

HP-UX Reference

Section 7: Device (Special) Files Section 9: General Information Index

HP-UX 11i Version 3

Volume 10 of 10



**Manufacturing Part Number : B2355-91026
E0207**

Printed in USA

© Copyright 1983-2007 Hewlett-Packard Development Company LP.

Legal Notices

The information in this document is subject to change without notice.

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

U.S. Government License

Confidential computer software. Valid license from HP required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Additional Copyright Notices

This document and the software it describes may also be protected under one or more of the following copyrights. Additional copyrights are acknowledged in some individual manpages.

- © Copyright 1979, 1980, 1983, 1985-1993 The Regents of the University of California.
- © Copyright 1980, 1984, 1986 Novell, Inc.
- © Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
- © Copyright 1986-2000 Sun Microsystems, Inc.
- © Copyright 1988 Carnegie Mellon University
- © Copyright 1989-1991 The University of Maryland
- © Copyright 1989-1993 The Open Software Foundation, Inc.
- © Copyright 1990 Motorola, Inc.
- © Copyright 1990-1992 Cornell University
- © Copyright 1991-2003 Mentat, Inc.

© Copyright 1996 Morning Star Technologies, Inc.

© Copyright 1996 Progressive Systems, Inc.

Trademark Notices

Intel and Itanium are registered trademarks of Intel Corporation in the US and other countries and are used under license.

Java is a US trademark of Sun Microsystems, Inc.

Microsoft and MS-DOS are U.S. registered trademarks of Microsoft Corporation.

OSF/Motif is a trademark of The Open Group in the US and other countries.

UNIX is a registered trademark of The Open Group.

X Window System is a trademark of The Open Group.

Preface

HP-UX is the Hewlett-Packard Company's implementation of a UNIX® operating system that is compatible with various industry standards. It is based on the System V Release 4 operating system and includes important features from the Fourth Berkeley Software Distribution.

The ten volumes of this manual contain the system reference documentation, made up of individual entries called **manpages**, named for the `man` command (see *man* (1)) that displays them on the system. The entries are also known as manual pages or reference pages.

General Introduction

For a general introduction to HP-UX and the structure and format of the manpages, please see the *introduction* (9) manpage in volume 9.

Section Introductions

The manpages are divided into sections that also have introduction (*intro*) manpages that describe the contents. These are:

<i>intro</i> (1)	<i>Section 1: User Commands</i> (A-M in volume 1; N-Z in volume 2)
<i>intro</i> (1M)	<i>Section 1M: System Administration Commands</i> (A-M in volume 3; N-Z in volume 4)
<i>intro</i> (2)	<i>Section 2: System Calls</i> (in volume 5)
<i>intro</i> (3C)	<i>Section 3: Library Functions</i> (A-M in volume 6; N-Z in volume 7)
<i>intro</i> (4)	<i>Section 4: File Formats</i> (in volume 8)
<i>intro</i> (5)	<i>Section 5: Miscellaneous Topics</i> (in volume 9)
<i>intro</i> (7)	<i>Section 7: Device (Special) Files</i> (in volume 10)
<i>intro</i> (9)	<i>Section 9: General Information</i> (in volume 10)
Index	<i>Index, All Volumes</i> (in volume 10)

Typographical Conventions

<i>audit</i> (5)	An HP-UX manpage reference. For example, <i>audit</i> is the name and 5 is the section in the <i>HP-UX Reference</i> . On the web and on the Instant Information CD, it may be a hyperlink to the manpage itself. From the HP-UX command line, you can enter “man audit” or “man 5 audit” to view the manpage. See <i>man</i> (1).
<i>Book Title</i>	The title of a book. On the web and on the Instant Information CD, it may be a hyperlink to the book itself.
Command	A command name or qualified command phrase.
ComputerOutput	Text displayed by the computer.
<i>Emphasis</i>	Text that is emphasized.
Emphasis	Text that is strongly emphasized.
ENVIRONVAR	The name of an environment variable.
[ERRORNAME]	The name of an error number, usually returned in the <code>errno</code> variable.
KeyCap	The name of a (usually) nonprinting keyboard key, such as Ctrl-X or Tab . Note that Return and Enter both refer to the same key.
<i>Replaceable</i>	The name for a value that you replace in a command or function, or information in a display that represents several possible values.
Term	The defined use of an important word or phrase.
UserInput	Commands and other text that you type.
\$	User command prompt.
#	Superuser (<code>root</code>) command prompt.

Command Syntax

Literal	A word or character that you enter literally.
<i>Replaceable</i>	A word or phrase that you replace with an appropriate value.
<i>-chars</i>	One or more grouped command options, such as <i>-ikx</i> . The <i>chars</i> are usually a string of literal characters that each represent a specific option. For example, the entry <i>-ikx</i> is equivalent to the individual options <i>-i</i> , <i>-k</i> , and <i>-x</i> . The plus character (+) is sometimes used as an option prefix.
<i>-word</i>	A single command option, such as <i>-help</i> . The <i>word</i> is a literal keyword. The difference from <i>-chars</i> is usually obvious and is clarified in an Options description. The plus character (+) and the double hyphen (--) are sometimes used as option prefixes.
[]	The bracket metacharacters enclose optional content in formats and command descriptions.
{ }	The brace metacharacters enclose required content in formats and command descriptions.
	The bar metacharacter separates alternatives in a list of choices, usually in brackets or braces.
...	The ellipsis metacharacter after a token (<i>abc...</i>) or a right bracket ([]...) or a right brace ({ }...) metacharacter indicates that the preceding element and its preceding whitespace, if any, may be repeated an arbitrary number of times.
...	Ellipsis is sometimes used to indicate omitted items in a range.

Function Synopsis and Syntax

HP-UX functions are described in a definition format rather than a usage format. The definition format includes type information that is omitted when the function call is actually included in a program.

The function syntax elements are the same as for commands, except for the options; see “Command Syntax” on page 7.

Function General Definition

The general definition form is:

```
type func ( type param [ , type param ]... );
```

For example:

```
int setuname ( const char *name , size_t namelen );
```

Function Usage

The usage form is:

```
func ( param [ , param ]... );
```

For example:

```
setuname ( name [ , namelen ]... );
```


Revision History

Part Number	Release; Date; Format; Distribution
B2355-60130	HP-UX 11i Version 3; February 2007; one volume HTML; http://docs.hp.com and Instant Information.
B2355-91017-26	HP-UX 11i Version 3; February 2007; ten volumes PDF; http://docs.hp.com , Instant Information and print.
B2355-60127	HP-UX 11i Version 1; September 2005 Update; one volume HTML; http://docs.hp.com and Instant Information.
B2355-90902-11	HP-UX 11i Version 1; September 2005 Update; ten volumes PDF; http://docs.hp.com and print.
B2355-60105	HP-UX 11i Version 2; September 2004 Update; one volume HTML; http://docs.hp.com and Instant Information.
B2355-90839-48	HP-UX 11i Version 2; September 2004 Update; ten volumes PDF; http://docs.hp.com and print.
B2355-60103	HP-UX 11i Version 2; August 2003; one volume HTML; http://docs.hp.com and Instant Information.
B2355-90779-87	HP-UX 11i Version 2; August 2003; nine volumes PDF; http://docs.hp.com and print.
B9106-90010	HP-UX 11i Version 1.6; June 2002; one volume HTML; http://docs.hp.com and Instant Information.
B9106-90007	HP-UX 11i Version 1.5; June 2001; seven volumes HTML; http://docs.hp.com and Instant Information.
B2355-90688	HP-UX 11i Version 1; December 2000; nine volumes.
B2355-90166	HP-UX 11.0; October 1997; five volumes.
B2355-90128	HP-UX 10.X; July 1996; five volumes; online only.
B2355-90052	HP-UX 10.0; July 1995; four volumes.

**Volume Ten
Table of Contents**

Section 7

Section 9

Index

Volume Ten
Table of Contents

Section 7
Section 9

Index

Table of Contents Volume Ten

Section 7: Device (Special) Files

Entry Name(Section): name	Description
intro(7): intro	introduction to device special files
arp(7P): arp	address resolution protocol
autochanger(7): schgr, eschgr	SCSI interfaces for medium changer device
blmode(7): blmode	terminal block mode interface
cent(7): cent	Centronics-compatible interface
clone(7)	open a major and minor device pair on a STREAMS driver
console(7): console, systty, syscon	system console interface
ddfa(7): ddfa	Data Communications and Terminal Controller Device File Access software
diag0(7): diag0	diagnostic interface to I/O subsystem
diag1(7): diag1	diagnostic interface to I/O subsystem
diag2(7): diag2	diagnostic interface
disk(7): disk	direct disk access
dlpi(7): dlpi	data link provider interface
eschgr: SCSI interfaces for medium changer device	see autochanger(7)
framebuf(7): framebuf	information for raster frame-buffer devices
gang_sched(7): gang_sched	Gang Scheduler
hil(7): hil	HP-HIL device driver
hilkbd(7): hilkbd	HP-HIL mapped keyboard driver
inet(7F): inet	Internet protocol family
iomap(7): iomap	physical address mapping
IP(7P): IP	Internet Protocol
ip6: Internet Protocol Version 6	see IPv6(7P)
IPv6(7P): IPv6, ipv6, ip6	Internet Protocol Version 6
ipv6: Internet Protocol Version 6	see IPv6(7P)
kmem(7): kmem	perform I/O on kernel memory, based on symbol name
lan(7): lan	network I/O card access information
ldterm(7): ldterm	STREAMS terminal line discipline module
lp(7): lp	line printer
lvm(7): lvm	Logical Volume Manager (LVM)
mem(7): mem	main memory image file
modem(7): modem	asynchronous serial modem line control
mt(7): mt	magnetic tape interface for stape and tape2
ndp(7P): ndp	Neighbor Discovery Protocol, NDP
nfs(7): nfs, NFS	network file system
null(7): null	null file
pckt(7): pckt	Packet Mode module for STREAMS pty
poll(7): poll	monitor I/O conditions on multiple file descriptors
ps2(7): ps2, ps2kbd, ps2mouse	PS/2 keyboard and mouse device driver and files
ptem(7): ptem	STREAMS pty (pseudo-terminal) Emulation module
ptm(7): ptm	STREAMS master pty (pseudo-terminal) driver
pts(7): pts	STREAMS slave pty driver
pty(7): pty	pseudo-terminal driver
random(7): random, urandom, rng	strong random number generator
rng: strong random number generator	see random(7)
route(7P): route	kernel packet forwarding database
routing(7): routing	system support for local network packet routing
sad(7)	STREAMS administrative driver
schgr: SCSI interfaces for medium changer device	see autochanger(7)
schgr: SCSI media changer device drivers	see autochanger(7)
scsi(7): scsi	Small Computer System Interface (SCSI) device drivers
scsi_ctl(7): scsi_ctl	SCSI pass-through driver (esctl/sctl)
scsi_disk(7): scsi_disk	SCSI direct access device driver
scsi_tape(7): scsi_tape	SCSI sequential access device driver
scsimgr_eschgr(7): scsimgr_eschgr	SCSI class driver eschgr plug-in for scsimgr
scsimgr_esdisk(7): scsimgr_esdisk	SCSI class driver esdisk plug-in for scsimgr

Table of Contents

Volume Ten

Entry Name(Section): name	Description
scsimgr_estape(7): scsimgr_estape	SCSI class driver estape plug-in for scsimgr
sioc_io(7): sioc_io	SCSI pass-through interface
slp_syntax(7): slp_syntax	SLP Service URL Syntax
socket(7): socket	Interprocess communications
ssrfc: SCSI media changer device drivers	see autochanger(7)
streamio(7)	STREAMS ioctl commands
strlog(7): strlog	STREAMS log driver
stty: terminal interface for Version 6/PWB compatibility	see sttyv6(7)
sttyv6(7): stty	terminal interface for Version 6/PWB compatibility
syscon: system console interface	see console(7)
sysstty: system console interface	see console(7)
TCP(7P): TCP	Internet Transmission Control Protocol
telm: STREAMS Telnet master driver	see tels(7)
tels(7): tels, telm	STREAMS slave and master drivers
termio(7): termio, termios	general terminal interface
termios: general terminal interface	see termio(7)
termiox(7): termiox	extended general terminal interface
timod(7)	STREAMS module for reads and writes by Transport Interface users
tirdwr(7)	STREAMS module for reads and writes by Transport Interface users
tty(7): tty	controlling terminal interface
UDP(7P): udp	Internet user datagram protocol
UNIX(7P): UNIX	local communication domain protocol
urandom: strong random number generator	see random(7)
VLAN(7): VLAN	virtual local area network
xopen_networking(7): xopen_networking	X/Open Networking Interfaces
zero(7): zero	/dev/zero special file

Section 9: General Information

Entry Name(Section): name	Description
intro(9): intro	introduction to HP-UX general information section
glossary(9): glossary	description of common HP-UX terms
introduction(9): introduction	HP-UX operating system and <i>HP-UX Reference</i>

Index: All Volumes

Section 7

Device (Special) Files

Section 7

Device (Special) Files

NAME

intro - introduction to device special files

DESCRIPTION

This section describes the device special files (DSFs) and hardware paths used to access HP peripherals and device drivers. The names of the entries are generally derived from the type of device being described (disk, tape, terminal, and so on.), not the names of the device special files or device drivers themselves. Characteristics of both the hardware device and the corresponding HP-UX device driver are discussed where applicable.

Device Types

Devices can be classified in two device access modes, *raw* and *block*. A raw or character-mode device, such as a line printer, transfers data in an unbuffered stream and uses a character device special file.

A block-mode device, as the name implies, transfers data in blocks by means of the system's normal buffering mechanism. Block devices use block device special files and may have a character device interface too.

Device File Naming Convention

A device special file name becomes associated with a device when the file is created, either automatically by the special file daemon **sfd**, or explicitly with the **insf**, **mknod**, or **mksf** command. When creating device special files, it is recommended that the following standard naming convention be used:

/dev/subdir/class#[options]

subdir An optional subdirectory for the device class (for example, **rdisk** for raw device special files for disks, **disk** for block device special files for disks, **rtape** for raw tape devices).

class The class of device, such as **tape**, **disk**, or **lan**.

The instance number assigned by the operating system to the device. Each class of device has its own set of instance numbers, so each combination of class and instance number refers to exactly one device.

options Further qualifiers, such as disk partition (**p#**), tape density selection for a tape device, or surface specification for magneto-optical media.

Naming conventions for each type of device are described in their respective manpage entries.

Legacy mass storage device special files have a different naming convention that encodes the hardware path; this is described in the *Device File Types (Mass Storage Devices)* section.

Hardware Paths

Hardware path information, as well as class names and instance numbers, can be derived from **ioscan** output; see **ioscan(1M)**. There are three different types of paths to a device: *legacy hardware path*, *lunpath hardware path*, and *LUN hardware path*. All three are numeric strings of hardware components, notated sequentially from the system bus address to the device address. Each number typically represents the location of a hardware component on the path to the device.

The *legacy hardware path* is composed of a series of bus-nexus addresses separated by slash (/) characters, leading to a host bus adapter (HBA). Beneath the HBA, additional address elements are separated by period (.) characters. All the elements are represented in decimal. This is the format printed by default by the **ioscan** command for most devices. An example of a legacy hardware path is **0/0/2/0.1.7.0**.

The *lunpath hardware path* is used for mass storage devices, also known as logical units (LUNs). It is identical in format to a legacy hardware path, up to the HBA. Beneath the HBA, additional elements are printed in hexadecimal. The leading elements representing a transport-dependent target address, and the final element is a LUN address, which is a 64-bit representation of the LUN identifier reported by the target. This format is printed by the **ioscan** command when the **-N** option is specified. The string **0/2/1/0.0x50001fe1500170ac.0x4017000000000000** is an example of a lunpath hardware path.

Note that the address elements beneath the HBA may not correspond to physical hardware addresses; instead, the lunpath hardware path should be considered a *handle*, not a physical path to the device.

The *LUN hardware path* is a virtualized path that can represent multiple hardware paths to a single mass storage device. Instead of a series of bus-nexus addresses leading to the HBA, there is a virtual bus-nexus (known as the *virtual root node*) with an address of 64000. Addressing beneath that virtual root node

consists of a virtual bus address and a virtual LUN identifier, delimited by slash (/) characters. The string **64000/0xfa00/0x22** is an example of a LUN hardware path.

As a virtualized path, the LUN hardware path is only a handle to the LUN, and does not represent the LUN's physical location; rather, it is linked to the LUN's World Wide Identifier (WWID). Thus, it remains the same if new physical paths to the device are added, if existing physical paths are removed, or if any of the physical paths changes. This LUN binding persists across reboots, but it is not guaranteed to persist across installations — that is, reinstalling a system or installing an identically configured system may create a different set of LUN hardware paths.

Device File Types (Mass Storage Devices)

Mass storage devices, such as disk devices and tape devices, have two types of device files, *persistent* device special files and *legacy* device special files. Both can be used to access the mass storage device independently, and can coexist on the same system.

A *persistent* device special file is associated with a LUN hardware path, and thus transparently supports agile addressing and multipathing. In other words, a persistent device special file is unchanged if the LUN is moved from one HBA to another, moved from one switch/hub port to another, presented via a different target port to the host, or configured with multiple hardware paths. Like the LUN hardware path, the binding of device special file to device persists across reboots, but is not guaranteed to persist across installations. The device special file name follows the standard naming convention above, and the minor number contains no hardware path information.

A *legacy* device special file is locked to a particular physical hardware path, and does not support agile addressing. Such a device special file contains hardware path information such as SCSI bus, target, and LUN in the device file name and minor number. Specifically, the *class* and *instance* portions of the device special file name indicate hardware path information and are in the format **c#t#d#** as follows:

- c#** The instance number assigned by the operating system to the interface card, in decimal. It is a decimal number with a range of 0 to 255. There is no direct correlation between instance number and physical slot number.
- t#** The target address on a remote bus (for example, SCSI address). It is a decimal number with a typical range of 0 to 15.
- d#** The device unit number at the target address (for example, the LUN in a SCSI device). It is a decimal number with a typical range of 0 to 7.

Note that the legacy naming convention supports a maximum of 256 external buses and a maximum of 32768 LUNs. Systems with mass storage devices beyond those limits will be unable to address them using legacy naming conventions.

Legacy device special files are deprecated, and their support will be removed in a future release of HP-UX.

Viewing Mass Storage

With the advent of persistent and legacy device special files, commands dealing with mass storage can choose between two *views* of the I/O system. A command presenting the *legacy* view uses legacy device special files and legacy hardware paths. The *agile* view uses persistent device special files, lunpath hardware paths, and LUN hardware paths.

Depending on the command, both views may be presented, or the choice of view may be controlled by a command option or an environment variable. For example, the **ioscan** command shows the legacy view by default, and switches to the agile view if the **-N** option is specified.

EXAMPLES

Example 1

The following is an example of a persistent device special file name:

```
/dev/disk/disk3
```

where **disk** indicates block disk access and **disk3** indicates device class *disk* and instance number 3. The absence of **p#** indicates access to the entire disk; see *disk(7)* for details.

Example 2

The following is an example of a legacy disk device special file name:

```
/dev/dsk/c0t6d0s2
```

where **dsk** indicates block disk access and **c0t6d0** indicates logical disk access at interface card instance 0, target address 6, and unit 0. The **s2** indicates access to section 2 of the disk.

Example 3

The following is an example of a persistent tape device special file name:

```
/dev/rtape/tape4QIC150
```

where **rtape** indicates raw magnetic tape, **tape4** indicates tape device instance number 4, and **QIC150** identifies the tape format as QIC150; see *mt(7)* for details.

WARNINGS

The support of legacy device special files is deprecated and will be removed in a future release of HP-UX.

SEE ALSO

insf(1M), *ioscan(1M)*, *lssf(1M)*, *mksf(1M)*, *mknod(1M)*, *hier(5)*, *introduction(9)*.

System Administration's Guide at <http://docs.hp.com>.

The Next Generation Mass Storage Stack whitepaper at:

<http://docs.hp.com/en/netsys.html#Storage%20Area%20Management>.

NAME

arp - Address Resolution Protocol

DESCRIPTION

ARP is a protocol used to dynamically map between DARPA Internet and hardware station addresses. It is used by all LAN drivers.

ARP caches Internet-to-hardware station address mappings. When an interface requests a mapping for an address not in the cache, ARP queues the message that requires the mapping, and broadcasts a message on the associated network requesting the address mapping if the **ether** encapsulation method has been enabled for the interface. If a response is provided, the new mapping is cached and any pending message is transmitted. ARP queues at most one packet while waiting for a mapping request to be responded to; only the most recently “transmitted” packet is kept.

To facilitate communications with systems that do not use ARP, **ioctl** calls are provided to enter and delete entries in the Internet-to-hardware station address tables.

Application Usage:

```
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/if_ether.h>
struct arpreq arpreq;

ioctl(s, SIOCSARP, (caddr_t)&arpreq);
ioctl(s, SIOCGARP, (caddr_t)&arpreq);
ioctl(s, SIOCDDARP, (caddr_t)&arpreq);
```

Each **ioctl** call takes the same structure as an argument. **SIOCSARP** sets an ARP entry, **SIOCGARP** gets an ARP entry, and **SIOCDDARP** deletes an ARP entry. These **ioctl** calls can be applied to any socket descriptor *s*, but only by the super-user. The **arpreq** structure contains:

```
/*
 * ARP ioctl request
 */
struct arpreq {
    int32_t ifindex;
    int32_t arp_flags;           /* flags */
    int32_t arp_hw_addr_len;    /* hardware address length */
    struct sockaddr arp_pa;     /* protocol address */
    struct sockaddr arp_ha;     /* hardware address */
    u_char  arp_pad[242];      /* buffer for link specific info. */
};
/* arp_flags field values */
#define ATF_COM          0x02    /* ARP on ether */
#define ATF_PERM        0x04    /* permanent entry */
#define ATF_PUBL        0x08    /* publish entry */
#define ATF_SNAPFDDI    0x200   /* SNAP - FDDI */
#define ATF_SNAP8025    0x400   /* SNAP - 8025 */
#define ATF_IEEE8025    0x800   /* IEEE - 8025 */
#define ATF_FCSNAP      0x4000  /* Fibre Channel SNAP */
```

The address family for the *arp_pa* **sockaddr** must be **AF_INET**; for the *arp_ha* **sockaddr** it must be **AF_UNSPEC**. The only flag bits that can be written are **ATF_PERM**, and **ATF_PUBL**. Fibre Channel hosts only support the **ATF_PERM** flag. **ATF_PERM** causes the entry to be permanent. **ATF_PUBL** specifies that the ARP code should respond to ARP requests for the indicated host coming from other machines. This allows a host to act as an *ARP server*, which may be useful in convincing an ARP-only machine to talk to a non-ARP machine.

ARP watches passively for hosts impersonating the local host (i.e., a host that responds to an ARP mapping request for the local host’s address).

DIAGNOSTICS

duplicate IP address!! sent from ethernet address: %x:%x:%x:%x:%x:%x.

This message printed on the console screen means that ARP has discovered another host on the local network that responds to mapping requests for its own Internet address.

WARNINGS

To enable the **ether** encapsulation method, use the **ifconfig** command (see *ifconfig(1M)*).

AUTHOR

ARP was developed by the University of California, Berkeley.

SEE ALSO

ifconfig(1M), *inet(3N)*, *lan(7)*, *arp(1M)*.

An Ethernet Address Resolution Protocol, RFC826, Dave Plummer, Network Information Center, SRI.


a

NAME

autochanger: schgr, eschgr - SCSI interfaces for medium changer device

DESCRIPTION

An autochanger is a SCSI mass storage device, consisting of a mechanical changer device, one or more data transfer devices (such as optical disk drives), and media (such as optical disks) for data storage. The mechanical changer moves media between storage and usage locations within the autochanger.

Two medium changer drivers (**schgr** or **eschgr**) provide access to the medium changer device; **eschgr** is the current preferred method of access and **schgr** is provided for legacy compatibility. The mechanical changer device can be accessed via these drivers directly to move media within the autochanger.

The **schgr** and **eschgr** medium changer device drivers follow the SCSI specification for medium changer devices to provide a generic medium changer interface, making it feasible to construct an application level driver for any mechanical changer, jukebox, library, or autochanger device (MO, tape, CD-ROM).

Device Naming Convention

The device naming convention for the autochanger driver enables accessing the changer device.

Legacy character device file names reside in **/dev/rac**. Within this directory, names are derived from the **c#t#d#** device naming convention (explained in *intro(7)*). Unique legacy device names are determined by the card instance, target address of the SCSI changer device and LUN of the SCSI changer device.

Persistent device file names have the form **/dev/rchgr/autochx** for character devices. The card instance, target address and LUN are no longer encoded in the persistent device file name itself (see *intro(7)*).

Major and Minor Number Descriptions

The following shows the bit assignments (**dev_t** format) used by the **schgr** changer driver to access the changer device using legacy device files:

0-7	8-15	16-19	20-22	
MAJOR	INSTANCE	TARGET	LUN	

MAJOR is the major number of the appropriate driver, INSTANCE is the card instance of the SCSI interface to which the changer device is attached, TARGET is the SCSI target address of the changer device, LUN is the SCSI LUN of the changer device.

All fields in the device number are specified in hexadecimal notation. Note that there is no support for hard partitions (sections) in this minor number. If desired, partitioning can be achieved via LVM soft-partitioning schemes.

Note: The major numbers used by the changer drivers are dynamically assigned starting with release HP-UX 11i v3.

Following is a long listing showing the major and minor numbers associated with the device special file name of the changer:

schgr:

```
crw-rw-rw- 1 root sys 231 0x015000 Apr 22 10:22 /dev/rac/c1t5d0
```

SCSI MEDIUM CHANGER DEVICE DRIVER

The SCSI medium changer device driver performs moves between different media locations within an autochanger. Each potential media location has a specific element address and is one of the following element types:

<i>storage</i>	A location to hold a unit of media not currently in use. Typically most media will be located in this type of element.
<i>import/export</i>	A location for inserting and removing media from the device. Movement of a unit of media to this type of location is in effect an eject operation. Movement of a unit of media from this type of location is a load operation.
<i>data transfer</i>	A location for accessing media data. This is generally the location of a device that reads and/or writes data on the media being handled by the media changer device. Movement to this type of location is a physical-media-mount operation. Movement from this type of location is a physical-media-unmount operation.

media transport A location for media movement. Media is generally temporarily located in this type of element only during actual media movement.

Changer Control Requests

The following ioctl functions and structure definitions are included from `<sys/scsi.h>`:

```
#define SIOC_INIT_ELEM_STAT      _IO('S', 51)
#define SIOC_ELEMENT_ADDRESSES  _IOW('S', 52, struct element_addresses)
#define SIOC_ELEMENT_STATUS     _IOWR('S', 53, struct element_status)
#define SIOC_RESERVE            _IOW('S', 54, struct reservation_parms)
#define SIOC_RELEASE            _IOW('S', 55, struct reservation_parms)
#define SIOC_MOVE_MEDIUM        _IOW('S', 56, struct move_medium_parms)
#define SIOC_EXCHANGE_MEDIUM    _IOW('S', 57, struct exchange_medium_parms)

/* structure for SIOC_ELEMENT_ADDRESSES ioctl */
struct element_addresses {
    unsigned short  first_transport;
    unsigned short  num_transports;
    unsigned short  first_storage;
    unsigned short  num_storages;
    unsigned short  first_import_export;
    unsigned short  num_import_exports;
    unsigned short  first_data_transfer;
    unsigned short  num_data_transfers;
};

/* structure for SIOC_ELEMENT_STATUS ioctl */
struct element_status {
    unsigned short element;          /* element address */

    unsigned int   resv1:2;
    unsigned int   import_enable:1; /* allows media insertion (load) */
    unsigned int   export_enable:1; /* allows media removal (eject) */
    unsigned int   access:1;        /* transport element accessible */
    unsigned int   except:1;        /* is in an abnormal state */
    unsigned int   operatr:1;       /* medium positioned by operator */
    unsigned int   full:1;          /* holds a a unit of media */

    unsigned char  resv2;
    unsigned char  sense_code;      /* info. about abnormal state */
    unsigned char  sense_qualifier; /* info. about abnormal state */

    unsigned int   not_bus:1;       /* transfer device SCSI bus differs */
    unsigned int   resv3:1;
    unsigned int   id_valid:1;      /* bus_address is valid */
    unsigned int   lu_valid:1;      /* lun is valid */
    unsigned int   sublun_valid:1;  /* sub_lun is valid */
    unsigned int   lun:3;           /* transfer device SCSI LUN */

    unsigned char  bus_address;     /* transfer device SCSI address */
    unsigned char  sub_lun;         /* sub-logical unit number */

    unsigned int   source_valid:1;  /* source_element is valid */
    unsigned int   invert:1;        /* media in element was inverted */
    unsigned int   resv4:6;

    unsigned short source_element;  /* last storage medium location */
    char           pri_vol_tag[36];  /* volume tag (device optional) */
    char           alt_vol_tag[36];  /* volume tag (device optional) */
    unsigned char  misc_bytes[168]; /* device specific */
};

/* structure for SIOC_RESERVE and SIOC_RELEASE ioctls */
struct reservation_parms {
```

a

```

        unsigned short  element;
        unsigned char   identification;
        unsigned char   all_elements;
    };

    /* structure for SIOC_MOVE_MEDIUM ioctl */
    struct move_medium_parms {
        unsigned short  transport;
        unsigned short  source;
        unsigned short  destination;
        unsigned char   invert;
    };

    /* structure for SIOC_EXCHANGE_MEDIUM ioctl */
    struct exchange_medium_parms {
        unsigned short  transport;
        unsigned short  source;
        unsigned short  first_destination;
        unsigned short  second_destination;
        unsigned char   invert_first;
        unsigned char   invert_second;
    };

```

SIOC_INIT_ELEM_STAT

Cause the media changer device to take inventory. As a result, the media changer device determines the status of each and every element address, including the presence or absence of a unit of media. This is a mechanical operation which can take time. This function only necessary in the event of a severe error of the media changer.

SIOC_ELEMENT_ADDRESSES

Determine the element addresses supported by a media changer device. The first valid element address and the number of elements is indicated for each element type. These element addresses may be used as source and destination location arguments.

SIOC_ELEMENT_STATUS

Determine the status of an element. The element address for which status information is requested is specified via the **element** field. The resulting status data indicates the presence or absence of a unit of media in that element address as well as other information about the element address.

SIOC_RESERVE and SIOC_RELEASE

Control access to element addresses. Depending on the device, reservations may limit operator control of those element addresses in the media changer device. Specific element addresses can be reserved to handle interlocking between multiple requesters if each requester has a unique reservation identification. The value zero in the **all_elements** field specifies that a single element address should be reserved or released. An element address reserved in this manner can not be reserved by another single element address reservation using a different reservation identification. The **reservation** field specifies the reservation identification. The **element** field specifies the element address to be reserved.

The value "1" in the **all_elements** field indicates that all element addresses should be reserved. The **reservation** and **element** fields should contain the value zero since these fields are not meaningful when reserving all element addresses. Reserving all element addresses is primarily useful for limiting operator control.

SIOC_MOVE_MEDIUM and SIOC_EXCHANGE_MEDIUM

Reposition unit(s) of media. Depending on the source and destination element types, this may result in a media load, eject, or simple repositioning. Media can be "flipped" using values of "1" in the **invert**, **invert_first**, or **invert_second** fields. The **SIOC_EXCHANGE_MEDIUM** ioctl repositions two different units of media. One unit of media is moved from the element specified by the **source** field to the element specified by the **first_destination** field. A second unit of media is moved from the element specified by the **first_destination** field to the element specified by the **second_destination** field. In an autochanger with multiple changer mechanisms, or a media staging area, an exchange occurs if the **source** and **second_destination** fields are the same.

DEFAULT CONFIGURATIONS

By default, **schgr** and **eschgr** are not included in the system configuration (**/stand/system**) file.

EXAMPLES

The following example uses the **SIOC_ELEMENT_ADDRESSES** and **SIOC_ELEMENT_STATUS ioctl** functions to get bus address information about the drives in an autochanger device:

```
int                last_drive_el;
struct element_addresses  el_addrs;
struct element_status    el_stat; drive[1024];
int fd = -1, error = 0, i = 0;

fd = open("/dev/rchgr/autoch0",O_RDWR);
if ((error = ioctl(fd, SIOC_ELEMENT_ADDRESSES, &el_addrs)) != 0) {
    perror("ioctl: SIOC_ELEMENT_ADDRESSES");
    return -1;
} else {
    last_drive_el = el_addrs.first_data_transfer
        + el_addrs.num_data_transfers - 1;
    for (i = el_addrs.first_data_transfer; i <= last_drive_el; i++) {
        el_stat.element = i;
        if ((error = ioctl(fd, SIOC_ELEMENT_STATUS, &el_stat)) != 0) {
            perror("ioctl: SIOC_ELEMENT_ADDRESSES");
            return -1;
        } else {
            /*
             * You may wish to also check some of the other fields
             * in the el_stat structure to verify that the data is
             * valid.  Fields: el_stat.access (ac accessible),
             * el_stat.except (exception).
             */
            if (! el_stat.not_bus && el_stat.id_valid) {
                drive[i].bus_address = el_stat.bus_address;
                if (! el_stat.lu_valid) {
                    drive[i].lun = 0;
                } else {
                    drive[i].lun = el_stat.lun;
                }
            }
        }
    }
}
}
```

WARNINGS

Some non-HP media changer devices do not support the **SIOC_INIT_ELEM_STAT** and **SIOC_ELEMENT_STATUS ioctls**.

Some older media changer devices do not support the **SIOC_EXCHANGE_MEDIUM ioctl**. For these devices, multiple **SIOC_MOVE_MEDIUM ioctl** operations may be used to accomplish the same results, provided a suitable temporary element address may be found.

SEE ALSO

insf(1M), mknod(1M), scsictl(1M), ioctl(2), scsi(7), scsi_ctl(7), intro(7).

NAME

blmode - terminal block mode interface

DESCRIPTION

This terminal interface adds functionality to the current *termio(7)* functionality to allow for efficient emulation of MPE terminal driver functionality. Most importantly, it adds the necessary functionality to support block mode transfers with HP terminals. The block mode interface only affects input processing and does not affect write requests. Write requests are always processed as described in *termio(7)*. In character mode the terminal sends each character to the system as it is typed. However, in block mode data is buffered and possibly edited locally in the terminal memory as it is typed, then sent as a block of data when the **Enter** key is pressed on the terminal. During block mode data transmissions, the incoming data is not echoed and no special character processing is performed, other than recognizing a data block terminator character. For subsequent character mode transmissions, the existing *termio* state continues to determine echo and character processing.

There are two parts of the block mode protocol. The first part is the block mode handshake, which works as follows:

- At the beginning of a read, a *trigger* character is sent to the terminal to notify it that the system is requesting a block of data. (The *trigger* character, if defined, is sent at the beginning of all reads, whether character or block. The *trigger* character must be defined for block mode reads.)
- After receiving the *trigger* character, the terminal waits until the user has typed data into the terminal's memory and pressed the terminal **Enter** key. The terminal then sends an *alert* character to the system to notify it that the terminal has a block of data to send.
- The system may then send user-definable cursor positioning or other data sequences to the terminal. When that is done, the system sends another *trigger* character to the terminal, repeating the cycle.

The second part of the block mode protocol is the block mode transmission. During this transmission of data, the incoming data is not echoed and no special character processing is performed, other than recognizing the data block termination character. It is possible to bypass the block mode handshake and have the block mode transmission occur after the first *trigger* character is sent.

To prevent data loss, XON/XOFF flow control should be used between the system and the terminal. The IXOFF bit should be set and the terminal strapped appropriately. If flow control is not used, it is possible for incoming data to overflow and be lost. (Note: some older terminals do not deal correctly with this flow control.)

It is possible to intermix both character mode and block mode data transmissions. If block mode transmissions are enabled, all transfers are handled as block mode transfers. When block mode transmissions are not enabled, character mode transmissions are processed as described in *termio(7)*. If block mode transmissions are not enabled, but an *alert* character is received anywhere in the input data, the transmission mode is switched to block mode automatically for a single transmission.

Read requests that receive data from block mode transmissions will not be returned until the transmission is complete; i.e., the terminal has transmitted all characters. If the read is satisfied by byte count or if a data transmission error occurs, any subsequent data will be discarded. The read waits until completion of the data transmission before returning.

The data block terminator character is included in the data returned to the user, and is included in the byte count. If the number of bytes transferred by the terminal in a block mode transfer exceeds the number of bytes requested by the user, the read returns the requested number of bytes, and the remaining bytes are discarded. The user can determine if data was discarded by checking the last character of the returned data. If the last character is not the terminator character, more data was received than was requested, and data was discarded.

If desired, the application program can provide its own handshake mechanism in response to the *alert* character by selecting the OWNTERM mode. With this mode selected, the driver completes a read request when the *alert* character is received. The second *trigger* is sent by the driver when the application issues the next read.

Several special characters (both input and output) are used with block mode. These characters and the normal values used for block mode are described below. The initial value for these characters is 0377, which causes them to be disabled.

CBTRIG1C	(DC1) is the initial <i>trigger</i> character sent to the terminal at the beginning of a read request.
CBTRIG2C	(DC1) is the secondary <i>trigger</i> character sent to the terminal after the <i>alert</i> character has been received.
CBALERTC	(DC2) is the <i>alert</i> character sent by the terminal in response to the first <i>trigger</i> character. It signifies that the terminal is ready to send the data block. The <i>alert</i> character can be escaped by preceding it with a backslash (\).
CBTERMC	(RS) is sent by the terminal after the block mode transfer is complete. It signifies the end of the data block to the computer.

The two *ioctl(2)* requests that apply to block mode use the **blmodeio** structure, which defined in `<blmodeio.h>`, and includes the following members:

```

unsigned long  cb_flags;      /* Modes */
unsigned char  cb_trig1c;    /* First trigger */
unsigned char  cb_trig2c;    /* Second trigger */
unsigned char  cb_alertc;    /* Alert character */
unsigned char  cb_termc;     /* Terminating char */
unsigned char  cb_replen;    /* cb_reply length */
char          cb_reply[];    /* optional reply */

```

The *cb_flags* member controls the basic block mode protocol:

```

CB_BMTRANS    0000001    Enable mandatory block mode transmission.
CB_OWNTERM    0000002    Enable user control of handshake.

```

The **CB_BMTRANS** bit is only effective when the **ICANON** flag in *termio(7)* is set. If **ICANON** is clear, all transfers are done in raw mode, regardless of the **CB_BMTRANS** bit. If **CB_BMTRANS** is not set, input processing is performed as described in *termio(7)*. During this time, if the *alert* character is defined and is detected anywhere in the input stream, the input buffer is flushed and block-mode handshake is invoked. The system then sends the *cb_trig2c* character to the terminal, and a block mode transfer follows. The *alert* character can be escaped by preceding it with a backslash (\).

If **CB_BMTRANS** is set, then all transmissions are processed as block mode transmissions. Block mode handshake is not required and data read is processed as block mode transfer data. Block mode handshake can still be invoked by receipt of an *alert* character as the first character received. Reads issued while the **CB_BMTRANS** bit is set cause any existing input buffer data to be flushed.

If **CB_OWNTERM** is set, reads are terminated upon receipt of a non-escaped *alert* character. No input buffer flushing is performed and the *alert* character is returned in the data read. This allows application code to perform its own block-mode handshaking. If the bit is clear, an *alert* character causes normal block mode handshaking to be used.

The initial **cb_flags** value is all-bits-cleared.

The **cb_trig1c** character is the initial *trigger* character sent to the terminal at the beginning of a read request. The initial value is undefined (0377); i.e., no *trigger* character is sent.

The **cb_trig2c** character is the secondary *trigger* character sent to the terminal after the *alert* character has been received. The initial value is undefined (0377).

The **cb_alertc** character is the *alert* character sent by the terminal in response to the first *trigger* character sent by the computer. It signifies that the terminal is ready to transmit data. The initial value is undefined (0377).

The **cb_termc** character is sent by the terminal after the block mode transfer has completed. It signifies the end of the data block to the computer. The initial value is undefined (0377).

The **cb_replen** member specifies the length in bytes of the **cb_reply** array. The maximum length of the **cb_reply** array is **NBREPLY** bytes. If set to zero, the *cb_reply* string is not used. It is initially set to zero.

The *cb_reply* array contains a string to be sent out after receipt of the *alert* character but before the second *trigger* character is sent by the computer. Any character can be included in the reply string. The number of characters sent is specified by **cb_replen**. The maximum length of the **cb_reply** array is **NBREPLY** bytes. The initial value of all characters in the **cb_reply** array is null.

On systems that support process group control, *ioctl* requests are restricted from use by background processes, unless otherwise noted for a specific request. An attempt to issue an *ioctl* request from a background process causes the process to block and may cause a **SIGTTOU** signal to be sent to the process group.

The primary *ioctl(2)* calls have the form:

```
int ioctl(int fildes, int request, struct blmodeio *arg);
```

Requests using this form include:

- CBGETA** Get the parameters associated with the block mode interface and store them in the *blmodeio* structure referenced by *arg*. This request is allowed from a background process. However, the information may be subsequently changed by a foreground process.
- CBSETA** Set the parameters associated with the block mode interface from the *blmodeio* structure referenced by *arg*. The change is immediate.

RETURN VALUE

Refer to *read(2)*, *write(2)*, and *ioctl(2)*.

ERRORS

If an error value is returned during a read, it is possible for the user's buffer to be altered. In this case, the data in the user's buffer should be ignored because it is incomplete.

The global variable *errno* will be set to indicate the following error, in addition to those errors described on *read(2)*, *write(2)*, and *ioctl(2)*:

- [EIO] A read error occurred during the transmission of the block mode data block.

WARNINGS

The [EIO] error that is returned for read errors can be caused by many events. The read returns [EIO] for transmission, framing, parity, break, and overrun errors, or if the internal timer expires. The internal timer starts when the second *trigger* character is sent by the computer, and ends when the terminating character is received by the computer. The length of this timer is determined by the number of bytes requested in the read and the current baud rate, plus an additional ten seconds.

AUTHOR

The *blmode* driver was developed by HP.

SEE ALSO

termio(7).

(Workstations Only)

NAME

cent - Centronics-compatible interface

DESCRIPTION

cent is a simple, widely used communication protocol most commonly associated with printers, plotters and scanners. It is an eight-bit parallel data interface with additional control signals from the host computer, and status signals from the peripheral.

The **cent** interface driver does no character processing; that is, it does not interpret the data being transferred between computer and peripheral. Therefore, all bytes sent to or received from a device are handled without alteration. The **cent** interface driver always operates in **raw mode**; therefore, any desired data interpretation must be performed by a user program (such as the “lp” spooler in conjunction with an appropriate model file). The **cent** driver supports six different handshake modes for data transfer. The last four bits of the minor number of the device special file specify the mode used. The format of the device minor number is:

0xII000A

where each letter after the “0x” prefix represents a single hexadecimal digit, as follows:

II Specifies the instance number of the centronic interface.

000 Always zero.

A Specifies the handshake mode. The handshake modes are:

- mode 1 Automatic handshaking using both ACK and BUSY.
Minor number format: **0xII0001**.
- mode 2 Automatic handshaking using only BUSY.
Minor number format: **0xII0002**.
- mode 3 Bidirectional read/write used for ScanJet.
Minor number format: **0xII0003**.
- mode 4 Stream mode. Data is essentially transmitted to the peripheral without any handshaking protocol.
Minor number format: **0xII0004**.
- mode 5 Pulsed mode using both ACK and BUSY for automatic handshaking. Similar to mode 1 except that the data strobe line, **nSTROBE**, is pulsed for a fixed amount of time by the sender, then released.
Minor number format: **0xII0005**.
- mode 6 Pulsed mode, using only BUSY for automatic handshaking. Similar to mode 1 except that the data strobe line, **nSTROBE**, is pulsed for a fixed amount of time by the sender, then released.
Minor number format: **0xII0006**.

Modes 1 and 2 support most HP *Jet series printers (LaserJet, DeskJet, QuietJet, etc.).

AUTHOR

cent was developed by HP.

SEE ALSO

lp(1), ioctl(2), intro(7), lp(7).

NAME

clone - opens a major and minor device pair on a STREAMS driver

DESCRIPTION

The **clone** driver is a "pass through" device driver that allows other drivers to select unique minor device numbers on each **open()**. In effect, the driver passes an open operation through to the other driver. This mechanism allows for multiple instantiations of a driver, each with a different minor number, through a single device file.

When the **clone** driver is opened, it is passed a major and minor device number by the operating system. The major number is the **clone** driver's major number (72), and the minor number is the major number of the driver the user wishes to clone (referred to here as the target driver). The **clone** driver calls the open routine of the target driver with the **CLONEOPEN** flag which specifies a clone open. The target driver's open routine allocates an unused minor number. The target driver must use **makedev** to make a new device number for the newly created device, and must set ***devp** to the new device number returned by **makedev**. The new device number is returned to the **clone** open through ***devp**. The **clone** open then returns to the user a file descriptor that points to the new instantiation of the target driver.

The **echo** driver is an example of a clonable driver.

Notes

It is not possible to do multiple opens of a device with the same major and minor number using the **clone** driver. This is because the **clone** driver is only given the major number of the driver to be cloned, and that driver will then select a minor number which has not been opened.

When called with a pathname which corresponds to the clonable driver, **stat()** will return different results than **fstat()** when it is called on a file descriptor returned from **open()** of the same clonable driver pathname.

RETURN VALUES

If the **clone** driver is given an invalid minor number, or if the driver indicated is not a clonable driver, the **open()** fails and **errno** is set to [ENXIO].

SEE ALSO

open(2), fstat(2).

NAME

console, systty, syscon - system console interface

DESCRIPTION

/dev/console provides a **termio** interface to the device configured as the system console. The *init*(1M) manpage discusses the uses of **/dev/systty** and **/dev/syscon**.

Output data normally sent to the console, either through **/dev/console** or generated by a kernel **printf()**, may be redirected to another terminal or pseudo-terminal device through the **TIOCCONS** **ioctl()**. See *termio*(7) for details.

FILES

/dev/console
/dev/systty
/dev/syscon

SEE ALSO

init(1M), *termio*(7).

STANDARDS CONFORMANCE

console: SVID2, SVID3, XPG2

NAME

ddfa - Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software

DESCRIPTION

The Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software allows access from HP-UX system utilities and user applications to terminal servers using standard HP-UX structures. DDFA provides an interface to remote LAN-connected terminal server ports that is similar to the interface for local directly-connected ports.

The basic principle is that a daemon is created for each configured terminal server port based on information in a configuration file (a Dedicated Ports file). When the daemon is spawned, it takes a **pty** from the pool and creates a device file with the same major and minor number as the **pty** slave. The device file is known as the "pseudonym" and utilities and applications use the pseudonym to access the terminal server port by exercising standard HP-UX system functions (**open()**, **close()**, **read()**, **write()**, and **ioctl()**). The daemon listens on the **pty** until an application does an **open()** on the pseudonym. It then sets up and manages the connection to the terminal server port until the application does a **close()** on the pseudonym. The end result is that the terminal server port is addressed via a device file, but the mechanism that makes it happen is transparent to the user. A second configuration file (a port configuration file) contains information to profile the terminal server port.

DDFA consists of the following items:

- | | |
|----------------|---|
| dp | Dedicated Ports file. This text file contains the information that DDFA needs to set up and manage a connection between a pseudonym and a terminal server port.

The dp file is parsed by the Dedicated Port Parser (dpp) which spawns an Outbound Connection Daemon (ocd) for each outbound connection specified in the file. The dp file is also used by the HP-UX Telnet daemon (telnetd) to identify incoming connections from a DTC and map them to a pseudonym (the Telnet port identification feature). |
| pcf | Port Configuration File. This text file is used by DDFA to profile the terminal server port. The generic name of the template file is pcf . A port configuration file is referenced by an entry in the Dedicated Ports file (dp). |
| dpp | Dedicated Port Parser. This command parses the Dedicated Ports file (dp) and spawns an Outbound Connection Daemon (ocd) for each valid entry in the dp file. It can be run from the shell or it can be included in a system initialization script to automatically run the DDFA software each time the system is booted. |
| ocd | Outbound Connection Daemon. This daemon manages the connection and data transfer to the remote terminal server port. Normally, it is spawned by the Dedicated Ports Parser (dpp), but it can be run directly from the shell.

As it starts, it creates its pseudonym for the connection. As it terminates normally, it removes the pseudonym. If the pseudonym is removed while it is running, ocd will terminate with an error condition. |
| ocdebug | Outbound Connection Daemon debug mode. This is a special version of ocd that contains debugging code. It must be run from the shell. |

CONFIGURATION

There are two basic steps to configuring the DDFA software:

- Enter information in the **dp** file.
- Enter information in the port configuration files.

Configuring the dp File

The **dp** file contains one line for each outbound connection that is to be established and one line for each incoming connection request. A default file `/usr/examples/ddfa/dp` should be copied to a new file and the copy edited as needed. It is recommended that a directory be created to hold the **dp** file and the port configuration files.

Each line of the **dp** file must contain the location of the terminal server port and the location of the pseudonym. In addition, for an outbound connection, the port configuration file must be specified and a logging level may be specified.

Configuring the Port Configuration Files

A port configuration file is used to configure individual terminal server ports. A master port configuration file is `/usr/examples/ddfa/pcf`. In practice, it is renamed for each port that needs different configuration values and the values are altered appropriately for the device attached to the port. It is recommended that a directory be created to hold the port configuration files and the `dp` file.

Each line of a port configuration file must consist of a name of a variable and its value. The variable-value pairs contain information on how to open a connection to a terminal server port, how to close a connection to a terminal server port, and how to manage the data transfer to a terminal server port.

Configuring a System Initialization Script

DDFA can be run at boot time by including a reference to `dpp` in a system initialization script. It is recommended that the `-k` option be used when running `dpp` in this environment.

KILLING DAEMONS

Note that `ocd` should be killed using `kill -15`. Do not use `kill -9` for this purpose as it does not remove the device file. `ocd` verifies the validity of an existing pseudonym before trying to use it. `dpp` and `ocd` use data stored in the file `/var/adm/utmp.dfa` to verify whether a process still owns a pseudonym before taking it over. If `ocd` finds an unowned pseudonym, it uses it.

ERROR HANDLING

When `ocd` receives a serious error condition, such as when the LAN goes down, it transmits the error condition to the application by closing the `pty`. Any `open()`, `close()`, `read()`, or `write()` to the pseudonym returns the error condition `0 bytes read`. If the pseudonym is the controlling terminal for the group to which the application belongs, `SIGHUP` is sent to all the processes in the group, including the application.

ioctl() LIMITATIONS

Not all `ioctl()` functionality is available, due to the lack of a protocol that allows the transmission of such commands over the LAN to the remote port.

termio Attribute Limitations

The main restrictions on `termio` attributes (see `termio(7)`) include modem signal control and parity checking. The following are not available:

CBAUD IGMPAR INPCK IXANY IXOFF PARMRK

ioctl() Request Limitations

The following `ioctl()` request limitations apply:

CSTOPB flag	DTC only supports one stop bit.
CSIZE	DTC only supports 8 bits per character. Value cannot be modified.
PARODD flag	DTC offers static configuration to handle even or odd parity. It also handles auto parity detection for even or odd parity.
PARENB flags	Enabling/disabling done via static configuration. No programmatic interface supplied.
INPCK flag	No way to separate input from output parity features.
IGMPAR flag	Cannot be configured on DTC.
PARMRK	Bad characters are forwarded to the system without marking them with OFFH or OH.
CBAUD	Speed is part of static configuration.
IXOFF flag	Flow control is enabled if the DTC static configuration specifies an ASCII access mode. If binary is selected, no flow control is provided.
IXON flags	Pacing of output to a terminal via a programmatic interface is enabled when ASCII mode is selected in static port configuration and disabled when binary mode is selected.
IXANY flag	DTC does not offer the ability to restart output on any character received if XOFF was previously received.

HUPCL flag	DDFA does not support the hanging up of modem signals on the last close of the device file. If the modem signals used on the DTC drop, the connection is closed.
CLOCAL flag	Not supported.
c_flags	IENQACK not supported. OFILL, OFDEL, NLDLY, CRDLY, TABDLY, BSDLY, FFDLY not supported by Telnet port identification software.
BINARY mode flags	Part of static configuration is done in DTC Manager by selecting binary mode. If switching is enabled, binary can be selected at user interface level. There is no way to automatically negotiate binary mode when proper termio flags are reset when using telnetd . Binary/ASCII switching is possible with DDFA. The DTC cannot support large reads in pure binary mode, so transferred blocks of data should not be more than 256 bytes. If half-duplex with remote acknowledgement is implemented, binary applications can be supported.

ioctl() System Call Requests

The following **ioctl()** system call limitations apply:

TCSBRK	The ability to send a break without waiting for previous data to be sent is not provided at the system level in telnetd or DDFA. Receiving a Telnet break command in the DTC allows it to generate a break on asynchronous ports.
TCFLSH	The DTC output queue cannot be flushed.
Hardware handshake request	Not supported on DTC.
TCXONC	Local handshake cannot be disabled on DTC.
MCGETA	Not supported.
MCSETA, MCSETAF, MCSETAW	There is no way to separately set modem lines of a DTC port.
MCGETT	Modem timers, CD timer, connect timer, and disconnect cannot be configured.
CCITT simple, and direct call-in/call-out modes	DTC cannot handle simple mode because there is programmatic interface for modem signals. Call-in mode cannot be simulated if the port is opened, because modem signals (or the call) must be present within 2 minutes or the connection is cleared.
DACIDY get device adapter info	No way to get device adapter information.
Download ioctl() DACRADDR, DACDLADDR, DACDLGO, DACDLVER	No programmatic call to download the DTC.
DACHWSTATUS, DACSELFTEST, DACLOADED, DACISBROKE status	No programmatic interface to get such info.
DACLOOPBACK DACSUBTEST port test	

WARNINGS

In order to ensure that commands (such as *ps*) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, in order to ensure that commands (such as **w, passwd, finger, and wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
stty exta <&1 2>/dev/null
```

The printer interface scripts reside in the directory `/etc/lp/interface`. The line must be added just prior to the final `exit` command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

FILES

```
/usr/examples/ddfa/dp  
/usr/examples/ddfa/pcf  
/usr/sbin/dpp  
/usr/sbin/ocd  
/usr/sbin/ocdebug  
/var/adm/dpp_login.bin  
/var/adm/utmp.dfa
```

SEE ALSO

dpp(1M), ocd(1M), ocdebug(1M), ioctl(2), dp(4), pcf(4), ioctl(5), termio(7).


d

NAME

diag0 - diagnostic interface to HP-PB I/O subsystem

DESCRIPTION

diag0 is a diagnostic pseudo-driver, which provides HP support tools with access to the HP-PB I/O subsystem. This driver is used by hardware monitors and tools within the Support Tools Manager (STM), to interact with peripherals connected to the system via HP-PB. The I/O drivers also send diagnostic events to **diag0** for diagnostic logging by the Support Tools Manager.

Without **diag0**, information that could help prevent a peripheral failure will be lost. In addition, if a failure occurs, HP will not have the tools or data to diagnose the cause of the problem in a timely manner. This may cause increased downtime and possible future failures.

AUTHOR

diag0 was developed by HP.

FILES

/stand/vmunix
/dev/diag/diag0
/dev/diag directory containing diagnostic device files

SEE ALSO

stm(1M) from the Support Tools Manager

NAME

diag1 - diagnostic interface to the PCI I/O subsystem

DESCRIPTION

diag1 is a diagnostic pseudo-driver, which provides support tools with access to the PCI I/O subsystem. This driver is used by tools within the Support Tools Manager (STM) to interact with PCI cards connected to the system. Without **diag1**, support tools for PCI cards will not be able to operate.

WARNINGS

diag1 is not supported for HP-UX 11i Version 1.5.

AUTHOR

diag1 was developed by HP.

FILES

/stand/vmunix
/dev/diag/diag1
/dev/diag directory containing diagnostic device files

SEE ALSO

stm(1M) from the Support Tools Manager.


d

NAME

diag2 - interface for diagnostic logging and interface to processors

DESCRIPTION

diag2 is used by hardware monitors and tools within the Support Tools Manager (STM), to interact with processor hardware via Processor Dependent Code (PDC). Without **diag2**, support tools for processors will not be able to operate.

diag2 is also the key component for the following support features:

- I/O error logging
- Low priority machine check (LPMC) logging
- Memory error logging
- Pro-active memory page deallocation.

Without the above, information that could help prevent a system or peripheral failure will be lost. In addition, if a failure occurs, HP will not have the tools or data to diagnose the cause of the problem in a timely manner. This may cause increased downtime and possible future failures.

AUTHOR

diag2 was developed by HP.

FILES

- /stand/vmunix**
- /dev/diag/diag2**
- /dev/diag2**
- /dev/diag** directory containing diagnostic device files

SEE ALSO

stm(1M) from the Support Tools Manager

NAME

disk - direct disk access

DESCRIPTION

This entry describes the actions of HP-UX disk drivers when referring to a disk as either a block-special or character-special (raw) device.

Device File Naming Conventions

Standard disk device files are named according to the following conventions (see *intro(7)*):

Block-mode Devices	<code>/dev/disk/diskN[_pX]</code>
Character-mode Devices	<code>/dev/disk/diskN[_pX]</code>
Legacy block-mode Devices	<code>/dev/dsk/cxydn[sm]</code>
Legacy character-mode Devices	<code>/dev/rdisk/cxydn[sm]</code>

Legacy device special filenames are those used on HP-UX 11i Version 2 and earlier releases. They can still be used for backward compatibility, but only for part of the configuration within the limits of HP-UX 11i Version 2.

The component parts of the device filename are constructed as follows:

- N** Required. A decimal number corresponding to the instance number assigned to the direct access device by the operating system.
- X** Required if **_p** is specified. A decimal number corresponding to a partition number.
- c** Required. Identifies the following hexadecimal digits as the "Instance" of the interface card.
- x** Hexadecimal number identifying controlling bus interface, also known as the "Instance" of this interface card. The instance value is displayed in the *ioscan(1M)* output, column "I" for the H/W Type, "INTERFACE".
Required.
- t** Identifies the following hexadecimal digits as a "drive number" or "target".
Required.
- y** Hexadecimal number identifying the drive or target number (bus address).
Required.
- d** Identifies the following hexadecimal digits as a "unit number".
Required.
- n** Hexadecimal unit number within the device.
Required.
- s** Optional. Defaults to that corresponding to whole disk. Identifies the following value as a "section number".
- m** Required if **s** is specified. Defaults to section 0 (zero), whole disk. Drive section number.

Assignment of controller, drive, logical unit and section numbers is described in the system administrator manuals for your system.

Block-special access

Block-special device files access disks via the system's block buffer cache mechanism. Buffering is done in such a way that concurrent access through multiple opens and mounting the same physical device is correctly handled to avoid operation sequencing errors. The block buffer cache permits the system to do physical I/O operations when convenient. This means that physical write operations may occur substantially later in time than their corresponding logical write requests. This also means that physical read operations may occur substantially earlier in time than their corresponding logical read requests.

Block-special files can be read and written without regard to physical disk records. Block-special file **read()** and **write()** calls requiring disk access result in one or more **BLKDEV_IOSIZE** byte (typically 2048 byte) transfers between the disk and the block buffer cache. Applications using the block-special device should ensure that they do not read or write past the end of last **BLKDEV_IOSIZE** sized block in the device file. Because the interface is buffered, accesses past this point behave unpredictably.

Character-special access

Character-special device files access disks without buffering and support the direct transmission of data between the disk and the user's read or write buffer. Disk access through the character special file interface causes all physical I/O operations to be completed before control returns from the call. A single read or write operation up to **MAXPHYS** bytes (typically 64 Kbytes or 256 Kbytes) results in exactly one disk operation. Requests larger than this are broken up automatically by the operating system. Since large I/O operations via character-special files avoid block buffer cache handling and result in fewer disk operations, they are typically more efficient than similar block-special file operations.

There may be implementation-dependent restrictions on the alignment of the user buffer in memory for character special file **read()** and **write()** calls. Also, each read and write operation must begin and end on a logical block boundary and must be a whole number of logical blocks in size. The logical block size is a hardware-dependent value that can be queried with the **DIOC_DESCRIBE_EXT** and **DIOC_DESCRIBE** ioctl calls, which are described below.

In addition to reading and writing data, the character-special file interface can be used to obtain device specific information and to perform special operations. These operations are controlled through use of ioctl calls. Details related to these ioctls are contained in **<sys/diskio.h>**.

The **DIOC_DESCRIBE_EXT** and **DIOC_DESCRIBE** ioctl can be used to obtain device specific identification information. The information returned includes the disk's model identification, the disk interface type, maximum offset address, device type, and the disk's logical block size.

The **DIOC_CAPACITY** ioctl can be used to obtain the capacity of a disk device in **DEV_BSIZE** units. (**DEV_BSIZE** is defined in **<sys/param.h>**).

The **DIOC_EXCLUSIVE** ioctl can be used to obtain and release exclusive access to a disk device. Exclusive access is required for some special operations, such as media reformatting, and may be desirable in other circumstances. The value one specifies that exclusive access is requested. The value zero specifies the exclusive access should be released. Exclusive access causes other open requests to fail. Exclusive access can only be granted when the device is not currently opened in block-mode and there is only one open file table entry for that disk device (the one accessible to the exclusive access requester).

ERRORS

The following errors can be returned by a disk device driver call:

[EACCES]	Required permission is denied for the the device or operation.
[EIO]	I/O error (e.g., media defect or device communication problem).
[EINVAL]	From an open() call: the device is not a disk device. For other calls: Invalid request or parameter. Note that for legacy, 32-bit access, this error can result when the size of the device overflows the argument of the DIOC_DESCRIBE or DIOC_CAPACITY ioctls.
[ENXIO]	If resulting from an open() call, this indicates there is no device at the specified address. For other calls, this indicates the specified address is out of range or the device can no longer be accessed.

WARNINGS

The interaction of block-special and character-special file access to the same **BLKDEV_IOSIZE**-sized block is not specified, and in general is unpredictable.

On some systems, having both a mounted file system and a block special file open on the same device can cause unpredictable results; this should be avoided if possible. This is because it may be possible for some files to have private buffers in some systems.

Although disk devices have historically had small (typically 512-byte) block sizes, some disk devices (such as optical disks and disk arrays) have relatively large block sizes. Applications using direct raw disk access should use **ioctl()** calls to determine appropriate I/O operation sizes and alignments.

Any disk with removable media (for example, floppy or CD-ROM) containing a mounted file system should not be removed prior to being unmounted. Removal of disk media containing mounted file systems is likely to result in file system errors and system panics.

AUTHOR

disk was developed by HP and AT&T.

SEE ALSO

ioscan(1M), mknod(1M), intro(7).

System Administrator manuals included with your system.



d

NAME

dlpi - data link provider interface

DESCRIPTION

This manual page gives a brief description on DLPI (the data link provider interface) and how to interface with the set of API's that are provided by DLPI.

HP-UX DLPI serves as a Layer 2 (Data Link Layer) of an OSI architecture. DLPI serves as an interface between LAN device drivers and DLPI users. DLPI is intended for use by experienced network users only.

HP-UX DLPI has two broader sets of interface. The first set of interfaces are provided as per the DLPI 2.0 standard and the second set that are HP extensions to the standard.

HP-UX DLPI also provides interfaces to device drivers to interface with STREAMS modules and DLPI applications.

For STREAMS Modules and DLPI Applications

Hewlett-Packard's implementation of DLPI is a Style 2 service provider. The Style 2 provider requires a DLS user to identify a PPA explicitly, using a special attach service primitive. Refer to the *lan(7)* manual page for more information on PPA.

HP DLPI offers the following services to STREAMS modules and DLPI applications:

- Clone (maximum of 3992) and non-clone (maximum of 100) access.
- Support for Ethernet/IEEE802.3, FDDI and Token Ring interfaces.
- Support for connectionless and connection-mode services (connection-mode services are supported only over IEEE802.3 and Token Ring).
- Supports raw-mode services.
- **I_STR** ioctl is supported for doing device-specific control and diagnostic requests.
- Support for third-party device drivers.
- Support for all levels of promiscuous mode.

HP DLPI does not offer the following for STREAMS modules and DLPI applications:

- Quality of Service (QOS) management.
- Connection Management STREAMS: **DL_SUBS_BIND_REQ** and **DL_SUBS_UNBIND_REQ** over connection-oriented STREAMS.
- Acknowledged connectionless-mode services.

The DLPI requests based on DLPI 2.0 standard are defined in **<dlpi.h>**; see *dlpi(4)*. HP extensions for DLPI are defined in **<dlpi_ext.h>**; see *dlpi_ext(4)*.

Device File Format

To access LAN drivers via DLPI interface, DLS users must use the following device files:

Name	Type	Major #	Minor #	Access Type
----	---	-----	-----	-----
/dev/dlpi	c	72	0x77	Clone access
/dev/dlpiX	c	119	0xX	Non-Clone access

For Device Drivers

HP-UX DLPI is of non-native design. The drivers and DLPI are not coupled together and exists as individual components on the system. The non-native DLPI supports two kinds of drivers. Tightly coupled and loosely coupled drivers.

DLPI provides interfaces to tightly coupled and loosely coupled drivers. DLPI serves as a sole interface to DLS users for tightly coupled drivers. Whereas, a loosely coupled driver depends on DLPI only to provide information to user-space commands *lanscan(1M)* and *nwmgr(1M)* for display purposes.

The interfaces for device drivers is defined in **<dlpi_drv.h>**, see *dlpi_drv(4)*.

DLPI provides the following functionality for tightly coupled drivers:

- Infrastructure that allows drivers to communicate with upper layer STREAMS modules or applications.

- Infrastructure for protocol, multicast and promiscuous processing.
- Infrastructure for asynchronous processing of control.
- Inbound frame processing.
- Processing link up and down events.
- Repository for all registered interfaces and associated information.
- Outbound processing before hand off to physical drivers.

DLPI provides its services through three header files that are exported. The header files `<dlpi.h>` and `<dlpi_ext.h>` are for user space applications and kernel level STREAMS modules. The header file `<dlpi_drv.h>` is for physical and logical drivers.

WARNINGS

Various implementations of DLPI exists within HP-UX for special technologies like ATM, Hyper Fabric, etc.; but the DLPI that supports LAN class drivers (tightly coupled) is the one covered by this manual page.

The `lanadmin`, `lanscan`, and `linkloop` commands are deprecated. These commands will be removed in a future HP-UX release. HP recommends the use of replacement command `nwmgr(1M)` to perform all network interface-related tasks.

AUTHOR

`dlpi` was developed by HP, based on DLPI 2.0 standard.

SEE ALSO

`lanscan(1M)`, `nwmgr(1M)`, `dlpi(4)`, `dlpi_drv(4)`, `dlpi_ext(4)`, `lan(7)`.

DLPI Programmer's Guide, 2003, Hewlett-Packard
Driver Development Guide, Hewlett-Packard
Device Driver Reference, Hewlett-Packard

NAME

framebuf - information for raster frame-buffer devices

SYNOPSIS

```
#include <sys/framebuf.h>
```

DESCRIPTION

Frame-buffer devices are raster-based displays. These devices use memory-mapped I/O to obtain much higher performance than possible with tty-based graphic terminals. Frame-buffer devices can be accessed directly using this interface, although access through the graphics libraries is recommended. Direct access to frame-buffer devices entails precise knowledge of the frame-buffer architecture being used. Input cannot be piped into or redirected to frame-buffer devices because they are not serial devices.

Each frame-buffer device is associated with a character special file. Major and minor numbers for frame-buffer devices are implementation-dependent. The minor numbers for these devices denote different frame buffers. Implementation-specific details are discussed in the appropriate systems administrator's manuals.

Communication with a frame-buffer device begins with an **open()** system call. Multiple processes can have the frame-buffer device open concurrently.

close() invalidates the file descriptor associated with the frame-buffer device. After a **close()** system call, any access to the frame-buffer device address range might result in a memory fault and the signal SIGSEGV being sent to the process (see *signal(2)*). A process cannot unmap the frame buffer from its address space after the frame-buffer special file is closed. To unmap a frame buffer, use the **GCUNMAP ioctl()** call (see below).

Once a process acquires a lock for the frame-buffer device, it must unlock it explicitly before calling **close()**; see **GCUNLOCK** below.

read() and **write()** system calls are undefined and always return an error. In this case **errno** is set to [ENODEV].

The **ioctl()** system call is used to control a frame-buffer device. The **select()** system call is used to test the frame-buffer device for exceptional conditions. Interrupts from the graphic hardware are considered exceptional conditions. An exceptional condition is automatically cleared after any process that opens the frame-buffer device is notified of the exception by a **select()** call. A call to **select()** for read or write on the file descriptor associated with the frame-buffer device returns a false condition in the read and write bit masks (see *select(2)*).

A frame-buffer device can be accessed by multiple processes at once. However, each process overwrites the output of the others unless one of the lock mechanisms described here or some other synchronization mechanism is used. The lock mechanisms described here are intended for cooperating processes only.

For all frame buffers, data bytes scan from left to right and from top to bottom. A pixel, which is a visible dot on the screen, is associated with a location in the frame buffer. Each device maps one or more bits in memory to a pixel on the screen, although the bits in the frame buffer might not be continuous. Information describing the frame-buffer structure and attributes is found in the **crt_frame_buffer_t** data structure. The **crt_frame_buffer_t** data structure includes the following fields:

```
int crt_id;                /*display identifier*/
unsigned int crt_attributes; /*flags denoting attributes*/
char *crt_frame_base;     /*first byte in frame-buffer memory*/
char *crt_control_base;   /*first byte of the control*/
                           /*registers*/
char *crt_region [ CRT_MAX_REGIONS ];
                           /*other regions associated with the*/
                           /*frame-buffer device*/
```

The following are valid **ioctl()** requests:

GCDESCRIBE

Describe the size, characteristics, and mapped regions of the frame buffer. The information is returned to the calling process in a **crt_frame_buffer_t** data structure, and the parameter is defined as **crt_frame_buffer_t *arg**. Although some structure fields contain addresses of one or more frame-buffer device regions, the values of these fields are not always defined. Only after a successful **GCMAP** command is issued (see below) are the correct addresses returned so the user can access the frame-buffer regions directly using the returned addresses.

- GCID** Provide a device identification number. The parameter is defined as `int *arg;`. The information returned when using this command is a subset of the information provided by **GCDESCRIBE**, and is provided here for backward compatibility only.
- GCON, GCOFF** Turn graphics on or off. These operations are valid for devices whose `CRT_GRAPHICS_ON_OFF` bit is set in the `crt_attributes` field of the `crt_frame_buffer_t` data structure returned by the **GCDESCRIBE** command. Otherwise, these commands have no effect.
- GCAON, GCAOFF** Turn alpha on or off. These operations are valid for devices whose `CRT_ALPHA_ON_OFF` bit is set in the `crt_attributes` field of the `crt_frame_buffer_t` data structure returned by the **GCDESCRIBE** command. Otherwise, these commands have no effect.
- GCMAP** Make the frame-buffer memory, graphics control, and other device regions accessible to the user process making the call. Only processes that request this can directly access frame-buffer memory and control registers. After a successful **GCMAP** call, the fields `crt_frame_base` and `crt_control_base` in the `crt_frame_buffer_t` data structure (returned by a subsequent **GCDESCRIBE** `ioctl()` call), hold the valid addresses of these two regions of the frame buffer. If, for a specific device, more than two regions are to be mapped to the user's address space, the base addresses of up to `CRT_MAX_REGIONS` extra device regions will be placed in the array `crt_region` in successive order. Only the regions pertinent to a specific frame buffer are mapped. Irrelevant region fields in the `crt_frame_buffer_t` data structure are set to 0. Use of the `arg` parameter is implementation dependent (see *DEPENDENCIES* below). The base addresses for frame-buffer regions are always page aligned.
- GCUNMAP** Cause access to the frame-buffer memory, graphics control, and possibly other device regions to be removed from the requesting process. The parameter `arg` is ignored and should be set to 0. Any attempt to access these memory regions after a successful **GCUNMAP** call results in a memory fault and sends the signal `SIGSEGV` to the process.
- GCLOCK** Provide for exclusive use of the frame-buffer device by cooperating processes. The calling process either locks the device and continues or is blocked. Blocking in this case means that the call returns only when the frame buffer is available or when the call is interrupted by a signal. If the call is interrupted, it returns an error and `errno` is set to `[EINTR]`. Waiting occurs if another process has previously locked this frame buffer using the **GCLOCK** command and has not executed a **GCUNLOCK** command yet. The **GCLOCK** command does not prevent other non-cooperating processes from writing to the frame buffer; thus, **GCLOCK** is an advisory lock only. The parameter `arg` is ignored and should be set to 0.
- This call prevents the Internal Terminal Emulator (ITE) from corrupting the state of the graphics hardware (see *termio(7)*). On some systems, as long as the frame buffer is locked with a **GCLOCK** command, the ITE does not output text to it (see *DEPENDENCIES* below). Any attempt to lock the device more than once by the same process fails, and causes `errno` to be set to `[EBUSY]`.
- GCLOCK_NOWAIT** Provide for exclusive use of the frame-buffer device by cooperating processes. This request has the same effect on the frame-buffer device as does the **GCLOCK** request. However, this call does not wait for the frame buffer to be released by other processes. If the frame-buffer device is locked, the process is not blocked; instead, the system call returns an error and causes `errno` to be set to `[EAGAIN]`. The parameter `arg` is ignored and should be set to 0.
- GCLOCK_BLOCKSIG** Provide for exclusive use of the frame-buffer device by cooperating processes while blocking all incoming signals for the calling process that otherwise might have been caught. This call is a superset of the **GCLOCK** call. The parameter `arg` is ignored and should be set to 0. When the display is acquired for exclusive use (and thus locked), all signals sent to the process that otherwise would have been caught by the process "at the time of the" **GCLOCK** call are withheld (blocked) until **GCUNLOCK** is requested. Any attempt to modify the signal mask of the process (see *sigsetmask(2)*) before a **GCUNLOCK** request is made will not have any effect on these blocked signals.

The signals are not blocked until the lock is actually acquired, and might be received while still awaiting the lock.

The signal SIGTSTP is also blocked whether or not it is being caught. The signals SIGTTIN and SIGTTOU are also blocked on frame-buffer devices where the ITE does not output to the device while it is locked. See *DEPENDENCIES* below.

Except for the three signals mentioned above, this call does not block signals that the process did not expect to catch, nor does it block signals that cannot be caught or ignored. This command does not prevent other non-cooperating processes from writing to the frame buffer.

GCLOCK_BLOCKSIG_NOWAIT

Provide for exclusive use of the frame-buffer device by cooperating processes, while blocking all incoming signals for the calling process that otherwise would have been caught. This request has the same effect on the frame-buffer device as does the **GCLOCK_BLOCKSIG** request. However, this call does not wait for the frame buffer to be released by other processes. If the frame-buffer device is locked, the process is not blocked, but the system call returns an error and causes **errno** to be set to [EAGAIN]. The parameter *arg* is ignored and should be set to 0.

GCUNLOCK

Relinquish exclusive use of the frame-buffer device. If the device is locked with a **GCLOCK_BLOCKSIG** or **GCLOCK_BLOCKSIG_NOWAIT ioctl()** request, the signal mask of the calling process is restored to its state prior to the locking request.

GCRESET

Reset the graphic hardware associated with the frame-buffer device to a defined initial state. The call enables the frame-buffer device to respond to the **ioctl()** requests defined here.

GCDMA_OUTPUT

Send DMA output to the frame-buffer device. This system call is used to transfer data from a user's array to a rectangular area of the graphics frame-buffer, or optionally, to the device's graphics control space.

The parameters for the DMA are passed in a **crt_dma_ctrl_t** data structure, which includes the following fields:

```
char *mem_addr;      /* Starting address of data
                     being transferred */
char *fb_addr;      /* Address of framebuffer
                     destination */
int length;         /* Number of bytes to transfer,
                     including those "skipped" */
int linelength;    /* Number of bytes written
                     on each framebuffer row */
int skipcount;     /* Number of source bytes to
                     ignore after each "linelength" */
unsigned int flags; /* Specified options to the driver */
```

To write to the graphics frame-buffer, set **fb_addr** to the address of the upper-left corner of the rectangle to be drawn. The DMA will write **linelength** bytes on each frame-buffer row, ignore the next **skipcount** bytes of memory data, then resume writing at the same starting position on each succeeding frame-buffer row. This is continued until **length** bytes are either written or ignored.

To write to the graphics control space, set **fb_addr** to the address of the first graphics control register to write. In this case, **linelength** and **skipcount** are ignored.

The **flags** parameter specifies options for the DMA. Currently, there are no supported flags and this parameter should be set to zero, otherwise the system call will fail and **errno** is set to [EINVAL].

The DMA has the same effect on the frame-buffer device as using store instructions to write the data. Thus, various graphics control registers may affect the results of the DMA. It is the responsibility of the user program to perform any necessary set-up of the frame-buffer device so that the DMA has the desired results.

The **skipcount** parameter allows the user to refresh a portion of a window image that the user has stored in memory for those cases where only a portion of the image

needs to be refreshed. The window image is then a superset of the rectangle being updated, and might thus have different dimensions. The **skipcount** specifies the portion of the row in the larger window image that is excluded from the rectangle. Thus, **linelength** plus **skipcount** would be the number of bytes in each row of the larger window image array.

If a particular framebuffer device supports this system call, the **CRT_DMA_OUTPUT** flag in the **crt_attributes** field of the **crt_frame_buffer_t** structure is set. Some framebuffer devices supporting DMA might restrict alignment of the various parameters, and are specified in the **DEPENDENCIES** section below. The kernel ensures that these restrictions are obeyed, and if they are not the system call will fail and set **errno** to **[EINVAL]**.

It is the responsibility of the application to guarantee that the system's physical memory is up-to-date by flushing the processor's data cache. One should use the **GCDMA_DATAFLUSH** ioctl to ensure that the data is consistent before initiating a DMA transfer.

GCDMA_DATAFLUSH

Flush the specified data from the processor's data cache to the system's main memory. This system call is intended to be used before DMA to ensure that an up-to-date version of the data is transferred to the framebuffer or to control space.

The parameters for the flush are passed in a **crt_flush_t** data structure, which includes the following fields:

```
char *flush_addr; /* Starting address of data
                  to be flushed */
int flush_len;   /* Number of bytes to flush */
```

The kernel ensures that the **flush_len** bytes starting at **flush_addr** are consistent in main memory with respect to the cache.

GCSLOT

Provide pertinent information about the calling process's participation in the system-wide graphics locking mechanism (see the discussion under **GCLOCK** above). The **GCSLOT** request does not carry out any actual locking functionality. The lock information is returned to the calling process in a **crt_gcslot_t** data structure. The parameter is defined as **crt_gcslot_t *arg**; the **crt_gcslot_t** data structure is defined in the file **<sys/framebuf.h>**.

GCSTATIC_MAP

Prevent the Internal Terminal Emulator (ITE) from modifying the device's color map.

GCVARIABLE_MAP

Allow the Internal Terminal Emulator (ITE) to modify the device's color map.

DEPENDENCIES

When requesting **GCMAP**, the parameter *arg* is ignored and should be set to **0**.

All supported ITEs ignore the frame buffer lock for output.

ERRORS

- | | |
|-----------------|---|
| [EAGAIN] | The operation would result in suspension of the calling process, but the request was either GCLOCK_NOWAIT or GCLOCK_BLOCKSIG_NOWAIT . |
| [EBUSY] | Attempted to lock the device, which is already locked by the same process. |
| [EINTR] | A call to ioctl() was interrupted by a signal. |
| [EINVAL] | An invalid ioctl() command was made. |
| [ENODEV] | Attempted to use read() or write() system calls on the device. |
| [ENOMEM] | Sufficient memory for mapping could not be allocated. |
| [ENOSPC] | Required resources for mapping could not be allocated. |
| [ENXIO] | The minor number on the device file refers to a nonexistent device. |
| [EPERM] | Requested GCUNLOCK ioctl() command, but the device was locked by a different process. |

AUTHOR

framebuf was developed by HP.

SEE ALSO

mknod(1M), close(2), ioctl(2), lockf(2), open(2), select(2), signal(2), sigsetmask(2), termio(7).



f

NAME

gang_sched - Gang Scheduler

DESCRIPTION

The gang scheduler permits a set of MPI (Message Passing Interface) processes, or multiple threads from a single process, to be scheduled concurrently as a group.

Gang scheduling is enabled and disabled by setting the **MP_GANG** environment variable to **ON** or **OFF**.

The gang scheduling feature can significantly improve parallel application performance in loaded timeshare environments that are oversubscribed. Oversubscription occurs when the total number of runnable parallel threads, runnable MPI processes, and other runnable processes exceeds the number of processors in the system.

Gang scheduling also permits low-latency interactions among threads in shared-memory parallel applications.

Only applications using the HP-UX V11.0 MPI or pthread libraries can be gang scheduled. Because HP compiler parallelism is primarily built on the pthread library, programs compiled with HP compilers can benefit from gang scheduling.

INTERFACE

The HP-UX gang scheduler is enabled and disabled using an environment variable. The variable is defined as:

```
MP_GANG [ON | OFF]
```

Setting **MP_GANG** to **ON** enables gang scheduling and setting it to **OFF** disables it. If **MP_GANG** is not set, or if it is set to an undefined value, no action is taken.

Gang scheduling is a process attribute that is inherited by child processes created by **fork** (see *fork(2)*). The state of gang scheduling for a process can change only following a call to **exec** (see *exec(2)*).

BEHAVIOR

After the **MP_GANG** environment variable is set to **ON**, any MPI or pthread application to execute and find this variable will enable gang scheduling for that process.

Only the pthread and MPI libraries query the **MP_GANG** variable--the operating system does not.

Gang scheduling is an inherited process attribute. When a process with gang scheduling enabled creates a child process, the following occurs:

- The child process inherits the gang scheduling attribute.
- A new gang is formed for the child process. The child does not become part of its parent's gang.

The gang scheduler is engaged only when a gang consists of multiple threads. For a pthread application, this is when a second thread is created. For an MPI application, it is when a second process is added.

As a process creates threads, the new threads are added to the process's gang if gang scheduling is enabled for the process. However, once the size of a gang equals the number of processors in the system, the following occurs:

- New threads or processes are not added to the gang.
- The gang remains intact and continues to be gang scheduled.
- The spill-over threads are scheduled with the regular timeshare policies.
- If threads in the gang exit (thus making room available), the spill-over threads are not added into the gang. However, newly created threads are added into the gang when room is available.

MPI processes are allocated statically at the beginning of execution. When **MP_GANG** is set to **ON**, all processes in an MPI application are made part of the same gang.

Thread and process priorities for gangs are managed identically to timeshare policy. The timeshare priority scheduler determines when to schedule a gang and adheres to the timeshare policies.

Although it is likely that scheduling a gang will preempt one or more higher priority timeshare threads, over the long run the gang scheduler policy is generally fair. All threads in a gang will have been highest priority by the time a gang is scheduled. Because all threads in a gang must execute concurrently, some threads do not execute when they are highest priority (the threads must wait until all other threads have

also been selected, allowing other processes to run first).

Gangs are scheduled for a single time-slice. The time-slice is the same for all threads in the system, whether gang-scheduled or not.

When a single gang executes on a system, the gang's threads are assigned to processors in the system and are not migrated to different processors.

In an oversubscribed system with multiple gangs, all gangs are periodically moved in order to give an equalized percentage of CPU time to each of the different threads. This rebalancing occurs every few seconds.

EXTERNAL INFLUENCES

Environment Variables

The following environment variables affect gang scheduling of processes:

- **MP_GANG** enables (when set to **ON**) and disables (when set to **OFF**) gang scheduling of processes. For details see the **INTERFACE** section of this man page.
- **MP_NUMBER_OF_THREADS** specifies the number of processors available to execute programs compiled for parallel execution. If not set, the default is the number of processors in the system.

PERFORMANCE

Gang scheduling ensures that all runnable threads and processes in a gang are scheduled simultaneously. This improves the synchronization latency in parallel applications. For instance, threads waiting at a barrier do not have to wait for currently unscheduled threads.

However, applications with lengthy parallel regions and infrequent synchronization may perform best when not gang scheduled. For those applications, some threads can be scheduled even if all threads are not scheduled at once.

A gang-scheduled application's performance can be affected by the number of gang-scheduled applications on a system, and by the number of threads in each. The gang scheduler assigns parallel applications to CPUs using a "best fit" algorithm that attempts to minimize CPU overlap among applications.

On systems with complex workloads including gangs of varying sizes, or odd combinations of sizes, the workload may not optimally match the number of CPUs available. In this situation an application may perform better when not gang scheduled, thus enabling some threads to be scheduled rather than waiting for all threads to be scheduled as a gang.

Scheduling Overhead

Gang scheduling incurs overhead when the scheduler collects a set of threads, assigns a set of processors to the threads, and rendezvous the set of threads and processors to achieve concurrent execution.

On an idle system, the gang scheduling overhead can be seen in the execution time of a single parallel application.

Kernel Blocking of Threads

If a thread from a gang blocks in the kernel, the thread's processor is available to run other non-gang-scheduled threads. When the blocked thread resumes and its gang is currently running, the thread can join the other ganged threads without having to rendezvous again.

In a multi-gang environment, thread blocking can result in lower throughput. This occurs if an application's threads block often in the kernel for long periods of time.

Preempting by Realtime Threads

Gang-scheduled threads can be preempted from execution by realtime threads. This affects only the gang-scheduled thread running on the processor being preempted by a realtime thread. The remaining threads of the gang continue to run through the end of their time-slice.

RESTRICTIONS

For this implementation of gang scheduling, the following restrictions exist. Some of these may be removed in future releases.

- Gang scheduling of processes being debugged is not supported. When a debugger attaches to a process, gang scheduling for the process is disabled. This avoids gang scheduling processes with one or more threads stopped by a debugger.

- Gang scheduling is completely shut down when Process Resource Manager (PRM) is enabled.
- If a gang-scheduled process is selected to be swapped out, the process will not be gang-scheduled when it is swapped back in.
- Realtime processes are not gang-scheduled.
- Gang scheduling is only supported for processes with timeshare scheduling policies.
- When a gang-scheduled process contains the maximum number of threads (or the maximum number of processes, for MPI applications), threads or processes created after this point are not scheduled as part of the gang. For details see the BEHAVIOR section of this man page.
- Multiprocess applications that do not use MPI are not supported by the gang scheduler.
- Gang scheduling is not supported for **PTHREAD_SCOPE_PROCESS** threads. From release 11i Version 1.6 of HP-UX, the default scheduling contention scope for threads is **PTHREAD_SCOPE_PROCESS**. If any **PTHREAD_SCOPE_PROCESS** threads are created by an application, the initial thread will be treated as a **PTHREAD_SCOPE_PROCESS**.

FILES

The following are libraries used in providing gang scheduling:

/usr/lib/libpthread.1 The pthread library.

/opt/mpi The directory containing MPI libraries and MPI software. HP MPI is an optional product.

SEE ALSO

fork(2), exec(2).

NAME

hil - HP-HIL device driver

SYNOPSIS

```
#include <sys/hilioctl.h>
```

DESCRIPTION

HP-HIL, the Hewlett-Packard Human Interface Link, is the Hewlett-Packard standard for interfacing a personal computer, terminal, or workstation to its input devices. **hil** supports devices such as keyboards, mice, control knobs, ID modules, button boxes, digitizers, quadrature devices, bar code readers, and touchscreens.

On systems with a single link, HP-HIL device file names use the following format:

```
/dev/hiln
```

where *n* represents a single digit that specifies the physical HP-HIL device address, which ranges from 1 to 7. For example, `/dev/hil3` is used to access the third HP-HIL device.

On systems with more than one link, HP-HIL device file names use the following format:

```
/dev/hil_m.n
```

where *m* represents the instance number, and *n* represents the physical HP-HIL device address. For example, `/dev/hil_0.2` would be used to access the second device on the link which has an instance number of zero. Likewise, `/dev/hil_12.7` references the seventh device on the link with instance number twelve.

Note that HP-HIL device addresses are determined only by the order in which devices are attached to the link. The first device attached to the link becomes device one, the second device attached becomes device two, etc.

HP-HIL devices are classified as "slow" devices. This means that system calls to **hil** can be interrupted by caught signals (see *signal(5)*).

hil can only read HP-HIL keyboards in raw keycode mode. Raw keycode mode means that all keyboard input is read unfiltered. HP-HIL keyboards return keycodes that represent key press and key release events.

Use *hilkbd(7)* to read mapped keycodes from HP-HIL keyboards. Use the Internal Terminal Emulator (ITE) described in *termio(7)* to read ASCII characters from HP-HIL keyboards.

System Calls

open(2) gives exclusive access to the specified HP-HIL device. Any previously queued input from the device is discarded. If the device is a keyboard, it is opened in raw keycode mode. A side effect of opening a keyboard in raw keycode mode is that the ITE (see *termio(7)*) and mapped keyboard driver (see *hilkbd(7)*) lose input from that keyboard until it is closed. Only device implemented auto-repeat functionality is available while in raw keycode mode (see HILER1 and HILER2).

The file status flag, `O_NDELAY`, can be set to enable nonblocking reads (see *open(2)*).

close(2) returns an HP-HIL keyboard to mapped keycode mode, making its input available to the ITE or mapped keyboard driver (see *hilkbd(7)*).

read(2) returns data from the specified HP-HIL device, in time-stamped packets:

```
unsigned char packet_length;
unsigned char time_stamp[4];
unsigned char poll_record_header;
unsigned char data[ packet_length - 6 ];
```

packet_length specifies the number of bytes in the packet including itself, and can range from six to twenty bytes. *time_stamp*, when repacked into an integer, specifies the time, in tens of milliseconds, that the system has been running since the last system boot. The most significant byte of the time stamp is *time_stamp*[0]. *poll_record_header* indicates the type and quantity of information to follow, and reports simple device status information. The number of data bytes is device dependent. Refer to the text listed in *SEE ALSO* for descriptions of the *poll_record_header* and device-specific data.

Usually two system calls are required to read each data packet, the first system call reads the data packet length; the second system call reads the actual data packet. Some devices always return the same amount

of data in each packet, in which case the count and the packet can both be read in the same system call.

If the file status flag, `O_NDELAY`, is set and no data is available, `read(2)` returns `0` instead of blocking.

`write(2)` is not supported by `hil`.

`select(2)` can be used to poll for available input from HP-HIL devices. `select(2)` for write or for exception conditions always returns a false indication in the file descriptor bit masks.

`ioctl(2)` is used to perform special operations on HP-HIL devices. `ioctl(2)` system calls all have the form:

```
int ioctl(int fildes, int request, char *arg);
```

The following `request` codes are defined in `<sys/hilioctl.h>`:

HILID Identify and Describe

This request returns the Identify and Describe Record in the `char` variable to which `arg` points, as supplied by the specified HP-HIL device. The Identify and Describe Record is used to determine the type and characteristics of each device connected to the link. The Identify and Describe Record can vary in length from 2 to 11 bytes. The record contains at least:

- A Device ID byte, and
- A Describe Record Header byte.

The Device ID byte is used to identify the general class of a device, and its nationality in the case of a keyboard or keypad. The Describe Record Header byte describes the position report capabilities of the device. The Describe Record Header byte also indicates if an I/O Descriptor byte follows at the end of the Describe Record. It also indicates support of the Extended Describe and the Report Security Code requests. If the device is capable of reporting any coordinates, the Describe Record contains the device resolution immediately after the Describe Record Header byte. If the device reports absolute coordinates, the maximum count for each axis is specified after the device resolution. The I/O Descriptor byte indicates how many buttons the device has. The I/O Descriptor byte also indicates device proximity detection capabilities and specifies Prompt/Acknowledge functions. All HP-HIL devices support the Identify and Describe request.

HILPST Perform Self Test

This request causes the addressed device to perform its self test, and returns the one-byte test result in the `char` variable to which `arg` points. A test result of zero indicates a successful test, non-zero results indicate device-specific failures. All HP-HIL devices support the Self Test request.

HILRR Read Register

The Read Register request expects an HP-HIL device register address in the `char` variable to which `arg` points, and returns the one-byte contents of that register in `*arg`. The Extended Describe Record indicates whether a device supports the Read Register request.

HILWR Write Register

The Write Register request expects `*arg` to contain a record containing one or more packets of data, each containing the HP-HIL device register address and one or more data bytes to be written to that register. There are two types of Register Writes. Type 1 can be used to write a single byte to each individual device register. Type 2 can be used to write several bytes to one register. The Extended Describe Record indicates if a device supports either or both types of register write requests.

HILRN Report Name

The Report Name request returns the device description string in the character array to which `arg` points. The string may be up to fifteen characters long. The Extended Describe Record indicates support of the Report Name request.

HILRS Report Status

The Report Status request returns the device-specific status information string in the character array to which `arg` points. The string can be up to fifteen bytes long. The Extended Describe record indicates support of the Report Status request.

- HILED** Extended Describe
- The Extended Describe request returns the Extended Describe Record in the character array to which *arg* points. The Extended Describe Record may contain up to fifteen bytes of additional device information. The first byte is the Extended Describe Header, which indicates whether a device supports the Report Status, Report Name, Read Register, or Write Register requests. If the device implements the Read Register request, the maximum readable register is specified. If the device supports the Write Register request, the Extended Describe Record specifies whether the device implements either or both of the two types of register writes and the maximum writeable register. If the device supports Type 2 register writes, the maximum write buffer size is specified. The Extended Describe Record can also contain the localization (language) code for a device. Support of the Extended Describe request is indicated in the Describe Record Header byte.
- HILSC** Report Security Code
- The Report Security Code request returns the Security Code Record in the character array to which *arg* points. The Security Code Record can be between one and fifteen bytes of data that uniquely identifies that particular device. Applications can use this request to implement a hardware "key" that restricts each copy of the application to a single machine or user. An application can read the Security Code Record from an HP-HIL ID Module and then verify that the application is running on a specific machine or that the application is being used by a legitimate user. Devices indicate support of the Report Security Code request in the Describe Record Header.
- HILER1** Enable Auto Repeat Rate = 1/30 Second
- This request is used to enable the "repeating keys" feature implemented by the firmware of some HP-HIL keyboard and keypad devices. It also sets the cursor key repeat rate to 1/30 sec. This request does not use *arg*.
- HILER2** Enable Auto Repeat Rate = 1/60 Second
- This request is used to enable the "repeating keys" feature implemented in the firmware of some HP-HIL keyboard and keypad devices. It also sets the cursor key repeat rate to 1/60 sec. This request does not use *arg*.
- HILDKR** Disable Keyswitch Auto Repeat
- This request turns off the "repeating keys" feature implemented in the firmware of some HP-HIL keyboard and keypad devices. This request does not use *arg*.
- HILP1..HILP7** Prompt 1 through Prompt 7
- These seven requests are supported by some HP-HIL devices to give an audio or visual response to the user, perhaps indicating that the system is ready for some type of input. A device specifies acceptance of these requests in the I/O Descriptor Byte in the Describe Record. These requests do not use *arg*.
- HILP** Prompt (General Purpose)
- This request is intended as a general purpose stimulus to the user. Devices accepting this request indicate so in the I/O Descriptor Byte in the Describe Record. This request does not use *arg*.
- HILA1..HILA7** Acknowledge 1 through Acknowledge 7
- These seven requests are intended to provide an audio or visual response to the user, generally to acknowledge a user's input. The I/O Descriptor Byte in the Describe Record indicates whether an HP-HIL device implements this request. These requests do not use *arg*.
- HILA** Acknowledge (General Purpose)
- The Acknowledge request is intended to provide an audio or visual response to the user. Devices accepting this request indicate so in the I/O Descriptor Byte in the Describe Record. This request does not use *arg*.

ERRORS

- [EBUSY] The specified HP-HIL device is already opened.

[EFAULT]	A bad address was detected while attempting to use an argument to a system call.
[EINTR]	A signal interrupted an <i>open(2)</i> , <i>read(2)</i> , or <i>ioctl(2)</i> system call.
[EINVAL]	An invalid parameter was detected by <i>ioctl(2)</i> .
[ENXIO]	No device is present at the specified address; see the <i>WARNINGS</i> section.
[EIO]	A hardware or software error occurred while executing an <i>ioctl(2)</i> system call.
[ENODEV]	<i>write(2)</i> is not implemented for HP-HIL devices.

WARNINGS

An [ENXIO] error is returned by *open(2)* and *ioctl(2)* if any attempt is made to access a device while **hil** is reconfiguring the link during power-failure recovery.

hil cannot detect whether or not a device executed an *ioctl(2)* request.

HP-HIL devices have no status bit available to indicate whether they support the HILER1, HILER2, or HILDKR requests.

AUTHOR

hil was developed by HP.

FILES

/dev/hil [1-7]
/dev/hil_* [1-7]

SEE ALSO

close(2), *errno(2)*, *fcntl(2)*, *ioctl(2)*, *open(2)*, *read(2)*, *select(2)*, *signal(5)*, *hilkbd(7)*, *termio(7)*.

For detailed information about HP-HIL hardware and software in general, see the *HP-HIL Technical Reference Manual*.

NAME

hilkbd - HP-HIL mapped keyboard driver

DESCRIPTION

HP-HIL, the Hewlett-Packard Human Interface Link, is the Hewlett-Packard standard for interfacing a personal computer, terminal, or workstation to its input devices. **hilkbd** supplies input from all mapped keyboards on a specified HP-HIL link.

hilkbd returns mapped keycodes, not ASCII characters. "Raw" keycodes are the individual key down-strokes and upstrokes, and are different for each type of keyboard. **hilkbd** maps the raw input into the keycodes and protocol expected by the HP-UX, Pascal Workstation, and BASIC/UX operating systems. The **hil** driver can usurp a keyboard from **hilkbd** by changing it from mapped mode to raw mode.

System Calls

open() gives exclusive access to the keyboard. If there is an ITE (internal terminal emulator) associated with the keyboard, the ITE loses input from the keyboard until the keyboard device is closed. Any previous queued input for the keyboard device is flushed from the input queue.

close() returns control of the keyboard to the ITE, if present. Any unread input is discarded at that time.

read() returns data from the keyboard in time-stamped packets:

```
unsigned char time_stamp [4];
unsigned char status;
unsigned char data;
```

time_stamp, when repacked into an integer data type of four or more bytes, specifies the time since an arbitrary point in the past (for example, system start-up time). This point does not change between packets, but time during a power failure may or may not be counted. The time is in units of tens of milliseconds.

The *status* byte encodes the state of the keyboard **Shift** and **Ctrl** keys:

```
0x8X  shift and control
0x9X  control only
0xAC  shift only
0xBX  no shift or control
```

The *data* byte contains the actual keystroke.

If the file status flag **O_NDELAY** is set, **read()** returns **0** instead of blocking, when no data is available. The **read()** system call on an HP-HIL keyboard is considered "slow"; that is, it can be interrupted by caught signals (see *signal(2)*).

write() is not supported by **hilkbd**.

select() can be used to poll for input to read from **hilkbd** devices. **select()** for write or for exceptional conditions always returns a false indication in the bit masks.

ioctl() is used to perform special operations on the device. **ioctl()** system calls have the form:

```
int ioctl(int fildes, int request, char *arg);
```

The following **hilkbd** request codes are defined in **<sys/hilioctl.h>**:

KBD_READ_CONFIG

Read the configuration code.

This request returns a one-byte configuration code in the **char** variable to which *arg* points. This contains a field, defined by **KBD_IDCODE_MASK**, which specifies the keyboard identification code. The possible values of this field are defined in the header file, and this identification code affects interpretation of the language code. All other fields in the configuration code are currently undefined.

KBD_READ_LANGUAGE

Read the language code.

This request returns a one-byte language code, as read from the keyboard, in the **char** variable to which *arg* points. If there is more than one keyboard, the language is taken from the first keyboard on the link. Interpretation of the language code is affected by the

keyboard identification field within the configuration code.

KBD_STATUS Read the keyboard status register.

This request returns a one-byte value containing bit flags specifying the state of the shift and control keys in the **char** variable to which *arg* points:

KBD_STAT_LEFTSHIFT	The left shift key is up
KBD_STAT_RIGHTSHIFT	The right shift key is up
KBD_STAT_SHIFT	Both shift keys are up
KBD_STAT_CTRL	The control key is up

Other bits are undefined.

KBD_REPEAT_RATE

Set the keyboard auto-repeat rate.

The one-byte value to which *arg* points is the negative of the repeat period, in tens of milliseconds. The repeat rate is the reciprocal of the repeat period. A parameter of zero disables auto-repeat.

KBD_REPEAT_DELAY

Set the keyboard auto-repeat delay.

The one-byte value to which *arg* points is the negative of the repeat delay, in tens of milliseconds.

KBD_BEEP

Cause an audible beep.

The one-byte value to which *arg* points specifies the volume of the beep, within the range 0 through **KBD_MAXVOLUME**. Implementations with fewer than **KBD_MAXVOLUME** discrete levels of volume will scale the parameter into the smaller range.

ERRORS

[EINVAL]	An invalid parameter was detected by ioctl() .
[EINTR]	A signal was caught during a read() system call.
[ENXIO]	No keyboard is present on the HP-HIL link specified by the minor number.
[ENODEV]	An attempt was made to use write() using hilkbd .
[EBUSY]	The device is already open.

AUTHOR

hilkbd was developed by the Hewlett-Packard Company.

FILES

/dev/hilkbd*

SEE ALSO

mknod(1M), select(2), signal(2), hil(7), termio(7).

NAME

inet - Internet protocol family

SYNOPSIS

```
#include <sys/types.h>
#include <netinet/in.h>
```

DESCRIPTION

The internet protocol family is a collection of protocols layered on top of the **Internet Protocol** (IP) network layer, which utilizes the internet address format. The internet family supports the **SOCK_STREAM** and **SOCK_DGRAM** socket types.

Addressing

Internet addresses are four byte entities. The include file `<netinet/in.h>` defines this address as the structure **struct in_addr**.

Sockets bound to the internet protocol family utilize an addressing structure called **struct sockaddr_in**. Pointers to this structure can be used in system calls wherever they ask for a pointer to a **struct sockaddr**.

There are three fields of interest within this structure. The first is **sin_family**, which must be set to **AF_INET**. The next is **sin_port**, which specifies the port number to be used on the desired host. The third is **sin_addr**, which is of type **struct in_addr**, and specifies the address of the desired host.

Protocols

The internet protocol family is comprised of the IP network protocol, Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). TCP is used to support the **SOCK_STREAM** socket type while UDP is used to support the **SOCK_DGRAM** socket type. The ICMP message protocol and IP network protocol are not directly accessible.

The local port address is selected from independent domains for TCP and UDP sockets. This means that creating a TCP socket and binding it to local port number 10000, for example, does not interfere with creating a UDP socket and also binding it to local port number 10000 at the same time.

Port numbers in the range 1-1023 inclusive are reserved for use by the super-user only. Attempts to bind to port numbers in this range by non-super-users fail and result in an error returned.

AUTHOR

inet was developed by the University of California, Berkeley.

SEE ALSO

TCP(7P), UDP(7P).

(OBSOLETE)

NAME

iomap - physical I/O address mapping

SYNOPSIS**#include <sys/iomap.h>****DESCRIPTION**

The **iomap** mechanism allows the mapping (thus direct access) of physical I/O addresses into the user process address space. For PA-RISC machines, the physical I/O address space begins at **0xf0000000** and extends to **0xffffffff**.

The special (device) files for **iomap** devices are character special files using the dynamic major number allocation scheme.

The minor number for **iomap** devices is of the form:

0xAAAASM

The physical I/O address is formed by prefixing 0xAAAA with 0xF, and by appending 0x000 (this forces the I/O address to be page-aligned). The size of the region to be mapped is given by the expression $M*(2^S)$ 4K pages. For example, the minor number for a device starting at **0xf4000000** that occupies 64MB is **0x4000e1**.

The **iomap** driver must be explicitly added to the **/stand/system** file, the kernel rebuilt, and the system subsequently rebooted prior to first using **iomap**.

I/O space is always mapped with both read and write access rights, regardless of the actual permissions on the device special file.

Multiple processes can have concurrently a single **iomap** device opened and mapped. It is the responsibility of the processes to synchronize their access.

Successive calls to **iomap** to map the same I/O space must be identical to the first mapping. Identical mappings have the same address and size.

Note that a process can additionally share I/O space (mapped by **iomap**) with a kernel driver. However, this is only possible if the driver maps in the I/O space with user read/write access rights using the appropriate driver I/O mapping services. Any I/O space mapped by drivers with kernel read/write access rights cannot be concurrently mapped by processes using **iomap**.

No **read()** or **write()** system calls are supported by the **iomap** driver.

The **ioctl()** function is used to control the **iomap** device. The following **ioctl()** requests are defined in **<iomap.h>**:

IOMAPMAP Map the **iomap** device into user address space at the location specified by the pointer to which the **(void **)** third argument to **ioctl()** points. If the argument points to a variable containing a null pointer, the system selects an appropriate address. **ioctl()** then returns the user address where the device was mapped, storing it at the address pointed to by the third argument (see *EXAMPLES* below). Multiple processes can concurrently have the same **iomap** device mapped.

IOMAPUNMAP Unmap the **iomap** device from the user address space.

close() shuts down the file descriptor associated with the **iomap** device. If the close is for the last system wide open on the device, the **iomap** device is also unmapped from the user address space; otherwise it is left mapped into the user address space (see **IOMAPUNMAP** above).

WARNINGS

Be extremely careful when creating and using **iomap** devices. Inappropriate accesses to I/O devices or RAM can result in a system crash.

ERRORS

[EINVAL] The address field was out of range, or the **ioctl** request was invalid.

[ENOMEM] Not enough memory could be allocated for the mapping.

[EBUSY] Device was already mapped and this mapping was not identical to the initial mapping (same address, size and access rights).

[ENODEV] Read and write calls are unsupported.

(OBSOLETED)

- [ENXIO] No such device at the address specified by the minor number.
- [ENOSPC] Required resources for mapping could not be allocated.
- [ENOTTY] Inappropriate **ioctl** request for this device type; *fildev* is not a file descriptor for an **iomap** device file.

EXAMPLES

Consider the following code fragment:

```
#include <sys/iomap.h>
...
int fildev;
void *addr;
...
addr = REQUESTED_ADDRESS;
(void) ioctl(fildev, IOMAPMAP, &addr);
(void) printf("actual address = 0x%x\n", addr);
```

where **fildev** is an open file descriptor for the device special file and **REQUESTED_ADDRESS** is the address originally requested by the program.

If **addr** is a null pointer, the system selects a suitable address then returns the selected address in *addr*.

If the value in *addr* is not a null pointer, it is used as a specified address for allocating memory. If the specified address cannot be used, an error is returned (see *ERRORS*).

SEE ALSO

mkknod(1M).

NAME

IP - Internet Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_DGRAM, 0);
```

DESCRIPTION

IP is the network-layer protocol used by the Internet protocol family. It encapsulates TCP and UDP messages into datagrams to be transmitted by the network interface. Normally, applications do not need to interface directly to IP. However, certain multicast socket options are controlled by passing options to the **IPPROTO_IP** protocol level through a UDP socket, and IP Type of Service is controlled by passing an option to the **IPPROTO_IP** protocol level through either a TCP or UDP socket. (See the *getsockopt(2)* manual page.)

The following socket options are defined in the include file **<netinet/in.h>**. The type of the variable pointed to by the *optval* parameter is indicated in parentheses. The data types **struct ip_mreq** and **struct in_addr** are defined in **<netinet/in.h>**.

- IP_TOS** (**unsigned int**) Sets the IP Type of Service. Allowable values for *optval* are 4 for high reliability, 8 for high throughput, and 16 for low delay. Other values will not return an error, but may have unpredictable results. Default: zero.
- IP_ADD_MEMBERSHIP** (**struct ip_mreq**) Requests that the system join a multicast group.
- IP_DROP_MEMBERSHIP** (**struct ip_mreq**) Allows the system to leave a multicast group.
- IP_MULTICAST_IF** (**struct in_addr**) Specifies a network interface other than the default to be used when sending multicast datagrams through this socket. Default: multicast datagrams are sent from the interface associated with the specific multicast group, with the default multicast route or with the default route.
- IP_MULTICAST_LOOP** (**unsigned char**; boolean) Enables or disables loopback in the IP layer for multicast datagrams sent through this socket. The value of the variable pointed to by *optval* is zero (disable) or non-zero (enable). This option is provided for compatibility only. Normally, multicast datagrams are always looped back if the system has joined the group. See *DEPENDENCIES* below. Default: enabled.
- IP_MULTICAST_TTL** (**unsigned char**) Specifies the time-to-live value for multicast datagrams sent through this socket. The value of the variable pointed to by *optval* can be zero through 255. Default: one.

IP_ADD_MEMBERSHIP requests that the system join a multicast group on the specified interface. For example:

```
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr = net_addr("224.1.2.3");
mreq.imr_interface.s_addr = INADDR_ANY;
setsockopt(s, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq));
```

A system must join a group on an interface in order to receive multicast datagrams sent on the network to which that interface connects. If **imr_interface** is set to **INADDR_ANY**, the system joins the specified group on the interface that datagrams for that group would be sent from, based the routing configuration. Otherwise, **imr_interface** should be the IP address of a local interface. An application can join up to **IP_MAX_MEMBERSHIPS** multicast groups on each socket. **IP_MAX_MEMBERSHIPS** is defined in **<netinet/in.h>**. However, each network interface may impose a smaller system-wide limit because of interface resource limitations and because the system uses some link-layer multicast addresses.

The application must also bind to the destination port number in order to receive datagrams that are sent to that port number. If the application binds to the address **INADDR_ANY**, it may receive all datagrams that are sent to the port number. If the application binds to a multicast group address, it may receive only datagrams sent to that group and port number. It is not necessary to join a multicast group in order to send datagrams to it.

IP_DROP_MEMBERSHIP allows the system to leave a multicast group. For example:

```
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr = net_addr("224.1.2.3");
mreq.imr_interface.s_addr = INADDR_ANY;
setsockopt(s, IPPROTO_IP, IP_DROP_MEMBERSHIP, &mreq, sizeof(mreq));
```

The system remains a member of the multicast group until the last socket that joined the group is closed or has dropped membership in the group.

IP_MULTICAST_IF specifies a local network interface to be used when sending multicast datagrams through this socket. For example:

```
#include <arpa/inet.h>
struct in_addr addr;
addr.s_addr = inet_addr("192.1.2.3");
setsockopt(s, IPPROTO_IP, IP_MULTICAST_IF, &addr, sizeof(addr));
```

Normally, applications do not need to specify the interface. By default, multicast datagrams are sent from the interface specified by the routing configuration, namely the interface associated with the specific multicast group, with the default multicast route or with the default route. If **addr** is set to the address **INADDR_ANY**, the default interface is selected. Otherwise, **addr** should be the IP address of a local interface.

IP_MULTICAST_LOOP enables or disables loopback for multicast datagrams sent through this socket. For example:

```
unsigned char loop = 1;
setsockopt(s, IPPROTO_IP, IP_MULTICAST_LOOP, &loop, sizeof(loop));
```

Note that the type of the *optval* parameter is **unsigned char** instead of **int**, which is common for boolean socket options. This option is provided for compatibility only. Normally, if a multicast datagram is sent to a group that the system has joined, a copy of the datagram is always looped back and delivered to any applications that are bound to the destination port. See *DEPENDENCIES* below.

IP_MULTICAST_TTL controls the scope a multicast by setting the time-to-live value for multicast datagrams sent through this socket. For example:

```
unsigned char ttl = 64;
setsockopt(s, IPPROTO_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl));
```

Note that the type of *optval* parameter is **unsigned char** instead **int**, which is common for socket options. By default, the time-to-live field (TTL) is one, which limits the multicast to the local network. If the TTL is zero, the multicast is limited to the local system (loopback). If the TTL is two, the multicast can be forwarded through at most one gateway; and so forth. Multicast datagrams can be forwarded to other networks only if there are special multicast routers on the local and intermediate networks.

DEPENDENCIES

The behavior of **IP_MULTICAST_LOOP** depends on the network driver and interface card. Normally, loopback cannot be disabled, even if **IP_MULTICAST_LOOP** is set to zero, because it occurs in the driver or in the network interface. However, if the outbound interface is **lo0** (127.0.0.1), or if **IP_MULTICAST_TTL** is set to zero, setting **IP_MULTICAST_LOOP** to zero will disable loopback for multicast datagrams sent through the socket.

ERRORS

One of the following errors may be returned if a call to **setsockopt()** or **getsockopt()** fails.

[EADDRINUSE]	The specified multicast group has been joined already on socket.
[EADDRNOTAVAIL]	The specified IP address is not a local interface address; or there is no route for the specified multicast address; or the specified multicast group has not been joined.
[EINVAL]	The parameter <i>level</i> is not IPPROTO_IP ; or <i>optval</i> is the NULL address; or the specified multicast address is not valid.
[ENOBUFS]	Insufficient memory is available for internal system data structures.
[ENOPROTOOPT]	The parameter <i>optname</i> is not a valid socket option for the IPPROTO_IP level.

[EOPNOTSUPP] The socket type is not **SOCK_DGRAM**.
[ETOOMANYREFS] An attempt to join more than **IP_MAX_MEMBERSHIPS** multicast groups on
a socket.

AUTHOR

The socket interfaces to IP were developed by the University of California, Berkeley. Multicast extensions were developed by the Stanford University.

SEE ALSO

bind(2), getsockopt(2), recv(2), send(2), socket(2), inet(7F).

NAME

IPv6, ipv6, ip6 - Internet Protocol Version 6

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET6, SOCK_DGRAM, 0);
s = socket(AF_INET6, SOCK_STREAM, 0);
```

DESCRIPTION

IPv6 is the next generation network-layer protocol designed to be the successor to the current Internet Protocol version 4 (IPv4). It provides the packet delivery service for TCP, UDP and ICMPv6.

IPv6 has significant advantages over IPv4 in terms of increased address space, simplified header format, integrated QoS support and mandatory security. IPv6 also allows optional internet-layer information to be encoded in separate headers called extension headers which are placed between the IPv6 header and upper layer headers. Extension headers currently supported are hop-by-hop option header, destination option header, fragment header and routing (type 0) header. An IPv6 packet may carry zero, one, or more extension headers, each identified by the next header field of the preceding header.

IPv6 has extended the address size from 32 bits to 128 bits and they are textually represented in hex-colon notation as **x:x:x:x:x:x:x:x**, where the **x**'s are the hexadecimal values of the eight 16-bit pieces of the address. For example **fedc:83ff:fef6:417a:210:83ff:fef6:3dc0**.

IPv6 has three types of addresses: **unicast**, **anycast**, and **multicast**.

- An **unicast address** is an identifier for a single interface. A packet sent to an unicast address is delivered to the interface identified by that address.
- An **anycast address** is an identifier for a set of interfaces. A packet sent to an anycast address is delivered to one of the interfaces identified by that address.
- A **multicast address** is an identifier for a set of interfaces. A packet sent to a multicast address is delivered to all interfaces identified by that address.

There are no broadcast addresses in IPv6, their function is superseded by multicast addresses.

Every IPv6 address has a **scope** associated with it. A scope is a topological span within which the address may be used as a unique identifier for an interface or set of interfaces.

An unicast address has three defined scopes: link-local, site-local and global.

- Link-local address uniquely identifies interfaces within a single link and it has a fixed prefix of **fe80::/10**. For example, **fe80::210:84c0:ef6f:cd30**.
- Site-local address uniquely identifies interfaces within a single site only and it has a fixed prefix of **fec0::/10**. For example, **fec0::210:84c0:ef6f:cd30**.
- Global address uniquely identifies interfaces anywhere in the internet.

There are 2 special unicast addresses which hold an embedded IPv4 address in the low order 32-bits.

- The first type is termed as IPv4-compatible IPv6 address and is of the form **0:0:0:0:0:0:d.d.d.d**. This type of address is used by dual stack (IPv4/IPv6) nodes to perform automatic IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined from the IPv4 address embedded in the IPv4-compatible destination address of the IPv6 packet being tunneled.
- The second type is termed as IPv4-mapped IPv6 address and is of the form **0:0:0:0:0:ffff:d.d.d.d**. This address facilitates IPv6 applications to interoperate with IPv4 applications. Applications can automatically generate this address using **getaddrinfo()** (see **getaddrinfo(3N)**) when the specified host has only IPv4 address.

IPv6 Socket Options

New socket options are defined for IPv6 to send and receive extension headers and to exchange other optional information between the kernel and application. The options are supported at the **IPPROTO_IPV6** protocol level. The type of the variable pointed to by the *optval* parameter is indicated in parenthesis.

IPV6_UNICAST_HOPS (**integer**) Set or get the hop limit used in outgoing unicast packets. When this option is set using **setsockopt()** (see **setsockopt(2)**), the new

option value specified is used as the hop limit for all subsequent unicast packets sent via that socket. Valid values are in the range 0-255 (both inclusive) and the default value is 64. For example,

```
int hoplimit = 50;
setsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS,
           &hoplimit, sizeof(hoplimit));
```

This option can be used with `getsockopt()` (see *getsockopt(2)*) to determine the hop limit value the system will use for subsequent unicast packets sent via that socket.

IPV6_MULTICAST_HOPS

(**integer**) Set or get the hop limit used in outgoing multicast packets. When this option is set, the new option value specified is used as the hop limit for all subsequent multicast packets sent via that socket. Valid values are in the range 0-255 (both inclusive) and the default value is 1.

IPV6_MULTICAST_IF (**integer**) Sets the interface to use for outgoing multicast packets. The option value is the index of the selected outgoing interface. For example,

```
unsigned int index;
index = if_nametoindex("lan0");
setsockopt(s, IPPROTO_IPV6, IPV6_MULTICAST_IF,
           &index, sizeof(index));
```

IPV6_MULTICAST_LOOP

(**boolean**) Enables or disables loopback in the IP layer for multicast datagrams sent through this socket. The value of the variable pointed to by *optval* is zero (disable) or non-zero (enable). Default: enabled.

IPV6_JOIN_GROUP

(**struct ipv6_mreq**) Join a multicast group on a specified local interface. The IPv6 multicast address of the group to join and the index of the interface on which to join should be specified using **struct ipv6_mreq** which is defined in `<netinet/in6.h>` as:

```
struct ipv6_mreq {
    struct in6_addr ipv6mr_multiaddr;
                               /* IPv6 multicast addr */
    unsigned int   ipv6mr_interface;
                               /* interface index */
};
```

If the interface index is specified as 0 then the default multicast interface is used.

IPV6_LEAVE_GROUP

(**struct ipv6_mreq**) Leave a multicast group on a specified local interface. The IPv6 multicast address of the group to leave and the interface index should be specified using **struct ipv6_mreq**. The interface index should match the index used while joining the group. Set index to 0, to specify default interface.

IPV6_CHECKSUM

(**integer**) When this option is set, kernel computes the checksum for outbound packets and verifies checksum on inbound packets. The option value is the byte offset of the checksum location in the user data. This option is not valid for **IPPROTO_ICMPV6** since checksum computation is mandatory for **IPPROTO_ICMPV6**. The default value is -1 (checksums not computed nor verified for protocols other than **IPPROTO_ICMPV6**).

IPV6_RECVPKTINFO

(**boolean**) When this option is enabled, **PKTINFO** (destination IPv6 address and the arriving interface index) is returned as ancillary data by `recvmsg()`. (See *recvmsg(2)*). The information is returned in **struct**

in6_pktinfo structure and it is defined in `<netinet/in6.h>` as:

```
struct in6_pktinfo {
    struct in6_addr  ipi6_addr;
    uint32_t        ipi6_ifindex;
};
```

By default this option is disabled.

IPV6_RECVMHOPLIMIT (**boolean**) When this option is enabled, inbound packet's hoplimit is returned as ancillary data by `recvmsg()`. For example,

```
int on = 1;
setsockopt(s, IPPROTO_IPV6, IPV6_RECVMHOPLIMIT,
           &on, sizeof(on));
```

By default this option is disabled.

IPV6_RECVDSTOPTS (**boolean**) When this option is enabled, the inbound packet's destination options (when present) is returned as ancillary data by `recvmsg()`. By default this option is disabled.

IPV6_RECVHOPOPTS (**boolean**) When this option is enabled, the inbound packet's hop-by-hop options (when present) is returned as ancillary data by `recvmsg()`. By default this option is disabled.

IPV6_RECVRTHDR (**integer; boolean**) When this option is enabled, the inbound packet's routing options (when present) is returned as ancillary data by `recvmsg()`. By default this option is disabled.

IPV6_RECVRTHDRDSTOPTS (**integer; boolean**) When this option is enabled, the inbound packet's destination options appearing before a routing header (when present) is returned as ancillary data by `recvmsg()`. By default this option is disabled.

The next seven socket options can be used with both `setsockopt()` and as option name in ancillary data to `sendmsg()`. (See `sendmsg(2)`)

IPV6_PKTINFO (**struct in6_pktinfo**) Used to set the source address and interface index for outgoing packets.

IPV6_HOPLIMIT (**integer**) Used to set the hop limit for outbound packets. This hop limit is valid for only a single output operation. To set hop limit for all unicast or multicast IPv6 packets use **IPV6_UNICAST_HOPS** or **IPV6_MULTICAST_HOPS** options respectively.

IPV6_NEXTHOP (**struct sockaddr_in6**) Used to set the next hop address. The node identified by this address must be a neighbor of the sending host. When this address is the same as the destination IPv6 address then this is equivalent to **SO_DONTROUTE** socket option.

IPV6_RTHDR (**variable length**) Used to specify the routing header for outgoing packets. Only Type 0 routing header is currently supported.

IPV6_DSTOPTS (**variable length**) Used to specify one or more destination options to be sent in subsequent IPv6 packets.

IPV6_HOPOPTS (**variable length**) Used to specify one or more hop-by-hop options to be sent in subsequent IPv6 packets.

IPV6_RTHDRDSTOPTS (**variable length**) Used to specify one or more destination options preceding a routing header. This option will be silently ignored when sending packets unless a routing header is also specified.

IPv6 uses the enhanced version of ICMP called ICMPv6 to report errors encountered in processing packets and for diagnostic purposes (like ping). ICMPv6 is an integral part of IPv6 and has a next header value of 58.

All the options and the associated structures are defined in `<netinet/in6.h>`, applications are not required to include this header file explicitly, it is automatically included by `<netinet/in.h>`.

ERRORS

One of the following errors may be returned when a socket operation fails.

[EADDRINUSE]	The specified multicast group has been joined already.
[EADDRNOTAVAIL]	The specified IPv6 address is not a local interface address or there is no route for the specified multicast address or the specified multicast group has not been joined.
[EINVAL]	The parameter 'level' is not IPPROTO_IPV6 , or <i>optval</i> is the NULL address, or the specified multicast address is not valid, or the specified hop limit is not in the range $0 \leq x \leq 255$.
[ENOBUFS]	Insufficient memory is available for internal system data structures.
[ENOPROTOOPT]	The parameter <i>optname</i> is not a valid socket option for the IPPROTO_IPV6 level.

AUTHOR

The socket interfaces to IP were developed by the University of California, Berkeley.

SEE ALSO

`bind(2)`, `getsockopt(2)`, `recv(2)`, `send(2)`, `socket(2)`, `inet6_opt_init(3N)`, `inet6_rth_space(3N)`, `inet(7F)`, `ndp(7P)`.

RFC 2460 *Internet Protocol Version 6*.

RFC 2553 *Basic Socket Interface Extensions for IPv6*.

RFC 2292 *Advanced Socket Interface Extensions for IPv6*.

NAME

kmem - perform I/O on kernel memory, based on symbol name

SYNOPSIS

```
#include <sys/ksym.h>

int ioctl(
    int kmemfd,
    int command,
    void *rks
);
```

DESCRIPTION

When used with a valid file descriptor for `/dev/kmem` (*kmemfd*), `ioctl` can be used to manipulate kernel memory. The specifics of this manipulation depend on the *command* given as follows:

MIOC_READKSYM Read *mirk_buflen* bytes of kernel memory starting at the address for *mirk_symname* into *mirk_buf*. *rks* is a pointer to a **mioc_rksym** structure, defined below.

MIOC_IREADKSYM Indirect read. Read `sizeof(void *)` bytes of kernel memory starting at the address for *mirk_symname* and use that as the address from which to read *mirk_buflen* bytes of kernel memory into *mirk_buf*. *rks* is a pointer to a **mioc_rksym** structure.

MIOC_WRITEKSYM Write *mirk_buflen* bytes from *mirk_buf* into kernel memory starting at the address for *mirk_symname*. *rks* is a pointer to a **mioc_rksym** structure.

MIOC_IWRITEKSYM Indirect write. Read `sizeof(void *)` bytes of kernel memory starting at the address for *mirk_symname* and use that as the kernel memory address into which *mirk_buflen* bytes from *mirk_buf* are written. *rks* is a pointer to a **mioc_rksym** structure.

MIOC_LOCKSYM Increase the hold count by one for the dynamically loaded module whose name is given by *rks*, a pointer to a character string, thereby preventing its unloading.

MIOC_UNLOCKSYM Decrease the hold count by one for the dynamically loaded module whose name is given by *rks*, a pointer to a character string. If the count is thereby reduced to 0, the module becomes a candidate for unloading.

The **struct mioc_rksym** definition is:

```
struct mioc_rksym {
    char * mirk_modname; /* limit search for symname
                        to module modname; if NULL
                        use standard search order */
    char * mirk_symname; /* name of symbol whose address
                        is the basis for this
                        operation */
    void * mirk_buf; /* buffer into/from which
                    read/write takes place */
    size_t mirk_buflen; /* length (in bytes) of desired
                        operation */
};
```

RETURN VALUE

`ioctl` returns one of the following values:

- 0 Successful completion.
- 1 Failure. **errno** is set to indicate the error.

ERRORS

In addition to the values described in `ioctl(2)`, the **kmem ioctl** also sets **errno** to one of the following values if the corresponding condition is detected.

[EINVAL] *modname* does not represent a currently loaded module or this is an **MIOC_UNLOCKSYM** and the hold count is already 0.

[ENXIO] *kmemfd* open on wrong minor device (i.e., not **/dev/kmem**).

[EBADF] *kmemfd* open for reading and this is an **MIOC_WRITEKSYM**.

[ENOMATCH] *symname* not found.

[ENAMETOOLONG]

modname is greater than **MODMAXNAMELEN** characters long, or *symname* is greater than **MAXSYMNMLEN** characters long.

SEE ALSO

getksym(2), ioctl(2), ioctl(5).

NAME

lan - network I/O card access information

DESCRIPTION

This manual entry gives a brief description on how to access the LAN device driver at Layer 2 (Data Link Layer) of the OSI architecture. The LAN device driver controls the various LAN interface cards (e.g. Ethernet/IEEE 802.3, FDDI, Token Ring) at Layer 1 (Physical Layer).

The Data Link Provider Interface (DLPI) is the supported method for accessing the LAN device driver at Layer 2. DLPI is intended for use by knowledgeable network users only. Refer to the *DLPI Programmer's Guide* for complete programming details.

There are HP and non-HP drivers and interface cards which will provide their own DLPI module. These types of DLPI are referred to as "native" DLPI.

Overview

The Physical Point of Attachment (PPA) is a numerical value that uniquely identifies a particular device. The PPA value can be obtained from the **nwmgr** and **lanscan** commands. The "ClassInstance" identifier in the **nwmgr** output is the concatenation of the driver class (lan) and the PPA number. The "NamePPA" identifier in the **lanscan** output is a concatenation of the interface name and the PPA number. The **card instance** value for a lan device is equivalent to the PPA number for that device.

A single hardware device may have multiple "NamePPA" identifiers, which indicates multiple encapsulation methods supported for to the device. For Ethernet/IEEE 802.3 links, the "Name" **lan** is used to designate Ethernet encapsulation, and **snap** for IEEE 802.3 encapsulation. For other links (FDDI, Token Ring), only the **lan** encapsulation designation is used.

Methods of transfer over the DLPI interface through the lan devices include "raw", "connectionless", and "connection-oriented" data transfers.

WARNINGS

The **lanadmin**, **lanscan** and **linkloop** commands are deprecated. These commands will be removed in a future HP-UX release. HP recommends the use of replacement command **numgr(1M)** to perform all network interface-related tasks.

AUTHOR

lan was developed by HP.

SEE ALSO

lanscan(1M), **lanadmin(1M)**, **linkloop(1M)**, **nwmgr(1M)**.

DLPI Programmer's Guide, 1995, Hewlett-Packard

The Ethernet, A LAN: Data Link Layer and Physical Specification, Version 2.0, November 1982, Digital Equipment Corporation, Intel Corporation, Xerox Corporation

CSMA/CD Access Method and Physical Layer Specification, 1996, Institute of Electrical and Electronic Engineers

Demand-Priority Access Method, Physical Layer & Repeater Specifications, 1996, Institute of Electrical and Electronic Engineers

Fiber Distributed Data Interface (FDDI) Physical Layer Medium Dependent (PMD), 1995, ANSI

Token Ring Access Method and Physical Layer Specification, 1995, Institute of Electrical and Electronic Engineers

802.3u Media Access Control Parameters, Physical Layer, Medium Attachment Units, and Repeater for 100 Mb/s Operation, Type 100BASE-T, 1995, Institute of Electrical and Electronic Engineers

NAME

ldterm - standard STREAMS terminal line discipline module

SYNOPSIS

```
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/termios.h>
#include <sys/bsdttty.h>
#include <sys/ttold.h>
#include <sys/strtio.h>
#include <sys/eucioctl.h>

int ioctl( fd, I_PUSH, "ldterm");
```

DESCRIPTION

ldterm is a STREAMS module that supplies the line discipline for streams-based terminal or pseudo-terminal device drivers. This module provides most of the functions of the general terminal interface described in *termio(7)*. However, it does not perform the low-level device control functions specified by the **c_cflag** word defined by the POSIX **termios** structure or the System V **termio** structure (defined in **termios.h** and **termio.h**, respectively). Also, some operations require the cooperation of the modules and drivers pushed below the **ldterm** module in a tty or pty (slave) stream. This man page only covers **ldterm** specific interface here and refers to the readers to *termio(7)* for the detail terminal interface.

Internally, the **ldterm** module uses the Extended UNIX Code (EUC) character encoding scheme. This encoding scheme enables the **ldterm** module to process multibyte characters as well as simple 8-bit characters. It correctly handles backspacing, word erasing, and tab expansion for multibyte EUC characters.

The **ldterm** module provides standard terminal operation consistent with the behavior specified by POSIX 1003.1 and System V Interface Definition (SVID) Third Edition. It also provides compatibility with the behavior of the BSD 4.3 line discipline. Notice that on other STREAMS systems, the BSD 4.3 compatibility feature is usually provided by a separate STREAMS module called **ttcompat**. Hence, applications on HP-UX need not push **ttcompat** on top of **ldterm** to get BSD 4.3 compatibility. In fact, the **ttcompat** module is not provided on the HP-UX system at all.

The **ldterm** module normally sits above either a STREAMS tty driver or a STREAMS pty slave driver. The user issues an **STREAMS I_PUSH ioctl(2)** system call to push **ldterm** onto the stream once the STREAMS tty or STREAMS pty slave device is opened.

STREAMS Messages

The **ldterm** module processes various types of STREAMS messages. The line discipline will act on any of the following message types. Any others that the module receives, however, are passed onto the next module on the stream.

Read-side Behavior

ldterm processes the following STREAMS messages on its input stream:

M_FLUSH

If **FLUSHR** is set, the read put routine flushes the read queue, discards characters in the input message buffers, and discards any partially buffered multibyte EUC characters. Then, it forwards the message upstream.

M_BREAK

The read put routine processes the message according to POSIX rules for processing **BREAK** events, parity errors, and framing errors and signal generation (see *termio(7)* for detail). If there is no data in the message, the message is assumed to represent an input **BREAK** event, which is represented by a framing error with a character value of 0 (zero). If there is data in the message, the data value is an integer that indicates the occurrence of an input **BREAK** event, or a character received with a parity or framing error. The low-order 8 bits of the data value is the byte that was read. If the **TTY_PE** flag is set in the higher-order bits of this integer, then a parity error was detected. If the **TTY_FE** flag is set in the higher-order bits of this integer, a framing error was detected.

After reading the data value, the read put routine discards the message.

M_DATA The read put routine processes the message according to the POSIX 1003.1 specification, using multibyte processing for backspacing, word erasing, and tab expansion as appropriate.

It generates echo characters and places them in the output buffer to be sent downstream to the write queue. While processing incoming data, it scans for **START** and **STOP** characters and sends **M_START**, **M_STOP** messages downstream to the write queue, if needed.

If the total number of buffered input characters is more than the high-water mark and **IXOFF** is set, the read put routine sends an **M_STOPI** message downstream. When the queue reduces its backlog below the low water mark, it sends an **M_STARTI** message downstream.

If the number of buffered input characters reaches **MAX_INPUT**, and the **IMAXBEL** flag is set, the read put routine discards new input characters and sends a **BEL** character (**Ctrl-G**) downstream. If **IMAXBEL** is not set, it flushes the input queue.

If the **ISIG** flag is set, the read put routine sends **M_PCSIG** messages upstream when the appropriate signal characters are encountered. Then it discards the characters.

If a character matching **c_cc[VDISCARD]** is encountered, and the **IEXTEN** flag is set, the read put routine sends an **M_FLUSH (FLUSHW)** message upstream to flush all write queues. The **M_FLUSH** message is reflected by the stream head and sent downstream through all the write queues.

If the character signifies the logical termination of input, the read put routine sends the currently buffered characters upstream to the stream head.

Logical termination of input depends on the state of the **ICANON** flag. If **ICANON** is set, the **ldterm** module is in canonical input mode. In that case, the read put routine logically terminates input at the end of a line of input. Canonical line termination characters are **NEWLINE**, **EOF**, **EOL**, and **EOL2**. If **ICANON** is clear, the **ldterm** discipline module is in noncanonical or raw input mode. In that case, the read put routine terminates input when at least **VMIN** bytes are present in the input message buffer or the timer specified by **VTIME** expires (see *termio(7)* for more details).

M_IOCACK

If the message acknowledges the POSIX **termios TCGETS** command, the read put routine copies the **c_cflag** and speeds information, which is sent by the console driver downstream, from the message into the internal POSIX **termios** structure. Then it copies the internal POSIX **termios** structure into the message.

If the message acknowledges one of the POSIX **termios** set commands (i.e. **TCSETS**, **TCSETSW**, and **TCSETSF**) the read put routine copies all of the data from the message into the internal POSIX **termios** structure.

After this processing is done, the read put routine determines if the I/O control command was originally a BSD 4.3 or System V I/O control command that was converted to a POSIX **termios** command by the write service routine. If so, it restores the original data so that the message acknowledges the original I/O control command. Then it forwards the message upstream.

M_CTL This message was sent by the driver to make special requests to **ldterm**. The structure of **M_CTL** messages is the same as that of **M_IOCTL** messages. The **M_CTL** message block points to a message buffer containing an **iocblk** data structure (defined in **<sys/stream.h>**). The **ioc_cmd** member of this structure contains a command, just as it does in an **M_IOCTL** message. The **b_cont** member of the **M_CTL** message block contains a pointer to an **M_DATA** message block, which contains data associated with the **M_CTL** message.

The read put routine processes **M_CTL** messages containing the following commands:

MC_NO_CANON

Turn off input processing normally performed on upstream **M_DATA** messages. This is for the use of modules or drivers that perform their own input processing such as pseudo-terminal (see *ptm(7)* and *pts(7)*) in **REMOTE** mode connected to a program that performs the input processing.

MC_DO_CANON

Turn on input processing normally performed on upstream **M_DATA** messages. This message is sent when the driver want **ldterm** to exit the **REMOTE** mode.

Write-side Behavior

ldterm processes the following STREAMS messages on its output stream. Messages not listed here are simply forwarded downstream.

M_FLUSH

The write put routine flushes the write queue and discards any buffered output data. Then, it forwards the message downstream.

M_DATA

The write service routine processes the data according to the POSIX 1003.1 specification output flags. It sends the processed characters downstream to the driver when the output queue fills up and all of the data is processed.

M_IOCTL

The write put routine validates the format of the **M_IOCTL** message and checks for known commands. If the message format is invalid, it turns the **M_IOCTL** message into an **M_IOCNAK** message, and returns the message upstream. If the I/O control command is not recognized, it forwards the **M_IOCTL** message downstream for processing by other modules.

The write put routine determines if the command is one that must be processed in the proper sequence relative to **M_DATA** messages. If so, it queues the **M_IOCTL** message to the write queue for later processing by the write service routine. Commands that require processing in sequence are:

TCSETSW, TCSETSF, TCSETAW, TCSETAF, TCSBRK

Otherwise, the module's write put routine processes the command immediately. Detailed descriptions of the preceding **ioctl** commands are provided in the *ioctl* Commands subsection, below.

M_READ

This message is sent by the stream head to notify downstream modules when an application has issued a read request and there is not enough data queued at the stream head to satisfy the request. The **M_READ** is sent downstream normally when **ldterm** is operating in non-canonical input mode. If **VTIME** is positive, the write put routine starts an input timer. When the timer expires, it sends all buffered input upstream. Then, it forwards the **M_READ** message downstream.

ioctl Commands

The **ldterm** module acts on two categories of **ioctl** commands:

- Primary terminal I/O control commands
- BSD 4.3 compatibility terminal I/O control commands

Detail descriptions on how to use these **ioctls** can be found on the *termio(7)* man page. **NOTE:** the **FIO[xyz]** **ioctls** documented on *termio(7)* are currently not supported on **ldterm**.

Primary Terminal I/O Control Commands

The **ldterm** module acts on the following primary terminal I/O commands:

TCSETS, TCSETSW, TCSETSF

When the **ldterm** module receives any of these commands in an **M_IOCTL** message, it forwards them downstream. When it receives the **M_IOCACK** message in the read queue, it copies the POSIX **termios** information from the message into the internal POSIX **termios** structure and forwards the message upstream. If a mode change requires options at the stream head to be changed, an **M_SETOPTS** message is sent upstream. If the **ICANON** flag is turned on or off, the read mode at the stream head is changed to message-nondiscard (**RMSGN**) with read notification on (**SO_MREADON**) or byte-stream mode (**RNORM**) with read notification off (**SO_MREADOFF**), respectively. If the **TOSTOP** flag is turned on or off, the **tostop** mode at the stream head is turned on (**SO_TOSTOP**) or off (**SO_TONSTOP**), respectively.

TCGETS The **ldterm** module forwards the **M_IOCTL** message downstream. When it receives the **M_IOCACK** message in the read queue, it copies the **CLOCAL** flags and speeds from the message into the internal POSIX **termios** structure. Then, it copies the entire structure into the **M_IOCACK** message and forwards the message upstream.

TCSETA, TCSETAW, TCSETAF

These commands set the old System V **termio** information. The **ldterm** module converts the message to a POSIX **termios** **M_IOCTL** message, then forwards the message with a corresponding POSIX **termios** command (i.e. **TCSETS, TCSETSW, TCSETSF**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TCGETA This command get the old System V **termio** information. The **ldterm** module converts the message to a POSIX **termios** **M_IOCTL** message, then forwards the message with the

TCGETS command. The original I/O control command and **M_IOCTL** message are stored to be used on **M_IOCTL**. When it receives the matching **M_IOCTL** message, the **ldterm** module processes it as for a **TCGETS** command, then converts the POSIX **termios** information into the System V **termio** information and replies.

TCSBRK The **ldterm** module forwards this command downstream to be handled by the driver so that the driver has a chance to drain the data before sending an **M_IOCTL** message upstream.

TCXONC This command controls the behavior of input/output flow control. If the argument is 0 and output is not already stopped, an **M_STOP** message is sent downstream. If the argument is 1 and the output is stopped, an **M_START** message is sent downstream. If the argument is 2 and input is not already stopped, an **M_STOPT** message is sent downstream. If the argument is 3 and input is stopped, an **M_STARTI** message is sent downstream.

TCFLSH This command flush the input or/and output streams. If the argument is 0, an **M_FLUSH** message with a flag byte of **FLUSHR** is sent downstream. This **M_FLUSH (FLUSHR)** message will be reflected back upstream by the driver to flush the entire input stream. If the argument is 1, an **M_FLUSH** message with a flag byte of **FLUSHW** is sent upstream. This **M_FLUSH (FLUSHW)** message will be reflected downstream by the stream head to flush the entire output stream.

TIOCSWINSZ

This command sets the window size variables. The argument of this command takes a pointer to a **winsize** structure. The **ldterm** module does not use the window size variable, but maintains it here for any needed replies to **TIOCGWINSZ** commands. The module forwards the message downstream.

TIOCGWINSZ

When the **ldterm** module receives this command, it returns the window size variable that was set by the last **TIOCSWINSZ** command. The argument of this command takes a pointer to a **winsize** structure.

EUC_WSET

This command sets the character widths and screen widths for the EUC character sets. The argument of this command takes a pointer to an **euclioc** structure which contains the information for setting the character widths and screen widths of the EUC character sets. After processing the command, **ldterm** forwards this message downstream to the next module.

EUC_WGET

This command returns the character widths and screen widths for the EUC character sets. This command takes a pointer to an **euclioc** structure via which the EUC character widths and screen widths information will be returned.

EUC_SET_HP15

This command put **ldterm** to the so called **HP15** mode which enable **ldterm** to recognize the **HP15_SJIS**, **HP15_BIG5**, **HP15_CCDC**, and **HP15_GB** character sets and process them in such a way that they behave like EUC characters. The argument for this command takes a pointer to an integer value which specify on of the above-mentioned four supported HP15 character sets. If the argument is set to **HP15_ASCII**, then **ldterm** will switch back to normal ASCII processing. **EUC_WSET** is mutually exclusive with **EUC_SET_HP15**.

EUC_GET_HP15

This command returns the current HP15 character that has been set via the **EUC_SET_HP15** command. This command takes a pointer to an integer via which the result is returned. If no previous **EUC_SET_HP15** has been issued, then it will return **HP15_ASCII**.

BSD 4.3 Compatible Terminal I/O Commands

The **ldterm** module acts on the following I/O commands, which are compatible with the BSD I/O environment:

TIOCEXCL

Set 'exclusive-use' mode. No further opens are permitted until the file has been closed.

TIOCNCXCL

Turn off 'exclusive-use' mode.

TIOCSFD

The **ldterm** module does nothing but reply to this command. In a BSD system, the command is used to set the current line discipline type. It does not have much meaning in a STREAMS

environment, because line discipline modules are changed by popping the current module from the stream and pushing a different one onto the stream.

TIOCGETD

In a BSD system, this command is used to get the current line discipline type. The command does not have much meaning in a STREAMS environment. The **ldterm** module replies with a value of 2 for binary compatibility, since **ldterm** supports job control.

TIOCFLUSH

This command flush the input or/and output streams similar to that of the **TCFLSH** command. The argument is a pointer to an **int** variable. If its value is zero, both the input and output streams are flushed by sending the appropriate **FLUSHR/FLUSHW M_FLUSH** messages upstream and downstream. Otherwise, the value of the **int** is treated as the logical **OR** of the **FREAD** and **FWRITE** flags defined by `<sys/file.h>`. If the **FREAD** flag is set, the input stream is flushed. If the **FWRITE** flag is set, the output stream is flushed. Then, **ldterm** acknowledges the message with **M_IOCACK**.

TIOCOUTQ

This command takes a pointer to an integer and returns the number of characters buffered up in the **ldterm**'s output buffer.

TIOCHPCL

This command sets the POSIX **termios HUPCL** flag to indicate that the terminal line should be disconnected when the last file descriptor associated with that line is closed. The **ldterm** module converts the command into a compatible POSIX **termios** I/O control command by sending an **M_IOCTL** message containing the **TCSETS** command with current **termios** settings downstream.

TIOCSTART

The command restarts output. If the terminal was stopped, the **ldterm** module sends an **M_START** message downstream.

TIOCSTOP

This command stops output. The **ldterm** module sends an **M_STOP** message downstream.

TIOCSBRK

This command sets the break condition on a line. The **ldterm** module sends an **M_BREAK** message containing a value of 1 as data to the driver, then replies with **M_IOCACK**.

TIOCCBRK

This command clears the break condition on a line. The **ldterm** module sends an **M_BREAK** message containing a value of 0 (zero) as data to the driver, then replies with **M_IOCACK**.

TIOCSETP, TIOCSETN

These commands set the **sgttyb** information, defined in `<sys/ttold.h>`. The argument is a pointer to an **sgttyb** structure. The **ldterm** module converts the message to a POSIX **termios M_IOCTL** message. Then, it forwards the POSIX **termios M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETSW, TCSETS**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TIOCGETP

This command returns the **sgttyb** information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**. The argument is a pointer to an **sgttyb** structure via where the information is returned.

TIOCSETC

This command sets the **tchars** information, defined in `<sys/strtio.h>`. The argument is a pointer to an **tchars** structure. The **ldterm** module converts the message to a POSIX **termios M_IOCTL** message. Then, it forwards the POSIX **termios M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETS**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TIOCGETC

This command returns the **tchars** information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**. The argument is a pointer to an **tchars** structure via where the information is returned.

TIOCSLTC

This command sets the **ltchars** information defined in `<sys/bsdty.h>`. The **ldterm**

module converts the message to a POSIX **termios M_IOCTL** message. Then, it forwards the POSIX **termios M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETS**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TIOCGITC

The **ldterm** module returns the **ltchars** information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**.

TIOCLBIS, TIOCLBIC, TIOCLSET

These commands set the BSD 4.3 flags information, defined in `<sys/strtio.h>`. For **TIOCLBIS** and **TIOCLBIC**, the argument is a pointer to an **int** whose value is a mask containing flags to be set/clear. For **TIOCLSET**, the argument is a pointer to an **int** whose value is a new set of flags to be set. The **ldterm** module converts the message to a POSIX **termios M_IOCTL**, then forwards the POSIX **termios M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETS**). It stores the original I/O control command and **M_IOCTL** message to be used on **M_IOCACK**.

TIOCLGET

The **ldterm** module returns the BSD 4.3 flags information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**.

TIOCSTI

This command takes an argument of a pointer to a character and pretends that the character was typed on the terminal. The user must either have the **DEVOPS** privilege or have read permission on the controlling terminal against which the **ioctl** is issued. See *privileges(5)* for more information about privileged access on systems that support fine-grained privileges.

FIONREAD

This command takes an argument of a pointer to an integer and returns the number of immediately readable characters.

AUTHOR

ldterm was developed by HP and OSF.

SEE ALSO

ioctl(2), **privileges(5)**, **pem(7)**, **ptm(7)**, **pts(7)**, **streamio(7)**, **termio(7)**.

NAME

lp - line printer

SYNOPSIS**#include <sys/lprio.h>****Remarks**

This manual entry applies **only** to a certain group of printers. For Series 800, it applies to printers controlled by the device driver **lpr2**. It does *not* apply to any printers on Series 700 systems.

DESCRIPTION

This section describes capabilities provided by many line printers supported by various versions of the HP-UX operating system. A line printer is a character special device that may optionally have an interpretation applied to the data.

If the character special device file has been created with the raw option (see the HP-UX System Administrator manuals for information about creating device files with the raw option), data is sent to the printer in **raw mode** (as, for example, when handling a graphics printing operation). In raw mode, no interpretation is done on the data to be printed, and no page formatting is performed. Data bytes are simply sent to the printer and printed exactly as received.

If the device file does not contain the raw option, data can still be sent to the printer in raw mode. Raw mode is set and cleared by the **LPRSET** request.

If the line printer device file does not contain the raw option, data is interpreted according to rules discussed below. The driver understands the concept of a printer page in that it has a page length (in lines), line length (in characters), and offset from the left margin (in characters). The default line length, indent, lines per page, open and close page eject, and handling of backspace are set to defaults determined when the printer is opened and recognized by the system the first time. If the printer is not recognized, the default line length is 132 characters, indent is 4 characters, lines per page is 66, one page is ejected on close and none on open, and backspace is handled for a character printer.

The following rules describe the interpretation of the data stream:

- A form feed causes a page eject and resets the line counter to zero.
- Multiple consecutive form-feeds are treated as a single form-feed.
- The new-line character is mapped into a carriage-return/line-feed sequence, and if an offset is specified a number of blanks are inserted after the carriage-return/line-feed sequence.
- A new-line that extends over the end of a page is turned into a form-feed.
- Tab characters are expanded into the appropriate number of blanks (tab stops are assumed to occur every eight character positions as offset by the current indent value).
- Backspaces are interpreted to yield the appropriate overstrike either for a character printer or a line printer.
- Lines longer than the line length minus the indent (i.e., 128 characters, using the above defaults) are truncated.
- Carriage-return characters cause the line to be overstruck.
- When it is opened or closed, a suitable number of page ejects is generated.

Two *ioctl(2)* requests are available to control the lines per page, characters per line, indent, handling of backspaces, and number of pages to be ejected at open and close times. At either open or close time, if no page eject is requested the paper will not be moved. For opens, line and page counting will start assuming a top-of-form condition.

The *ioctl* requests have the following form:

```
#include <sys/lprio.h>
int ioctl(int fildes, int request, struct lprio *arg);
```

The possible values of *request* are:

LPRGET Get the current printer status information and store in the **lprio** structure to which *arg* points.

(Seires 800 Only)

LPRSET Set the current printer status information from the structure to which *arg* points.

The **lprio** structure used in the **LPRGET** and **LPRSET** requests is defined in `<sys/lprio.h>`, and includes the following members:

```
short int  ind;          /* indent */
short int  col;         /* columns per page */
short int  line;        /* lines per page */
short int  bksp;        /* backspace handling flag */
short int  open_ej;     /* pages to eject on open */
short int  close_ej;    /* pages to eject on close */
short int  raw_mode;    /* raw mode flag */
```

These are remembered across opens, so the indent, page width, and page length can be set with an external program. If the **col** field is set to zero, the defaults are restored at the next open.

If the backspace handling flag is 0, a character printer is assumed and backspaces are passed through the driver unchanged. If the flag is a 1, a line printer is assumed, and sufficient print operations are generated to generate the appropriate overstruck characters.

If the raw mode flag is 0, data sent to the printer is formatted according to indent, columns per page, lines per page, backspace handling, and pages to eject on open and close.

If the raw mode flag is 1, data sent to the printer is not formatted.

If the raw mode flag is changed from 1 to 0 (raw mode is turned off) and the format settings (indent, columns per page, etc.) have not been modified, the data is formatted according to the prior format settings.

AUTHOR

lp was developed by HP and AT&T.

FILES

`/dev/lp` default or standard printer used by some HP-UX commands;
`/dev/[r]lp*` special files for printers

SEE ALSO

lp(1), slp(1), ioctl(2), cent(7), intro(7).

NAME

lvm - Logical Volume Manager (LVM)

DESCRIPTION

The Logical Volume Manager (LVM) is a subsystem for managing disk space. The HP LVM subsystem offers value-added features, such as mirroring (with the optional HP MirrorDisk/UX software), high availability (with the optional HP ServiceGuard software), and striping, that enhance availability and performance.

Unlike earlier arrangements where disks were divided into fixed-sized sections, LVM allows the user to consider the disks, also known as **physical volumes**, as a pool (or volume) of data storage, consisting of equal-sized extents. The default size of an extent is 4 MB.

An LVM system consists of arbitrary groupings of physical volumes, organized into **volume groups**. A volume group can consist of one or more physical volumes. There can be more than one volume group in the system. Once created, the volume group, and not the disk, is the basic unit of data storage. Thus, whereas earlier one would move disks from one system to another, with LVM, one would move a volume group from one system to another. For this reason it is often convenient to have multiple volume groups on a system.

Volume groups can be subdivided into virtual disks, called **logical volumes**. A logical volume can span a number of physical volumes or represent only a portion of one physical volume. The pool of disk space that is represented by a volume group can be apportioned into logical volumes of various sizes. The size of a logical volume is determined by its number of extents. Once created, logical volumes can be treated just like disk partitions. Logical volumes can be assigned to file systems, used as swap or dump devices, or used for raw access.

Commands

LVM information can be created, displayed, and manipulated with the following commands:

lvchange	Change logical volume characteristics
lvcreate	Stripe, create logical volume in volume group
lvdisplay	Display information about logical volumes
lvextend	Increase space, increase mirrors for logical volume
lvlnboot	Prepare logical volume to be root, primary swap, or dump volume
lvreduce	Decrease number of physical extents allocated to logical volume
lvremove	Remove one or more logical volumes from volume group
lvrmboot	Remove logical volume link to root, primary swap, or dump volume
pvchange	Change characteristics of physical volume in volume group
pvcreate	Create physical volume for use in volume group
pvdisk	Display information about physical volumes within volume group
pvmove	Move allocated physical extents from one physical volume to other physical volumes
vgcfgbackup	Create or update volume group configuration backup file
vgcfgrestore	Display or restore volume group configuration from backup file
vgchange	Set volume group availability
vgcreate	Create volume group
vgdisplay	Display information about volume groups
vgexport	Export a volume group and its associated logical volumes
vgextend	Extend a volume group by adding physical volumes
vgimport	Import a volume group onto the system
vgmodify	Modify volume group attributes
vgreduce	Remove physical volumes from a volume group
vgremove	Remove volume group definition from the system
vgscan	Scan physical volumes for volume groups

The following commands are also available if the HP MirrorDisk/UX software is installed:

lvmerge	Merge two logical volumes into one logical volume
lvsplit	Split mirrored logical volume into two logical volumes
lvsync	Synchronize stale mirrors in logical volumes
vgsync	Synchronize stale logical volume mirrors in volume groups

Device Special Files

In this release of HP-UX 11i, the Mass Storage Stack supports two naming conventions for the device special files used to identify devices (see *intro(7)*). Devices can be represented using:

- Persistent device special files, (**/dev/disk/disk3**), or
- Legacy device special file names, (**/dev/dsk/c0t6d6**).

While LVM supports the use of both conventions within the same volume group, the examples shown in the LVM man pages are all using the legacy device special file convention.

Alternate Links (PVLlinks)

In this release of HP-UX, LVM continues to support Alternate Links to a device to allow continued access to the device, if the primary link fails. This multiple link or multipath solution increases data availability, but continues disallowing the use of multiple paths simultaneously.

A new feature was introduced in the Mass Storage Subsystem on HP-UX 11i Version 3 that supports multiple paths to a device and allows simultaneous access to these paths. The Mass Storage Subsystem will balance the I/O load across the valid paths. Multipathing is the default unless the **scsimgr** command is used to enable legacy multipathing and also the active path is a legacy device special file. See *scsimgr(1M)* for details.

Even though the Mass Storage Subsystem supports 32 multiple paths per physical volume on this version of HP-UX, LVM does not support more than eight paths to any physical volume. As a result, commands like **vgcreate** and **vgextend** will not succeed in adding more than eight paths per physical volume. Additionally, **vgimport** and **vgscan** cannot write more than eight paths per physical volume in the **/etc/lvmtab** file. If users want to use any specific path other than these eight paths, then they have to **vgreduce** one of the alternate paths in the volume group and add that specific path using **vgextend**.

It is no longer required or recommended to configure LVM with alternate links. However, it is possible to maintain the traditional LVM behavior. To do so, both of the following criteria must be met:

- Only the legacy device special file naming convention is used in the volume group configuration.
- The **scsimgr** command is used to enable the legacy multipath behavior for each physical volume in the volume group.

EXAMPLES

The basic steps to take to begin using LVM are as follows:

- Identify the disks to be used for LVM.
- Create an LVM data structure on each identified disk (see *pvcreate(1M)*).
- Collect all the physical volumes to form a new volume group (see *vgcreate(1M)*).
- Create logical volumes from the space in the volume group (see *lvcreate(1M)*).
- Use each logical volume as if it were a disk section (create a file system, or use for raw access).

To configure disk **/dev/dsk/c0t0d0** as part of a new volume group named **vg01**:

First, initialize the disk for LVM with the **pvcreate** command.

```
pvcreate /dev/rdisk/c0t0d0
```

Then, create the pseudo device file that is used by the LVM subsystem.

```
mkdir /dev/vg01
mkknod /dev/vg01/group c 64 0x010000
```

The minor number for the **group** file should be unique among all the volume groups on the system. It has the format **0xNN0000**, where **NN** ranges from **00** to **ff**.

Create the volume group, **vg01**, containing the physical volume, **/dev/dsk/c0t0d0**, with the **vgcreate** command.

```
vgcreate /dev/vg01 /dev/dsk/c0t0d0
```

You can view information about the newly created volume group with the **vgdisplay** command.

```
vgdisplay -v /dev/vg01
```

Create a logical volume of size 100 MB, named **usrvol**, on this volume group with the **lvcreate** command.


```
lvcreate -L 100 -n usrvol /dev/vg01
```

This creates two device files for the logical volume, `/dev/vg01/usrvol`, which is the block device file, and `/dev/vg01/rusrvol`, which is the character (raw) device file.

You can view information about the newly created logical volume with the `lvdisplay` command.

```
lvdisplay /dev/vg01/lvol1
```

Any operation allowed on a disk partition is allowed on the logical volume. Thus, you can use `usrvol` to hold a file system.

```
newfs /dev/vg01/rusrvol
mount /dev/vg01/usrvol /usr
```

SEE ALSO

`lvchange(1M)`, `lvcreate(1M)`, `lvdisplay(1M)`, `lvextend(1M)`, `lvlnboot(1M)`, `lvreduce(1M)`, `lvremove(1M)`, `lvrmboot(1M)`, `pvchange(1M)`, `pvcreeate(1M)`, `pvdisplay(1M)`, `pvmove(1M)`, `vgcfgbackup(1M)`, `vgcfgrestore(1M)`, `vgchange(1M)`, `vgcreate(1M)`, `vgdisplay(1M)`, `vgexport(1M)`, `vgextend(1M)`, `vgimport(1M)`, `vgmodify(1M)`, `vgreduce(1M)`, `vgremove(1M)`, `vgscan(1M)`, `intro(7)`.

Managing Systems and Workgroups.

If HP MirrorDisk/UX is installed: `lvmerge(1M)`, `lvsplit(1M)`, `lvsync(1M)`, `vgsync(1M)`.

If HP ServiceGuard is installed: `cmcheckconf(1M)`, `cmquerycl(1M)`, *Managing MC/ServiceGuard.*

NAME

mem - main memory image file

DESCRIPTION

mem is a special file that is an image of the main memory of the computer. It may be used, for example, to examine and patch the system.

Byte addresses in **mem** are interpreted as physical memory addresses. References to non-existent locations cause errors to be returned.

File **kmem** is the same as **mem** except that kernel virtual memory rather than physical memory is accessed. Please refer to *kmem(7)* for information about ioctl operations that are supported on **/dev/kmem**.

WARNINGS

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

FILES

/dev/mem

/dev/kmem

SEE ALSO

kmem(7).

NAME

modem - asynchronous serial modem line control

SYNOPSIS

```
#include <sys/modem.h>
```

DESCRIPTION

This section describes the two modes of modem line control and the three types of terminal port access. It also discusses the effect of the bits of the *termio* structure that affect modem line control. The modem-related `ioctl()` system calls (see *ioctl(2)*) are discussed at the end of the manpage.

Definitions

There are several terms that are used within the following discussion which will be defined here for reference.

"Modem control lines" (CONTROL) are generally defined as those outgoing modem lines that are automatically controlled by the driver.

"Modem status lines" (STATUS) are generally defined as those incoming modem lines that are automatically monitored by the driver.

CONTROL and STATUS for a terminal file vary according to the modem line control mode of the file (see the *Modem Line Control Modes* section below).

An `open()` (see *open(2)*) to a port is considered to be BLOCKED if it is waiting for another file on the same port to be closed.

An `open()` to a port is considered to be PENDING if it is waiting for the STATUS to be raised.

An `open()` to a port is considered to be SUCCESSFUL if the `open()` system call has returned to the calling process without error.

Open Flag Bits

Currently, the only `open()` flag bits recognized by the driver are the `O_NDELAY` and `O_NONBLOCK` bits. When either of these bits is set, an `open()` call to the driver will never become blocked. If possible, the `open()` will be returned immediately as SUCCESSFUL, and the driver will continue the process of opening the tty file. If it is not possible, then the `open()` will be returned immediately with the appropriate error code as described in the appropriate section.

Termio Bits

When set, the `CLOCAL` bit in the *termios* or *termio* structure (see *termio(7)*) is used to remove the driver's automatic monitoring of the modem lines. However, the user's ability to control the modem lines is determined only by the mode in effect and does not depend on the state of `CLOCAL`. Normally, the driver will monitor and require the STATUS to be raised. An `open()` system call will raise the CONTROL and wait for the STATUS before completing unless the `CLOCAL` bit is set. (If the `O_NDELAY` or `O_NONBLOCK` bit is set, the `open()` will be returned immediately, but the driver will otherwise continue to monitor the modem lines as normal based on the state of the `CLOCAL` bit.) Normally, loss of the STATUS will cause the driver to break the modem connection and lower the CONTROL. However, if `CLOCAL` is set, any changes in the STATUS will be ignored. A connection is required before any data may be read or written, unless `CLOCAL` is set. Any timers that would normally be in effect (see the *Modem Line Control Modes* and *Modem Timers* sections below) will be stopped while `CLOCAL` is set.

When the `CLOCAL` bit is changed from clear to set, the driver will assume the existence of an active device (such as a modem) on the port regardless of the STATUS. If any of the CONTROL are raised at that point in time, they will continue in that state. The STATUS will no longer be actively monitored. When the `CLOCAL` bit is changed from set to clear, the driver will resume actively monitoring the STATUS. If all of the CONTROL and STATUS are raised at that point in time, the driver will continue the modem connection. If any of the STATUS are not raised, the driver will act as though those signals were lost (as described in the *Modem Line Control Modes* section below) and, if the device is a controlling terminal, a *hangup* signal will be sent to the controlling process. If any of the CONTROL are not raised, the driver will break the modem connection by lowering all the CONTROL.

The `HUPCL` bit in the *termios* or *termio* structure determines the action of the driver regarding the CONTROL when the last `close()` system call (see *close(2)*) is issued to a terminal file. If the HUPCL bit is set, the driver will lower the CONTROL at `close()` time and the modem connection will be broken. If `HUPCL` is not set and a modem connection exists, it will continue to exist, even after the `close()` is

issued. The driver will not change the CONTROL.

Terminal Port Access Types

There are three types of modem access: call-in connections, call-out connections, and direct (no modem control) connections. A given port may be accessed through all three types of connection by accessing different files. The modem access type of a terminal file is determined by the file's major and/or minor device numbers.

The call-in type of access is used when the connection is expected to be established by an incoming call. This is the type that would be used by *getty*(1M) to accept logins over a modem. When an `open()` is issued to such a file, the driver may wait for an incoming call and will then raise the CONTROL based on the current mode (see below) of the port. When the port is closed, the driver may or may not lower the CONTROL depending on the **HUPCL** bit.

The call-out type of access is used when the connection is expected to be established by an outgoing call. This would be used by programs such as *uucp*(1). When an `open()` is issued to such a file, the driver will immediately raise the CONTROL and wait for a connection based on the mode currently in effect. When the port is closed, the driver may or may not lower the CONTROL depending on the **HUPCL** bit.

The direct type of access is used when no driver modem control is desired. This could then be used for directly connected terminals that use a three-wire connection, or to talk to a modem before a connection has been established. The second case allows a program to give dialing instructions to the modem. Neither the **CLOCAL** nor the **HUPCL** bits have any effect on a port accessed through a direct file. (However, both bits may be inherited by other types of files; see the *Terminal Port Access Interlock* section below.) An `open()` to a direct file does not affect the CONTROL and does not depend on any particular state of the STATUS to succeed. When the file is closed, the driver will not affect the state of the CONTROL. If a modem connection has been established, it will continue to exist. Setting the speed of a direct file to B0 (see *termio*(7)) will be considered an impossible speed change and will be ignored. It will not affect the CONTROL.

Modem Line Control Modes

There are two modes of modem line control: CCITT mode and simple mode. A given port may have only one of these two modes in effect at any given point in time. An attempt to open a port with a mode other than the one in effect (from a PENDING or SUCCESSFUL `open()` on a different file) will cause the `open()` to be returned with an [ENXIO] error. The modem access type of a terminal file is determined by the file's major and/or minor device numbers.

CCITT mode is used for connections to switched line modems. The CONTROL for CCITT mode are Data Terminal Ready (DTR) and Request to Send (RTS). The STATUS are Data Set Ready (DSR), Data Carrier Detect (DCD), and Clear to Send (CTS). Additionally, the Ring Indicator (RI) signal indicates the presence of an incoming call. When a connection is begun (an incoming call for a call-in file or an `open()` issued to a call-out file), the CONTROL are raised and a connection timer (see the *Modem Timers* section below) is started. If the STATUS become raised before the time period has elapsed, a connection is established and the `open()` request is returned successfully. If the time period expires, the CONTROL are lowered and the connection is aborted. For a call-in file, the driver will wait for another incoming call; for a call-out file, the `open()` will be returned with an [EIO] error. Once a connection is established, loss of either DSR or CTS will cause the CONTROL to be lowered and, if the device is a controlling terminal, a *hangup* signal will be sent to the controlling process.

If DCD is lost, a timer is started. If DCD resumes before the time period has expired, the connection will be maintained. However, no data transfer will occur during this time. The driver will stop transmitting characters, and any characters received by the driver will be discarded. (However, on some implementations data transmission cannot be stopped. See the *DEPENDENCIES* section.) If DCD is not restored within the allotted time, the connection will be broken as described above for DSR and CTS.

If the modem connection is to be broken when the `close()` system call is issued (i.e. **HUPCL** is set), then the CONTROL will be lowered and the `close()` will be returned as successful. However, no further `open()`s will be allowed until after both DSR and CTS have been lowered by the modem, and the hangup timer (see the *Modem Timers* section below) has expired. The action taken in response to an `open()` during this time will be the same as if the port were still open. (See the *Terminal Port Access Interlock* section below.)

When a port is in CCITT mode, the driver has complete control of the modem lines and the user is not allowed to change the setting of the CONTROL or affect which STATUS are actively monitored by the driver (see the *Modem Ioctl's* section below). This is to provide strict adherence with the CCITT recommendations.

Simple mode is used for connections to devices which require only a simple method of modem line control. This can include devices such as black boxes, data switches, or for system-to-system connections. It can also be used with modems which cannot operate under the CCITT recommendations. The CONTROL for simple mode consists of only DTR. The STATUS consists of only DCD. When an **open()** is issued, the CONTROL is raised but no connection timer is started. When the STATUS becomes raised, a connection is established and the **open()** request is returned as SUCCESSFUL. Once a connection is established, loss of the STATUS will cause the CONTROL to be lowered and, if the device is a controlling terminal, a *hangup* signal will be sent to the controlling process.

When a port is in simple mode, the driver will normally control the modem lines. However, the user is allowed to change the setting of the CONTROL (see the *Modem Ioctl*s section below).

Terminal Port Access Interlock

An interlock mechanism is provided between the three access types of terminal files. It prevents more than one file from being successfully opened at a time, but allows certain **open()**s to succeed while others are PENDING so that a port can be opened through a call-out connection while *getty* has a pending **open()** at a call-in connection. The three access types are given a priority that determines which **open()** will succeed if more than one file has an **open()** issued against it. The three access types are ordered from lowest priority to highest as follows: call-in, call-out, and direct.

If an **open()** is issued to a port which already has a SUCCESSFUL **open()** on it of a lower priority type, the new **open()** will be returned with an [EBUSY] error. ([EBUSY] will also be returned by an attempted **open()** on a CCITT call-out file if an incoming call indication is currently being received. In this case, if there is a PENDING **open()** on the corresponding CCITT call-in file, this PENDING **open()** will complete.) If the lower priority **open()** is PENDING, the new **open()** will succeed if possible, or will be left PENDING if waiting for the STATUS and the lower priority **open()** will become BLOCKED. If a higher priority **open()** has succeeded or is PENDING, the new **open()** will be BLOCKED, unless the new **open()** has the **O_NDELAY** flag bit set, in which case the **open()** will be returned with an [EBUSY] error. Once an **open()** on one type of file is SUCCESSFUL, any PENDING *opens* on lower priority files will become BLOCKED.

When a file of one priority is closed, a BLOCKED **open()** on the next lower priority type file will become active. If all of the STATUS are raised, the **open()** will be SUCCESSFUL, otherwise the **open()** will become PENDING waiting for the STATUS. If the lower priority **open()** is SUCCESSFUL (because the connection was maintained when the higher priority file was closed), the port characteristics (speed, parity, etc.) that were set by the higher priority file will be inherited by the lower priority file. If the connection is not maintained through the **close()**, the port characteristics will be set to default values.

Modem Timers

There are four timers currently defined for use with modem connections. The first three of the timers are applicable only to CCITT mode connections. In general, the effect of changing a timer value while the timer is running is system dependent. However, setting the timer value to zero is guaranteed to disable the timer even if it is running.

The connect timer is used to limit the amount of time to wait for a connection to be established once it has been begun. This timer is started when an incoming call has been received on a call-in file, or when an **open()** has been issued on a call-out file for which no *opens* are already pending. If the connection is completed in time, the timer is aborted. If the time period expires, the connection is aborted. For a call-in file, the driver will again wait for an incoming call and the **open()** will remain pending. For a call-out file, the **open()** will be returned with an [EIO] error.

The carrier detect timer is used to limit the amount of time to wait before causing a disconnect if DCD drops. If carrier is not re-established in this time, a disconnect will occur. If carrier is re-established before the timeout, the timer will be aborted and the connection maintained. During the period when carrier is not raised, no data will be transferred across the line.

The no activity timer is used to limit the amount of time a connection will remain open with no data transfer across the line. When the data line becomes quiescent with no data transfer, this timer will be started. If data is again transferred over the line in either direction before the time limit, the timer will be aborted. If no activity occurs before the timeout has occurred, the driver will disconnect the line. This can be used to avoid long and costly telephone connections when data transfer has been stopped either normally or abnormally.

The last timer defined, the hangup timer, is used for both CCITT and simple modes. This timer controls the amount of time to wait after disconnecting a modem line before allowing another **open()**. This time period should be made long enough to guarantee that the connection has been terminated by the telephone

switching equipment. If this period is not long enough, the telephone connection may not be broken and a succeeding `open()` may complete with the old connection.

HP-UX Modem Ioctl's

Several `ioctl()` system calls apply to manipulation of modem lines. They use the following information defined in `<sys/modem.h>`:

```
#define NMTIMER 6
typedef unsigned long mflag;
struct mtimer {
    unsigned short m_timers[NMTIMER];
};
```

Each bit of the `mflag` long corresponds to one of the modem lines as follows:

MRTS	Request to Send	outbound
MCTS	Clear to Send	inbound
MDSR	Data Set Ready	inbound
MDCD	Data Carrier Detect	inbound
MDTR	Data Terminal Ready	outbound
MRI	Ring Indicator	inbound
MDRS	Data Rate Select	outbound

The timer values are defined in the array `m_timers`. The relative position of the timer and default initial values and units for each timer are as follows:

0	MTCCONNECT	25 s
1	MTCARRIER	400 ms
2	MTNOACTIVITY	0 min
3	MTHANGUP	250 ms
4	Reserved	
5	Reserved	

A value of zero for any timer will disable that timer.

The modem line `ioctl()` system calls have the form:

```
int ioctl(int fildes, int command, mflag *arg);
```

The commands using this form are:

MCGETA	Get the current state of both inbound and outbound modem lines and store in the <code>mflag</code> long referenced by <code>arg</code> . A raised line will be indicated by a one bit in the appropriate position.
MCSETA	Set the outbound modem lines from the <code>mflag</code> long referenced by <code>arg</code> . Setting an outbound bit to one causes that line to be raised and zero to be lowered. Setting bits for inbound lines has no effect. Setting any bits while in CCITT mode has no effect. The change to the modem lines is immediate and using this form while characters are still being output may cause unpredictable results.
MCSETAW	Wait for the output to drain and set the new parameters as described above.
MCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters as described above.

The timer value `ioctl()` system calls have the form:

```
int ioctl(int fildes, int command, mtimer *arg);
```

The commands using this form are:

MCGETT	Get the current timer value settings and store in the <code>mtimer</code> structure referenced by <code>arg</code> .
MCSETT	Set the timer values from the structure referenced by <code>arg</code> .

For any timer, setting the timer value to its previous value has no effect.

SVID3 Modem Ioctl's

System V Interface Definition, Third Edition (SVID3) specifies additional `ioctl()` system calls to manipulate the modem lines. They use information defined in `<termios.h>`.

Each `ioctl()` passes an integer argument in which each of the following bit definitions correspond to one of the modem lines as follows:

TIOCM_RTS	Request to Send	outbound
TIOCM_CTS	Clear to Send	inbound
TIOCM_DSR	Data Set Ready	inbound
TIOCM_CAR	Data Carrier Detect	inbound
TIOCM_DTR	Data Terminal Ready	outbound
TIOCM_RNG	Ring Indicator	inbound

Additionally, **TIOCM_CD** is equivalent to **TIOCM_CAR**, and **TIOCM_RI** is equivalent to **TIOCM_RNG**.

The modem line `ioctl()` system calls have the form:

```
int ioctl(int fildes, int command, int *arg);
```

The commands using this form are:

TIOCMGET	Get the current state of both inbound and outbound modem lines and store in the int referenced by arg . A raised line will be indicated by a one bit in the appropriate position.
TIOCMSET	Set the outbound modem lines from the int referenced by arg .
TIOCMBS	Raise the control lines specified by a one in the corresponding bit positions of the int referenced by arg .
TIOCMBSIC	Lower the control lines specified by a one in the corresponding bit positions of the int referenced by arg .

Note that setting bits for inbound lines has no effect, and setting any bits while in CCITT mode has no effect. Also, the change to the modem lines is immediate and using these `ioctl`'s while characters are still being output may cause unpredictable results.

WARNINGS

Occasionally it is possible that a process may open a call-out file at approximately the same time as an incoming call is received. In some cases, the call-out connection may be satisfied by the incoming call. In general, however, the results are indeterminate. If necessary, the situation can be avoided by the use of two modems and ports, one for call-out connections and the other for receiving incoming calls.

DEPENDENCIES

Some hardware implementations may not have access to all modem lines supported by MCSETA. If a particular hardware does not support a given line, attempts to set the value of a line will be ignored, and reading the current state of the line will return zero. The appropriate I/O card manual should be referenced to determine the lines supported by the hardware installed.

Some hardware implementations may not have access to all timers supported by MCSETT. Also, the granularity of the individual timers may vary depending on the hardware and system in use. The effect of setting a timer out of range or with a granularity outside the capability of a particular system should be documented by that system. The effect of changing the value for a timer while that timer is running is system dependent and should be documented by each system.

Setting the **CLOCAL** bit while a timer is running will cause the timer to be stopped. It is a system dependency whether or not the timer is restarted, and if so, the value at which it is restarted when the **CLOCAL** bit is subsequently cleared.

On those implementations supporting the HP27140A 6-Channel Multiplexer, transmission of characters cannot be stopped during loss of DCD. The driver cannot detect loss of DCD until the connection is broken. Also, the I/O card may still have characters in its internal buffers and will still try to transmit them.

AUTHOR

modem was developed by HP and AT&T.

FILES

```
/dev/cua*
/dev/cul*
/dev/tty*
/dev/ttyd*
```

modem(7)

modem(7)

SEE ALSO

stty(1), mknod(1M), ioctl(2), open(2), termio(7).



m

NAME

mt - magnetic tape interface and controls for stape and estape

DESCRIPTION

This entry describes the behavior of HP magnetic tape interfaces and controls. The files `/dev/rtape/*` refer to specific raw tape drives controlled by the estape driver. The major number of these device special files is dynamically allocated and the minor number does not encode any device specific information.

The files `/dev/rmt/*` refer to specific raw tape drives controlled by the legacy stape driver, and the behavior of each given unit is specified in the major and minor numbers of the DSF. The legacy driver and DSFs are deprecated and will be removed in a future version of HP-UX.

Naming Conventions

The device special files (referred to as DSFs) for the **estape** driver have the following naming conventions:

`/dev/rtape/tape#_BEST[n][b]`

There are four such files (referred to as persistent DSFs) corresponding to each of the four different permutations of the **n** and **b** options. These are claimed by the **estape** driver. See *intro(7)* for more details on persistent device special file names.

There are two naming conventions for legacy DSFs. The standard (preferred) convention is used on systems that support long file names. An alternate convention is provided for systems limited to short file names. The following standard convention is recommended because it allows for all possible configuration options in the device name and is used by *mksf(1M)* and *insf(1M)*:

`/dev/rmt/c#t#d#[o][z][e][p][s][#][w]BEST[C#[#]][n][b]`

The following alternate naming convention is provided to support systems in which the `/dev/rmt` directory requires short file names. These DSF names are less descriptive, but guarantee unique device naming and are used by *mksf(1M)* and *insf(1M)* where required.

`/dev/rmt/c#t#d#[f#|i#][n][b]`

For each tape device present, twelve DSFs are automatically created when the system is installed. If legacy mode is disabled (via the `-L` option in **rmsf**), only four DSFs in `/dev/rtape` will be created post installation. These are claimed by the **estape** driver.

Four legacy DSFs will be created in the `/dev/rmt` directory using the following naming convention. These are legacy DSFs and are claimed by the **stape** driver.

`/dev/rmt/c#t#d#BEST[n][b]`.

Four more legacy DSFs with the format `/dev/rmt/#m[n][b]` will be automatically created when the system is installed using the pre-HP-UX 10.0 device file naming convention. This includes an arbitrary number to distinguish this tape device from others in the system, followed by the letter **m**. There are four such DSFs because each of the four different permutations of the **n** and **b** options (see below) are created. These files are created for compatibility with pre-HP-UX 10.0 scripts and for users who find the old convention easier to remember.

Each of the automatically created DSFs which utilize the standard or alternate naming conventions is linked to a device file which utilizes the pre-HP-UX 10.0 naming convention. That is, the DSFs in the format `/dev/rmt/#m[n][b]` are created as hardlinks to the corresponding `/dev/rmt/c#t#d#BEST[n][b]` DSFs mentioned above.

Thus, the DSFs which utilize the pre-HP-UX 10.0 naming convention provide the same functionality as the device files which contain the density specification **BEST** (standard naming convention).

Options

The options described here are common to all legacy tape drivers. The `c#t#d#` notation in the legacy DSF name derives from **ioscan** output and is described in the manpages for *ioscan(1M)* and *intro(7)*.

- c#** Instance number assigned by the operating system to the interface card.
- t#** Target address on a remote bus (for example, SCSI address)
- d#** Device unit number at the target address (for example, SCSI LUN).
- w** Writes wait for physical completion of the operation before returning status. The default behavior (buffered mode or immediate reporting mode) requires the tape device to buffer the

data and return immediately with successful status.

<i>density</i>	Density or format used in writing data to tape. This field is designated by the following values:
BEST	Highest-capacity density or format will be used, including data compression, if the device supports compression.
NOMOD	Maintains the density used for data previously written to the tape. Behavior using this option is dependent on the type of device. This option is only supported on DDS drives.
<i>DDS</i>	Selects one of the known DDS formats; can be used to specify DDS1 or DDS2 , as required.
<i>DLT</i>	Selects one of the known DLT formats; can be used to specify DLT42500_24 , DLT42500_56 , DLT62500_64 , DLT81633_64 , or DLT85937_52 , as required.
<i>D[#]</i>	Specifies density as a numeric value to be placed in the SCSI mode select block descriptor. The header file <code><sys/mtio.h></code> contains a list of the standard density codes. The numeric value is used only for density codes which <i>cannot</i> be found in this list.
C[#]	Write data in compressed mode, on tape drives that support data compression. If a number is included, use it to specify a compression algorithm specific to the device. Note, compression is also provided when the density field is set to BEST .
n	No rewind on close. Unless this mode is requested, the tape is automatically rewound upon close.
b	Specifies Berkeley-style tape behavior. When the b is absent, the tape drive follows AT&T-style behavior. The details are described in <i>Tape Behavioral Characteristics</i> below.
f#	Specify format (or density) value encoded in the minor number. The meaning of the value is dependent on the type of tape device in use. (Used for short file name notation only.)
i#	Specify an internal Property Table index value maintained by the tape driver, containing an array of configuration options. The contents of this table are not directly accessible. Use the <code>lssf(1M)</code> command to determine which configuration options are invoked. (Used for short file name notation only.)
o	Console message disabled. See <code>mksf(1M)</code> .
z	RTE compatible close. See <code>mksf(1M)</code> .
e	Exhaustive mode. See <i>DEPENDENCIES</i> section.
p	Tape partition. See <i>DEPENDENCIES</i> section.
s	Fixed-block mode. See <i>DEPENDENCIES</i> section.
#m	For pre-HP-UX 10.x device file naming convention.

Sample Tape Device Special File Names

For a HP Ultrium-2 drive at card instance 1, target 2, LUN 3 the legacy DSFs would be `/dev/rmt/c1t2d3BEST[n][b]`. The corresponding persistent DSFs assuming an instance number "1" allocated to the DSF would be `/dev/rtape/tape1_BEST[n][b]`. Corresponding device special files in the pre-HP-UX 10.0 naming convention would be `/dev/rmt/0m[n][b]`. In this particular example, 0 (zero) in `0m[n][b]` denotes an instance number of 0 (zero) assigned to the DSF. The files in the `/dev/rmt/#m[n][b]` format are created as hardlinks to the corresponding `/dev/rmt/c#t#d#BEST[n][b]` DSFs.

Use the `lssf(1M)` command to determine which configuration options are actually used with any device file. The naming convention defined above should indicate the options used, but device files may be created with any user defined name.

Tape Behavioral Characteristics

When opened for reading or writing, the tape is assumed to be positioned as desired.

When a file opened for writing is closed, two consecutive EOF (End of File) marks are written if, and only if, one or more writes to the file have occurred. The tape is rewound unless the no-rewind mode has been specified, in which case the tape is positioned before the second EOF just written.

When a file open for reading (only) is closed and the no-rewind bit is not set, the tape is rewound. If the no-rewind bit is set, the behaviour depends on the *style* mode. For AT&T-style devices, the tape is positioned after the EOF following the data just read (unless already at BOT or Filemark). For Berkeley-style devices, the tape is not repositioned in any way.

Each *read(2)* or *write(2)* call reads or writes the next record on the tape. For writes, the record has the same length as the buffer given (within the limits of the hardware).

During a read, the record size is passed back as the number of bytes read, up to the buffer size specified. Since the minimum read length on a tape device is a complete record (to the next record mark), the number of bytes ignored (for records longer than the buffer size specified) is available in the **mt_resid** field of the **mtget** structure via the **MTIOCGET** call of *ioctl(2)*. Current restrictions require tape device application programs to use 2-byte alignment for buffer locations and I/O sizes. To allow for more stringent future restrictions (4-byte aligned, etc.) and to maximize performance, page alignment is suggested. For example, if the target buffer is contained within a structure, care must be taken that structure elements before the buffer allow the target buffer to begin on an even address. If need be, placing a filler integer before the target buffer will insure its location on a 4-byte boundary.

The ascending hierarchy of tape marks is defined as follows: record mark, filemark (EOF), setmark and EOD (End of Data). Not all devices support all types of tape marks but the positioning within the hierarchy holds true. Each type of mark is typically used to contain one or more of the lesser marks.

When spacing over a number of a particular type of tape mark, hierarchically superior marks (except EOD) do not terminate tape motion and are included in the count. For instance, MTFSR can be used to pass over record marks and filemarks.

Reading an EOF mark is returned as a successful zero-length read; that is, the data count returned is zero and the tape is positioned after the EOF, enabling the next read to return the next record.

DDS devices also support setmarks, which are used to delineate a group (set) of files. Reading a setmark is also returned as a zero-length read. Filemarks, setmarks and EOD can be distinguished by unique bits in the **mt_gstat** field.

Spacing operations (back or forward space, setmark, file or record) position past the object being spaced to in the direction of motion. For example, back-spacing a file leaves the tape positioned before the file mark; forward-spacing a file leaves the tape positioned after the file mark. This is consistent with standard tape usage.

lseek(2) type seeks on a magnetic tape device are ignored. Instead, the *ioctl(2)* operations below can be used to position the tape and determine its status.

The header file **<sys/mtio.h>** has useful information for tape handling.

Macros to Decode Options

The minor number of the device ID (**dev_t**) of persistent tape device special files no longer encode the tape device options (such as, density, style of access and so on). Hence the macros given below, that are defined in **<sys/mtio.h>** header file do not interpret the options correctly for persistent (agile) DSFs. The macros are:

M_INSTANCE (dev)	M_TARGET (dev)
M_LUN (dev)	M_BERKELEY (dev)
M_NO_REWIND (dev)	M_USER_CONFIG (dev)
M_INDEX (dev)	M_INDEX_PUT (dev, index)
M_DFLT_DENSITY (dev)	M_DFLT_DENSITY_PUT (dev, density)
M_TRANSPARENT_MODE (dev)	M_PROP_TBL_ACCESS (dev)

These macros continue to work on the legacy DSFs as before.

Applications should use the method described below to decode the tape device options from persistent device files.

libIO(3X) API **io_dev_to_options** is used to decode the device options from the persistent device files as given below:

```
#include <libIO.h>
#include <sys/_inttypes.h>
#include <fcntl.h>
```

Note: **libIO** calls should be within calls to **io_init()** and **io_end()**. Refer to *libIO(3X)* manpage for more details. Applications have to link with **libIO** library to access these APIs.

`mt_get_newdev_options()` and `mt_check_newdev_options()` are utility functions used by the code snippets below.

```
uint64_t
mt_get_newdev_options(dev_t dev, int dev_type) {
    uint64_t    options;
    int         err;
    err = io_dev_to_options(dev, dev_type, &options);
    if (err == IO_ERROR)
        return 0;
    return (options);
}

uint64_t
mt_check_newdev_options(dev_t dev, int dev_type, uint64_t bitmask) {
    uint64_t    options;
    int         err;
    err = io_dev_to_options(dev, dev_type, &options);
    if (err == IO_ERROR)
        return 0;
    return (options & bitmask);
}
```

For example, the macro `M_BERKELEY_AGILE` given below decodes the device options of both legacy and persistent (agile) DSFs. This macro returns true if the device ID is that of a device special file supporting Berkeley style of access.

Example

```
File test.c :

#include <stdlib.h>
#include <sys/libIO.h>
#include <sys/_inttypes.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <fcntl.h>
#include <sys/mtio.h>

#define MT_IS_LEGACY_DEV 1

#define M_BERKELEY_AGILE(dev) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \
     (dev & MT_BSD_MASK) : \
     (mt_check_newdev_options(dev, D_CHR, MT_BSD_MASK)))

/*
 * It is assumed that definitions of mt_get_newdev_options() and
 * mt_check_newdev_options() are defined by the application and
 * available. Omitted here for the sake of simplicity.
 */

int
main(int argc, char *argv[]) {

    struct stat stbuf;
    dev_t dev;

    /* Device special file is passed as argv[1] */

    if (stat(argv[1], &stbuf) < 0)
    {
        perror("stat(): ");
        exit (1);
    }
}
```

```

    }

    dev = stbuf.st_rdev;

    io_init(O_RDWR);

    if(M_BERKELEY_AGILE(dev))
        printf(" This is a Berkeley style device file ");
    else
        printf(" This is not a Berkeley style device file ");

    io_end();

    exit(0);
}

```

Compile Line: cc -Ae -o test test.c -lIO

Sample Output:

```

# ./test /dev/rtape/tape1_BESTn
This is not a Berkeley style device file
# ./test /dev/rtape/tape1_BESTb
This is a Berkeley style device file
# ./test /dev/rmt/0mnb
This is a Berkeley style device file
# ./test /dev/rmt/c5t4d0BEST
This is not a Berkeley style device file
# ./test /dev/rmt/c5t4d0BESTnb
This is a Berkeley style device file

```

Macros similar to the one above, can be written in place of their respective legacy macros as follows:

```

#define M_INSTANCE_AGILE(dev) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \
    ((dev) >> MT_INSTANCE_BIT_POS) & MT_INSTANCE_MASK) : \
    ((mt_get_newdev_options(dev, D_CHR) >> MT_INSTANCE_BIT_POS) \
    & MT_INSTANCE_MASK))

#define M_TARGET_AGILE(dev) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \
    ((dev) >> MT_TARGET_BIT_POS) & MT_TARGET_MASK) : \
    ((mt_get_newdev_options(dev, D_CHR) >> MT_TARGET_BIT_POS) \
    & MT_TARGET_MASK))

#define M_LUN_AGILE(dev) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \
    ((dev) >> MT_LUN_BIT_POS) & MT_LUN_MASK) : \
    ((mt_get_newdev_options(dev, D_CHR) >> MT_LUN_BIT_POS) \
    & MT_LUN_MASK))

#define M_USER_CONFIG_AGILE(dev) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \
    (dev & MT_USER_CONFIG_MASK) : \
    (mt_check_newdev_options(dev, D_CHR, MT_USER_CONFIG_MASK)))

#define M_INDEX_AGILE(dev) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \
    ((dev) & MT_INDEX_MASK) >> MT_INDEX_BIT_POS) : \
    ((mt_check_newdev_options(dev, D_CHR, MT_INDEX_MASK)) >> \
    MT_INDEX_BIT_POS));

#define M_INDEX_PUT_AGILE(dev, index) \
    ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ? \

```

```

        (((dev) & (~MT_INDEX_MASK)) |
        (index << MT_INDEX_BIT_POS) |
        MT_USER_CONFIG_MASK) :
        ((mt_check_newdev_options(dev, D_CHR, ~MT_INDEX_MASK)) |
        (index << MT_INDEX_BIT_POS)))

#define M_DFLT_DENSITY_PUT_AGILE(dev,density)
        ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ?
        (((dev) & (~MT_DENSITY_MASK)) |
        (density << MT_DENSITY_BIT_POS)) :
        ((mt_check_newdev_options(dev, D_CHR, ~MT_DENSITY_MASK)) |
        (density << MT_DENSITY_BIT_POS)))

#define M_TRANSPARENT_MODE_AGILE(dev)
        ((io_is_legacy_dev(dev, D_CHR) == MT_IS_LEGACY_DEV) ?
        (((dev) & MT_TRANSPARENT_MASK) ==
        MT_TRANSPARENT_VAL) :
        ((mt_check_newdev_options(dev, D_CHR, MT_TRANSPARENT_MASK))
        == MT_TRANSPARENT_VAL))

```

The following is included from `<sys/mtio.h>` and describes the possible tape operations:

```

/* mag tape I/O control requests */

#define MTIOCTOP _IOW('m', 1, struct mtop) /* do mag tape op */
#define MTIOCGET _IOR('m', 2, struct mtget) /* get tape status */

/* structure for MTIOCTOP - mag tape op command */

struct mtop {
    short mt_op;          /* operations defined below */
    int32_t mt_count;    /* how many of them */
};

/* operations */

#define MTWEOF 0 /* write filemark (end-of-file record) */
#define MTFFSF 1 /* forward space file */
#define MTBSF 2 /* backward space file */
#define MTFSR 3 /* forward space record */
#define MTBSR 4 /* backward space record */
#define MTREW 5 /* rewind */
#define MTOFFL 6 /* rewind and put the drive offline (may eject) */
#define MTNOP 7 /* no operation, may set status */
#define MTEOD 8 /* DDS, QIC and 8MM only - seek to end-of-data */
#define MTWSS 9 /* DDS and 8MM only - write setmark(s) */
#define MTFSS 10 /* DDS and 8MM only - space forward setmark(s) */
#define MTBSS 11 /* DDS and 8MM only - space backward setmark(s) */
#define MTSTARTVOL 12 /* Start a new volume (for ATS) */
#define MTENDVOL 13 /* Terminate a volume (for ATS) */
#define MTRES 14 /* Reserve Device */
#define MTREL 15 /* Release Device */
#define MTERASE 16 /* Erase media */

/* structure for MTIOCGET - mag tape get status command */

struct mtget {
    long mt_type;        /* type of magtape device */
    long mt_resid;      /* residual count */
};

/* The following two registers are device dependent */

```

m

```

    long      mt_dsreg1;    /* status register (msb) */
    long      mt_dsreg2;    /* status register (lsb) */

/* The following are device-independent status words */

    long      mt_gstat;     /* generic status */
    long      mt_erreg;     /* error register */
    int32_t   mt_fileno;    /* No longer used - always set to -1 */
    int32_t   mt_blkno;     /* No longer used - always set to -1 */

```

Information for decoding the **mt_type** field can be found in `<sys/mtio.h>`.

Tape operations work the same way for both legacy and agile devices.

Other Tape Status Characteristics

Efficient use of streaming tape drives with large internal buffers and immediate-reporting require the following end-of-tape procedures:

All writes near LEOT (Logical End of Tape) complete without error if actually written to the tape. Once the tape driver determines that LEOT has been passed, subsequent writes do not occur and an error message is returned.

To write beyond this point (keep in mind that streaming drives have already written well past LEOT), simply ask for status using the **MTIOCGET** ioctl. If status reflects the EOT condition, the driver drops all write barriers.

Both the **estape** and **stape** drivers will flush the device buffers when a write filemark (all devices) or write setmark (devices that support setmarks) command is given with the count set to zero.

When immediate-reporting is disabled, the write encountering LEOT returns an error with the tape driver automatically backing up over that record.

When reading near the end-of-tape, the user is not informed of LEOT. Instead, the typical double EOF marks or a pre-arranged data pattern signals the logical end-of-tape.

Since magnetic tape drives vary in EOT sensing due to differences in the physical placement of sensors, any application (such as multiple-tape *cpio*(1) backups) requiring that data be continued from the EOT area of one tape to another tape must be restricted. Therefore, the tape drive type and mode should be identical for the creation and reading of the tapes.

The following macros are defined in `<sys/mtio.h>` for decoding the status field **mt_gstat** returned from **MTIOCGET**. For each macro, the input parameter *x* is the **mt_gstat** field.

GMT_BOT (<i>x</i>)	Returns TRUE at beginning of tape.
GMT_EOD (<i>x</i>)	Returns TRUE if End-of-Data is encountered for DDS, QIC or 8MM.
GMT_EOF (<i>x</i>)	Returns TRUE at an End-of-File mark.
GMT_EOT (<i>x</i>)	Returns TRUE at end of tape.
GMT_IM_REP_EN (<i>x</i>)	Returns TRUE if immediate reporting mode is enabled.
GMT_ONLINE (<i>x</i>)	Returns TRUE if drive is online.
GMT_SM (<i>x</i>)	Returns TRUE if setmark is encountered.
GMT_WR_PROT (<i>x</i>)	Returns TRUE if tape is write protected.
GMT_COMPRESS (<i>x</i>)	Returns TRUE if data compression is enabled.
GMT_DENSITY (<i>x</i>)	Returns the currently configured 8-bit density value. Supported values are defined in <code><sys/mtio.h></code> .
GMT_D_800 (<i>x</i>)	Returns TRUE if the density encoded in mt_gstat is 800 bpi.
GMT_D_1600 (<i>x</i>)	Returns TRUE if the density encoded in mt_gstat is 1600 bpi.
GMT_D_6250 (<i>x</i>)	Returns TRUE if the density encoded in mt_gstat is 6250 bpi (with or without compression).
GMT_D_6250c (<i>x</i>)	Returns TRUE if the density encoded in mt_gstat is 6250 bpi plus compression.

<code>GMT_D_DDS1(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is DDS1 (with or without compression).
<code>GMT_D_DDS1c(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is DDS1 plus compression.
<code>GMT_D_DDS2(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is DDS2 (with or without compression).
<code>GMT_D_DDS2c(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is DDS2 plus compression.
<code>GMT_D_DLT_42500_24(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 42500 bpi, 24 track pairs.
<code>GMT_D_DLT_42500_56(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 42500 bpi, 56 track pairs.
<code>GMT_D_DLT_62500_64(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 62500 bpi (with or without compression).
<code>GMT_D_DLT_62500_64c(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 62500 bpi plus compression.
<code>GMT_D_DLT_81633_64(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 81633 bpi (with or without compression).
<code>GMT_D_DLT_81633_64c(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 81633 bpi plus compression.
<code>GMT_D_DLT_85937_52(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 85937 bpi (with or without compression).
<code>GMT_D_DLT_85937_52c(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is 85937 bpi plus compression.
<code>GMT_D_3480(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is for a 3480 device (with or without compression).
<code>GMT_D_3480c(x)</code>	Returns TRUE if the density encoded in <code>mt_gstat</code> is for a 3480 device with compression.
<code>GMT_DR_OPEN(x)</code>	Does not apply to any currently supported devices. Always returns FALSE.

HP-UX silently enforces a tape record blocking factor (**MAXPHYS**) on large I/O requests. For example, a user write request with a length of ten times **MAXPHYS** will actually reach the media as ten separate records. A subsequent read (with ten times **MAXPHYS** as a length) will look like a single operation to the user, even though HP-UX has broken it up into ten separate read requests to the driver. The blocking function is transparent to the user during writes. It is also transparent during reads unless:

- The user picks an arbitrary read length greater than **MAXPHYS**.
- The user attempts to read a third-party tape containing records larger than **MAXPHYS**.

Since the value for **MAXPHYS** is relatively large (usually \geq 256K bytes), this is typically not a problem.

The **MTNOP** operation does not set the device-independent status word.

EXAMPLES

Assuming that *fd* is a valid file descriptor, the following example writes two consecutive filemarks on the tape:

```
#include <sys/types.h>
#include <sys/mtio.h>

struct mtop mtop;

mtop.mt_op = MTWEOF;
mtop.mt_count = 2;
ioctl(fd, MTIOCTOP, &mtop);
```


If *fd* is a valid file descriptor for an open DDS drive, the following example spaces forward to just past the next setmark:

```
#include <sys/types.h>
#include <sys/mtio.h>

struct mtop mtop;

mtop.mt_op = MTFSS;
mtop.mt_count = 1;
ioctl(fd, MTIOCTOP, &mtop);
```

Given that *fd* is a valid file descriptor for an opened tape device, and that it has just returned 0 from a *read(2)* request. The following system call verifies that the tape has just read a filemark:

```
#include <sys/types.h>
#include <sys/mtio.h>

struct mtget mtget;

ioctl(fd, MTIOCGET, &mtget);
if (GMT_EOF (mtget.mt_gstat)) {
/* code for filemark detection */
}
```

WARNINGS

Density specification **BEST** (standard naming convention) activate data compression on tape devices which support compression. This is also true for the files using the pre-HP-UX 10.0 naming convention which are linked to these files (see "Naming Conventions" above).

For the persistent tape DSFs the minor number does not encode any configuration option. The minor number represents an index into a persistent kernel database where the configuration options are stored.

It is recommended that all legacy tape device files be put in the **/dev/rmt** directory. Legacy Device files using extended configuration options located outside the **/dev/rmt** directory may not provide consistent behavior across system reboots.

Although persistent DSFs may be created in directories other than **/dev/rtape**, HP recommends that persistent tape DSFs only be created in **/dev/rtape**.

Use the *rmsf(1M)* command to clean up unused device files. Otherwise, the property table may overflow and cause the *mksf(1M)* command to fail.

Density codes listed in **<sys/mtio.h>** have device-dependent behaviors. See the hardware manual for your tape device to find which densities are valid. For some devices, these values may be referred to as formats instead of densities.

Use of unbuffered mode can reduce performance and increase media wear.

DEPENDENCIES

Driver-Specific Options for stape (Major Number 205)

The following options may be used in creating legacy DSFs for tape drives that access the **stape** driver:

- e** Exhaustive mode is enabled (default is disabled).
When exhaustive mode is enabled, the driver will, if necessary, attempt several different configuration options when opening a device. The first attempt follows the minor number configuration exactly, but if that fails, the driver attempts other likely configuration values.
With Exhaustive mode disabled, the driver makes only one attempt to configure a device using the configuration indicated in the minor number.
- p** Specifies a partitioned tape whose currently active partition is partition 1 (closest to BOT (beginning of tape)). Optional partition 1 is closest to BOT for possible use as a volume directory. The default partition without this option is partition 0. If partitioning is unsupported, the entire tape is referred to as partition 0.
- s [#]** Specifies fixed-block mode; the optional number indicates the block size. If the number is not present, the driver selects a default block size appropriate to the device type.

AUTHOR

mt was developed by HP and the University of California, Berkeley.

FILES

/dev/rtape/* Persistent tape DSFs claimed by the estape driver
/dev/rmt/* Legacy tape DSFs
<sys/mtio.h> Constants and macros for use with tapes
/etc/mtconfig Configuration property table for tapes
/dev/rmt/*config Device files for accessing configuration properties table - for internal use only

SEE ALSO

dd(1), mt(1), insf(1M), ioscan(1M), lssf(1M), mkssf(1M), rmsf(1M), ioctl(2), lseek(2), libIO(3X), intro(7).

Configuring HP-UX for Peripherals

NAME

ndp - Neighbor Discovery Protocol, NDP

DESCRIPTION

Neighbor Discovery Protocol (NDP) is a protocol used by hosts and routers to:

1. Find the link-layer address of the neighbors known to be attached to the same link.
2. Find the neighboring routers that are willing to forward packets on their behalf.
3. Actively keep track of which neighbors are reachable and which are not.
4. Search for alternate routers when the path to a router fails.

To accomplish the above mentioned tasks, NDP defines the following processes:

1. Router and Prefix Discovery

Router discovery is a process through which hosts locate the neighboring routers and learn prefix plus other parameters necessary for address autoconfiguration.

Prefix discovery is used by the hosts to learn the range of IPv6 addresses that reside on-link and can be reached without going through a router.

Routers send Router Advertisements which will make the hosts treat them as the default routers. The Router Advertisements will also contain prefix information options that will identify the range of IPv6 addresses that are on-link (Subnet prefix).

2. Router and Host Requirements

Router requirements in NDP specify a set of rules for host to act as a router. These rules include:

- Router configuration variables.
These configuration variables include intervals between successive unsolicited router advertisements, etc.
- How to make an interface an advertising interface.
When an interface is made an advertising interface, it means that the node is going to send periodic router advertisements and is willing to forward packets on behalf of hosts on that link.
- Message content for router advertisements.
A router will send periodic as well as solicited Router Advertisements on an advertising interface. NDP specifies the format of these messages.
- Sending unsolicited router advertisements.
Apart from sending solicited router advertisements in response to router solicitations, routers can send unsolicited router advertisements. For example, unsolicited router advertisements can be sent to expire a prefix or to advertise a new prefix, etc.
- Stopping router advertisements on an interface.
A router can stop advertising prefixes on an interface. This can happen due to system management decisions when a router may be stopped from being one. NDP specifies what the router should be doing under these circumstances.
- Processing router solicitation messages.
Hosts as part of the stateless autoconfiguration process will send Router Solicitations. Routers should respond to such solicitations with a router advertisement.
- Steps to be taken when the link-local address for the router changes.
Normally the link-local address of a Router should not change. However, NDP still defines the steps should be taken by the router when its link-local address changes for any of its interfaces.

Host requirements are a set of rules that apply for a IPv6 host. They are:

- IPv6 variables that have to be maintained.
These variables include the time between retransmissions of neighbor solicitations, link MTU for each interface, etc.

- Processing router advertisements.

This rule discusses what actions should be taken on receipt of router advertisements.

- Timing out prefixes and default routers.

Whenever routers send router advertisements, they include the lifetime of the router as well as the prefixes that they advertise. NDP specifies what actions the host should take when these lifetimes expire.

- Selecting a default router.

When there is more than one router in the link, the default router selection algorithm comes into effect. This algorithm helps select the default router based on factors like reachability, etc.

- Sending a router solicitation.

When an interface is enabled, a host need not wait for the unsolicited router advertisement. Instead, it can send a router solicitation and get a router advertisement as a response. This will help in receiving the default router and prefix information as soon as the interface is enabled.

3. Algorithm for Sending a Packet

Any IPv6 host is required to maintain some data structures that will be used by the algorithm for sending a packet. These data structures are:

Neighbor Cache

A set of entries that will maintain IPv6 Address to link-layer address mappings for neighbors to which a packet has been sent recently. In addition to that it maintains information needed for neighbor unreachable detection like the reachability state, etc.

Destination Cache

A set of entries for hosts to whom packets have been sent recently. This includes hosts which are both on-link and off-link. It contains a level of indirection to the neighbor cache.

Prefix List

This is a list of prefixes which define the set of IPv6 address that are on-link. This information is maintained on a per interface basis. Typically this list is built from Router Advertisements received from the router.

Default Router List

A list of routers which will forward packets on behalf of this host. This list will again have a pointer to a neighbor cache entry for the respective router.

A host will use the above data structures while sending a packet to a host. Following is the conceptual algorithm for sending a packet to a unicast destination.

- Before a packet is sent out, the next hop should be determined. Normally, next hop determination is not done on all packets. The results of a next hop determination are stored in the destination cache. The host should first check the destination cache for any entry that matches with the current destination address. If it finds a match, then it proceeds to step c, below.
- If there is no entry for the destination in the destination cache, a longest prefix match is made with all prefixes in the prefix list. If there is a match, the destination is determined to be on-link and the destination address will be considered as the next hop. Otherwise, the next hop is determined from the routing table.
- Once the next hop is determined, the address resolution process and neighbor unreachable detection are done for the next hop. This process is explained in the next section.
- Once the neighbor is known to be reachable, the packet is sent to that destination.

4. Address Resolution and Neighbor Unreachability Detection

Address resolution is a process used to determine the link-layer address of a neighbor. The IPv6 Address to link-layer address mapping found through this process is cached in the Neighbor Cache. Following are the steps involved in Address Resolution.

- First, the neighbor cache is checked for an entry which matches the current destination address. If the entry is not present, the host sends a Neighbor Solicitation Message to the solicited-node multicast group. This multicast address is derived based on the destination IPv6 address and all nodes with the particular IPv6 address are required to join that group.

- b. If a host with the specified IPv6 address is present in the network, it will reply this solicitation with a Neighbor Advertisement Message.
- c. On receiving the Neighbor Advertisement, the node will search for an entry in the neighbor cache for the sender's IPv6 address. A new entry is created in the neighbor cache and the reachability flag is set to REACHABLE.

Once the Address resolution is completed, neighbor unreachability detection will be performed. This process depends on the reachability field of the neighbor cache. An entry in the neighbor cache can have any of the following states:

INCOMPLETE

The address resolution is in progress and the link-layer address of the destination is yet to be determined.

REACHABLE The destination is reachable until recently.

STALE The destination is no longer known to be reachable, but reachability detection need not be made until a packet has to be sent to that destination.

DELAY This state is an optimization that gives additional time for the upper layer protocols to provide the reachability confirmation.

PROBE A reachability confirmation is actively requested by repeatedly sending Neighbor Solicitations.

During neighbor unreachability detection, the node checks for the state in the neighbor cache. If the state for the destination is REACHABLE, the packet is sent. Otherwise, the following steps are taken:

- a. When an address resolution is made on a destination, an entry is created in the neighbor cache for that destination and the reachability state will be set to INCOMPLETE. If the address resolution fails, the entry is deleted.
- b. When the address resolution passes, the entry will be filled with the destination's link-layer address and the state will be set to REACHABLE.
- c. There is a timer maintained called the Reachability timer which will expire the state of an entry in the neighbor cache. Once this timer expires, the reachability state changes from REACHABLE to STALE.
- d. When a packet is being sent to a destination whose state is STALE in the neighbor cache, the node sets the state to DELAY and starts a timer associated with that state. By the time the timer expires if the node received reachability confirmation, the state is set to REACHABLE. Otherwise, it is set to PROBE.
- e. Once the entry's state is in PROBE, the node sends unicast neighbor solicitations to the link-layer address specified in the entry. If it receives a neighbor advertisement in response the state is set to REACHABLE. This solicitation will be sent repeatedly; the maximum number of times is configurable. If the reachability confirmation is not received after maximum solicitations, the entry is deleted from the neighbor cache and the address resolution is done again.

Note: Entries in the neighbor cache can also be created as a result of node receiving unsolicited Neighbor Advertisements, Router Advertisements and Router Solicitations, etc. However, for the entry created under these circumstances the reachability state will always be set to STALE.

5. Redirect Function

A router will send a host a redirect message when it finds that there is a better next-hop router on the same link. This is a requirement for a router.

On receiving a router redirect message, a host should update its destination cache with the new next hop address.

AUTHOR

NDP was developed by the IPng Working Group of the Internet Engineering Task Force.

SEE ALSO

ifconfig(1M), ndp(1M), IPv6(7P), lan(7).

Neighbor Discovery for IPv6, RFC2461, T. Narten et al. NDP Neighbor Discovery Protocol



n

NAME

nfs, NFS - network file system

DESCRIPTION

The Network File System (NFS) allows a client node to perform transparent file access over the network. By using NFS, a client node operates on files residing on a variety of servers and server architectures, and across a variety of operating systems. File access calls on the client (such as read requests) are converted to NFS protocol requests and sent to the server system over the network. The server receives the request, performs the actual file system operation, and sends a response back to the client.

NFS operates in a stateless manner using remote procedure calls (RPC) built on top of an external data representation (XDR) protocol. The RPC protocol enables version and authentication parameters to be exchanged for security over the network.

A server grants access to a specific file system to clients by adding an entry for that file system to the server's `/etc/dfs/dfstab` file.

A client gains access to that file system using the `mount` command to request a file handle for the file system (see `mount(1M)`). (A file handle is the means by which NFS identifies remote files.) Once a client mounts the file system, the server issues a file handle to the client for each file (or directory) the client accesses. If the file is removed on the server side, the file handle becomes stale (dissociated with a known file), and the server returns an error with `errno` set to [ESTALE].

A server can also be a client with respect to file systems it has mounted over the network; however, its clients cannot directly access those file systems. If a client attempts to mount a file system for which the server is an NFS client, the server returns with `errno` set to [EREMOTE]. The client must mount the file system directly from the server on which the file system resides.

The user ID and group ID mappings must be the same between client and server. However, the server maps UID 0 (the superuser) to UID -2 before performing access checks for a client. This process prevents gaining superuser privileges on remote file systems.

RETURN VALUE

Generally, physical disk I/O errors detected at the server are returned to the client for action. If the server is down or inaccessible, the client receives the message:

NFS: file server xxx not responding: still trying.

where `xxx` is the hostname of the NFS server. The client continues resending the request until it receives an acknowledgement from the server. Therefore, the server can crash or power down, and come back up without any special action required by the client. The client process requesting the I/O will block, but remains sensitive to signals (unless mounted with the `nointr` option) until the server recovers. However, if mounted with the `soft` option, the client process returns an error instead of waiting indefinitely.

AUTHOR

`nfs` was developed by Sun Microsystems, Inc.

SEE ALSO

`exportfs(1M)`, `share(1M)`, `mount(1M)`, `mount_nfs(1M)`, `nfsd(1M)`, `mount(2)`, `fstab(4)`, `dfstab(4)`.

NAME

null - null file

DESCRIPTION

Data written on a null special file is discarded.

Reads from a null special file always return 0 bytes.

EXAMPLES

To create a zero-length file, use either of the following:

```
cat /dev/null > file  
cp /dev/null file
```

FILES

/dev/null

STANDARDS CONFORMANCE

null: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1, POSIX.2

NAME

pckt - Packet Mode module for STREAMS pty (pseudo-terminal)

SYNOPSIS

```
#include <sys/stropts.h>
int ioctl(fd_slave, I_PUSH, "pckt");
```

DESCRIPTION

The **Packet Mode** feature for STREAMS pty devices allows the user process on the master side of the pty device to be informed of state changes in the pty. To enable **Packet Mode** in the STREAMS pty device, the user process must push the **pckt** module onto the master side of the pty with a call to the STREAMS **I_PUSH** *ioctl*(2) system call. When the **pckt** module is pushed onto a STREAMS pty master, certain STREAMS messages going upstream on the master side will get packetized so they can be subsequently retrieved by the master side with a **getmsg** function.

When the user process writes data, the **pckt** module passes the message unchanged downstream on to the next module or driver. When the user process reads data or when the **pckt** module receives certain STREAMS message types, it constructs a packet out of the message for forwarding upstream. To construct a message packet, the module creates an **M_PROTO** message. This **M_PROTO** message contains the original message type in the first data block and the original message in as many data blocks as needed. The user process can then retrieve the **M_PROTO** message with a call to the **getmsg**() function.

The **pckt** module packetizes the following STREAMS message types:

M_DATA, **M_IOCTL**, **M_PROTO**, **M_PCPROTO**, **M_FLUSH**, **M_START**, **M_STOP**, **M_STARTI**, **M_STOPI**, **M_READ**.

All other messages are passed unchanged upstream.

If the message is an **M_FLUSH** message, the **pckt** module looks at the flag and takes the following actions:

- If the flag is **FLUSHW**, the module changes it to **FLUSHR** before creating the **M_PROTO** message and passing the message upstream. This prevents the stream head's read queue from being flushed by the original **M_FLUSH** message.
- If the flag is **FLUSHR**, the module changes it to **FLUSHW** before creating the **M_PROTO** message and passing it upstream. To flush the write queues properly, the module also sends an **M_FLUSH** message with the **FLUSHW** flag set.
- If the flag is **FLUSHRW**, the module changes it to **FLUSHW** before creating the **M_PROTO** message and passing it upstream. To flush the write queues properly, the module also sends an **M_FLUSH** message with the **FLUSHW** flag set.

AUTHOR

pckt(7) was developed by HP and OSF.

SEE ALSO

getmsg(2), ioctl(2), ptm(7), pts(7), ldterm(7), ptem(7), streamio(7).

NAME

poll - monitor I/O conditions on multiple file descriptors

SYNOPSIS

```
#include <sys/devpoll.h> #include <fcntl.h>
int open("/dev/poll", O_RDWR);
int write(int filedes, const struct pollfd *buf, size_t nbyte);
int ioctl(int filedes, DP_POLL, struct dvpoll *arg);
int ioctl(int filedes, DP_ISPOLLED, struct pollfd *arg);
```

DESCRIPTION

`/dev/poll` provides an interface to the event port driver allowing a user to synchronously monitor a specific set of conditions associated with a registered set of file descriptors. Poll conditions include the ability to read or write data without blocking and certain exceptional conditions.

Access to `/dev/poll` is provided through the `open()`, `write()`, and `ioctl()` system calls.

The `/dev/poll` event port provides functionality comparable to the `select(2)` and `poll(2)` system calls and supports the following types of file descriptors: network (`AF_INET`) and Unix Domain (`AF_UNIX`) sockets, named FIFO files and pipes, XTI endpoints, and STREAMS devices.

General operations supported by the event port driver are:

- Opening an event port.
- Registering and deregistering file descriptors on an event port.
- Polling registered file descriptors on an event port.
- Retrieving registered poll conditions for a file descriptor.
- Closing an event port.

Opening An Event Port

Each open of the `/dev/poll` device enables an event port from which a different set of file descriptors can be polled. The file descriptor returned by the `open()` system call represents the event port. Users wishing to monitor multiple sets of file descriptors should open the `/dev/poll` device multiple times. For example:

```
int evpfd;
evpfd = open("/dev/poll", O_RDWR);
```

Only the process that performed the `open()` on `/dev/poll` can perform general event port operations. Specifically, any event port file descriptor inherited by a child from its parent or that is received from another process using the Unix Domain Sockets access rights can only be closed. (See `sendmsg` in the `send(2)` man page or the STREAMS `I_FDINSERT` `ioctl` request in the `streamio(7)` man page.)

Registering and Deregistering File Descriptors

An interest set of file descriptors and poll conditions is registered with an event port by using the `write()` system call. By writing an array of `pollfd` structures to an event port the user can register multiple file descriptors in one `write()` service call. The `pollfd` structure and related poll conditions are defined in `<poll.h>`, (included by `<sys/devpoll.h>`). Other flags are defined in the `<sys/devpoll.h>` file. See the `poll(2)` man page for the definition of the poll conditions.

To register a file descriptor, the `fd` field is set to the file descriptor to be registered, and the `events` field is set to one or more poll conditions, such as `POLLIN`. Multiple poll conditions can be `OR`ed together. A given file descriptor can be registered with multiple event ports. Re-registering a file descriptor with the same event port will cause the the specified poll conditions to join the previous conditions for the given file descriptor.

To deregister, `fd` is set to the file descriptor to be deregistered, and `events` is set to `POLLREMOVE`. `POLLREMOVE` is defined in `<sys/devpoll.h>`. `POLLREMOVE` must not be `OR`ed together with any other poll conditions.

When a polled file descriptor is closed, it is automatically deregistered.

Continuing our example, the following registers two file descriptors on the opened event port, `fd1` and `fd2`:

```

struct pollfd pfd[2];
int err;

pfd[0].fd = fd1;
pfd[0].events = POLLIN;
pfd[1].fd = fd2;
pfd[1].events = (POLLIN | POLLRDBAND);
err = write(evpfd, pfd, sizeof(pfd));

```

Polling File Descriptors

Polling an event port's interest set is initiated by calling `ioctl()` specifying the `DP_POLL` request.

The `ioctl` *arg* parameter is a pointer to a `dvpoll` structure, defined in `<sys/devpoll.h>`. It contains the following members:

```

struct dvpoll {
    pollfd_t *dp_fds; /* pollfd[] to be used */
    nfd_t dp_nfds; /* number of pollfd entries */
    int dp_timeout; /* milliseconds or -1 */
}

```

`dp_fds` is a pointer to an array of `pollfd` structures. `dp_nfds` is the maximum number of `pollfd` structures to be returned in that array. `dp_timeout` is the maximum time, in milliseconds, to wait for at least one of the registered poll conditions to be met in the event port.

When one or more registered poll conditions are met for any of the registered file descriptors, `ioctl()` stores the valid poll conditions in the `revents` of each `pollfd` structure in the array, one array element for each active file descriptor. The return value of `ioctl()` is the number of valid `pollfd` structures.

If no poll conditions are met and if `dp_timeout` is `-1`, `ioctl()` sleeps until a poll condition is met on any of the registered file descriptors. If `dp_timeout` is non-negative, `ioctl()` returns after `dp_timeout` milliseconds expires or when a poll condition is met. If the time limit expires, the `ioctl()` return value is `0`.

Retrieving Registered Poll Conditions for a File Descriptor

The registered poll conditions for a given file descriptor in an interest set can be determined by calling `ioctl()` with the `DP_ISPOLLED` request. For example, for file descriptor `fd1`:

```

struct pollfd pfd;
int ispolled;

pfd.fd = fd1;
ispolled = ioctl(evpfd, DP_ISPOLLED, &pfd);

```

If the file descriptor is registered with the event port, the `ioctl()` return value is `1`, and the registered poll conditions are returned in the `events` member of the `pollfd` structure.

The `ioctl()` return value is `0` if the file descriptor is not registered or is not open.

Closing an Event Port

An event port is closed with the `close()` system call specifying the event port file descriptor. All file descriptors registered with that event port are automatically deregistered from that event port.

RETURN VALUES

`open()` returns the event port file descriptor. If the `open()` system call fails, it returns `-1`, and `errno` is set to the error condition.

`write()` returns the number of bytes in the array of the `pollfd` structure that was passed in `buf`. If the `write()` returns `-1`, `errno` is set to the error condition.

`ioctl(DP_POLL)` returns the number of file descriptors for which one or more poll conditions are met. `ioctl(DP_POLL)` returns `0` if a timeout occurred before any poll conditions were satisfied for any of the registered file descriptors.

`ioctl(DP_ISPOLLED)` returns `1` if the file descriptor specified in the `pollfd` structure is registered. `ioctl(DP_ISPOLLED)` returns `0` if the file descriptor is not registered or is closed.

If `ioctl()` returns `-1`, `errno` is set to the error condition.

ERRORS

The following errors are returned by the event port driver.

If `open()` fails, `errno` is set to one of the following values.

[EACCES]	The minor number of the device file name passed to <code>open()</code> is not <code>0</code> .
[EAGAIN]	Allocation of internal data structures failed due to a temporary condition. Calling <code>open()</code> again might succeed.
[EMFILE]	The maximum number of file descriptors allowed for the process is already open.
[ENFILE]	The maximum number of files allowed for the system is already open.
[ENXIO]	Some of the requisite file types are not supported by the <code>/dev/poll</code> driver. See the <code>WARNINGS</code> section below.

If `write()` or `ioctl()` fails, `errno` is set to one of the following values.

[EACCES]	The calling process did not open the event port.
[EBADF]	The <code>filedes</code> argument passed to <code>write()</code> is not an open file descriptor.
[EFAULT]	An attempt was made to access a <code>pollfd</code> structure whose location is outside the process address space.
[EINTR]	A signal interrupted the <code>ioctl(DP_POLL)</code> system call.
[EINVAL]	The <code>nbyte</code> argument passed to <code>write()</code> is less than <code>0</code> .
[ENODEV]	The <code>filedes</code> argument passed to <code>write()</code> is not an event port file descriptor.

EXAMPLES

The following examples show how to use the `/dev/poll` driver to poll for events on network socket file descriptors.

To register a TCP socket file descriptor (`sd`) so that `ioctl(DP_POLL)` will notify the application when a new connection is established or when input data is available:

```
struct pollfd regpfd;
int err;

regpfd.fd = sd;
regpfd.events = POLLIN;
err = write(evpfid, &regpfd, sizeof(regpfd));
```

`POLLRDBAND` should be `ORed` with `POLLIN` if the application needs to distinguish the arrival of out-of-band data.

To wait for events on one or more registered sockets, up to 100 connections:

```
struct pollfd pollpfd[100];
struct dvpoll dvp;
int npoll;

dvp.dp_fds = pollpfd;
dvp.dp_nfds = 100;
dvp.dp_timeout = -1;
npoll = ioctl(evpfid, DP_POLL, &dvp);
```

If a non-blocking write to a socket is incomplete, the following can be used to register the socket so that `ioctl(DP_POLL)` will notify the application when the socket is writable again later. Typically, the socket is already registered to receive input notifications. The following will add the `POLLOUT` notification.

