кроме того, допустимы другие конфигурации (см. следующий рисунок). Запустить Common Desktop Environment можно из:

- Локальной консоли
- Удаленной консоли
- Графическом или текстовом терминале X, работающем на хосте в сети

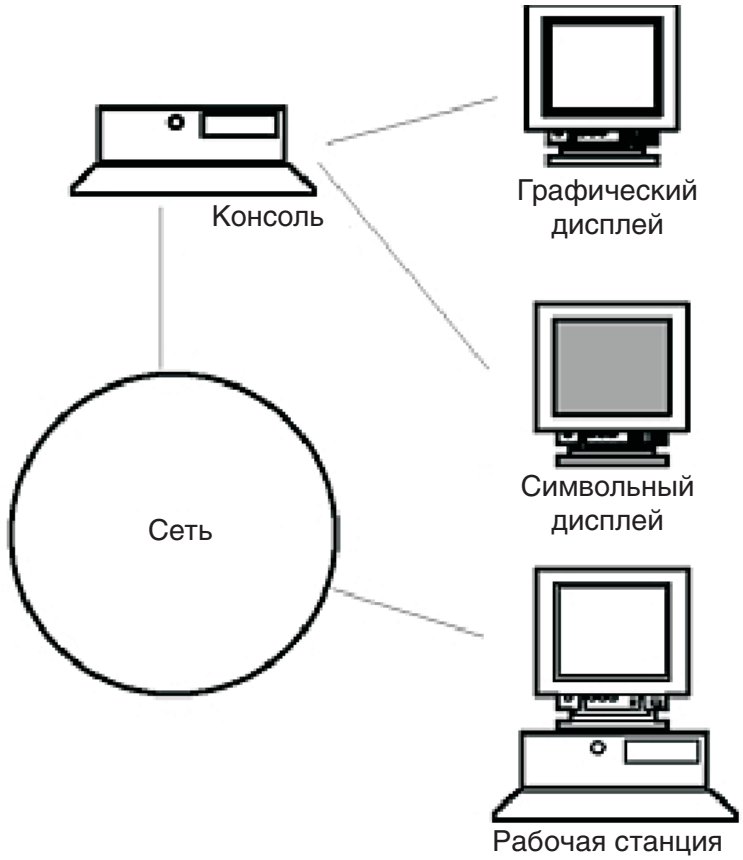


Рисунок 17. Точки интерфейса CDE. На этом рисунке показаны соединения между консолью, сетью, растровым дисплеем, символьным дисплеем и рабочей станцией.

Система X состоит из дисплея, клавиатуры и мыши. Она предназначена только для работы терминала X. Клиенты, в том числе Common Desktop Environment, расположены на одном или нескольких хостах сети. Вывод клиентов отправляется на дисплей X.

Следующие задачи настройки Администратора входа в систему поддерживают различные конфигурации:

- Удаление локального дисплея
- Добавление терминала ASCII или текстового терминала

Для применения рабочей станции в качестве терминала X, введите следующую команду в командной строке:

```
/usr/bin/X11/X -query имя-хоста
```

К серверу X той рабочей станции, которая играет роль терминала X, предъявляются следующие требования:

- Он должен поддерживать XDMCP и опцию командной строки **-query**.
- Он должен предоставлять хосту терминала права доступа xhost (эти права доступа задаются в файле `/etc/X*.hosts`).

Для удаления локального дисплея удалите запись об этом дисплее из файла `Xservers` в каталоге `/usr/dt/config`.

Текстовый терминал или *терминал ASCII* - это терминал, который не является растровым устройством.

Добавление консоли ASCII (текстовой консоли) при отсутствии растрового дисплея:

1. Если файл /etc/dt/config/Xservers не существует, скопируйте файл /usr/dt/config/Xservers в каталог /etc/dt/config.
2. Если файл Xservers был скопирован в каталог /etc/dt/config, добавьте следующую строку Dtlogin.servers: в файл /etc/dt/config/Xconfig или измените ее соответствующим образом:

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```
3. Поместите в комментарий строку файла /etc/dt/config/Xservers, содержащую команду запуска сервера X.

   ```
   # * Local local@console /path/X :0
   ```

   В результате будет отключено **меню ощий входа в систему**.
4. Заново загрузите файлы конфигурации Администратора входа в систему.

Добавление текстовой консоли при наличии растрового дисплея:

1. Если файл /etc/dt/config/Xservers не существует, скопируйте файл /usr/dt/config/Xservers в каталог /etc/dt/config.
2. Если файл Xservers был скопирован в каталог /etc/dt/config, добавьте следующую строку Dtlogin.servers: в файл /etc/dt/config/Xconfig или измените ее соответствующим образом:

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```
3. Измените строку файла /etc/dt/config/Xservers, содержащую команду запуска сервера X, следующим образом:

   ```
   * Local local@none /path/X :0
   ```
4. Заново загрузите файлы конфигурации Администратора входа в систему.

Просмотр настройки устройства для Common Desktop Environment

Администратор входа в систему Common Desktop Environment можно настроить для применения в системах с несколькими дисплейными устройствами.

Если в системе предусмотрено несколько дисплеев, должны быть выполнены следующие требования к конфигурации:

- На каждом дисплее должен быть запущен сервер.
- Для каждого дисплея необходимо настроить режим работы без Windows.

Для дисплеев рекомендуется выделить разные ресурсы dtlogin; в некоторых случаях это условие является обязательным.

Для дисплеев рекомендуется использовать разные глобальные переменные среды; в некоторых случаях это условие является обязательным.

Запуск сервера на всех дисплейных устройствах.:

С помощью этой процедуры можно запустить сервер на всех дисплейных устройствах.

1. Если файл /etc/dt/config/Xservers не существует, скопируйте файл /usr/dt/config/Xservers в каталог /etc/dt/config.
2. Если файл Xservers был скопирован в каталог /etc/dt/config, добавьте следующую строку Dtlogin.servers: в файл /etc/dt/config/Xconfig или измените ее соответствующим образом:

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```
3. Измените файл /etc/dt/config/Xservers таким образом, чтобы X-сервер запускался на всех дисплейных устройствах.

Общий формат команды запуска сервера выглядит следующим образом:

```
имя-дисплея класс-дисплея тип-дисплея [ @ite ] команда
```

В режиме *без Windows* могут работать только те дисплеи, с которыми связан Внутренний эмулятор терминала (ITE). В режиме без Windows временно отключается рабочий стол дисплея и запускается процесс *getty*, если такой процесс еще не активен. Процесс getty - это программа UNIX, определяющая тип терминала и используемая в процессе входа в систему.

Это дает возможность пользователю войти в систему и выполнить ряд задач, недоступных в Common Desktop Environment. После выхода из системы выполняется перезапуск рабочего стола дисплея. При включении режима без Windows Администратор входа в систему запускает процесс getty, если такой процесс еще не запущен.

В конфигурации по умолчанию если параметр `ite` не указан, дисплей `display:0` связан с ITE (`/dev/console`).

Выбор другого дисплея в качестве ITE:

С помощью этой процедуры можно выбрать другой дисплей в качестве ITE.

Выбор другого дисплея в качестве ITE:

- На дисплее ITE укажите в качестве ITE текстовое устройство.
- На всех остальных дисплеях укажите в параметре ITE значение Нет.

Ниже приведен пример записей файла `Xserver` для запуска сервера на трех локальных дисплеях в `sysaaa:0`. Display `:0` будет использоваться в качестве консоли (ITE).

```
sysaaa:0 Local local /usr/bin/X11/X :0
sysaaa:1 Local local /usr/bin/X11/X :1
sysaaa:2 Local local /usr/bin/X11/X :2
```

На хосте sysbbb растровый дисплей `:0` не является ITE; роль ITE играет устройство `/dev/ttyi1`. Ниже приведен пример записей файла `Xserver` для запуска сервера на двух растровых дисплеях; на дисплее `:1` включен режим без Windows.

```
sysaaa:0 Local local@none /usr/bin/X11/X :0
sysaaa:1 Local local@ttyi1 /usr/bin/X11/X :1
```

Настройка отображаемого имени в Xconfig:

В файле `/etc/dt/config/Xconfig` имя дисплея нельзя указывать в обычном формате `hostname:0`.

Выбор дисплея в Xconfig:

- Вместо двоеточия следует указать символ подчеркивания.
- В полном имени хоста точки следует заменить на символы подчеркивания.

В следующем примере показана настройка отображаемого имени в Xconfig:

```
Dtlogin.claaa_0.resource: значение
Dtlogin.sysaaa_prsm_ld_edu_0.resource: значение
```

Применение различных ресурсов Администратора входа в систему для разных дисплеев:

Для применения различных ресурсов Администратора входа в систему для разных дисплеев выполните следующие действия:

1. Если файл `/etc/dt/config/Xconfig` не существует, скопируйте файл `/usr/dt/config/Xconfig` в каталог `/etc/dt/config`.
2. Измените файл `/etc/dt/config/Xconfig`, указав различные файлы ресурсов для каждого дисплея. Например:

   ```
   Dtlogin.DisplayName.resources: путь/файл
   ```

   где *путь* - это имя каталога файла Xresource, который следует применять, а *файл* - имя файла Xresource.

3. Создайте файлы ресурсов, указанные в файле Xconfig. Файл Xresources для установленного языка расположен в каталоге /usr/dt/config/<LANG>.
4. В каждом файле укажите ресурсы dtlogin для конкретного дисплея.

В следующем примере показаны строки файла Xconfig, задающие различные файлы ресурсов для трех дисплеев:

```
Dtlogin.sysaaa_0.resources: /etc/dt/config/Xresources0
Dtlogin.sysaaa_1.resources: /etc/dt/config/Xresources1
Dtlogin.sysaaa_2.resources: /etc/dt/config/Xresources2
```

Выполнение различных сценариев для каждого дисплея:

С помощью этой процедуры можно выполнить определенный сценарий для определенного дисплея.

1. Если файл /etc/dt/config/Xconfig не существует, скопируйте файл /usr/dt/config/Xconfig в каталог /etc/dt/config.
2. Укажите разные сценарии для дисплеев в ресурсах startup, reset и setup файла /etc/dt/config/Xconfig (указанные файлы будут применяться вместо файлов Xstartup, Xreset и Xsetup):

   ```
   Dtlogin*DisplayName*startup: /путь/файл
   Dtlogin*DisplayName*reset: /путь/ файл
   Dtlogin*DisplayName*setup: /путь/ файл
   ```

где *путь* - это имя каталога файла, а *файл* - имя файла. Сценарий запуска выполняется от имени пользователя root после входа пользователя в систему, но до запуска сеанса Common Desktop Environment.

Сценарий /usr/dt/config/Xreset позволяет восстановить значения, измененные в файле Xstartup. Сценарий Xreset запускается при выходе пользователя из системы.

Ниже приведен пример записей файла Xconfig, задающих разные сценарии для двух дисплеев:

```
Dtlogin.sysaaa_0*startup:   /etc/dt/config/Xstartup0
Dtlogin.sysaaa_1*startup:    /etc/dt/config/Xstartup1
Dtlogin.sysaaa_0*setup:      /etc/dt/config/Xsetup0
Dtlogin.sysaaa_1*setup:      /etc/dt/config/Xsetup1
Dtlogin.sysaaa_0*reset:      /etc/dt/config/Xreset0
Dtlogin.sysaaa_1*reset:      /etc/dt/config/Xreset1
```

Настройка разных глобальных переменных среды для дисплеев:

С помощью этой процедуры можно настроить переменные среды уровня системы для каждого дисплея.

Для настройки разных глобальных переменных среды для дисплеев выполните следующие действия:

1. Если файл /etc/dt/config/Xconfig не существует, скопируйте файл /usr/dt/config/Xconfig в каталог /etc/dt/config.
2. В файле /etc/dt/config/Xconfig задайте для каждого дисплея свой ресурс environment:

   ```
   Dtlogin*DisplayName*environment: значение
   ```

При настройке переменных среды для отдельного дисплея следует учесть следующее:

- Разделяйте значения переменных пробелами или символами табуляции.
- Не настраивайте в ресурсе environment переменные **TZ** и **LANG**.
- Файл Xconfig не обрабатывается оболочкой.

В следующем примере приведены строки из файла Xconfig, в которых задаются переменные для двух дисплеев:

```
Dtlogin*syshere_0*environment:EDITOR=vi SB_DISPLAY_ADDR=0xB00000
Dtlogin*syshere_1*environment: EDITOR=emacs \
         SB_DISPLAY_ADDR=0xB00000
```

Печать заданий печати

В AIX существует огромное количество опций настройки печати.

В зависимости от принтера операционная система AIX может управлять внешним видом и параметрами вывода на принтер. Необязательно, чтобы принтеры находились рядом системным блоком и системной консолью. Принтер может быть локальным, т.е. напрямую подключенным к системе, или удаленным - тогда задания печати необходимо передавать по сети в другую систему.

Для того чтобы обрабатывать задания печати с максимальной эффективностью, операционная система AIX помещает их в очередь, где они находятся до тех пор, пока принтер не станет доступным. Система может хранить несколько файлов в очереди. Пока принтер выполняет одно задание, система обрабатывает следующее задание из очереди. Процесс продолжается, пока все задания в очереди не будут обработаны.

Запуск задания печати

Для запуска задачи печати используйте команду **qprt** или **smit**.

- Локальные принтеры должны быть физически подключены к вашей системе, а сетевые принтеры - подключены и настроены для работы в сети.
- Для выполнения заданий удаленной печати необходимо, чтобы в вашей системе была настроена связь с удаленным сервером печати.
- Для печати файла необходимо иметь права на *чтение* этого файла. Для удаления файла после печати необходимы права на *запись* в каталог, содержащий файл.

Для запуска задачи печати укажите следующие сведения:

- Имя файла для печати
- Имя очереди печати
- Число печатаемых копий
- Создавать ли копию файла на удаленном хосте
- Удалять ли файл после печати
- Отправлять ли уведомление о состоянии задания
- Отправлять ли уведомление о состоянии задания по системной почте
- Печатать ли разделительные страницы с перфорацией
- Имя пользователя для поля "Кому"
- Подтверждающее сообщение консоли для удаленной печати
- Регистрировать ли подтверждающее сообщение для удаленной печати
- Приоритет

Для печати указанного файла команда **qprt** создает и помещает в очередь задание печати. Если вы укажете несколько файлов, то для них будет создано одно задание печати. Файлы будут напечатаны в том же порядке, в котором они перечислены в командной строке.

Основной формат команды **qprt** следующий:

```
qprt -Pимя_очереди имя_файла
```

Ниже описаны наиболее употребительные флаги команды **qprt**:

Флаг	Описание
-b *Число*	Задает нижнее поле. Нижнее поле - это число пустых строк в конце каждой страницы.
-B *Значение*	Указывает, будут ли напечатаны разделительные страницы, отделенные перфорацией. Переменная *Значение* состоит из двух символов. Первый символ относится к начальным, второй - к конечным разделительным страницам. Для каждого из двух символов допустимы следующие варианты: **a** Всегда печатать данную разделительную (начальную или конечную) страницу для каждого файла каждого задания печати. **n** Не печатать данную разделительную (начальную или конечную) страницу. **g** Печатать данную разделительную страницу (начальную или конечную) один раз для каждого задания печати (группы файлов). Например, флаг **-B ga** означает, что начальная разделительная страница будет напечатана в начале каждого задания печати, а конечная разделительная страница - в конце каждого файла каждого задания печати. В среде удаленной печати конфигурация по умолчанию определяется удаленной очередью сервера.
-e *Опция*	Позволяет задать или отменить печать с выделением. **+** Разрешает печать с выделением. **!** Запрещает печать с выделением.
-E *Опция*	Позволяет задать или отменить печать с двойной высотой символов. **+** Разрешает печать с двойной высотой символов. **!** Запрещает печать с двойной высотой символов.
-f *Тип_фильтра*	Идентификатор из одного символа, задающий фильтр для обработки печатаемых файлов перед их отправкой на принтер. Допустимы следующие идентификаторы фильтров: **p**, запускающий фильтр **pr**, и **n**, обрабатывающий вывод команды **troff**.
-i *Число*	Задает в начале каждой строки отступ на заданное число пробелов. Переменная *Число* должна быть указана также для ширины страницы, задаваемой флагом **-w**.
-K *Опция*	Позволяет задать или отменить уплотненную печать. **+** Разрешает уплотненную печать. **!** Запрещает уплотненную печать.
-l *Число*	Задает длину страницы в строках. Если *Число* равно нулю, то длина страницы будет проигнорирована, и вывод будет напечатан на бумажной ленте. Длина страницы включает верхнее и нижнее поле и обозначает длину бумаги, отведенной для печати.
-L *Опция*	Указывает, будут ли строки, длина которых превышает ширину страницы, перенесены на следующую строку или усечены по правому полю. **+** Означает, что длинные строки будут перенесены. **!** Означает, что длинные строки будут усечены по правому полю.
-N *Число*	Задает число печатаемых копий. Если этот флаг не задан, то будет напечатана одна копия.
-p *Число*	Задает ширину шрифта в символах на дюйм. Стандартные значения переменной *Число*: 10 и 12. Фактическая ширина шрифта также зависит от флагов **-K** (уплотненная печать) и **-W** (двойная ширина).
-P*Очередь*[: *Связанное_Устройство*]	Задает имя очереди печати, а также, необязательно, имя связанного с ней устройства. Если этот флаг не задан, то будет применен принтер по умолчанию.
-Q *Значение*	Задает размер бумаги для текущего задания печати. *Значение* зависит от конкретного принтера. Стандартные значение: **1** - Letter, **2** - Legal, и т.д. Значения, соответствующие различным форматам бумаги, приведены в руководстве по принтеру.
-t *Число*	Задает верхнее поле. Верхнее поле - это число пустых строк в начале каждой страницы.
-w *Число*	Задает ширину страницы в символах, число которых указано в переменно *Число*. Ширина страница должна включать ширину отступа в пробелах, заданную флагом **-i**.