```
struct pollfd regpfd;
int err;

regpfd.fd = sd;
regpfd.events = POLLOUT;
err = write(evpfid, &regpfd, sizeof(regpfd));
```

After the last non-blocking write succeeds, the following should be used to deregister for **POLLOUT**, but continue to be registered for input notifications. Note that **POLLREMOVE** must be used in order to remove the **POLLOUT** registration.

```
struct pollfd regpfd[2];
int err;

regpfd[0].fd = sd;
regpfd[0].events = POLLREMOVE;
regpfd[1].fd = sd;
regpfd[1].events = POLLIN;
err = write(evpf, regpfd, sizeof(regpfd));
```

The following uses **ioctl(DP_ISPOLLED)** to demonstrate how to accomplish the same thing in the more general case, for example, when an application library might not know how the file descriptor is normally registered.

```
struct pollfd regpfd[2];
int err;

regpfd[0].fd = sd;
regpfd[0].events = POLLREMOVE;
regpfd[1].fd = sd;
err = ioctl(evpf, DP_ISPOLLED, &regpfd[1]);
regpfd[1].events &= ~POLLOUT; /* clear POLLOUT */
err = write(evpf, regpfd, sizeof(regpfd));
```

WARNINGS

/dev/poll usually performs better than **select()** and **poll()** especially when the application has registered a very large number of file descriptors. However, in cases where specified conditions are likely to occur simultaneously on a large number of registered file descriptors, performance levels will be diminished.

If **open()** returns **-1** and **errno** is set to [ENXIO], this indicates that some of the necessary system patches have not been installed, and the system administrator must install the File System, Transport, and STREAMS patches that support **/dev/poll** (event ports).

The **write()** system call does not return any error indication if one or more of the file descriptors in the **pollfd** structure could not be registered or deregistered.

If **POLLREMOVE** is **ORed** with other poll conditions in a **pollfd** structure passed to **write()**, **POLLREMOVE** is ignored. The other poll conditions will be **ORed** with any existing poll conditions for the registered file descriptor.

The **ioctl(DP_POLL)** system call returns only the first *dp_nfds* active file descriptors. There is no indication if there are additional active file descriptors.

The **ioctl(DP_ISPOLLED)** system call also returns its result in the **revents** member of the **pollfd** structure, in order to be compatible with the implementation of the **/dev/poll** driver by some other vendors.

The **ioctl(DP_ISPOLLED)** system call does not return any error indication if the file descriptor in the **pollfd** structure is not open.

When an event port is closed, the **close()** system call might take a noticeable amount of time to complete if a very large number of file descriptors is still registered.

AUTHOR

The event port driver was developed independently by HP.

FILES

/dev/poll	driver device file
/sbin/init.d/devpoll	start-up script that creates /dev/poll
/etc/rc.config.d/devpoll	configuration parameters for start-up script

SEE ALSO

ioctl(2), mknod(2), open(2), pipe(2), poll(2), select(2), send(2), socket(2), socketpair(2), write(2), t_open(3).



p

NAME

ps2, ps2kbd, ps2mouse - PS/2 keyboard/mouse device driver and files

SYNOPSIS

```
#include <sys/ps2io.h>
```

DESCRIPTION

The **ps2** driver allows the use of IBM Personal System/2 (PS/2) compatible keyboards and mouse devices on Hewlett-Packard workstations equipped with PS/2 interface hardware.

On systems with a single interface, PS/2 device file names use the following format:

```
/dev/ps2_n
```

where *n* represents the interface port number, ranging from 0 to 15. For example, the device file `/dev/ps2_1` is used to access port one.

On systems with more than one interface, PS/2 device file names use the following format:

```
/dev/ps2_m.n
```

where *m* represents the interface number, and *n* represents the port number. For example, the device file `/dev/ps2_1.2` is used to access port two on interface one.

At boot time, the **ps2** driver scans all interface ports from port zero to the maximum number of ports implemented and attempts to identify attached PS/2 devices. The `/dev/ps2mouse` device file accesses the first mouse detected by **ps2**. The `/dev/ps2kbd` device file accesses the first keyboard detected by **ps2**.

PS/2 devices are classified as "slow" devices. This means that system calls to **ps2** can be interrupted by caught signals (see *signal(5)*).

The mouse may be placed in one of two output modes. In stream mode, the mouse generates a three-byte report packet in response to mouse movement and/or button presses. These reports can be obtained with the `read()` system call (see *read(2)*). In prompt mode, an `ioctl()` request polls the mouse, returning a three-byte report packet in a buffer whose address is passed as an argument to the `ioctl()` call.

PS/2 keyboards return keycodes that represent key-press and key-release events. Use the Internal Terminal Emulator (ITE) to read ASCII characters from PS/2 keyboards. The ASCII terminal interface used by the ITE is described in *termio(7)*.

The **ps2** driver provides a low-level programming interface to PS/2 keyboards and mice. To access these devices in a hardware independent way, use the X Window programming environment.

System Calls

The `open()` system call gives exclusive access to the specified PS/2 device (see *open(2)*). If a port is open, all `open()` calls made on that port will fail with `errno` set to [EBUSY] (see *errno(2)*).

If an `open` is attempted on a nonexistent port, the `open()` call fails with `errno` set to [ENXIO].

If no keyboard is detected at system boot and an `open()` is attempted on `/dev/ps2kbd`, or if no mouse is detected at system boot and an `open()` is attempted on `/dev/ps2mouse`, the `open()` call fails with `errno` set to [ENXIO].

Attempts to open an existing **ps2** port with no device connected will succeed.

Upon a successful `open`, any previously queued input from the device is discarded. Keystrokes are routed to the ITE by default. While a keyboard is open, ITE does not receive keystrokes from that keyboard; until the keyboard device is closed, it has exclusive access to keyboard input.

The file status flags `O_NDELAY` and `O_NONBLOCK` can be set to enable nonblocking reads (see *open(2)*).

`read()` returns bytes from a PS/2 device. HP-UX maintains a 512-byte buffer for each port. When this buffer is full, additional bytes received from the device are discarded.

If enough buffered data is available to satisfy the entire number of bytes requested, the `read()` call completes successfully, having read all of the data requested and returning the number of bytes read.

If there is not enough buffered data available to satisfy the entire request, but at least one byte is available, the `read()` call completes successfully, having read all available data and returning the number of bytes actually read.

If both file status flags **O_NDELAY** and **O_NONBLOCK** are clear and no data is available, the **read()** call blocks until data becomes available or a signal is received.

If the file status flag **O_NDELAY** is set and no data is available, the **read()** call returns zero instead of blocking.

If the file status flag **O_NONBLOCK** is set and no data is available, the **read()** call returns **-1** with **errno** set to [EAGAIN] (see *errno(2)*).

The **write()** system call is not supported by **ps2**.

The **select()** system call can be used to determine if data is currently available to be read from a **ps2** port. Using **select()** for write or for exception conditions always returns a false indication in the file descriptor bit masks (see *select(2)*).

The **ioctl()** system call is used to perform special operations on PS/2 mouse and keyboard devices (see *ioctl(2)*). The set of **ps2** driver **ioctl()** requests are divided into three groups: general requests to both mouse and keyboard, keyboard-specific requests, and mouse-specific requests. Mouse-specific requests used on keyboards, and keyboard-specific requests used on mice, fail, returning **-1** with **errno** set to [EINVAL].

Any **ioctl()** request (except **PS2_PORTSTAT**) used on a port not connected to a PS/2 device will time out, returning **-1** with **errno** set to [EIO].

All **ioctl()** system calls use the following syntax:

```
int ioctl(int fildes, int request, char *arg);
```

All requests that require parameters or return data use a 4-byte unsigned character buffer addressed by the *arg* argument.

The *request* codes that follow are defined in **<sys/ps2io.h>**.

General ioctl() Requests for Both Keyboard and Mouse

PS2_PORTSTAT

Return driver status information.

Two bytes of data are returned in the character buffer addressed by *arg*.

Byte 0, which indicates the type of connected device, can have four possible values:

PS2_NONE	No device is detected.
PS2_MOUSE	Mouse is detected.
PS2_KEYBD	Keyboard is detected.
PS2_UNKNOWN	Unknown device is detected.

Byte 1 contains bit flags for various pieces of driver information. The following bit masks for this byte are defined in the file **<usr/include/sys/ps2io.h>**:

INTERFACE_HAS_ITE	If set, the interface containing this port is used by the Internal Terminal Emulator (ITE) for keyboard input.
PORT_HAS_FIRST_KEYBD	If set, this port is connected to the first keyboard detected by the driver.
PORT_HAS_FIRST_MOUSE	If set, this port is connected to the first mouse detected by the driver.

All other bits are currently unused, and are cleared to zero.

PS2_DISABLE

Disable a PS/2 device.

Further output from the device is prevented by the device itself. This request does not use *arg*. Certain devices perform actions in addition to disabling themselves.

The keyboard resets its internal state to the default state, stops scanning the keys, and waits for further commands.

The mouse stops transmission of reports, and then disables itself.

PS2_ENABLE	Enable a PS/2 device Transmissions from the device are enabled. This request does not use <i>arg</i> .
PS2_IDENT	Identify a PS/2 device. A value identifying the type of device is returned in the 4-byte buffer addressed by <i>arg</i> . The keyboard returns two bytes (<i>arg</i> [0]=0xAB and <i>arg</i> [1]=0x83). The mouse returns one byte (<i>arg</i> [0]=0x00).
PS2_SETDEFAULT	Set the device to its default (power-up) state. The device is returned to its default internal state. This request does not use <i>arg</i> .
PS2_RESET	Reset a PS/2 device. The device is told to execute its internal reset routine and execute its power-up test. The result of the power-up test is returned in the 4-byte buffer addressed by <i>arg</i> . The mouse returns two bytes to indicate a successful reset (<i>arg</i> [0]=0xAA and <i>arg</i> [1]=0x00). The keyboard returns one byte (<i>arg</i> [0]=0xAA).

Keyboard-Specific ioctl() Requests

PS2_SCANCODE	Select the keyboard scancode set The scancode set to be used by the keyboard is passed as the first byte of the buffer addressed by <i>arg</i> . The following are valid values for this byte: SCANCODE_1 Selects scancode set 1. SCANCODE_2 Selects scancode set 2. SCANCODE_3 Selects scancode set 3. GET_SCANCODE Returns the scancode used. When GET_SCANCODE is specified, the scancode used by the keyboard is returned as the first byte of the character buffer addressed by <i>arg</i> . Some keyboards do not support all scancode sets.
PS2_ALL_TMAT	Set all keys to typematic behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used. The typematic rate and delay are set via the PS2_RATEDELAY ioctl() request.
PS2_ALL_MK	Set all keys to make-only behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used.
PS2_ALL_MKBRK	Set all keys to make/break behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used.
PS2_ALL_TMAT_MKBRK	Set all keys to typematic make/break behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used. The typematic rate and delay are set via the PS2_RATEDELAY ioctl() request.
PS2_KEY_TMAT	Set typematic behavior for an individual key. The key code from scancode set 3 for the individual key is passed as the first byte in the character buffer addressed by <i>arg</i> . This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The typematic rate and delay are set via the PS2_RATEDELAY ioctl() request. Because keyboards might be left in a disabled state after this request, the PS2_ENABLE request should be

performed after **PS2_KEY_TMAT**.

PS2_KEY_MAKE

Set make-only behavior for an individual key.

The key code from scancode set 3 for the individual key is passed as the first byte in the character buffer addressed by *arg*. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. Because keyboards might be left in a disabled state after this request, the **PS2_ENABLE** request should be performed after **PS2_KEY_MAKE**.

PS2_KEY_MKBRK

Set make/break for an individual key.

The key code from scancode set 3 for the individual key is passed as the first byte in the character buffer addressed by *arg*. Make/break behavior will be set for this key. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. Because keyboards might be left in a disabled state after this request, the **PS2_ENABLE** request should be performed after **PS2_KEY_MKBRK**.

PS2_INDICATORS

Set the state of keyboard indicators, Num Lock, Caps Lock, and Scroll Lock, according to the value passed in the first byte of the character buffer addressed by *arg*.

The indicators are bit-mapped as follows:

NONE_LED	No indicators active
CAPS_LED	Caps Lock indicator active
NUM_LED	Num Lock indicator active
SCROLL_LED	Scroll Lock indicator active

PS2_RATEDELAY

Set the rate and delay for all typematic keys by specifying the value passed as the first byte in the character buffer addressed by *arg*.

Bits zero through four give the rate. Bits five and six give the delay. Bit seven (the most significant bit) is unused and should be set to zero. The delay in milliseconds is determined by the following equation, where *X* is the numeric value of bits five through six:

$$\text{delay} = (1+X) * 250 \quad (+|- 20\%)$$

The period (interval from one output key code to the next) in seconds is determined by the following equation, where *Y* is the numeric value of bits zero through two, and *Z* is the numeric value of bits three through four:

$$\text{period} = (8+Y) * (2^Z) * 0.00417 \quad (+|- 20\%)$$

The typematic rate (expressed in make codes per second) is one for each period using the above equation. The default typematic rate is 10.9 characters per second. The default delay is 500 milliseconds.

Mouse-Specific ioctl() Requests

PS2_SAMPLERATE

Set the mouse sampling rate used in stream mode by specifying the value passed as the first byte in the character buffer addressed by *arg*.

Seven specific rates are supported:

SAMPLE_10	10 reports/second maximum
SAMPLE_20	20 reports/second maximum
SAMPLE_40	40 reports/second maximum
SAMPLE_60	60 reports/second maximum
SAMPLE_80	80 reports/second maximum
SAMPLE_100	100 reports/second maximum
SAMPLE_200	200 reports/second maximum

The default rate is 100 reports/second maximum. This request updates the mouse sampling rate only in stream mode. If the mouse is in prompt mode, this request is ignored.

PS2_PROMPTMODE

Put mouse into prompt mode.

In prompt mode, the mouse updates its internal values due to movement or button presses, but issues reports only in response to the **PS2_REPORT ioctl()** request. The *arg* parameter is not used.

PS2_REPORT

Obtain a prompt mode mouse report.

This request polls the mouse, obtaining a three-byte report returned in the character buffer addressed by the *arg* parameter. The report has the following format:

Byte 1 A bit map of buttons, signs, and overflows

Bit 0	Left button (1=depressed)
Bit 1	Right button (1=depressed)
Bit 2	Center button (1=depressed)
Bit 3	Always 1
Bit 4	X data sign (1=negative)
Bit 5	Y data sign (1=negative)
Bit 6	X data overflow (1=overflow)
Bit 7	Y data overflow (1=overflow)

Byte 2 X-coordinate data byte

Byte 3 Y-coordinate data byte

The X and Y coordinate values are expressed in two's complement. The scaling behavior specified via the **PS2_2TO1_SCALING ioctl()** request does not apply to reports obtained with the **PS2_REPORT ioctl()** request. **PS2_2TO1_SCALING** affects only reports sent in stream mode.

PS2_STREAMMODE

Put mouse into stream mode.

When in stream mode, the mouse sends a three-byte report whenever the mouse is moved, or a button is pressed or released since the last report. The maximum report rate is set with the **PS2_SAMPLERATE ioctl()** request. If a button is both pressed and then released within a sample interval, it will be reported as pressed at the end of that interval.

The stream-mode reports are obtained via the **read()** system call (see *read(2)*). The format of the report is identical to reports returned by the **PS2_REPORT ioctl()** request described above.

When in stream mode, the **PS2_DISABLE** request must be sent prior to any other **ioctl()** requests.

The *arg* parameter is not used.

PS2_STATUS

Obtain mouse status.

This request polls the mouse, obtaining a three-byte report returned in the character buffer addressed by the *arg* parameter.

The status report has the following format:

Byte 1 A bit map of buttons and mouse internal state

Bit 0	Right button (1=depressed)
Bit 1	Center button (1=depressed)
Bit 2	Left button (1=depressed)
Bit 3	Always 0
Bit 4	If 0, scaling 1:1; if 1, scaling 2:1
Bit 5	If 0, disabled; if 1, enabled
Bit 6	If 0, stream mode; if 1, prompt mode
Bit 7	Always 0

Byte 2 Current resolution setting

Byte 3 Current sampling rate

PS2_RESOLUTION

Set mouse resolution for X and Y coordinate values by specifying the value passed as the first byte in the character buffer addressed by *arg*. Four discrete resolutions are supported:

Resolution	200 DPI	320 DPI
RES_1	1 count/mm	1 count/mm
RES_2	2 count/mm	3 count/mm
RES_3	4 count/mm	6 count/mm
RES_4	8 count/mm	12 count/mm

PS2_2TO1_SCALING Set mouse scaling at 2 to 1. The X and Y coordinate values returned in stream-mode reports are doubled, except for absolute values less than six, which are converted to new values in a nonlinear fashion. The conversion is detailed in this table:

Mouse Internal Value	Converted Value
0	0
+ - 1	+ - 1
+ - 2	+ - 1
+ - 3	+ - 3
+ - 4	+ - 6
+ - 5	+ - 9
All other <i>n</i>	2 * <i>n</i>

This conversion does not apply to reports obtained via the **PS2_REPORT_IOCTL()** request.

The *arg* parameter is not used.

PS2_1TO1_SCALING Set mouse scaling at 1 to 1.

The X and Y values returned in mouse reports are not scaled. This request does not use the *arg* parameter.

ERRORS

If a system call fails, as noted above in the DESCRIPTION section **errno** is set to one of the following values:

[EBUSY]	The specified PS/2 device is already opened.
[EFAULT]	A bad address was detected while attempting to use an argument to a system call.
[EINTR]	A signal interrupted an open() , read() , or ioctl() system call.
[EINVAL]	An invalid parameter was detected by ioctl() .
[EIO]	A hardware or software error occurred while executing an ioctl() system call.
[ENODEV]	write() is not implemented for PS/2 devices.
[ENXIO]	No device is present at the specified address.

EXAMPLES

Assume that *files* is a valid file descriptor for a **ps2** port connected to a keyboard. The first example blinks the keyboard indicators, selects scancode set 3, and loops forever while printing keycodes.

```
#include <sys/ps2io.h>

unsigned char kdbuf[4]; /* buffer for ioctl operations */
unsigned char inchar; /* keycode read */

/* flash the LED indicators */
kdbuf[0] = CAPS_LED | SCROLL_LED | NUM_LED; /* all on */
if( ioctl( files, PS2_INDICATORS, &kdbuf) < 0){
    perror("ioctl PS2_INDICATORS failed");
    exit(1);
}
printf("Indicators on\n");
sleep(1);

kdbuf[0] = NONE_LED; /* all off */
if( ioctl( files, PS2_INDICATORS, &kdbuf) < 0){
    perror("ioctl PS2_INDICATORS failed");
```

```

    exit(1);
}
printf("Indicators off\n");

/* use scancode set 3 */
kbdbuf[0] = SCANCODE_3;
if( ioctl( fildes, PS2_SCANCODE, &kbdbuf) < 0){
    perror("ioctl PS2_SCANCODE failed");
    exit(1);
}

/* identify our scancode set */
kbdbuf[0] = GET_SCANCODE;
if( ioctl( fildes, PS2_SCANCODE, &kbdbuf) < 0){
    perror("ioctl PS2_SCANCODE failed");
    exit(1);
}
printf("Keyboard reports it is using scancode set %d\n",
       (unsigned int) kbdbuf[0]);

/* now, loop forever while printing keycodes */
while( 1){
    read( fildes, &inchar, 1);
    printf("Keycode: %x\n", (unsigned int)inchar);
}

```

The following example puts the mouse in stream mode, sets the report limit to 80 per second, enables the mouse, and then loops forever printing mouse reports. Assume that *fildes* is a valid file descriptor for a ps2 port connected to a mouse.

```

#include <sys/ps2io.h>

unsigned char buf[3];          /* mouse report buffer */
unsigned char ioctl_buf[4];   /* mouse ioctl buffer */

/* first, disable the mouse */
if( ioctl( fildes, PS2_DISABLE) < 0){
    perror("ioctl PS2_DISABLE failed\n");
    exit(1);
}
printf("Mouse disabled\n");

/* Put mouse in stream mode */
if( ioctl( fildes, PS2_STREAMMODE) < 0){
    perror("ioctl PS2_STREAMMODE failed\n");
    exit(1);
}
printf("Mouse in stream mode\n");

/* set samplerate */
ioctl_buf[0] = SAMPLE_80;
if( ioctl( fildes, PS2_SAMPLERATE, ioctl_buf) < 0){
    perror("ioctl PS2_SAMPLERATE failed\n");
    exit(1);
}
printf("Mouse sample rate set to SAMPLE_80\n");

/* Enable mouse */
if( ioctl( fildes, PS2_ENABLE) < 0){
    perror("ioctl PS2_ENABLE failed\n");
    exit(1);
}
printf("Mouse enabled.\n");

```

```

for (;;) {
    if (read(fildes, &buf[0], 1) != 1){
        perror("Read of report byte 1 failed");
        return 1;
    }
    if (read(fildes, &buf[1], 1) != 1){
        perror("Read of report byte 2 failed");
        return 1;
    }
    if (read(fildes, &buf[3], 1) != 1){
        perror("Read of report byte 3 failed");
        return 1;
    }
    printf("mouse: 0x%02x, %d %d\n", buf[0], buf[1], buf[2]);
}

```

AUTHOR

ps2 was developed by the Hewlett-Packard Company.

PS/2 and Personal System/2 are registered trademarks of International Business Machines, Incorporated, in the U.S. and other countries.

FILES

```

/usr/include/sys/ps2io.h
/dev/ps2_[0-15]
/dev/ps2_*. [0-15]
/dev/ps2mouse
/dev/ps2kbd

```

SEE ALSO

close(2), errno(2), fcntl(2), ioctl(2), open(2), read(2), select(2), signal(5), termio(7).

SoftPC User's Guide

SoftPC Installation Guide

Sun System Administrators Guide for the HP700/RX

NAME

ptem - STREAMS pty (pesudo-terminal) Emulation module

SYNOPSIS

```
#include <sys/stropts.h>
int ioctl(fd_slave, I_PUSH, "ptem");
```

DESCRIPTION

ptem is a STREAMS module that emulates a terminal when used in conjunction with **ldterm** (STREAMS line discipline) and **pts** (STREAMS slave pty driver). The **ptem** module normally sits above **pts** and below **ldterm**. The user process must push the **ptem** module onto the slave side of the pty with a call to the STREAMS **I_PUSH** *ioctl*(2) system call before **ldterm** is pushed. **ptem** is responsible for processing all of the terminal **ioctl** commands that are passed downstream from **ldterm** or from **ptm** (STREAMS pty master driver).

ldterm and **ptem** together provide a real terminal behavior for the STREAMS pty slave. However, some of the terminal **ioctl** commands are ignored and cause only an acknowledgement of the command since there is no real terminal or modem in the pty subsystem. In fact, none of the flags in the **c_cflag** field of the **termio** or **termios** structures, (which is used by the **TCSETA** or **TCSETS** *ioctls*, respectively), have any effect on the pty except if the baud rate is set to zero. Setting the baud rate to zero will have the effect of hanging up the pty connection. Similarly, the parity or delay flags in the **c_iflag** field will not have any effect at all on the pty.

As a summary, the **ptem** module performs the following tasks:

- The following **ioctls** are processed, if appropriate, and acknowledged by sending an **M_IOCACK** message upstream when they are received on **ptem**'s write queue:
TCSETA, **TCSETAW**, **TCSETAF**, **TCSETS**, **TCSETSW**, **TCSETSF**, **TCGETA**, **TCGETS**, and **TCSBRK**.
- Keeps track of the window size needed for the **TIOCSWINSZ**, **TIOCGWINSZ**, and **JWINSIZE** *ioctls*.
- Upon receiving any other **ioctl** on its write queue, **ptem** acknowledges them negatively by sending an **M_IOCNAK** message upstream.
- The following **ioctls** are passed downstream by **ptem** after they have been processed:
TCSETA, **TCSETAW**, **TCSETAF**, **TCSETS**, **TCSETSW**, **TCSETSF**, **TCSBRK**, and **TIOCSWINSZ**.
- Any **M_IOCNAK** message that is received on **ptem**'s read queue will be freed in case the **pckt** module is not pushed on the **ptm** and the above **ioctls** get to the pty master STREAMS head, which would then send an **M_IOCNAK** message downstream.
- When **ptem** is opened and all conditions for setting up a controlling terminal are met, it sends an **M_SETOPTS** message (with the **SO_ISATTY** flag set) upstream to the STREAMS head to allocate a controlling terminal.
- Upon receiving an **M_IOCTL** message of type **TCSBRK** on its read queue, **ptem** sends an **M_IOCACK** message downstream and an **M_BREAK** message upstream.
- When an **ioctl** message is received on its write queue to set the baud rate to zero (e.g. **TCSETA** with **CBAUD** set to **B0**), **ptem** sends an **M_IOCACK** message upstream and a zero-length message downstream to be read by the pty master process.
- When an **M_IOCTL** message of type **TIOCSIGNAL** is received on its read queue, **ptem** sends an **M_IOCACK** message downstream and an **M_PCSIG** message upstream with the signal number set to the same value used in the **M_IOCTL** message.
- When an **M_IOCTL** message of type **TIOCREMOTE** is received on its read queue, **ptem** sends an **M_IOCACK** message downstream and an **M_CTL** message (with **ioc_cmd** set to **MC_DO_CANON** or **MC_NO_CANON**) upstream to enable or disable the input processing on **ldterm**.
- When an **M_DELAY** message is received on its read or write queue, **ptem** simply discards the message without any action.
- When an **M_IOCTL** message of type **JWINSIZE** is received on its write queue and if the values in the **jwinsize** structure in **ptem** are not zero, **ptem** sends an **M_IOCACK** message

upstream with the **jwinsize** structure. If the values are zero, **ptem** sends an **M_IOCNAK** message upstream.

- When an **M_IOCTL** message of type **TIOCGWINSZ** is received on its write queue and if the values in the **winsize** structure in **ptem** are not zero, **ptem** sends an **M_IOCACK** message upstream with the **winsize** structure. If the values are zero, **ptem** sends an **M_IOCNAK** message upstream.
- When an **M_IOCTL** message of type **TIOCSWINSZ** is received in its write queue, **ptem** saves the information passed to it in the **winsize** structure and sends an **M_PCSIG** (with the signal number set to **SIGWINCH**) upstream to the pty slave process if the window size is changed.
- When an **M_IOCTL** message of type **TIOCGWINSZ** is received on its read queue and if the values in the **winsize** structure in **ptem** are not zero, **ptem** sends an **M_IOCACK** message downstream with the **winsize** structure. If the values are zero, **ptem** sends an **M_IOCNAK** message downstream.
- When an **M_IOCTL** message of type **TIOCSWINSZ** is received in its read queue, **ptem** saves the information passed to it in the **winsize** structure and sends an **M_PCSIG** (with the signal number is set to **SIGWINCH**) upstream to the pty slave process if the window size is changed.
- All other messages not mentioned above are passed to the next module or driver.

AUTHOR

ptem was developed by HP.

SEE ALSO

ioctl(2), streamio(7), ptm(7), pts(7), ldterm(7).

NAME

ptm - STREAMS master pty (pseudo-terminal) driver

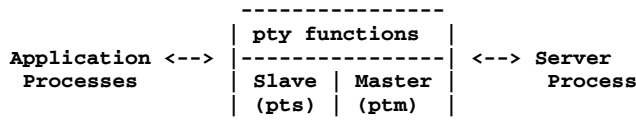
SYNOPSIS

```
#include <sys/stropts.h>
#include <sys/ptyio.h>
#include <sys/strtio.h>

int open("/dev/ptmx", O_RDWR);
```

DESCRIPTION

A pseudo-terminal (pty) consists of a tightly-coupled pair of character devices, called the master device and slave device. The pty master and slave device drivers work together to simulate a terminal connection where the master provides a connection to the pseudo terminal server process and the slave provides a terminal device special file access for the terminal application processes, as depicted below:



The slave driver, **pts** with **ptem** (STREAMS pty emulation module) and **ldterm** (STREAMS line discipline module) pushed on top (not shown for simplicity), provides a terminal interface as described in *termio(7)*. Whereas devices that provide the terminal interface described in *termio(7)* have a hardware device behind them; in contrast, the slave device has another process manipulating it through the master side of the pty. Data written on the master device is given to the slave device as input and data written on the slave device is presented as input on the master device.

In order to use the STREAMS pty subsystem, a node for the master pty driver **/dev/ptmx** and *N* number of slave pty devices must be installed (see *pts(7)* for details on slave pty). There are no nodes in the file system for each individual master device. Rather, the master driver is set up as a STREAMS clone driver (see *clone(7)*) with its major device number set to the major for the clone driver and its minor device number set to the major for the **ptm** driver. The master driver is opened using the **open()** system call with **/dev/ptmx** as the device file parameter. The clone open finds the next available minor number for the master device. The master device is available only if it and its corresponding slave device are not already opened. Only one open is allowed on a master device whereas multiple open are allowed on the slave device. When the master device is opened, the corresponding slave device is automatically locked out (see *pts(7)* on how to unlock the slave and obtain the slave device name). After both the master and slave have been opened, the user has two file descriptors which represent the end points of a full duplex connection composed of two streams. These two streams are automatically connected by the master and slave devices when they are opened. The user may then push the necessary modules on the master and slave streams (e.g., **ptem** and **ldterm**, on **pts** for terminal semantics, and **pckt** on **ptm** for Packet Mode feature).

The master and slave drivers pass all STREAMS messages to their adjacent drivers. Only the **M_FLUSH** message needs some special processing because the read queue of the master is connected to the write queue of the slave and vice versa. Hence, the **FLUSHR** flag is changed to **FLUSHW** flag and vice versa whenever a **M_FLUSH** message travels across the master–slave link. When the master device is closed, an **M_HANGUP** message is sent to the corresponding slave device which will render that slave device unusable. The process on the slave side gets the errno [ENXIO] when attempting a **write()** system call on the slave device but it will be able to read any data remaining on the slave stream. Finally, when all the data have been read, the **read()** system call will return 0 (zero) indicating that the slave can no longer be used. On the last close of the slave device, a zero-length **M_DATA** message is sent to the corresponding master device. When the application on the master side issues a **read()** or **getmsg()** system calls and a 0 is returned. The user of the master device decides whether to close the master device file which will dismantle the streams on the master side. If the master device remains opened, the corresponding slave device can be opened and used again by another user.

Unlike the slave device, the master device does not act like a terminal. If **O_NDELAY** or **O_NONBLOCK** is set, a read on the master device returns -1 with errno set to [EAGAIN] if no data is available, and a write returns -1 with errno set to [EAGAIN] if there is internal flow control on the stream.

The master **ptm** driver supports the following **ioctl()** requests:

ISPTM Determines whether the file descriptor is that of an open master device. On success, it returns the major and minor number (type `dev_t`) of the master device which can be used to determine the name of the corresponding slave device. On failure, it returns `-1` with `errno` set to `[EINVAL]`. **ISPTM** on HP-UX can return valid device number with negative value. For example, with major number of the STREAMS pty master being `0x9c`, **ICPTM** will return `0x9C000000` which is a negative number. Therefore, it is imperative that applications check for an explicit `-1` instead of "`< 0`" (less than 0) on the return value.

ISPTM is used by functions `grantpt()`, `unlockpt()`, and `ptsname()`. User applications normally do not need to invoke this `ioctl`. The format of this `ioctl` is:

```
int ioctl(master_fd, ISPTM, 0)
```

UNLKPT Unlocks the master and the corresponding slave devices. On success, it returns 0. On failure, it returns `-1` with `errno` set to `[EINVAL]`. **UNLKPT** is used by function `unlockpt()`. User applications normally do not need to invoke this `ioctl`. The format of this `ioctl` is:

```
int ioctl(master_fd, UNLKPT, 0)
```

TIOCREMOTE This `ioctl` puts the STREAMS pty in and out of Remote Mode. When Remote Mode is on, input data will be flow-controlled and passed through `ldterm` without any input processing regardless of the terminal mode. When the pty master driver receives this `ioctl`, it will send an `M_CTL` message downstream to `ldterm` via `ptm`, `pts`, and `ptem`. The command in the `M_CTL` message is set to `MC_NO_CANON` or `MC_DO_CANON` depending whether to turn on or off the Remote Mode. The format of this `ioctl` is:

```
int ioctl(master_fd, TIOCREMOTE, argument)
```

where the argument is set to 1 to turn on Remote Mode and 0 to turn it off. Remote Mode is normally used when doing remote line editing in a window manager, or whenever flow-controlled input is required. Each write to the master device produces a record boundary for the process reading the slave devices. In normal usage, a write of data is like the data typed as a line on the terminal; a write of 0 (zero) bytes is like typing an **EOF** (End-of-File) character.

TIOCSIGNAL This `ioctl` allows the master process to send a signal to the slave process. The format of this `ioctl` is:

```
int ioctl(master_fd, TIOCSIGNAL, argument)
```

where the argument is the signal number as defined in the header file `<sys/signal.h>`. For example the master process can send an **SIGINT** signal to the slave process by doing:

```
ioctl(master_fd, TIOCSIGNAL, SIGINT)
```

AUTHOR

`ptm` was developed by HP and OSF.

FILES

`/dev/ptmx` Streams pty master clone device
`/dev/pts/N` Streams pty slave devices ($0 \leq N < \mathbf{NSTRPTY}$), where **NSTRPTY** is a kernel tunable parameter which can be changed via SAM.

SEE ALSO

`insf(1M)`, `getmsg(2)`, `ioctl(2)`, `open(2)`, `read(2)`, `write(2)`, `grantpt(3C)`, `ptsname(3C)`, `unlockpt(3C)`, `clone(7)`, `ldterm(7)`, `pckt(7)`, `ptem(7)`, `pts(7)`, `streamio(7)`, `termio(7)`.

NAME

pts - STREAMS slave pty (pseudo-terminal) driver

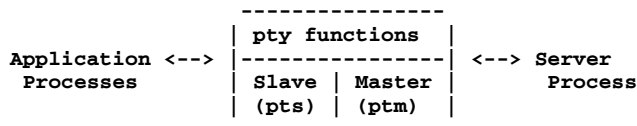
SYNOPSIS

```
#include <sys/stropts.h>
#include <sys/termios.h>
#include <sys/strtio.h>

int open("/dev/pts/N", O_RDWR);
```

DESCRIPTION

A pseudo-terminal (pty) consists of a tightly-coupled pair of character devices, called the master device and slave device. The pty master and slave device drivers work together to simulate a terminal connection where the master provides a connection to the pseudo terminal server process and the slave provides a terminal device special file access for the terminal application processes, as depicted below:



The slave driver, **pts** with **ptem** (STREAMS pty emulation module) and **ldterm** (STREAMS line discipline module) pushed on top (not shown for simplicity), provides a terminal interface as described in *termio(7)*. Whereas devices that provide the terminal interface described in *termio(7)* have a hardware device behind them; in contrast, the slave device has another process manipulating it through the master side of the pty. Data written on the master device is given to the slave device as input and data written on the slave device is presented as input on the master device.

In order to use the STREAMS pty subsystem, a node for the master pty driver **/dev/ptmx** and *N* number of slave pty devices must be installed (see *ptm(7)* for more details on master pty). When the master device is opened, the corresponding slave device is automatically locked out. No user can open that slave device until its permissions are changed (via the **grantpt()** function) and the device is unlocked (via the **unlockpt()** function). The user then call the **ptsname()** function to obtain the name of the slave device and invoke the **open()** system call to open the slave device. Although only one open is allowed on a master device, multiple opens are allowed on the slave device. After both the master and slave have been opened, the user has two file descriptors which represent the end points of a full duplex connection composed of two streams that are automatically connected by the master and slave devices when they are opened. The user may then push the desired modules (for example, **ptem** and **ldterm**, on **pts** for terminal semantics and **pckt** on **ptm** for Packet Mode feature).

The master and slave drivers pass all STREAMS messages to their adjacent drivers. Only the **M_FLUSH** message needs some special processing because the read queue of the master is connected to the write queue of the slave and vice versa. For example, the **FLUSHR** flag is changed to **FLUSHW** flag and vice versa whenever a **M_FLUSH** message travels across the master-slave link. When the master device is closed, an **M_HANGUP** message is sent to the corresponding slave device which will render that slave device unusable. The process on the slave side gets the **errno** [ENXIO] when attempting a **write()** system call to the slave device file but it will be able to read any data remaining in the slave stream. Finally, when all the data has been read, the **read()** system call will return 0, indicating that the slave can no longer be used. On the last close of the slave device, a zero-length **M_DATA** message is sent to the corresponding master device. When the application on the master side issues a *read(2)* or *getmsg(2)* system calls, a 0 (zero) is returned. The user of the master device may decide to close the master device file, which dismantles the stream on the master side. If the master device remains opened, the corresponding slave device can be opened and used again by another user.

EXAMPLES

The following example shows how a STREAMS pty master and slave devices are typically opened.

```
int fd_master, fd_slave;
char *slave;
...
fd_master = open("/dev/ptmx", O_RDWR);
grantpt(fd_master);
unlockpt(fd_master);
```

```
slave = ptsname(fd_master);
fd_slave = open(slave, O_RDWR);
ioctl(fd_slave, I_PUSH, "ptem");
ioctl(fd_slave, I_PUSH, "ldterm");
```

AUTHOR

pts was developed by HP and OSF.

FILES

/dev/ptmx Streams pty master clone device
/dev/pts/N Streams pty slave devices ($0 \leq N < \mathbf{NSTRPTY}$), where **NSTRPTY** is a kernel tunable parameter which can be changed via SAM (see *sam*(1M)).

SEE ALSO

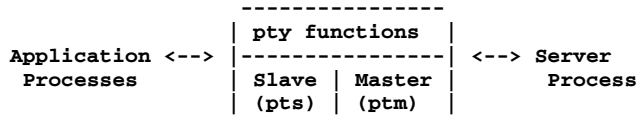
insf(1M), *sam*(1M), *getmsg*(2), *ioctl*(2), *open*(2), *read*(2), *write*(2), *grantpt*(3C), *ptsname*(3C), *unlockpt*(3C), *ldterm*(7), *ptem*(7), *ptm*(7), *streamio*(7), *termio*(7).

NAME

pty - pseudo-terminal driver

DESCRIPTION

The **pty** driver provides support for a device-pair termed a pseudo terminal. A pseudo terminal is a pair of character devices, a master device and a slave device. The slave device provides to application processes an interface identical to that described in *termio(7)*. Unlike all other devices that provide the interface described in *termio(7)*, the slave device does not have a hardware device behind it. Instead, it has another process manipulating it through the master half of the pseudo terminal. Thus anything written on the master device is given to the slave device as input, and anything written on the slave device is presented as input on the master device.

**Open and Close Processing**

The slave side of the **pty** interprets opening or closing the master side as a modem connection or disconnection on a real terminal. Only one open to the master side of a **pty** is permitted. An attempt to open an already open master side returns **-1** and sets the external variable **errno** to [EBUSY]. An attempt to open the master side of a **pty** that has a slave with an open file descriptor returns **-1** and sets **errno** to [EBUSY]. The potential problem of **pty**s being found busy at opens can be avoided by using the *clone open* functionality discussed in the next section.

An attempt to open a nonexistent **pty** returns **-1** and sets **errno** to [ENXIO]. If **O_NDELAY** is not specified, opens on the slave side hang until the master side is opened. If **O_NDELAY** is specified, opens on the slave side return error if the master side is closed. Any **ioctl()** or **write()** request made on the slave side of a **pty** after the master side is closed returns **-1** and sets the external variable **errno** to [EIO]. A **read()** request made on the slave side of a **pty** after the master side is closed returns 0 bytes. Closing the master side of a **pty** sends a **SIGHUP** hangup signal to the tty process group number of the corresponding slave side and flushes pending input and output.

Clone Open

In typical **pty** usage, there is no preference among **pty** pairs. Thus, it is useful to be able to issue a single **open()** that internally opens any available **pty**. An open on **/dev/ptym/clone** returns an open file descriptor of a free master **pty** device. If there are no free devices, the open returns **-1** and sets **errno** to [EBUSY]. The name of the slave device corresponding to the opened master device can be found through a **ptsname()** request.

Processing ioctl() Requests

By default, any **ioctl()** request defined by *termio(7)* is recognized by both the master and slave sides of a **pty**. These **ioctl()** requests are processed by the **pty** driver as specified by *termio(7)*. In addition, the **ioctl()** requests defined below are recognized by the master side of a **pty**. The slave side only recognizes **ioctl()** requests defined by *termio(7)*. An **ioctl()** request made on the slave side of a **pty** after the master side is closed returns **-1** and sets the external variable **errno** to [EIO]. An **ioctl()** request not recognized by the **pty** returns **-1** and sets the external variable **errno** to [EINVAL]. Note that some of the master-side-only **ioctl()** requests affect which **ioctl()** requests are recognized by the master and slave side of the **pty**. These master-side-only **ioctl()** requests also affect the way recognized **ioctl()** requests, **open()** requests, and **close()** requests are processed by the **pty** driver.

The following **ioctl()** requests, defined in **<sys/ptyio.h>**, apply only to the master side of **pty**:

TIOCSIGSEND

Cause a signal to be sent from the slave side of the **pty** to the current tty process group of the slave side. The value of the parameter is taken to be the signal number sent. An [EINVAL] error is returned and no signal is sent if the specified signal number does not refer to a legitimate signal (see *signal(5)*). Note that this request allows the server process to send signals to processes not owned by the same user ID.

TIOCTTY

Enable or disable all **termio** processing by a **pty**. **termio** processing is enabled if the **int** addressed by *arg* is nonzero and disabled if the **int** addressed by *arg* is zero. By

default, **termio** processing is enabled. **termio** processing refers to processing of input and output described by *termio(7)* (such as tab expansion), as well as the processing of the **ioctl()** requests described by *termio(7)*. When disabled, all input and output data is passed through the **pty** without modification. Issuing a **TIOCTTY ioctl()** request flushes all data buffered in the pseudo terminal and releases any processes blocked waiting for data. Enabling and disabling **TIOCTTY** affects the operation of the following **ioctl()** requests: **TIOCPKT**, **TIOCREMOTE**, **TIOCBREAK**, **TIOCSTOP**, **TIOCSTART**, **TIOC-TRAP**, and **TIOCMONITOR**.

When **TIOCTTY** is enabled, all **termio ioctl()** requests execute as specified in *termio(7)*, regardless of the side from which the **ioctl()** request is made. When **TIOCTTY** is disabled, master side **termio ioctl()** requests set and return the the external variable **errno** to [EINVAL]. Slave side **termio ioctl()** requests are processed like any other **ioctl()** request when **TIOCTTY** is disabled. In particular, slave side **termio ioctl()** requests set and return the external variable **errno** to [EINVAL] when both **TIOCTTY** and **TIOCTRAP** are disabled. (See the discussion of **ioctl()**, **open()**, and **close()** trapping below). **ioctl()** requests not defined by *termio(7)* are not affected by the state of **TIOCTTY**.

Data written through a pseudo terminal with **TIOCTTY** disabled is handled in a manner similar to data flowing through a pipe. A write request blocks in the **pty** until all data has been written into the **pty**. A read request blocks if there is no data available unless the **O_NDELAY** flag is set (see *fcntl(2)*). When data is available to be read, the read request returns whatever is available, and does not wait for the number of bytes requested to be satisfied. The number of bytes a **pty** can contain in its internal memory is implementation dependent, but is at least 256 bytes in each direction. For example, a write on the slave side of a **pty** of 1024 bytes might be read on the master side by four read requests returning 256 bytes each. The size of the chunks of data that are read is not guaranteed to be consistent, but no data is lost.

The following **ioctl()** requests, defined in `<sys/ptyio.h>`, apply only to the master side of a **pty**. In particular, these **ioctl()** requests enable/disable specific modes of **pty** driver operation. These **ioctl()** requests work in series with **TIOCTTY**; that is, the mode must be enabled by its **ioctl()** request and **TIOCTTY** must be enabled for the mode to operate. The mode can be enabled or disabled regardless of the state of **TIOCTTY**.

TIOCPKT Enable or disable packet mode. Packet mode is enabled if the **int** addressed by *arg* is nonzero and disabled if the **int** addressed by *arg* is zero. By default, packet mode is disabled. When applied to the master side of a pseudo terminal, each subsequent **read()** from the master side returns data written on the slave part of the pseudo terminal preceded by a zero byte (symbolically defined as **TIOCPKT_DATA**), or a single byte reflecting control status information. The value of such a status byte is composed of zero or more bit flags:

TIOCPKT_FLUSHREAD

The read queue for the slave side has been flushed.

TIOCPKT_FLUSHWRITE

The write queue for the slave side has been flushed.

TIOCPKT_STOP

Data flowing from the slave side of the **pty** to the master side has been stopped by means of **^S**, **TIOCSTOP**, or **TCXONC**.

TIOCPKT_START

Data flowing from the slave side of the **pty** to the master side has been restarted.

TIOCPKT_DOSTOP

Stop and start characters have been set to **^S** or **^Q**.

TIOCPKT_NOSTOP

Stop and start characters are set to something other than **^S** or **^Q**.

TIOCREMOTE Enable or disable remote mode. Remote mode is enabled if the **int** value of *arg* is nonzero and disabled if the **int** value of *arg* is zero. By default, remote mode is disabled. Remote mode is independent of packet mode. This mode causes input to the pseudo terminal to be flow controlled and not input edited (regardless of the terminal mode). Each write to the master side produces a record boundary for the process reading the slave side. In normal

usage, writing data is like typing the data as a line on a terminal; writing zero bytes is equivalent to typing an end-of-file character (that is, the EOF character as defined in *termio(7)*). The data read by the slave side is identical to the data written on the master side. Data written on the slave side and read on the master side with **TIOCREMOTE** enabled is still subject to the normal *termio(7)* processing. **TIOCREMOTE** can be used when doing remote line editing in a window manager, or whenever flow-controlled input is required. Issuing a **TIOCMONITOR ioctl()** request flushes all data buffered in the pseudo terminal.

The following **ioctl()** requests, defined in `<sys/ptyio.h>`, apply only to the master side of **pty**. In particular, these **ioctl()** requests are only recognized when **TIOCTTY** is enabled. When **TIOCTTY** is disabled, these **ioctl()** requests set and return the external variable **errno** to [EINVAL].

- TIOCBREAK** Cause a break operation to be done on the slave side of the **pty**, as if a user had pressed the break key on a real terminal. Takes no parameter.
- TIOCSTOP** Stop data flowing from the slave side of the **pty** to the master side (equivalent to typing **^S**). Takes no parameter.
- TIOCSTART** Restart output (stopped by **TIOCSTOP** or by typing **^S**). Takes no parameter.

Flow-Control Input and Output Processing

The following terms are used to describe the flow of data through pseudo terminals. INPUT refers to data flowing from the master side of a **pty** to the slave side. OUTPUT refers to data flowing from the slave side of a **pty** to the master side.

When packet mode (**TIOCPKT**) is disabled and INPUT is stopped (see IXOFF, input modes, in *termio(7)*), the next **read()** from the master side of a **pty** returns a STOP character. When INPUT is restarted, the next **read()** from the master side returns a START character. If packet mode (**TIOCPKT**) is enabled, the STOP or START character is preceded by a data packet indicator (**TIOCPKTDATA**). **select()** should be used by the master-side server before each **write()** request to properly handle INPUT flow control (see *select(2)*).

When INPUT flow control is enabled, **write()** and **select()** are handled as follows: Write-selects on the master side of a **pty** return true only if INPUT has not been stopped. If INPUT becomes stopped while data is being written into the master side of a **pty**, the write returns with the number of bytes written before INPUT was stopped. Writes done after INPUT is stopped return immediately with zero bytes written.

When packet mode (**TIOCPKT**) is disabled and OUTPUT is stopped (see IXON, input modes in *termio(7)*), each subsequent **read()** from the master side of a **pty** returns with no data read. When OUTPUT is restarted, each subsequent **read()** from the master side returns data written on the slave side. If packet mode (**TIOCPKT**) is enabled, the first **read()** after OUTPUT has been stopped returns a **TIOCPKTSTOP** packet. All subsequent reads from the master side while OUTPUT is stopped returns a **TIOCPKTDATA** packet with no data. When OUTPUT is restarted, the next **read()** from the master side returns a **TIOCPKTSTART** packet. All subsequent reads from the master side return data written on the slave side preceded by a **TIOCPKTDATA** packet. **select()** should be used by the master-side server before each **read()** to properly handle OUTPUT flow control. Otherwise, reads from the master side of a **pty** will not be prevented when OUTPUT is stopped.

Trapping **ioctl()**, **open()**, **close()** Requests

When trapping is enabled, the master side is notified when the application on its slave side makes an **ioctl()**, **open()**, or **close()** request. For trapped **ioctl()** and **open()** requests, the slave side is blocked (that is, the request does not complete) until the server on its master side acknowledges the trapped request. For trapped **close()** requests, the slave side does not block for an acknowledgement.

select() should be used by the master side server to receive notification of trapped **ioctl()**, **open()**, and **close()** requests. When one of these requests is trapped, the **select()** returns with an "exceptional condition" indicated for the slave side's file descriptor. Other mechanisms for receiving notification of trapped requests are defined below, but these mechanisms should be used only if **select()** is not available.

When trapping is disabled (default condition), unrecognized slave **ioctl()** requests return an error, with the external variable **errno** set to [EINVAL]. The only **ioctl()** requests recognized by the slave side are those defined by *termio(7)* and only when **TIOCTTY** is enabled. When **TIOCTTY** is disabled, no **ioctl()** requests are recognized by the slave side. If trapping is enabled and the master side closes, trapping is disabled. If the master closes during the middle of a handshake with the slave, the handshake

is done automatically.

Trapping occurs in two forms that are identified by the `ioctl()` requests that enable or disable them — **TIOCTRAP** and **TIOCMONITOR**. These two forms are distinguished by the types of requests they affect and by the capabilities they provide. Trapping `open()` and `close()` requests is enabled or disabled by **TIOCTRAP**. Trapping `ioctl()` requests not defined by *termio(7)* are enabled or disabled by **TIOCTRAP**. Trapping `ioctl()` requests defined by *termio(7)* are enabled or disabled by **TIOCTRAP** only when **TIOCTTY** is also disabled. When **TIOCTTY** is enabled, trapping `ioctl()` requests defined by *termio(7)* are enabled or disabled by **TIOCMONITOR**. Briefly, both **TIOCTRAP** and **TIOCMONITOR** trapping allow the server on the master side to examine the request's parameters, the pid making the request, etc. In addition, **TIOCTRAP** trapping allows the server to modify the parameters and return values of an `ioctl()` request.

The following `ioctl()` calls apply only to the master side of a **pty** and pertain to trapping `ioctl()`, `open()`, and `close()` requests. They are defined in `<sys/ptyio.h>`:

TIOCTRAP Enable or disable trapping of `ioctl()`, `open()`, and `close()` requests made by the application on the slave side of a **pty**. Trapping is enabled if the `int` addressed by `arg` is nonzero and disabled if the `int` addressed by `arg` is zero. By default, **TIOCTRAP** trapping is disabled.

TIOCTRAPSTATUS

Check for a pending `ioctl()`, `open()`, or `close()` trap. The argument points to an `int` that is set to one if a trap is pending and to zero if nothing is pending. Use **TIOCTRAPSTATUS** when the preferred method of a `select()` "exceptional condition" is not available.

TIOCREQCHECK

Return the trapped `ioctl()`, `open()`, or `close()` information to the master side. Use **TIOCREQCHECK** in response to either a `select()` "exceptional condition" or a **TIOCTRAPSTATUS** indicating that a trap is pending. A **TIOCREQCHECK** reads the pending `ioctl()`, `open()`, or `close()` information into the memory pointed to by the `arg` of **TIOCREQCHECK**. The information takes the form of the following `request_info` structure, defined in `<sys/ptyio.h>`:

```
struct request_info {
    int request;
    int argget;
    int argset;
    pid_t pgrp;
    pid_t pid;
    int errno_error;
    int return_value;
};
```

All elements of `request_info` refer to the slave side of the **pty** and include the following:

request	The <code>ioctl()</code> command received.
argget	The <code>ioctl()</code> request applied to master side to receive the trapped <code>ioctl()</code> structure, if one exists (a zero value means there is none). (When nonzero, argget is a TIOCARGGET request with the size field precomputed.)
argset	The <code>ioctl()</code> request applied to master side to send back the resulting <code>ioctl()</code> structure, if one exists (a zero value means there is none). (When nonzero, argset is a TIOCARGSET request with the size field precomputed.)
pgrp	The process group number of the process doing the operation.
pid	The process ID of the process doing the operation.
errno_error	The <code>errno</code> external variable error code (initialized to zero) returned by <code>ioctl()</code> on the slave side. When open error mode is enabled,

errno_error can be used to return an error for trapped slave **pty open()** requests. See the discussion of the **TIOCSMODES ioctl()** for further information on open error mode.

return_value

The success value (initialized to zero) returned by **ioctl()** on the slave side when **errno_error** is not set.

When the **ioctl()** argument received on the slave side is not a pointer, its value is stored as four bytes retrievable with an **ioctl()** request to the master side equal to **argget**.

When an **open()** or **close()** is being passed, **request** is set to **TIOCOOPEN** or **TIOCCLOSE**, respectively. For **TIOCOOPEN** and **TIOCCLOSE**, both **argget** and **argset** are zero because there is no **ioctl()** structure. When **TIOCTTY** is enabled, the *termio(7)* definition of open/close is executed first before being passed to the master side. Note that while all opens are trapped, only the last close on a particular inode for a **pty** slave side is trapped by the **pty**.

A **TIOCREQCHECK** returns the external variable **errno** error [EINVAL] if no **ioctl()**, **open()**, or **close()** trap is pending. Accordingly, a **TIOCREQCHECK** that returns [EINVAL] in response to a **select()** "exceptional condition" indicates that the trapped **ioctl()**, **open()**, or **close()** request was terminated by a signal after **select()** returned.

TIOCREQGET Identical to **TIOCREQCHECK** except when no **ioctl()**, **open()**, or **close()** trap is pending. A **TIOCREQGET** blocks until a slave side **ioctl()**, **open()**, or **close()** is trapped; whereas a **TIOCREQCHECK** returns [EINVAL]. Use **TIOCREQGET** when neither the preferred method of a **select()** "exceptional condition" nor the master side **ioctl()** **TIOCTRAPSTATUS** is available.

TIOCREQSET Complete the handshake started by a previous **TIOCREQCHECK** or **TIOCREQGET**. The argument should point to the **request_info** structure, as defined by the **TIOCREQCHECK**.

Before doing this **ioctl()** request to complete the handshake, the server should set **errno_error** to an external variable **errno** error value to be passed back to the slave side. If there is no error, **errno_error** can be left alone because the **pty** initializes it to zero. Also, when there is no error, **return_value** should be set if other than a zero result is desired. The server can set **return_value** and **errno_error** if the trapped request is an **ioctl()** and may set **errno_error** for a trapped **open()** if open error mode is enabled. Setting either **return_value** or **errno_error** for a trapped **close()** affects neither the return value of the request nor the external variable **errno** value of the slave side. Setting either **return_value** or **errno_error** for a trapped **open()** affects neither the return value of the request nor the external variable **errno** value of the slave side unless open error mode is enabled. Open error mode allows the server to return an error to a trapped slave **open()** by setting **errno_error**. Unlike **ioctl()** requests, setting **return_value** never affects slave **pty open()** requests. Further, setting either **return_value** or **errno_error** does not cause **TIOCREQSET** to return an error to the server.

If the **TIOCREQSET** request is made and the request value in the passed **request_info** structure does not equal the trapped value, the external variable **errno** is set and returned as [EINVAL]. [EINVAL] is also returned if there are no trapped **ioctl()**, **open()**, or **close()** requests. If the trapped request has been interrupted by a signal between the time that the server has done the **TIOCREQGET** and the **TIOCREQSET**, the **TIOCREQSET** request returns [EINVAL].

TIOCGFLAGS Get the file status flags associated with a trapped request. Upon successful return, the **ioctl()** returns in an integer referenced by *arg* the file status flags for the trapped request. The flag definitions in *<sys/file.h>* can be used to interpret the flags. If no trap is currently pending, the **TIOCGFLAGS ioctl()** returns an error with the external variable **errno** set to [EINVAL].

TIOCMONITOR

Enable or disable read-only trapping of *termio* **ioctl()** requests. **TIOCMONITOR**

trapping is enabled if the **int** addressed by *arg* is nonzero and disabled if the **int** addressed by *arg* is zero. By default, **TIOCMONITOR** trapping is disabled. **TIOCMONITOR** works in series with **TIOCTTY**; that is, the **TIOCMONITOR** trapping must be enabled and **TIOCTTY** must be enabled for **termio** **ioctl()** requests to be trapped by **TIOCMONITOR**. **TIOCMONITOR** trapping can be enabled or disabled regardless of the state of **TIOCTTY**.

When **TIOCTTY** is disabled, **termio** **ioctl()** requests are not trapped by **TIOCMONITOR**. However, **ioctl()** requests are trapped by **TIOCTRAP** if **TIOCTTY** is disabled and **TIOCTRAP** is enabled. **TIOCTRAP** trapping allows the master side server to modify the parameters and return values of an **ioctl()** request, whereas **TIOCMONITOR** trapping does not.

TIOCMONITOR trapping allows the server on the master side to know when characteristics of the line discipline in the **pty** are changed by an application on its slave side. The mechanism for handshaking **termio** requests trapped by **TIOCMONITOR** is the same as the mechanism described above for requests trapped by **TIOCTRAP**. (It is recommended that **termio** **ioctl()** requests be used on the master side to interrogate the configured state of the line discipline in the **pty**. This compensates for the window of time before **TIOCMONITOR** is enabled, when **termio** **ioctl()** requests are not trapped.)

When using **select()** on the master side of a **pty**, the "exceptional condition" refers to an **open()**, **close()**, or **ioctl()** request pending on the slave side, while "ready for reading or writing" indicates that the device can be read from or written to successfully.

Of the **ioctl()** requests subject to being trapped, only one-per-pty can be handled at a time. This means that when an application does a non-**termio** **ioctl()** request to the slave side, all other **ioctl()** requests to the same **pty** slave side are blocked until the first one is handshaked back by the master side. (**ioctl()** requests that are not trapped, such as **termio** when **TIOCTTY** is enabled and **TIOCMONITOR** is disabled, are not blocked.) This permits the implementation of indivisible operations by an **ioctl()** call on the slave side that is passed to the server process.

In summary, the following method of handling trapped **ioctl()**, **open()**, and **close()** requests is preferred:

1. Call **select()**. This system call blocks the master side until a slave side **ioctl()**, **open()**, or **close()** request is trapped.
2. Make **TIOCREQCHECK** **ioctl()** request. This step returns information about a trapped **ioctl()**, **open()**, or **close()** request. If **TIOCREQCHECK** returns the external variable **errno** error [EINVAL], loop back to the **select()** call.
3. Make **argget** **ioctl()** request. This optional step is used if **argget** is nonzero and the server wants to do more than just reject the trapped slave **ioctl()** request.
4. Make **argset** **ioctl()** request. This optional step is done if **argset** is nonzero and the server wants to pass back a modified **ioctl()** structure. It is done after the trapped **ioctl()** request is processed via the server on the master side.
5. Set **errno_error** and **return_value**. If the trapped request is an **ioctl()**, set **errno_error** appropriately. If the appropriate value for **errno_error** is zero, **return_value** must be set. If open error mode is enabled, set **errno_error** to a nonzero value to return an error to a trapped **open()** request.
6. Make **TIOCREQSET** **ioctl()** request. This step completes the trapped **ioctl()**, **open()**, or **close()** request.

While a process is waiting in the slave side of the **pty** for the server to complete a handshake, it is susceptible to receiving signals. The following master side **ioctl()** request allows the server process to control how the **pty** responds when a signal attempts to interrupt a trapped **open()** or **ioctl()** request:

TIOCSIGMODE

Set the signal handling state of the **pty** to the mode specified as the argument. The mode can have three values, which are **TIOCSIGBLOCK**, **TIOCSIGABORT**, and **TIOCSIGNALNORMAL**.

TIOCSIGBLOCK

Cause some signals to be postponed that are destined for the slave-side process whose **open()** or **ioctl()** request is trapped. Signals are postponed if they would

otherwise cause the process to jump to an installed signal handler. Signals are not postponed if they would otherwise cause the process to abort or if they are being ignored. When the server process completes the handshake by means of the **TIOCREQSET ioctl()** request, the process returns to the calling program and any pending signals are then acted upon. Any signals that the user has blocked by means of **sigblock()** continues to be blocked.

TIOCSIGABORT

Prevent a trapped **open()** or **ioctl()** request from being restarted. The server process sets this mode when it wants the interrupted requests to return to the calling program with an [EINTR] error.

TIOCSIGNORMAL

This is the default mode of the **pty**. If a signal interrupts a trapped **open()** or **ioctl()** request, the user's signal handler routine can specify whether the request is to be restarted. If the request is restarted, it executes again from the beginning and the server has to make another **TIOCREQSET** request to start the handshake over again. If the user's signal handler routine specifies that the interrupted request should not be restarted, the request returns to the calling program with [EINTR] upon completion of the signal handler. Note that the restarted request is not necessarily the very next one to be trapped.

The following **ioctl()** requests, defined in **<sys/ptyio.h>**, provide a mechanism to get and set **pty** modes. Five of the modes can also be manipulated using other **ioctl()** requests discussed previously. See the bit definitions for the **ioctl()** equivalents. The effect of enabling or disabling them by either means is identical. Commonly, an application would use the **TIOCGMODES ioctl()** to get the **pty** modes currently in effect, set or clear the bits for the modes being changed, and issue a **TIOCGMODES ioctl()** to effect the desired change.

TIOCGMODES Get the **pty** modes currently in effect. The **ioctl()** returns in a long referenced by *arg* bits indicating the states of various **pty** modes. If a bit is set, the associated mode is enabled. If a bit is clear, the associated mode is disabled. Unused bits are clear. The meaning of the bits is described under the description of the **TIOCSMODES ioctl()**.

TIOCSMODES Set the **pty** modes according to the value of type long referenced by *arg*. Unused bits are ignored but should be set to zero. The bit values for **pty** modes are listed below.

PM_REMOTE

Enable or disable remote mode. See the discussion of the **TIOCREMOTE ioctl()**.

PM_TTY

Enable or disable tty mode. See the discussion of the **TIOCTTY ioctl()**.

PM_PKT

Enable or disable packet mode. See the discussion of the **TIOCPKT ioctl()**.

PM_TRAP

Enable or disable trap mode. See the discussion of the **TIOCTRAP ioctl()**.

PM_MONITOR

Enable or disable monitor mode. See the discussion of the **TIOCMONITOR ioctl()**.

PM_OPEN_ERROR

Enable or disable open error mode. Open error mode allows a server process to return an error to a trapped slave **pty open()** through the **TIOCREQSET ioctl()**. When open error mode is enabled, the server may return a trapped **open()** with an error by setting the **errno_error** field in the **request_info** structure passed to the **TIOCREQSET ioctl()**. When open error mode is disabled (the default state), setting **errno_error** to handshake a slave **open()** has no effect. Note that unlike the **ioctl()** trap handshaking, setting **return_value** has no effect for a slave **open()** regardless of the state of open error mode. See the discussion of the **TIOCREQSET ioctl()** for further details on handshaking a trapped request.

WARNINGS

The slave side cannot indicate an end-of-file condition to the master side.

When using **TIOCREMOTE**, a single **write()** request to the master side of greater than 256 bytes may result in multiple smaller records being read from the slave side instead of only one record.

AUTHOR

pty was developed by the University of California, Berkeley.

FILES

<code>/dev/ptym/pty[a-ce-su-z][0-9][0-9]</code>	master pseudo terminals
<code>/dev/ptym/pty[a-ce-su-z][0-9][0-9][0-9]</code>	master pseudo terminals
<code>/dev/ptym/pty[a-ce-su-z][0-9a-f]</code>	master pseudo terminals
<code>/dev/pty[pqr][0-9a-f]</code>	master pseudo terminals
<code>/dev/pty/tty[a-ce-su-z][0-9][0-9]</code>	slave pseudo terminals
<code>/dev/pty/tty[a-ce-su-z][0-9][0-9][0-9]</code>	slave pseudo terminals
<code>/dev/pty/tty[a-ce-su-z][0-9a-f]</code>	slave pseudo terminals
<code>/dev/tty[pqr][0-9a-f]</code>	slave pseudo terminals

SEE ALSO

`close(2)`, `fcntl(2)`, `ioctl(2)`, `open(2)`, `read(2)`, `select(2)`, `sigblock(2)`, `write(2)`, `ptsname(3C)`, `signal(5)`, `termio(7)`.

NAME

random, urandom, rng - strong random number generator

SYNOPSIS

```
#include <sys/random.h>
```

DESCRIPTION

The character special files `/dev/random` and `/dev/urandom` provide an interface to the kernel-resident random number generator, `rng`. A `read()` from `/dev/random` is potentially blocking. A `read()` from `/dev/urandom` is always nonblocking. Data from `/dev/urandom` can potentially have lower entropy than data from `/dev/random`.

The `rng` module is a dynamically loadable kernel module (DLKM). That is, it can be dynamically unconfigured or reconfigured by an administrator with root authority without rebooting the system.

A sequence from `rng` has unlimited entropy. In contrast, a sequence generated computationally by a pseudorandom number generator, such as `random(3M)`, has limited entropy, derived only from its initial seed. The `rng` module should be considered a quality source for randomness. It has passed extensive statistical testing, including the NIST (National Institute of Standards and Technology) tests for randomness.

The `rng` module uses the uncertainty in completion times of interrupt threads triggered by external events. The `rng` module extracts a sequence of bits from the interrupt time stamps. Any existing bit bias is removed to yield a sequence with uniform distribution of 0's and 1's. The resulting sequence is divided between the holding buffers for the special files `/dev/random` and `/dev/urandom`. For each `read()` on `/dev/random` and `/dev/urandom`, data is retrieved from the corresponding holding buffer. A hash function based on AES (Advanced Encryption Standard) is applied and the result is placed in the buffer provided by the user. All requests on the holding buffers are serialized to ensure that returned random data is not shared between different requests even for simultaneous requests on a multiprocessor system.

There is no `write()` function associated with either `/dev/random` or `/dev/urandom`, and both devices are read-only by all users. A single `ioctl()` is defined for `/dev/random` to facilitate independent verification of `rng` production.

The file `/usr/include/sys/random.h` contains the following definitions:

```
/* The maximum request size, for read() or ioctl(), in bytes */
#define RNG_READMAX    256

/* ioctl() to retrieve data from the entropy collector directly*/
#define RNG_GETRAW     _IOR('Q', 0, uint8_t[RNG_READMAX])
```

If a `read()` request is for more than `RNG_READMAX` bytes, it is treated as if it was for exactly `RNG_READMAX` bytes. This holds for both `/dev/random` and `/dev/urandom`.

Specific Information About /dev/random

When there are a large number of requests on `/dev/random` within a short time interval, the demand on the holding buffer can exceed the rate at which data is supplied by `rng`. A `read()` on the `/dev/random` device blocks the requesting thread if the random data stored in the holding buffer is too low to complete the request. The thread blocks until the holding buffer has been updated with enough random data to complete the request.

For `/dev/random open()` flags, only `O_NONBLOCK` and `O_NDELAY` have device-specific actions. If neither of these flags is set, a `read()` on `/dev/random` will block until the amount of data requested, up to `RNG_READMAX` bytes, can be returned. When the requested number of bytes is not available and either of the above flags are set, `read()` returns immediately. If the `O_NONBLOCK` flag is set, `read()` returns -1 and `errno` is set to `EAGAIN`. If `O_NONBLOCK` is not set and `O_NDELAY` is set, `read()` returns zero.

The `RNG_GETRAW ioctl()` permits an application with superuser privilege to fetch `RNG_READMAX` bytes of data directly from the `/dev/random` holding buffer, after bias has been removed but before the AES hash. This interface is not intended to be used for cryptographic applications, rather, for statistical testing of the randomness of the data in the `/dev/random` holding buffer. This `RNG_GETRAW ioctl()` blocks for the same reason as a read on `/dev/random`. If the requesting thread does not have superuser authority, `EACCESS` is returned.

Specific Information About /dev/urandom

To address the limited random data collection rate problem, the `/dev/urandom` device is strictly non-blocking. The `/dev/urandom` holding buffer is regularly updated with random data, yet a high number of reads can decrease the entropy in its holding buffer. Under this conditions, the entropy of the data from `/dev/urandom` will be slightly lower than the one from `/dev/random`, yet `/dev/urandom` can still be considered a good source of random numbers.

There are no `open()` flags that result in device-specific actions with `/dev/urandom read()`.

ERRORS

- [EAGAIN] For `/dev/random read()`, `O_NONBLOCK` was set when `/dev/random` was opened, and there is insufficient content in the holding buffer to complete the request.
- [EACCES] For the `/dev/random RNG_GETRAW ioctl()`, the requesting thread did not have superuser authority.

AUTHOR

The random number generator was developed by HP.

For bias removal, the generator uses an algorithm by Dr. Yuval Perez, University of California.

The secure hashing uses an AES implementation provided by Dr. Brian Gladman, UK.

The NIST statistical tests are available at <http://csrc.nist.gov/rng>.

FILES

- `/dev/random`
- `/dev/urandom`

SEE ALSO

- random(3M).

NAME

route - kernel packet forwarding database

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>
#include <net/route.h>
#include <net/if.h>

s = socket(AF_ROUTE, SOCK_RAW, family);
```

DESCRIPTION

This manpage describes routing socket interface to read and write kernel routing messages.