Флаг	Описание
-W *Опция*	Позволяет задать или отменить печать с двойной шириной символов. **+** Разрешает печать с двойной шириной символов. **!** Запрещает печать с двойной шириной символов.
-z *Значение*	Указывает, сколько *раз* нужно повернуть вывод на 90 градусов по часовой стрелке. Значения длины (**-l**) и ширины (**-w**) страницы при необходимости меняются местами. **0** Книжная ориентация **1** Правая альбомная ориентация **2** Книжная перевернутая ориентация **3** Левая альбомная ориентация
-# Значение	Задает специальную функцию. **j** Показывает номер указанного задания печати. **h** Помещает задание печати в очередь, но переводит его в состояние **HELD**. **v** Проверяет указанные значения флагов базовой программы принтера. Эта проверка полезна, если необходимо найти неверные значения флагов при передаче задания печати на выполнение. Если проверка не задана, то впоследствии неверное значение флага может прервать обработку задания печати.

Ниже приведены примеры применения флагов команды **qprt**:

- Для того чтобы напечатать файл `myfile` на первом доступном принтере, настроенном для очереди печати по умолчанию, со значениями по умолчанию, введите:
  ```
  qprt myfile
  ```
- Для печати файла `myfile` из конкретной очереди с определенными значениями флагов, проверяемыми при передаче задания на выполнение, введите:
  ```
  qprt -f p -e + -Pfastest -# v myfile
  ```
  Файл `myfile` будет обработан командой фильтра **pr** (флаг **-f p**) и напечатан в уплотненном формате (флаг **-e +**) на первом доступном принтере, настроенном для очереди с именем **fastest** (флаг **-Pfastest**).
- Для печати файла `myfile` на бумаге размера Legal введите:
  ```
  qprt -Q2 myfile
  ```
- Для печати трех копий каждого из файлов `new.index.c`, `print.index.c` и `more.c` из очереди печати `Msp1` введите:
  ```
  qprt -PMsp1 -N 3 new.index.c print.index.c more.c
  ```
- Для печати трех копий объединения трех файлов `new.index.c`, `print.index.c` и `more.c` введите:
  ```
  cat new.index.c print.index.c more.c | qprt -PMsp1
  -N 3
  ```

Примечание: В базовой операционной системе поддерживается также команда печати BSD UNIX (**lpr**) и команда печати System V UNIX (**lp**). Полная информация о синтаксисе приведена в описании команды **lpr** и **lp** в книге *Справочник по командам, том 3*.

Полная информация о синтаксисе приведена в описании команды **qprt** в книге *Справочник по командам, том 4*.

Кроме того для выполнения задачи печати можно использовать SMIT. Для того чтобы запустить задание печати с помощью SMIT, введите:

```
smit qprt
```

Отмена задания печати (команды qcan)

С помощью команды qcan можно отменить задание печати.

- Локальные принтеры должны быть физически подключены к вашей системе, а сетевые принтеры - подключены и настроены для работы в сети.
- Для выполнения заданий удаленной печати необходимо, чтобы в вашей системе была настроена связь с удаленным сервером печати.

Для того чтобы отменить задание печати, вам понадобится указать имя очереди печати, в которой находится задание, и его номер.

Эта процедура одинакова для заданий локальной и удаленной печати.

Команда **qcan** позволяет отменить либо одно задание с указанным номером в удаленной или локальной очереди, либо все задания в локальной очереди печати. Выяснить номер задания можно с помощью команды **qchk**.

Основной формат команды **qcan** следующий:

```
qcan -P имя-очереди -x номер-задания
```

Полная информация о синтаксисе приведена в описании команды **qcan** в книге *Справочник по командам, том 4*.

Ниже приведены примеры применения флагов команды **qcan**:

- Отменить задание печати номер **123**, находящееся на любом принтере, можно следующей командой: `qcan -x 123`
- Отменить все задания на принтере **lp0** можно следующей командой: `qcan -X -Plp0`

Примечание: В базовой операционной системе поддерживается также команда отмена печати BSD UNIX (**lprm**) команда отмены печати System V UNIX (**cancel**). Полная информация о синтаксисе приведена в описании **lprm** и **cancel** команд в книге *Справочник по командам*.

Отмена задания печати (SMIT)

Для отмены задания печати можно использовать SMIT.

- Локальные принтеры должны быть физически подключены к вашей системе, а сетевые принтеры - подключены и настроены для работы в сети.
- Для выполнения заданий удаленной печати необходимо, чтобы в вашей системе была настроена связь с удаленным сервером печати.

Для того чтобы отменить задание печати с помощью SMIT, введите:

```
smit qcan
```

Вы можете выбрать номер задания или принтер.

Выбор приоритета задачи печати (команда qpri)

Приоритет можно назначать только заданиям из локальных очередей с помощью команды **qpri**.

Принтер должен быть физически подключен к системе.

Чем больше значение, тем выше приоритет задания печати. По умолчанию приоритет равен **15**. Максимальный приоритет равен **20** для большинства пользователей, и **30** для пользователей с правами доступа root и пользователей, входящих в состав группы **printq** (группа **9**).

Примечание: Нельзя присваивать приоритет заданиям удаленной печати.

Команда **qpri** позволяет изменить приоритет задания печати, которое было передано вами на выполнение. Пользователь root и пользователи, входящие в состав группы **printq**, могут изменять приоритет любых заданий, находящихся в очереди печати.

Основной формат команды **qpri** следующий:

```
qpri -# номер_задания -a приоритет
```

Полная информация о формате приведена в описании команды **qpri** в книге *Справочник по командам, том 4*.

Ниже приведены примеры применения команды **qpri**:

- Для того чтобы присвоить заданию номер 123 приоритет 18, введите:
  ```
  qpri -# 123 -a 18
  ```
- Для того чтобы задать приоритет локального задания печати при его передаче на выполнение, введите:
  ```
  qprt -Pимя_очереди -R приоритет имя_файла
  ```

Выбор приоритета задачи печати (SMIT)

Приоритет можно назначать только заданиям из локальных очередей.

Принтер должен быть физически подключен к системе.

Чем больше значение, тем выше приоритет задания печати. По умолчанию приоритет равен **15**. Максимальный приоритет равен **20** для большинства пользователей, и **30** для пользователей с правами доступа root и пользователей, входящих в состав группы **printq** (группа **9**).

Примечание: Нельзя присваивать приоритет заданиям удаленной печати.

Для того чтобы изменить приоритет задания печати с помощью SMIT, введите:

```
smit qpri
```

Перемещение задания печати в другую очередь печати (команда qmov)

Перемещать задание печати между очередями можно с помощью команды **qmov**.

Перед выполнением этой задачи необходимо обеспечить соблюдение следующих требований:

- Принтер должен быть физически подключен к системе.
- Вы должны быть владельцем задания печати.
- У вас должны быть права доступа root.
- Вы должны являться участником группы **printq**.

Примечание: Перемещать удаленное задание печати в другую очередь нельзя.

Команда **qmov** позволяет переместить задание печати из одной очереди в другую. Можно переместить как отдельное задание, так и все задания из указанной очереди печати. Также можно переместить все задания, отправленные определенным пользователем. Выяснить номер задания печати можно с помощью команды **qchk**. Дополнительная информация приведена в описании команды **qchk**.

Общий формат команды **qmov**:

```
qmov  -mновая-очередь {[ -#номер-задания ] [ -Pочередь ] [ -uпользователь ]}
```

Перемещать задание печати можно с помощью одной из следующих команд:

- `qmov -m целевая-очередь -# номер-задания`
- `qmov -m целевая-очередь -P очередь`
- `qmov -m целевая-очередь -u пользователь`

Полная информация о синтаксисе приведена в описании команды **qmov** в в книге *Справочник по командам*.

Ниже приведены примеры применения команды **qmov**:

- Для перемещения задания номер 280 в очередь печати hp2 введите:

```
qmov -mhp2 -#280
```

- Перемещение всех заданий из очереди печати hp4D в очередь печати hp2:

```
qmov -mhp2 -Php4D
```

Перемещение задания печати в другую очередь (SMIT)

Если принтер подключен к системе, задание печати можно перемещать из одной очереди в другую с помощью SMIT.

Если принтер физически подключен к системе, задание печати можно перемещать из одной очереди в другую с помощью SMIT.

Для выполнения этой задачи должны быть выполнены следующие предварительные требования:

- Принтер должен быть физически подключен к вашей системе.
- Вы должны быть владельцем задания печати.
- У вас должны быть права доступа root.
- Вы должны быть участником группы **printq**.

Примечание: Перемещать удаленное задание печати в другую очередь нельзя.

Введите следующую команду:

```
smit qmov
```

Блокировка и разблокировка заданий печати (команды qhld)

С помощью команды **qhld** можно блокировать и разблокировать задания печати.

Примечание: Нельзя блокировать и разблокировать удаленные задания печати.

Перед выполнением этой задачи необходимо обеспечить соблюдение следующих требований:

- Принтер должен быть физически подключен к вашей системе.
- Вы должны быть владельцем задания печати.
- У вас должны быть права доступа root.
- Вы должны являться участником группы **printq**.

С помощью команды **qhld** можно заблокировать задание печати после того, как оно было отправлено. Можно блокировать как отдельное задание, так и все задания печати в указанной очереди. Номер задания печати можно определить с помощью команды **qchk**. Ниже приведен основной формат вызова команды **qhld**:

```
qhld [ -r ] {[ -#номер-задания ] [ -Pочередь ] [ -uпользователь ]}
```

Для того чтобы заблокировать задание печати, можно использовать одну из следующих команд:

- `qhld -#` *`номер-задания`*
- `qhld -P` *`очередь`*
- `qhld -u` *`пользователь`*

Для разблокирования задания печати можно использовать одну из следующих команд:

- `qhld -r -#` *`номер-задания`*
- `qhld -r -P` *`очередь`*
- `qhld -r -u` *`пользователь`*

Ниже приведены примеры применения команды **qhld**:

1. Для того чтобы заблокировать задание с номером 452 независимо от того, в какой очереди печати оно находится, введите следующую команду:

```
qhld -#452
```

2. Для того чтобы заблокировать все задания из очереди печати hp2, введите следующую команду:
   ```
   qhld -Php2
   ```
3. Для того чтобы разблокировать задание с номером 452, независимо от того, в какой очереди оно находится, введите следующую команду:
   ```
   qhld -#452 -r
   ```
4. Для того чтобы разблокировать все задания из очереди печати hp2, введите следующую команду:
   ```
   qhld -Php2 -r
   ```

Блокировка и разблокировка заданий печати (SMIT)

С помощью SMIT можно блокировать и разблокировать задания печати.

Блокировать и разблокировать задания печати могут следующие пользователи:

- Владелец задания печати
- Пользователь с правами доступа root
- Пользователи, входящие в состав группы **printq**

Для блокирования или разблокирования задания печати выполните следующие действия:

- `smit qhld`

Проверка состояния задания печати (команда qchk)

С помощью команды **qchk** можно проверить состояние задания печати.

- Для локальных заданий печати принтер должен быть либо физически подключен к системе, либо быть настроенным сетевым принтером.
- В случае заданий удаленной печати, в системе должно быть настроено соединение с удаленным сервером печати.

Команда **qchk** показывает информацию о текущем состоянии указанных заданий печати, очередей печати и пользователей.

Основной формат команды **qchk** следующий:

`qchk -P` *`имя-очереди -# номер-задания`* `-u` *`имя-владельца`*

Примечание: В базовой операционной системе поддерживается также команда проверки очереди печати BSD UNIX (**lpq**) и команда проверки очереди печати System V UNIX (**lpstat**). Полная инфоормация о синтаксисе приведена в описании команды **lpq** и **lpstat** в книге в книге *Справочник по командам*.

Ниже приведены некоторые примеры применения команды **qchk**:

- Для просмотра очереди печати по умолчанию введите:
  ```
  qchk -q
  ```
- Для того чтобы просматривать информацию о состоянии всех очередей до тех пор, пока они не станут пустыми, с обновлением через каждые **5** секунд, введите следующую команду:
  ```
  qchk -A -L -w 5
  ```
- Для просмотра состояния очереди печати **lp0** введите:
  ```
  qchk -P lp0
  ```
- Для просмотра состояния задания с номером **123** введите:
  ```
  qchk -# 123
  ```
- Для просмотра состояния всех заданий во всех очередях введите команду:
  ```
  qchk -A
  ```

Состояния очереди печати:

Ниже приведены некоторые возможные состояния очереди печати:

Элемент	Описание
DEV_BUSY	Означает, что: • Для принтера (**lp0**) определено несколько очередей, и в настоящее время он используется другой очередью. • Демон **qdaemon** попытался обратиться к порту принтера (**lp0**), но в настоящее время этот порт применяется другим приложением. Для того чтобы вывести очередь печати из состояния **DEV_BUSY**, подождите, пока очередь или приложение освободят принтер, или отмените задание или процесс, занимающие порт принтера.
DEV_WAIT	Означает, что очередь находится в состоянии ожидания из-за того, что: принтер отключен, бумагу заело, бумага закончилась или кабель неправильно подключен или поврежден. Для того чтобы вывести очередь печати из состояния **DEV_WAIT**, устраните причину неполадки. Выполнить диагностическую проверку будет легче, если с помощью команды **enq** переместить все задания из очереди **DEV_WAIT** в другую очередь, которая либо активна, либо находится в состоянии **DOWN**. После устранения неполадок можно вернуть невыполненные задания обратно в исходную очередь печати.
DOWN	Обычно очередь переходит в состояние **DOWN** из состояния **DEV_WAIT**. Причина этого состоит в том, что из-за нарушения связи с принтером драйвер не может определить, существует ли принтер. В некоторых принтерах не предусмотрена возможность отправки сигнала о разрыве связи, поэтому они отправляют сигнал о том, что принтер выключен. Когда принтер сообщает о том, что он выключен, или кажется таковым, очередь переходит в состояние **DOWN**. Для того чтобы вывести очередь из состояния **DOWN**, устраните причину неполадки и обратитесь к системному администратору с просьбой восстановить работу очереди печати. Для продолжения работы с очередью ее необходимо восстановить вручную.
HELD	Указывает на то, что задание печати блокировано. Программа буферизации не сможет обработать задание печати до тех пор, пока оно не будет освобождено.
QUEUED	Указывает, что задание печати поставлено в очередь и ожидает выполнения.
READY	Указывает, что все готово для постановки задания в очередь и его печати.
RUNNING	Указывает, что задание печати выполняется.

Проверка состояния задания печати (SMIT)

С помощью команды **smit** можно проверить состояние задания печати.

- Локальные принтеры должны быть физически подключены к вашей системе, а сетевые принтеры - подключены и настроены для работы в сети.
- Для выполнения заданий удаленной печати необходимо, чтобы в вашей системе была настроена связь с удаленным сервером печати.

Можно просмотреть сведения о текущем состоянии отдельных заданий, очередей, принтеров или пользователей. Для просмотра состояния задания печати с помощью SMIT введите команду:

```
smit qchk
```

Состояния очереди печати:

Ниже приведены некоторые возможные состояния очереди печати:

Состояние	Описание
DEV_BUSY	Означает, что: • Для принтера (**lp0**) определено несколько очередей, и в настоящее время он используется другой очередью. • Демон **qdaemon** попытался обратиться к порту принтера (**lp0**), но в настоящее время этот порт применяется другим приложением. Для того чтобы вывести очередь печати из состояния **DEV_BUSY**, подождите, пока очередь или приложение освободят принтер, или отмените задание или процесс, занимающие порт принтера.

Состояние	Описание
DEV_WAIT	Означает, что очередь находится в состоянии ожидания из-за того, что: принтер отключен, бумагу заело, бумага закончилась или кабель неправильно подключен или поврежден. Для того чтобы вывести очередь печати из состояния **DEV_WAIT**, устраните причину неполадки. Выполнить диагностическую проверку будет легче, если с помощью команды **enq** переместить все задания из очереди **DEV_WAIT** в другую очередь, которая либо активна, либо находится в состоянии **DOWN**. После устранения неполадок можно вернуть невыполненные задания обратно в исходную очередь печати.
DOWN	Обычно очередь переходит в состояние **DOWN** из состояния **DEV_WAIT**. Причина этого состоит в том, что из-за нарушения связи с принтером драйвер не может определить, существует ли принтер. В некоторых принтерах не предусмотрена возможность отправки сигнала о разрыве связи, поэтому они отправляют сигнал о том, что принтер выключен. Когда принтер сообщает о том, что он выключен, или кажется таковым, очередь переходит в состояние **DOWN**. Для того чтобы вывести очередь из состояния **DOWN**, устраните причину неполадки и обратитесь к системному администратору с просьбой восстановить работу очереди печати. Для продолжения работы с очередью ее необходимо восстановить вручную.
HELD	Указывает на то, что задание печати блокировано. Программа буферизации не сможет обработать задание печати до тех пор, пока оно не будет освобождено.
QUEUED	Указывает, что задание печати поставлено в очередь и ожидает выполнения.
READY	Указывает, что все готово для постановки задания в очередь и его печати.
RUNNING	Указывает, что задание печати выполняется.

Форматирование файлов для печати (команда pr)

С помощью команды **pr** можно выполнить простое форматирование файлов, отправляемых на принтер.

Для форматирования текста вывод команды **pr** записывается в стандартный ввод команды **qprt**.

Ниже описаны некоторые наиболее употребительные флаги команды **pr**:

Флаг	Описание
-d	Отделяет вывод двойным интервалом.
-h " *Строка***"**	В качестве заголовка страницы вместо имени файла вставляется строка, заключенная в кавычки (" "). Флаг должен быть отделен от строки пробелом.
-l *Число_строк*	Переопределяет длину страницы по умолчанию, равную 66 строкам, устанавливая ее равной значению *Число_строк*. Если значение *Число-строк* меньше суммы длин заголовка и концевика (в строках), то заголовок и концевик не печатаются (аналогично действию флага **-t**).
-m	Объединяет файлы. Команда **pr** записывает последовательно по одной строке из каждого файла, указанного в параметре *Файл* , в виде текстовых столбцов фиксированной ширины, которая зависит от числа столбцов. Этот флаг несовместим с флагом **-Столбец**.
-n [*Ширина*][*Символ*]	Печатает номера строк. Число цифр в номере строки задается в параметре *Ширина*. По умолчанию выводится 5 цифр. Если задана переменная *Символ* (любой символ, отличный от цифры), то указанный символ вставляется в качестве разделителя между номером строки и ее содержимым. По умолчанию в качестве разделителя применяется символ табуляции ASCII.
-o *Смещение*	Сдвигает каждую строку на число символов, указанное в параметре *Смещение*. Общее число символов в строке - это сумма ширины строки и смещения. По умолчанию *Смещение* равно 0.
-s *Символ*	Применяет в качестве разделителя столбцов символ, указанный в параметре *Символ*, вместо соответствующего числа пробелов. По умолчанию *Символ* - это символ табуляции ASCII.
-t	Отменяет печать верхнего и нижнего колонтитулов по пять строк каждый. Печать прекращается после вывода последней строки файла, и остаток страницы не заполняется пробелами.
-w *Ширина*	Ширина столбца устанавливается равной значению *Ширина*. По умолчанию выводятся одинаковые столбцы шириной 72 символа. Значение параметра может быть любым. Если флаг **-w** не указан, но задан флаг **-s**, то по умолчанию ширина столбца будет равна 512 символам.
-*Столбец*	Параметр *Столбец* задает число столбцов. Значение по умолчанию равно 1. Этот флаг нельзя использовать вместе с флагом **-m**. Если задано несколько столбцов, то применяются флаги **-e** и **-i**. Длина текстового столбца не должна превышать длину страницы (см. описание флага **-l**). Если этот флаг указывается вместе с флагом **-t**, то число строк вывода будет минимальным.
+*Страница*	Показывает страницы начиная с той, номер которой указан в параметре *Страница*. Значение по умолчанию равно 1.

Полная информация о формате приведена в описании команды **pr** в книге в книге *Справочник по командам*.

Ниже приведено несколько примеров применения команды **pr**:

- Для печати файла с именем `prog.c` с выводом заголовков и номеров страниц введите:

  ```
  pr prog.c | qprt
  ```

  В результате выполнения этой команды к файлу `prog.c` будут добавлены заголовки, а затем он будет отправлен команде **qprt**. В заголовке указывается такая информация, как дата последнего изменения файла, имя файла и номер страницы.

- Заголовок для файла `prog.c` можно задать так:

  ```
  pr -h "ОСНОВНАЯ ПРОГРАММА" prog.c | qprt
  ```

  В результате файл `prog.c` будет напечатан с заголовком, в котором вместо имени файла будет подставлена строка `ОСНОВНАЯ ПРОГРАММА`. Кроме того, в заголовке будут указаны дата последнего изменения файла и номер страницы.

- Для печати файла `word.lst` в виде нескольких столбцов введите:

  ```
  pr -3 word.lst | qprt
  ```

  Файл `word.lst` будет напечатан в виде трех вертикальных столбцов.

- Для печати нескольких файлов на одной странице введите:

  ```
  pr -m -h "Сотрудники и посетители" member.lst visitor.lst | qprt
  ```

  Файлы `member.lst` и `visitor.lst` будут напечатаны рядом на одной странице под общим заголовком **Сотрудники и посетители**.

- Для того чтобы изменить файл `prog.c` для использования в будущем, введите:

  ```
  pr -t -e prog.c > prog.notab.c
  ```

  Этой командой в файле `prog.c` символы табуляции заменяются пробелами. Результат помещается в файл `prog.notab.c`. Позиции табуляции - это столбцы 9, 17, 25, 33 и т.д. Флаг **-e** указывается в команде **pr** для замены символов табуляции; флаг **-t** подавляет вывод заголовков страниц.

- Для печати файла `myfile` в виде двух столбцов с альбомной ориентацией и со шрифтом размером 7 пунктов введите:

  ```
  pr -l66 -w172 -2 myfile | qprt -z1 -p7
  ```

Печать файлов ASCII на принтере PostScript

Система форматирования текста содержит специальный фильтр **enscript**, преобразующий файлы принтера из формата ASCII в формат PostScript для их печати на принтере PostScript.

- Принтер должен быть физически подключен к системе.
- Принтер должен быть определен и настроен.
- Должна быть установлена утилита Служб форматирования текста, отвечающая за преобразование.

Фильтр **enscript** вызывается командой **qprt -da** при помещении задания в очередь печати PostScript. При помещении файлов ASCII в очередь печати PostScript можно задавать некоторые флаги команды **qprt**, предназначенные для настройки вывода:

Элемент	Описание
-1+	Добавляет заголовки страниц.
-2+	Форматирует вывод в виде двух колонок.
-3+	Печатает заголовки, даты и номера страниц особым шрифтом. Иногда этот режим называют "gaudy" ("витиеватый").
-4+	Печатает файл, даже если он содержит непечатаемые символы.
-5+	Выдает список символов, не включенных в шрифт.
-h *string*	Задает строку, которая будет применяться в заголовках страниц. Если этот флаг не указан, заголовок будет содержать имя файла, дату его изменения и номер страницы.
-l *значение*	Задает максимальное число строк, печатаемых на странице. В зависимости от размера в пунктах, в действительности страница может содержать меньше строк.
-L!	Усекает строки, длина которых превышает ширину страницы.

Элемент	Описание
-p	Задает размер шрифта в пунктах. Если этот флаг не указан, то размер в пунктах будет равен 10; однако, если установлен режим печати в виде двух колонок с поворотом (**-2+ -z1**), то размер в пунктах будет равен 7.
-s	Задает начертание шрифта. Если этот флаг не указан, будет применяться начертание шрифта Courier. Указанный шрифт должен быть доступен на принтере PostScript. Допустимы следующие значения: Courier-Oblique Helvetica Helvetica-Oblique Helvetica-Narrow Helvetica-Narrow-Oblique NewCenturySchlbk-Italic Optima Optima-Oblique Palatino-Roman Palatino-Italic Times-Roman Times-Italic
-z1	Поворачивает печатаемый текст на 90 градусов (ориентация становится альбомной).

Ниже приведено несколько примеров применения флагов команды **qrpt**:

- Для того чтобы направить текстовый файл `myfile.ascii` на принтер PostScript с именем **Msps1**, введите:
  ```
  qprt -da -PMsps1 myfile.ascii
  ```
- Для того чтобы направить текстовый файл `myfile.ascii` на принтер PostScript с именем **Msps1** и напечатать его шрифтом Helvetica, введите:
  ```
  qprt -da -PMsps1 -sHelvetica myfile.ascii
  ```
- Для того чтобы направить текстовый файл `myfile.ascii` на принтер PostScript с именем **Msps1** и напечатать его шрифтом размером 9 пунктов, введите:
  ```
  qprt -da -PMsps1 -p9 myfile.ascii
  ```

Автоматизация преобразования ASCII в PostScript:

Вы можете настроить систему для автоматического распознавания файлов принтера ASCII в очереди печати PostScript и преобразования их в формат PostScript.

Большинство приложений, создающие файлы принтера в формате PostScript, следуют соглашению о том, что в начале файла PostScript должны быть указаны символы **%!**, обозначающие этот формат. Если система должна автоматически распознавать файлы принтера ASCII в очереди печати PostScript и преобразовывать их в формат PostScript перед отправкой на принтер PostScript, то выполните следующие действия:

1. Введите в командной строке `smit chpq`
2. Введите имя очереди PostScript или выберите его в списке с помощью функции **Список**.
3. Выберите пункт меню **Настройка принтера**.
4. В поле **Автоматически определять тип файла принтера?** укажите **yes**.

Теперь все команды будут преобразовывать файлы ASCII в файлы PostScript и отправлять их на принтер PostScript. Для преобразования файла `myfile.ascii` введите одну из следующих команд:

```
qprt -Pps myfile.ps myfile.ascii
lpr -Pps myfile.ps myfile.ascii
lp -dps myfile.ps myfile.acsii
```

где *ps* - это очередь печати PostScript.

Переопределение автоматического определения типов файлов принтера:

В некоторых случаях может потребоваться переопределение автоматически определяемых типов файлов принтера.

Переопределить автоматическое определение типа файла принтера для печати данных в формате PostScript можно с помощью флагов **-d** и **-s**. Флаг **-d** переопределяет тип файла принтера по умолчанию, а флаг **-s** задает печать в формате PostScript.

В следующих ситуациях может потребоваться отменить автоматическое определение типа файла принтера для печати данных в формате PostScript:

- Для печати файла PostScript с именем `myfile.ps`, в начале которого не указаны символы **%!**, введите команду:

  ```
  qprt -ds -Pps myfile.ps
  ```

- Для печати исходного текста из файла PostScript с именем `myfile.ps`, который начинается с символов **%!**, введите команду:

  ```
  qprt -da -Pps myfile.ps
  ```

Обзор команд для печати

Для печати и управления очередями печати используется несколько команд.

Команда	Описание
cancel	Отменяет запросы к построчному принтеру.
lp	Направляет запросы к построчному принтеру.
lpq	Проверяет очередь буфера.
lpr	Помещает задания печати в очередь.
lprm	Удаляет задания из очереди буфера построчного принтера.
lpstat	Показывает данные о состоянии построчного принтера.
pr	Записывает файл в стандартный вывод.
qcan	Отменяет задание печати.
qchk	Показывает состояние очереди печати.
qhld	Блокирует или разблокирует задание печати.
qmov	Перемещает задание в другую очередь печати.
qpri	Назначает приоритет заданию печати.
qprt	Запускает задание печати.

Live Partition Mobility с адаптерам Host Ethernet Adapter

Совместное применение Live Partition Mobility (LPM) с функцией Host Ethernet Adapter (HEA) программного обеспечения IBM PowerVM позволяет выполнить миграцию логических разделов AIX вместе с приложениями из одного физического раздела в другой физический раздел, в то время как HEA присвоен к разделу миграции.

В ходе миграции HEA будет удален из исходного раздела и не будет восстановлен после завершения миграции. Соединение с сетью не перестает работать.

Требования для Live Partition Mobility с помощью HEA

Для применения LPM вместе с HEA должны быть выполнены предварительные требования к конфигурации системы и доступу к ней.

Требования к разделам

- Исходный и целевой CEC должны поддерживать миграцию разделов.

- Исходный логический раздел AIX не должен обладать физическими ресурсами; в его профайле должен быть указан параметр Обязательный.
- Исходный логический раздел AIX не должен обладать физическими ресурсами за исключением HEA.

Требования к доступу

- Вы должны обладать правами администратора целевого раздела миграции.
- Вы должны обладать правами hscroot в исходной и целевой HMC или аналогичными правами доступа, необходимыми для миграции разделов.

Требования к конфигурации

- В профайле раздела HEA не должен быть указан параметр Обязательный; допустимо значение Предпочитаемый.
- Все HEA должны быть настроены в EtherChannel как основные адаптеры.
- В качестве основных адаптеров EtherChannel должны быть настроены только HEA.
- В качестве резервного адаптера EtherChannel должен быть настроен адаптер виртуального Ethernet.
- По крайней мере один EtherChannel должен быть настроен с HEA в качестве основного адаптера и адаптером виртуального Ethernet в качестве резервного адаптера.
- Поддерживается не более четырех EtherChannel.
- Функция переключения EtherChannel должно быть активна.
- Убедитесь, что исходная и целевая системы настроены для миграции разделов.
- В случае миграции между двумя HMC необходимо настроить идентификацию SSH между исходной и удаленной HMC. Перед тем, как приступить к миграции, в исходной HMC следует выполнить команду **mkauthkeys**.

Запуск Live Partition Mobility совместно с HEA

Запустить LPM с HEA можно с помощью интерфейса SMIT.

Предварительно рекомендуется ознакомиться с разделом “Требования для Live Partition Mobility с помощью HEA” на стр. 609.

Для того чтобы выполнить миграцию разделам LPM с HEA, выполните следующие действия:

1. В командной строке выполните следующую команду SMIT для отображения меню Live Partition Mobility с Host Ethernet Adapter (HEA): **smitty migration**.
2. Укажите имя хоста или IP-адрес исходной HMC.
3. Укажите имя пользователя исходной HMC.
4. Введите **no**, если исходная и целевая системы работают под управлением одной HMC, и перейдите к шагу 5. Введите **yes**, если исходная и целевая системы работают под управлением разных HMC, и выполните следующие действия:

   Примечание: Перед вводом значения **yes** в исходной HMC необходимо выполнить команду **mkauthkeys**.

   a. Укажите имя хоста или IP-адрес удаленной HMC.
   b. Укажите имя пользователя исходной HMC.
5. Укажите имя исходной системы.
6. Укажите имя целевой системы.
7. Укажите имя раздела, который требуется перенести.
8. Введите **no**, если миграцию требуется выполнить без проверки. Введите **yes** для выполнения только проверки миграции. Если указано значение **yes**, то миграция не выполняется - выполняется только проверка.

   Примечание: Перед миграцией раздела рекомендуется выполнить проверку миграции раздела.

9. Убедитесь, что во всех полях указана правильная информация и нажмите клавишу **Enter** для запуска миграции.

   Примечание: Система дважды запросит пароль пользователя. Введите пароль пользователя исходной HMC, указанный на шаге 4b.

В этом примере раздел X переносится из CEC C, работающей под управлением HMC A, в CEC D, работающую под управлением HMC B. Выполните команду **mkauthkeys** с помощью HMC A для обеспечения идентификации между HMC A и HMC B.

Для миграции в SMIT указываются следующие значения:

```
Имя хоста или
IP-адрес исходной HMC: A
Имя пользователя исходной HMC: hscroot
Миграция между двумя HMC: yes
 Имя хоста или IP-адрес удаленной HMC: B
 Имя пользователя удаленной HMC: hscroot
Исходная система: C
Целевая система: D
Имя переносимого раздела: X
Только проверка миграции: no
```

Другой пример: Раздел X переносится из CEC C, работающего под управлением HMC A, в CEC D, который также работает под управлением HMC A.

Для миграции в SMIT указываются следующие значения:

```
Имя хоста или IP-адрес исходной
HMC: A
Имя пользователя исходной HMC: hscroot
Миграция между двумя HMC: no
 Имя хоста или IP-адрес удаленной HMC:
 Имя пользователя удаленной HMC:
Исходная система: C
Целевая система: D
Имя переносимого раздела: X
Только проверка миграции: no
```

В случае сбоя миграции выполните процедуру Live Partition Mobility в исходной HMC.

Перенос клиента NIM с использованием LPM:

Если для переноса системы с одного физического сервера на другой используется Live Partition Mobility (LPM) и система определена как клиент управления сетевой установкой (NIM), то после завершения переноса LPM администратор NIM должен указать в атрибуте *cpuid* клиента NIM значение, соответствующее новому аппаратному обеспечению.

Для обновления атрибута *cpuid* выполните следующие действия:

1. На клиенте NIM получите новый ИД атрибута cpuid, запустив команду:

   ```
   uname -a
   ```

2. На сервере NIM запустите следующую команду:

   ```
   nim -o change -a cpuid=ИД-cpu клиент
   ```

   Примечание: Сетевая программа установки **OS_install** больше не поддерживает установку операционной системы Linux из-за удаления поддержки Cluster Systems Management (CSM) в операционной системе AIX.

Перемещение адаптера для DLPAR

Необходимо настроить графический адаптер до перемещения адаптера для операций динамического распределения ресурсов (DLPAR).

Используйте следующие инструкции для динамического перемещения графического адаптера, такого как FC 5748:

1. Убедитесь в том, что никакие текущие процессы (Desktop и Xserver) не используют графический адаптер (например, /dev/lft0).
2. Убедитесь в том, что консоль не установлена как lft0. Для идентификации определенных или доступных зависимых интерфейсов (lft или rcm) введите следующую команду:

   ```
   lsdev -C | grep lft
   lsdev -C | grep rcm
   ```

   Для того чтобы получить родительский адаптер Взаимосвязи периферийных компонентов (PCI) после того как определен графический адаптер, введите следующую команду:

   ```
   odmget -q name=<cortina adapter name. for instance cor0> CuDv
   ```

   Эта команда предоставляет информацию о родителе и cortina.
3. Для того чтобы удалить все зависимые интерфейсы и сделать адаптер готовым к операции DLPAR, введите следующую команду:

   ```
   # pdisable lft0

   # rmdev -l rcm0
   rcm0 Defined

   # rmdev -l lft0
   lft0 Defined

   # rmdev -Rdl pci23
   cor0 deleted
   pci23 deleted
   ```

Интерфейс консоли управления может быть использован для операций DLPAR над графическим адаптером.

Устройство обратного вызова

Устройство обратного вызова - это устройство, которое можно использовать в качестве блочного устройства для обращения к файлам.

Устройство обратного вызова может содержать образ ISO, образ диска, файловую систему или образ логического тома. Например, смонтировав образ ISO компакт-диска в качестве устройства обратного вызова, можно работать с образом как с обычным компакт-диском.

Команда **loopmount** позволяет создать устройство обратного вызова, связать с ним указанный файл и смонтировать устройство обратного вызова. Команда **loopumount** позволяет размонтировать файл образа и удалить устройство обратного вызова. Число устройств обратного вызова в операционной системе AIX не ограничено. Устройство обратного вызова никогда не создается по умолчанию; оно должно быть создано явным образом. Размер блока устройства обратного вызова всегда составляет 512 байт.

Кроме того, новое устройство можно создать с помощью команды **mkdev**, изменить с помощью команды **chdev** и удалить с помощью команды **rmdev**. После создания устройство обратного вызова можно смонтировать для работы со связанным образом или использовать в качестве блочного устройства для ввода-вывода. Сведения о связанном образе можно просмотреть с помощью команды **ioctl (IOCINFO)**.

В операционной системе AIX действуют следующие ограничения устройства обратного вызова:

- Команда **varyonvg** не поддерживается для образов дисков.

- Образы компакт-дисков ISO, DVD UDF+ISO и другие образы компакт-дисков и дисков DVD поддерживаются только для чтения.
- Файл образа можно связать только с одним устройством обратного вызова.
- Устройства обратного вызова не поддерживаются в разделах рабочей схемы.