The information on how to transmit network packets is maintained by the HP-UX kernel in the routing information database, also known as the routing table. A user process can read or update information in the routing table by sending routing messages to the kernel via an AF_ROUTE socket. The message types are described in more detail in the *Message Types* section below.

The *family* parameter in the **socket** system call shown in the *SYNOPSIS* may be used to filter the routing messages the caller receives. The valid values for *family* are:

AF_INET	get routing messages affecting the Internet Protocol.
AF_INET6	get routing messages affecting the Internet Protocol version 6.
AF_UNSPEC	get routing messages affecting both AF_INET and AF_INET6 protocols.

Entries in the routing table specify the appropriate remote host or gateway to use when transmitting packets. These entries are either host-specific, or are applicable to all hosts located on a generic subnetwork, as specified by a netmask value.

After the system boots, each protocol family adds entries to the routing table for each network interface configured and ready to transmit network traffic. Normally, the route entry is specified as a **direct connection** to the destination host or network. For direct routes, the transport layer of the network stack sends packets directly to the host specified in the packet header. For non-direct routes, the interface forwards the packet to the gateway listed in the routing entry for that interface.

When routing packets, the kernel attempts to find an optimal route for each destination. If more than one entry matches the netmask of the destination, the kernel selects the route with the greater number of 1's in the netmask.

A default (wildcard) route is used if no other route to a particular remote host or network can be located. A default route is specified with an all 0 destination address value and a netmask of all 0's. Default routes, in combination with routing redirects, provide an economical mechanism for routing network traffic.

If no routing entry is found, the destination is declared as unreachable, and a routing-miss message (RTM_MISS) is generated to any user processes using the routing socket facilities, as described below.

Message Types

After creating a routing socket, the process can send commands to the kernel by writing to the socket. The process can read information from the kernel by reading from the socket. The following message types can be used to communicate routing information between the user process and the kernel:

RTM_ADD	add route
RTM_CHANGE	change gateway, metrics or flags
RTM_DELADDR	address being removed from interface
RTM_DELETE	delete route
RTM_GET	report metrics and other information
RTM_IFINFO	interface going up, down, etc.
RTM_LOSING	kernel suspects route is failing
RTM_LOCK	lock specified metrics
RTM_MISS	lookup on this address failed
RTM_NEWADDR	address being added to interface
RTM_REDIRECT	kernel instructs to use different route
RTM_RESOLVE	request to resolve destination to link-layer address

All 12 message types can be used to read information from the kernel. To write to the kernel, the process can issue RTM_ADD, RTM_DELETE, or RTM_GET message types to update information in the routing table.

Message types RTM_CHANGE and RTM_LOCK are not supported on HP-UX. If a user process issues these messages, [EOPNOTSUPP] error will be returned.

Message Structure

Messages are formed by a message header followed by a small number of socket address structures.

What message header to use depends on the message type. The RTM_IFINFO messages use the **if_msghdr** header. The RTM_NEWADDR and RTM_DELADDR messages use the **ifa_msghdr** header. All other message types use the **rt_msghdr** header.

The **rt_msghdr** structure contains the following members:

```
uint16_t rtm_msglen;      /* to skip over unrecognized messages */
uint8_t  rtm_version;    /* future binary compatibility */
uint8_t  rtm_type;       /* message type */
uint16_t rtm_index;      /* index for associated ifp */
int32_t  rtm_flags;      /* flags, incl. kern & message,
                          * e.g. DONE */
int32_t  rtm_addrs;      /* bitmask identifying sockaddrs in
                          * the message */
pid_t    rtm_pid;        /* identify sender */
int32_t  rtm_seq;        /* for sender to identify action */
int32_t  rtm_errno;      /* error indicator */
int32_t  rtm_use;        /* from rtentry */
int32_t  rtm_inits;      /* which metrics we are initializing */
struct rt_metrics rtm_rmx; /* metrics themselves */
```

The **if_msghdr** structure contains the following members:

```
uint16_t ifm_msglen;      /* to skip over unrecognized messages */
uint8_t  ifm_version;    /* future binary compatibility */
uint8_t  ifm_type;       /* message type */
int32_t  ifm_addrs;      /* bitmask identifying sockaddrs in
                          * the message */
int32_t  ifm_flags;      /* value of if_flags */
uint16_t ifm_index;      /* index for associated ifp */
struct if_data ifm_data;  /* statistics and other data about
                          * interface */
```

The **ifa_msghdr** structure contains the following members:

```
uint16_t ifam_msglen;    /* to skip over unrecognized messages */
uint8_t  ifam_version;   /* future binary compatibility */
uint8_t  ifam_type;      /* message type */
int32_t  ifam_addrs;     /* bitmask identifying sockaddrs in
                          * the message */
int32_t  ifam_flags;     /* value of ifa_flags */
uint16_t ifam_index;     /* index for associated ifp */
int32_t  ifam_metric;    /* value of ifa_metric */
```

To determine retransmission behavior, reliable protocols use the **rt_metrics** structure included in the **rt_msghdr** message header. The **rt_metrics** structure contains the following members:

```
uint32_t rmx_locks;      /* Kernel must leave these values alone */
uint32_t rmx_mtu;        /* MTU for this path */
uint32_t rmx_hopcount;   /* max hops expected */
uint32_t rmx_expire;     /* lifetime for route, e.g. redirect */
uint32_t rmx_recvpipe;   /* inbound delay-bandwidth product */
uint32_t rmx_sendpipe;   /* outbound delay-bandwidth product */
uint32_t rmx_ssthresh;   /* outbound gateway buffer limit */
uint32_t rmx_rtt;        /* estimated round trip time */
uint32_t rmx_rttvar;     /* estimated rtt variance */
uint32_t rmx_pktsent;    /* packets sent using this route */
```

The **if_data** structure included in the **if_msghdr** message header defines a queue for a network interface and contains the following members:


```

/* generic interface information */
uint8_t ifi_type;          /* ethernet, tokenring, etc */
uint8_t ifi_physical;     /* AUI, Thinnet, 10base-T, etc */
uint8_t ifi_addrlen;     /* media address length */
uint8_t ifi_hdrlen;      /* media header length */
uint8_t ifi_recvquota;    /* polling quota for receive intrs */
uint8_t ifi_xmitquota;    /* polling quota for xmit intrs */
uint32_t ifi_mtu;        /* maximum transmission unit */
uint32_t ifi_metric;     /* routing metric (external only) */
uint32_t ifi_baudrate;   /* linespeed */

/* volatile statistics */
uint32_t ifi_ipackets;    /* packets received on interface */
uint32_t ifi_ierrors;    /* input errors on interface */
uint32_t ifi_opackets;   /* packets sent on interface */
uint32_t ifi_oerrors;    /* output errors on interface */
uint32_t ifi_collisions; /* collisions on csma interfaces */
uint32_t ifi_ibytes;     /* total number of octets received */
uint32_t ifi_obytes;     /* total number of octets sent */
uint32_t ifi_imcasts;    /* packets received via multicast */
uint32_t ifi_omcasts;    /* packets sent via multicast */
uint32_t ifi_iqdrops;    /* dropped on input, this interface */
uint32_t ifi_noproto;    /* destined for unsupported protocol */
uint32_t ifi_hwassist;   /* HW offload capabilities */
uint32_t ifi_unused;     /* XXX was ifi_xmittiming */
struct timeval ifi_lastchange; /* time of last administrative change */

```

(Note that the position of items in all previously mentioned data structures does not necessarily reflect the order of the members in the structure.)

The members `rtm_addrs`, `ifm_addrs`, and `ifam_addrs` of the message headers are bitmasks that specify what socket address structure(s) follow the message. When multiple `sockaddrs` follow the message, they are interpreted based on their order in the message and the value stored in the bitmask. The sequence is least significant to the most significant bit within the vector.

The following constants are defined to indicate which socket addresses are present in the routing message:

```

#define RTA_DST      0x01 /* destination sockaddr present */
#define RTA_GATEWAY  0x02 /* gateway sockaddr present */
#define RTA_NETMASK  0x04 /* netmask sockaddr present */
#define RTA_GENMASK  0x08 /* cloning mask sockaddr present */
#define RTA_IFP      0x10 /* interface name sockaddr present */
#define RTA_IFA      0x20 /* interface address sockaddr present */
#define RTA_AUTHOR   0x40 /* author of redirect sockaddr present */
#define RTA_BRD      0x80 /* for NEWADDR, broadcast or
 * point-to-point destination
 * address */

```

Any messages sent to the kernel are returned back to the process issuing the command, and message copies are sent to all interested listeners. The sender may provide its process ID to be stored in the message header. An additional sequence field can be used to distinguish between outstanding messages. However, message replies may be lost when kernel buffers are exhausted.

Any messages generated by the kernel would have process ID and sequence field set to zero.

The kernel may spontaneously emit routing messages in response to external events, such as receipt of a redirect command, or failure to locate an appropriate route for a request. A process may ignore all messages from the routing socket by doing a `shutdown(2)` system call for further input.

Security Restrictions

Only users with appropriate privileges can make changes to the routing table.

Notes

Some fields in the message header structures are not used on HP-UX. This means when the kernel generates routing messages it sets these fields to 0. Also, when the kernel receives routing messages, it ignores any values contained in these fields. This applies to the following fields:

Structure Fields Not Used

rt_msghdr rtm_use, rtm_inits, rtm_rmx, except for rtm_rmx.rmx_mtu and rtm_rmx.rmx_rtt

if_msghdr ifm_data, except for ifm_data.ifm_mtu, ifm_data.ifm_metric, ifm_data.ifm_ipackets, and ifm_data.ifm_opackets

ifa_msghdr ifam_metric

ERRORS

If the kernel rejects a routing message, the **rtm_errno** field in the **rt_msghdr** structure may be set to one of the following values:

[EEXIST] The specified entry already exist. Requested to duplicate an existing entry.

[ENETUNREACH] Network is unreachable.

[ENOENT] The specified entry does not exist. Requested to delete non-existent entry.

[ENOBUFS] No buffer space is available. Insufficient resources were available to install a new route.

[EOPNOTSUPP] Operation not supported. Message types RTM_CHANGE and RTM_LOCK are not supported on HP-UX.

[EPERM] Permission to issue a command is denied. The user needs appropriate privileges to make changes to the routing table.

EXAMPLES

The following sample program illustrates how a user process can add a route to the kernel's routing table.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <net/route.h>
#include <net/if.h>
#include <netinet/in.h>

int main(int argc, char **argv)
{
    int    s;
    char  buf[1024];
    struct rt_msghdr *rtm;
    struct sockaddr_in *sin1, *sin2;

    if (argc != 3) {
        printf("usage: %s <destinationIP> <gatewayIP>\n",
            argv[0]);
        return -1;
    }

    if ((s = socket(AF_ROUTE, SOCK_RAW, AF_UNSPEC)) < 0) {
        perror("failed to create socket");
        return -1;
    }

    rtm = (struct rt_msghdr *)buf;

    rtm->rtm_msglen = sizeof(struct rt_msghdr) +
        (2 * sizeof(struct sockaddr_in));
    rtm->rtm_version = RTM_VERSION;
    rtm->rtm_type = RTM_ADD;
    rtm->rtm_addrs = (RTA_DST | RTA_GATEWAY);
    rtm->rtm_rmx.rmx_hopcount = 1;
    rtm->rtm_pid = getpid();
    rtm->rtm_errno = 0;
    rtm->rtm_seq = 0001;
}
```

```

/*
 * the destination address being added follows
 * the routing header
 */
sin1 = (struct sockaddr_in *) (rtm + 1);
sin1->sin_family = AF_INET;
sin1->sin_addr.s_addr = inet_addr(argv[1]);

/*
 * the gateway address being added follows the
 * destination address
 */
sin2 = (struct sockaddr_in *) (sin1 + 1);
sin2->sin_family = AF_INET;
sin2->sin_addr.s_addr = inet_addr(argv[2]);

if (write(s, (caddr_t) rtm, rtm->rtm_msglen) < 0) {
    perror("Failed to send routing message");
    return -1;
}

return 0;
}

```

AUTHOR

Routing socket interface was developed by HP and the University of California, Berkeley.

SEE ALSO

route(1M), ioctl(2), shutdown(2), socket(2), routing(7).

NAME

routing - system support for local network packet routing

DESCRIPTION

The network facilities for HP-UX provide general packet routing support. Routing table maintenance is handled by application processes.

A routing table consists of a set of data structures used by the network facilities to select the appropriate remote host or gateway when transmitting packets. The table contains a single entry for each route to a specific network or host, as displayed by the **netstat** command with the **-r** or **-rn** options (see *netstat(1)*). Routes that are not valid are not displayed.

```
# netstat -r
Routing tables
Destination      Gateway          Flags  Refs  Use Interface  Pmtu
hpindwr.cup.hp.com
  localhost      localhost        UH      1    39 lo0          4608
localhost       localhost        UH      0    68 lo0          4608
147.253.56.195   localhost        UH      0     0 lo0          4608
147.253.144.66  localhost        UH      0     0 lo0          4608
default         hpinsmh.cup.hp.com
  15.13.136      hpindwr.cup.hp.com
                    UG      1    21 lan0          1500
                    U      1    92 lan0          1500
147.253.56      147.253.56.195  U      0     7 lan2          1500
147.253.144.64  147.253.144.66  U      0     7 lan1          1500
```

```
# netstat -rn
Routing tables
Destination      Gateway          Flags  Refs  Use Interface  Pmtu
15.13.136.66    127.0.0.1        UH      1    39 lo0          4608
127.0.0.1      127.0.0.1        UH      0    68 lo0          4608
147.253.56.195  127.0.0.1        UH      0     0 lo0          4608
147.253.144.66  127.0.0.1        UH      0     0 lo0          4608
default         15.13.136.11     UG      2    30 lan0          1500
15.13.136.0    15.13.136.66     U      1   113 lan0          1500
147.253.56.0   147.253.56.195  U      0     7 lan2          1500
147.253.144.64  147.253.144.66  U      0     7 lan1          1500
```

```
# netstat -rv
Routing tables
Dest/Netmask     Gateway          Flags  Refs  Use Interface  Pmtu
hpindwr.cup.hp.com/0xffffffff
  localhost      localhost        UH      1    39 lo0          4608
localhost/0xffffffff
  localhost      localhost        UH      0    68 lo0          4608
147.253.56.195/0xffffffff
  localhost      localhost        UH      0     0 lo0          4608
147.253.144.66/0xffffffff
  localhost      localhost        UH      0     0 lo0          4608
default/0x00000000
  hpinsmh.cup.hp.com
                    UG      2    31 lan0          1500
15.13.136/0xfffff800
  hpindwr.cup.hp.com
                    U      1   129 lan0          1500
147.253.56/0xfffffe00
  147.253.56.195  U      0     7 lan2          1500
147.253.144.64/0xfffffff0
  147.253.144.66  U      0     7 lan1          1500
```

r

```
# netstat -rnv
Routing tables
Dest/Netmask      Gateway          Flags  Refs  Use Interface  Pmtu
15.13.136.66/255.255.255.255
                  127.0.0.1       UH          1    39 lo0          4608
127.0.0.1/255.255.255.255
                  127.0.0.1       UH          0    68 lo0          4608
147.253.56.195/255.255.255.255
                  127.0.0.1       UH          0     0 lo0          4608
147.253.144.66/255.255.255.255
                  127.0.0.1       UH          0     0 lo0          4608
default/0.0.0.0  15.13.136.11    UG          3    40 lan0         1500
15.13.136.0/255.255.248.0
                  15.13.136.66    U           1   153 lan0         1500
147.253.56.0/255.255.254.0
                  147.253.56.195  U           0     8 lan2         1500
147.253.144.64/255.255.255.240
                  147.253.144.66  U           0     8 lan1         1500
```

The following columns are of particular interest:

Destination	The destination Internet address: host name, network name, or default . The default keyword indicates a wildcard route, used as a last resort if no route is specified for a particular remote host or network. See Flags .
Netmask	The netmask and the destination Internet address together define a range of IP addresses that may be reached by the route's gateway. A host route by default has a netmask of all 1's. A default route by default has a netmask of all 0's. The netmask is also used in selecting a route to forward an IP packet. See the <i>Routing Algorithm</i> subsection.
Gateway	The gateway to use to get to the destination: a remote gateway or the local host. See Flags .
Flags	The type of route: <ul style="list-style-type: none"> U The route is "up" or available (see <i>ifconfig(1M)</i>). G The route uses a remote host as a gateway; otherwise, the local host is shown as the gateway (see <i>route(1M)</i>). H The destination is a host; otherwise, the destination is a network (see <i>route(1M)</i>).
Interface	The interface connections: <ul style="list-style-type: none"> lo0 The local loopback after system boot. lan0, lan1,... The interface cards installed on the local host after the ifconfig command is executed at boot time (see <i>ifconfig(1M)</i>).

The values of the *count* and *destination* type fields in the **route** command determine the presence of the **G** and **H** flags in the **netstat -r** display and thus the route type, as shown in the following table.

Count	Destination Type	Flags	Route Type
=0	network	U	Route to a network directly from the local host
>0	network	UG	Route to a network through a remote host gateway
=0	host	UH	Route to a remote host directly from the local host
>0	host	UGH	Route to a remote host through a remote host gateway
=0	default	U	Wildcard route directly from the local host
>0	default	UG	Wildcard route through a remote host gateway

Subnets

The network facilities support variable-length subnetting. An Internet address is made up of a **network address** portion, and a **host address** portion of an address in the form:

192.34.17.0

Subnet addresses are defined as a portion of the network's Internet address. This scheme provides for:

- Network addresses that identify physically distinct networks.
- Subnet addresses that identify physically distinct subnetworks of the same network.

A network manager can subdivide the Internet address of the local network into subnets using the host number space. This facility allows several physical networks to share a single Internet address.

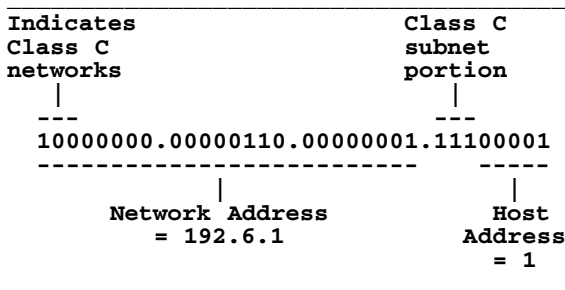
To allow for this, three Internet classes are defined, each accommodating a different amount of network and host addresses. The address classes are defined by the most significant bit of the binary form of the address.

The following table lists the number of networks, nodes, and the address ranges for each address class:

Class	Networks	Nodes per Network	Address Range
A	127	16777215	0.0.0.1 - 127.225.225.254
B	16383	65535	128.0.0.1 - 191.255.255.254
C	2097151	255	192.0.0.1 - 223.244.244.243
Reserved	—	—	224.0.0.0 - 255.255.255.255

The first 8 bits of a Class A network has network space for only 127, while accommodating the largest number of nodes possible among the classes defined. A single class B network has the network address limitation of 16 bits, and 16 bits to define the nodes.

For example, a Class C address space is as follows:



A subnet for a given host is specified with the **ifconfig** command (see *ifconfig(1M)*), using the **netmask** parameter with a 32-bit subnet *mask*.

The default masks for the three classes of Internet addresses are as follows:

- Class A: 255.0.0.0
- Class B: 255.255.0.0
- Class C: 255.255.255.0

An example Class C network number is 192.34.17.0. The last field specifies the host number. Thus, all hosts with the prefix 192.34.17 are recognized as being on the same logical and physical network.

If subnets are not in use, the default mask used is 255.255.255.0.

If subnets are used and the 8-bit host field is partitioned into 3 bits of subnet and 5 bits of host as in the above example, then the subnet mask would be 255.255.255.192.

If a host has multiple interfaces, then it can belong to different subnets. Unlike past releases, the subnets can have different sizes even if they may have the same network address. This is accomplished by using a different netmask on each of the host interfaces. For example, the **lan1** and **lan2** interface shown in the **netstat** tables above are connected to two distinct subnets of the same network, 147.253. The subnet that **lan1** belongs to can have at most 14 hosts, because its netmask is 255.255.255.240.

Note:

The host portion of those IP addresses in the subnet cannot be all 1's or all 0's, therefore this subnet can support only 14 hosts, not 16.

The subnet that **lan2** belongs to can have up to 510 hosts, because its netmask is 255.255.254.0.

Supernets

A supernet is a collection of smaller networks. Supernetting is a technique of using the netmask to aggregate a collection of smaller networks into a supernet. This technique is particularly useful for class C networks. A Class C network can only have 254 hosts. This can be too restrictive for some companies. For these companies, a netmask that only contains a portion of the network part can be applied to the hosts in these class C networks to form a supernet. This supernet netmask should be applied to those interfaces that connect to the supernet using the *ifconfig* command (see *ifconfig*(1M)). For example, a host can configure its interface to connect to a class C supernet, for example, 192.6, by configuring an IP address of 192.6.1.1 and a netmask of 255.255.0.0 to its interface.

Routing Algorithm

The routing table entries are of three types:

- Entries for a specific host.
- Entries for all hosts on a specific network.
- Wildcard entries for any destination not matched by entries of the first two types.

To select a route for forwarding an IP packet, the network facilities select the complete set of "matching" routing table entries from the routing table. A routing table entry is considered a match, if the result of the bit-wise AND operation between the netmask in the routing entry and the IP packet's destination address equals the destination address in the routing entry.

The network facilities then select from the set the routing entries that have the longest netmask. The length of a netmask is defined as the number of contiguous 1 bits starting from the leftmost bit position in the 32-bit netmask field. In other words, the network facilities select the routing entry that specifies the narrowest range of IP addresses. For example, the host route entry that has a destination/netmask pair of (147.253.56.1, 0xFFFFFFFF), is more specific than the network route entry that has a destination/netmask pair of (147.253.56.0, 0xFFFFFE00); therefore, the network facilities select the host route entry. The default route by default has a destination/netmask pair of (0,0). Therefore, the default route matches all destinations but it is also the least specific. The default route will be selected only if there is not a more specific route.

There may still be multiple routing entries remaining. In that case, the IP packet is routed over the first entry displayed by **netstat -r**. Such multiple routes include:

- Two or more routes to a host via different gateways.
- Two or more routes to a network via different gateways.

A superuser can change entries in the table by using the **route** command (see *route*(1M)), or by information received in Internet Control Message Protocol (ICMP) redirect messages.

If there are more than one default gateways for a particular net or subnet, each will be used in turn to effect the even distribution of datagrams to the different gateways.

WARNINGS

Reciprocal **route** commands must be executed on the local host and the destination host, as well as all intermediate hosts, if routing is to succeed in the cases of virtual circuit connections or bidirectional datagram transfers.

AUTHOR

routing was developed by the University of California, Berkeley.

FILES

/etc/hosts
/etc/networks

SEE ALSO

netstat(1), ifconfig(1M), route(1M), route(7P).

NAME

sad - STREAMS Administrative Driver

SYNOPSIS

```
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/sad.h>
#include <stropts.h>

int ioctl(
    int fildes,
    int command,
    ...
    /* arg */
);
```

DESCRIPTION

The **sad** driver provides an interface to the **autopush** facility using the **ioctl()** function. As an interface, the **sad** driver enables administrative tasks to be performed on STREAMS modules and drivers. By specifying the *command* parameter to the **ioctl()** function, an administrator can configure **autopush** information for a device, get information on a device, or check a list of modules.

fildes is a file descriptor obtained by opening **/dev/sad** using **open()**. *command* specifies the administrative function to be performed. *arg* points to a data structure. If *command* is **SAD_SAP** or **SAD_GAP**, *arg* points to a struct of type **strapush**. If *command* is **SAD_VML**, *arg* points to a struct of type **str_list**.

Security Restrictions

The **SAD_SAP ioctl()** is restricted to superusers or users with the **NETADMIN** privilege. See *privileges(5)* for more information about privileged access on systems that support fine-grained privileges.

ioctl Commands

The commands used to perform administrative functions on a STREAMS module or driver are specified by the following **ioctl()** commands:

SAD_SAP

Allows you to configure **autopush** information for a device. The *arg* parameter points to a **strapush** structure (defined in the **<sys/sad.h>** header file), whose members are as follows:

```
struct strapush {
    uint sap_cmd;
    long sap_major;
    long sap_minor;
    long sap_lastminor;
    long sap_npush;
    char sap_list[MAXAPUSH][FMNAMESZ+1];
};
```

sap_cmd

Allows you to specify the type of configuration to perform. This field can have the following values:

SAP_ALL

Configures all minor devices.

SAP_RANGE

Configures a range of minor devices.

SAP_ONE

Configures a single minor device.

SAP_CLEAR

Clears the previous settings. Specify only the **sap_major** and **sap_minor** fields when using this command. If a previous entry specified

SAP_ALL, set the **sap_minor** field to 0 (zero). If a previous entry was specified as **SAP_RANGE**, set the **sap_minor** field to the lowest minor device number in the range.

- sap_major**
Specifies the major device number.
- sap_minor**
Specifies the minor device number.
- sap_lastminor**
Specifies the range of minor devices.
- sap_npush**
Specifies the number of modules to push. This number must be no more than **MAXA-PUSH**, which is defined in **<sad.h>**. Additionally, this number must not exceed **NSTRPUSH**.
- sap_list**
Specifies, in order, the array of modules to push.

SAD_GAP

Lets you use the **sad** driver to obtain **autopush** configuration information for a device by setting the **sap_major** and **sap_minor** fields of the **strapush** structure (see the **SAD_SAP** command) to the major and minor device numbers of the device being queried.

arg should point to a struct of type **strapush**. Upon successful completion, the **strapush** structure contains all of the information used to configure the device. Values of 0 (zero) will appear in any unused entry in the module list.

SAD_VML

Enables you to check a list of modules. For example, you can determine if a specific module has been installed. The *arg* parameter points to a **str_list** structure (defined in the **<stropts.h>** header file), whose members are as follows:

```
struct str_list {
    int sl_nmods;
    struct str_mlist *sl_modlist;
};
```

sl_nmods
Specifies the number of entries you have allocated in an array.

sl_modlist
Points to the array of module names. The **str_mlist** structure (also in the **<stropts.h>** header file) is as follows:

```
struct str_mlist {
    char l_name[FMNAMESZ+1];
};
```

where **l_name** specifies the array of module names.

If the **l_name** array is valid, the **SAD_VML** command returns a value of 0 (zero). If the array contains an invalid module name, the command returns a value of 1. Upon failure, the command returns a value of -1.

Notes

As a STREAMS driver, **sad** also supports the normal STREAMS **I_STR ioctl()**:

```
int ioctl(fildes, I_STR, strp);
int fildes;
struct strioctl *strp;
```

In this form, specify the **ic_cmd** field in the **strioctl** structure to either **SAD_SAP**, **SAD_GAP**, or **SAD_VML**. The **ic_dp** field points to the **strapush** structure (see the **SAD_SAP** command in the **DESCRIPTION** section). Refer to the *streamio(7)* reference page for further details.

RETURN VALUE

Unless specified otherwise, upon successful completion, the **sad ioctl()** commands return a value of 0 (zero). Otherwise, a value of -1 is returned.

ERRORS

If any of the following conditions occur, the **sad ioctl** commands return the corresponding value:

SAD_SAP

[EEXIST]	The specified major/minor device number pair (sad_major/sad_minor) has already been configured.
[EFAULT]	The <i>arg</i> parameter points outside the allocated address space.
[EINVAL]	The major device number (sad_major) is invalid, the number of modules (sap_list [MAXAPUSH] [FMNAMESZ+1]) is invalid, or the list of module names is invalid.
[ENODEV]	The device is not configured for autopush . This value is returned from a SAD_GAP command.
[ENOSR]	A internal autopush data structure cannot be allocated.
[ENOSTR]	The major device does not represent a STREAMS driver.
[ERANGE]	The sap_lastminor field is less than the sap_minor field when the command is SAP_RANGE , or the minor device specified in a SAP_CLEAR command does not exist.
[EACCES]	Only a superuser or user with NETADMIN privilege is allowed to execute the SAD_SAP ioctl() .

SAD_GAP

[EFAULT]	The <i>arg</i> parameter points outside the allocated address space.
[EINVAL]	The major device number (sad_major) is invalid.
[ENODEV]	The device is not configured for autopush .
[ENOSTR]	The major device does not represent a STREAMS driver.

SAD_VML

[EFAULT]	The <i>arg</i> parameter points outside the allocated address space.
[EINVAL]	The list of module names is invalid.

SEE ALSO

autopush(1M), ioctl(2), open(2), privileges(5), streamio(7).

NAME

scsi - Small Computer System Interface device drivers

DESCRIPTION

The Small Computer System Interface (SCSI) is an American National Standard for interconnecting computers and peripheral devices. HP-UX supports the SCSI device protocol on parallel SCSI interfaces (see ANSI Std X3.131-199X, "SCSI-2"), Fibre Channel interfaces (see ANSI Std X3.269-199X, "Fibre Channel Protocol for SCSI"), and Serial Attached SCSI interfaces (SAS).

The SCSI standard includes specifications for a variety of device types. This section describes the general SCSI interface for all SCSI device drivers. Information about specific device types can be found in the manual sections which describe SCSI peripheral device drivers for those device types.

The ioctls described here can be issued either on persistent device files or legacy devices (see *intro(7)*). Legacy device files are deprecated with HP-UX release 11i V3. They are maintained for backward compatibility, and may be obsolete in future releases.

The behavior of some ioctls may differ depending on whether issued on persistent device files or legacy device files, and whether multi-pathing is enabled on legacy device files. Typically ioctls issuing SCSI commands to a device may use any available LUN path to send the commands. However, when multi-pathing is disabled on legacy device files (see **leg_mpath_enable** attribute in *scsimgr(1M)*), the ioctl only attempts to use the LUN path corresponding to the legacy device file. If this LUN path is not available, the ioctl will fail even if there are other LUN paths available. This behavior corresponds to the legacy behavior.

The **SIOC_INQUIRY** ioctl is supported by all SCSI device drivers. This ioctl returns the SCSI device-specific INQUIRY command data. This data contains device identification and capability information. Since there have been multiple versions of the SCSI standard for inquiry data, multiple versions of the inquiry data declaration are provided. The SCSI-1 version is provided for backward compatibility only. If issued on a legacy device file, this ioctl only tries to use the LUN path corresponding to the legacy device file even if multi-pathing is enabled on legacy device files.

The **SIOC_CAPACITY** ioctl indicates the current device size. A device size is defined to be a logical block size and some number of logical blocks. The means of determining this device-size data is particular to the specific device type. Logical block size and/or number of logical blocks equal to zero indicates: the device size is unknown, the device is not currently capable of I/O operations, or I/O operations are not meaningful for the device. Note that for very large devices, the ioctl argument can overflow, **SIOC_STORAGE_CAPACITY** is a better choice, than **SIOC_CAPACITY** where devices can be large. Also note that **DIIOC_CAPACITY** is preferred (see *disk(7)*).

The header file `<sys/scsi.h>` has useful information for SCSI devices. The following is included from `<sys/scsi.h>`:

```
#define SIOC_INQUIRY          _IOR('S', 2, union inquiry_data)
#define SIOC_CAPACITY        _IOR('S', 3, struct capacity)
#define SIOC_STORAGE_CAPACITY _IOR('S', 101, storage_capacity_t)

/* SCSI-1 inquiry structure */
struct inquiry {
    unsigned char    dev_type;
    unsigned int     rmb:1;
    unsigned int     dtq:7;
    unsigned int     iso:2;
    unsigned int     ecma:3;
    unsigned int     ansi:3;
    unsigned int     resv:4;
    unsigned int     rdf:4;
    unsigned char    added_len;
    unsigned char    dev_class[3];
    char             vendor_id[8];
    char             product_id[16];
    char             rev_num[4];
    unsigned char    vendor_spec[20];
    unsigned char    resv4[40];
    unsigned char    vendor_parm_bytes[32];
};
```

```

/* SCSI-2 inquiry structure */
struct inquiry_2 {
    unsigned int    periph_qualifier:3;
    unsigned int    dev_type:5;
    unsigned int    rmb:1;
    unsigned int    dtq:7;
    unsigned int    iso:2;
    unsigned int    ecma:3;
    unsigned int    ansi:3;
    unsigned int    aenc:1;
    unsigned int    trmiop:1;
    unsigned int    resv1:2;
    unsigned int    rdf:4;
    unsigned char   added_len;
    unsigned char   resv2[2];
    unsigned int    reladr:1;
    unsigned int    wbus32:1;
    unsigned int    wbus16:1;
    unsigned int    sync:1;
    unsigned int    linked:1;
    unsigned int    resv3:1;
    unsigned int    cmdque:1;
    unsigned int    sftre:1;
    char            vendor_id[8];
    char            product_id[16];
    char            rev_num[4];
    unsigned char   vendor_spec[20];
    unsigned char   resv4[40];
    unsigned char   vendor_parm_bytes[32];
} inquiry_2_t;

/* Definition for version description in SCSI-3 inquiry */
typedef uint8_t    vdesc_t[2];

/* SCSI-3 inquiry structure */
typedef struct inquiry_3 {
    uint32_t        pq                :3;
    uint32_t        pdt                :5;
    uint32_t        rmb                :1;
    uint32_t        rsvd1              :7;
    uint32_t        version            :8;
    uint32_t        aerc               :1;
    uint32_t        obslt1             :1;
    uint32_t        naca               :1;
    uint32_t        hisup              :1;
    uint32_t        rdf                :4;
    uint32_t        added_len          :8;
    uint32_t        sccs               :1;
    uint32_t        rsvd2              :7;
    uint32_t        bque               :1;
    uint32_t        encserv            :1;
    uint32_t        vs1                :1;
    uint32_t        multip             :1;
    uint32_t        mchngr             :1;
    uint32_t        obslt2             :1;
    uint32_t        obslt3             :1;
    uint32_t        addr16             :1;
    uint32_t        reladr             :1;
    uint32_t        obslt4             :1;
    uint32_t        wbus16            :1;
    uint32_t        sync               :1;
    uint32_t        linked             :1;
}

```

```

        uint32_t      obslt5          :1;
        uint32_t      cmdque          :1;
        uint32_t      vs2            :1;
        uint8_t       vendor_id[8];
        uint8_t       product_id[16];
        uint8_t       rev_num[4];
        uint8_t       vendor_spec[20];
        uint16_t      rsvd3           :4;
        uint16_t      clcking        :2;
        uint16_t      gas             :1;
        uint16_t      ius             :1;
        uint16_t      rsvd4           :8;
        vdesc_t       vers_desc[8];
        uint8_t       rsvd6[22];
        uint8_t       vendor_parm_bytes[32];
    } inquiry_3_t;

    /* union for SIOC_INQUIRY ioctl */
    union inquiry_data {
        struct inquiry   inq1;      /* SCSI-1 inquiry */
        struct inquiry_2 inq2;      /* SCSI-2 inquiry */
        inquiry_3_t      inq3;      /* SCSI-3 inquiry */
    };

    /* structure for SIOC_CAPACITY ioctl */
    struct capacity {
        uint32_t lba;
        uint32_t blkksz;
    };

    /* structure for SIOC_STORAGE_CAPACITY ioctl */
    typedef struct {
        uint64_t lba;
        uint32_t blkksz;
    } storage_capacity_t;

```

The **SIOC_XSENSE** ioctl returns detailed information about device status and errors when such information is available. Since there have been multiple versions of the SCSI standard for sense (status) data, multiple versions of the sense data declaration are provided. The SCSI-1 and non-aligned versions are provided for backward compatibility only. If no new CHECK-CONDITION-caused REQUEST SENSE command data has been obtained since the last **SIOC_XSENSE** ioctl call, the **xsense_aligned.error_class** and **sense_2_aligned.error_code** fields will contain the value zero. Applications which require more accurate REQUEST SENSE data handling should use the SCSI device-control driver (see *scsi_ctl(7)*).

The following information is included from `<sys/scsi.h>`:

```

#define SIOC_XSENSE          _IOR('S', 7, union sense_data)

/* structure for SIOC_XSENSE ioctl */
typedef union sense_data {
    xsense_aligned_t      r_sense1a;    /* SCSI and CCS devices */
    sense_2_aligned_t     r_sense2a;    /* SCSI-2 devices */
    xsense_t              r_sense1;     /* Do not use; for
    * compatibility only
    */
    sense_2_t             r_sense2;     /* Do not use; for
    * compatibility only
    */
} sense_data_t;

/*
 * Struct xsense_aligned is for examining the sense data of SCSI-1
 * and CCS devices.

```

```

*/
typedef struct xsense_aligned {
    unsigned int    valid           :1;
    unsigned int    error_class     :3;
    unsigned int    error_code      :4;
    unsigned char   seg_num;
    unsigned int    parms:4;
    unsigned int    sense_key       :4;
    unsigned char   lba[4];
    unsigned char   add_len;
    unsigned char   copysearch[4]; /* Unused by HP-UX */
    unsigned char   sense_code;
    unsigned char   resv;
    unsigned char   fru;
    unsigned char   field;
    unsigned char   field_ptr[2];
    unsigned char   dev_error[4];
    unsigned char   misc_bytes[106];
} xsense_aligned_t;

/*
 * Struct sense_2_aligned is for examining the sense data
 * of SCSI-2 devices
 */
typedef struct sense_2_aligned {
    unsigned int    info_valid      :1;
    unsigned int    error_code      :7;
    unsigned char   seg_num;
    unsigned int    filemark       :1;
    unsigned int    eom            :1;
    unsigned int    ili            :1;
    unsigned int    resv           :1;
    unsigned int    key            :4;
    unsigned char   info[4];
    unsigned char   add_len;
    unsigned char   cmd_info[4];
    unsigned char   code;
    unsigned char   qualifier;
    unsigned char   fru;
    unsigned char   key_specific[3];
    unsigned char   add_sense_bytes[113];
} sense_2_aligned_t;

/*
 * Struct xsense is provided for backward source code
 * compatibility only.
 * Struct xsense_aligned is the appropriate struct for
 * examining the sense
 * data of SCSI-1 and CCS devices.
 */
typedef struct xsense {
    unsigned int    valid           :1;
    unsigned int    error_class     :3;
    unsigned int    error_code      :4;
    unsigned char   seg_num;
    unsigned int    parms           :4;
    unsigned int    sense_key       :4;
    unsigned char   lba[4];
    unsigned char   add_len;
    unsigned char   copysearch[4]; /* Unused by HP-UX */
    unsigned char   sense_code;
    unsigned char   resv;

```

S

```

        unsigned char    fru;
        unsigned char    field;
        unsigned short   field_ptr;
        uint32_t         dev_error;
        unsigned char    misc_bytes[106];
    } xsense_t;

/*
 * Struct sense_2 is provided for backward source code
 * compatibility only.
 * Struct sense_2_aligned is the appropriate struct for
 * examining the sense
 * data of SCSI-2 devices.
 */
typedef struct sense_2 {
    unsigned int    info_valid        :1;
    unsigned int    error_code        :7;
    unsigned char   seg_num;
    unsigned int    filemark          :1;
    unsigned int    ecm                :1;
    unsigned int    ili                :1;
    unsigned int    resv              :1;
    unsigned int    key                :4;
    unsigned char   info[4];
    unsigned char   add_len;
    unsigned int    cmd_info;
    unsigned char   code;
    unsigned char   qualifier;
    unsigned char   fru;
    unsigned char   key_specific[3];
    unsigned char   add_sense_bytes[113];
} sense_2_t;

```

ERRORS

The following errors may result from a call to a SCSI device driver:

- [EACCES] Required permission is denied for the device or operation.
- [ENXIO] If resulting from an open call, this indicates there is no device at the specified address. For other calls, this indicates the specified address is out of range or the device may no longer be accessed.
- [EINVAL] If resulting from an open call, this indicates the device is not supported by the device driver (e.g., incorrect device type). For other calls, this indicates the request or some request argument is invalid. If resulting from the **STIOC_CAPACITY** ioctl, one or more of the fields in the argument structure may have overflowed.
- [EBUSY] This indicates the device is not ready for use or that the requested operation conflicts with other operations (e.g., the device is currently open via another device driver or exclusive access is in effect).
- [EIO] Indicates a SCSI protocol or communication problem has occurred, or that a SCSI command resulted in a non-good status.

Manual entries that describe specific SCSI peripheral device drivers may provide additional qualification of error results.

WARNINGS

Use of devices that are not officially supported can cause data loss, system panics and device damage. HP-UX device drivers expect devices to be SCSI-2 compliant. Unsupported devices that are only SCSI-CCS compliant may work but their use is discouraged. Use of unsupported devices that are only SCSI-1 compliant is strongly discouraged.

Changing SCSI bus connectivity (recabling) while the system is running is not supported. Switching SCSI device power on or off while the device is connected to a system that does not support powerfail recovery is not supported. These activities are known to cause data loss and system panics.

On systems that support the **scsi_ctl** interface, the **SIOC_CMD_MODE**, **SIOC_SET_CMD**, and **SIOC_RETURN_STATUS** ioctls are obsolete (see *scsi_ctl(7)*). Direct manipulation of SCSI devices via the **scsi_ctl** interface provides a more functionally complete and easier-to-use means of low level SCSI device control (see *scsi_ctl(7)*).

Drivers that support only devices which have no meaningful size may not support the **SIOC_CAPACITY** ioctl. Total device size in bytes may exceed $2^{32}-1$ for some devices.

DEPENDENCIES

esdisk/estape/eschgr/sdisk/schgr/stape

The **SIOC_EXCLUSIVE** ioctl may be used to obtain and release exclusive access. Exclusive access, which prevents simultaneous access by other applications, is required for some operations and may be desirable in other circumstances. The following exclusive access control arguments are supported. The corresponding values are defined in **<sys/scsi.h>** If the ioctl is issued on a persistent device file, target and bus exclusive access actually result to LUN exclusive access.

SIOC_REL_LUN_EXCL	Release exclusive access to logical unit (LUN).
SIOC_SET_LUN_EXCL	Gain exclusive access to logical unit (LUN).
SIOC_REL_TGT_EXCL	Release exclusive access to associated SCSI target.
SIOC_SET_TGT_EXCL	Gain exclusive access to associated SCSI target.
SIOC_REL_BUS_EXCL	Release exclusive access to associated SCSI bus.
SIOC_SET_BUS_EXCL	Gain exclusive access to associated SCSI bus.

The **SIOC_MEDIUM_CHANGED** ioctl indicates when the media in a removable-media device may have changed. A value of "1" indicates the device media may have changed since the last **SIOC_MEDIUM_CHANGED** ioctl call. Note that only the first such call after a media change receives this indication. This means that media changes are likely to be missed if multiple applications are attempting to detect media changes. Exclusive access, obtained through use of the **SIOC_EXCLUSIVE** ioctl, can be used to avoid this problem.

The following information is included from **<sys/scsi.h>**:

```
#define SIOC_MEDIUM_CHANGED  _IOR('S', 42, int)
#define SIOC_EXCLUSIVE      _IOR('S', 68, int)
```

disc3

The **SIOC_VPD_INQUIRY** ioctl allows access to detailed device specific information. The **page_code** field specifies which SCSI vital product data page is requested. The **page_buf** field is filled with the requested page data. This ioctl when issued on a legacy device file only attempts to send the INQUIRY command through the LUN path corresponding to the legacy device file even if multi-pathing is enabled on legacy device files.

The following information is included from **<sys/scsi.h>**:

```
#define SIOC_VPD_INQUIRY  _IOWR('S', 10, struct vpd_inquiry)

/* union for SIOC_VPD_INQUIRY ioctl */
struct vpd_inquiry {
    char    page_code;        /* VPD page code          */
    char    page_buf[126];    /* buffer for VPD page info */
};
```

FILES

/usr/include/sys/scsi.h

SEE ALSO

diskinfo(1M), ioctl(2), autochanger(7), intro(7), scsi_ctl(7), scsi_disk(7), scsi_tape(7).

NAME

scsi_ctl - SCSI pass-through driver (esctl/sctl)

DESCRIPTION

SCSI devices are controlled by a device-specific driver, when one exists. Device-specific drivers, such as those for SCSI direct access (disk) and sequential access (tape) devices, coordinate device and driver states to accomplish correct logical device behavior. The SCSI pass-through driver enables use of SCSI devices and commands not normally supported by these device-specific drivers.

esctl is the SCSI pass-through driver and works with persistent device files (see *intro(7)*). **sctl** is the SCSI pass-through driver already used on HP-UX releases prior to HP-UX 11i V3. It is maintained here for backward compatibility, and works with legacy device files. In this document **scsi_ctl** refers to both **esctl** and **sctl**.

Once the device is opened through **scsi_ctl** driver, ioctl calls can be used to change SCSI communication parameters or attempt SCSI commands and other SCSI operations. Since pass-through driver does not attempt to logically understand the target device, **read()** and **write()** calls are not supported.

Except where noted, the ioctls described here are available through all SCSI device drivers (including device-specific drivers). All **reserved** fields in the data structures associated with these ioctls must be zero-filled.

The following ioctls which are specific to parallel SCSI, are deprecated for issuance on LUN device special files (DSF). They are not supported on persistent device special files. They continue to be supported on legacy device special files for backward compatibility. But, it is recommended now to issue them or equivalent ioctls introduced with HP-UX 11i V3, directly on the parallel SCSI HBA device special file (DSF).

```
SIOC_GET_TGT_PARMS
SIOC_GET_BUS_PARMS
SIOC_GET_TGT_LIMITS
SIOC_GET_BUS_LIMITS
SIOC_SET_TGT_LIMITS
SIOC_SET_BUS_LIMITS
```

The following parallel SCSI specific ioctls introduced with HP-UX 11i V3 should be issued directly on the parallel SCSI HBA DSF. They replace some existing ioctls, which can no longer be issued on LUN persistent device files starting with HP-UX 11i V3:

```
PSIOC_GET_TGT_LIMITS replaces SIOC_GET_TGT_PARMS
PSIOC_GET_TGT_PARMS replaces SIOC_GET_BUS_PARMS
PDIOC_RSTCLR replaces DIIOC_RSTCLR
PSIOC_RESET_DEV replaces SIOC_RESET_DEV
```

Legacy device files are deprecated with HP-UX release 11i V3. They are maintained for backward compatibility, and may be obsolete in a future release (see *intro(7)* for details about legacy device file and persistent device files). It is recommended to use persistent device files for new applications.

Most of the ioctls described here can be issued either on persistent device files or legacy device files. The behavior of some ioctls may differ depending on whether issued on persistent device files or legacy device files, and whether multi-pathing is enabled on legacy device files. Typically ioctls issuing SCSI commands to a device may use any available LUN path to the device to send the commands. However, when multi-pathing is disabled on legacy device files (see **leg_mpath_enable** attribute in *scsimgr(1M)*), the ioctl only attempts to use the LUN path corresponding to the legacy device file. If this LUN path is not available, the ioctl will fail even if there are other LUN paths available. This behavior corresponds to the legacy behavior.

Device Special File Minor Number

The pass-through driver (**esctl/sctl**) is the preferred method to perform the ioctls **SIOC_IO_EXT** (**esctl** only) and **SIOC_IO** ioctls, rather than going through a device-specific driver (such as **esdisk**). To do this, you must create the device special file for the pass-through driver. *mksf(1M)* is the recommended method to create a pass-through device file for **esctl**. To create a device file for the legacy pass-through driver **sctl**, use *mknod(1M)*, substituting the values in the minor number as noted:

```
/usr/sbin/mknod name c 203 0xii10o
```

where component parts of the minor number are constructed as follows:

- ii Two hexadecimal digits, identifying the controlling interface card by its "Instance" number. The Instance value is displayed in *ioscan(1M)* output, under column **I** for the "Interface" hardware type.
- t One hexadecimal digit identifying the drive (target) address.
- l One hexadecimal digit identifying the logical unit number (LUN) within the device.
- 0 Hexadecimal digit zero, for reserved portion of the minor of the minor number.
- o Optional values as follows:
 - 0 To perform Inquiry on open to ensure the device exists (recommended); or
 - 2 To inhibit Inquiry on open. Starting with HP-UX 11i V3, option 2 is deprecated. It is maintained for binary compatibility with existing applications already setting it. Inquiry command will actually be sent during open, regardless of this option being set or not to 2.

SCSI Communication Parameters

HP-UX supports the SCSI device protocol on parallel SCSI interfaces, Fibre Channel interfaces, and Serial Attached SCSI interfaces. The SCSI communication parameters described here might only apply to certain SCSI interfaces and are noted as such in the descriptions.

SCSI communication parameters control features related to communication for three different scope levels: bus (link), target, and logical unit number (LUN). Bus communication parameters apply to all targets connected to a specific bus. Target communication parameters apply to all LUNs associated with a specific target. LUN communication parameters apply to a specific LUN. SCSI communication parameters apply to all device drivers (both device-specific and **scsi_ctl**).

At power-up and after being reset, all parallel SCSI devices and hosts communicate using asynchronous data transfers. Asynchronous data transfers use request (REQ) and acknowledge (ACK) signaling. The strict ordering of REQ and ACK signaling simplifies the communication protocol but limits I/O performance. A SCSI target and host pair may agree to use synchronous data transfers to increase I/O performance.

Synchronous data transfers improve I/O performance by lessening the ordering requirements on REQs and ACKs. By allowing multiple outstanding REQs, signal propagation delays and temporary rate imbalances are better tolerated. To make use of synchronous data transfers, a SCSI target and host must negotiate to determine mutually acceptable maximum REQ-ACK-offset and data-transfer rate parameters.

The maximum REQ-ACK-offset parameter indicates the maximum allowable number of outstanding REQs. The value zero is used to indicate asynchronous data transfer. Other values indicate synchronous data transfer. The appropriate value is generally dependent on the size of the receive data FIFO. High values tend to improve data transfer rates. The maximum data-transfer rate parameter indicates the "burst" data transfer rate (minimum allowable time between successive synchronous data transfers). A SCSI synchronous data transfer request (SDTR) message, used to initiate the negotiation process, is associated with the processing of a SCSI command.

At power-up and after being reset, all parallel SCSI devices and hosts communicate using eight-bit data transfers. A SCSI target and host pair may agree to use sixteen-bit (wide) data transfers to increase I/O performance. To make use of wide data transfers, a SCSI target and host must negotiate to determine a mutually acceptable data transfer width parameter. A SCSI wide data transfer request (WDTR) message, used to initiate the negotiation process, is associated with the processing of a SCSI command.

Some SCSI devices are able to simultaneously manage multiple active commands. Such a device has a command queue that holds commands for processing. Command queuing can improve I/O performance by reducing the time spent by the device waiting for new commands from the host. Note that command queuing might not improve I/O performance substantially for devices that support "read-ahead" and "immediate-reporting" (see *scsi_disk(7)* and *scsi_tape(7)*). The SCSI device and host use command tags to correctly manage these multiple simultaneously active commands. At all times when command queuing is in effect, each active command being handled by a specific LUN has a unique command tag.

SCSI devices indicate their ability to support the special communication features described above in their SCSI **INQUIRY** command data. Normally the SCSI **INQUIRY** command data and negotiation protocols allow hosts and devices to determine the optimal communication parameters so that I/O performance is maximized.

The current operating communication parameters may be determined by use of the: **SIOC_GET_LUN_PARMS**, **PSIOC_GET_TGT_PARMS** (recommended) or **SIOC_GET_TGT_PARMS** (for backward compatibility), and **SIOC_GET_BUS_PARMS** ioctls.

Occasionally, it is desirable to limit SCSI communication parameters to work around a communication problem or to provide external insight in determining optimal parameters. SCSI communication parameter limit suggestions can be specified by use of the: **SIOC_SET_LUN_LIMITS**, **SIOC_SET_TGT_LIMITS**, and **SIOC_SET_BUS_LIMITS** ioctls.

Note that there might be substantial differences between specified communication parameter limit suggestions and the corresponding actual current communication parameters being used for communication. These differences are a result of device-specific driver capabilities, interface driver capabilities, interface hardware capabilities, device capabilities, delays due to the negotiation process, delays due to currently active commands, and delays due to commands waiting to be sent to devices. Note that communication parameter limit suggestions might not survive between **close()** and **open()** calls, when no SCSI device drivers (device-specific or **scsi_ctl**) have associated LUN(s) open.

The current SCSI communication parameter limit suggestions may be determined by use of the **SIOC_GET_LUN_LIMITS**, **SIOC_GET_TGT_LIMITS**, and **SIOC_GET_BUS_LIMITS** ioctls.

Logical unit communication parameters may be managed by use of the **SIOC_GET_LUN_PARMS**, **SIOC_SET_LUN_LIMITS**, and **SIOC_GET_LUN_LIMITS**, **SIOC_RESET_DEV**, **SIOC_RESET_BUS** ioctls.

The **SIOC_GET_LUN_PARMS** ioctl indicates the current LUN communication parameter values. The *max_q_depth* field indicates whether or not tagged queuing is enabled, and if enabled, the maximum number of simultaneously active commands allowed. When *max_q_depth* is zero, tagged queuing is disabled. When it is one, tags are being used but commands are still being serially processed. When it is greater than one, tags are being used and *max_q_depth* specifies the maximum number of simultaneously active commands allowed.

The **SIOC_SET_LUN_LIMITS** ioctl may be used to provide LUN communication parameter limit suggestions. The *max_q_depth* field indicates whether or not tagged queuing should be enabled, and if enabled, the maximum number of simultaneously active commands that should be allowed. The **SIOC_GET_LUN_LIMITS** ioctl indicates the current LUN communication parameter limit suggestions.

Target communication parameters may be managed by use of the **PSIOC_GET_TGT_PARMS** ioctl on any associated HBA DSF, or **SIOC_GET_TGT_PARMS**, **SIOC_SET_TGT_LIMITS**, and **SIOC_GET_TGT_LIMITS** ioctls to any associated LUN.

The **PSIOC_GET_TGT_PARMS** and **SIOC_GET_TGT_PARMS** ioctls indicate the current target communication parameter values. The *width*, *reqack_offset*, and *xfer_rate* fields indicate the currently negotiated data transfer parameters. When *width* is eight, narrow transfers are in effect. When it is sixteen, wide transfers are in effect. When *reqack_offset* is zero, asynchronous transfers are in effect and *xfer_rate* is meaningless. When *reqack_offset* is non-zero, synchronous transfers are in effect and the maximum "burst" data transfer rate is *xfer_rate* words per second, where the size of a word is as indicated in *width*.

The **SIOC_SET_TGT_LIMITS** ioctl specifies the target communication parameter limit suggestions. The *max_width* field specifies maximum bus width that should be used for data transfers. The *max_reqack_offset* field specifies the maximum number of outstanding REQs that should be attempted during data transfers. The *max_xfer_rate* field specifies the maximum "burst" data rate that should be allowed during synchronous data transfers. The **SIOC_GET_TGT_LIMITS** ioctl indicates the current target communication parameter limit suggestions. The *width*, *reqack_offset*, *xfer_rate*, *max_width*, *max_reqack_offset*, *max_xfer_rate* fields only apply to parallel SCSI.

Bus communication parameters may be managed by use of the **SIOC_GET_BUS_PARMS**, **SIOC_SET_BUS_LIMITS**, and **SIOC_GET_BUS_LIMITS** ioctls to any associated LUN.

The **SIOC_GET_BUS_PARMS** ioctl indicates the current bus communication parameter values. The *max_width* field indicates the maximum data transfer width that will be attempted for data transfers to any target device connected to the associated bus. The *max_reqack_offset* field indicates the maximum number of outstanding REQs that will be attempted during data transfers to any target device connected to the associated bus. The *max_xfer_rate* field indicates the maximum "burst" data transfer rate that will be attempted for data transfers to any target device connected to the associated bus.

The **SIOC_SET_BUS_LIMITS** ioctl specifies the bus communication parameter limit suggestions for targets connected to the associated bus. The *max_width* field specifies the suggested maximum data transfer width that should be attempted for data transfers to any target device connected to the associated bus. The *max_reqack_offset* field specifies the maximum number of outstanding REQs that should be attempted during data transfers to any target device connected to the associated bus. The *max_xfer_rate* field specifies the maximum synchronous "burst" data transfer rate that should be attempted for data transfers to any target device connected to the associated bus. The **SIOC_GET_BUS_LIMITS** ioctl indicates the current

bus communication parameter limit suggestions. The *max_width*, *max_reqack_offset*, and *max_xfer_rate* fields only apply to parallel SCSI.

The following is included from `<sys/scsi.h>`:

```

/* SCSI communication parameter ioctls */
#define SIOC_GET_LUN_PARMS      _IOR('S', 58, struct sioc_lun_parms)
#define SIOC_GET_TGT_PARMS     _IOR('S', 59, struct sioc_tgt_parms)
#define SIOC_GET_BUS_PARMS     _IOR('S', 60, struct sioc_bus_parms)
#define SIOC_GET_LUN_LIMITS    _IOR('S', 61, struct sioc_lun_limits)
#define SIOC_GET_TGT_LIMITS    _IOR('S', 62, struct sioc_tgt_limits)
#define SIOC_GET_BUS_LIMITS    _IOR('S', 63, struct sioc_bus_limits)
#define SIOC_SET_LUN_LIMITS    _IOW('S', 64, struct sioc_lun_limits)
#define SIOC_SET_TGT_LIMITS    _IOW('S', 65, struct sioc_tgt_limits)
#define SIOC_SET_BUS_LIMITS    _IOW('S', 66, struct sioc_bus_limits)

struct sioc_lun_parms {
    unsigned int flags;
    unsigned int max_q_depth;          /* maximum active I/O's */
    unsigned int reserved[4];         /* reserved for future
    * use
    */
} sioc_lun_parms_t;

struct sioc_lun_limits {
    unsigned int flags;
    unsigned int max_q_depth;
    unsigned int reserved[4];         /* reserved for
    * future use
    */
} sioc_lun_limits_t;

typedef struct sioc_tgt_parms {
    unsigned int flags;
    unsigned int width;                /* bits per word */
    unsigned int xfer_rate;            /* words per second */
    unsigned int reqack_offset;        /* REQ/ACK offset */
    unsigned int tgt_id;               /* target Id */
    unsigned int reserved[3];         /* reserved
    * for future use
    */
} sioc_tgt_parms_t;

typedef struct sioc_tgt_limits {
    unsigned int flags;
    unsigned int max_width;            /* Bits per word */
    unsigned int max_xfer_rate;        /* Words per second */
    unsigned int max_reqack_offset;    /* REQ/ACK offset */
    unsigned int tgt_id;               /* target Id */
    unsigned int reserved[3];         /* Reserved for future
    * use
    */
} sioc_tgt_limits_t;

struct sioc_bus_parms {
    unsigned int flags;                /* reserved for future
    * use
    */

    unsigned int max_width;
    unsigned int max_reqack_offset;
    unsigned int max_xfer_rate;        /* bytes/sec */
    unsigned int reserved[4];         /* reserved for future
    * use

```

```

    */
} sioc_bus_parms_t;

struct sioc_bus_limits {
    unsigned int flags;           /* reserved for future
    * use
    */

    unsigned int max_width;
    unsigned int max_reqack_offset;
    unsigned int max_xfer_rate; /* bytes/sec */
    unsigned int reserved[4];    /* reserved for future
    * use
    */
} sioc_bus_limits_t;

```

The following is included from `<sys/pscsi.h>`:

```

#define PSIOC_GET_TGT_PARMS    _IOWR('S', 114, struct sioc_tgt_parms)
#define PSIOC_GET_TGT_LIMITS  _IOWR('S', 115, struct sioc_tgt_limits)
#define PSIOC_RESET_DEV      _IOW('S', 116, int)
#define PDIOC_RSTCLR         _IOW('S', 117, int)

```

SCSI Commands and Operations

SIOC_IO_EXT and **SIOC_IO** ioctls allow an arbitrary SCSI command to be sent to a device. All details of the SCSI command protocol are handled automatically. **SIOC_IO_EXT** should only be issued on persistent device files. It allows to send the SCSI command through any available LUN path or through a selected LUN path. **SIOC_IO** is deprecated. It can be issued on both persistent and legacy device files. When issued on a persistent device file, the SCSI command is sent through any available LUN path.

The following flags can be used to specify the *flags* field value of both **SIOC_IO_EXT** and **SIOC_IO**, unless indicated otherwise:

SCTL_READ	Data read operation is expected if <i>data_length</i> field is non-zero. The absence of this flag implies that data write operation is expected if the <i>data_length</i> field is non-zero.
SCTL_INIT_SDTR	Synchronous data transfer request negotiations should be attempted with this command. This flag only applies to parallel SCSI and is maintained for backward compatibility.
SCTL_INIT_WDTR	Wide data transfer request negotiations should be attempted with this command. This flag only applies to parallel SCSI and is maintained for backward compatibility.
SCTL_NO_DISC	discprv bit in Identify message is not set. This flag only applies to parallel SCSI and is maintained for backward compatibility.
ESCTL_IO_LPT	The SCSI command is to be issued on a given LUN path. This flag can only be specified with SIOC_IO_EXT ioctl. When specified the hardware path of the LUN path to use is specified in field <i>lpt_hwp</i>

The *cdb* field specifies the SCSI command bytes. The number of command bytes is specified by the *cdb_length* field. These command bytes are sent to the target device during the SCSI command phase.

The address of the data area for the data phase of the SCSI command is specified by the *data* field. The *data_length* field specifies the maximum number of data bytes to be transferred. A zero-valued *data_length* indicates that no data phase should occur. Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. The caller is responsible for correctly specifying both the *data_length* field and any *cdb* data length values. The length may not be larger than **SCSI_MAXPHYS** and some implementations further restrict this length.

The *max_msecs* field specifies the maximum time, in milliseconds, that the device should need to complete the command. If this period of time expires without command completion, the system might attempt recovery procedures to regain the device's attention. These recovery procedures might include abort tag, abort, and device and bus reset operations. A zero value in the *max_msecs* field indicates that the timeout period is infinite and the system should wait indefinitely for command completion.

When the **SIOC_IO_EXT** or **SIOC_IO** ioctl call returns, all command processing has been completed. Most **SIOC_IO_EXT/SIOC_IO** ioctl calls will return zero (success). The resulting detailed ioctl data

should be used to evaluate "success" or "failure" from the caller's perspective. The *cdb_status* field indicates the results of the **cdb** command. If the *cdb_status* field indicates a **S_CHECK_CONDITION** status, the *sense_status* field indicates the results of the **SCSI REQUEST SENSE** command used to collect the associated sense data. These status fields will contain one of the following values:

SCTL_INVALID_REQUEST	The SCSI command request is invalid and thus not attempted.
SCTL_SELECT_TIMEOUT	The target device does not answer to selection by the host SCSI interface (the device does not exist or does not respond).
SCTL_INCOMPLETE	The device answered selection but the command is not completed (the device took too long or a communication failure occurred).
S_GOOD	Device successfully completed the command.
S_CHECK_CONDITION	Device indicated sense data is available.
S_CONDITION_MET	Device successfully completed the command and the requested (search or pre-fetch) operation is satisfied.
S_BUSY	Device indicated it is unable to accept the command because it is busy doing other operations.
S_INTERMEDIATE	Device successfully completed this command, which is one in a series of linked commands (not supported, see <i>WARNINGS</i>).
S_I_CONDITION_MET	Device indicated both S_INTERMEDIATE and S_CONDITION_MET (not supported, see <i>WARNINGS</i>).
S_RESV_CONFLICT	Device indicated the command conflicted with an existing reservation.
S_COMMAND_TERMINATED	Device indicated the command is terminated early by the host system.
S_QUEUE_FULL	Device indicated it is unable to accept the command because its command queue is currently full.

The *data_xfer* field indicates the number of data bytes actually transferred during the data phase of the **cdb** command. This field is valid only when the *cdb_status* field contains one of the following values: **S_GOOD** or **S_CHECK_CONDITION**. The *sense_xfer* field indicates the number of valid sense data bytes. This field is valid only when the *cdb_status* field contains the value **S_CHECK_CONDITION** and the *sense_status* field contains the value **S_GOOD**.

The **SIOC_ABORT** ioctl causes a SCSI **ABORT** message to be sent to the LUN. This clears all active commands to the LUN from this initiator.

The **SIOC_TASK_MGMT** ioctl causes a SCSI task management function to be performed if supported by the SCSI transport. The following task management function values can be specified. They are defined in `<sys/scsi.h>`:

SIOC_TM_LUN_RESET	Lun Reset
SIOC_TM_WARM_TGT_RESET	Warm Target Reset
SIOC_TM_COLD_TGT_RESET	Cold Target Reset

The **SIOC_RESET_DEV** ioctl causes a SCSI device to be reset (including clearing all active commands). On parallel SCSI a **PSIOC_RESET_DEV** and **SIOC_RESET_DEV** ioctls cause a SCSI **BUS DEVICE RESET** message to be sent to the associated target. On Fibre Channel a **SIOC_RESET_DEV** ioctl causes a "TARGET RESET" task management function to be sent to the associated target followed by a Global Process Logout (GPRLO).

The **SIOC_RESET_BUS** ioctl causes the system to generate a SCSI bus reset condition on the associated bus. A SCSI bus reset condition causes all devices on the bus to be reset (including clearing all active commands on all devices). The **SIOC_RESET_BUS** ioctl does not apply to Fibre Channel.

Often it is necessary or useful to prohibit other SCSI commands while performing device-control operations. This should be done by gaining exclusive access via the **SIOC_EXCLUSIVE** ioctl. The associated argument points to an integer with one of these values defined in `<sys/scsi.h>`. Note that if the ioctl is issued on a persistent device file, target and bus exclusive access requests result to a LUN exclusive access being performed.

SIOC_REL_LUN_EXCL	release exclusive access to logical unit
SIOC_SET_LUN_EXCL	obtain exclusive access to logical unit

SIOC_REL_TGT_EXCL	release exclusive access to target
SIOC_SET_TGT_EXCL	obtain exclusive access to target
SIOC_REL_BUS_EXCL	release exclusive access to bus
SIOC_SET_BUS_EXCL	obtain exclusive access to bus

The ioctl **SIOC_PRIORITY_MODE** is deprecated with HP-UX release 11i V3. If called, it will just fake success. This ioctl was used to workaround situations where it is not possible to set exclusive access to the device. It put the device in "Priority mode". This caused all device-specific driver I/O operations (for example, file system I/O and virtual memory page swapping) and all SCSI device driver open calls (including pass-through driver open calls) to the associated LUN to block. These I/O operations and open calls were blocked for the entire duration that priority mode was in effect. While priority mode was in effect only **SIOC_IO** operations could be attempted. (these operations will not be blocked). It was very easy to cause system deadlock through incorrect use of the **SIOC_PRIORITY_MODE** ioctl. It normally required to lock the calling process into memory (see *plock(2)*) prior to enabling priority mode.

The header file `<sys/scsi.h>` has useful information for SCSI device control. The following is included from `<sys/scsi.h>`:

```

/* SCSI device control ioctls */
#define SIOC_IO          _IOWR('S', 22, struct sctl_io)
#define SIOC_RESET_DEV  _IO('S', 16)
#define SIOC_RESET_BUS  _IO('S', 9)
#define SIOC_PRIORITY_MODE _IOW('S', 67, int)

#define SIOC_IO_EXT      _IOWR('S', 102, esctl_io_t)
#define SIOC_TASK_MGMT   _IOWR('S', 104, sioc_task_mgmt_t)

/* Structure for SIOC_IO_EXT ioctl */
typedef struct {
    int                version;
    escsi_sctl_io_flags_t flags;
    int                max_msecs;
    uint32_t           cdb_length;
    uint32_t           data_length;
    ptr64_t            data;
    union sense_data   sense;
    escsi_hw_path_t    lpt_hwp;
    uint32_t           data_xfer;
    uint32_t           sense_xfer;
    uint32_t           cdb_status;
    uint32_t           sense_status;
    uint32_t           cdb[ESCSI_MAX_CDB_LEN];
    uint8_t            rsvd[32]; /* Reserved for
                                * future use
                                */
} esctl_io_t;

/* Structure for SIOC_IO ioctl */
struct sctl_io
{
    unsigned           flags;
    unsigned char     cdb_length;
    unsigned char     cdb[16];
    void               *data;
    unsigned           data_length;
    unsigned           max_msecs;
    unsigned           data_xfer;
    unsigned           cdb_status;
    unsigned char     sense[256];
    unsigned           sense_status;
    unsigned char     sense_xfer;
    unsigned char     reserved[64];
} sctl_io_t;

```


Security Restrictions

Superuser or **DEVOPS** privilege, or device write permissions are required to use these ioctls. See *privileges(5)* for more information about privileged access on systems that support fine-grained privileges.

EXAMPLES

Assume that *fildev* is a valid file descriptor for a persistent device file of a SCSI device, and *leg_fildev* is a valid file descriptor for a legacy device file of a SCSI device, and *lpt_hwp* contains a valid hardware path of a LUN path to the device. The first example attempts a SCSI **INQUIRY** command:

```
#include <sys/scsi.h>

esctl_io_t esctl_io;
#define MAX_LEN 255
unsigned char inquiry_data[MAX_LEN];

memset(&esctl_io, 0, sizeof(esctl_io)); /* clear reserved fields */
esctl_io.flags = SCTL_READ; /* input data expected */
esctl_io.cdb[0] = CMDInquiry;
esctl_io.cdb[1] = 0x00;
esctl_io.cdb[2] = 0x00;
esctl_io.cdb[3] = 0x00;
esctl_io.cdb[4] = MAX_LEN; /* allocation length */
esctl_io.cdb[5] = 0x00;
esctl_io.cdb_length = 6; /* 6 byte command */
esctl_io.data = &inquiry_data[0]; /* data buffer location */
esctl_io.data_length = MAX_LEN; /* maximum transfer length */
esctl_io.max_msecs = 10000; /* allow 10 seconds for cmd */
if (ioctl(fildev, SIOC_IO_EXT, &esctl_io) < 0) {
    /* request is invalid */
} else {
    if ( esctl_io.cdb_status == S_GOOD) {
        /* success. display inquiry data */
    } else {
        /* failure. process depending on cdb_status */
    }
}
}
```

The second example attempts a SCSI **INQUIRY** command via a specific LUN path.