Информация, связанная с данной:

Команда loopmount

Команда loopumount

Команда ioctl

Инфраструктура обработки событий AIX для AIX и кластеров AIX - AHAFS

Инфраструктура обработки событий AIX для AIX и кластеров AIX - это среда отслеживания событий для отслеживания стандартных и пользовательских событий.

Инфраструктура обработки событий AIX - Введение

Инфраструктура обработки событий AIX - это среда отслеживания событий для отслеживания стандартных и пользовательских событий.

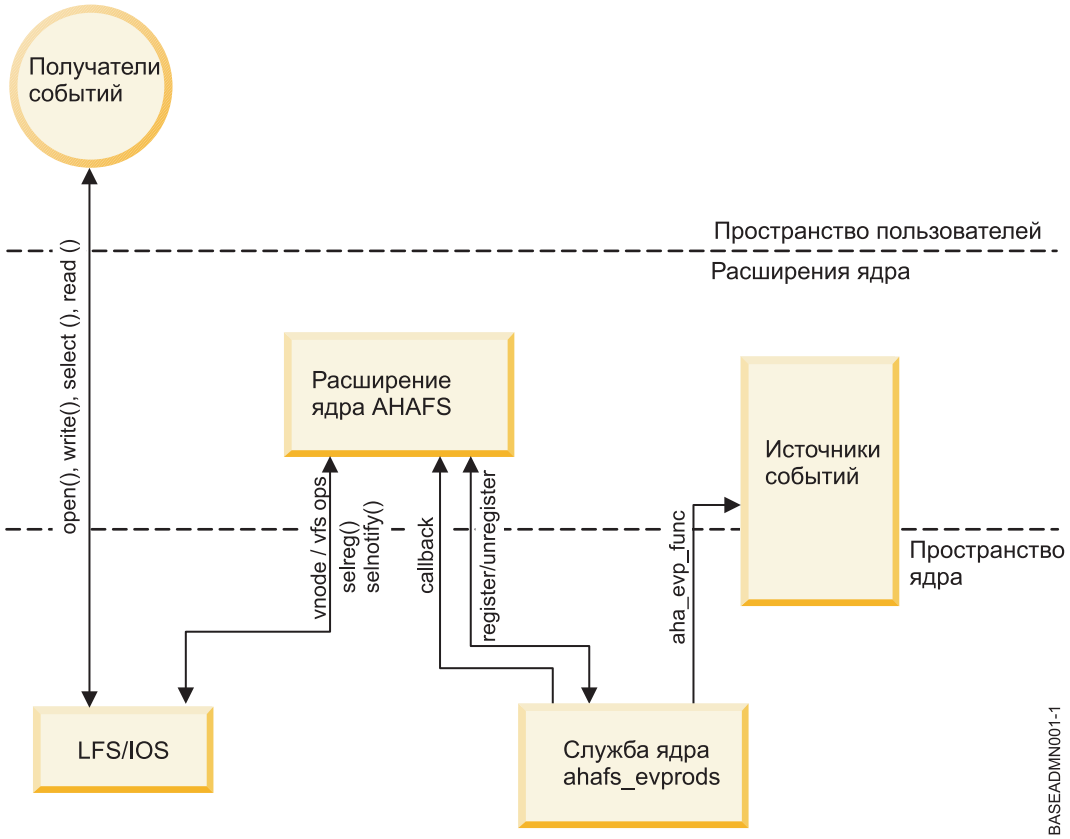
В инфраструктуре обработки событий AIX событие определяется как любое изменение состояния или значения, которое может быть обнаружено ядром или расширением ядра во время возникновения этого изменения. События, которые могут быть отслежены, представлены как файлы в файловой псевдосистеме. Вот некоторые из преимуществ Инфраструктуры обработки событий AIX:

- Нет необходимости выполнять постоянные опросы. Пользователи, отслеживающие события, получают уведомления при возникновении этих событий.
- Пользователю, отслеживающему событие, предоставляется подробная информация о событии (такая как трассировка стека или информация о пользователе и процессе).
- Применяются существующие интерфейсы файловой системы, поэтому нет необходимости в новом интерфейсе прикладных программ (API).
- Управление передается Инфраструктуре обработки событий AIX точно в момент возникновения события.

Компоненты инфраструктуры обработки событий AIX

Инфраструктура обработки событий AIX состоит из следующих четырех компонентов:

- Расширение ядра, реализующее файловую псевдосистему.
- Приемники событий, принимающие события.
- Источники событий, отправляющие события.
- Компонент ядра, который выполняет роль интерфейса между расширением ядра и источниками событий.



Расширение ядра инфраструктуры обработки событий AIX

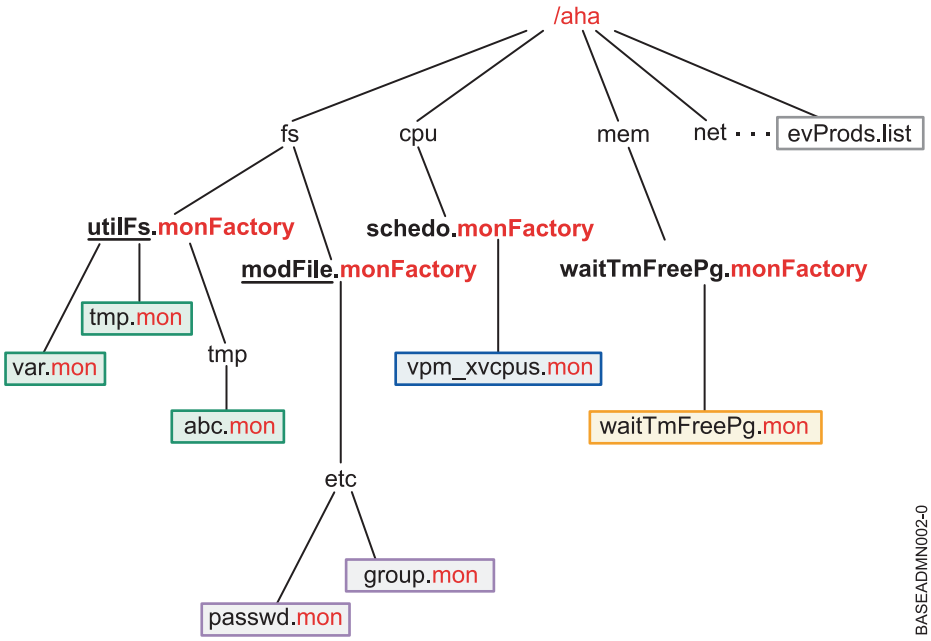
Расширение ядра инфраструктуры обработки событий AIX реализует файловую псевдосистему.

Все события представлены как файлы этой файловой системы. Существует четыре типа файловых объектов:

- **Файлы .list**: В файловой псевдосистеме имеется только один файл **.list - evProds.list**. Это особый файл, который при чтении возвращает имена всех определенных на данный момент источников событий.
- **Каталоги .monFactory**: Фабрики монитора - каталоги особого типа. Это источники событий, представленные в виде каталогов. Каталоги фабрики монитора и их родительские каталоги создаются для пользователя автоматически.
- **Подкаталоги**: Подкаталоги используются как для упрощения управления, так и для представления полных имен файлов монитора (см. **Файлы .mon**).
- **Файлы .mon**: Файлы монитора представляют события, которые можно отслеживать. Полный путь к файлу монитора от его родительской фабрики монитора, кроме расширения **.mon**, - это полное представление отслеживаемого события. Например, файл **/aha/fs/modFile.monFactory/etc/password.mon** используется для отслеживания изменений файла **/etc/passwd**. Файлы монитора могут существовать только в структуре фабрики монитора.

В этой файловой псевдосистеме невозможно создать другие обычные файлы. Поскольку файловая система инфраструктуры обработки событий AIX является файловой системой, расположенной в памяти, максимальное количество i-узлов ограничено 32 Кб. Число используемых i-узлов будет показано в выводе команды **df**.

Пример разметки файловой системы Инфраструктуры обработки событий AIX приведен ниже:



Примечание:

Файл **evProds.list** находится непосредственно в каталоге файловой системы и содержит список источников событий, определенных и пригодных для использования в этом экземпляре операционной системы.

С помощью интерфейса LFS Инфраструктура обработки событий AIX преобразует текстовый ввод, записанный в файлы монитора, в спецификации, указывающие, каким способом требуется уведомлять пользователя о возникших событиях. После того как пользователь выполнил вызов **select()** или блокирующий вызов **read()** для указание на начало отслеживания, Инфраструктура обработки событий AIX уведомит соответствующий источник событий о начале отслеживания указанного события.

При обнаружении вхождения события Инфраструктура обработки событий AIX отправит уведомление во все ожидающие приемники с совпадениями критериев отслеживания.

Приемники событий

Приемники событий представляют собой процессы пользовательского пространства, ожидающие возникновения событий.

Приемники настраивают отслеживание событий посредством записи информации в файл монитора с указанием, как и когда им должны отправляться уведомления. Приемники могут ожидать уведомления о событии в вызове **select()** или в блокирующем вызове **read()**.

Инфраструктура обработки событий AIX не поддерживает нити. В процессах нельзя использовать несколько нитей для отслеживания одного события.

Источники событий

Источники событий - это разделы кода в ядре или расширении ядра, способные обнаруживать событие.

При возникновении отслеживаемого события источник событий уведомляет расширение ядра Инфраструктуры обработки событий AIX и отправляет всю связанную информацию о событии для передачи в приемник.

В настоящее время существует два главных класса источников событий:

- Отслеживающие изменение состояния
- Отслеживающие превышение порогового значения, определенного пользователем

Служба ядра ahafs_evprods

Служба ядра **ahafs_evprods** помогает взаимодействию между расширением ядра Инфраструктуры обработки событий AIX и источниками событий.

Для поддержания взаимодействия между расширением ядра Инфраструктуры обработки событий AIX и источниками событий применяется экспорт службы ядра **ahafs_evprods**. Внутри ядра для поиска источников событий и передачи информации между соответствующими источниками событий и расширением ядра применяется список зарегистрированных источников событий.

Настройка Инфраструктуры обработки событий AIX

Действия, необходимые для настройки Инфраструктуры обработки событий AIX.

Для настройки Инфраструктуры обработки событий AIX необходимы только следующие действия:

1. Установить набор файлов **bos.ahafs**.
2. Создать каталог для требуемой точки монтирования.
3. Выполните следующую команду:

   ```
   mount -v ahafs <точка-монтирования> <точка-монтирования>
   ```

   Пример

   ```
   mkdir /aha
   mount -v ahafs /aha /aha
   ```

   При монтировании файловой системы Инфраструктуры обработки событий AIX автоматически загружается расширение ядра и создаются все фабрики монитора. За один раз можно смонтировать только один экземпляр файловой системы Инфраструктуры обработки событий AIX. Файловую систему Инфраструктуры обработки событий AIX можно смонтировать в любом обычном каталоге, но пользователям рекомендуется использовать каталог **/aha**.

Работа Инфраструктуры обработки событий AIX - Обзор высокого уровня

Приемник может отслеживать несколько событий, а одно и то же событие может отслеживаться несколькими приемниками. Каждый приемник может отслеживать основанные на значении события с отличающимся пороговым значением. В таком случае расширение ядра Инфраструктуры обработки событий AIX ведет список информации о каждом приемнике, включая следующее:

- Указанный тип ожидания (**WAIT_IN_READ** или **WAIT_IN_SELECT**)
- Уровень запроса информации
- Пороговое значение для которого требуется выполнять отслеживание (при отслеживании события на основе порогового значения)
- Буфер, используемый для хранения информации о вхождениях события.

Информация о событии хранится с учетом процесса, чтобы разные процессы, отслеживающие одно и то же событие, не изменили данные события. Когда процесс приемники выполняет чтение из файла монитора, он считывает только свою копию данных события.

Обычный поток отслеживания события

В этом разделе описаны этапы отслеживания события.

1. Процесс пытается открыть или создать файл монитора.
2. Инфраструктура обработки событий AIX передает путь файла монитора в соответствующий источник событий. Источник событий проверяет допустимость события, представляемого файлом монитора, и наличие у процесса прав доступа на отслеживание события.

3. Процесс записывает информацию в файл, указывая следующее:
   a. Тип ожидания (**WAIT_TYPE=WAIT_IN_READ** или **WAIT_TYPE=WAIT_IN_SELECT**). Тип ожидания по умолчанию - **WAIT_IN_SELECT**.
   b. Когда требуются уведомления. Для событий изменения состояния пользователь должен указать **CHANGED=YES**. Для событий на основе пороговых значений пользователь должен указать **THRESH_HI=**<значение>, **THRESH_LO=**<значение> или оба, в зависимости от функций связанного источника событий. Для данной спецификации не предусмотрено значений по умолчанию, причем **CHANGED=YES** и **THRESH_*=**<значение> не могут быть заданы одновременно.
4. Затем Инфраструктура обработки событий AIX захватит блок для определенного процесса, чтобы сохранить эту информацию, если она еще не существует для данного процесса, и заполнит его информацией, записанной пользователем.
5. Процесс выполняет вызов **select()** или **read()** с объединением в блоки для файла монитора
6. Инфраструктура обработки событий AIX вызовет **ahafs_evprods** для проверки допустимости указанных пороговых значений для данного события. Например, источник событий **utilFs** не допускает значений > 100%. Если пороговое значение окажется недопустимым, вызов **select()** или **read()** возвратит **RC_FROM_EVPROD**, а после прочтения файла монитора будет возвращено **EINVAL**.
7. Если применяются источники событий на основе пороговых значений, то в источник событий для каждого порогового значения (**hi** или **lo**) для отслеживания отправляется только одно значение. Во время вызова **select()** или **read()** с объединением в блоки Инфраструктура обработки событий AIX зарегистрирует это новое пороговое значение в источнике событий, если одно из следующих условий окажется истинным:
   a. Если данное событие не отслеживается какими-либо другими процессами, пороговые значения, заданные этим приемником, регистрируются в источнике событий.
   b. Если данное событие отслеживается другими процессами, тогда, если значение **THRESH_LO**, заданное приемником, превышает отслеживаемое в данный момент нижнее пороговое значение ИЛИ если значение **THRESH_HI**, заданное приемником, ниже отслеживаемого в данный момент верхнего порогового значения, Инфраструктура обработки событий AIX вызывает службу ядра **ahafs_evprods** для обновления отслеживаемого в данный момент порогового значения.
8. После возвращения из службы ядра **ahafs_evprods** возвращается фактическое значение (в некоторых случаях). Если возвращенное фактическое значение уже достигло или превысило какое либо пороговое значение, то возврат вызова **read()** или **select()** произойдет немедленно, а параметр **RC_FROM_EVPROD**, зарегистрированный в буфере, будет иметь значение **EALREADY**. Вызовы **read()** или **select()** возвратят 0.
9. Если применяются источники событий изменения состояния, то для регистрации события всегда вызывается функция **ahafs_evprods**.
10. После успешной регистрации Инфраструктура обработки событий AIX настраивает уведомление. Для приемников, ожидающих в **select()**, уведомление настраивается с помощью **selreg()**. Для приемников, создающих блоки в вызове **read()**, нить переводится в состояние ожидания с **e_sleep_thread()**.
11. Как только источник событий обнаруживает, что событие возникло, он отправляет в Инфраструктуру обработки событий AIX уведомление с информацией о событии (такой как информация о процессе, активировавшем событие, текущее значение, код возврата и др.).
12. Во время этого обратного вызова из источника событий Инфраструктура обработки событий AIX выполнит следующее:
    a. Определит **ahaNode**, соответствующий событию
    b. Выполнит поиск ожидающих приемников, чтобы определить, чьи пороговые значения достигнуты или превышены, чтобы отправить уведомление с вызовом **selnotify()** или **e_wakeup()**. Будут уведомлены все приемники, ожидающие события изменения состояния.
13. Как только процесс будет уведомлен о событии, он выполнит чтение из файла монитора для получения данных события. Ниже приведен пример вывода из события.

    Пример вывода для источника событий изменения состояния, указавшего на необходимость получения трассировки стека:

```
BEGIN_EVENT_INFO
TIME_tvsec=1269377315
TIME_tvnsec=955475223
SEQUENCE_NUM=0
PID=2490594
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=cat
RC_FROM_EVPROD=1000
END_EVENT_INFO
```

Пример события с пороговым значением:

```
BEGIN_EVENT_INFO
TIME_tvsec=1269378095
TIME_tvnsec=959865951
SEQUENCE_NUM=0
CURRENT_VALUE=2
RC_FROM_EVPROD=1000
END_EVENT_INFO
```

Примечание: Из-за асинхронного характера уведомления процесса возвращенное текущее значение может оказаться устаревшим к моменту чтения процессом файла монитора. При первом достижении или превышении порогового значения пользователи будут уведомлены, но другие операции, которые могут изменить отслеживаемое значение, не будут заблокированы.

Работа с Инфраструктурой обработки событий AIX

Все каталоги в файловой системе Инфраструктуры обработки событий AIX имеют режим доступа 1777, а все файлы имеют режим доступа 0666.

В настоящее время все каталоги в файловой системе Инфраструктуры обработки событий AIX имеют режим 1777, а все файлы имеют режим 0666. Эти режимы нельзя изменить, но можно изменить владельцев файлов и каталогов. Управление доступом для отслеживания событий осуществляется на уровне источника событий. При вводе команды **stat ()** над файловым объектом в файловой псевдосистеме время создания или изменения не поддерживается и всегда возвращается как текущее время. Попытка изменить это время возвратит ошибку.

Отслеживание событий

Создание файла монитора

Для отслеживания события необходимо создать файл монитора, соответствующий событию.

Перед началом отслеживания события необходимо создать файл монитора, соответствующий событию. Инфраструктура обработки событий AIX поддерживает **open()** с флагом **O_CREAT**. Для примера выполним этапы, необходимые для отслеживания файловой системы **/filesys/clj-fs** при использовании на 90%.

- Необходимо также создать обязательные подкаталоги:
  ```
  mkdir /aha/fs/utilFs.monFactory/filesys
  ```
- Откройте файл **/aha/fs/utilFs.monFactory/filesys/clj-fs.mon**.