```
#include <sys/scsi.h>

esctl_io_t esctl_io;
#define MAX_LEN 255
unsigned char inquiry_data[MAX_LEN];

memset(&esctl_io, 0, sizeof(esctl_io)); /* clear reserved fields */
esctl_io.flags = SCTL_READ | SCTL_IO_LPT; /* input data
* expected and command
* to be sent on given
* LUN path
*/
memcpy(&esctl_io.lpt_hwp, lpt_hwp, sizeof(lpt_hwp)); /* specify
* the hardware path of
* LUN path through which
* command must be sent
*/

esctl_io.cdb[0] = CMDInquiry;
esctl_io.cdb[1] = 0x00;
esctl_io.cdb[2] = 0x00;
esctl_io.cdb[3] = 0x00;
esctl_io.cdb[4] = MAX_LEN; /* allocation length */
esctl_io.cdb[5] = 0x00;
esctl_io.cdb_length = 6; /* 6 byte command */
esctl_io.data = &inquiry_data[0]; /* data buffer location */
```

S

```

esctl_io.data_length = MAX_LEN;          /* maximum transfer length */
esctl_io.max_msecs = 10000;             /* allow 10 seconds for cmd */
if (ioctl(fildes, SIOC_IO_EXT, &esctl_io) < 0) {
    /* request is invalid */
} else {
    if ( esctl_io.cdb_status == S_GOOD) {
        /* success. display inquiry data */
    } else {
        /* failure. process depending on cdb_status */
    }
}

```

The following example attempts a SCSI **TEST UNIT READY** command and checks to see if the device is ready, not ready, or in some other state.

```

#include <sys/scsi.h>

struct sctl_io sctl_io;

memset(&sctl_io, 0, sizeof(sctl_io)); /* clear reserved fields */
sctl_io.flags = 0;                    /* no data transfer expected */
sctl_io.cdb[0] = 0x00;                 /* can use CMDtest_unit_ready */
sctl_io.cdb[1] = 0x00;
sctl_io.cdb[2] = 0x00;
sctl_io.cdb[3] = 0x00;
sctl_io.cdb[4] = 0x00;
sctl_io.cdb[5] = 0x00;
sctl_io.cdb_length = 6;                /* 6 byte command */
sctl_io.data = NULL;                  /* no data buffer is provided */
sctl_io.data_length = 0;               /* do not transfer data */
sctl_io.max_msecs = 10000;            /* allow 10 seconds for cmd */
if (ioctl(leg_fildes, SIOC_IO, &sctl_io) < 0) {
    /* request is invalid */
} else if (sctl_io.cdb_status == S_GOOD) {
    /* device is ready */
} else if (sctl_io.cdb_status == S_BUSY ||
           (sctl_io.cdb_status == S_CHECK_CONDITION &&
            sctl_io.sense_status == S_GOOD &&
            sctl_io.sense_xfer > 2 &&
            (sctl_io.sense[2] & 0x0F) == 2)) {
    /* can use sense_data */
    /* device is not ready */
} else {
    /* unknown state */
}

```

WARNINGS

Incorrect use of **scsi_ctl** operations (even those attempting access to non-existent devices) can cause data loss, system panics, and device damage.

The **SIOC_EXCLUSIVE** ioctl should be used to gain exclusive access to a device prior to attempting **SIOC_IO** commands. If exclusive access is not obtained, **SIOC_IO** commands will be intermixed with device-specific driver commands, which can lead to undesirable results.

Device-specific drivers can reject inappropriate or troublesome **SIOC_IO_EXT/SIOC_IO** commands. However, since not all such operations are known and detected, care should be exercised to avoid disrupting device-specific drivers when using commands that modify internal device states.

Most SCSI commands have a logical unit number (LUN) field. Parallel SCSI implementations on the HP-UX operating system select logical units via the SCSI **IDENTIFY** message. The LUN portion of the cdb should normally be set to zero, even when the LUN being accessed is not zero.

Use of linked commands is not supported.

Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. Both the *data_length* field and any cdb data length values must be correctly specified

to get correct command results.

Very large (or infinite) timeout values can cause a parallel SCSI bus (potentially the entire system) to hang.

Device and/or bus reset operations can be used to regain a device's attention when a timeout expires.

Resetting a device can cause I/O errors and/or loss of cached data. This can result in loss of data and/or system panics.

Obtaining SCSI **INQUIRY** data by use of the **SIOC_INQUIRY** ioctl instead of by use of the **SIOC_IO** ioctl is generally preferable since SCSI implementations on the HP-UX operating system synchronize access of inquiry data during driver open calls.

Since communication parameters can be affected by device-specific driver capabilities, device-specific driver use might result in communication parameter changes.

The **SIOC_CAPACITY** ioctl is not supported by **scsi_ctl** because the meaning of capacity is device-specific.

FILES

/usr/include/sys/scsi.h
/usr/include/sys/scsi_ctl.h

SEE ALSO

mknod(1M), mksf(1M), ioctl(2), plock(2), privileges(5), intro(7), scsi(7).

NAME

scsi_disk - SCSI direct access device drivers (esdisk/sdisk)

DESCRIPTION

This section describes the interface for access of SCSI disk, CD-ROM, and optical disk devices through the character special device driver. **esdisk** is the default driver for direct access devices starting at HP-UX 11i Version 3. **sdisk** is the default driver used on HP-UX 11i Version 2 and earlier releases. It is maintained for backward compatibility.

SCSI direct access devices store a sequence of data blocks. Each direct access device has a specific device size consisting of a number of data blocks and a logical block size. All data blocks have the same logical block size.

Since I/O operations must have a size that is an integral number of blocks, one logical block size is the smallest possible I/O quantity. The device block size can be determined through use of the **DIOC_DESCRIBE**, **DIOC_CAPACITY**, **SIOC_CAPACITY**, **DIOC_DESCRIBE_EXT**, and **SIOC_STORAGE_CAPACITY** ioctls (see *disk(7)* and *scsi(7)*; **SIOC_CAPACITY** is not supported on **disc3**). A direct access device that is not ready for use, whether due to no media installed or another reason, is interpreted to mean the device has zero size. An **open()** call to such a device succeeds, but subsequent **read()** and **write()** calls fail.

The *ioctl(2)* manpage explains how the operations and arguments are used. Note, the *arg* used is commonly the address of the parameter cited in the particular ioctl **#define** statement. See the *EXAMPLES* section for sample code.

To improve performance, many SCSI disk devices have caches, which can be used for both read and write operations.

Read cache use, called "read ahead", causes the disk drive to read data in anticipation of read requests. Read ahead is only apparent to users in the increased performance that it produces.

Write cache use is called "immediate reporting". Immediate reporting increases I/O performance by reporting a completed write status before the data being written is actually committed to media. If the subsequent physical write operation does not complete successfully, data may be lost.

Physical write failures due to media defects are largely eliminated by use of automatic sparing in disk drives. Power failure between immediate reporting and media commit can result in cached data being lost. However, the period of time between these events is typically relatively small, making such losses unlikely.

The **SIOC_GET_IR** ioctl can be used to determine if immediate-reporting functionality is currently being used by the device. The value **1** indicates immediate reporting is enabled. The value zero indicates immediate reporting is disabled. The **SIOC_SET_IR** ioctl can be used to enable or disable immediate reporting. A zero value disables immediate reporting. The value **1** enables immediate reporting.

The **SIOC_SYNC_CACHE** ioctl can be used to force data cached in the device to media.

Most SCSI removable media disk devices support "prevent" and "allow" media-removal commands. To avoid data corruption and data accessibility problems, media removal is prevented for the entire duration a removable media disk device is open. Because media removal is not supported, the **SIOC_MEDIUM_CHANGED** ioctl is not supported.

The header file `<sys/scsi.h>` has useful information for direct access device control, including the following:

```
/* ioctl support for SCSI disk devices */
#define SIOC_GET_IR      _IOR('S', 14, int)
#define SIOC_SET_IR      _IOW('S', 15, int)
#define SIOC_SYNC_CACHE _IOW('S', 70, int)
```

The **SIOC_FORMAT** ioctl reformats the entire media surface. Exclusive access to the device, obtained through use of the **DIOC_EXCLUSIVE** ioctl (see *disk(7)*), is required prior to reformatting to ensure that other applications are not affected. The **fmt_optn** field can be used to select the desired media geometry. Only one media geometry is supported on most devices. The value zero should be used for these devices. The value zero can also be used to select the default geometry on devices that support multiple media geometries. The **interleave** field can be used to specify sector interleaving. The value zero specifies that an appropriate default interleave should be used.

EXAMPLES

The following sample code shows how to use ioctls that affect `scsi_disk`.

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/errno.h>
#include <sys/diskio.h>
#include <sys/scsi.h>
Describe_ext (dfd)
    int dfd;
{
    int ret;
    disk_describe_type_ext_t disk_descr;
    uint64_t capacity;

    if ((ret = ioctl (dfd, DIOC_DESCRIBE_EXT, &descr_type)) != 0) {
        exit(1);
    }
    printf("\nSuccessful ioctl DIOC_DESCRIBE_EXT \n");
    printf("  model number: %s\n", disk_descr.model_num);
    printf("  interface:      %d <20=scsi>\n", disk_descr.intf_type);
    capacity = (disk_descr.maxsva_high << 32) + disk_descr.low_lba;
    printf("  Capacity:      %llu (blocks)\n", capacity);
    printf("  block size:    %u (bytes)\n", disk_descr.lgblksz);
    printf("  Device type:   %u (0=disk, 5=CD, 7=OM)\n",
        disk_descr.dev_type);
    printf("  Write Protected: %s \n",
        disk_descr.flags & WRITE_PROTECT_FLAG ? "yes" : "No");
}

Describe (dfd)
    int dfd;
{
    int ret;
    disk_describe_type descr_type;
    if ((ret = ioctl (dfd, DIOC_DESCRIBE, &descr_type)) != 0) {
        exit(1);
    }
    printf ("\nSuccessful ioctl DIOC_DESCRIBE \n");
    printf ("  model number: %s\n", descr_type.model_num);
    printf ("  interface:      %d <20=scsi>\n", descr_type.intf_type);
}

Exclusive (dfd)
    int dfd;
{
    int ret, flag=1;
    if ((ret = ioctl (dfd, DIOC_EXCLUSIVE, &flag)) != 0) {
        exit(1);
    }
}

Enable_WOE (dfd)
    int dfd;
{
    int ret, flag=1;
    if ((ret = ioctl (dfd, SIOC_WRITE_WOE, &flag)) != 0) {
        exit(1);
    }
    printf ("\nSuccessful ioctl SIOC_WRITE_WOE \n");
}

main (argc, argv)
    int argc;
```

```

char ** argv;
{ int ret, fd; if (argc != 2) {
    printf ("Usage: %s <disk_device> \n", argv[0]);
    exit(1);
}
if ((fd = open (argv[1], O_RDWR)) < 0) {
    exit (1);
}
Describe_ext (fd);
Describe (fd);
Exclusive (fd);
Enable_WOE (fd);
}

```

WARNINGS

Historically, disk devices have had small (typically 512 byte) block sizes; however, many newer disk devices (such as optical disks and disk arrays) have relatively large block sizes. Applications using direct raw disk access should use the **DIOC_DESCRIBE**, **DIOC_CAPACITY**, **DIOC_DESCRIBE_EXT**, or **SIOC_CAPACITY** ioctl to determine the appropriate minimum I/O size.

Media removal and insertion while a disk device is open is unsupported and unpredictable. Do not attempt to circumvent prevention of media removal. Device capacity changes resulting from such intervention may not be recognized.

Often larger I/O operation sizes are expected to be more efficient. However, SCSI disk I/O operations that are large relative to the device's cache can result in insufficient cache space for the device to maintain full-media-speed data transfer rates. This can result in decreased I/O performance relative to smaller I/O sizes.

DEPENDENCIES

Optical Disk Devices

The **SIOC_VERIFY_WRITES** ioctl controls the write mode. Normally written data is assumed to be correctly stored on the media. Verify-writes mode causes verification of written data to ensure that data has been correctly written. Verification can substantially reduce write performance and is not generally needed.

The **SIOC_VERIFY_WRITES** ioctl can be used to enable or disable write verification. A zero value disables write verification. The value **1** enables write verification. Although write verification is primarily intended for optical media, some systems may support write verification on normal disk devices.

The **SIOC_VERIFY** ioctl verifies that a media area contains valid data (that is, data that has been correctly written). Verified media will not cause I/O errors when reading is attempted. The media area to be verified is specified via the **start_lba** and **block_cnt** fields. Although verification is intended primarily for optical media, some systems may support verify operations on normal disk devices.

The **SIOC_WRITE_WOE** ioctl controls the write mode used for magneto-optical disk devices. Normally magneto-optical write operations require two physical head passes. The first pass erases the media area to be written. The second pass actually writes the data. Write-without-erase mode dramatically increases write performance by skipping the first (erase media area) pass. To ensure that the correct data results, it is essential that write-without-erase operations be performed only on media that is known to be blank (previously erased or never used). The **SIOC_WRITE_WOE** ioctl can be used to enable or disable write-without-erase. A zero value disables write-without-erase. The value **1** enables write-without-erase.

The **SIOC_ERASE** ioctl allows media areas to be explicitly erased. The media area to be erased is specified via the **start_lba** and **block_cnt** fields. Media areas erased in this manner can be written using write-without-erase mode. Note that an erased media area is different from a media area written with some data values (e.g. zeros). An erased media area should not be read. Attempting to read an erased media area generally results in an I/O error.

The **SIOC_VERIFY_BLANK** ioctl verifies that a media area has been erased and is suitable for being written using write-without-erase mode. The media area to be verified is specified via the **start_lba** and **block_cnt** fields.

The following optical disk device specific information is included from **<sys/scsi.h>**:

```

#define SIOC_WRITE_WOE          _IOW('S', 17, int)
#define SIOC_VERIFY_WRITES     _IOW('S', 18, int)
#define SIOC_ERASE             _IOW('S', 19, struct scsi_erase)

```

```
#define SIOC_VERIFY_BLANK      _IOW('S', 20, struct scsi_verify)
#define SIOC_VERIFY          _IOW('S', 21, struct scsi_verify)

/* structure for SIOC_ERASE ioctl */
struct scsi_erase {
    unsigned int    start_lba;
    unsigned short  block_cnt;
};

/* structure for SIOC_VERIFY_BLANK and SIOC_VERIFY ioctls */
struct scsi_verify {
    unsigned int    start_lba;
    unsigned short  block_cnt;
};
```

FILES

/usr/include/sys/scsi.h

SEE ALSO

mediainit(1), mknod(1M), ioctl(2), disk(7), scsi(7).

NAME

scsi_tape - SCSI sequential access device driver

DESCRIPTION

SCSI sequential-access (tape) devices store a sequence of data blocks. Data can be read and written using either fixed or variable sized block mode. If supported by the device, variable sized block mode is normally used (even when all blocks are the same size). Fixed sized block mode is generally only used for tape devices which do not support variable sized blocks. Fixed sized block mode can be used on some tape devices which support variable sized blocks to increase I/O performance.

Generally SCSI tape devices are controlled through the **mt** (see *mt(7)*) generic tape device interface. This section describes features that are specific to SCSI tape devices.

The **SIOC_CAPACITY** ioctl (see *scsi(7)*) can be used to determine remaining tape capacity for some tape devices. The **blkksz** field indicates the "natural" block size of the device. This value may or may not be the current block size of the device. The number of blocks, indicated by the **lba** field, is an estimate of how much data can be written on the remaining media. A zero size is returned for devices that do not provide remaining-capacity information. The quantity of data that can actually be written may be higher or lower than indicated, depending on such factors as block size, media defects, data compression, and ability to maintain streaming.

To improve performance, most SCSI tape devices have caches. Read-cache use, called "read ahead", causes the tape drive to read data in anticipation of read requests. Read ahead is only apparent to users in the increased performance that it produces. Write-cache use is called "immediate reporting". Immediate reporting increases I/O performance by reporting a completed write status before the data being written is actually committed to media. This allows the application program to supply additional data so that continuous media motion, called "streaming", can be achieved. The **SIOC_GET_IR** ioctl can be used to determine if immediate-reporting functionality is currently being used by the device. The value "1" indicates immediate reporting is enabled. By default, the device driver attempts to enable immediate reporting. The **SIOC_SET_IR** ioctl can be used to explicitly enable or disable immediate reporting. A zero value disables immediate reporting. The value "1" enables immediate reporting. The **MTIOCTOP** ioctl **MTNOP** command can be used to cause any cached data to be written (committed) to media. Note that the device immediate reporting mode set by the **SIOC_SET_IR** ioctl survives between **close()** and **open()** calls, but not through system reboot.

The **SIOC_GET_BLOCK_SIZE** ioctl indicates the device's current block size. A block size of zero indicates the device is in variable-sized-block mode. A non-zero block size indicates the device is in fixed-sized-block mode.

The **SIOC_SET_BLOCK_SIZE** ioctl changes the current block size to the specified number of bytes. Setting the block size to zero specifies that variable-sized-block mode should be used. Any non-zero block size specifies that fixed-sized-block mode should be used. By default, the device driver attempts to set the block size to zero during open. If variable-sized-block mode is not supported by the device, the driver selects an appropriate block size for fixed-sized-block mode use. Note that the device block size set by the **SIOC_SET_BLOCK_SIZE** ioctl survives between **close()** and **open()** calls, but not through system reboot.

The **SIOC_GET_BLOCK_LIMITS** ioctl indicates the device's maximum and minimum fixed block-size limits. The device's minimum fixed block size is indicated by the **min_blk_size** field. The **max_blk_size** field contains the smaller of the maximum block size supported by the device and the maximum block size supported by the system (**MAXPHYS**). This is the largest valid block size for the specific combination of device, driver, and host system being used.

The **SIOC_GET_POSITION** ioctl can be used to determine the current media position for some devices. For devices that support this capability, the resultant value can be used to reposition the media to the same position in the future.

The **SIOC_SET_POSITION** ioctl can be used to cause media repositioning on some devices. For devices that support this capability, media repositioning via this mechanism can generally be completed more quickly than might be similarly accomplished using record, filemark, or setmark spacing. The argument value specified should be the result of a previous **SIOC_GET_POSITION** for that media volume.

The following is included from **<sys/scsi.h>**:

```
/* ioctl support for SCSI tape commands */
#define SIOC_GET_IR          _IOR('S', 14, int)
#define SIOC_SET_IR         _IOW('S', 15, int)
```



```

#define SIOC_GET_BLOCK_SIZE      _IOR('S', 30, int)
#define SIOC_SET_BLOCK_SIZE      _IOW('S', 31, int)
#define SIOC_GET_BLOCK_LIMITS    _IOW('S', 32, struct scsi_block_limits)
#define SIOC_GET_POSITION        _IOR('S', 33, int)
#define SIOC_SET_POSITION        _IOW('S', 34, int)

/* structure for SIOC_GET_BLOCK_LIMITS ioctl */
struct scsi_block_limits {
    unsigned min_blk_size;
    unsigned max_blk_size;
};

```

WARNINGS

SCSI bus and device resets cause some devices to reposition media to beginning-of-tape (BOT). This unintentional media repositioning can cause loss of data. The **scsi_tape** driver causes the first subsequent **open()** attempt to fail as an indication of potential data loss.

The **scsi_tape** driver does not write filemarks at close if the media has been programmatically repositioned. Applications that reposition the media prior to closing the device should write any required tape-marks.

SEE ALSO

mknod(1M), mt(7), scsi(7).

NAME

scsimgr_eschgr - SCSI class driver eschgr plug-in for scsimgr

DESCRIPTION

The SCSI class driver **eschgr** plug-in for **scsimgr** implements management and diagnostic operations specific to classes of devices bound to the **eschgr** driver. **eschgr** is the native HP-UX SCSI class driver that handles, by default, all library/changer devices.

The plug-in handles the following operations for driver **eschgr**:

- Display and clear driver **eschgr** global statistics and the statistics it maintains on instances of LUNs bound to it, and on related LUN paths.
- Display status and other information maintained by driver **eschgr** on LUNs bound to it.
- Get, set and save driver **eschgr** global and per-lun instance attributes.

Commands

The user can explicitly send the following **scsimgr** commands to driver **eschgr** plug-in by specifying the **-d eschgr** option:

clear_stat Clears statistics.
get_attr Displays information on attributes.
get_info Displays status and other information.
get_stat Displays statistics.
save_attr Saves value of attributes in a persistent store.
set_attr Set current values of attributes.

Note: Refer to *scsimgr*(1M) for the syntax of the above commands.

However, the only instances where it is necessary to explicitly send a command to the plug-in is when performing operations on objects global to driver **eschgr**: global statistics, attributes or status information. In all the other cases, **scsimgr** automatically invokes the plug-in to perform the driver specific part of the operation, when the operation applies to LUNs bound to driver **eschgr** or to their LUN paths.

Attributes

The following table lists driver **eschgr** specific attributes. For details on the concept of attribute refer to *scsimgr*(1M).

Note: The following conventions are used:

- RO is Read Only.
- RW is Read Write.
- uint32 is unsigned 32 bits integer.

Object	Attribute Name	RO/RW	Type	Description
Global	version	RO	string	Version of driver eschgr
LUN	default_secs	RW	uint32	Timeout for all commands not referenced below. Default: 30
	move_secs	RW	uint32	Timeout for the move command. Default: 1200
	readelem_secs	RW	uint32	Timeout for the read element status command. Default: 600
	initelem_secs	RW	uint32	Timeout for the initialize element status command. Default: 600

readaddr_secs	RW	uint32	Timeout for modesense 0x1D command. Default: 600
exchange_secs	RW	uint32	Timeout for the exchange command. Default: 600

I/O Load Balancing and Multi-Pathing Policies

The **eschgr** driver does not support load balancing and has minimal support for multi-pathing.

When the device is first opened after a system boot, a path is chosen and will remain fixed. If the path fails, the next open will pick a new path.

EXAMPLES

To display **scsimgr eschgr** plug-in general help and supported commands:

```
scsimgr -h -d eschgr
```

To get **eschgr** driver global statistics:

```
scsimgr get_stat -d eschgr
```

To clear **eschgr** driver global statistics:

```
scsimgr clear_stat -d eschgr
```

To get **eschgr** driver global status information:

```
scsimgr get_info -d eschgr
```

To display information about **eschgr** driver global attributes:

```
scsimgr get_attr -d eschgr
```

AUTHOR

SCSI class driver **eschgr** plug-in for **scsimgr** was developed by HP.

SEE ALSO

scsictl(1M), scsimgr(1M), autochanger(7), intro(7), scsi(7).

NAME

scsimgr_esdisk - SCSI class driver esdisk plug-in for scsimgr

DESCRIPTION

The SCSI class driver **esdisk** plug-in for **scsimgr** implements management and diagnostic operations specific to classes of devices bound to the **esdisk** driver. **esdisk** is the native HP-UX SCSI class driver that handles, by default, all block devices including the following types : direct access, CD/DVD, write-once read-multiple (WORM), and optical memory (OM).

The plug-in handles the following operations for driver **esdisk**:

- Displays and clears driver **esdisk** global statistics and the statistics it maintains on instances of LUNs bound to it, and on related LUN paths.
- Displays status and other information maintained by driver **esdisk** on LUNs bound to it.
- Gets, sets, and saves driver **esdisk** global, per-lun instance attributes or attributes for a set of devices bound to the driver.

Commands

The user can explicitly send the following **scsimgr** commands to driver **esdisk** plug-in by specifying the **-d esdisk** option:

clear_stat Clears statistics.

get_attr Displays information on attributes.

get_info Displays status and other information.

get_stat Displays statistics.

save_attr Saves value of attributes in a persistent store.

set_attr Set current values of attributes.

Note: Refer to *scsimgr*(1M) for the syntax of the above commands.

However, the only instance when it is necessary to explicitly send a command to the plug-in is when performing operations on objects global to driver **esdisk**: global statistics, attributes or status information. In all the other cases, **scsimgr** automatically invokes the plug-in to perform the driver specific part of the operation, when the operation applies to LUNs bound to driver **esdisk** or to their LUN paths.

Attributes

The following table lists driver **esdisk** specific attributes. Also, under the category "Device Set", it lists the attributes the **esdisk** driver can set at scopes; including, device type, vendor identifier, product identifier and product revision. On the concept of attribute and attribute scope, refer to *scsimgr*(1M).

Note: The following conventions are used :

- RO is Read Only.
- RW is Read Write.
- uint32 is unsigned 32 bit integer.
- Range of values for applicable attributes is listed.

Object	Attribute Name	RO/RW	Type	Description
Global	version	RO	string	Version of driver esdisk
LUN	capacity	RO	uint32	Device capacity in number of blocks
	block_size	RO	uint32	Block size in bytes
	path_fail_secs	RW	uint32	Delay in seconds before declaring a LUN path offline after failure of first I/O. Range: 0-600
	load_bal_policy	RW	string	I/O load balancing policy. May be: round_robin , least_cmd_load , cl_round_robin ,

			preferred_path.
	infinite_retries_enable	RW	boolean Enable or disable infinite retry of I/Os. May be: true: enable, false: disable.
	preferred_path	RW	string Hardware path of the lunpath to use preferably for I/O transfer, when I/O load balancing policy is set to preferred_path.
Device Set	transient_secs	RW	uint32 Seconds to wait after a LUN transitioned out of ONLINE state before failing I/Os. Range: 0-600
	format_secs	RW	uint32 Timeout in secs of SCSI command FORMAT. Range: 0-0xFFFFFFFF
	start_unit_secs	RW	uint32 Timeout in secs of SCSI command START UNIT. Range: 0-0xFFFFFFFF
	max_retries	RW	uint32 Maximum number of I/O retries. Range: 1-0xFFFFFFFF
	path_fail_secs	RW	uint32 Timeout in secs before declaring a LUN path offline. Range: 0-600
	esd_secs	RW	uint32 Maximum time in secs for the transmission of an I/O. Range: 0-0xFFFFFFFF
	max_q_depth	RW	uint32 Maximum queue depth. Range: 1-0xFFFFFFFF
	load_bal_policy	RW	string I/O load balancing policy. May be: round_robin , least_cmd_load , cl_round_robin , preferred_path .
	disable_flags	RW	string A set of flags representing SCSI task management and other functions. If a flag is set, the corresponding function is disabled for the set of devices. The following flags are currently defined: WCE : Write Cache Enable, RW16 : 16 bytes READ/WRITE CDB, ABT : SCSI task management function Abort Task Set, CTS : SCSI task management function Clear Task Set, LR : SCSI task management function LUN Reset, WTR : SCSI task management function Warm Target Reset, CTR : SCSI task management function Cold Target Reset, BR : Bus Reset, PR : Persistent Reservation, WERO : Persistent Reservation WERO (Write Exclusive Read-Only), AERO : Persistent Reservation AERO (Access Exclusive Read-Only).
	infinite_retries_enable	RW	boolean Enable or disable infinite retries of I/Os. May be: true: enable, false: disable.

I/O Load Balancing Policy

The I/O load balancing policy attribute, **load_bal_policy**, is a tunable that controls how I/Os are distributed across the paths to a LUN:

- **round_robin**

Paths are selected in a round robin manner. This is more appropriate when all the paths to the device have similar I/O turn-around characteristics.

- **least_cmd_load**

The LUN path with the least number of active I/O requests is selected to execute the next I/O. This policy is appropriate when the paths to the LUN exhibit asymmetric latency characteristics. The load is distributed to optimize the bandwidth on each LUN path.

- **cl_round_robin** (cell aware round robin)

This load balancing policy is applicable to HP cell-based platforms. The LUN paths are selected in a round robin manner within the locality of CPU on which the I/O was initiated, to ensure that memory access latencies are optimized.

- **preferred_path**

The I/O path set in the preferred_path attribute is preferably used for I/O transfer. If this I/O path is not available or if the preferred_path attribute was not set, any other path is selected for I/O transfer. This policy is useful for certain disk arrays, which may exhibit some performance degradation if I/Os are transferred via several I/O paths to a LUN simultaneously.

EXAMPLES

To display **scsimgr esdisk** plug-in general help and supported commands:

```
scsimgr -h -d esdisk
```

To get esdisk driver global statistics

```
scsimgr get_stat -d esdisk
```

To clear esdisk driver global statistics

```
scsimgr clear_stat -d esdisk
```

To get esdisk driver global status information

```
scsimgr get_info -d esdisk
```

To display information about esdisk driver global attributes

```
scsimgr get_attr -d esdisk
```

To set the load balancing policy for disk0 to preferred_path and set the I/O path to be used preferably

```
scsimgr set_attr -D /dev/rdisk/disk0 -a load_bal_policy=preferred_path  
-a preferred_path=0/3/1/0.0x21000020371972eb.0x0
```

To add a settable attribute scope corresponding to all disk devices from HP with product identifier "MSA VOLUME", for allowing modification of some settable attribute at this scope

```
scsimgr ddr_add -N "/escsi/esdisk/0x0/HP /MSA VOLUME "
```

To persistently change the default I/O load balancing policy, I/O timeout, and maximum concurrent I/O for all disk devices from HP with product identifier "MSA VOLUME"

```
scsimgr save_attr -N "/escsi/esdisk/0x0/HP /MSA VOLUME "  
-a load_bal_policy=least_cmd_load -a esd_secs=60  
-a path_fail_secs=60
```

To disable write cache, Persistent Reservation and 16 bytes read/write CDB for all disk devices bound to the **esdisk** driver

```
scsimgr set_attr -N /escsi/esdisk -a disable_flags='WCE PR RW16'
```

AUTHOR

SCSI class driver **esdisk** plug-in for **scsimgr** was developed by Hewlett Packard Company.

SEE ALSO

diskinfo(1M), scsictl(1M), scsimgr(1M), intro(7), scsi(7), scsi_disk(7).

NAME

scsimgr_estape - SCSI class driver estape plug-in for scsimgr

DESCRIPTION

The SCSI class driver **estape** plug-in for **scsimgr** implements management and diagnostic operations specific to classes of devices bound to driver **estape**. **estape** is the native HP-UX SCSI class driver that handles, by default, all tape devices.

The plug-in handles the following operations for driver estape:

- Display and clear driver **estape** global statistics and the statistics it maintains on instances of LUNs bound to it, and on related LUN paths.
- Display status and other information maintained by driver **estape** on LUNs bound to it.
- Get, set and save driver **estape** global and per-lun instance attributes.

Commands

The user can explicitly send the following scsimgr commands to driver **estape** plug-in by specifying the **-d estape** option:

clear_stat Clears statistics.

get_attr Displays information on attributes.

get_info Displays status and other information.

get_stat Displays statistics.

save_attr Saves value of attributes in a persistent store.

set_attr Set current values of attributes.

Note: Refer to *scsimgr(1M)* for syntax of the above commands.

However, the only instances where it is necessary to explicitly send a command to the plug-in is when performing operations on objects global to driver **estape**: global statistics, attributes or status information. In all the other cases, **scsimgr** automatically invokes the plug-in to perform the driver specific part of the operation, when the operation applies to LUNs bound to driver estape or to their LUN paths.

Attributes

The following table lists driver estape specific attributes. For details on the concept of attribute refer to *scsimgr(1M)*.

Note: The following conventions are used:

- RO is Read Only.
- RW is Read Write.
- VBM is Variable Block Mode.
- uint32 is unsigned 32 bits integer.
- uint64 is unsigned 64 bits integer.
- Range of values for applicable attributes is listed.

Object	Attribute Name	RO/RW	Type	Description
Global	version	RO	string	Version of driver estape .
	norewind_ close_disable	RW	uint32	Disables the ability to open a "rewind" device. Default: 0. Values: 0 (disabled), 1 (enabled).
	st_ats_enabled	RW	uint32	Determines whether to reserve the device on open and release on close. See <i>st_ats_enabled(5)</i> . Default: 1.

				Values: 0 (disabled), 1 (enabled).
LUN	default_secs	RW	uint32	Timeout for all commands not referenced below. Default: 30.
	space_secs	RW	uint32	Timeout for the space command. Default: 1200.
	write_secs	RW	uint32	Timeout for the write command. Default: 600.
	read_secs	RW	uint32	Timeout for the read command. Default: 600.
	unload_secs	RW	uint32	Timeout for the unload command. Default: 600.
	rewind_secs	RW	uint32	Timeout for the rewind command. Default: 600.
	erase_secs	RW	uint32	Timeout for the erase command. Default: 18000.
	mt_type	RW	uint32	The type of device a particular LUN is associated with. Default: 0. Values: 0 = unknown, 5 = HPIB 9-track, 6 = DDS1, 7 = All other DDS/DAT, 8 = SCSI 9-track, 9 = QIC, 10 = 8mm, 11 = IBM 3480, STK 9XXX, STK T10000, 12 = Quantum DLT, 13 = Sony AIT, 14 = IBM 3590, 15 = LTO.
	default_blocksize	RW	uint32	Default blocksize. Default: 0. Values: 0 = variable, overridden by a custom DSF.
	default_ir	RW	uint32	Default immediate reporting. Default: 1. Values: 0 (disabled), 1 (enabled).
	close_marks	RW	uint32	Number of filemarks to indicate End of Data. Default: 2.
	num_partitions_supp	RW	uint32	Number of partitions supported Default: 1. Range: 1+
	characteristics	RW	uint64	Driver characteristics bitwise ORed together Default: 0. Values: 1 = Device supports setmarks, 2 = Logpage 31 contains capacity information, 4 = Logpage 38 contains capacity information, 8 = Device supports Reserve/Release.

best_density	RW	uint32	Tape density to write. Default: 0x7F. Values: 0xFFFFFFFF = Best density, 0x00 = Let the device choose the density, 0x7F = Do not modify tape density, other = a valid density code for the Mode Parameter Block Descriptor.
best_compression	RW	uint32	Compression Algorithm to use. Default: 0. Values: 0x00 = Compression Disabled, 0xDEF = default for drive, other = a valid compression value for the Data Compression Mode Page (0x0F).
clean_req_ sns_info	RW	uint32	The Key/Code/Qualifier representing "Cleaning Required" Default: 0xFFFFFFFF.

I/O Load Balancing and Multipathing Policies

The **estape** driver does not support load balancing and has minimal support for multipathing.

When the device is first opened after a system boot, a path is chosen and will remain fixed. If the path fails, the next open will pick a new path.

EXAMPLES

To display **scsimgr estape** plug-in general help and supported commands:

```
scsimgr -h -d estape
```

To get **estape** driver global statistics:

```
scsimgr get_stat -d estape
```

To clear **estape** driver global statistics:

```
scsimgr clear_stat -d estape
```

To get **estape** driver global status information:

```
scsimgr get_info -d estape
```

To display information about **estape** driver global attributes:

```
scsimgr get_attr -d estape
```

AUTHOR

SCSI class driver **estape** plug-in for **scsimgr** was developed by HP.

SEE ALSO

scsictl(1M), scsimgr(1M), st_ats_enabled(5), intro(7), scsi(7), scsi_tape(7).

NAME

sioc_io - SCSI pass-through interface

DESCRIPTION

SCSI devices are controlled by a device-specific driver, when one exists. Device-specific drivers, such as those for SCSI direct access (disk) and sequential access (tape) devices, coordinate device and driver states to accomplish correct logical device behavior. The **sioc_io** pass-through interface enables the use of SCSI devices and commands not normally supported by these device-specific drivers. It is composed of two ioctls: **SIOC_IO_EXT**, and **SIOC_IO**.

SIOC_IO_EXT is the pass-through interface introduced with HP-UX 11i V3 release. It is the recommended interface. It should be issued on persistent device files (see *intro(7)*). It allows to send the SCSI command through any of the available LUN paths or through a specific LUN path.

SIOC_IO is the pass-through interface that existed prior to HP-UX 11i V3. This interface is deprecated with HP-UX 11i V3 release. It is maintained for backward compatibility. It can be used on persistent device files or legacy device files. If issued on a persistent device file, the SCSI command is sent through any of the available LUN paths. If issued on a legacy device file, the SCSI command will be sent through any available LUN paths. However, if multi-pathing is disabled legacy device files (see **leg_mpath_enable** in *scsimgr(1M)*), the SCSI command will be sent only through the LUN path corresponding to the legacy device file.

All reserved fields in the data structure associated to the interface must be zero-filled.

The **SIOC_IO_EXT/SIOC_IO** ioctl allows an arbitrary SCSI command to be sent to a device. All details of the SCSI command protocol are handled automatically.

The data structure for the **SIOC_IO_EXT/SIOC_IO** ioctl is included from **<sys/scsi.h>**:

```

/* SCSI device control ioctls */

#define SIOC_IO_EXT      _IOWR('S', 102, esctl_io_t)
#define SIOC_IO         _IOWR('S', 22, struct sctl_io)

/* Structure for SIOC_IO_EXT ioctl */
typedef struct {
    int                version;
    escsi_sctl_io_flags_t  flags;
    int                max_msecs;
    uint32_t           cdb_length;
    uint32_t           data_length;
    ptr64_t            data;
    union sense_data   sense;
    escsi_hw_path_t    lpt_hwp;
    uint32_t           data_xfer;
    uint32_t           sense_xfer;
    uint32_t           cdb_status;
    uint32_t           sense_status;
    uint8_t            cdb[ESCSI_MAX_CDB_LEN];
    uint32_t           rsvd[32]; /* Reserved for
                                * future use
                                */
} esctl_io_t;

/* Structure for SIOC_IO ioctl */
typedef struct sctl_io {
    unsigned           flags;
    unsigned char      cdb_length;
    unsigned char      cdb[16];
    void               *data;
    unsigned           data_length;
    unsigned           max_msecs;
    unsigned           data_xfer;
    unsigned           cdb_status;
    unsigned char      sense[256];
}

```

```

        unsigned      sense_status;
        unsigned char  sense_xfer;
        unsigned char  reserved[64];
    } sctl_io_t;

```

The following flags can be used to specify the **flags** field value of both **SIOC_IO_EXT** and **SIOC_IO** unless indicated otherwise:

SCTL_READ Data-in phase expected if the *data_length* field is non-zero. The absence of this flag implies that a data-out phase is expected if the *data_length* field is non-zero.

ESCTL_IO_LPT The SCSI command is to be issued on a given LUN path. This flag can only be specified with **SIOC_IO_EXT** ioctl. When specified the hardware path of the LUN path to use is specified in field *lpt_hwp*

The *cdb* field specifies the SCSI command bytes. The number of command bytes is specified by the *cdb_length* field. These command bytes are sent to the target device during the SCSI command phase.

The address of the data area for the data phase of the SCSI command is specified by the *data* field. The **data_length** field specifies the maximum number of data bytes to be transferred. A zero-valued *data_length* indicates that no data phase should occur. Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. The caller is responsible for correctly specifying both the *data_length* field and any *cdb* data length values. The length may not be larger than **SCSI_MAXPHYS** and some implementations further restrict this length.

The *max_msecs* field specifies the maximum time, in milliseconds, that the device should need to complete the command. If this period of time expires without command completion, the system might attempt recovery procedures to regain the device's attention. These recovery procedures might include abort tag, abort, and device and bus reset operations. A zero value in the *max_msec* field indicates that the timeout period is infinite and the system should wait indefinitely for command completion.

When the **SIOC_IO_EXT/SIOC_IO** ioctl call returns, all command processing has been completed. Most **SIOC_IO_EXT/SIOC_IO** ioctl calls will return zero (success). The resulting detailed ioctl data should be used to evaluate "success" or "failure" from the caller's perspective. The **cdb_status** field indicates the results of the **cdb** command. If the *cdb_status* field indicates a **S_CHECK_CONDITION** status, the *sense_status* field indicates the results of the SCSI **REQUEST SENSE** command used to collect the associated sense data. These status fields will contain one of the following values:

SCTL_INVALID_REQUEST The SCSI command request is invalid and thus not attempted.

SCTL_SELECT_TIMEOUT The target device does not answer to selection by the host SCSI interface (the device does not exist or does not respond).

SCTL_INCOMPLETE The device answered selection but the command is not completed (the device took too long or a communication failure occurred).

S_GOOD Device successfully completed the command.

S_CHECK_CONDITION Device indicated sense data is available.

S_CONDITION_MET Device successfully completed the command and the requested (search or pre-fetch) operation is satisfied.

S_BUSY Device indicated it is unable to accept the command because it is busy doing other operations.

S_INTERMEDIATE Device successfully completed this command, which is one in a series of linked commands (not supported, see **WARNINGS**).

S_I_CONDITION_MET Device indicated both **S_INTERMEDIATE** and **S_CONDITION_MET** (not supported, see **WARNINGS**).

S_RESV_CONFLICT Device indicated the command conflicted with an existing reservation.

S_COMMAND_TERMINATED Device indicated the command is terminated early by the host system.

S_QUEUE_FULL Device indicated it is unable to accept the command because its command queue is currently full.

The **data_xfer** field indicates the number of data bytes actually transferred during the data phase of the **cdb** command. This field is valid only when the *cdb_status* field contains one of the following values:

S_GOOD or **S_CHECK_CONDITION**. The *sense_xfer* field indicates the number of valid sense data bytes. This field is valid only when the *cdb_status* field contains the value **S_CHECK_CONDITION** and the *sense_status* field contains the value **S_GOOD**.

Security Restrictions

Use of the **SIOC_IO** ioctl requires the superuser or **DEVOPS** privilege, or device write permissions. See *privileges(5)* for more information about privileged access on systems that support fine-grained privileges.

EXAMPLES

Assume that *fildes* is a valid file descriptor for a persistent device file of a SCSI device, and *leg_fildes* is a valid file descriptor for a legacy device file of a SCSI device, and *lpt_hwp* contains a valid hardware path of a LUN path to the device. The first example attempts a SCSI **INQUIRY** command:

```
#include <sys/scsi.h>

esctl_io_t esctl_io;
#define MAX_LEN 255
unsigned char inquiry_data[MAX_LEN];

memset(&esctl_io, 0, sizeof(esctl_io)); /* clear reserved fields */
esctl_io.flags = SCTL_READ; /* input data expected */
esctl_io.cdb[0] = CMDInquiry;
esctl_io.cdb[1] = 0x00;
esctl_io.cdb[2] = 0x00;
esctl_io.cdb[3] = 0x00;
esctl_io.cdb[4] = MAX_LEN; /* allocation length */
esctl_io.cdb[5] = 0x00;
esctl_io.cdb_length = 6; /* 6 byte command */
esctl_io.data = &inquiry_data[0]; /* data buffer location */
esctl_io.data_length = MAX_LEN; /* maximum transfer length */
esctl_io.max_msecs = 10000; /* allow 10 seconds for cmd */
if (ioctl(fildes, SIOC_IO_EXT, &esctl_io) < 0) {
    /* request is invalid */
} else {
    if (esctl_io.cdb_status == S_GOOD) {
        /* success. display inquiry data */
    } else {
        /* failure. process depending on cdb_status */
    }
}
}
```

The second example attempts a SCSI **INQUIRY** command via a specific LUN path.

```
#include <sys/scsi.h>

esctl_io_t esctl_io;
#define MAX_LEN 255
unsigned char inquiry_data[MAX_LEN];
memset(&esctl_io, 0, sizeof(esctl_io)); /* clear reserved fields */
esctl_io.flags = SCTL_READ | SCTL_IO_LPT; /* input data
    * expected and command
    * to be sent on given
    * LUN path
    */
memcpy(&esctl_io.lpt_hwp, lpt_hwp, sizeof(lpt_hwp)); /* specify
    * the hardware path of
    * LUN path through which
    * command must be sent
    */

esctl_io.cdb[0] = CMDInquiry;
esctl_io.cdb[1] = 0x00;
esctl_io.cdb[2] = 0x00;
esctl_io.cdb[3] = 0x00;
esctl_io.cdb[4] = MAX_LEN; /* allocation length */
```

```

esctl_io.cdb[5] = 0x00;
esctl_io.cdb_length = 6;           /* 6 byte command */
esctl_io.data = &inquiry_data[0]; /* data buffer location */
esctl_io.data_length = MAX_LEN;    /* maximum transfer length */
esctl_io.max_msecs = 10000;       /* allow 10 seconds for cmd */
if (ioctl(fildev, SIOC_IO_EXT, &esctl_io) < 0) {
    /* request is invalid */
} else {
    if ( esctl_io.cdb_status == S_GOOD) {
        /* success. display inquiry data */
    } else {
        /* failure. process depending on cdb_status */
    }
}

```

The following example attempts a SCSI **TEST UNIT READY** command and checks to see if the device is ready, not ready, or in some other state.

```

#include <sys/scsi.h>

struct sctl_io sctl_io;

memset(&sctl_io, 0, sizeof(sctl_io)); /* clear reserved fields */
sctl_io.flags = 0;                   /* no data transfer expected */
sctl_io.cdb[0] = 0x00;               /* can use CMDtest_unit_ready */
sctl_io.cdb[1] = 0x00;
sctl_io.cdb[2] = 0x00;
sctl_io.cdb[3] = 0x00;
sctl_io.cdb[4] = 0x00;
sctl_io.cdb[5] = 0x00;
sctl_io.cdb_length = 6;             /* 6 byte command */
sctl_io.data = NULL;                /* no data buffer is provided */
sctl_io.data_length = 0;            /* do not transfer data */
sctl_io.max_msecs = 10000;         /* allow 10 seconds for cmd */
if (ioctl(leg_fildev, SIOC_IO, &sctl_io) < 0) {
    /* request is invalid */
}
else if (sctl_io.cdb_status == S_GOOD) {
    /* device is ready */
}
else if (sctl_io.cdb_status == S_BUSY ||
        (sctl_io.cdb_status == S_CHECK_CONDITION &&
         sctl_io.sense_status == S_GOOD &&
         sctl_io.sense_xfer > 2 &&
         (sctl_io.sense[2] & 0x0F) == 2)) {
    /* can use sense_data */
    /* device is not ready */
} else {
    /* unknown state */
}

```

WARNINGS

Incorrect use of **sioc_io** operations (even those attempting access to non-existent devices) can cause data loss, system panics, and device damage.

The **SIOC_EXCLUSIVE** ioctl should be used to gain exclusive access to a device prior to attempting **SIOC_IO** commands. If exclusive access is not obtained, **SIOC_IO** commands will be intermixed with device-specific driver commands, which can lead to undesirable results.

Device-specific drivers can reject inappropriate or troublesome **SIOC_IO** commands. However, since not all such operations are known and detected, care should be exercised to avoid disrupting device-specific drivers when using commands that modify internal device states.

Most SCSI commands have a logical unit number (LUN) field. Parallel SCSI implementations on the HP-UX operating system select logical units via the SCSI **IDENTIFY** message. The LUN portion of the **cdb**

should normally be set to zero, even when the LUN being accessed is not zero.

Use of linked commands is not supported.

Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. Both the *data_length* field and any *cdb* data length values must be correctly specified to get correct command results.

Very large (or infinite) timeout values can cause a parallel SCSI bus (potentially the entire system) to hang.

Device and/or bus reset operations can be used to regain a device's attention when a timeout expires.

Resetting a device can cause I/O errors and/or loss of cached data. This can result in loss of data and/or system panics.

Obtaining SCSI **INQUIRY** data by use of the **SIOC_INQUIRY** ioctl instead of by use of the **SIOC_IO** ioctl is generally preferable since SCSI implementations on the HP-UX operating system synchronize access of inquiry data during driver open calls.

Since communication parameters can be affected by device-specific driver capabilities, device-specific driver use might result in communication parameter changes.

FILES

```
/usr/include/sys/scsi.h  
/usr/include/sys/scsi_ctl.h
```

SEE ALSO

ioctl(2), privileges(5), intro(7), scsi(7), scsi_ctl(7).

NAME

slp_syntax - SLP Service Type Syntax

DESCRIPTION

The SLP API expects service type information to be passed while querying for SLP service information and also while registering and deregistering services. The SLP API accepts service type information in URL format also.

The service type string contains the following information.

Name of the service type.

Naming Authority responsible for the service name.

The service type string is of the form:

service : *abstract-type* . *naming-authority* : *concrete-type*

The *abstract-type* is a short descriptive string that describes the type of service.

The *naming-authority* is the name of the organization that named the service. The *naming-authority* is optional, but if it is omitted, then IANA is assumed to be the naming authority and IANA requires service-types to be registered (see RFC 2609).

concrete-type, also optional, is a kind of sub-type of the abstract-type.

For example,

printer is an abstract type (owned by IANA) and **printer:lpr** is a concrete type (owned by IANA).

The official definition of Service Type strings can be found in RFC 2609, "Service Templates and Service Schemes".

Examples of Service Type Strings

weather.nasa:wtp	A (fictitious) weather service type owned by NASA that uses WTP protocol.
weather.nasa:swtp	A (fictitious) weather service type owned by NASA that uses SWTP protocol.
chat.superchat	A chat service type owned by SuperChat.
printer.samba	A samba printer service type.
ftp	An IANA ftp service type.
telnet	An IANA telnet service type.

Comparing Service Types

Since service types are important in determining the URLs that are returned by the **SLPFindSrvs()** call, you should understand how services are compared. Suppose that three services were registered with **SLPReg()** using a *srvtype* of **printer:lpr**, **printer** and **printer.acme**. If a client program calls **SLPFindSrvs()** with a *srvtype* of **service:printer**, the urls for both **printer:lpr** and **printer** are returned (**printer.acme** is not). However, if **SLPFindSrvs()** is called with *srvtype* of **printer:lpr** or **printer.acme**, then the urls for **printer:lpr** or **printer.acme** would be returned. In other words, if a concrete-type is used, only services with the same abstract and concrete-type are returned. If only the abstract type is used, then all services of that abstract type (and naming authority) are returned.

SLP Service URL Syntax

SLP APIs accept service type strings in URL syntax format. URL strings are passed as parameters to **SLPReg()**, **SLPDeReg()**, **SLPFindSrvs()**, and **SLPParseSrvURL()** functions and returned as a result to the **SLPSrvURLCallback()** callback function. SLP defines a special type of URL called a Service URL that MUST be used when calling SLP API functions. The syntax of a service URL is:

SLP Service URL = service:service-type://addrspec

service-type is a service type as explained above. *addrspec* can be any address that fits URL syntax and can be translated as a network location. The **service:** and **://** strings are required.

Service URL Examples

```
service:weather.nasa:wtp://weather.nasa.com:12000
service:weather.nasa:swtp://weather.nasa.com:12001
service:chat.superchat://chat.superchat.com;auth=ldap
```

SLP requires you to use Service URLs. API functions will return **SLP_PARSE_ERROR** if you do not. Service URLs are required because the SLP API designers do not allow the service-type to be passed in as a parameter to the **SLPDeReg()** call. Without the service-type, **SLPDeReg()** does not allow the caller to distinguish between services of varying types that were registered with the same standard URL.

The **SLPFindSrvs()** function expects the search strings to be passed in LDAPv3 Search Filter Syntax.

SEE ALSO

slpd(1M), libslp(3N), slp.reg(4).

NAME

socket - interprocess communications

DESCRIPTION

Sockets are communication endpoints that allow processes to communicate either locally or remotely. They are accessed by means of a set of system calls (see *socket(2)*).

The following **ioctl()** requests are defined in `<sys/ioctl.h>` (see *ioctl(2)*):

FIONBIO If the int with the address *arg* is non-zero, the socket is put into non-blocking mode. Otherwise, the socket is put into blocking mode. Blocking mode is the default. The **FIONBIO** request is equivalent to the **FIOFNIO** request, although using **FIONBIO** is not recommended. See *accept(2)*, *connect(2)*, *recv(2)*, and *send(2)* for an explanation of how non-blocking mode is used.

FIONREAD For SOCK_STREAM sockets, the number of bytes currently readable from this socket is returned in the integer with the address *arg*. For SOCK_DGRAM sockets, the number of bytes currently readable, plus the size of the *sockaddr* structure (defined in `<sys/socket.h>`), is returned in the integer with the address *arg*.

SIOCATMARK For SOCK_STREAM TCP sockets, on return the integer with the address *arg* is non-zero if the inbound TCP stream has been read up to where the out-of-band data byte starts. Otherwise, the inbound TCP stream has not yet been read up to where the out-of-band data byte starts. For sockets other than SOCK_STREAM TCP sockets, on return the integer with the address *arg* is always zero.

SIOCSPGRP This request sets the process group or process ID associated with the socket to be the value of the integer with the address *arg*. A process group or process ID associated with the socket in this manner is signaled when the state of the socket changes: **SIGURG** is delivered upon the receipt of out-of-band data; **SIGIO** is delivered if the socket is asynchronous, as described in **FIOASYNC** below. If the value of the integer with the address *arg* is positive, the signal is sent to the process whose process ID matches the value specified. If the value is negative, the signal is sent to all the processes that have a process group equal to the absolute value of the value specified. If the value is zero, no signal is sent to any process. It is necessary to issue this request with a non-zero integer value to enable the signal delivery mechanism described above. The default for the process group or process ID value is zero.

SIOCGPRP This request returns the process group or process ID associated with the socket in the integer with the address *arg*. See the explanation for **SIOCSPGRP** above for more details on the meaning of the integer value returned.

FIOASYNC If the integer whose address is *arg* is non-zero, this request sets the state of the socket as asynchronous. Otherwise, the socket is put into synchronous mode (the default). Asynchronous mode enables the delivery of the **SIGIO** signal when either of the following conditions is met.

- New data arrives.
- For connection-oriented protocols, whenever additional outgoing buffer space becomes available or the connection is established or broken.

The process group or process ID associated with the socket must be non-zero in order for **SIGIO** signals to be sent. The signal is delivered according to the semantics of **SIOCSPGRP** described above.

The *fcntl(2)* **O_NDELAY** and **O_NONBLOCK** flags (defined in `<fcntl.h>`) are supported by sockets. If the **O_NONBLOCK** flag is set, the socket is put into POSIX-style non-blocking mode. If the **O_NDELAY** flag is set, the socket is put into non-blocking mode. Otherwise, the socket is put into blocking mode. Blocking mode is the default. See *accept(2)*, *connect(2)*, *recv(2)*, and *send(2)* for an explanation of how these forms of non-blocking mode are used.

Since the *fcntl()* **O_NONBLOCK** and **O_NDELAY** flags and *ioctl()* **FIONBIO** requests are supported, the following clarifies on how these features interact. If the **O_NONBLOCK** or **O_NDELAY** flag has been set, *recv()* and *send()* requests behave accordingly, regardless of any **FIONBIO** requests. If neither the **O_NONBLOCK** flag nor the **O_NDELAY** flag has been set, **FIONBIO** requests control the the behavior of *recv()* and *send()*.

DEPENDENCIES

AF_CCITT Only

Only the **FIOSNBIO**, **FIONREAD**, **SIOCGPGRP**, and **SIOCSPGRP** **ioctl()** requests are defined for **af_ccitt** sockets.

AUTHOR

socket was developed by the University of California, Berkeley.

SEE ALSO

fcntl(2), getsockopt(2), ioctl(2), socket(2).

NAME

streamio - STREAMS ioctl commands

SYNOPSIS

```
#include <sys/types.h>
#include <stropts.h>

int ioctl(int fildes, int command, ... /* arg */);
```

DESCRIPTION

STREAMS **ioctl** commands are a subset of the **ioctl()** system calls which perform a variety of control functions on streams.

fildes is an open file descriptor that refers to a stream. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure. The *command* and *arg* are interpreted by the stream head. Certain combinations of these arguments may be passed to a module or driver in the stream.

Since these STREAMS commands are a subset of **ioctl**, they are subject to the errors described there. In addition to those errors, the call will fail with **errno** set to [EINVAL], without processing a control function, if the stream referenced by *fildes* is linked below a multiplexor, or if *command* is not a valid value for a stream.

Also, as described in **ioctl**, STREAMS modules and drivers can detect errors. In this case, the module or driver sends an error message to the stream head containing an error value. This causes subsequent system calls to fail with **errno** set to this value.

The following **ioctl** commands, with error values indicated, are applicable to all STREAMS files:

I_ATMARK Allows the user to see if the current message on the stream head read queue is "marked" by some module downstream. *arg* determines how the checking is done when there are multiple marked messages on the stream head read queue. It may take the following values:

ANYMARK Checks if the message is marked.

LASTMARK Checks if the message is the last one that is marked on the queue.

If both **ANYMARK** and **LASTMARK** are set, **ANYMARK** supersedes **LASTMARK**.

The return value is 1 if the mark condition is satisfied and 0 otherwise.

I_CANPUT Checks if a certain band is writable. *arg* is set to the priority band in question. The return value is 0 if the priority band *arg* is flow controlled, 1 if the band is writable, or -1 on error.

I_CKBAND Check if the message of a given priority band exists on the stream head read queue. This returns 1 if a message of a given priority exists, or -1 on error. *arg* should be an integer containing the value of the priority band in question.

I_FDINSERT Creates a message from user specified buffer(s), adds information about another stream and sends the message downstream. The message contains a control part and an optional data part. The data and control parts to be sent are distinguished by placement in separate buffers, as described below.

arg points to a **strfdinsert** structure which contains the following members:

```
struct strbuf      ctlbuf;
struct strbuf      databuf;
long               flags;
int                fildes;
int                offset;
```

The *len* field in the **ctlbuf strbuf** structure (see *putmsg(2)*) must be set to the size of a pointer plus the number of bytes of control information to be sent with the message. *fildes* in the **strfdinsert** structure specifies the file descriptor of the other stream. *offset*, which must be word-aligned, specifies the number of bytes beyond the beginning of the control buffer where **I_FDINSERT** will store a pointer. This pointer will be the address of the read queue structure of the driver for the

streams corresponding to *fildest* in the **strfdinsert** structure. The *len* field in the **datbuf strbuf** structure must be set to the number of bytes of data information to be sent with the message or zero if no data part is to be sent.

flags specifies the type of message to be created. An ordinary (non-priority) message is created if *flags* is set to 0, a high priority message is created if *flags* is set to **RS_HIPRI**. For normal messages, **I_FDINSERT** will block if the stream write queue is full due to internal flow control conditions. For high priority messages, **I_FDINSERT** does not block on this condition. For normal messages, **I_FDINSERT** does not block when the write queue is full and the **O_NONBLOCK** is set. Instead, it fails and sets **errno** to [EAGAIN].

I_FDINSERT also blocks, unless prevented by the lack of internal resources, waiting for the availability of message blocks, regardless of priority or whether **O_NONBLOCK** has been specified. No partial message is sent.

I_FDINSERT can also fail if an error message was received by the stream head of the stream corresponding to *fildest* in the **strfdinsert** structure. In this case, **errno** will be set to the value in the message.

I_FIND Compares the names of all modules currently present on the stream to the name specified in *arg*. The command returns a value of 1 if the module is present and a value of 0 (zero) if the module is not present.

I_FLUSH This request flushes all input and/or output queues, depending on the value of *arg*. Valid *arg* values are:

FLUSHRW	Flush write and read queues.
FLUSHW	Flush write queues.
FLUSHR	Flush read queues.

If a pipe or FIFO does not have any modules pushed, the read queue of the streams head on either end is flushed depending on the value of *arg*.

If **FLUSHR** is set and *fildest* is a pipe, the read queue for that end of the pipe is flushed and the write queue for the other end is flushed. If *fildest* is a FIFO, both queues are flushed.

If **FLUSHW** is set and *fildest* is a pipe and the other end of the pipe exists, the read queue for the other end of the pipe is flushed and the write queue for this end is flushed. If *fildest* is a FIFO, both queues of the FIFO are flushed.

If **FLUSHRW** is set, all read queues are flushed, that is the read queue for the FIFO and the read queue of both ends of the pipe are flushed.

Correct flushing handling of a pipe or FIFO with modules pushed is achieved via the **pipemod** module. This module should be the first module pushed onto a pipe so that it is at the midpoint of the pipe itself.

I_FLUSHBAND Flushes a particular band of messages. *arg* points to a **bandinfo** structure that has the following members:

unsigned char	bi_pri:
int	bi_flag;

The value of the *bi_flag* field can be **FLUSHR**, **FLUSHW**, or **FLUSHRW** as described for the **I_FLUSH** command.

I_GETBAND Returns the priority band of the first message on the stream head read queue in the integer referenced by *arg*.

I_GETCLTIME Returns the close time delay in the long pointed by *arg*.

I_SETCLTIME Allows the user to set the time that the stream head will delay when a stream is closing, and there is data on the write queues. Before closing each module and driver, the stream head will delay for the specified amount of time to allow the data to drain. If, after the delay, data is still present, data will be flushed. *arg* is a pointer to the

number of milliseconds to delay, rounded up to the nearest valid value on the system. The default is fifteen seconds.

- I_GETSIG** Returns the events for which the calling process has registered to receive a **SIG-POLL** signal. Events are returned as in *arg* bitmask as defined for the **I_SETSIG** command.
- I_GRDOPT** Returns the current read mode setting in an *int* pointed to by the argument *arg*. Read modes are described in *read(2)*.
- I_GWROPT** Returns the current write mode setting, as described in **I_SWROPT**, in the *int* that is pointed to by the argument *arg*.
- I_LINK** Connects two streams, where *fildev* is the file descriptor of the stream connected to the multiplexing driver, and *arg* is the file descriptor of the stream connected to another driver. The stream designated by *arg* gets connected below the multiplexing driver. **I_LINK** requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call returns a multiplexor ID number (an identifier used to disconnect the multiplexor, see **I_UNLINK**) on success, and -1 on failure.
- I_LIST** Allows the user to list all the module names on the stream, up to and including the topmost driver name. If *arg* is **NULL**, the return value is the number of modules, including the driver, that are on the stream pointed to by *fildev*. This allows the user to allocate enough space for the module names. If *arg* is not **NULL**, it should point to a **str_list** structure that has the following members:

```
int          sl_nmods;
struct      str_mlist  *sl_modlist;
```

The **str_mlist** structure has the following member:

```
char        l_name[FMNAMESZ+1];
```

sl_nmods indicates the number of entries the user has allocated in the array. On success, the return value is 0, **sl_modlist** contains the list of module names, and **sl_nmods** indicates the number of entries that have been filled in.