Перед возможным созданием файла монитора Инфраструктура обработки событий AIX вызовет источник событий, чтобы определить, допустимо ли регистрируемое событие и есть ли у пользователя достаточные права доступа для отслеживания указанного события. Ниже приведены некоторые из распространенных ошибок, которые могут быть возвращены при создании или открытии файла монитора:

Таблица 71. Коды возврата

Код возврата	Сведения
ENODEV	Указанному пути не соответствует какое-либо событие. **Примечание:** Ошибка **ENODEV** может быть также возвращена при попытке открытия существующего файла монитора, если события уже не существует.
EPERM	У пользователя нет прав доступа для отслеживания указанного события.
ENOTSUP	Указанное событие не поддерживает отслеживания с помощью Инфраструктуры обработки событий AIX.

Запись в файл монитора

Процесс приемника записывает данные в файл монитора, указывая, как и где он должен уведомляться о событиях.

После того как требуемый файл монитора будет создан и открыт, процесс приемника запишет данные в файл монитора, указывая, как и где он должен уведомляться о событиях. Эти данные записываются в виде пар **<ключ>=<значение>**, а между парами может стоять разделитель ; или пробел. Допускаются следующие пары **<ключ>=<значение>**:

Таблица 72. Допустимые пары <ключ>=<значение>

Ключ	Допустимые значения	Сведения
CHANGED	YES	Указывает, что отслеживаемое событие имеет тип **AHAFS_THRESHOLD_STATE** и что приемник должен уведомляться при изменении состояния события.
THRESH_HI	64-разрядное целое без знака (unsigned) в десятичном виде	Этот ключ задает верхнее пороговое значение для события. Как только событие достигнет порогового значения (будет больше или равно ему), приемник получит уведомление. **Примечание:** Поскольку применяется 64-разрядное целое, для некоторых источников событий могут быть предусмотрены ограничения на то, какие значения можно действительно отслеживать. Например, допустимые значения для **THRESH_HI** для источника событий **utilFs** лежат в диапазоне от 1 до 100 включительно. Допустимость порога для источника событий проверяется не во время записи, а во время вызова **select()** или вызова **read()** с объединением в блоки.
THRESH_LO	64-разрядное целое без знака (unsigned) в десятичном виде	Этот ключ задает нижнее пороговое значение для события. Как только событие достигнет порогового значения (будет меньше или равно ему), приемник получит уведомление. **Примечание:** Поскольку применяется 64-разрядное целое, для некоторых источников событий могут быть предусмотрены ограничения на то, какие значения можно действительно отслеживать. Например, допустимые значения для **THRESH_LO** для источника событий **utilFs** лежат в диапазоне от 1 до 100 включительно. Допустимость порога для источника событий проверяется не во время записи, а во время вызова **select()** или вызова **read()** с объединением в блоки.

Таблица 72. Допустимые пары <ключ>=<значение> (продолжение)

Ключ	Допустимые значения	Сведения
WAIT_TYPE	WAIT_IN_SELECT (по умолчанию), WAIT_IN_READ	Указывает, каким образом приемник будет ожидать события. Если приемник предпочитает при работе с событием применять блоки в вызове **select()**, необходимо указать **WAIT_IN_SELECT**. Если приемник предпочитает при работе с событием применять блоки в вызове **read()**, необходимо указать **WAIT_IN_READ**.
INFO_LVL	1, 2 (по умолчанию) или 3	Указывает, какие данные событий должны регистрироваться в буфере пользователя: • **INFO_LVL=1** позволяет регистрировать системное время события, порядковый номер, код возврата источника событий, информацию о пользователе*, информацию о процессе*, имя программы* и текущее значение события (если применимо). • **INFO_LVL=2** позволяет регистрировать все данные из уровня 1, а также сообщения из источника событий, если применимо. • **INFO_LVL=3** позволяет регистрировать все данные из уровня 2, а также стек события, если применимо. **Примечание:** Информация о пользователе, информация о процессе, имя программы и трассировка стека доступны только для источников событий, указавших флаг **AHAFS_STKTRACE_AVAILABLE**. Не все источники событий передают сообщения. Для того чтобы определить доступную информацию, обратитесь к документации по источнику событий. Примеры вывода события показаны в разделе “Чтение данных события” на стр. 624.
NOTIFY_CNT	-1 (по умолчанию) или любое значение от 1 до 32767 включительно	Ключ **NOTIFY_CNT** указывает, сколько раз должно произойти событие, прежде чем произойдет уведомление процесса. Если указано значение -1, приемник будет уведомляться после каждого вхождения события, а каждое вхождение события будет регистрироваться в буфере пользователя. Если приемник укажет положительное ненулевое значение, приемник будет заблокирован до тех пор, пока событие не произойдет заданное число раз. Как только событие произойдет заданное число раз, регистрация событий прекратится до тех пор, пока приемник не будет заблокирован в другом вызове **select()** или блокирующем вызове **read()**. Дополнительная информация приведена в разделе “Ожидание возникновения событий” на стр. 621.

Таблица 72. Допустимые пары <ключ>=<значение> (продолжение)

Ключ	Допустимые значения	Сведения
CLUSTER	YES	Если система является частью кластера, а кластер активен, приемники могут указать этот ключ, чтобы получать уведомления о вхождениях данного события на других узлах кластера. Не все источники событий поддерживают отслеживание на уровне кластера. По умолчанию эта функция **выключена**. Дополнительная информация приведена в разделе “События кластера” на стр. 642.
BUF_SIZE	Положительное целое до 1048576	Этот ключ задает размер буфера в байтах, который должен использоваться для записи данных события. Размер по умолчанию равен 2048, а минимальный размер, выделяемый для буфера, составляет 1024 байта, даже если приемник запрашивает меньший размер.

Запись информации в файл монитора - это только подготовка к последующему вызову **select()** или вызову чтения блоками **read()**. Отслеживание начинается только после выполнения функции **select()** или функции чтения блоками **read()**.

Например, для того чтобы отслеживать файловую систему **/filesys/clj-fs** на первое вхождение события использования на 90% в вызове чтения блоками **read()**, в файл **/aha/fs/utilFs.monFactory/filesys/clj-fs.mon** записывается следующая строка:

```
WAIT_TYPE=WAIT_IN_READ THRESH_HI=90 NOTIFY_CNT=1
```

В файл монитора из вызова **write()** могут быть внесены следующие коды возврата:

Таблица 73. Коды возврата

Код возврата	Сведения
EINVAL	Если какому-либо из вышеуказанных ключей присвоено недопустимое значение, произойдет сбой записи в файл монитора с кодом **EINVAL**. Кроме того, если указанные параметры уведомления (**CHANGED** или **THRESH_HI/LO**) не соответствуют функциям источника событий, произойдет сбой записи с кодом **EINVAL**. Например, если приемник укажет **CHANGED=YES** для источника событий **utilFs** (с отслеживанием только **THRESH_HI/LO**), вызов записи возвратит **EINVAL**. Указание **CLUSTER=YES** без активного кластера также приведет к выдаче кода **EINVAL**.
EBUSY	Если в процессе есть другая нить, ожидающая событие, запись в файл монитора любой другой нитью возвратит **EBUSY**.
ESTALE	Файл монитора был удален. Для того чтобы отслеживать это событие потребуется закрыть файл дескриптора, а затем открыть его с **O_CREAT**
ENOMEM	Не удалось выделить временную память или память для буфера событий.
ENOSPC	Файл монитора может отслеживаться не более чем 512 процессами. Если уже имеется 512 процессов с этим открытым файлом и они выполнили запись в этот файл, произойдет сбой записи с кодом **ENOSPC**.

Ожидание возникновения событий

Спецификации отслеживания записываются в файл монитора.

После успешной записи спецификаций отслеживания в файл монитора процесс приемника создаст блоки для вхождения событий с помощью **select()** или **read()**. Приемники уведомляются о событиях только после их объединения в блоки в **select()** или **read()**. Процесс может выполнить возврат из вызова **select()** или вызова **read()** с объединением в блоки тремя способами:

1. Событие возникло заданное число раз.
   - Ошибок нет. Приемник должен прочитать данные события, чтобы определить способ обработки события.
2. При настройке события внутри расширения ядра Инфраструктуры обработки событий AIX возникла ошибка.

   Ошибки могут произойти до регистрации события для отслеживания в источнике событий:
   - **read()**
     - Если в операции чтения имеется другая ожидающая нить, произойдет сбой чтения с выдачей **EBUSY**
     - Если перед этой операцией чтения не было операции записи, операция чтения возвратит 0 и число прочитанных байт будет равно 0.
   - **select()**

     Примечание:

     Вследствие реализации системного вызова select, для того чтобы **select()** возвратил ошибку, базовые операции файловой системы должны возвратить **EBADF**. Поэтому при соблюдении любого из следующих условий вызов **select()** возвратит **EBADF**.
     - Другая нить пытается вызвать select
     - Файл монитора был удален
     - Не было выполнено ни одной операции записи с указанием спецификаций отслеживания
     - Произошла ошибка при регистрации в подсистеме IOS

     В этих случаях данные событий для чтения будут отсутствовать.
3. Произошла ошибка при настройке события в источнике событий.

   Если выполнена попытка регистрации события в источнике событий, в буфер будет внесена запись, предназначенная для чтения приемником. Для того чтобы определить возникшую ошибку, **RC_FROM_EVPROD**, возвращенный в данных события, должен быть упомянут в документации источника событий. Обратите внимание, что в выводе события в этом случае будет содержаться только системное время, порядковый номер и код возврата из источника событий, независимо от того, какой **INFO_LVL** был задан. См. пример в разделе “Чтение данных события” на стр. 624.

   В этом случае **select()** возвратит **EBADF**, а **read()** возвратит код возврата из базовой операции **uio_move**.

Если процесс приемника указал **NOTIFY_CNT** больше единицы, информация о каждом вхождении события будет регистрироваться в буфере приемника, пока не произойдет запрос на число событий. Процесс приемника будет включен только в том случае, если событие произойдет запрошенное число раз или если произойдет недоступное событие. После включения процесса приемника, он больше не будет отслеживать событие до повторного вызова **select()** или вызова **read()** с объединением в блоки для файла монитора.

Если приемник указал **NOTIFY_CNT** со значением -1, процесс приемника будет включаться после каждого вхождения события и любое событие, возникшее после начального успешного вызова **select()** или вызова **read()** с объединением в блоки будет регистрироваться в буфере приемника.

Вызовы **select()** и **read()** не будут создавать блоки, если в буфере имеются непрочитанные данные события.

Недоступные вхождения события

Для некоторые источников событий могут существовать такие вхождения события, после которых отслеживаемое событие становится недопустимым.

Вот некоторые примеры:

- Уничтожение процесса для **processMon** и **pidProcessMon**.
- Размонтирование файловой системы, содержащей отслеживаемые файлы для **modDir** и **modFile**.
- Размонтирование файловой системы, отслеживаемой с помощью **utilFs**.
- Удаление или переименование файла, отслеживаемого с помощью **modDir** или **modFile**
- Удаление источника событий, используемого в данный момент для отслеживания событий (в этом случае значением **RC_FROM_EVPROD** будет **ENODEV**).

Как только будет активировано вхождение недоступного события, приемники не смогут продолжать отслеживать это событие, пока оно опять не станет допустимым. Примеры ситуаций, в которых события опять становятся допустимыми:

- Размонтирование отслеживаемой файловой системы.
- Восстановление отслеживаемого файла, который был удален.
- Восстановление процесса, который отслеживался.

Если активируется локальное недоступное событие, расширение ядра Инфраструктуры обработки событий AIX удалит затронутые файлы монитора. После удаления файла монитора приемники, в который этот файл еще открыт, смогут прочитать свои данные события, но не смогут записать или заблокировать ожидание для вхождения события в этом файле монитора. Если получатель сталкивается с недоступным вхождением события, он должен принять соответствующие меры (после которых событие должно опять стать допустимым), закрыть дескриптор файла для файла монитора и повторно открыть файл монитора с флагом **O_CREAT**.

Локальные вхождения недоступного события также приведут к разблокированию **select()** и **read()** до того, как будет активировано запрошенное число вхождений события, если приемник указал **NOTIFY_CNT > 1**. Например, если приемник отслеживает файл **/foo** с **NOTIFY_CNT=3**, приемник выполнит возвращение из **select()** или **read()**, если **/foo** был удален, даже если это первое вхождение события с **/foo**.

Применение Инфраструктуры обработки событий AIX для опроса

Инфраструктура обработки событий AIX не требует того, чтобы источники событий всегда поддерживали текущее значение событий, которые могут отслеживаться.

Это поможет повысить производительность, поскольку источники событий не будут расходовать ресурсы на обслуживание этого значения, если вхождения этого события никем не отслеживаются.

Это ведет к неполадке в случае использования синхронного опроса. Поскольку не всегда возможно получить текущее значение события в каждый момент времени, вызовы **poll()** и синхронные вызовы **select()** обрабатываются следующим образом:

- При первом вызове **select()** или **poll()** для файла монитора расширение ядра Инфраструктуры обработки событий AIX зарегистрирует это событие для отслеживания в источнике событий.
  - При работе с источниками событий на основе пороговых значений, поддерживающими текущее значение, текущее значение будет возвращено в расширение ядра Инфраструктуры обработки событий AIX после регистрации события. В этот момент данное значение будет сверено с пороговым значением приемника. Если пороговое значение приемника превышено, **select()** или **poll()** покажет, что событие произошло, и задаст **RC_FROM_EVPROD** значение **EALREADY**.
- Флаги POLLSYNC игнорируются. Событие остается зарегистрированным в источнике событий до тех пор, пока это событие не произойдет заданное число раз или пока пользователь не закроет файл.
- Для последующих вызовов **poll()** предусмотрен следующий алгоритм:
  - Если событие еще не происходило, возвращенный вызов не будет содержать ни одного события возврата
  - Если событие произошло указанное число раз с последнего вызова **poll()**, будут отправлены события возврата, показывающие, что событие произошло.

Чтение данных события

Данные события в Инфраструктуре обработки событий AIX состоят из пар ключевое слово - значение.

Данные из события могут быть прочитаны только один раз, а в одном вызове **read()** могут быть возвращены данные только одного вхождения события. Допустим, перед чтением из файла монитора приемником произошло два события, в каждом из которых содержится по 256 байт данных. Если приемник выполнит вызов **read()** для 4096 байт, пользователю будет возвращено только 256 байт первого события. Потребуется второй вызов **read()** для получения остальных данных из второго события. Любое заданное смещение будет игнорировано, а данные будут считываться с последнего непрочитанного байта.

Данные события не будут превышать 4096 байт, причем большинство событий будут иметь значительно меньший размер (< 512 байт). Рекомендуется использовать при чтении событий достаточно большой буфер, чтобы событие считывалось не частично, а полностью.

Данные событий в AHAFS состоят из пар **ключевое слово = значение**, за исключением **BUF_WRAP**, **EVENT_OVERFLOW**, **BEGIN_EVENT_INFO**, **END_EVENT_INFO**, **BEGIN_EVPROD_INFO**, **END_EVPROD_INFO** и **STACK_TRACE**, представляющих специальные ключевые слова без значений. В данных события можно встретить следующие ключевые слова:

Таблица 74. Ключевые слова

Ключ	Значение	Сведения
BUF_WRAP	Нет	Буфер приемника обрабатывается как циклический буфер. Если какие-либо непрочитанные данные заменяются последними данными события, то это ключевое слово будет содержаться в следующей строке, возвращенной вызовом **read()**, даже если приемник частично прочитал предыдущую запись. Последующий вызов **read()** возвратит следующее целое событие.
EVENT_OVERFLOW	Нет	Если размер данных события слишком велик для буфера данных событий приемника, это ключевое слово будет возвращено из первого вызова **read()**. Следующий вызов **read()** возвратит данные события, которые поместились в буфере. **Примечание:** При возникновении **EVENT_OVERFLOW** конечная строка **END_EVENT_INFO** не появится.
BEGIN_EVENT_INFO	Нет	Это ключевое слово обозначает начало данных для вхождения события.
END_EVENT_INFO	Нет	Это ключевое слово обозначает конец данных для указанного вхождения события.
TIME_tvsec TIME_tvnsec	Целочисленный	В этих двух полях записывается системное время вхождения события в секундах и наносекундах от начала отсчета.
SEQUENCE_NUM	Целочисленный	В этом поле записывается число вхождений события с первого успешного вызова **select()** или блокирующего вызова **read()**. Это число будет сброшено до нуля, если произойдет сбой вызова **select()** или блокирующего вызова **read()** или если приемник прекратит отслеживание события (посредством замены спецификаций отслеживания событий или в результате достижения счетчиком вхождений события указанного значения **NOTIFY_CNT**).

Таблица 74. Ключевые слова (продолжение)

Ключ	Значение	Сведения
PID	Целочисленный	ИД процесса, активировавшего возникновение события. Доступен только с источником событий, указавшим функцию **AHAFS_STKTRACE_AVAILABLE**, но не функцию **AHAFS_CALLBACK_INTRCNTX**.
UID	Целочисленный	Действующий ИД пользователя, активировавшего возникновение события. Доступен только с источником событий, указавшим функцию **AHAFS_STKTRACE_AVAILABLE**, но не функцию **AHAFS_CALLBACK_INTRCNTX**.
UID_LOGIN	Целочисленный	Идентификатор пользователя, активировавшего возникновение события. Доступен только с источником событий, указавшим функцию **AHAFS_STKTRACE_AVAILABLE**, но не функцию **AHAFS_CALLBACK_INTRCNTX**.
GID	Целочисленный	Действующий ИД группы пользователя, активировавшего возникновение события. Доступен только с источником событий, указавшим функцию **AHAFS_STKTRACE_AVAILABLE**, но не функцию **AHAFS_CALLBACK_INTRCNTX**.
PROG_NAME	Строка	Имя процесса, активировавшего возникновение события. Доступен только с источником событий, указавшим функцию **AHAFS_STKTRACE_AVAILABLE**, но не функцию **AHAFS_CALLBACK_INTRCNTX**.
CURRENT_VALUE	64-разрядное значение типа unsigned integer в десятичном виде	Этот тип предназначен только для источников событий **AHAFS_THRESHOLD_VALUE** и возвращает текущее значение события в момент обнаружения возникновения события. Обратите внимание, что из-за задержки между моментом, когда процесс получил уведомление, и моментом чтения данных события, фактическое текущее значение события может оказаться измененным.
RC_FROM_EVPROD	32-разрядное целочисленное значение в десятичном виде	Этот код возврата поступает из источника событий либо в результате ошибки при попытке настроить событие, либо в результате возникновения события. Как правило, коды возврата менее 256 указывают на ошибку при попытке зарегистрировать событие в источнике событий. Некоторые источники событий возвращают коды более 256 для предоставления дополнительной информации о возникновении события. Эти коды возврата описаны в файле **sys/ahafs_evProds.h**

Таблица 74. Ключевые слова (продолжение)

Ключ	Значение	Сведения
BEGIN_EVPROD_INFO **END_EVPROD_INFO**	Строка*	Эти два ключевых слова помечают начало и конец строки, возвращенной источником событий. После **BEGIN_EVPROD_INFO** и перед **END_EVPROD_INFO** всегда будет находиться разрыв строки. Для получателей, задавших **CLUSTER=YES**, в этом месте будет указана информация об узле.
STACK_TRACE	Строка*	Для получателей, задавших **INFO_LVL=3**, с источником событий, указавшим функцию **AHAFS_STKTRACE_AVAILABLE**, но не функцию **AHAFS_CALLBACK_INTRCNTX**, будет предоставлена трассировка стека данного вхождения события. Ключевое слово **STACK_TRACE** указывает, что остальные данные события, вплоть до строки **END_EVENT_INFO**, представляют собой трассировку стека этого вхождения события.
NUM_EVDROPS_INTRCNTX	Целочисленный	Это ключевое слово представляет число отброшенных событий прерывания контекста со времени, указанного в ключевых словах отчета **TIME0_tvsec** и **TIME0_tvnsec**. Экземпляры событий отбрасываются только при высокой частоте их появления.
TIME0_tvsec **TIME0_tvnsec**	Целочисленный	Эти ключевые слова записывают системное время первого отброшенного события в секундах и нано-секундах с начала отсчета времени. Эти ключевые слова сообщаются вместе с ключевым словом **NUM_EVDROPS_INTRCNTX**.

Объединение повторяющихся событий:

Если одно и то же событие возникает несколько раз перед прочтением данных приемником, повторяющиеся записи будут объединены в одну запись. На такое объединение будут указывать непоследовательные порядковые номера без соответствующего ключевого слова **BUF_WRAP**. Самое последнее вхождение события можно будет определить по системному времени и порядковому номеру.

Пример данных события

Для источника событий, задавшего **AHAFS_THRESHOLD_STATE** и **AHAFS_STKTRACE_AVAILABLE** и передающего сообщение приемникам событий, вывод представлен следующими тремя уровнями:

INFO_LVL=1	INFO_LVL=2	INFO_LVL=3
BEGIN_EVENT_INFO TIME_tvsec=1269863383 TIME_tvnsec=455993143 SEQUENCE_NUM=0 PID=6947038 UID=0 UID_LOGIN=0 GID=0 PROG_NAME=cat RC_FROM_EVPROD=1000 END_EVENT_INFO	BEGIN_EVENT_INFO TIME_tvsec=1269863383 TIME_tvnsec=455993143 SEQUENCE_NUM=0 PID=6947038 UID=0 UID_LOGIN=0 GID=0 PROG_NAME=cat RC_FROM_EVPROD=1000 BEGIN_EVPROD_INFO место для сообщения источника событий END_EVPROD_INFO END_EVENT_INFO	BEGIN_EVENT_INFO TIME_tvsec=1269863383 TIME_tvnsec=455993143 SEQUENCE_NUM=0 PID=6947038 UID=0 UID_LOGIN=0 GID=0 PROG_NAME=cat RC_FROM_EVPROD=1000 BEGIN_EVPROD_INFO место для сообщения источника событий END_EVPROD_INFO STACK_TRACE ahafs_prod_callback+3C4 ahafs_cbfn_wrapper+30 ahafs_vn_write+204 vnop_rdwr+7E4 vno_rw+B4 rwuio+12C rdwr+184 kewrite+16C .svc_instr write+1A4 _xwrite+6C _xflsbuf+B0 __flsbuf+9C copyopt_ascii+2C0 scat+388 main+11C __start+68 END_EVENT_INFO

Для источника событий, задавшего **AHAFS_THRESHOLD_VALUE_HI**, но не указавшего **AHAFS_STKTRACE_AVAILABLE**, и передающего сообщение приемникам событий, вывод представлен следующими тремя уровнями:

INFO_LVL=1	INFO_LVL=2	INFO_LVL=3
BEGIN_EVENT_INFO TIME_tvsec=1269866715 TIME_tvnsec=16678418 SEQUENCE_NUM=0 CURRENT_VALUE=3 RC_FROM_EVPROD=1000 END_EVENT_INFO	BEGIN_EVENT_INFO TIME_tvsec=1269866715 TIME_tvnsec=16678418 SEQUENCE_NUM=0 CURRENT_VALUE=3 RC_FROM_EVPROD=1000 BEGIN_EVPROD_INFO место для сообщения источника событий END_EVPROD_INFO END_EVENT_INFO	BEGIN_EVENT_INFO TIME_tvsec=1269866715 TIME_tvnsec=16678418 SEQUENCE_NUM=0 CURRENT_VALUE=3 RC_FROM_EVPROD=1000 BEGIN_EVPROD_INFO место для сообщения источника событий END_EVPROD_INFO END_EVENT_INFO

Формат ошибки:

Если из источника событий получена ошибка, все источники событий будут иметь следующий формат для всех **INFO_LVL**:

```
BEGIN_EVENT_INFO
TIME_tvsec=1269868036
TIME_tvnsec=966708948
SEQUENCE_NUM=0
RC_FROM_EVPROD=20
END_EVENT_INFO
```

Если приемник отслеживает событие **AHAFS_THRESHOLD_VALUE**, а текущее значение уже превысило запрошенное пороговое значение, формат ошибки будет также использован для записи этого события **EALREADY**:

```
BEGIN_EVENT_INFO
TIME_tvsec=1269868036
TIME_tvnsec=966708948
SEQUENCE_NUM=0
CURRENT_VALUE=1
RC_FROM_EVPROD=56
END_EVENT_INFO
```

BUF_WRAP и EVENT_OVERFLOW:

Если непрочитанные данные будут заменены данными из нового вхождения события, то первым выводом из **read()** для файла монитора будет ключевое слово **BUF_WRAP**. Если имеет место перенос данных в буфере И переполнение события, то сначала всегда будет находиться **BUF_WRAP**, а затем **EVENT_OVERFLOW**. Ниже приведен пример вывода из **read()** в том случае, когда имеются и перенос данных в буфере и переполнение события:

Первый вызов **read()** возвратит:

```
BUF_WRAP
```

Второй вызов **read()** возвратит:

```
EVENT_OVERFLOW
```

Третий вызов **read()** возвратит данные события, которые поместились в буфере:

```
BEGIN_EVENT_INFO
TIME_tvsec=1269863383
TIME_tvnsec=455993143
SEQUENCE_NUM=0
PID=6947038
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=cat
RC_FROM_EVPROD=1000
BEGIN_EVPROD_INFO
место для сообщения источника событий
END_EVPROD_INFO
STACK_TRACE
ahafs_prod_callback+3C4
ahafs_cbfn_wrapper+30
ahafs_vn_write+204
vnop_rdwr+7E4
vno_rw+B4
rwuio+12C
rdwr+184
kewrite+16C
.svc_instr
write+1A4
_xwri
```

Если информация событий поступает достаточно быстро, возможно получение двух записей **BUF_WRAP** подряд. При появлении **BUF_WRAP** увеличьте размер буфера (с помощью **BUF_SIZE** при записи в файл монитора).

NUM_EVDROPS_INTRCNTX:

Если какое-либо событие отбрасывается из-за высокой частоты появления событий, то в выводе функции **read()** для файла событий, представляющего это событие, непосредственно после строки, содержащей ключевое слово **BEGIN_EVENT_INFO**, будет указано ключевое слово **NUM_EVDROPS_INTRCNTX**.

В следующем примере показан вывод, полученный в результате вызова **read()**:

```
BEGIN_EVENT_INFO
BEGIN_EVENT_INFO
NUM_EVDROPS_INTRCNTX=5508
TIME0_tvsec=1353437661
TIME0_tvnsec=75494625
TIME_tvsec=1353437661
TIME_tvnsec=741365037
SEQUENCE_NUM=6663
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
...сообщение от источника событий...
END_EVPROD_INFO
END_EVENT_INFO
```

В этом примере вывода указана следующая информация:

- Значение `NUM_EVDROPS_INTRCNTX=5508` - это число отброшенных событий прерывания контекста с момента, указанного в полях **TIME0_tvsec** и **TIME0_tvnsec**.
- Остальная информация (то есть, `SEQUENCE_NUM=6663`, `RC_FROM_EVPROD=0`, `...сообщение от источника событий...` и другие данные) описывает событие, которое произошло в момент, указанный в полях **TIME_tvsec** и **TIME_tvnsec**.

Стандартные источники событий

modFile

Источник событий **modFile** отслеживает изменения содержимого файла.

Обзор

Источник событий **modFile** находится в каталоге **fs** и отслеживает изменения файла. Отслеживаются следующие операции **vnode**: **vnop_rdwr()**, **vnop_map_lloff()**, **vnop_remove()**, **vnop_ftrunc()**, **vnop_fclear()** и **vnop_rename()**. Нельзя отслеживать изменения, которые не проходят через уровень LFS (запись в преобразованные файлы).

Нельзя отслеживать файлы, если:

- они находятся в удаленной файловой системе;
- они находятся в файловой системе типа **ahafs**, **procfs** или **namefs**;
- они представляют собой ссылку;
- они находятся внутри каталога, оканчивающегося расширением инфраструктуры обработки событий AIX (**.mon**, **.list**, **.monFactory**);
- файлы монитора, полное имя которых превышает **MAXPATHLEN** в файловой псевдосистеме Инфраструктуры обработки событий AIX, не могут отслеживаться.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_STKTRACE_AVAILABLE
AHAFS_REMOTE_EVENT_ENABLED
```

Коды возврата

Источник событий **modFile** использует коды возврата, определенные в **<sys/ahafs_evProds.h>**.

Эти коды возврата служат для указания того, каким образом было изменено содержимое отслеживаемого каталога:

AHAFS_MODFILE_WRITE
Отслеживаемый файл записан в указанное расположение.

AHAFS_MODFILE_UNMOUNT
Файловая система, содержащая отслеживаемый файл, была размонтирована. Это недоступное событие.

AHAFS_MODFILE_MAP
Процесс выполнил преобразование фрагмента отслеживаемого файла для записи.

AHAFS_MODFILE_REMOVE
Отслеживаемый файл был удален. Это недоступное событие.

AHAFS_MODFILE_RENAME
Отслеживаемый файл был переименован. Это недоступное событие.

AHAFS_MODFILE_FCLEAR
Процесс запустил команду **fclear** для отслеживаемого файла.

AHAFS_MODFILE_FTRUNC
Процесс запустил команду **ftrunc** для отслеживаемого файла.

AHAFS_MODFILE_OVERMOUNT
Отслеживаемый файл был смонтирован повторно.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания изменений файла необходимо создать в каталоге **modFile.monFactory** файл монитора с таким же путем, как у файла, который требуется отслеживать. Например, для отслеживания **/etc/passwd** будет использоваться файл монитора **/aha/fs/modFile.monFactory/etc/passwd.mon**.

Пример данных события

На основе процесса с записью в файл монитора были созданы следующие данные события. Это вывод, наблюдаемый, если **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271703118
TIME_tvnsec=409201093
SEQUENCE_NUM=0
PID=5701678
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=cat
RC_FROM_EVPROD=1000
STACK_TRACE
aha_cbfn_wrapper+30
ahafs_evprods+510
aha_vn_write+154
vnop_rdwr+7E8
vno_rw+B4
rwuio+100
rdwr+188
kewrite+104
.svc_instr
write+1A4
_xwrite+6C
_xflsbuf+A8
__flsbuf+C0
copyopt+2E8
scat+22C
main+11C
__start+68
END_EVENT_INFO
```

modFileAttr

Источник событий **modFileAttr** отслеживает изменения атрибутов файла.

Обзор

Источник событий **modFileAttr** находится в каталоге **fs** и отслеживает изменения атрибутов файла или каталога: режима, принадлежности и списков управления доступом. Отслеживаются следующие операции vnode: **vnop_setattr()** (только для операций **V_OWN** и **V_MODE**), **vnop_setacl()**, **vnop_setxacl()**, **vnop_remove()**, **vnop_rename()** и **vnop_rmdir()**.

Отслеживание файлов и каталогов невозможно, если:

- они находятся в удаленной файловой системе;
- они находятся в файловой системе типа **ahafs**, **procfs** или **namefs**;
- они находятся внутри каталога, оканчивающегося расширением инфраструктуры обработки событий AIX (**.mon, .list, .monFactory**).
- файлы монитора, полное имя которых превышает **MAXPATHLEN** в файловой псевдосистеме Инфраструктуры обработки событий AIX, не могут отслеживаться.

Функции

```
AHAFS_THRESHOLD_STATE

AHAFS_STKTRACE_AVAILABLE

AHAFS_REMOTE_EVENT_ENABLED
```

Коды возврата

Источник событий **modFileAttr** использует коды возврата, определенные в **<sys/ahafs_evProds.h>**.

Эти коды возврата служат для указания того, каким образом было изменено содержимое отслеживаемого каталога:

AHAFS_MODFILEATTR_UNMOUNT
Файловая система, содержащая отслеживаемый файл или каталог, была размонтирована. Это недоступное событие.

AHAFS_MODFILEATTR_REMOVE
Отслеживаемый файл или каталог был удален. Это недоступное событие.

AHAFS_MODFILEATTR_RENAME
Отслеживаемый файл или каталог был переименован. Это недоступное событие.

AHAFS_MODFILEATTR_OVERMOUNT
Отслеживаемый файл или каталог был перемонтирован. Это недоступное событие.

AHAFS_MODFILEATTR_SETACL
Списки управления доступом отслеживаемого файла или каталога были изменены.

AHAFS_MODFILEATTR_SETOWN
Принадлежность отслеживаемого файла или каталога была изменена.

AHAFS_MODFILEATTR_SETMODE
Режим отслеживаемого файла или каталога был изменен.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания изменений файла необходимо создать в каталоге **modFileAttr.monFactory** файл монитора с таким же путем, как у файла, который требуется отслеживать. Например, для отслеживания **/etc/passwd** будет использоваться файл монитора **/aha/fs/modFileAttr.monFactory/etc/passwd.mon**.

Пример данных события

На основе процесса, изменяющего режим отслеживаемого файла, были созданы следующие данные события. Это вывод, наблюдаемый, если **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1291994430
TIME_tvnsec=760097298
SEQUENCE_NUM=0
PID=5767216
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=chmod
RC_FROM_EVPROD=1010
STACK_TRACE
ahafs_evprods+70C
aha_process_attr+120
vnop_setattr+21C
vsetattr@AF13_1+20
setnameattr+B4
chmod+110
.svc_instr
change+3C8
main+190
__start+68
END_EVENT_INFO
```

modDir

Источник событий **modDir** отслеживает изменения содержимого каталога.

Обзор

Источник событий **modDir** находится в каталоге **fs** и отслеживает изменения содержимого каталога. Отслеживаются следующие операции **vnode**: **vnop_create()**, **vnop_link()**, **vnop_symlink()**, **vnop_remove()**, **vnop_rename()**, **vnop_mkdir()** и **vnop_rmdir()**.

Нельзя отслеживать каталоги, если:

- они находятся в удаленной файловой системе;
- они находятся в файловой системе типа **ahafs**, **procfs** или **namefs**;
- они представляют собой ссылку;
- они находятся внутри каталога, оканчивающегося расширением инфраструктуры обработки событий AIX (**.mon**, **.list**, **.monFactory**);
- файлы монитора, полное имя которых превышает **MAXPATHLEN** в файловой псевдосистеме Инфраструктуры обработки событий AIX, не могут отслеживаться.

Источник событий **modDir** не отслеживает изменения каталога рекурсивно. Отслеживаются только изменения заданного каталога.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_STKTRACE_AVAILABLE
AHAFS_REMOTE_EVENT_ENABLED
```

Коды возврата

Источник событий **modDir** использует коды возврата, определенные в **<sys/ahafs_evProds.h>**.

Эти коды возврата служат для указания того, каким образом было изменено содержимое отслеживаемого каталога:

AHAFS_MODDIR_CREATE
В отслеживаемом каталоге создан новый объект файловой системы.

AHAFS_MODDIR_UNMOUNT
Файловая система, содержащая отслеживаемый каталог, была размонтирована. Это недоступное событие.

AHAFS_MODDIR_REMOVE
Объект файловой системы в отслеживаемом каталоге был удален.

AHAFS_MODDIR_REMOVE_SELF
Сам отслеживаемый каталог был удален или переименован. Это недоступное событие.

Сообщение источника событий

Имя объекта файловой системы, активировавшего событие, включено в данные события.

Допустимые файлы монитора

Для отслеживания изменений содержимого каталога необходимо создать в каталоге **modDir.monFactory** файл монитора с таким же путем, как у каталога, который требуется отслеживать. Например, для отслеживания изменений в каталоге **/home/cheryl** будет использоваться файл монитора **/aha/fs/modDir.monFactory/home/cheryl.mon**.

Пример данных события

На основе нового файла с именем **file1**, создаваемого в отслеживаемом каталоге, были созданы следующие данные события. Это вывод, наблюдаемый, если **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271780397
TIME_tvnsec=24369022
SEQUENCE_NUM=0
PID=6095102
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=touch
RC_FROM_EVPROD=1000
BEGIN_EVPROD_INFO
file1
END_EVPROD_INFO
STACK_TRACE
aha_cbfn_wrapper+30
ahafs_evprods+510
aha_process_vnop+138
vnop_create_attr+4AC
openpnp+418
openpath+100
copen+294
kopen+1C
.svc_instr
open+F8
creat64+1C
main+1EC
__start+68
END_EVENT_INFO
```

utilFs

Источник событий **utilFs** отслеживает использование файловой системы.

Обзор

Источник событий **utilFs** отслеживает использование файловой системы в процентах. Он находится в каталоге **fs**. В настоящее время отслеживание **utilFs** поддерживается только в файловых системах JFS2. После каждой операции записи файла, создания файла и удаления файла проверяется использование файловой системы на предмет достижения или превышения заданного порогового значения. Некоторые операции, характерные для определенной файловой системы, могут повлиять на показатель использования файловой системы, причем **utilFs** может их не обнаружить до следующей операции записи, создания или удаления файла. Превышение пороговых значений в результате удаления файловых объектов не будет сопровождаться уведомлением, пока не произойдет следующая операция записи, создания или удаления файла.

Файловые системы, полное имя файла монитора которых превышает **MAXPATHLEN** в AHAFS, отслеживаться не могут.

Во избежание чрезмерного поступления уведомлений о событии и возможного снижения производительности настоятельно рекомендуется при отслеживании событий **utilFs** задавать для **NOTIFY_CNT** значение 1.

Функции

```
AHAFS_THRESHOLD_VALUE_HIGH
AHAFS_THRESHOLD_VALUE_LOW
AHAFS_REMOTE_EVENT_ENABLED
```

Указанные пороговые значения должны находиться в диапазоне от 1 до 100 включительно.

Коды возврата

Источник событий **utilFs** использует коды возврата, определенные в **<sys/ahafs_evProds.h>**.

Эти коды возврата служат для указания того, каким образом было изменено содержимое отслеживаемого каталога:

AHAFS_UTILFS_THRESH_HIT
Отслеживаемая файловая система достигла указанного порогового значения.

AHAFS_UTILFS_UNMOUNT
Отслеживаемая файловая система была размонтирована. Это недоступное событие.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания использования файловой системы необходимо создать в каталоге **utilFs.monFactory** файл монитора с таким же путем, как у точки монтирования отслеживаемой файловой системы, который требуется отслеживать. Например, для отслеживания файловой системы **/data/fs1** будет использоваться файл монитора **/aha/fs/utilFs.monFactory/data/fs1.mon**.

Пример данных события

Ниже приведены данные события из события **AHAFS_UTILFS_THRESH_HIT**, если значение **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271705858
TIME_tvnsec=704241888
SEQUENCE_NUM=0
CURRENT_VALUE=10
RC_FROM_EVPROD=1000
END_EVENT_INFO
```

waitTmCPU

Источник событий **waitTmCPU** отслеживает среднее время ожидания выполнимых нитей.

Обзор Источник событий **waitTmCPU** отслеживает среднее время в миллисекундах ожидания выполнимых нитей, ожидающих получения времени CPU с интервалами в одну секунду. **waitTmCPU** находится в каталоге **cpu**.

Функции

```
AHAFS_THRESHOLD_VALUE_HIGH
AHAFS_CALLBACK_INTRCNTX
AHAFS_REMOTE_EVENT_ENABLED
```

Указанные пороговые значения должны быть больше 0.

Коды возврата

Этот источник событий при возникновении события всегда возвращает 0.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания этого события необходимо использовать следующий файл монитора:

```
/aha/cpu/waitTmCPU.monFactory/waitTmCPU.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **waitTmCPU**, если значение **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271779504
TIME_tvnsec=18056777
SEQUENCE_NUM=0
CURRENT_VALUE=4
RC_FROM_EVPROD=0
END_EVENT_INFO
```

waitersFreePg

Источник событий **waitersFreePg** отслеживает число нитей, ожидающих свободного кадра.

Обзор Источник событий **waitersFreePg** отслеживает число нитей, ожидающих свободного кадра с интервалами в одну секунду. **waitersFreePg** находится в каталоге **mem**.

Функции

```
AHAFS_THRESHOLD_VALUE_HIGH
AHAFS_CALLBACK_INTRCNTX
AHAFS_REMOTE_EVENT_ENABLED
```

Указанные пороговые значения должны быть больше 0.

Коды возврата

Этот источник событий при возникновении события всегда возвращает 0.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания этого события необходимо использовать следующий файл монитора:

```
/aha/mem/waitersFreePg.monFactory/waitersFreePg.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **waitersFreePg**, если значение **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271779680
TIME_tvnsec=347233732
SEQUENCE_NUM=0
CURRENT_VALUE=19843
RC_FROM_EVPROD=0
END_EVENT_INFO
```

waitTmPgInOut

Источник событий **waitTmPgInOut** отслеживает среднее время ожидания в миллисекундах для нитей, ожидающих операций загрузки страницы или выгрузки страницы.

Обзор Источник событий **waitTmPgInOut** отслеживает среднее время ожидания в миллисекундах для

нитей, ожидающих выполнения операций загрузки страницы или выгрузки страницы дольше одной секунды. Источник событий **waitTmPgInOut** находится в каталоге **mem**.

Функции

```
AHAFS_THRESHOLD_VALUE_HIGH
AHAFS_CALLBACK_INTRCNTX
AHAFS_REMOTE_EVENT_ENABLED
```

Указанные пороговые значения должны быть больше 0.

Коды возврата

Этот источник событий при возникновении события всегда возвращает 0.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания этого события необходимо использовать следующий файл монитора:

```
/aha/mem/waitTmPgInOut.monFactory/waitTmPgInOut.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **waitTmPgInOut**, если значение **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271779359
TIME_tvnsec=941699413
SEQUENCE_NUM=0
CURRENT_VALUE=12
RC_FROM_EVPROD=0
END_EVENT_INFO
```

vmo

Источник событий **vmo** отслеживает изменения переменных **vmo**.

Обзор

Источник событий **vmo** находится в каталоге **mem** и отслеживает изменения следующих переменных **vmo**.

Примечание: Команда **vmo** является самодокументируемой. Некоторые из переменных, приведенных в следующем списке, могут не поддерживаться.

- **npskill**
- **npswarn**
- **force_relalias_lite**
- **low_ps_handling**
- **maxpin%** (должна отслеживаться как файл **maxpin_pct.mon**)
- **nokilluid**
- **relalias_percentage**
- **vmm_default_pspa**
- **npsrpgmin**
- **npsrpgmax**
- **npsscrubmin**
- **npsscrubmax**
- **scrubclean**
- **rpgcontrol**

- **rpgclean**
- **vm_modlist_threshold**
- **vmm_fork_policy**
- **lru_poll_interval**

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_STKTRACE_AVAILABLE
AHAFS_REMOTE_EVENT_ENABLED
```

Коды возврата

Этот источник событий при возникновении события всегда возвращает 0.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания приведенных выше переменных необходимо использовать файлы монитора следующего формата:

```
/aha/mem/vmo.monFactory/<tunable>.mon
```

В этом каталоге нельзя создавать файлы, не относящиеся к вышеупомянутым событиям.

Пример данных события

Ниже приведены данные события, полученные при изменении отслеживаемой переменной, если значение **INFO_LVL** равно 3.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271770698
TIME_tvnsec=787565808
SEQUENCE_NUM=0
PID=5701808
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=vmo
RC_FROM_EVPROD=0
STACK_TRACE
aha_cbfn_wrapper+30
ahafs_evprods+510
vm_mon_tunable+B0
vm_chk_mod_tun+5CC
_vmgetinfo+53C
vmgetinfo+48
.svc_instr
vmo_write_vmsetkervars+134
vmo_write_dynamic_values+404
main+BC
__start+70
END_EVENT_INFO
```

schedo

Этот источник событий отслеживает изменения переменных **schedo**.

Обзор В настоящий момент можно отслеживать только переменную **vpm_xvcpus**. Этот источник событий при возникновении события возвращает трассировку стека и информацию о пользователе. Этот источник событий находится в каталоге **cpu**.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_STKTRACE_AVAILABLE
AHAFS_REMOTE_EVENT_ENABLED
```

Коды возврата

Этот источник событий при возникновении события всегда возвращает 0.

Сообщение источника событий

Этот источник событий не передает со своими данными события никаких сообщений.

Допустимые файлы монитора

Для отслеживания этой переменной применяется следующий файл монитора:

```
/aha/cpu/schedo.monFactory/vpm_xvcpus.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события, полученные при изменении переменной **vpm_xvcpus**, если значение **INFO_LVL** равно 3:

```
BEGIN_EVENT_INFO
TIME_tvsec=1271771009
TIME_tvnsec=251723285
SEQUENCE_NUM=0
PID=7143474
UID=0
UID_LOGIN=0
GID=0
PROG_NAME=schedo
RC_FROM_EVPROD=0
STACK_TRACE
aha_cbfn_wrapper+30
ahafs_evprods+510
schedtune+394
.svc_instr
schedo_write_schedparams+94
schedo_write_dynamic_values+6F0
main+1B0
__start+68
END_EVENT_INFO
```

pidProcessMon

Источник событий **pidProcessMon** отслеживает уничтожение процесса на основе PID.

Обзор Источник событий **pidProcessMon** находится в каталоге **cpu** и отслеживает уничтожение процесса на основе PID.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

Источник событий **pidProcessMon** возвращает только один код возврата - 0.

Сообщение источника событий

Этот источник событий передает сообщение **PROCESS_DOWN** как часть своих данных события.

Допустимые файлы монитора

Для отслеживания уничтожения процессов необходимо создать в каталоге **pidProcessMon.monFactory** файл монитора. Необходимо использовать имя файла монитора в формате

```
/aha/cpu/pidProcessMon.monFactory/<PID-процесса>.mon
```

Пример данных события

При уничтожении отслеживаемого процесса были созданы следующие данные события. Это вывод, наблюдаемый со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1272348759
TIME_tvnsec=379259175
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=PROCESS_DOWN
END_EVPROD_INFO
END_EVENT_INFO
```

processMon

Источник событий **processMon** отслеживает уничтожение процесса.

Обзор Источник событий **processMon** находится в каталоге cpu и отслеживает уничтожение процесса на основе имени процесса. Отслеживается только родительский процесс для указанного процесса с тем же именем. Это означает, если имеется структура процессов **abc (pid 123)->xyz (pid 345)->xyz (pid 567)** и кто-то запросил отслеживать процесс **xyz**, то фактически будет отслеживаться процесс (**pid = 345**).

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

Источник событий **processMon** возвращает только один код возврата - 0.

Сообщение источника событий

Этот источник событий передает сообщение **PROCESS_DOWN** как часть своих данных события.

Допустимые файлы монитора

Для отслеживания уничтожения процессов необходимо создать в каталоге **processMon.monFactory** файл монитора с таким же путем, который используется для запуска процесса. Например, для отслеживания процесса с именем **test**, расположенного в каталоге **/usr/samples/ahafs**, будет использоваться файл монитора **/aha/cpu/processMon.monFactory/usr/samples/ahafs/test.mon**.

Пример данных события

При уничтожении отслеживаемого процесса были созданы следующие данные события. Это вывод, наблюдаемый со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1272348909
TIME_tvnsec=482502597
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=PROCESS_DOWN
END_EVPROD_INFO
END_EVENT_INFO
```

inetsock

Источник событий **inetsock** отслеживает операции сокетов Transmission Control Protocol (TCP) И User Datagram Protocol (UDP).

Обзор Источник событий **inetsock** размещается в каталоге net и отслеживает операции сокетов.

Для TCP отслеживаются следующие операции сокетов:

- Создание сокета
- Связывание порта или адреса с сокетом
- Отслеживание сокета
- Прием и установка соединения через сокет
- Установка соединения с сокетом

- Отсоединение от сокета
- Закрытие сокета

Для UDP отслеживаются следующие операции сокетов:

- Создание сокета
- Связывание порта или адреса с сокетом
- Закрытие сокета

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_CALLBACK_INTRCNTX
AHAFS_REMOTE_EVENT_ENABLED
```

Сообщение источника событий

Этот источник событий передает информацию из управляющего блока протокола и сокета в составе данных о событии.

Для операций сокета TCP передаются следующие данные:

Операция сокета	Данные
PRU_ATTACH	Общие сведения: • PROG_NAME • SO_FAMILY • SO_PID • SO_PROTO • SO_TYPE • SO_UID
PRU_BIND	Общие сведения и следующие элементы: • SO_LADDR • SO_LPORT
PRU_LISTEN	Общие сведения и следующие элементы: • SO_LADDR • SO_LPORT
PRU_ACCEPT	Общие сведения и следующие элементы: • SO_FADDR • SO_FPORT • SO_LADDR • SO_LPORT
PRU_CONNECT	Общие сведения и следующие элементы: • SO_FADDR • SO_FPORT • SO_LADDR • SO_LPORT
PRU_DISCONNECT	Общие сведения и следующие элементы: • SO_FADDR • SO_FPORT • SO_LADDR • SO_LPORT
PRU_DETACH, PRU_ABORT	Общие сведения и следующие элементы: • SO_LADDR • SO_LPORT • SO_FADDR • SO_FPORT

Для операций сокета UDP передаются следующие данные:

Операция сокета	Данные
PRU_ATTACH	Общие сведения: • PROG_NAME • SO_FAMILY • SO_PID • SO_PROTO • SO_TYPE • SO_UID
PRU_BIND, PRU_DYNBIND	Общие сведения и следующие элементы: • SO_LADDR • SO_LPORT
PRU_DETACH, PRU_ABORT	Общие сведения и следующие элементы: • SO_LADDR • SO_LPORT

Допустимые файлы монитора

Для отслеживания работы сокета необходимо в каталоге `inetsock.monFactory` создать файл монитора с тем же именем, что и у операции сокета, которую требуется отслеживать. Например, для отслеживания создания сокета TCP используется файл монитора `/aha/net/inetsock.monFactory/streamCreate.mon`. Подобным образом, для отслеживания сокета UDP используется файл монитора `/aha/net/inetsock.monFactory/dgramCreate.mon`.

Следующие файлы используются для всех операций сокета TCP, которые может отслеживать файловая система автономного советника по работоспособности (AHAFS):

- `/aha/net/inetsock.monFactory/streamCreate.mon`
- `/aha/net/inetsock.monFactory/streamBind.mon`
- `/aha/net/inetsock.monFactory/streamListen.mon`
- `/aha/net/inetsock.monFactory/streamAccept.mon`
- `/aha/net/inetsock.monFactory/streamConnect.mon`
- `/aha/net/inetsock.monFactory/streamDisconnect.mon`
- `/aha/net/inetsock.monFactory/streamClose.mon`

Следующие файлы используются для всех операций сокета UDP, которые может отслеживать AHAFS:

- `/aha/net/inetsock.monFactory/dgramCreate.mon`
- `/aha/net/inetsock.monFactory/dgramBind.mon`
- `/aha/net/inetsock.monFactory/dgramClose.mon`

Пример данных события

На основе процесса создания сокета были созданы следующие данные о событиях. Ниже приведен пример вывода с уровнем вывода 2 (INFO_LVL=2):

```
Событие AHAFS: /aha/net/inetsock.monFactory/streamCreate.mon
---------------------------------------------------

BEGIN_EVENT_INFO
Time          : Mon Jan 23 23:04:06 2012
Sequence Num: 1
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
PROG_NAME=xmtopas
SO_FAMILY=2
SO_TYPE=1
SO_PROTO=6
```

```
SO_UID=0
SO_PID=5243048
END_EVPROD_INFO
END_EVENT_INFO
```

События кластера

Если система является частью кластера, она может получать уведомления о событиях, происходящих на других узлах, относящихся к тому же кластеру. Источники событий, которые задают функцию **AHAFS_REMOTE_EVENT_ENABLED**, поддерживают отслеживание на уровне кластера. Все источники событий Инфраструктуры обработки событий AIX, кроме **pidProcessMon** и **diskState**, могут предоставлять такие удаленные уведомления.

Работа команды **mkcluster** с инфраструктурой событий AIX:

Если Инфраструктура обработки событий AIX не загружена в системе и выполняется команда **mkcluster**, то в каталоге **/aha** будет смонтирована файловая псевдосистема Инфраструктуры обработки событий AIX, а имена файлов монитора будут начинаться с каталога **/aha**. Если Инфраструктура обработки событий AIX уже загружена в системе и выполняется команда **mkcluster**, то файловая псевдосистема Инфраструктуры обработки событий AIX не будет размонтирована, а имена файлов монитора будут начинаться с каталога, на основе которого была смонтирована файловая псевдосистема Инфраструктуры обработки событий AIX. Приложения-приемники должны проверять, где была смонтирована файловая псевдосистема Инфраструктуры обработки событий AIX, чтобы получить пути к файлам монитора.

Для получения событий кластера процессы приемника должны указать **CLUSTER=YES** во время записи в файл монитора, представляющий событие для отслеживания в кластере. Для возможности обнаружения удаленных событий процесс приемника должен отслеживать событие на каждом узле, на котором указано **CLUSTER=YES**.

События, полученные с удаленного узла, не содержат ни информации о пользователе или процессе, ни трассировки стека, даже если это поддерживается источником событий.

Для событий, полученных на удаленном узле, трассировка стека не предоставляется, даже если она поддерживается источником событий.

Информация о кластере **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID** будет находиться между ограничителями **BEGIN_EVPROD_INFO** и **END_EVPROD_INFO** для всех событий кластера. Это помогает программе отслеживания определить, на каком узле произошло событие. Информация, возвращаемая в выводе команды **lscluster -m** в полях Стенографический ИД для узла, **uuid** для узла и **uuids** кластера, возвращается в выводе события Инфраструктуры обработки событий AIX соответственно в полях **NODE_NUMBER**, **NODE_ID**, **CLUSTER_ID**.

Ниже приведен пример вывода как из локального, так и из удаленного вхождения события, в котором **INFO_LVL** равно 2, причем источник событий задает функцию **AHAFS_STKTRACE_AVAILABLE**.

Данные локального события	Данные удаленного события
BEGIN_EVENT_INFO TIME_tvsec=1262670289 TIME_tvnsec=453840229 SEQUENCE_NUM=0 PID=4194474 UID=0 UID_LOGIN=0 GID=0 PROG_NAME=rpc.statd RC_FROM_EVPROD=0 BEGIN_EVPROD_INFO NODE_NUMBER=1 NODE_ID=0xF079E8C801C11DF CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404 EVENT_TYPE=PROCESS_DOWN END_EVPROD_INFO END_EVENT_INFO	BEGIN_EVENT_INFO TIME_tvsec=1262670289 TIME_tvnsec=248144872 SEQUENCE_NUM=0 RC_FROM_EVPROD=0 BEGIN_EVPROD_INFO EVENT_TYPE=PROCESS_DOWN NODE_NUMBER=1 NODE_ID=0xF079E8C801C11DF CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404 END_EVPROD_INFO END_EVENT_INFO

Стандартные источники событий для экземпляра AIX, поддерживающего работу с кластером

Эти источники событий доступны только в том случае, если система входит в состав активного кластера.

nodeList

Источник событий **nodeList** отслеживает изменения в принадлежности к кластеру.

Обзор Источник событий **nodeList** находится в каталоге cluster и отслеживает добавление или удаление узлов из кластера. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при добавлении или удалении узла из кластера (например, с помощью команды **chcluster**).

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

nodeList возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **NODE_ADD** и **NODE_DELETE** как часть своих данных события. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке узлов необходимо создать в каталоге **nodeList.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/cluster/nodeList.monFactory/nodeListEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **nodeList** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271922590
TIME_tvnsec=886742634
SEQUENCE_NUM=1
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=NODE_ADD
NODE_NUMBER=1
```

```
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

clDiskList

Источник событий **clDiskList** отслеживает изменения в принадлежности к кластеру.

Обзор Источник событий **clDiskList** находится в каталоге disk и отслеживает добавление или удаление дисков из кластера. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при добавлении или удалении диска из кластера (например, с помощью команды **chcluster**).

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

clDiskList возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **DISK_ADD** и **DISK_DELETE** как часть своих данных события в поле **EVENT_TYPE**. Будут переданы **DISK_NAME** и **DISK_UID** соответствующего диска. Кроме того, являясь частью кластера, источник событий автоматически передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке дисков необходимо создать в каталоге **clDiskList.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/disk/clDiskList.monFactory/clDiskListEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **clDiskList** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271927983
TIME_tvnsec=696543410
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=DISK_ADD
DISK_NAME=cldisk1
DISK_UID=3E213600A0B800016726C000000FF4B8677C80F1724-100 FAStT03IBMfcp
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

linkedCl

Источник событий **linkedCl** создается при создании или удалении связи с другим кластером.

Обзор Источник событий **linkedCl** находится в каталоге cluster и отслеживает изменения состояния связей. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при создании или удалении связи с другим кластером.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

linkedCl возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **LINK_UP** или **LINK_DOWN** как часть своих данных события. Он передаст информацию о **LINK_ID**. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке узлов необходимо создать в каталоге **linkedCl.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/cluster/linkedCl.monFactory/linkedClEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **linkedCl** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271224025
TIME_tvnsec=795042625
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=LINK_DOWN
LINK_ID=0x7BE9C1BD
NODE_NUMBER=1
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

nodeContact

Источник событий **nodeContact** отслеживает состояние последнего контакта узла в кластере.

Обзор Источник событий **nodeContact** находится в каталоге cluster и отслеживает состояние последнего контакта узла в кластере. Этот источник доступен только в том случае, если система является частью кластера.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

nodeContact возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **CONNECT_UP** и **CONNECT_DOWN** как часть своих данных события. Он передаст соответствующее **INTERFACE_NAME**. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке узлов необходимо создать в каталоге **nodeContact.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/cluster/nodeContact.monFactory/nodeContactEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **nodeContact** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271921874
TIME_tvnsec=666770128
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=CONNECT_DOWN
INTERFACE_NAME=en0
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

nodeState

Источник событий **nodeState** отслеживает состояние узла в кластере.

Обзор Источник событий **nodeState** находится в каталоге cluster и отслеживает состояние узла в кластере. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается, например, при сбое или выключении узла.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

nodeState возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **NODE_UP** и **NODE_DOWN** как часть своих данных события. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений состояния узлов необходимо создать в каталоге **nodeState.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/cluster/nodeState.monFactory/nodeStateEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **nodeState** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271921536
TIME_tvnsec=68254861
SEQUENCE_NUM=1
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=NODE_UP
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

nodeAddress

Источник событий **nodeAddress** отслеживает сетевой адрес узла.

Обзор Источник событий **nodeAddress** находится в каталоге cluster и отслеживает сетевой адрес узла. Этот

источник доступен только в том случае, если система является частью кластера. Это событие создается, например, при добавлении или удалении псевдонима из сетевого интерфейса.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

nodeAddress возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **ADDRESS_ADD** и **ADDRESS_DELETE** как часть своих данных события. Он передаст **INTERFACE_NAME** соответствующего интерфейса, а также **FAMILY**, **ADDRESS** и **NETMASK** IP-адреса. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке узлов необходимо создать в каталоге **nodeAddress.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/cluster/nodeAddress.monFactory/nodeAddressEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **nodeAddress** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271922254
TIME_tvnsec=9053410
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=ADDRESS_ADD
INTERFACE_NAME=et0
FAMILY=2
ADDRESS=0x0A0A0A0A
NETMASK=0xFF000000
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

networkAdapterState

Источник событий **networkAdapterState** отслеживает сетевой интерфейс узла в кластере.

Обзор Источник событий **networkAdapterState** находится в каталоге cluster и отслеживает сетевой интерфейс узла в кластере. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при выключении или включении сетевой интерфейса.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

networkAdapterState возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **ADAPTER_UP**, **ADAPTER_DOWN**, **ADAPTER_ADD** и **ADAPTER_DEL** как часть своих данных события. Он передаст соответствующее

INTERFACE_NAME. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке узлов необходимо создать в каталоге **networkAdapterState.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/cluster/networkAdapterState.monFactory/networkAdapterStateEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **networkAdapterState** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271920539
TIME_tvnsec=399378269
SEQUENCE_NUM=1
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=ADAPTER_UP
INTERFACE_NAME=en0
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

clDiskState

Источник событий **clDiskState** отслеживает диски кластера.

Обзор Источник событий **clDiskState** находится в каталоге disk и отслеживает диски кластера. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при выключении или включении диска кластера.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

clDiskState возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **DISK_UP** и **DISK_DOWN** как часть своих данных события в поле **EVENT_TYPE** вместе с соответствующим именем диска кластера. Кроме того, являясь источником событий кластера, он также передает информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания дисков кластера необходимо создать в каталоге **clDiskState.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/disk/clDiskState.monFactory/clDiskStateEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **clDiskState** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271935734
TIME_tvnsec=265210314
SEQUENCE_NUM=1
```

```
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=DISK_DOWN
DISK_NAME=cldisk1
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

repDiskState

Источник событий **repDiskState** отслеживает диски хранилища.

Обзор Источник событий **repDiskState** находится в каталоге disk и отслеживает диски хранилища. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при выключении или включении диска хранилища.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

repDiskState возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE (-1)** возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **REP_UP** и **REP_DOWN** как часть своих данных события в поле **EVENT_TYPE** вместе с именем именем диска соответствующего диска хранилища. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания дисков хранилища необходимо создать в каталоге **repDiskState.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/disk/ repDiskState.monFactory/repDiskStateEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **repDiskState** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271933757
TIME_tvnsec=134003703
SEQUENCE_NUM=1
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=REP_UP
DISK_NAME=hdisk2
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

diskState

Источник событий **diskstate** отслеживает изменения локальных дисков.

Обзор Источник событий **diskState** находится в каталоге disk и отслеживает изменения локальных дисков. Этот источник доступен только в том случае, если система является частью кластера. Это событие создается при выключении или включении локального диска. Это событие выдается только для тех дисков, которые поддерживаются средой хранения информации.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

diskState возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE** (-1) возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает в составе данных события также сообщения **LOCAL_UP** и **LOCAL_DOWN** вместе с соответствующим именем диска. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания локальных дисков необходимо создать в каталоге **diskState.monFactory** файл монитора. Необходимо использовать имя файла монитора в формате

```
/aha/disk/diskState.monFactory/<hdiskn>.mon
```

вместе с именем локального диска, который требуется отслеживать.

Пример данных события

Ниже приведены данные события из события **diskState** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271935029
TIME_tvnsec=958362343
SEQUENCE_NUM=1
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=LOCAL_UP
DISK_NAME=hdisk4
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

vgState

Источник событий **vgstate** может проверить состояние VG на диске.

Обзор Источник событий **vgState** находится в каталоге disk. Этот источник доступен только в том случае, если система является частью кластера. При каждом возникновении события включения или выключения локального диска (зарегистрированного в **diskState**) или диска кластера активируется соответствующее событие **VG_UP** или **VG_DOWN** для группы томов, расположенной на этом диске. С помощью этого источника событий приложение может проверить состояние VG на диске с подсистемой LVM.

Функции

```
AHAFS_THRESHOLD_STATE
AHAFS_REMOTE_EVENT_ENABLED
AHAFS_CALLBACK_INTRCNTX
```

Коды возврата

vgState возвращает 0 в качестве кода возврата. **AHAFS_CLUSTER_REMOVE** (-1) возвращается только в случае удаления кластера.

Сообщение источника событий

Этот источник событий передает сообщения **VG_UP** и **VG_DOWN** как часть своих данных события. Он передаст соответствующее имя диска и имя группы томов. Кроме того, являясь источником событий кластера, он также передаст информацию **NODE_NUMBER**, **NODE_ID** и **CLUSTER_ID**.

Допустимые файлы монитора

Для отслеживания изменений в списке узлов необходимо создать в каталоге **vgState.monFactory** файл монитора. Необходимо использовать имя файла монитора

```
/aha/disk/vgState.monFactory/vgStateEvent.mon
```

В этом каталоге нельзя создавать другие файлы монитора.

Пример данных события

Ниже приведены данные события из события **vgstate** со стандартным **INFO_LVL**.

```
BEGIN_EVENT_INFO
TIME_tvsec=1271915408
TIME_tvnsec=699408296
SEQUENCE_NUM=0
RC_FROM_EVPROD=0
BEGIN_EVPROD_INFO
EVENT_TYPE=VG_UP
DISK_NAME=hdisk3
VG_NAME=myvg
NODE_NUMBER=2
NODE_ID=0xF079E8C801C11DF
CLUSTER_ID=0x6EA7B08888D811DFB918BEB25635B404
END_EVPROD_INFO
END_EVENT_INFO
```

Примечания

Данная информация была разработана для продуктов и услуг, предлагаемых на территории США.

Компания IBM может не предоставлять в других странах продукты и услуги, обсуждаемые в данном документе. Информацию о продуктах и услугах, распространяемых в вашей стране, вы можете получить в местном представительстве IBM. Ссылки на продукты, программы или услуги IBM не означают, что можно использовать только указанные продукты, программы или услуги IBM. Вместо них можно использовать любые другие функционально эквивалентные продукты, программы или услуги, не нарушающие прав IBM на интеллектуальную собственность. Однако ответственность за проверку действия любых продуктов, программ и услуг других компаний лежит на пользователе.

Компания IBM может обладать заявками на патенты или патентами на предметы обсуждения в данном документе. Обладание данным документом не предоставляет лицензии на эти патенты. Запросы на получение лицензии можно отправлять в письменном виде по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

За получением лицензий, имеющих отношение к двухбайтовому набору символов (DBCS), обращайтесь в местное отделение компании IBM по интеллектуальной собственности или направьте запрос в письменной форме по следующему адресу:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

КОМПАНИЯ IBM ПРЕДОСТАВЛЯЕТ НАСТОЯЩУЮ ПУБЛИКАЦИЮ НА УСЛОВИЯХ "КАК ЕСТЬ", БЕЗ КАКИХ-ЛИБО ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ЭТИМ, НЕЯВНЫЕ ГАРАНТИИ СОБЛЮДЕНИЯ ПРАВ, КОММЕРЧЕСКОЙ ЦЕННОСТИ И ПРИГОДНОСТИ ДЛЯ КАКОЙ-ЛИБО ЦЕЛИ. В некоторых юрисдикциях освобождение от явных и подразумеваемых гарантий запрещено в некоторых сделках, поэтому это заявление может к вам не относиться.

Эта информация может содержать технические неточности или типографические ошибки. В информацию периодически вносятся изменения, которые будут учтены во всех последующих изданиях этой книги. IBM может вносить обновления или изменения в этот документ без предварительного уведомления.

Любые ссылки на веб-сайты других компаний приведены в данной публикации исключительно для удобства пользователей и не должны рассматриваться как рекомендация этих веб-сайтов. Материалы, размещенные на этих веб-сайтах, не являются частью информации по данному продукту IBM, и ответственность за применение этих материалов лежит на пользователе.

IBM может использовать и распространять предоставленную вами информацию любым способом без каких-либо обязательств перед вами.

Лицам, обладающим лицензией на данную программу и желающим получить информацию о ней с целью: (i) настройки обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) использования информации, полученной в результате обмена, этими программами, следует обращаться по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Такая информация может быть предоставлена на определенных условиях, а в некоторых случаях - и за дополнительную плату.

Описанная в этом документе лицензионная программа и все связанные с ней лицензионные материалы предоставляются IBM в соответствии с условиями Соглашения с заказчиком IBM, Международного соглашения о лицензии на программу IBM или любого другого эквивалентного соглашения.

Данные о производительности и примеры клиентов приведены исключительно в иллюстративных целях. Фактические результаты производительности зависят от конкретных конфигураций и рабочих сред.

Информация о продуктах других компаний была получена от поставщиков этих продуктов, их опубликованных материалов или других общедоступных источников. Компания IBM не проверяла эти продукты и не может подтвердить правильность их работы, совместимость или другие заявленные характеристики продуктов других компаний. По вопросам о возможностях продуктов других компаний следует обращаться к поставщикам этих продуктов.

Заявления относительно будущих намерений IBM могут быть изменены или отозваны без дополнительного уведомления и отражают только текущие цели и задачи.

Все указанные цены IBM являются рекомендуемыми розничными ценами IBM на данный момент и могут быть изменены без предварительного уведомления. Цены дилеров могут быть другими.

Данная информация предназначена только для планирования. Она может быть изменена до выпуска описанных в данном документе продуктов.

Настоящая документация содержит примеры данных и отчетов, применяемых в повседневной деятельности компаний. Для большего сходства с реальностью примеры содержат имена людей, названия компаний, товарных знаков и продуктов. Все эти имена и названия вымышленные. Любые совпадения с реально существующими физическими или юридическими лицами совершенно случайны.

Лицензия на авторские права:

Настоящая документация содержит примеры исходного кода программ, иллюстрирующие приемы программирования в различных операционных системах. Вы имеете право копировать, изменять и распространять эти примеры программ в любой форме без уплаты вознаграждения фирме IBM в целях разработки, применения, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладных программ операционной системы, для которой предназначены эти примеры. Эти примеры не были тщательно и всесторонне протестированы. В связи с этим IBM не может гарантировать их надежность, удобство обслуживания и отсутствие ошибок. Примеры программ предоставляются "КАК ЕСТЬ", без каких-либо гарантий. IBM не несет ответственности за ущерб, который может возникнуть в результате использования эти образцов программ.

Во все копии или фрагменты этих примеров программ, а также программы созданные на их основе, следует добавлять следующее замечание об авторских правах:

© (название вашей компании) (год).

Некоторые фрагменты исходного кода получены из примеров программ фирмы IBM Corp.

© Copyright IBM Corp. _год или годы_.

Замечания о правилах работы с личными данными

Продукты IBM Software, включая решения программного обеспечения как услуг, (“Предложения программного обеспечения”) могут использовать cookie или другие технологии для сбора информации об использовании продукта в целях усовершенствования пользовательского интерфейса, для приспособления взаимодействий к конечному пользователю или для других целей. Во многих случаях Предложениями программного обеспечения собирается информация, в которой невозможно опознать персональные данные. Некоторые из наших Предложений программного обеспечения могут позволить вам собирать опознаваемую персональную информацию. Если это Предложение программного обеспечения использует cookie для сбора опознаваемой персональной информации, то специфическая информация об этом использовании cookie в предложении приведена далее.

Это Предложение программного обеспечения не использует cookie или другие технологии для сбора опознаваемой персональной информации.

Если конфигурации, развернутые для этого Предложения программного обеспечения предоставляют вам как клиенту возможность собирать опознаваемую персональную информацию о конечных пользователях посредством cookie и других технологий, вы должны самостоятельно проконсультироваться с юристом о всех законах, применимых к такому сбору данных, включая требования к уведомлению и согласию.

Более подробная информация об использовании различных технологий, включая cookie, для этих целей, приведена в Политике конфиденциальности IBM (http://www.ibm.com/privacy) и Заявлении IBM о конфиденциальности в Интернет (http://www.ibm.com/privacy/details), а также в разделах “Cookies, Web Beacons and Other Technologies” и “IBM Software Products and Software-as-a-Service Privacy Statement” на странице http://www.ibm.com/software/info/product-privacy.

Товарные знаки

IBM, эмблема IBM и ibm.com являются товарными знаками или зарегистрированными товарными знаками International Business Machines Corp. во всем мире. Названия других продуктов и услуг могут быть товарными знаками IBM и других компаний. Текущий список товарных знаков IBM опубликован на веб-странице Copyright and trademark information по следующему адресу: www.ibm.com/legal/copytrade.shtml.

Linux является зарегистрированным товарным знаком Линуса Торвальдса в США и других странах.

UNIX - зарегистрированный товарный знак The Open Group в США и других странах.

Windows является товарным знаком Microsoft Corporation в США и/или других странах.

Индекс

Спец. символы

Числа

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

A

Б

B

Г

Д

Е

Ж

З

И

К

Л

М

H

O

П

P

C

T

У

Ф

Х

Ц

Ч

Ш

Э

Я

Напечатано в Дании