- I_LOOK** Retrieves the name of the module located just below the streams head of the stream pointed to by *fildev*, and places it in a null terminated character string pointed at by *arg*. The buffer pointed to by *arg* should be at least **FNAMESZ+1** bytes long. A **#include <stropts.h>** declaration is required.
- I_NREAD** Counts the number of data bytes in data blocks in the first message on the stream head read queue, and places this value in the location pointed to by *arg*. The return value for the command is the number of messages on the stream head read queue. For example, if zero is returned in *arg*, but the **ioctl** return value is greater than zero, this indicates that a zero-length message is next on the queue.
- I_PEEK** Allows the user process to look (peek) at the contents of the first message on the stream head read queue. This is done without taking the message off the queue. The **I_PEEK ioctl** operates the same way as the **getmsg()** function, except that it does not remove the message. The *arg* parameter points to a **strpeek** structure (in the **<stropts.h>** header file) with the following members:

```
struct strbuf  ctlbuf;
struct strbuf  databuf;
long          flags;
```

The **strbuf** structure pointed to by **ctlbuf** and **databuf** has the following members:

```
int    maxlen;
int    len;
char  *buf
```

The *maxlen* field of the **strbuf** structure must specify the number of bytes of control or data information to be retrieved. The *flags* field can be set to **RS_HIPRI** or 0 (zero). If this field is set to **RS_HIPRI**, the **I_PEEK ioctl** looks for a high priority message on the queue. If the field is set to 0, the **I_PEEK ioctl** looks at the

first message on the queue.

The **I_PEEK** returns a 1 if a message was retrieved, and returns a value of 0 (zero) if no message was found; it does not wait for a message. Upon successful completion, **ctlbuf** specifies control information in the control buffer, **databuf** specifies data information in the data buffer, and **flags** contains **RS_HIPRI** or 0 (zero).

I_PLINK Connects two streams, where *fildev* is the file descriptor of the stream connected to the multiplexing driver, and *arg* is the file descriptor of the stream connected to another driver. The stream designated by *arg* gets connected via a persistent link below the multiplexing driver. **I_PLINK** requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call creates a persistent link which can exist even if the file descriptor associated with the upper stream to the multiplexing driver is closed. This call returns a multiplexor ID number (an identifier that may be used to disconnect the multiplexor, see **I_PUNLINK**) on success and -1 on failure.

The **I_PLINK ioctl** can also fail if it is waiting for the multiplexing driver to acknowledge the link request and an error (**M_ERROR**) message, or hangup (**M_HANGUP**) message is received at the stream head for *fildev*. In addition, an error can be returned in an **M_IOACK** or **M_IONAK** message. When these occur, the **I_PLINK** fails with **errno** set to the value in the message.

I_POP Removes the module just below the stream head of the stream pointed to by *fildev*. To remove a module from a pipe requires that the module was pushed on the side it is being removed from. *arg* should be 0 in an **I_POP** request.

I_PUNLINK Disconnects the two streams specified by *fildev* and *arg* that are connected with a persistent link. *fildev* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexor ID number that was returned by **I_PLINK** when a stream was linked below the multiplexing driver. If *arg* is **MUXID_ALL**, then all streams which are persistent links to *fildev* are disconnected. As in **I_PLINK**, this command requires the multiplexing driver to acknowledge the unlink.

I_PUSH Pushes the module whose name is pointed by *arg* onto the top of the current stream, just below the stream head. If the stream is a pipe, the module will be inserted between the streams heads of both ends of the pipe. It then calls the open routine of the newly-pushed module.

I_RECVFD Retrieves the file descriptor associated with the message sent by an **I_SENDFD ioctl** over a stream pipe. *arg* is a pointer to a data buffer large enough to hold a **strecvfd** data structure containing the following members:

```

int      fd;
uid_t    uid;
gid_t    gid;
char     fill[8]

```

fd is an integer file descriptor. **uid** and **gid** are the user ID and group ID, respectively, of the sending stream.

If **O_NONBLOCK** is clear, **I_RECVFD** will block until a message is present at the stream head. If **O_NONBLOCK** is set, **I_RECVFD** will fail with **errno** set to [EAGAIN] if no message is present at the stream head.

If the message at the stream head is a message sent by a **I_SENDFD**, a new user file descriptor is allocated for the file pointer contained in the message. The new file descriptor is placed in the *fd* field of the **strecvfd** structure. The structure is copied into the user data buffer pointed to by *arg*.

I_SENDFD Requests the stream associated with *fildev* to send a message, containing a file pointer, to the stream head at the other end of a stream pipe. The file pointer corresponds to *arg*, which must be an open file descriptor.

I_SENDFD converts *arg* into the corresponding system file pointer. It allocates a message block and inserts the file pointer in the block. The user ID and group ID associated with the sending process are also inserted. This message is placed directly on the read queue of the stream head at the other end of the stream pipe to which it is connected.

I_SETCLTIME

Lets the user process set the time that the stream head delays when the stream is closing and the write queues contain data. The *arg* parameter contains a pointer to the number of milliseconds to delay, rounded up to the nearest legal value on the system. The default time is 15 seconds.

Before STREAMS modules and drivers are closed, the stream head delays for the specified amount of time. This allows the data on the write queues to drain. If data is still present on the writes queues after the delay, the queues are flushed.

I_SETSIG

Informs the stream head that the user wants the kernel to issue the **SIGPOLL** signal (see *signal(2)*) when a particular event has occurred on the stream associated with *fdes*. **I_SETSIG** supports an asynchronous processing capability in STREAMS. The value of *arg* is a bitmask that specifies the events for which the user should be signaled. It is the bitwise-OR of any combination, except where noted, of the following constants:

- S_BANDURG** When used in conjunction with **S_RDBAND**, **SIGURG** is generated instead of **SIGPOLL** when a priority message reaches the front of the stream head read queue.
- S_ERROR** An **M_ERROR** message has reached the stream head.
- S_HANGUP** An **M_HANGUP** message has reached the stream head.
- S_HIPRI** A high priority message is present on the stream head read queue. This is set even if the message is of zero length.
- S_INPUT** Any message other than an **M_PCPROTO** has arrived on a stream head read queue. This event is maintained for compatibility with prior releases. This is set even if the message is of zero length.
- S_MSG** A STREAMS signal message that contains the **SIGPOLL** signal has reached the front of the stream head read queue.
- S_OUTPUT** The write queue just below the stream head is no longer full. This notifies the user that there is room on the queue for sending (or writing) data downstream.
- S_RDBAND** A priority band message (band > 0) has arrived on a stream head read queue. This is set even if the message is of zero-length.
- S_RDNORM** An ordinary (non-priority) message has arrived on a stream head read queue. This is set even if the message is of zero-length.
- S_WRBAND** A priority band greater than 0 of a queue downstream exists and is writable. This notifies the user that there is room on the queue for sending (or writing) priority data downstream.
- S_WRNORM** This event is the same as **S_OUTPUT**.

A user process may choose to be signaled only of high priority messages by setting *arg* bitmask to the value **S_HIPRI**.

Processes that want to receive **SIGPOLL** signals must explicitly register to receive them using **I_SETSIG**. If several processes register to receive the signal for the same event on the same stream, each process will be signaled when the event occurs.

If the value of *arg* is zero, the calling process will be unregistered and will not receive further **SIGPOLL** signals.

I_SRDOPT

Sets the read mode (see *read(2)*) using the value of the argument *arg*. Valid *arg* values are:

- RNORM** Byte-stream mode (default).
- RMSGD** Message-discard mode.
- RMSGN** Message-nondiscard mode.

Setting both **RMSGD** and **RMSGN** is an error. **RMSGD** and **RMSGN** override **NORM**.

In addition, treatment of control messages by the stream head may be changed by setting the following flags in *arg*:

- RPROTNORM** Fail **read** with EBADMSG if a control message is at the front of the stream head read queue. This is the default behavior.
- RPROTDAT** Deliver the control portion of a message as data when a user issues **read**.
- RPROTDIS** Discard the control portion of a message, delivering any data portion, when a user issues a **read**.

I_STR Constructs an internal STREAMS **ioctl** message from the data pointed to by *arg*, and sends that message downstream.

This mechanism is provided to send user **ioctl** requests to downstream modules and drivers. It allows information to be sent with the **ioctl**, and will return to the user any information sent upstream by the downstream recipient. **I_STR** blocks until the system responds with either a positive or negative acknowledgement message, or until the request "times out" after some period of time. If the request times out, it fails with **errno** set to ETIME.

At most, one **I_STR** can be active on a stream. Further **I_STR** calls will block until the active **I_STR** completes at the stream head. The default timeout intervals for these requests is 15 seconds. The **O_NONBLOCK** (see *open(2)*) flags have no effect on this call.

To send requests downstream, *arg* must point to a **strioc1** structure which contains the following members:

```
int      ic_cmd;
int      ic_timeout;
int      ic_len;
char     *ic_dp;
```

ic_cmd is the internal **ioctl** command intended for the downstream module or driver and **ic_timeout** is the number of seconds (-1 =infinite, 0 = use default, >0 = as specified) an **I_STR** request will wait for acknowledgement before timing out. The default timeout is infinite. **ic_len** is the number of bytes in the data argument and **ic_dp** is a pointer to the data argument. The **ic_len** field has two uses: on input, it contains the length of the data argument passed in, and on return from the command, it contains the number of bytes being returned to the user (the buffer pointed to by **ic_dp** should be large enough to contain the maximum amount of data that any module or driver in the stream can return). The stream head will convert the information pointed to by **strioc1** structure to an internal **ioctl** command message and send it downstream.

S

I_SWROPT Sets the write mode using the value of the argument *arg*. Legal bit settings for *arg* are:

- SNDZERO** Sends a zero-length message downstream when a write of 0 bytes occurs. To not send a zero-length message when a write of 0 bytes occurs, this bit must not be set in *arg*.

I_UNLINK Disconnects the two streams specified by *fildev* and *arg*. *fildev* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexor ID number that was returned by the **I_LINK**. If *arg* is **MUXID_ALL**, then all streams which were linked to *fildev* are disconnected. As in **I_LINK**, this command requires the multiplexing driver to acknowledge the unlink.

RETURN VALUE

Unless specified differently for a command, the return value for a STREAMS **ioctl()** call is 0 (zero) on success and -1 (minus one) on failure.

ERRORS

A STREAMS **ioctl** command fails without performing the function and with **errno** set to [EINVAL] if:

- The stream referred to by *fildev* is linked below a multiplexing driver.

- The *command* parameter is not a valid value for the stream.

In addition, if any of the following conditions occur, the STREAMS **ioctl** commands return the corresponding value:

I_ATMARK

[EINVAL] *arg* has an illegal value.

I_CANPUT

[EINVAL] *arg* has an illegal value.

I_CKBAND

[EINVAL] *arg* has an illegal value.

I_FDINSERT

[EINVAL] The *fildes* parameter in the **strfdinsert** structure is an invalid open file descriptor.

[EINVAL] The size of the pointer plus *offset* exceeds the value of the *len* field for the buffer specified through *ctlptr*.

[EINVAL] *offset* does not specify a properly aligned location in the data buffer.

[EINVAL] *flags* contains an undefined value.

[EFAULT] *arg* points, or **ctrlbuf** or **databuf** is outside the allocated address space.

[EAGAIN] The **ioctl** request failed because a non-priority message was to be created, the **O_NONBLOCK** option was set, and the stream's write queue was full because of internal flow control conditions.

[ENOSR] Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.

[ENXIO] A hangup was received on the stream specified by *fildes* in the **I_FDINSERT ioctl** call or on the stream specified by *fildes* in the **strfdinsert**.

[ERANGE] The value of the *len* field for the buffer specified through **databuf** does not fall within the range for the minimum and maximum sizes of packets for the top-most module on the stream.

[ERANGE] The value of the *len* field for the buffer specified through **databuf** is larger than the maximum allowable size for the data part of a message.

[ERANGE] The value of the *len* field for the buffer specified through **ctlbuf** is larger than the maximum allowable size for the control part of a message.

The **I_FDINSERT ioctl** can also fail if an error (**M_ERROR**) message was received by the stream specified by the *fildes* field in the **strfdinsert** structure. In this case, **errno** is set to the error value in the error message.

I_FIND

[EINVAL] *arg* does not contain a valid module name.

[EFAULT] *arg* points outside the allocated address space.

I_FLUSH

[ENOSR] Could not allocate buffers for flush operation because of a lack of STREAMS memory resources.

[EINVAL] The *arg* parameter is an invalid value.

[ENXIO] A hangup was received on *fildes*.

I_FLUSHBAND

[EINVAL] The *bi_pri* parameter value exceeds the maximum band, or the *bi_flag* parameter is not **FLUSHR**, **FLUSHW**, or **FLUSHRW**.

I_GETBAND

- [ENODATA] No message exists on the stream head read queue.
- I_GETSIG**
- [EINVAL] User process is not registered to receive the **SIGPOLL** signal.
- [EFAULT] *arg* points outside the allocated address space.
- I_GRDOPT**
- [EFAULT] *arg* is pointing outside the allocated address space.
- I_LINK**
- [EAGAIN] Temporarily unable to allocate storage to perform the linking operation.
- [EBADF] The *arg* parameter not a valid open file descriptor.
- [ENXIO] A hangup was received on *fildev*.
- [EINVAL] The stream referred to by *fildev* does not support multiplexing.
- [EINVAL] The file referred to by *arg* is not a stream, or the stream is already linked under a multiplexor.
- [EINVAL] The link operation would cause a "cycle" in the resulting multiplexing configuration. In other words, the driver referred to by the *arg* parameter is linked into this configuration at multiple places
- [ENOSR] Not enough STREAMS memory resources to allocate storage for this command.
- [ETIME] Acknowledgement message not received at stream head before timeout.
- The **I_LINK ioctl** can also fail if an **M_ERROR** or **M_HANGUP** message is received at the stream head for *fildev* before receiving the driver acknowledgement. In addition, an error can be returned in an **M_IOCACK** or **M_IOCNAK** message. When these occur, the **I_LINK ioctl** fails with **errno** set to the value in the message.
- I_LIST**
- [EINVAL] **sl_nmods** is less than 1.
- [EAGAIN] Could not allocate buffers.
- I_LOOK**
- [EINVAL] There are no modules in the stream.
- [EFAULT] *arg* points outside the allocated address space.
- I_NREAD**
- [EFAULT] *arg* is pointing outside the allocated address space.
- I_PEEK**
- [EINVAL] The *flags* parameter is an illegal value.
- [EFAULT] *arg* points, or **ctrlbuf** or **databuf** is, outside the allocated address space.
- [EBADMSG] Message to be looked at is not valid for the **I_PEEK** command.
- I_PLINK**
- [ENXIO] A hangup was received on the stream referred to by the *fildev* parameter.
- [ETIME] A timeout occurred before an acknowledgement message was received at the stream head.
- [EAGAIN] Temporarily unable to allocate storage to perform the linking operation.
- [EBADF] *arg* is not a valid open file descriptor.
- [EINVAL] The stream referred to by *fildev* does not support multiplexing.
- [EINVAL] The file referred to by *arg* is not a stream or is already linked under a multiplexing driver.
- [EINVAL] The link operation would cause a "cycle" in the resulting multiplexing configuration. In other words, the driver referred to by *arg* is linked into the configuration at

multiple places.

I_POP

- [EINVAL] There are not modules in the stream.
- [ENXIO] Error value returned by the module being popped.
- [ENXIO] A hangup was received on *fildev*.

I_PUNLINK

- [ENXIO] A hangup was received on *fildev*.
- [ETIME] A timeout occurred before an acknowledgement message was received at the stream head.
- [EAGAIN] Temporarily unable to allocate storage to perform the linking operation.
- [EINVAL] *arg* is an invalid multiplexor ID number.
- [EINVAL] *fildev* is the file descriptor of a pipe.

An **I_PUNLINK ioctl** can also fail if it is waiting for the multiplexor to acknowledge the unlink request and an error (**M_ERROR**) message, or hangup (**M_HANGUP**) is received at the stream head for *fildev*. In addition, an error can be returned in an **M_IOCACK** or **M_IOCNAK** message. When these occur, the **P_UNLINK ioctl** fails with **errno** set to the value in the message.

I_PUSH

- [EINVAL] An invalid module name was used.
- [EFAULT] *arg* points outside the allocated address space.
- [ENXIO] Error value returned by the module being pushed. The push has failed.
- [ENXIO] A hangup was received on *fildev*.

I_RECVFD

- [EAGAIN] The **O_NONBLOCK** option was set, and a message was not present on the stream head read queue.
- [EFAULT] The *arg* parameter points outside the allocated address space.
- [EBADMSG] The message present on the stream head read queue did not contain a passed file descriptor.
- [EMFILE] Too many open files. No more file descriptors are permitted to be opened.
- [ENXIO] A hangup was received on *fildev*.

I_SENDFD

- [EAGAIN] The sending stream head could not allocate a message block for the file pointer.
- [EAGAIN] The read queue of the receiving stream head was full and could not accept the message.
- [EBADF] The *arg* parameter is not a valid open file descriptor.
- [EINVAL] The *fildev* parameter does not refer to a stream.
- [ENXIO] A hangup was received on *fildev*.

I_SETCLTIME

- [EINVAL] *arg* has an illegal value.

I_SETSIG

- [EINVAL] The user process is not registered to receive the **SIGPOLL** signal.
- [EAGAIN] A data structure to store the signal request could not be allocated.

I_SRDOPT

- [EINVAL] *arg* contains an illegal value.

I_STR

- [EINVAL] The **ic_len** field is less than 0 (zero) bytes or larger than the maximum allowable size of the data part of a message (**ic_dp**).
- [EINVAL] The **ic_timeout** field is less than -1.
- [EFAULT] *arg* points, or the buffer area specified by **ic_dp** or **ic_len** is, outside the allocated address space.
- [ENOSR] Buffers could not be allocated for the **ioctl** request because of a lack of STREAMS memory resources.
- [ENXIO] A hangup was received on the stream referred to by *fildev*.
- [ETIME] The **ioctl** request timed out before an acknowledgement was received.

The **I_STR ioctl** can also fail if the stream head receives a message indicating an error (**M_ERROR**) or a hangup (**M_HANGUP**). In addition, an error can be returned in an **M_IOCACK** or **M_IOCNAK** message. In these cases, the **ioctl** fails with **errno** set to the error value in the message.

I_SWROPT

- [EINVAL] The *arg* parameter is an illegal value.

I_UNLINK

- [ENXIO] A hangup was received on *fildev*.
- [ETIME] A timeout occurred before an acknowledgement message was received at the stream head.
- [EINVAL] *arg* is an invalid multiplexor ID number, or *fildev* is already linked under a multiplexing driver.

An **I_UNLINK ioctl** can also fail if it is waiting for the multiplexor to acknowledge the unlink request and an error (**M_ERROR**) message, or hangup (**M_HANGUP**) is received at the stream head for *fildev*. In addition, an error can be returned in **M_IOCACK** or **M_IOCNAK** message. When this occurs, the **I_UNLINK ioctl** fails with **errno** set to the value in the message.

SEE ALSO

close(2), fcntl(2), getmsg(2), ioctl(2), open(2), poll(2), putmsg(2), read(2), write(2), signal(5).

NAME

strlog - STREAMS log driver

DESCRIPTION

The STREAMS log driver allows user-level processes and STREAMS drivers and modules to perform error logging and event tracing. These tasks are done via a user interface and a kernel interface. Further, the STREAMS log driver delivers error logging and event tracing messages to the Network Tracing and Logging Facility (NetTL) (see *nettl(1M)*, *netfmt(1M)*, and *nettlconf(1M)*).

The interface that this driver presents to user-level processes is a subset of the `ioctl()` system calls and STREAMS message formats. These processes can be error loggers, trace loggers, or other user processes, that generate error or event messages. The user interface collects log messages from the log driver, and also generates log messages from user processes.

The driver also accepts log messages from STREAMS drivers and modules in the kernel via its function call interface. The kernel interface enters requests or calls from STREAMS drivers and modules into log messages.

The log messages accepted by the log driver are also delivered to NetTL. NetTL can be used to control which types of messages to log, and to format and filter the logged messages.

Kernel Interface

STREAMS drivers and modules generate log messages by calls to the `strlog` function.

```
#include <sys/strlog.h>
```

```
int strlog (mid, sid, level, flags, fmt [, value ]...);
short mid;
short sid;
char level;
ushort flags;
char *fmt;
int value;
```

mid specifies the STREAMS module ID number for the driver or module submitting the log message.

sid specifies the sub-ID number of a minor device associated with the STREAMS module or driver identified by *mid*.

level specifies a level for screening lower-level event messages from a tracer.

flags contains several flags that can be set in various combinations. The flags are as follows:

SL_ERROR The message is for the error logger.

SL_TRACE The message is for the tracer.

SL_CONSOLE The message will be printed to the console.

SL_FATAL Provides a notification of a fatal error.

SL_NOTIFY Makes a request to mail a copy of a message to the system administrator.

The following are additional flags. These flags are not used by `strerr` or `strace`. However, they are used to map STREAMS messages to NetTL messages as described below in *STREAMS-NetTL Link* section.

SL_WARN The message is a warning.

SL_NOTE The message is a note.

fmt is a `printf` style format string. This accepts the `%x`, `%l`, `%o`, `%u`, `%d`, `%c`, and `%s` conversion specifications.

values are numeric or character arguments for the format string. There is no maximum number of arguments that can be specified.

User Interface

User processes access the log driver with an `open()` call to `/dev/strlog`. Each open to the device will obtain a separate stream. After a process opens `/dev/strlog`, it indicates whether it is an error logger

or trace logger. It does this by issuing an `I_STR ioctl()` system call with the appropriate value in the `ic_cmd` field of the `striocctl` structure, and the appropriate data and control information in a `trace_ids` structure:

```
struct trace_ids {
    short    ti_mid;
    short    ti_sid;
    char     ti_level;
    short    ti_flags;
};
```

The values for `ic_cmd` are:

- I_ERRLOG** Indicates an error logger. No `trace_ids` data is needed.
- I_TRCLOG** Indicates a trace logger. A data buffer consisting of an array of one or more `trace_ids` structures must be included.

If any of the fields of the `trace_ids` structure contain a value of -1, `/dev/strlog` will accept whatever value it receives in that field. Otherwise, `strlog` only accepts messages only if the values of `mid` and `sid` are the same as their counterparts in the `trace_ids` structure, and if the message's level is equal to or less than the `level` value in the `trace_ids` structure.

Once the logger process has sent the `I_STR ioctl()` call, the STREAMS log driver begins to send log messages matching the restrictions to the logger process. The logger process obtains the log messages via the `getmsg()` system call. The control part of the messages passed in this call includes a `log_ctl` structure:

```
struct log_ctl {
    short    mid;
    short    sid;
    char     level;
    short    flags;
    long     ltime;
    long     ttime;
    int      seq_no;
};
```

The `log_ctl` structure indicates the `mid`, `sid`, and `level` time in ticks since the boot time that the message was submitted, the corresponding time in seconds since January 1, 1970, and a sequence number. The time in seconds since January 1, 1970 is provided so that the date and time of the message can be easily computed. The time in ticks since boot time is provided so that the relative timing of log messages can be determined.

A user process, other than an error or trace logger, can send a log message to `strlog`. The driver will accept only the `flags` and `level` fields of the `log_ctl` structure in the control part of the message, and a properly formatted data part of the message. The data part of the message is properly formatted if it contains a null-terminated format string, followed by up to three arguments packed one word each after the end of the string.

A different series of sequence numbers is provided for error and trace logging streams. These sequence numbers are intended to help track the delivery of the messages. A gap in a sequence of numbers indicates that the logger process did not successfully deliver them. This can happen if the logger process stops sending messages for one reason or another (see `strace(1M)` and `strerr(1M)` command reference pages for more information). The data part of messages contains text of the format string (null terminated), followed by up to three arguments.

STREAMS-NetTL Link

Both STREAMS error logging and event tracing messages are mapped to NetTL logging messages, and are delivered to NetTL. NetTL classifies messages into four log classes: DISASTER, ERROR, WARNING, and INFORMATIVE. The NetTL log class is determined by the `flags` according to the following rule:

```
If (flags & SL_ERROR)           NetTL log class
then
    if (flags & SL_FATAL) ==> DISASTER
    if (flags & SL_WARN)  ==> WARNING
    if (flags & SL_NOTE)  ==> INFORMATIVE
    otherwise             ==> ERROR
```

```

else
    all messages          =====>  INFORMATIVE

```

As a default, only DISASTER and ERROR messages are logged. This setting can be altered by the **nettl** command or the **nettlconf** command (see *nettl(1M)* and *nettlconf(1M)*).

The STREAMS subsystem ID used by NetTL is **STREAMS**.

The messages logged by NetTL facility can be formatted to a readable form by the **netfmt** command (see *netfmt(1M)*). The **netfmt** accepts a filter configuration file, which can be used to filter on STREAMS module ID and sub-ID. The filter configuration file syntax for STREAMS is the following:

```
STREAMS module_id sub_id
```

module_id and *sub_id* can be a decimal number or * as a wild card.

RETURN VALUE

Unless specified otherwise, upon successful completion, the **strlog ioctl()** commands return a value of 0 (zero). Otherwise, a value of -1 is returned.

ERRORS

If any of the following conditions occurs, **strlog** driver's **ioctl()** command sets **errno** to the corresponding value:

[ENXIO] The **I_TRCLOG ioctl()** call did not contain any **trace_ids** structures.

[ENXIO] The **I_STR ioctl()** call could not be recognized.

The driver does not return any errors for incorrectly formatted messages that user processes send.

EXAMPLES

The following examples illustrate some basic uses for the **strlog** interface.

This code example segment shows how a STREAMS module causes a message to be printed to the console:

```

strlog(TMUX,minor(mydev),0,SL_CONSOLE|SL_FATAL,
      "TMUX driver (minor:%d) suffers resource shortage.",
      minor(mydev));

```

This code example shows how a user process registers itself with the STREAMS log driver using the **ioctl()** command, **I_ERRLOG**.

```

struct strioctl iocerr;
int logfd;

if ((logfd = open("/dev/strlog", O_RDWR)) == -1) {
    printf("Cannot open /dev/strlog\n");
    exit(1);
}

iocerr.ic_cmd = I_ERRLOG;
iocerr.ic_timeout = 0;
iocerr.ic_len = 0;
iocerr.ic_dp = NULL;
ioctl(logfd, I_STR, &iocerr);

```

This code example shows a user-level process sending a message to the **strlog** driver.

```

struct strbuf control, data;
struct log_ctl log;
char *warning = "Fatal error for user level process";
int logfd;

if ((logfd = open("/dev/strlog", O_RDWR)) == -1) {
    printf("Cannot open /dev/strlog\n");
    exit(1);
}

control.len = control.maxlen = sizeof(log);

```

```

control.buf = (char *)&lc;

data.len = data.maxlen = strlen(warning);
data.buf = warning;

lc.level = 2;
lc.flags = SL_FATAL|SL_CONSOLE;

putmsg(logfd, &control, &data, 0);

```

The following examples illustrate how to use the NetTL facility for the STREAMS. See *nettl(1M)*, *netfmt(1M)*, *nettlconf(1M)* for the general NetTL usage. The STREAMS subsystem ID used by NetTL is **STREAMS**.

The **netfmt** accepts a filter configuration file as a command argument. The following filter configuration file example is used to format the messages whose module ID is 1 and sub-ID is 100:

```
STREAMS 1 100
```

This filter configuration file example can be used to display all the messages whose module ID is 2 and all the messages whose sub-ID is 101:

```
STREAMS 2 *
STREAMS * 101
```

FILES

/dev/strlog specifies the clone interface.

<sys/strlog.h> specifies the header file for streams logging.

<stropts.h> specifies the header file for STREAMS options and **ioctl()** commands.

SEE ALSO

netfmt(1M), *nettl(1M)*, *nettlconf(1M)*, *strace(1M)*, *strerr(1M)*, *getmsg(2)*, *ioctl(2)*, *open(2)*, *putmsg(2)*, *write(2)*, *clone(7)*, *streamio(7)*.

NAME

sttyv6: stty - terminal interface for Version 6/PWB compatibility

DESCRIPTION

These routines attempt to map the UNIX Time-Sharing System, Sixth Edition (Version 6), and PWB **stty()** and **gTTY()** calls into the current **ioctl**s that perform the same functions. The mapping cannot be perfect. The way the features are translated is described below. The reader should be familiar with **termio** before studying this entry.

The following data structure is defined in the include file **<sgtty.h>**:

```

struct sgttyb {
    char  sg_ispeed; /* input speed */
    char  sg_ospeed; /* output speed */
    char  sg_erase;  /* erase character */
    char  sg_kill;   /* kill character */
    int   sg_flags;  /* mode flags */
}

```

The flags, as defined in **sgtty.h**, are:

HUPCL	01
XTABS	02
LCASE	04
ECHO	010
CRMOD	020
RAW	040
ODDP	0100
EVENP	0200
ANYP	0300
NLDELAY	001400
TBDELAY	002000
CRDELAY	030000
VTDELAY	040000
BSDELAY	0100000
CR0	0
CR1	010000
CR2	020000
CR3	030000
NL0	0
NL1	000400
NL2	001000
NL3	001400
TAB0	0
TAB1	002000
NOAL	004000
FF0	0
FF1	040000
BS0	0
BS1	0100000

When the **stty** command (*ioctl* **TIOCSETP**) is executed, the flags in the old **sgttyb** structure are mapped into their new equivalents in the **termio** structure. Then the **TCSETA** command is executed.

The following table shows the mapping between the old **sgttyb** flags and the current **termio** flags. Note that flags contained in the **termio** structure that are not mentioned below are cleared.

HUPCL (if set)	Sets the termio HUPCL flag.
HUPCL (if clear)	Clears the termio HUPCL flag.
XTABS (if set)	Sets the termio TAB3 flag.
XTABS (if clear)	Clears the termio TAB3 flag.
TBDELAY (if set)	Sets the termio TAB1 flag.

TBDELAY (if clear)	Clears the termio TAB1 flag.
LCASE (if set)	Sets the termio IUCLC, OLCUC, and XCASE flags.
LCASE (if clear)	Clears the termio IUCLC, OLCUC, and XCASE flags.
ECHO (if set)	Sets the termio ECHO flag.
ECHO (if clear)	Clears the termio ECHO flag.
NOAL (if set)	Sets the termio ECHOK flag.
NOAL (if clear)	Clears the termio ECHOK flag.
CRMOD (if set)	Sets the termio ICRNL and ONLCR flags; also, if CR1 is set, the termio CR1 flag is set, and if CR2 is set, the termio ONOCR and CR2 flags are set.
CRMOD (if clear)	sets the termio ONLRET flag; also, if NL1 is set, the termio CR1 flag is set, and if NL2 is set, the termio CR2 flag is set.
RAW (if set)	Sets the termio CS8 flag, and clears the termio ICRNL and IUCLC flags; also, default values of 6 characters and 0.1 seconds are assigned to MIN and TIME, respectively.
RAW (if clear)	Sets the termio BRKINT, IGNPAR, ISTRIP, IXON, IXANY, OPOST, CS7, PARENB, ICANON, and ISIG flags; also, the default values control-D and null are assigned to the control characters EOF and EOL, respectively.
ODDP (if set)	If EVENP is also set, clears the termio INPCK flag; otherwise, sets the termio PARODD flag.
VTDELAY (if set)	Sets the termio FFDLY flag.
VTDELAY (if clear)	Clears the termio FFDLY flag.
BSDDELAY (if set)	Sets the termio BSDLY flag.
BSDDELAY (if clear)	Clears the termio BSDLY flag.

In addition, the **termio** CREAD bit is set, and, if the baud rate is 110, the CSTOPB bit is set.

When using **TIOCSETP**, the *ispeed* entry in the **sgttyb** structure is mapped into the appropriate speed in the **termio** CBAUD field. The *erase* and *kill* **sgttyb** entries are mapped into the **termio** erase and kill characters.

When the **gtty** (*ioctl* **TIOCGETP**) command is executed, the **termio** **TCGETA** command is first executed. The resulting **termio** structure is then mapped into the **sgttyb** structure, which is then returned to the user.

The following table shows how the **termio** flags are mapped into the old **sgttyb** structure. Note that all flags contained in the **sgttyb** structure that are not mentioned below are cleared.

HUPCL (if set)	Sets the sgttyb HUPCL flag.
HUPCL (if clear)	Clears the sgttyb HUPCL flag.
ICANON (if set)	Sets the sgttyb RAW flag.
ICANON (if clear)	Clears the sgttyb RAW flag.
XCASE (if set)	Sets the sgttyb LCASE flag.
XCASE (if clear)	Clears the sgttyb LCASE flag.
ECHO (if set)	Sets the sgttyb ECHO flag.
ECHO (if clear)	Clears the sgttyb ECHO flag.
ECHOK (if set)	Sets the sgttyb NOAL flag.
ECHOK (if clear)	Clears the sgttyb NOAL flag.
PARODD (if set)	Sets the sgttyb ODDP flag.
PARODD (if clear)	Clears the sgttyb ODDP flag.
INPCK (if set)	Sets the sgttyb EVENP flag.

PARODD, INPCK (if both clear)	Sets the sgttyb ODDP and EVENP flags.
ONLCR (if set)	Sets the sgttyb CRMOD flag; also, if CR1 is set, the sgttyb CR1 flag is set, and if CR2 is set, the sgttyb CR2 flag is set.
ONLCR (if clear)	If CR1 is set, the sgttyb NL1 flag is set, and if CR2 is set, the sgttyb NL2 flag is set.
TAB3 (if set)	Sets the sgttyb XTABS flag.
TAB3 (if clear)	Clears the sgttyb XTABS flag.
TAB1 (if set)	Sets the sgttyb TBDELAY flag.
TAB1 (if clear)	Clears the sgttyb TBDELAY flag.
FFDLY (if set)	Sets the sgttyb VTDELAY flag.
FFDLY (if clear)	Clears the sgttyb VTDELAY flag.
BSDLY (if set)	Sets the sgttyb BSDELAY flag.
BSDLY (if clear)	Clears the sgttyb BSDELAY flag.

When using **TIOCGETP**, the **termio** CBAUD field is mapped into the *ispeed* and *ospeed* entries of the **sgttyb** structure. Also, the **termio** erase and kill characters are mapped into the *erase* and *kill* **sgttyb** entries.

Note that, since there is not a one-to-one mapping between the **sgttyb** and **termio** structures, unexpected results may occur when using the older **TIOCSETP** and **TIOCGETP** calls. Thus, the **TIOCSETP** and **TIOCGETP** calls should be replaced in all future code by the current equivalents, **TCSETA** and **TCGETA**, respectively.

WARNINGS

These facilities are included to aid in conversion of old programs, and should not be used in new code. Use the interface described in **termio**. Note that these conversions do *not* work for programs ported from UNIX Time-Sharing System, Seventh Edition (Version 7), because some V7 flags are defined differently.

SEE ALSO

stty(2), termio(7).

NAME

TCP - Internet Transmission Control Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>

s = socket(AF_INET, SOCK_STREAM, 0);
s = socket(AF_INET6, SOCK_STREAM, 0);
```

DESCRIPTION

The TCP protocol provides reliable, flow-controlled, two-way transmission of data. It is a byte-stream protocol used to support the **SOCK_STREAM** socket type. TCP constructs virtual circuits between peer entities. A virtual circuit consists of remote Internet addresses, remote ports, local Internet addresses and local ports. IP uses the Internet addresses to direct messages between hosts, and the port numbers to identify a TCP entity at a particular host.

Sockets using TCP are either **active** or **passive**. **connect()** creates active sockets, which initiate connections to passive sockets (see *connect(2)*). To create a passive socket, use the **listen()** system call after binding the socket with the **bind()** system call (see *listen(2)* and *bind(2)*). Only passive sockets can use the **accept()** call to accept incoming connections (see *accept(2)*).

Passive sockets can *underspecify* their location to match incoming connection requests from multiple networks. This technique, called **wildcard addressing**, allows a single server to provide service to clients on multiple networks. To create a socket that listens on all networks, the Internet address **INADDR_ANY** must be bound for AF_INET family and **in6addr_any** for AF_INET6 family. The TCP port can still be specified even if wildcard addressing is being used. If the port is specified as zero, the system assigns a port.

Once **accept()** has a rendezvous with a connect request, a virtual circuit is established between peer entities. **bind()** supplies the local port and local Internet address and **accept()** gathers the remote port and remote Internet address from the peer requesting the connection.

Options

The system supports the following socket options: **TCP_MAXSEG**, **TCP_NODELAY**, **TCP_ABORT_THRESHOLD**, **TCP_CONN_ABORT_THRESHOLD**, **TCP_KEEPCNT**, **TCP_KEEPIDLE**, **TCP_KEEPINTVL**, **TCP_TSOPTENA**, and **TCP_SACKENA** (defined in the include file `<netinet/tcp.h>`). The **TCP_MAXSEG** option can only be used with **getsockopt()**, while **TCP_NODELAY**, **TCP_ABORT_THRESHOLD**, **TCP_CONN_ABORT_THRESHOLD**, **TCP_KEEPCNT**, **TCP_KEEPIDLE**, **TCP_KEEPINTVL**, **TCP_TSOPTENA**, and **TCP_SACKENA** can be set with **setsockopt()** and tested with **getsockopt()** (see *getsockopt(2)*). These options require *level* to be set to **IPPROTO_TCP** in the **getsockopt/setsockopt** call.

TCP_MAXSEG (non-boolean option) lets an application to receive the current segment size of the TCP SOCK_STREAM socket. The current segment size will be returned in *optval*.

TCP_NODELAY

(boolean option) causes small amounts of output to be sent immediately.

TCP_ABORT_THRESHOLD

(non-boolean option) sets the second threshold timer for the connections that are in ESTABLISHED state. The option value is the threshold time in milliseconds.

When it must retransmit packets because a timer has expired, TCP first compares the total time it has waited against the two thresholds, as described in RFC 1122, 4.2.3.5. If it has waited longer than the second threshold (R2), TCP terminates the connection. The default value for this option is the current value of the ndd tunable parameter **tcp_ip_abort_interval**. Refer to *ndd(1M)* online help for details on the **tcp_ip_abort_interval** default value.

TCP_CONN_ABORT_THRESHOLD

(non-boolean option) sets the second threshold timer during connection establishment. The option value is the threshold time in milliseconds.

This option is the same as **TCP_ABORT_THRESHOLD**, except that this value is used during connection establishment. When it must retransmit the SYN packet because a

timer has expired, TCP first compares the total time it has waited against the two thresholds. If it has waited longer than the second threshold, TCP terminates the connection. The default value for this option is the current value of the `ndd` tunable `tcp_ip_abort_cinterval`. See `ndd(1M)` online help for details on the `tcp_ip_abort_cinterval` default value.

TCP_KEEPCNT

(non-boolean option) When the `SO_KEEPALIVE` option is enabled, TCP probes a connection that has been idle for some amount of time. If the remote system does not respond to a keepalive probe, TCP retransmits the probe a certain number of times before a connection is considered to be broken. The `TCP_KEEPCNT` option can be used to affect this value for a given socket, and specifies the maximum number of keepalive probes to be sent. This option takes an `int` value, with a range of 1 to 32767.

TCP_KEEPIDLE

(non-boolean option) When the `SO_KEEPALIVE` option is enabled, TCP probes a connection that has been idle for some amount of time. The default value for this idle period is 2 hours. The `TCP_KEEPIDLE` option can be used to affect this value for a given socket, and specifies the number of seconds of idle time between keepalive probes. This option takes an `int` value, with a range of 1 to 32767.

TCP_KEEPIKIT

(non-boolean option) If a TCP connection cannot be established within some amount of time, TCP will time out the connect attempt. The default value for this initial connection establishment timeout is 75 seconds. The `TCP_KEEPIKIT` option can be used to affect this initial timeout period for a given socket, and specifies the number of seconds to wait before the connect attempt is timed out. For passive connections, the `TCP_KEEPIKIT` option value is inherited from the listening socket. This option takes an `int` value, with a range of 1 to 32767.

TCP_KEEPIKITVL

(non-boolean option) When the `SO_KEEPALIVE` option is enabled, TCP probes a connection that has been idle for some amount of time. If the remote system does not respond to a keepalive probe, TCP retransmits the probe after some amount of time. The default value for this retransmit interval is 75 seconds. The `TCP_KEEPIKITVL` option can be used to affect this value for a given socket, and specifies the number of seconds to wait before retransmitting a keepalive probe. This option takes an `int` value, with a range of 1 to 32767.

TCP_TSOPTENA

(boolean option) When this option is enabled, the sender places a timestamp in each data segment. The receiver, if configured to accept them, sends these timestamps back in ACK segments. This provides the sender with a mechanism with which to measure round-trip time. TCP provides a Boolean option, `TCP_TSOPTENA` (from the `<netinet/tcp.h>` header file) to enable or disable this option. This option takes an `int` value. When this option is enabled, the `TCP_PAWS` option is also enabled.

TCP_PAWS

(boolean option) When the PAWS (Protect Against Wrapped Sequence numbers) option is enabled, the receiver rejects any old duplicate segments that are received. This option is used on synchronized TCP connections only. TCP provides a Boolean option, `TCP_PAWS` (from the `<netinet/tcp.h>` header file) to enable or disable this option. This option takes an `int` value. This option automatically turns the `TCP_TSOPTENA` option on.

TCP_SACKENA

(boolean option) When the Selective Acknowledgment (SACK) option is enabled, the data receiver can inform the sender about all segments that have arrived successfully. In this way, the sender need retransmit only those segments that have actually been lost. This option is useful in cases where multiple segments are dropped. TCP provides a Boolean option, `TCP_SACKENA` (from the `<netinet/tcp.h>` header file) to enable or disable this option. This option takes an `int` value.

If `TCP_NODELAY` is set, the system sends small amounts of output immediately rather than gathering them into a single packet after an acknowledgement is received. If `TCP_NODELAY` is not set, the system sends data when it is presented, if there is no outstanding unacknowledged data. If there is outstanding

unacknowledged data, the system gathers small amounts of data to be sent in a single packet once an acknowledgement is received. For clients such as window managers that send a stream of mouse events which receive no replies, this packetization may cause significant delays. The **TCP_NODELAY** option can be used to avoid this situation. Note, however, that setting the **TCP_NODELAY** option may result in a large number of small packets being sent over the network.

By default, **TCP_NODELAY** is not set when a socket is created.

The option level to use for accessing the TCP option with the **setsockopt()** or **getsockopt()** calls is the protocol number for TCP which is available from **getprotobyname()** (see *getprotoent(3N)*).

If the **SO_KEEPAIVE** socket option is enabled on an established TCP connection and the connection has been idle for two hours, TCP sends a packet to the remote socket, expecting the remote TCP to acknowledge that the connection is still active. If the remote TCP does not respond in a timely manner, TCP continues to send keepalive packets according to its normal retransmission algorithm. If the remote TCP does not respond within a particular time limit, TCP drops the connection. The next socket system call (for example, **recv()**) returns an error, and **errno** is set to [ETIMEDOUT]. See *getsockopt(2)* for details on enabling **SO_KEEPAIVE**.

The default send and receives buffer size is 32768 bytes (see *WARNINGS* below). The send and receive buffer sizes for TCP stream sockets can be altered by using the **SO_SNDBUF** and **SO_RCVBUF** options of the **setsockopt()** system call or the **XTI_SNDBUF** and **XTI_RCVBUF** options of the **t_optmgmt()** system call. Refer to *getsockopt(2)* or *t_optmgmt(3)* for details.

The maximum transmit buffer size for a TCP stream socket is 2147483647 bytes. The maximum receive buffer size for a TCP stream socket is 1073725440 bytes. These maximum values can be lowered using the ndd variables **tcp_xmit_hiwater_max** and **tcp_recv_hiwater_max**.

ERRORS

One of the following errors may be returned in **errno** if a socket operation fails. For a more detailed list of errors, see the man pages for specific system calls.

[EISCONN]	The socket is already connected.
[ENOBUFS]	No buffer space is available for an internal data structure.
[ETIMEDOUT]	Connection dropped due to excessive retransmissions.
[ECONNRESET]	The connection was forcibly closed by the peer socket.
[ECONNREFUSED]	Remote peer actively refuses connection establishment (usually because no process is listening to the port).
[EADDRINUSE]	The specified address is already in use.
[EADDRNOTAVAIL]	The specified address is not available on this machine.

WARNINGS

The default socket buffer size might increase without notice in a future release or patch. Therefore, if an application calls **setsockopt()** with **SO_RCVBUF**, it should do so before calling **listen()**, or it should first call **getsockopt()** with **SO_RCVBUF** and ensure that the intended new receive buffer size is not less than the current buffer size. These programming conventions are consistent with TCP protocol restrictions against reducing the TCP receive window after a connection has been established.

AUTHOR

The socket interfaces to TCP were developed by the University of California, Berkeley.

SEE ALSO

ndd(1M), *getsockopt(2)*, *recv(2)*, *send(2)*, *socket(2)*, *t_open(3)*, *t_optmgmt(3)*, *socket(7)*, *inet(7F)*.

RFC 793	Transmission Control Protocol
RFC 1122	Requirements for Internet hosts
RFC 1323	TCP Extensions for High Performance

RFC 1878	Variable Length Subnet Table for IPv4
RFC 2018	TCP Selective Acknowledgement Options
RFC 2414	Increasing TCP's Initial Window
RFC 2582	NewReno Modifications to TCP's Fast Recovery Algorithm


t

NAME

tels, telm - STREAMS Telnet slave (pseudo-terminal) driver, STREAMS Telnet master driver (used by telnetd only), respectively

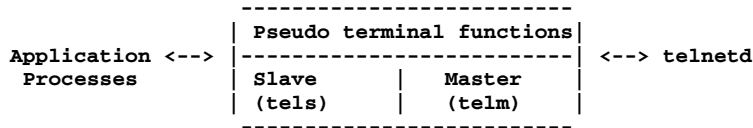
SYNOPSIS

```
#include <sys/termios.h>
#include <sys/strtio.h>

int open("/dev/pts/tN", O_RDWR);
```

DESCRIPTION

A Telnet pseudo-terminal consists of a tightly-coupled pair of character devices, called the master device and slave device. The master and slave device drivers work together to provide a Telnet connection on the server side where the master provides a connection to **telnetd** and the slave provides a terminal device special file access for the Telnet application processes, as depicted below:



The slave driver, **tels** with **ptem** (STREAMS pty emulation module) and **ldterm** (STREAMS line discipline module) pushed on top (not shown for simplicity), provides a terminal interface as described in *termio(7)*. Whereas devices that provide the terminal interface described in *termio(7)* have a hardware device behind them; in contrast, the slave device has **telnetd** manipulating it through the master side of the Telnet pseudo terminal.

There are no nodes in the file system for each individual master device. Rather, the master driver is set up as a STREAMS *clone(7)* driver with its major device number set to the major for the clone driver and its minor device number set to the major for the **telm** driver. The master driver is opened by telnetd using the *open(2)* system call with **/dev/telnetm** as the device file parameter. The clone open finds the next available minor number for the master device. The master device is available only if it and its corresponding slave device are not already opened.

In order to use the STREAMS Telnet subsystem, a node for the master driver **/dev/telnetm** and *N* number of Telnet slave devices must be installed.

The number of slave devices is set by a kernel tunable parameter called **nstrtel**. This can be modified using SAM; its default and minimum value is 60. The value of **nstrtel** is the upper limit of the number of telnet sessions that can be opened.

Multiple opens are allowed on the Telnet slave device.

The master and slave drivers pass all STREAMS messages to their adjacent drivers. When the connection is closed from the Telnet client side, an **M_HANGUP** message is sent to the corresponding slave device which will render that slave device unusable. The process on the slave side gets the errno **ENXIO** when attempting a *write(2)* system call to the slave device file but it will be able to read any data remaining in the slave stream. Finally, when all the data has been read, the *read(2)* system call will return 0, indicating that the slave can no longer be used.

AUTHOR

tels() and **telm()** were developed by HP.

FILES

```
/dev/telnetm   Streams Telnet master clone device
/dev/pts/tN   Streams slave devices where N is the minor number of the slave device and  $0 < N < nstrtel$ .
```

SEE ALSO

insf(1M), open(2), ioctl(2), streamio(7), ldterm(7), telnetd(1M), ptem(7).

NAME

termio, termios - general terminal interface

DESCRIPTION

All HP-UX asynchronous communications ports use the same general interface, regardless of what hardware is involved. Network connections such as **rlogin** (see *rlogin(1)*) use the pseudo-terminal interface (see *pty(7)*).

This discussion centers around the common features of this interface.

Opening a Terminal File

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users' programs seldom open these files; they are opened by special programs such as **getty** (see *getty(1M)*) and become a user's standard input, standard output, and standard error files.

If both the **O_NDELAY** and **O_NONBLOCK** flags (see *open(2)*) are clear, an *open* blocks until the type of modem connection requested (see *modem(7)*) is completed. If either the **O_NDELAY** or **O_NONBLOCK** flag is set, an *open* succeeds and return immediately without waiting for the requested modem connection to complete. The **CLOCAL** flag (see *Control Modes*) can also affect *open(2)*.

Process Groups

A terminal can have a foreground process group associated with it. This foreground process group plays a special role in handling signal-generating input characters.

Command interpreter processes can allocate the terminal to different *jobs* (process groups) by placing related processes in a single process group and associating this process group with the terminal. A terminal's foreground process group can be set or examined by a process, assuming that the permission requirements are met (see *tcsetgrp(3C)* or *tcgetgrp(3C)*). The terminal interface aids in this allocation by restricting access to the terminal by processes that are not in the foreground process group.

A process group is considered orphaned when the parent of every member of the process group is either itself a member of the process group or is not a member of the group's session (see *Sessions*).

Sessions

A process that creates a session (see *setsid(2)* or *setpgrp(2)*) becomes a session leader. Every process group belongs to exactly one session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its parent. A process can change its session membership (see *setpgid(2)* or *setpgrp(2)*). Usually a session comprises all the processes (including children) created as a result of a single login.

The Controlling Terminal

A terminal can belong to a process as its controlling terminal. Each process of a session that has a controlling terminal has the same controlling terminal. A terminal can be the controlling terminal for at most one session. The controlling terminal for a session is allocated by the session leader. If a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session without using the **O_NOCTTY** option (see *open(2)*), the terminal becomes the controlling terminal of the session and the controlling terminal's foreground process group is set to the process group of the session leader. While a controlling terminal is associated with a session, the session leader is said to be the controlling process of the controlling terminal.

The controlling terminal is inherited by a child process during a **fork()** (see *fork(2)*). A process relinquishes its controlling terminal if it creates a new session with **setsid()** or **setpgrp()** (see *setsid(2)* and *setpgrp(2)*), or when all file descriptors associated with the controlling terminal have been closed.

When the controlling process terminates, the controlling terminal is disassociated from the current session, allowing it to be acquired by a new session leader. A **SIGHUP** signal is sent to all processes in the foreground process group of the controlling terminal. Subsequent access to the terminal by other processes in the earlier session can be denied (see *Terminal Access Control*) with attempts to access the terminal treated as if a modem disconnect had been sensed.

Terminal Access Control

Read operations are allowed (see *Input Processing and Reading Data*) from processes in the foreground process group of their controlling terminal. If a process is not in the foreground process group of its controlling terminal, the process and all member's of its process group are considered to be in a background process group of this controlling terminal. All attempts by a process in a background process group to read

from its controlling terminal will be denied. If denied and the reading process is ignoring or blocking the **SIGTTIN** signal, or the process (on systems that implement *vfork* separately from *fork*) has made a call to *vfork(2)* but has not yet made a call to *exec(2)*, or the process group of the reading process is orphaned, **read()** returns **-1** with **errno** set to **EIO** and no signal is sent. In all other cases where the read is denied, the process group of the reading process will be sent a **SIGTTIN** signal. The default action of the **SIGTTIN** signal is to stop the process to which it is sent.

If the process is in the foreground process group of its controlling terminal, write operations are allowed (see *Writing Data and Output Processing*). Attempts by a process in a background process group to write to its controlling terminal are denied if **TOSTOP** (see *Local Modes*) is set, the process is not ignoring and not blocking the **SIGTTOU** signal, and the process (on systems that implement *vfork* separately from *fork*) has not made a call to *vfork(2)* without making a subsequent call to *exec(2)*. If the write is denied and the background process group is orphaned, the **write()** returns **-1** with **errno** set to **EIO**. If the write is denied and the background process group is not orphaned, the **SIGTTOU** signal is sent to the process group of the writing process. The default action of the **SIGTTOU** signal is to stop the process to which it is sent.

Certain calls that set terminal parameters are treated in the same fashion as write, except that **TOSTOP** is ignored; that is, the effect is identical to that of terminal writes when **TOSTOP** is set.

Input Processing and Reading Data

A terminal device associated with a terminal device file can operate in full-duplex mode, so that data can arrive, even while data output is occurring. Each terminal device file has an *input queue* associated with it into which incoming data is stored by the system before being read by a process. The system imposes a limit, **MAX_INPUT**, on the number of characters that can be stored in the input queue. This limit is dependent on the particular implementation, but is at least 256. When the input limit is reached, all saved characters are discarded without notice.

All input is processed either in canonical mode or non-canonical mode (see *Canonical Mode Input Processing* and *Non-Canonical Mode Input Processing*). Additionally, input characters are processed according to the **c_iflag** (see *Input Modes*) and **c_lflag** (see *Local Modes*) fields. For example, such processing can include *echoing*, which in general means transmitting input characters immediately back to the terminal when they are received from the terminal. This is useful for terminals that operate in full-duplex mode.

The manner in which data is provided to a process reading from a terminal device file depends on whether the terminal device file is in canonical or non-canonical mode.

Another dependency is whether the **O_NONBLOCK** or **O_NDELAY** flag is set by either *open(2)* or *fcntl(2)*. If the **O_NONBLOCK** and **O_NDELAY** flags are both clear, the read request is blocked until data is available or a signal is received. If either the **O_NONBLOCK** or **O_NDELAY** flag is set, the read request completes without blocking in one of three ways:

- If there is enough data available to satisfy the entire request, **read()** completes successfully, having read all of the data requested, and returns the number of characters read.
- If there is not enough data available to satisfy the entire request, **read()** completes successfully, having read as much data as possible, and returns the number of characters read.
- If there is no data available, **read()** returns **-1**, with **errno** set to **EAGAIN** when the **O_NONBLOCK** flag is set. Otherwise, (flag **O_NONBLOCK** is clear and **O_NDELAY** is set) **read()** completes successfully, having read no data, and returns a count of 0.

The availability of data depends upon whether the input processing mode is canonical or non-canonical. The following sections, *Canonical Mode Input Processing* and *Non-Canonical Mode Input Processing*, describe each of these input processing modes.

Canonical Mode Input Processing (Erase and Kill Processing)

In canonical mode input processing, terminal input is processed in units of lines, where a line is delimited by a new-line (NL) character, an end-of-file (EOF) character, or an end-of-line character (EOL) or (EOL2). See *Special Characters* for more information on **NL**, **EOF**, **EOL**, and **EOL2**. This means that a read request does not return until an entire line has been typed or a signal has been received. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters can be requested in a read, even one, without losing information.

MAX_CANON is the limit on the number of characters in a line. This limit varies with each particular implementation, but is at least 256.

When the **MAX_CANON** limit is reached, all characters in the current undelimited line are discarded without notice.

Erase and kill processing occur when any of three special characters, the ERASE, WERASE, or KILL characters (see *Special Characters*), is received. This processing affects data in the input queue that has not yet been delimited by a NL, EOF, EOL, or EOL2 character. This undelimited data makes up the current line. The ERASE character deletes the last character in the current line, if one exists. The WERASE character deletes the last word in the current line, if one exists. A word is defined as a series of non-blank characters (tabs are equivalent to blanks). The KILL character deletes all data in the current line, if any, and optionally outputs a new-line (NL) character. These characters operate on a key-stroke basis, independent of any backspacing or tabbing that may have preceded them. ERASE, WERASE, and KILL characters have no effect if the current line is empty. ERASE, WERASE, and KILL characters are not placed in the input queue.

Non-Canonical Mode Input Processing (MIN/TIME Interaction)

In non-canonical mode input processing, input characters are not assembled into lines, and erase and kill processing does not occur. The values of the **MIN** and **TIME** members of the **c_cc** array (see *termios Structure*) are used to determine how to process the characters received. **MIN** represents the minimum number of characters that should be received before **read()** successfully returns. **TIME** is a timer of 0.10 second granularity that is used to timeout bursty and short term data transmissions. The four possible cases for **MIN** and **TIME** and their interactions are described below.

Case A: **MIN** > 0, **TIME** > 0

In this case, **TIME** serves as an inter-character timer and is activated after the first character is received. Since it is an inter-character timer, it is reset after each character is received. The interaction between **MIN** and **TIME** is as follows:

- As soon as one character is received, the inter-character timer is started.
- If **MIN** characters are received before the inter-character timer expires (remember that the timer is reset upon receipt of each character), the read is satisfied. If the timer expires before **MIN** characters are received, the characters received to that point are returned to the user.
- Note that if **TIME** expires, at least one character will be returned because the timer would not have been enabled unless a character was received. In this case (**MIN** > 0, **TIME** > 0) the read blocks until the **MIN** and **TIME** mechanisms are activated by the receipt of the first character, or a signal is received.

Case B: **MIN** > 0, **TIME** = 0

In this case, since the value of **TIME** is zero, the timer plays no role and only **MIN** is significant. A pending read is not satisfied until **MIN** characters are received after any previous read completes (that is, the pending read blocks until **MIN** characters are received), or a signal is received. A program that uses this case to handle record-based terminal I/O can block indefinitely in the read operation.

Case C: **MIN** = 0, **TIME** > 0

In this case, since the value of **MIN** is zero, **TIME** no longer represents an inter-character timer. It now serves as a read timer that is activated as soon as the **read()** function is processed. A read is satisfied as soon as a single character is received *or* the read timer expires. If the timer expires, no character is returned. If the timer does not expire, the only way the read can be satisfied is by a character being received. A read cannot block indefinitely waiting for a character because if no character is received within **TIME** × 0.10 seconds after the read is initiated, **read()** returns a value of zero, having read no data.

Case D: **MIN** = 0, **TIME** = 0

The number of characters requested or the number of characters currently available, whichever is less, is returned without waiting for more characters to be input. If no characters are available, **read()** returns a value of zero, having read no data.

Some points to note about **MIN** and **TIME**:

1. In the above explanations, the interactions of **MIN** and **TIME** are not symmetric. For example, when **MIN** > 0 and **TIME** = 0, **TIME** has no effect. However, in the opposite case where **MIN** = 0 and **TIME** > 0, both **MIN** and **TIME** play a role in that **MIN** is satisfied with the receipt of a single character.
2. Also note that in case A (**MIN** > 0, **TIME** > 0), **TIME** represents an inter-character timer while in case C (**MIN** = 0, **TIME** > 0), **TIME** represents a read timer.

These two points highlight the dual purpose of the MIN/TIME feature. Cases A and B (where MIN > 0) exist to handle burst mode activity (such as file transfer programs) where a program would like to process at least MIN characters at a time. In case A, the inter-character timer is activated by a user as a safety measure while in case B it is turned off.

Cases C and D exist to handle single character timed transfers. These cases are readily adaptable to screen-based applications that need to know if a character is present in the input queue before refreshing the screen. In case C the read is timed, while in case D it is not.

Another important note is that MIN is always just a minimum. It does not denote a record length. For example, if a program initiates a read of 20 characters when MIN is 10 and 25 characters are present, 20 characters will be returned to the user. Had the program requested all characters, all 25 characters would be returned to the user.

Furthermore, if TIME is greater than zero and MIN is greater than **MAX_INPUT**, the read will never terminate as a result of MIN characters being received because all the saved characters are discarded without notice when **MAX_INPUT** is exceeded. If TIME is zero and MIN is greater than **MAX_INPUT**, the read will never terminate unless a signal is received.

Special Characters

Certain characters have special functions on input, output, or both. Unless specifically denied, each special character can be changed or disabled. To disable a character, set its value to **_POSIX_VDISABLE** (see *unistd(5)*). These special functions and their default character values are:

INTR	(Rubout or ASCII DEL) special character on input and is recognized if ISIG (see <i>Local Modes</i>) is enabled. Generates a SIGINT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. Normally, each such process is forced to terminate, but arrangements can be made to either ignore or hold the signal, or to receive a trap to an agreed-upon location; see <i>signal(2)</i> and <i>signal(5)</i> . If ISIG is set, the INTR character is discarded when processed. If ISIG is clear, the INTR character is processed as a normal data character, and no signal is sent.
QUIT	(Control-\ or ASCII FS) special character on input. Recognized if ISIG (see <i>Local Modes</i>) is set. The treatment of this character is identical to that of the INTR character except that a SIGQUIT signal is generated and the processes that receive this signal are not only terminated, but a core image file (called core) is created in the current working directory if the implementation supports core files.
SWTCH	(ASCII NUL) special character on input and is only used by the shell layers facility <i>sh(1)</i> . The shell layers facility is not part of the general terminal interface. No special functions are performed by the general terminal interface when SWTCH characters are encountered.
ERASE	(#) special character on input and is recognized if ICANON (see <i>Local Modes</i>) is enabled. Erases the preceding character. Does not erase beyond the start of a line, as delimited by a NL, EOF, EOL, or EOL2 character. If ICANON is enabled, the ERASE character is discarded when processed. If ICANON is not enabled, the ERASE character is treated as a normal data character.
WERASE	(disabled) special character on input and is recognized if ICANON (see <i>Local Modes</i>) is enabled. Erases the preceding word. Does not erase beyond the start of a line, as delimited by a NL, EOF, EOL, or EOL2 character. If ICANON is enabled, the WERASE character is discarded when processed. If ICANON is not enabled, the WERASE character is treated as a normal data character.
KILL	(@) special character on input and is recognized if ICANON is enabled. KILL deletes the entire line, as delimited by a NL, EOF, EOL, or EOL2 character. If ICANON is enabled, the KILL character is discarded when processed. If ICANON is not enabled, the KILL character is treated as a normal data character.
EOF	(Control-D or ASCII EOT) special character on input and is recognized if ICANON is enabled. EOF can be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program without waiting for a new-line, and the EOF is discarded. Thus, if there are no characters waiting, (that is, the EOF occurred at the beginning of a line) a character count of zero is returned from read() , representing an end-of-file indication. If ICANON is enabled, the EOF character is discarded when processed. If ICANON is not enabled, the EOF

character is treated as a normal data character.

NL	(ASCII LF) special character on input and is recognized if ICANON flag is enabled. It is the line delimiter (<code>\n</code>). If ICANON is not enabled, the NL character is treated as a normal data character.
EOL	(ASCII NUL) special character on input and is recognized if ICANON is enabled. EOL is an additional line delimiter similar to NL. It is not normally used. If ICANON is not enabled, the EOL character is treated as a normal data character.
EOL2	(disabled) special character on input and is recognized if ICANON is enabled. EOL2 is an additional line delimiter similar to EOL. It is not normally used. If ICANON is not enabled, the EOL2 character is treated as a normal data character.
SUSP	(disabled) special character recognized on input. If ISIG is enabled, receipt of the SUSP character causes a SIGTSTP signal to be sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the SUSP character is discarded when processed. If ISIG is not enabled, the SUSP character is treated as a normal data character. Command interpreter processes typically set SUSP to Control-Z.
DSUSP	(disabled) special character recognized on input. If ISIG is enabled, and a process in the foreground process group attempts to read the DSUSP character, a SIGTSTP signal is sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the DSUSP character is then discarded. If ISIG is not enabled, the DSUSP character is treated as a normal data character. Note that DSUSP is similar to SUSP except that the signal is sent when a process in the foreground process group attempts to read the DSUSP character, rather than when it is typed.
STOP	(Control-S or ASCII DC3) special character on both input and output. If IXON (output control) is enabled, processing of the STOP character temporarily suspends output to the terminal device. This is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended and IXON is enabled, STOP characters are ignored and not read. If IXON is enabled, the STOP character is discarded when processed. If IXON is not enabled, the STOP character is treated as a normal data character. If IXOFF (input control) is enabled, the system sends a STOP character to the terminal device when the number of unread characters in the input queue is approaching a system specified limit. This is an attempt to prevent this buffer from overflowing by telling the terminal device to stop sending data.
START	(Control-Q or ASCII DC1) special character on both input and output. If IXON (output control) is enabled, processing of the START character resumes output that has been suspended. While output is not suspended and IXON is enabled, START characters are ignored and not read. If IXON is enabled, the START character is discarded when processed. If IXON is not enabled, the START character is treated as a normal data character. If IXOFF (input control) is enabled, the system sends a START character to the terminal device when the input queue has drained to a certain system-defined level. This occurs when the input queue is no longer in danger of possibly overflowing.
CR	(ASCII CR) special character on input is recognized if ICANON is enabled. When ICANON and ICRNL are enabled and IGNCR is not enabled, this character is translated into a NL, and has the same affect as the NL character. If ICANON and IGNCR are enabled, the CR character is ignored. If ICANON is enabled and both ICRNL and IGNCR are not enabled, the CR character is treated as a normal data character.
LNEXT	(disabled) special character recognized on input. Causes the special meaning of the next character to be ignored. This works for all special characters specified above. It allows characters to be input that would otherwise be interpreted by the system for a special function.

The special characters are assigned their default character values when the terminal port is opened. The default values used are those specified by the System V Interface Definition, Third Edition (SVID3), except for the WERASE (Control-W) and LNEXT (Control-V) characters which are set to **_POSIX_VDISABLE** to maintain binary compatibility with previous releases of HP-UX. The default character values assigned when the port is opened can be changed for all ports on a system wide basis through the use of the **stty** command (see *stty*(1)). The character values may also be changed for a specific port after it is opened using

the **stty** command. The NL and CR characters cannot be changed or disabled. The character values for the remaining special characters can be changed or disabled to suit individual tastes.

If **ICANON** is set (see *Local Modes*), the ERASE, KILL, and EOF characters can be escaped by a preceding **** character, in which case no special function is performed. These characters, and the remaining special characters, may also be escaped by preceding them with the LNEXT character (see LNEXT above).

If two or more special characters have the same value, the function performed when the character is processed is undefined.

Modem Disconnect

If a modem disconnect is detected by the terminal interface for a controlling terminal, and if **CLOCAL** is clear in the **c_cflag** field for the terminal (see *Control Modes*), the **SIGHUP** signal is sent to the controlling process of the controlling terminal. Unless other arrangements have been made, this causes the controlling process to terminate. Any subsequent read from the terminal device returns with an end-of-file indication until the device is closed. Thus, processes that read a terminal file and test for end-of-file can terminate appropriately after a disconnect. Any subsequent **write()** to the terminal device returns **-1**, with **errno** set to **EIO**, until the device is closed.

Closing a Terminal Device File

The last process to close a terminal device file causes any output not already sent to the device to be sent to the device even if output was suspended. This last close always blocks (even if non-blocking I/O has been specified) until all output has been sent to the terminal device. Any input that has been received but not read is discarded.

Writing Data and Output Processing

When characters are written, they are placed on the output queue. Characters on the output queue are transmitted to the terminal as soon as previously-written characters are sent. These characters are processed according to the **c_oflag** field (see *Output Modes*). Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters for output more rapidly than they can be sent, the process is suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the process is resumed.

termios Structure

Routines that need to control certain terminal I/O characteristics can do so by using the **termios** structure as defined in the header file **<termios.h>**. The structure is defined as follows:

```
#define NCCS      16
struct termios {
    tcflag_t  c_iflag;    /* input modes */
    tcflag_t  c_oflag;    /* output modes */
    tcflag_t  c_cflag;    /* control modes */
    tcflag_t  c_lflag;    /* local modes */
    tcflag_t  c_reserved; /* reserved for future use */
    cc_t      c_cc[NCCS]; /* control chars */
};
```

The special characters are defined by the array **c_cc**. The relative positions and default values for each special character function are as follows:

INTR	VINTR	DEL
QUIT	VQUIT	Control-
ERASE	VERASE	#
KILL	VKILL	@
EOF	VEOF	Control-D
EOL	VEOL	NUL
EOL2	VEOL2	disabled
MIN	VMIN	NUL
TIME	VTIME	Control-D
SUSP	VSUSP	disabled

START	VSTART	Control-Q
STOP	VSTOP	Control-S
WERASE	VWERASE	disabled
LNEXT	VLNEXT	disabled
DSUSP	VDSUSP	disabled

termio Structure

The **termio** structure has been superseded by the **termios** structure and is provided for backward compatibility with prior applications (see *termio Caveats*). The structure is defined in the header file `<termio.h>` and is defined as follows:

```
#define NCC      8
struct termio {
    unsigned short  c_iflag;    /* input modes */
    unsigned short  c_oflag;    /* output modes */
    unsigned short  c_cflag;    /* control modes */
    unsigned short  c_lflag;    /* local modes */
    char            c_line;     /* line discipline */
    unsigned char   c_cc[NCC];  /* control chars */
};
```

Modes

The next four sections describe the specific terminal characteristics that can be set using the **termios** and **termio** structures (see *termio Caveats*). Any bits in the modes fields that are not explicitly defined below are ignored. However, they should always be clear to prevent future compatibility problems.

Input Modes

The **c_iflag** field describes the basic terminal input control:

IGNBRK	Ignore break condition.
BRKINT	Signal interrupt on break.
IGNPAR	Ignore characters with parity errors.
PARMRK	Mark parity errors.
INPCK	Enable input parity check.
ISTRIP	Strip character.
INLCR	Map NL to CR on input.
IGNCR	Ignore CR.
ICRNL	Map CR to NL on input.
IUCLC	Map uppercase to lowercase on input.
IXON	Enable start/stop output control.
IXANY	Enable any character to restart output.
IXOFF	Enable start/stop input control.
IMAXBEL	Enable BEL on input line too long.

A break condition is defined as a sequence of zero-value bits that continues for more than the time to send one character. For example, a character framing or parity error with data all zeros is interpreted as a single break condition.

If **IGNBRK** is set, the break condition is ignored. Therefore the break condition cannot be read by any process. If **IGNBRK** is clear and **BRKINT** is set, the break condition flushes both the input and output queues and, if the terminal is the controlling terminal of a foreground process group, the break condition generates a single **SIGINT** signal to that foreground process group. If neither **IGNBRK** nor **BRKINT** is set, a break condition is read as a single `\0` character, or if **PARMRK** is set, as the three-character sequence `\377, \0, \0`.

If **IGNPAR** is set, characters with other framing and parity errors (other than break) are ignored.

If **PARMRK** is set, and **IGNPAR** is clear, a character with a framing or parity error (other than break) is read as the three-character sequence: `\377, \0, X`, where *X* is the data of the character received in error. To avoid ambiguity in this case, if **ISTRIP** is clear, a valid character of `\377` is read as `\377, \377`. If both **PARMRK** and **IGNPAR** are clear, a framing or parity error (other than break) is read as the character `\0`.

If **INPCK** is set, input parity checking is enabled. If **INPCK** is clear, input parity checking is disabled. Whether input parity checking is enabled or disabled is independent of whether parity detection is enabled or disabled (see *Control Modes*). If **PARENB** is set (see *Control Modes*) and **INPCK** is clear, parity

generation is enabled but input parity checking is disabled; the hardware to which the terminal is connected will recognize the parity bit, but the terminal special file will not check whether this bit is set correctly or not.

The following table shows the interrelationship between the flags **IGNBRK**, **BRKINT**, **IGNPAR**, and **PARMRK**. The column marked **Input** gives various types of input characters received, indicated as follows:

O	NUL character (<code>\0</code>)
C	Character other than NUL
P	Parity error detected
F	Framing error detected

Items enclosed in brackets indicate one or more of the conditions are true.

If the **INPCK** flag is clear, characters received with parity errors are not processed according to this table, but instead, as if no parity error had occurred. Under the flag columns, **Set** indicates the flag is set, **Clear** indicates the flag is not set, and **X** indicates the flag may be set or clear. The column labeled **Read** shows the results that will be passed to the application code. A **—** indicates that no character or condition is passed to the application code. The value **SIGINT** indicates that no character is returned, but that the **SIGINT** signal is sent to the foreground process group of the controlling terminal.

Input	IGNBRK	BRKINT	IGNPAR	PARMRK	Read
0[PF]	Set	X	X	X	—
0[PF]	Clear	Set	X	X	SIGINT
0[PF]	Clear	Clear	X	Set	'\377','\0','\0'
0[PF]	Clear	Clear	X	Clear	'\0'
C[PF]	X	X	Set	X	—
C[PF]	X	X	Clear	Set	'\377','\0',C
C[PF]	X	X	Clear	Clear	'\0'
'\377'	X	X	X	Set	'\377','\377'

If **ISTRIP** is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If **INLCR** is set, a received NL character is translated into a CR character. If **IGNCR** is set, a received CR character is ignored (not read). If **IGNCR** is clear and **ICRNL** is set, a received CR character is translated into a NL character.

If **IUCLC** is set, a received uppercase alphabetic character is translated into the corresponding lowercase character.

If **IXON** is set, start/stop output control is enabled. A received STOP character suspends output and a received START character restarts output. If **IXANY** and **IXON** are set, any input character without a framing or parity error restarts output that has been suspended. When these three flags are set, output suspended, and an input character received with a framing or parity error, output resumes if processing it results in data being read. When **IXON** is set, START and STOP characters are not read, but merely perform flow control functions. When **IXON** is clear, the START and STOP characters are read.

If **IXOFF** is set, start/stop input control is enabled. The system transmits a STOP character when the number of characters in the input queue exceeds a system defined value (high water mark). This is intended to cause the terminal device to stop transmitting data in order to prevent the number of characters in the input queue from exceeding **MAX_INPUT**. When enough characters have been read from the input queue that the number of characters remaining is less than another system defined value (low water mark), the system transmits a START character which is intended to cause the terminal device to resume transmitting data (without risk of overflowing the input queue). In order to avoid potential deadlock, **IXOFF** is ignored in canonical mode whenever there is no line delimiter in the input buffer. In this case, the STOP character is not sent at the high water mark, but will be transmitted later if a delimiter is received. If all complete lines are read from the input queue leaving only a partial line with no line delimiter, the START character is sent, even if the number of characters is still greater than the low water mark. When **ICANON** is set and the input stream contains more characters between line delimiters than the high water mark allows, there is no guarantee that **IXOFF** can prevent buffer overflow and data loss, because the STOP character may not be sent in time, if at all.

If **IMAXBEL** is set, the ASCII BEL character is echoed if the input queue overflows. Further input is not stored, but any input present in the input queue is not discarded. If **IMAXBEL** is clear, no ASCII BEL character is echoed, and the input already present in the input queue is discarded when the input queue overflows.

The initial input control value is all bits clear.

Output Modes

The **c_oflag** field specifies the system treatment of output:

OPOST	Postprocess output.
OLCUC	Map lowercase to uppercase on output.
ONLCR	Map NL to CR-NL on output.
OCRNL	Map CR to NL on output.
ONOCR	No CR output at column 0.
ONLRET	NL performs CR function.
OFILL	Use fill characters for delay.
OFDEL	Fill is DEL, else NUL.
NLDLY	Select new-line delays:
NL0	No delay
NL1	Delay type 1
CRDLY	Select carriage-return delays:
CR0	No delay
CR1	Delay type 1
CR2	Delay type 2
CR3	Delay type 3
TABDLY	Select horizontal-tab delays:
TAB0	No delay
TAB1	Delay type 1
TAB2	Delay type 2
TAB3	Expand tabs to spaces.
XTABS	Expand tabs to spaces.
BSDLY	Select backspace delays:
BS0	No delay
BS1	Delay type 1
VTDLY	Select vertical-tab delays:
VT0	No delay
VT1	Delay type 1
FFDLY	Select form-feed delays:
FF0	No delay
FF1	Delay type 1

If **OPOST** is set, output characters are post-processed as indicated by the remaining flags; otherwise characters are transmitted without change.

If **OLCUC** is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This function is often used in conjunction with **IUCLC**.

If **ONLCR** is set, the NL character is transmitted as the CR-NL character pair. If **OCRNL** is set, the CR character is transmitted as the NL character. If **ONOCR** is set, no CR character is transmitted when at column 0 (first position). If **ONLRET** is set, the NL character is assumed to do the carriage-return function; the column pointer will be set to 0, and the delays specified for CR will be used. If **ONLRET** is clear, the NL character is assumed to perform only the line-feed function; the delays specified for NL are used and the column pointer remains unchanged. For all of these cases, the column pointer is always set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. The values of **NL0**, **CR0**, **TAB0**, **BS0**, **VT0**, and **FF0** indicate no delay. If **OFILL** is set, fill characters are transmitted for delay instead of a timed delay. This is useful for high baud rate terminals, that need only a minimal delay. If **OFDEL** is set, the fill character is DEL; otherwise NUL.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If **ONLRET** is set, carriage-return delays are used instead of the new-line delays. If **OFILL** is set, two fill characters are transmitted.

Carriage-return delay type 1 depends on the current column position; type 2 is about 0.10 seconds; type 3 about 0.15 seconds. If **OFILL** is set, delay type 1 transmits two fill characters; type 2, four fill characters.

Horizontal-tab delay type 1 is depends on the current column position. Type 2 is about 0.10 seconds; type 3 specifies that tabs are to be expanded into spaces. If **OFILL** is set, two fill characters are transmitted for

any delay.

Backspace delay lasts about 0.05 seconds. If **OFILL** is set, one fill character is transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

Control Modes

The **c_cflag** field describes the hardware control of the terminal:

CBAUD	Baud rate:	CSIZE	Character size:
B0	Hang up	CS5	5 bits
B50	50 baud	CS6	6 bits
B75	75 baud	CS7	7 bits
B110	110 baud	CS8	8 bits
B134	134.5 baud		
B150	150 baud	CSTOPB	Send two stop bits, else one.
B200	200 baud	CREAD	Enable receiver.
B300	300 baud	PARENB	Parity enable.
B600	600 baud	PARODD	Odd parity, else even.
B900	900 baud	HUPCL	Hang up on last close.
B1200	1200 baud	CLOCAL	Local line, else dial-up.
B1800	1800 baud	LOBLK	Reserved for use by <i>shl(1)</i> .
B2400	2400 baud		
B3600	3600 baud		
B4800	4800 baud		
B7200	7200 baud		
B9600	9600 baud		
B19200	19200 baud		
B38400	38400 baud		
EXTA	External A		
EXTB	External B		

The **CBAUD** bits specify the baud rate. The zero baud rate, **B0**, is used to hang up the connection. If **B0** is specified, the modem control lines (see *modem(7)*) cease to be asserted. Normally, this disconnects the line. For any particular hardware, impossible speed changes are ignored. **CBAUD** is provided for use with the **termio** structure. When the **termios** structure is used, several routines are available for setting and getting the input and output baud rates (see *termios Structure Related Functions*).

The **CSIZE** bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If **CSTOPB** is set, two stop bits are used; otherwise one stop bit. For example, at 110 baud, many devices require two stop bits.

If **PARENB** is set, parity generation is enabled (a parity bit is added to each output character). Furthermore, parity detection is enabled (incoming characters are checked for the correct parity). If **PARENB** is set, **PARODD** specifies odd parity if set; otherwise even parity is used. If **PARENB** is clear, both parity generation and parity checking are disabled.

If **CREAD** is set, the receiver is enabled. Otherwise no characters can be received.

The specific effects of the **HUPCL** and **CLOCAL** bits depend on the mode and type of the modem control in effect. See *modem(7)* for the details.

If **HUPCL** is set, the modem control lines for the port are lowered (disconnected) when the last process using the open port closes it or terminates.

If **CLOCAL** is set, a connection does not depend on the state of the modem status lines. If **CLOCAL** is clear, the modem status lines are monitored.

Under normal circumstances, a call to **read()** waits for a modem connection to complete. However, if either the **O_NDELAY** or the **O_NONBLOCK** flags are set or **CLOCAL** is set, the **open()** returns immediately without waiting for the connection. If **CLOCAL** is set, see *Modem Disconnect* for the effects of **read()** and **write()** for those files for which the connection has not been established or has been lost.

LOBLK is used by the shell layers facility (see *shl(1)*). The shell layers facility is not part of the general terminal interface, and the **LOBLK** bit is not examined by the general terminal interface.

The initial hardware control value after open is **B300**, **CS8**, **CREAD**, and **HUPCL**.

Local Modes

The **c_lflag** field is used to control terminal functions.

ISIG	Enable signals.
ICANON	Canonical input (erase and kill processing).
XCASE	Canonical upper/lower presentation.
ECHO	Enable echo.
ECHOE	Echo ERASE as correcting backspace sequence.
ECHOK	Echo NL after kill character.
ECHONL	Echo NL.
NOFLSH	Disable flush after interrupt, quit, or suspend.
TOSTOP	Send SIGTTOU for background output.
ECHOCTL	Echo control characters as ^char, DEL as ^?.
ECHOPRT	Echo erased character as character is erased.
ECHOKE	BS SP BS erase entire line on line kill.
FLUSHO	Output is being flushed.
PENDIN	Reprocess pending input at next read or input character.
IEXTEN	Enable extended functions.

If **ISIG** is set, each input character is checked against the special control characters INTR, QUIT, SUSP, and DSUSP (see *Process Group Control IOCTL Commands*). If an input character matches one of these control characters, the function associated with that character is performed and the character is discarded. If **ISIG** is clear, no checking is done and the character is treated as a normal data character. Thus these special input functions are possible only if **ISIG** is set.

If **ICANON** is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, EOL, or EOL2. If **ICANON** is clear, read requests are satisfied directly from the input queue. A read blocks until at least MIN characters have been received or the timeout value TIME has expired between characters. (See *Non-Canonical Mode Input Processing (MIN/TIME Interaction)*). This allows fast bursts of input to be read efficiently while still allowing single-character input. The time value represents tenths of seconds.

If **XCASE** is set, and if **ICANON** is set, an uppercase letter is accepted on input by preceding it with a \ character, and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

To obtain:	Use:
\	\\
	\\
{	\\{
}	\\}
\	\\

For example, **A** is input as **\a**, **\n** as **\\n**, and **\N** as **\\N**. **XCASE** would normally be used in conjunction with **IUCLC** and **OLCUC** for terminals that support only the first-sixty-four-character limited character set. In this case, **IUCLC** processing is done before **XCASE** for input, and processing is done after **XCASE** for output. Therefore typing **A** causes an **a** to be read because of **IUCLC**, and typing **\A** causes an **A** to be read since **IUCLC** produces **\a** which is turned into **A** by the **XCASE** processing.

If **ECHO** is set, characters are echoed back to the terminal when received. If **ECHO** is clear, characters are not echoed.

When **ICANON** is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, EOL and EOL2 as described in *Canonical Mode Input Processing*. Furthermore, the following echo functions are possible.

If **ECHO** and **ECHOE** are set, the ERASE and WERASE characters are echoed as the three-character ASCII sequence BS SP BS, which clears the last character or word from the CRT screen.

If **ECHO** and **ECHOPRT** are set, and **ECHOE** is clear, the first ERASE and WERASE character in a sequence echoes a backslash (\) followed by the characters being erased. Subsequent ERASE or WERASE characters echo the characters being erased in reverse order. The next non-erase character causes a slash (/) to be typed before it is echoed.

If **ECHOKE** and **ECHO** are set, the KILL character is echoed by erasing each character on the line from the CRT screen using the method selected by **ECHOE** and **ECHOPRT**.

If **ECHOCTL** and **ECHO** are set, all control characters (characters with codes between 0 and 37 octal) other than ASCII TAB, ASCII NL, the START and STOP characters, ASCII CR, and ASCII BS are echoed as `^char`, where `char` is the character given by adding 100 octal to the control character's code.

If **ECHOK** is set and **ECHOKE** is not set, the NL character is echoed after the kill character to emphasize that the line is being deleted.

If **ECHONL** is set, the NL character is echoed even if **ECHO** is clear. This is useful for terminals set to local echo (that is, half duplex).

Unless escaped, the EOF character is not echoed. Because ASCII EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If **NOFLSH** is set, the normal flush of the input and output queues associated with quit, interrupt, and suspend characters is not done. However, **NOFLSH** does not affect the flushing of data upon receipt of a break when **BRKINT** is set.

If the **TOSTOP** bit is set, an attempt by a process that is not in the foreground process group to write to its controlling terminal will be denied when the process is not ignoring and not blocking the **SIGTTOU** signal. If the write is denied and the process is a member of an orphaned process group **write()** returns -1 and sets **errno** to **EIO** and no signal is sent. If the write is denied and the process is not a member of an orphaned process group, the **SIGTTOU** signal is sent to that process group.

If **FLUSHO** is set, data written to the terminal device is discarded. This bit is set by a program. A program can cancel the **FLUSHO** effect by clearing **FLUSHO**.

If **PENDIN** is set, any input that has not been read is reprocessed and possibly re-echoed when the next character arrives as input.

If **ICANON** is set, the ERASE, KILL, and EOF characters can be escaped by a preceding `\` character, in which case no special function is done.

IEXTEN must be set before the **ECHOCTL**, **ECHOPRT**, **ECHOKE**, **FLUSHO**, and **PENDIN** functions are allowed. In addition, the special characters WERASE and LNEXT are allowed only if **IEXTEN** is set. **IEXTEN** does not affect any other functions.

The initial local control value is all-bits-clear.

Special Control Characters

Special control characters are defined in the array `c_cc`. All of these special characters can be changed. The subscript name and description for each element in both canonical and non-canonical mode are shown in the following table.

Canonical	Subscript Usage Non-Canonical	Description
VEOF		EOF character
VEOL		EOL character
VEOL2		EOL2 character
VERASE		ERASE character
VWERASE		WERASE character
VINTR	VINTR	INTR character
VKILL		KILL character
	VMIN	MIN value
VQUIT	VQUIT	QUIT character
VSTART	VSTART	START character
VSTOP	VSTOP	STOP character
VSUSP	VSUSP	SUSP character
VDSUSP	VDSUSP	DSUSP character
	VTIME	TIME value
VLNEXT	VLNEXT	LNEXT character

termios Structure-Related Functions

The following functions are provided when using the *termios* structure. Note that the effects on the terminal device of the **cfsetispeed()** and **cfsetospeed()** functions do not become effective until the **tcsetattr()** function is successfully called. Refer to the appropriate manual entries for details.

termios Structure Functions

Function	Description
<code>cfgetospeed()</code>	get output baud rate
<code>cfgetispeed()</code>	get input baud rate
<code>cfsetospeed()</code>	set output baud rate
<code>cfsetispeed()</code>	set input baud rate
<code>tcgetattr()</code>	get terminal state
<code>tcsetattr()</code>	set terminal state

termio Structure-Related IOCTL Commands

Several `ioctl()` system calls apply to terminal files that use the `termio` structure (see *termio Structure*). If a requested command is not recognized, the request returns `-1` with `errno` set to `[EINVAL]`.

`ioctl()` system calls that reference the `termio` structure have the form:

```
ioctl (fildes, command, arg)
struct termio *arg;
```

Commands using this form are:

TCGETA	Get the parameters associated with the terminal and store them in the <code>termio</code> structure referenced by <code>arg</code> . This command is allowed from a background process; however, the information may be subsequently changed by a foreground process.
TCSETA	Set the parameters associated with the terminal from the <code>termio</code> structure referenced by <code>arg</code> . The change is immediate. If characters are being output when the command is requested, results are undefined and the output may be garbled.
TCSETAW	Wait for the output to drain before setting new parameters. This form should be used when changing parameters that affect output.
TCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters.

termio Caveats

Only the first eight special control characters (see *termios Structure*) can be set or returned. The values of indices `VEOL` and `VEOF` are the same as indices `VTIME` and `VMIN` respectively. Hence if `ICANON` is set, `VEOL` or `VTIME` is the additional end-of-line character and `VEOF` or `VMIN` is the end-of-file character. If `ICANON` is clear, `VEOL` or `VTIME` is the inter-character-timer value and `VEOF` or `VMIN` is the minimum number of characters desired for reads.

Structure-Independent Functions

The following functions which are independent of both the `termio` and `termios` structures are provided for controlling terminals. Refer to the appropriate manual entries for details.

Structure-Independent Functions

Function	Description
<code>tcsendbreak()</code>	send a break
<code>tcdrain()</code>	wait until output has drained
<code>tcflush()</code>	flush input or output queue or both
<code>tcflow()</code>	suspend or resume input or output
<code>tcgetpgrp()</code>	get foreground process group id
<code>tcsetpgrp()</code>	set foreground process group id
<code>tcgetsid()</code>	get session id

System Asynchronous I/O IOCTL Commands

The following `ioctl()` system calls provide for system asynchronous I/O and have the form:

```
ioctl (fildes, command, arg)
int *arg;
```

Commands using this form are:

FIOSSAIOSTAT	If the integer referenced by <code>arg</code> is non-zero, system asynchronous I/O is enabled; that is, enable <code>SIGIO</code> to be sent to the process currently designated with <code>FIOSSAIOOWN</code> (see below) whenever the terminal device file status changes from "no read data available" to "read data available". If no process has been designated with <code>FIOSSAIOOWN</code> , enable <code>SIGIO</code> to be sent to the first process that opened the terminal device file.
---------------------	---

If the designated process has exited, the **SIGIO** signal is not sent to any process.

If the integer referenced by *arg* is 0, system asynchronous I/O is disabled.

The default on open of a terminal device file is that system asynchronous I/O is disabled.

FIGO_SAIostat The integer referenced by *arg* is set to 1 if system asynchronous I/O is enabled. Otherwise, the integer referenced by *arg* is set to 0.

FIGO_SAIoOwn Set the process ID that will receive the **SIGIO** signals due to system asynchronous I/O to the value of the integer referenced by *arg*. If no process can be found corresponding to that specified by the integer referenced by *arg*, the call returns -1 with **errno** set to [ESRCH]. A user with appropriate privileges can designate that any process receive the **SIGIO** signals. If the request is not made by a user with appropriate privileges and the calling process does not either designate that itself or another process whose real, saved, or effective user ID matches its real or effective user ID or the calling process does not designate a process that is a descendant of the calling process to receive the **SIGIO** signals, the call returns -1 with **errno** set to [EPERM]. See *privileges(5)* for more information about privileged access on systems that support fine-grained privileges.

If the designated process subsequently exits, the **SIGIO** signal is not sent to any process.

The default on open of a terminal device file is that the process performing the first open is set to receive the **SIGIO** signals.

FIGO_SAIoOwn The integer referenced by *arg* is set to the process ID designated to receive **SIGIO** signals.

Line Control IOCTL Commands

Several **ioctl()** system calls control input and output. Some of these calls have the form:

```
ioctl (fildes, command, arg)
int arg;
```

Commands using this form are:

TCsBRK Wait for the output to drain. If *arg* is 0, send a break (zero bits for at least 0.25 seconds). The **tcsendbreak()** function performs the same function (see *tcsendbreak(3C)*).

TCXONC Start/stop control. If *arg* is 0, suspend output; if 1, restart suspended output; if 2, transmit a STOP character; if 3, transmit a START character. If any other value is given for *arg*, the call returns -1 with **errno** set to [EINVAL]. The **tcflow()** function performs the same functions (see *tcflow(3C)*).

TCFLSH If *arg* is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues. If any other value is given for *arg*, the call returns -1 with **errno** set to [EINVAL]. The **tcflush()** function performs the same functions (see *tcflush(3C)*).

Sending a BREAK is accomplished by holding the data transmit line at a SPACE or logical zero condition for at least 0.25 seconds. During this interval, data can be sent to the device, but because of serial data interface limitations, the BREAK takes precedence over all data. Thus, all data sent to a device during a BREAK is lost. This includes system-generated XON/XOFF characters used for input flow control. Note also that a delay in transmission of the XOFF flow control character until after the BREAK is terminated could still result in data overflow because the flow control character may not be sent soon enough.

Other calls have the form:

```
ioctl (fildes, command, arg)
int *arg;
```

Commands using this form are:

FIONREAD Returns in the integer referenced by *arg* the number of characters immediately readable from the terminal device file. This command is allowed from a background process; however, the data itself cannot be read from a background process.

Non-blocking I/O IOCTL Commands

Non-blocking I/O is easily provided via the **O_NONBLOCK** and **O_NDELAY** flags available in both *open(2)* and *fcntl(2)*. The commands in this section are provided for backward compatibility with previously developed applications. **ioctl()** system calls that provide a style of non-blocking I/O different from **O_NONBLOCK** and **O_NDELAY** have the form:

```
ioctl (fildes, command, arg)
int *arg;
```

Commands using this form are:

FIOSNBIO If the integer referenced by *arg* is non-zero, **FIOSNBIO**-style non-blocking I/O is enabled; that is, subsequent reads and writes to the terminal device file are handled in a non-blocking manner (see below). If the integer referenced by *arg* is 0, **FIOSNBIO**-style non-blocking I/O is disabled.

For reads, **FIOSNBIO**-style non-blocking I/O prevents all read requests to that device file from blocking, whether the requests succeed or fail. Such a read request completes in one of three ways:

- If there is enough data available to satisfy the entire request, the read completes successfully, having read all of the data, and returns the number of characters read;
- If there is not enough data available to satisfy the entire request, the read completes successfully, having read as much data as possible, and returns the number of characters read;
- If there is no data available, the read returns `-1` with **errno** set to [EWOULDBLOCK].

For writes, **FIOSNBIO**-style non-blocking I/O prevents all write requests to that device file from blocking, whether the requests succeed or fail. Such a write request completes in one of three ways:

- If there is enough space available in the system to buffer all the data, the write completes successfully, having written out all of the data, and returns the number of characters written;
- If there is not enough space in the buffer to write out the entire request, the write completes successfully, having written as much data as possible, and returns the number of characters written;
- If there is no space in the buffer, the write returns `-1` with **errno** set to [EWOULDBLOCK].

To prohibit **FIOSNBIO**-style non-blocking I/O from interfering with the **O_NONBLOCK** and **O_NDELAY** flags (see *open(2)* and *fcntl(2)*), the functionality of **O_NONBLOCK** and **O_NDELAY** always supersedes the functionality of **FIOSNBIO**-style non-blocking I/O. This means that if either **O_NONBLOCK** or **O_NDELAY** is set, the driver performs read requests in accordance with the definition of **O_NDELAY** or **O_NONBLOCK**. When both **O_NONBLOCK** and **O_NDELAY** are clear, the definition of **FIOSNBIO**-style non-blocking I/O applies.

The default on open of a terminal device file is that **FIOSNBIO**-style non-blocking I/O is disabled.

FIOSNBIO The integer referenced by *arg* is set to 1, if **FIOSNBIO**-style non-blocking I/O is enabled. Otherwise, the integer referenced by *arg* is set to 0.

Process Group Control IOCTL Commands

The process group control features described here (except for setting and getting the delayed stop process character) are easily implemented using the functions **tcgetattr()**, **tcsetattr()**, **tcgetpgrp()**, **tcsetpgrp()**, and **tcsetsid()**, (see *tcattribute(3C)*, *tcgetpgrp(3C)*, *tcsetpgrp(3C)*, and *tcsetsid(3C)*)

respectively).

The following structure, used with process group control, is defined in `<bsdtty.h>`:

```
struct ltchars {
    unsigned char t_suspc; /* stop process character*/
    unsigned char t_dsuspc; /* delayed stop process character*/
    unsigned char t_rprntc; /* reserved; must be '_POSIX_VDISABLE'*/
    unsigned char t_flushc; /* reserved; must be '_POSIX_VDISABLE'*/
    unsigned char t_werasc; /* reserved; must be '_POSIX_VDISABLE'*/
    unsigned char t_inxctc; /* reserved; must be '_POSIX_VDISABLE'*/
};
```

The initial value for all these characters is `_POSIX_VDISABLE`, which causes them to be disabled. The meaning for each character is as follows:

- t_suspc** Suspend the foreground process group. A *suspend* signal (**SIGTSTP**) is sent to all processes in the foreground process group. Normally, each process is forced to stop, but arrangements can be made to either ignore or block the signal, or to receive a trap to an agreed-upon location; see *signal(2)* and *signal(5)*. When enabled, the typical value for this character is Control-Z or ASCII SUB. Setting or getting **t_suspc** is equivalent to setting or getting the SUSP special control character.
- t_dsuspc** Same as **t_suspc**, except that the *suspend* signal (**SIGTSTP**) is sent when a process reads the character, rather than when the character is typed. When enabled, the typical value for this character is Control-Y or ASCII EM.

Attempts to set any of the reserved characters to a value other than `_POSIX_VDISABLE` cause `ioctl()` to return `-1` with `errno` set to `[EINVAL]` with no change in value of the reserved character.

`ioctl()` system calls that use the above structure have the form:

```
ioctl (fildes, command, arg)
struct ltchars *arg;
```

Commands using this form are:

- TIOCGLTC** Get the process group control characters and store them in the *ltchars* structure referenced by *arg*. This command is allowed from a background process. However, the information may be subsequently changed by a foreground process.
- TIOCSLTC** Set the process group control characters from the structure referenced by *arg*.

Additional process group control `ioctl()` system calls have the form:

```
ioctl (fildes, command, arg)
unsigned int *arg;
```

Commands using this form are:

- TIOCGPRP** Returns in the integer referenced by *arg* the foreground process group associated with the terminal. This command is allowed from a background process. However, the information may be subsequently changed by a foreground process. This feature is easily implemented using the `tcgetpgrp()` function (see *tcgetpgrp(3C)*).

If the `ioctl()` call fails, it returns `-1` and sets `errno` to one of the following values:

- [EBADF] *fildes* is not a valid file descriptor.
- [ENOTTY] The file associated with *fildes* is not the controlling terminal, or the calling process does not have a controlling terminal.
- [EACCES] The file associated with *fildes* is the controlling terminal of the calling process, however, there is no foreground process group defined for the controlling terminal.

Note: [EACCES] may not be returned in future releases. Behavior in cases where no foreground process group is defined for the controlling terminal may change in future versions of the POSIX standard. Portable applications, therefore, should not rely on this error condition.

TIOCSPGRP Sets the foreground process group associated with the terminal to the value referenced by *arg*. This feature is easily implemented using the **tcsetpgrp()** function (see *tcsetpgrp(3C)*).

If the **ioctl()** call fails, it returns **-1** and sets **errno** to one of the following values:

- [EBADF] *fildev* is not a valid file descriptor.
- [EINVAL] The process ID referenced by *arg* is not a supported value.
- [ENOTTY] The calling process does not have a controlling terminal, or the *fildev* is not the controlling terminal, or the controlling terminal is no longer associated with the session of the calling process.
- [EPERM] The process ID referenced by *arg* is a supported value but does not match the process group ID of a process in the same session as the calling process.

TIOCGSID Returns in the integer referenced by *arg* the session ID of the terminal specified by *fildev*. This feature is easily implemented using the **tcgetsid()** function (see *tcgetsid(3C)*).

If the **ioctl()** call fails, it returns **-1** and sets **errno** to one of the following values:

- [EBADF] *fildev* is not a valid file descriptor.
- [ENOTTY] The device associated with *fildev* is not a terminal.
- [EACCES] The *fildev* is a terminal that is not allocated to a session.

TIOCLGET Get the process group control mode word and store it in the int referenced by *arg*. This command is allowed from a background process; however, the information may be subsequently changed by a foreground process.

TIOCLSET Set the process group control mode word to the value of the int referenced by *arg*.

TIOCLBIS Use the int referenced by *arg* as a mask of bits to set in the process group control mode word.

TIOCLBIC Use the int referenced by *arg* as a mask of bits to clear in the process group control mode word.

The following bit is defined in the process group control mode word:

LTOSTOP Send **SIGTTOU** for background writes.

Setting or clearing **LTOSTOP** is equivalent to setting or clearing the **TOSTOP** flag (see *Local Modes*). If **LTOSTOP** is set and a process is not in the foreground process group of its controlling terminal, a write by the process to its controlling terminal may be denied (see *Terminal Access Control*).

Terminal Size IOCTL Commands

The following **ioctl()** system calls are used to get and set terminal size information for the terminal referenced by *fildev*. These **ioctl()** system calls use the **winsize** structure to get and set the terminal size information. The **winsize** structure, defined in **<termios.h>**, has the following members :

```
unsigned short ws_row;      /* Rows, in characters */
unsigned short ws_col;      /* Columns, in characters */
unsigned short ws_xpixel;   /* Horizontal size, in pixels */
unsigned short ws_ypixel;   /* Vertical size, in pixels */
```

The initial values for all elements of terminal size are zero. The values for terminal size are neither set nor used by the general terminal interface, and have no effect on the functionality of the general terminal interface. The values for terminal size are set and used only by applications that access them through the terminal-size **ioctl()** system calls (see *ioctl(2)*).

ioctl() system calls that use the above structure have the form:

```
ioctl (fildev, command, arg)
struct winsize *arg;
```

Commands using this form are:

- TIOCGWINSZ** Get the terminal size values and store them in the **winsize** structure referenced by *arg*. This command is allowed from a background process.
- TIOCSWINSZ** Set the terminal size values from the **winsize** structure referenced by *arg*. If any of the new values differ from previous values, a **SIGWINCH** signal is sent to all processes in the terminal's foreground process group.

Console Output Redirection IOCTL Command

Output which would normally be sent to the system console may be redirected to any other TTY device or pseudo-device in the system. The **ioctl()** system call used to control console output redirection has the form:

```
ioctl (fildes, command, arg)
int arg;
```

The command using this form is:

- TIOCCONS** Redirect system console output. Any output that would normally be sent to the system console, either through kernel printf requests, or through the console special file, will instead be sent to the terminal referenced by *fildes*. The value of *arg* is ignored. The user must have the **DEVOPS** privilege to execute this request. Otherwise, the call returns **-1** with **errno** set to **[EPERM]**. If the console output has not been redirected to a different device by a later call to this command, it is redirected back to the physical console device when *fildes* is closed.

WARNINGS

Various HP-UX implementations use non-serial interfaces that look like terminals (such as bit-mapped graphics displays) or "smart cards" that cannot implement the exact capabilities described above. Therefore, not all systems can exactly meet the standard stated above. Each implementation is required to state any deviations from the standard as part of its system-specific documentation.

- FIOSSAIOSTAT** is similar to BSD 4.2 **FIOASYNC**, with the addition of provisions for security.
- FIOGSAIOSTAT** is of HP origin, complements **FIOSSAIOSTAT**, and allows saving and restoring system asynchronous I/O TTY states for command interpreter processes.
- FIOSSAIOOWN** is similar to BSD 4.2 **FIOSETOWN**, with additional provisions for security.
- FIOGSAIOOWN** is similar to BSD **FIOGETOWN**. 4.2 Note also the difference that the BSD 4.2 version of this functionality used process groups, while the HP-UX version only uses processes.
- FIOSNBIO** is the same as BSD **FIONBIO**, 4.2 except that it does not interfere with the **O_NDELAY** or **O_NONBLOCK** **open()** and **fcntl()** flags.
- FIOGNBIO** is of HP origin, complements **FIOSNBIO**, and allows saving and restoring the **FIOSNBIO**-style non-blocking I/O TTY state for command interpreter processes.

The general terminal interface uses a system resource known as a **cblock** to store data being transmitted or received through a communications port. These cblocks are continuously used and freed for reuse as data pass through the system. If too few cblocks are configured in the system, the cblock pool may be temporarily or permanently exhausted, and data loss, system hangs, or reduced system performance can result.

If cblock exhaustion is suspected, you can examine the system message buffer with **dmesg** (see **dmesg(1M)**) for messages indicating cblock exhaustion has occurred. Or, you can use **adb** (see **adb(1)**) if examining the corefile of a dump. The message format is

WARNING: cblock exhaustion occurred n times

where *n* indicates the number of times the operating system has requested a cblock and none could be provided. If this message is observed, the kernel should be reconfigured to generate a larger number of cblocks.

A cblock is 32 bytes in length. The default number of cblocks configured in the system is defined to be 8292. This can be overridden by using the optional tunable system parameter **nclist** to specify the desired number of cblocks to be used in the system.

SAM or *kctune*(1M) may be used to change the **nclist** value.

DEPENDENCIES

Workstations

Built-in serial ports on workstation machines support the following additional baud rate settings: 57 600, and 115 200. An RS-232-to-RS-422 converter may be required to achieve practical cable lengths at these baud rates (because RS-232 only specifies up to 19 200 baud).

Timed delays are not supported.

Built-in serial ports on workstation systems have RTS and CTS flow control capability, configurable receive FIFO trigger levels, and a configurable transmit limit. RTS/CTS hardware handshaking can be enabled through a bit in the device file minor number, through an **ioctl**() call (see *termiox*(7)), or through the **stty** command (see *stty*(1)).

The receive FIFO trigger level is configurable through two bits in the device file minor number. The receive FIFO trigger level is used to set the level at which a receive interrupt is generated to the system. Setting a smaller value for the receive FIFO trigger level enables the system to react more quickly to receipt of characters. However, using a smaller trigger level increases system overhead to process the additional interrupts. A higher receive FIFO trigger level reduces the system interrupt overhead for heavy inbound data traffic at the cost of less time for the system to read data from the hardware before receive FIFOs are overrun. When using RTS flow control, the receive FIFO trigger level also determines the point at which the hardware lowers RTS to protect the receive FIFO. Use of a higher receive FIFO trigger level also reduces XOFF flow control responsiveness because, under light inbound data flow conditions, receipt of the XOFF character by the system is slightly delayed. Choice of the appropriate receive FIFO trigger level should be based upon how the serial port is to be used. For most applications a receive FIFO trigger level of 8 (c3,c2 = 10) is suggested.

Two bits in the device file minor number specify the transmit limit, the number of characters which are successively loaded into the transmit FIFO. Setting a smaller transmit limit allows the transmitter to be more responsive to flow control either from receipt of an XOFF character or de-assertion of CTS at the cost of increased system interrupt overhead. Setting a larger transmit limit reduces interrupt overhead but is not as responsive to flow control since the remainder of the transmit FIFO can be transmitted even after the transmitter is flow controlled. When communicating with devices which have little tolerance for data receipt after flow control, one must choose the transmit limit appropriately.

Device File Minor Number

Workstation device file minor numbers take the form:

0xIIc0H

where:

- II** = Two hexadecimal digits (8 bits) to indicate the instance of the serial interface.
- C** = One hexadecimal digit (4 bits) for FIFO control. Values for each bit are as follows:

Receive FIFO Trigger Level			Transmit Limit		
c3	c2	Level	c1	c0	Limit
0	0	1	0	0	1
0	1	4	0	1	4
1	0	8	1	0	8
1	1	14	1	1	12

- H** = One hexadecimal digit (4 bits) which controls diagnostic access and hardware flow control.

Bit	Value
h3	Diagnostic telephony access
h2	Reserved
h1	Reserved
h0	Enables RTS/CTS hardware flow control

- M** = One hexadecimal digit (4 bits) to determine the port access type. Values for each bit are as follows:

Bit	Value
m3	TI/ALP
m2	0 = Simple protocol (U.S.), 1 = CCITT protocol (Europe)
m1m0	00 = Direct 01 = Dial-out modem 10 = Dial-in modem 11 = Invalid

Servers

Timed output delays are not directly supported. If used, an appropriate number of fill characters (based on the current baud rate) is output. The total time to output the fill characters is at least as long as the time requested.

The system specified input flow control values are as follows: low water mark is 60, high water mark is 180, and maximum allowed input is 512.

The HP 98196A (formerly 27140A option 800) interface does not support the following hardware settings:

CBAUD B200, B38400, EXTA, EXTB.

The HP A1703-60003 and the HP 28639-60001 interfaces do not support baud rates above 9600. Furthermore, changing the following hardware settings on port 0 from the default (9600 baud, 8 bit characters, 1 stop bit, no parity) is not supported:

CBAUD, CSIZE, CSTOPB, PARENB, PARODD.

The HP J2094A interface does not support baud rates above 19200.

The HP J2094A supports RTS and CTS flow control. The RTS/CTS hardware handshaking can be enabled through a bit in the device file minor number, through an `ioctl()` call (see *termiox(7)*), or through the `stty` command (see *stty(1)*).

Device File Minor Number

Server device file minor numbers take the form:

0xIIPPHM

where:

- II* = Two hexadecimal digits (8 bits) to indicate the instance of the serial interface.
- PP* = Two hexadecimal digits (8 bits) to indicate the port number of this device on the serial interface.
- H* = One hexadecimal digit (4 bits) which controls diagnostic access and hardware flow control (HP J2094A only).

Bit	Value
h3	Card diagnostic
h2	Port diagnostic
h1	Reserved
h0	Enables RTS/CTS hardware flow control

- M* = One hexadecimal digit (4 bits) for the port access type. Values for each bit are as follows:

Bit	Value
m3	TI/ALP
m2	0 = Simple protocol (U.S.), 1 = CCITT protocol (Europe)
m1m0	00 = Direct 01 = Dial-out modem 10 = Dial-in modem 11 = Invalid

AUTHOR

termios was developed by HP and the IEEE Computer Society.

termio was developed by HP, AT&T, and the University of California, Berkeley.

FILES

/dev/console
/dev/cua*
/dev/cul*
/dev/tty*
/dev/ttyd*

SEE ALSO

adb(1), shl(1), stty(1), dmesg(1M), kctune(1M), mknod(1M), fork(2), ioctl(2), setpgid(2), setsid(2), signal(2), stty(2), cfspeed(3C), tcattribute(3C), tccontrol(3C), tegetpgrp(3C), tcgetsid(3C), tcsetpgrp(3C), privileges(5), signal(5), unistd(5), modem(7), sttyV6(7), termiox(7), tty(7).

STANDARDS CONFORMANCE

termio: SVID2, SVID3, XPG2

termios: AES, SVID3, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

termiox - extended general terminal interface

SYNOPSIS

```
#include <sys/termiox.h>

ioctl (int fildes, int request, struct termiox * arg)
```

DESCRIPTION

The extended general terminal interface supplements the *termio(7)* general terminal interface by adding support for asynchronous hardware flow control and local implementations of additional asynchronous features. Some systems may not support all of these capabilities because of hardware or software limitations. Other systems may not permit certain functions to be disabled. In such cases, the appropriate bits are ignored. If the capabilities can be supported, the interface described here must be used.

Hardware Flow Control Modes

Hardware flow control supplements the *termio IXON*, *IXOFF*, and *IXANY* character flow control (see *termio(7)*). Character flow control occurs when one device controls the data transfer of another device by inserting control characters in the data stream between devices. Hardware flow control occurs when one device controls the data transfer of another device by using electrical control signals on wires (circuits) of the asynchronous interface. Character flow control and hardware flow control can be simultaneously set.

In asynchronous, full duplex applications, the use of the Electronics Industries Association's EIA-232-D Request To Send (RTS) and Clear To Send (CTS) circuits is the preferred method of hardware flow control.

The EIA-232-D standard specified only unidirectional hardware flow control where the Data Circuit-terminating Equipment or Data Communications Equipment (DCE) indicates to the Data Terminal Equipment (DTE) to stop transmitting data. The *termiox* interface allows both unidirectional and bidirectional hardware flow control; when bidirectional flow control is enabled, either the DCE or DTE can indicate to each other to stop transmitting data across the interface.

Clock Modes

Isochronous flow control and clock mode communication are not supported.

Terminal Parameters

Parameters that control the behavior of devices providing the *termiox* interface are specified by the **termiox** structure, defined in the `<sys/termiox.h>` header file. Several `ioctl()` system calls (see *ioctl(5)*) that fetch or change these parameters use the **termiox** structure which contains the following members:

```
unsigned short x_hflag;      /* hardware flow control modes */
unsigned short x_cflag;      /* clock modes */
unsigned short x_rflag;      /* reserved modes */
unsigned short x_sflag;      /* spare local modes */
```

The **x_hflag** field describes hardware flow control modes:

```
RTSXOFF  0000001  Enable RTS hardware flow control on input.
CTSXON   0000002  Enable CTS hardware flow control on input.
```

The RTS and CTS circuits are involved in establishing CCITT modem connections. Since RTS and CTS circuits are used both by CCITT modem connections and by hardware flow control, CCITT modem and hardware flow control cannot be simultaneously enabled.

Variations of different hardware flow control methods can be selected by setting the appropriate bits. For example, bidirectional RTS/CTS flow control is selected by setting both the **RTSXOFF** and **CTSXON** bits. Unidirectional CTS hardware flow control is selected by setting only the **CTSXON** bit.

If **RTSXOFF** is set, the Request to Send (RTS) circuit (line) is raised, and if the asynchronous port needs to have its input stopped, it lowers the Request to Send (RTS) line. If the RTS line is lowered, it is assumed that the connected device will stop its output until RTS is raised.

If **CTSXON** is set, output occurs only if the Clear To Send (CTS) circuit (line) is raised by the connected device. If the CTS line is lowered by the connected device, output is suspended until CTS is raised.

(HP-PB Only)

termiox Structure Related IOCTL Command

The `ioctl()` system calls that reference the `termiox` structure have the form:

```
ioctl (fildes, command, arg)
struct termiox *arg;
```

Commands using this form are:

- | | |
|----------------|---|
| TCGETX | The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are fetched and stored into that structure. |
| TCSETX | The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are set from the values stored in that structure. The change is immediate. Errors that can be returned include: <ul style="list-style-type: none"> [EINVAL] The port does not support hardware flow control. [ENOTTY] The file descriptor for this port is configured for CCITT mode access. Hardware flow control is not allowed on CCITT mode devices. |
| TCSETXW | The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted. This form should be used when changing parameters that affect output. Errors that can be returned include: <ul style="list-style-type: none"> [EINVAL] The port does not support hardware flow control. [ENOTTY] The file descriptor for this port is configured for CCITT mode access. Hardware flow control is not allowed on CCITT mode devices. |
| TCSETXF | The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted; all characters queued for input are discarded, then the change occurs. Errors that can be returned include: <ul style="list-style-type: none"> [EINVAL] The port does not support hardware flow control. [ENOTTY] The file descriptor for this port is configured for CCITT mode access. Hardware flow control is not allowed on CCITT mode devices. |

AUTHOR

`termiox` was developed by HP and AT&T.

FILES

Files in or under `/dev/tty*`.

SEE ALSO

`ioctl(2)`, `termio(7)`, `modem(7)`.

NAME

timod - STREAMS module for converting ioctl() calls into Transport Interface messages

DESCRIPTION

The **timod** module is a STREAMS module that converts **ioctl()** calls from a transport user supporting the Transport Interface (TI) into messages that a transport protocol provider supporting TI can consume. This allows the user to initiate certain TI functions as atomic operations. This release of HP-UX no longer automatically pushes **timod** whenever a **t_open(3)** is performed. The TLI and XTI libraries have been modified to no longer require this module to perform the atomic operations described within this man page. Binary compatibility is not a problem since the module will still exist within the kernel. But, any application which is recompiled and expects the module to be automatically pushed, may not work without code modification.

The user places and removes the **timod** module on a device stream by calling the STREAMS **I_PUSH ioctl()** and **I_POP ioctl()** functions. (The TLI function **t_open()** pushes **timod** onto the device stream for the user.) The **timod** module should only be pushed onto streams which are terminated by transport providers which conform to the Transport Interface. **tirdwr(7)** is an alternative interface to **timod** which supports the **read()** and **write()** system calls. If **tirdwr** has been pushed onto the stream, the user should use the **I_POP ioctl** to remove the **tirdwr** module from the stream before pushing **timod**.

The **timod** module transparently passes any STREAMS messages that are not generated by the **ioctl()** commands described below to the neighboring module or driver. **timod** will act on an **I_STR ioctl()** whose **strioc1.ic_cmd** field is one of the values below. (See **streamio(7)** for a description of the **I_STR ioctl** and the **strioc1** structure.)

TI_BIND This TI command binds an address to the transport protocol provider. The STREAMS message that the module issues to the **TI_BIND ioctl()** call is equivalent to the TI message type **T_bind_req**. The STREAMS message that the module returns in response to the successful completion of the **TI_BIND ioctl()** call is equivalent to the TI message type **T_bind_ack**.

TI_UNBIND This TI command unbinds an address from the transport protocol provider. The STREAMS message that the module issues to the **TI_UNBIND ioctl()** call is equivalent to the TI message type **T_unbind_req**. The STREAMS message that the module returns in response to the successful completion of the **TI_UNBIND ioctl()** call is equivalent to the TI message type **T_ok_ack**.

TI_GETINFO This TI command gets the TI protocol-specific information from the transport protocol provider. The STREAMS message that the module issues to the **TI_GETINFO ioctl()** call is equivalent to the TI message type **T_info_req**. The STREAMS message that the module returns in response to the successful completion of the **TI_GETINFO ioctl()** call is equivalent to the TI message type **T_info_ack**.

TI_OPTMGMT This TI command gets, sets, or negotiates TI protocol-specific options with the transport protocol provider. The STREAMS message that the module issues to the **TI_OPTMGMT ioctl()** call is equivalent to the TI message type **T_optmgmt_req**. The STREAMS message that the module returns in response to the successful completion of the **TI_OPTMGMT ioctl()** call is equivalent to the TI message type **T_optmgmt_ack**.

RETURN VALUES

If the **timod** module returns an error for an **ioctl()** call, the lower 8 bits of the return value will be one of the TI error codes defined in the **<tiuser.h>** header file. If the TI error is of the type TSYERR, then the second 8 bits of the return value will contain an error as defined in the **<errno.h>** header file. The STREAMS message that the module issues when an **ioctl()** call results in an error is equivalent to the TI message type **T_error_ack**.

FILES

<xti.h> defines the error codes for XTI functions.
<tiuser.h> defines the error codes for TI functions.
<tihdr.h> defines the message types for TI functions.
<errno.h> defines the error codes for system errors.

SEE ALSO

ioctl(2), t_open(3), streamio(7), tirdwr(7).



t

NAME

tirdwr - STREAMS module for reads and writes by Transport Interface users

DESCRIPTION

The **tirdwr** module is a STREAMS module that provides a transport user supporting the Transport Interface (TI) with an alternate interface to a transport protocol provider supporting TI. This alternate interface allows the transport user to communicate with the transport protocol provider using the **read()** and **write()** functions. It can also continue to use the **putmsg()** and **getmsg()** functions, but these functions will only transfer data messages between the user process and device stream. **getpmsg()** and **putpmsg()** should not be used with **tirdwr**.

The user places the **tirdwr** module on a device stream by calling the STREAMS **I_PUSH ioctl()** function. **tirdwr** is an alternative interface to *timod(7)*. If **timod** has been pushed onto the stream, the user should use the **I_POP ioctl** to remove the **timod** module from the stream before pushing **tirdwr**. The **tirdwr** module should only be pushed onto streams which are terminated by transport providers which conform to the Transport Interface. Once the module has been pushed on the device stream the user cannot make further calls to TI functions. If the user attempts to do this, an error occurs on the stream. After the error is detected, subsequent calls fail with **errno** set to [EPROTO]. The user removes the **tirdwr** module from a device stream by calling the STREAMS **I_POP ioctl()** function.

Module Behavior When Pushed and Popped

When the **tirdwr** module is pushed on a device stream, it checks any existing messages that are destined for the user to determine their message type. If existing messages are regular data messages, it forwards the messages to the user. It ignores any messages related to process management, such as messages that generate signals to the user. If any other messages are present, it returns an error to the user request with **errno** set to [EPROTO].

When the **tirdwr** module is popped from a device stream, it checks whether an orderly release indication has been previously received from the transport protocol provider. If an orderly release indication was received, it sends an orderly release request to the remote side of the transport connection. The **tirdwr** module also acts this way when the device stream is closed.

Module Behavior for Reads and Writes

When the **tirdwr** module receives messages from the transport protocol provider that do not contain a control part (see the *putmsg(2)* and *getmsg(2)* reference pages), it transparently passes the messages to its upstream neighbor. The exception is for zero-length data messages, where the module frees the message and does not pass them to its upstream neighbor.

When the module receives messages from the transport protocol provider that contain a control part, it takes one of the following actions:

For data messages with a control part, it removes this part, then passes the message to its upstream neighbor.

For messages that represent expedited data, it generates an error. Further system calls will fail with **errno** set to [EPROTO].

For messages that represent an orderly release indication from the transport protocol provider, it generates a zero-length data message, indicating the End-of-File (EOF), and sends this message upstream to the reading process. The original message containing the orderly release indication is freed.

For messages that represent an abortive disconnect indication from the transport protocol provider, it causes all further **write()** and **putmsg()** calls to fail with **errno** set to [ENXIO]. Subsequent **read()** and **getmsg()** calls will return zero-length data messages indicating the End-of-File (EOF), once all previous data has been read.

For all other messages, it generates an error, and further calls will fail with **errno** set to [EPROTO].

SEE ALSO

getmsg(2), *putmsg(2)*, *read(2)*, *write(2)*, *t_open(3)*, *streamio(7)*, *timod(7)*.

NAME

tty - controlling terminal interface

DESCRIPTION

The file `/dev/tty` is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that need to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired, and it is tiresome to find out what terminal is currently in use.

FILES

`/dev/tty`

`/dev/tty*`

SEE ALSO

`termio(7)`.

STANDARDS CONFORMANCE

`tty`: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

UDP - Internet User Datagram Protocol

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_DGRAM, 0);
s = socket(AF_INET6, SOCK_DGRAM, 0);
```

DESCRIPTION

UDP is a simple, unreliable datagram protocol used to support the **SOCK_DGRAM** socket type for the internet protocol family. UDP sockets are connectionless, and are normally used with the **sendto()** and **recvfrom()** calls (see *send(2)* and *recv(2)*). The **connect()** call can also be used to simulate a connection (see *connect(2)*). When used in this manner, it fixes the destination for future transmitted packets (in which case the **send()** or **write()** system calls can be used), as well as designating the source from which packets are received. The **recv()** and **read()** calls can be used at any time if the source of the message is unimportant.

UDP address formats are identical to those used by TCP. In particular, UDP requires a port identifier in addition to the normal Internet address format. Note that the UDP port domain is separate from the TCP port domain (in other words, a UDP port cannot be connected to a TCP port).

The default send buffer size for UDP sockets is 65535 bytes. The default receive buffer size for UDP sockets is 2147483647 bytes. The send and receive buffer sizes for UDP sockets can be set by using the **SO_SNDBUF** and **SO_RCVBUF** options of the **setsockopt()** system call or the **XTI_SNDBUF** and **XTI_RCVBUF** options of the **t_optmgmt()** system call. The maximum size for these buffers is 2147483647 bytes. The maximum receive buffer size may be lowered using the **ndd** parameter **udp_rcv_hiwater_max**.

The maximum message size for a UDP datagram socket is limited by the lesser of the maximum size of an IP datagram and the size of the UDP datagram socket buffer. The maximum size of an IP datagram limits the maximum message size of a UDP message to 65507 bytes. Therefore, using the maximum socket buffer size will allow multiple maximum-sized messages to be placed on the send queue. The default inbound and outbound message size limit for a UDP datagram socket is 65535 bytes.

The maximum message size for a UDP broadcast is limited by the MTU size of the underlying link.

ERRORS

One of the following errors may be returned in **errno** if a socket operation fails. For a more detailed list of errors, see the man pages for specific system calls.

[EISCONN]	Attempt to send a datagram with the destination address specified, when the socket is already connected.
[ENOBUFS]	No buffer space is available for an internal data structure.
[EADDRINUSE]	Attempt to create a socket with a port which has already been allocated.
[EADDRNOTAVAIL]	Attempt to create a socket with a network address for which no network interface exists.

AUTHOR

The socket interfaces to UDP were developed by the University of California, Berkeley.

SEE ALSO

ndd(1M), getsockopt(2), recv(2), send(2), socket(2), t_open(3), t_optmgmt(3), inet(7F), socket(7),

RFC 768 User Datagram Protocol

RFC 1122 Requirements for Internet hosts

NAME

UNIX - local communication domain protocol

SYNOPSIS

```
#include <sys/types.h>
#include <sys/un.h>
```

DESCRIPTION

The local communication domain protocol, commonly referred to in the industry as the **Unix domain protocol**, utilizes the path name address format and the **AF_UNIX** address family. This protocol can be used as an alternative to the Internet protocol family (TCP/IP or UDP/IP) for communication between processes executing on the same node. It has a significant throughput advantage when compared with local IP loop-back, due primarily to its much lower code execution overhead. Data is looped back at the protocol layer (OSI Level 4), rather than at the driver layer (OSI Level 2).

Only **SOCK_STREAM** is supported in the **AF_UNIX** address family.

The HP-UX implementation of the local communication domain protocol does not support the **MSG_OOB** flag in **recv()** (see *recv(2)*) and **send()** (see *send(2)*).

Addressing

AF_UNIX socket addresses are path names. They are limited to 92 bytes in length, including a terminating null byte. Calls to **bind()** to an **AF_UNIX** socket utilize an addressing structure called **struct sockaddr_un** (see *bind(2)*). Pointers to this structure should be used in all **AF_UNIX** socket system calls wherever they require a pointer to a **struct sockaddr**.

The include file **<sys/un.h>** defines this addressing structure. Within this structure are two notable fields. The first is *sun_family*, which must be set to **AF_UNIX**. The next is *sun_path*, which is the null-terminated character string that specifies the path name of the file associated with the socket (for example, */tmp/mysocket*).

Only the passive (listening) socket must bind to an address. The active socket connects to that address, but it does not need an address of its own.

For additional information on using **AF_UNIX** sockets for interprocess communication, refer to the BSD Sockets Interface Programmer's Guide.

Socket Buffer Size

For stream and datagram sockets, the maximum send and receive buffer size is 262142 bytes. The default buffer size is 32768 bytes. The send and receive buffer sizes can be altered by using the **SO_SNDBUF** and **SO_RCVBUF** options of the **setsockopt()** system call. Refer to *getsockopt(2)* for details.

AUTHOR

AF_UNIX was developed by the University of California, Berkeley.

SEE ALSO

getsockopt(2), *socket(2)*.

NAME

VLAN - virtual local area network

DESCRIPTION

This manpage provides a brief overview of VLAN (virtual LAN) technology.

VLANs are logical, or **virtual**, network segments that can span multiple physical network segments. A primary benefit of VLANs is that they can isolate broadcast and multicast traffic by determining which destinations should receive that traffic, thereby making better use of switch and end-station resources.

Logical separation using VLAN allows for the logical grouping of PCs, servers and other network resources to behave as if they were connected to the same, physical segment, even if they are not.

HP-UX VLAN is an implementation of IEEE 802.1p/Q standards.

VLAN interfaces can be configured in HP-UX servers using the command **nwmgr** (see *nwmgr_vlan(1M)*) or **lanadmin** (see *lanadmin_vlan(1M)*). HP recommends that you use **nwmgr** for HP-UX Release 11i Version 3 and forward. Interfaces can also be configured using the web-based management tool HP-UX System Management Homepage (HP SMH).

Each VLAN interface created is assigned a VLAN PPA (VPPA) that is unique across the system and a VLAN ID, that identifies the virtual LAN it is part of. The VLAN ID is unique on the interface on which the VLAN interface is created.

WARNINGS

The **lanadmin**, **lanscan**, and **linkloop** commands are deprecated. These commands will be removed in a future HP-UX release. HP recommends the use of replacement command *nwmgr(1M)* to perform all network interface-related tasks.

SEE ALSO

lanadmin(1M), *lanadmin_vlan(1M)*, *lanscan(1M)*, *nwmgr(1M)*, *nwmgr_vlan(1M)*, *smh(1M)*.

HP-UX VLAN Administrator's Guide

IEEE 802.1p, IEEE 802.1Q

NAME

xopen_networking - X/Open Networking Interfaces

DESCRIPTION

X/Open has defined **Sockets** and **IP Address Resolution** interfaces in *X/Open CAE Specification, Networking Services, Issue 4* (UNIX 95), *X/Open CAE Specification, Networking Services, Issue 5* (UNIX 98), and *The Single UNIX Specification, Version 3, System Interfaces* (UNIX 03).

X/Open has also defined XTI in *X/Open CAE Specification, Networking Services, Issue 4* (UNIX 95) and *X/Open CAE Specification, Networking Services, Issue 5* (UNIX 98). Beginning in UNIX 03, XTI is no longer part of *The Single UNIX Specification*.

For more information on the specifications or a detailed description of the X/Open Networking Interfaces, please refer to the above specifications at **The Open Group** website, <http://www.opengroup.org>.

Prior to HP-UX 11i v3, HP-UX is certified to UNIX 95 on PA-RISC and Integrity systems. Beginning with HP-UX 11i v3, HP-UX is certified to UNIX 95 on PA-RISC systems and to UNIX 95 and UNIX 03 on Integrity systems.

COMPILATION ENVIRONMENT

There are two ways to obtain X/Open Sockets functionality:

Method A is in compliance with X/Open compilation specification.

Method B slightly deviates from X/Open compilation specification. However, Method B allows a program to include both objects compiled to X/Open Sockets specification and objects compiled to BSD Sockets specification.

Either **cc**, **c89** or **c99** utilities can be used. Refer to *cc(1)* for details. Also note certain features in **UNIX 03** are only available if **c99** is used. For example, the "restrict" qualifier for pointers is only available if **c99** is used.

Method A) Strict Compliance Method

An X/Open conforming application is one that has all its parts compiled and built according to X/Open specifications. For such conforming applications, this compilation method would be appropriate.

Compilation**UNIX 03**

Applications should ensure that the feature test macro **_XOPEN_SOURCE** is defined with the value **600**. To ensure portability, applications should define the macro either on the compilation command line, or at the beginning of each source module prior to the inclusion of any headers.

For example, to compile a 64 bit object using **HP ANSI Compiler**:

```
c99 +DD64 -D_XOPEN_SOURCE=600 -c main.c -o main.o
c99 +DD64 -D_XOPEN_SOURCE=600 -c routines.c -o routines.o
```

UNIX 95

Applications should ensure that the feature test macros **_XOPEN_SOURCE** and **_XOPEN_SOURCE_EXTENDED** are defined. To ensure portability, applications should define the macros either on the compilation command line, or at the beginning of each source module prior to the inclusion of any headers.

For example, to compile a 64 bit object using **HP ANSI Compiler**:

```
c89 +DD64 -D_XOPEN_SOURCE -D_XOPEN_SOURCE_EXTENDED -c main.c -o
main.o
c89 +DD64 -D_XOPEN_SOURCE -D_XOPEN_SOURCE_EXTENDED -c routines.c -o
routines.o
```

Linkage

Link the program objects with **xnet** library.

For example:

```
ld main.o routines.o -lxnet -lc -o prog
```

Note if the **C** library is also specified in the link line, the **Xnet** library has to be specified before the **C** library. Otherwise, X/Open Sockets calls would have been resolved to BSD Sockets functions in the **C** library instead of X/Open Sockets functions in the **Xnet** library.

Method B) Alternative Method

HP-UX provides two styles of Sockets API:

- default BSD Sockets
- X/Open Sockets

These two styles of Sockets API have the same function names but they have differences in semantics and argument types. For example, the *optlen* field in X/Open `getsockopt()` is `size_t` type, while BSD `getsockopt()` is `int` type. In 64 bit mode, `size_t` is 64 bit and `int` is still 32 bit.

Linking objects compiled to X/Open Sockets specification and objects compiled to BSD Sockets specification in the same program using the linkage method in method A would erroneously resolve BSD Sockets calls to X/Open Sockets functions in the **Xnet** library. As a result, the program may result in application core dumps or unexpected Socket errors when it is run. These symptoms commonly occur when BSD Sockets `accept()`, `getpeername()`, `getsockname()`, `getsockopt()`, `recvfrom()`, `sendmsg()`, and `recvmsg()` are called.

For such mixed program configuration, the compilation and linkage methods described below in *Compilation* should be used.

Compilation

Define `_HPUX_ALT_XOPEN_SOCKET_API`, in addition to either defining `_XOPEN_SOURCE=600` in **UNIX 03** or `_XOPEN_SOURCE` and `_XOPEN_SOURCE_EXTENDED` in **UNIX 95**.

For example to compile a 64-bit X/Open Sockets object and a 64-bit BSD Sockets object using **HP ANSI Compiler**:

UNIX 03

```
c99 +DD64 -D_XOPEN_SOURCE=600 -D_HPUX_ALT_XOPEN_SOCKET_API -c main.c -o main.o
```

UNIX 95

```
c89 +DD64 -D_XOPEN_SOURCE -D_XOPEN_SOURCE_EXTENDED -D_HPUX_ALT_XOPEN_SOCKET_API -c main.c -o main.o
```

BSD Sockets

```
cc -Ae +DD64 -c routines.c -o routines.o
```

With this method, X/Open Sockets calls are remapped by the static Sockets functions in `<sys/socket.h>` to an alternative set of X/Open Sockets functions in *C* library. This alternative set has a prefix `_xpg_` in its function names, for example, `_xpg_getsockopt()`.

Because the alternative set has different function names, X/Open Sockets calls are not confused with BSD Sockets calls at link time.

Other than the naming difference, this alternative set is identical to the X/Open Sockets functions in *Xnet* library. Other than adding an additional macro, `_HPUX_ALT_XOPEN_SOCKET_API`, this compilation method is compliant to X/Open specifications.

Linkage

Link with *C* library instead of *Xnet* library. *Xnet* library should not be included in the application link line.

For example:

```
ld main.o routines.o -lc -o prog
```

Because *Xnet* library is not in the link line, BSD Sockets calls are not erroneously resolved to X/Open Sockets functions in *Xnet* library.

FUTURE DIRECTION

Method B might become the default method in a future release. At that time, `_HPUX_ALT_XOPEN_SOCKET_API` would be defined by default.

AUTHOR

X/Open **XTI**, **Sockets** and **IP Address Resolution** interfaces were developed by HP and X/Open Company Limited.

SEE ALSO**XTI:**

t_accept(3), t_alloc(3), t_bind(3), t_close(3), t_connect(3), t_error(3), t_free(3), t_getinfo(3), t_getprotaddr(3), t_getstate(3), t_listen(3), t_look(3), t_open(3), t_optmgmt(3), t_rcv(3), t_rcvconnect(3), t_rcvdis(3), t_rcvrel(3), t_rcvudata(3), t_rcvuderr(3), t_snd(3), t_snddis(3), t_sndrel(3), t_sndudata(3), t_strerror(3), t_sync(3), t_unbind(3).

Sockets:

accept(2), bind(2), close(2), connect(2), fcntl(2), fgetpos(3S), fsetpos(3S), ftell(3S), getpeername(2), getsockname(2), getsockopt(2), listen(2), lseek(2), poll(2), read(1), recv(2), recvfrom(2), recvmsg(2), select(2), send(2), sendmsg(2), sendto(2), setsockopt(2), shutdown(2), socketatmark(3N), socket(2), socketpair(2), write(1).

IP Address Resolution:

gethostname(2), endhostent(3N), endnetent(3N), endprotoent(3N), endservent(3N), freeaddrinfo(3N), gai_strerror(3N), getaddrinfo(3N), gethostbyaddr(3N), getnameinfo(3N), getnetbyaddr(3N), getprotobynumber(3N), getservbyport(3N), htonl(3N), if_freenameindex(3N), if_indextoname(3N), if_nameindex(3N), if_nametoindex(3N), inet_addr(3N), ntohl(3N), sethostent(3N), setnetent(3N), setprotoent(3N), setservent(3N).

NAME

zero - /dev/zero special file

DESCRIPTION

/dev/zero is a zero special file. Reads from a zero special file always return characters whose value is 0 (\0 characters).

Data written on a zero special file is discarded or ignored.

Seeks on a zero special file always succeed.

When **/dev/zero** is memory mapped by calling **mmap()**, the associated memory object behaves as a **MAP_ANONYMOUS** object. It is initialized to all zeros. Writes to the object modify the contents of the object which are observed by subsequent reads to this object.

Both **MAP_SHARED** and **MAP_PRIVATE** **mmap()** are allowed.

When it is mapped shared, the memory object can be shared only with the descendants of the current process. Modifications made to the **MAP_SHARED** object are visible only to the process and its descendants.

When it is mapped private, any modifications done after **fork()** are visible only to the process.

EXAMPLES

In the following example, the buffer **buf** is filled with **len** \0 characters.

```
fildes = open("/dev/zero",...)
read(fildes, buf, len)
```

In the following example, the process now has a range of **len** \0 characters at memory location **address**:

```
fildes = open("/dev/zero",...)
address = mmap(0, len, PROT_READ | PROT_WRITE,
              MAP_PRIVATE, fildes, any_offset)
```

FILES

/dev/zero

SEE ALSO

mmap(2), null(7).

Section 9

General Information

Section 9
General Information

NAME

intro - introduction to HP-UX general information section

DESCRIPTION

This section contains general information about HP-UX, including an introduction to HP-UX and the operating system and a glossary of common HP-UX terms.

SEE ALSO

glossary(9), introduction(9).

Web access to HP-UX documentation at <http://docs.hp.com>.

NAME

glossary - description of common HP-UX terms

DESCRIPTION

HP-UX and other UNIX-like systems use a specialized vocabulary in which certain words and terms have very specific meanings. This glossary is intended as an aid in promoting exactness in use of these specialized terms whose meanings sometimes differ from those that might be encountered in other environments. References to other HP-UX documentation are included as appropriate.

Entities in *italics* with a following parenthesized roman number (sometimes with a capital letter), such as *sh*(1), *wait*(2), or *fopen*(3S) refer to entries in the other sections of this manual. Items in **bold face** refer to other entries in this glossary. Items in **computer font** (**bold face** in the online manpages) are literals, such as file names and environment variables. Any italicized manual names refer to separate manuals that are either included with your system or available separately.

The definitions specifically reflect the HP-UX operating system, although some terms and definitions are also derived from those in the emerging IEEE POSIX standards and the *X/Open Portability Guide*. Differences in wording exist to more specifically reflect the characteristics of the HP-UX system.

GLOSSARY ENTRIES**. (dot)**

A special file name that refers to the **current directory**. It can be used alone or at the beginning of a directory path name. See also **path name resolution**. The **dot** also functions as a special command in the POSIX, Bourne, and Korn shells, and has special meaning in text editors and formatters, in parsing regular expressions and in designating file names.

.. (dot-dot)

A special file name that refers to the **parent directory**. If it begins a **path name**, **dot-dot** refers to the parent of the current directory. If it occurs in a path name, **dot-dot** refers to the parent directory of the directory preceding **dot-dot** in the path name string. As a special case, **dot-dot** refers to the current directory in any directory that has no parent (most often, the **root directory**). See also **path name resolution**.

.o (dot-oh)

The suffix customarily given to a relocatable object file. The term **dot-oh file** is sometimes used to refer to a relocatable object file. The format of such files is sometimes called **dot-oh format**. See *a.out*(4).

a.out

The name customarily given to an executable object code file on HP-UX. The format is machine-dependent, and is described in *a.out*(4) for each implementation. Object code that is not yet linked has the same format, but is referred to as a **.o (dot-oh)** file. **a.out** is also the default output file name used by the linker, *ld*(1).

absolute path name

A path name beginning with a slash (/). It indicates that the file's location is given relative to the **root directory** (/), and that the search begins there.

access

The process of obtaining data from or placing data in storage, or the right to use system resources. Accessibility is governed by three process characteristics: the effective user ID, the effective group ID, and the group access list. The *access*(2) system call determines accessibility of a file according to the bit pattern contained in its *amode* parameter, which is constructed to read, write, execute or check the existence of a file. The *access*(2) system call uses the **real user ID** instead of the **effective user ID** and the **real group ID** instead of the **effective group ID**.

access groups

The group access list is a set of **supplementary group IDs** used in determining resource accessibility. Access checks are performed as described below in **file access permissions**.

access mode

An access mode is a form of access permitted to a file. Each implementation provides separate read, write, and execute/search access modes.

address

A number used in information storage or retrieval to specify and identify memory location. An **address** is used to mark, direct, indicate destination, instruct or otherwise communicate with computer elements.

In mail, **address** is a data structure whose format can be recognized by all elements involved in transmitting information. On a local system, this might be as simple as the user's **login** name, while in a networked system, **address** specifies the location of the resource to the network software.

In a text editor (such as **vi**, **ex**, **ed**, or **sed**), an **address** locates the line in a file on which a given instruction is intended.

For **adb**, the **address** specifies at what assembly-language instruction to execute a given command.

In disk utilities such as **fsdb**, **address** might refer to a raw or **block special file**, the **inode** number, **volume header**, or other file attribute.

In the context of peripheral devices, **address** refers to a set of values that specify the location of an I/O device to the computer. The exact details of the formation of an address differ between systems.

address space

The range of memory locations to which a process can refer.

affiliation

See **terminal affiliation**.

agile addressing

An addressing scheme where an address or path to a logical unit that is independent of the physical path. See *intro(7)* for more information.

appropriate privileges

Each implementation provides a means of associating privileges with a process for function calls and function call options requiring special privileges. In the HP-UX system, **appropriate privileges** refers either to superuser status or to a privilege associated with privilege groups (see *setprivgrp(1M)*).

archive

A file comprised of the contents of other files, such as a group of object files (that is, **.o**) used by the linker, *ld(1)*). An archive file is created and maintained by *ar(1)* or similar programs, such as *tar(1)* or *cpio(1)*. An **archive** is often called a **library**.

ASCII

An acronym for American Standard Code for Information Interchange. ASCII is the traditional System V coded character set and defines 128 characters, including both control characters and graphic characters, each of which is represented by 7-bit binary values ranging from 0 through 127 decimal.

background process group

Any process group that is a member of a session which has established a connection with a controlling terminal that is not in the foreground process group.

backup

The process of making a copy of all or part of the file system in order to preserve it, in case a system crash occurs (usually due to a power failure, hardware error, etc.). This is a highly recommended practice.

block

- (1) The fundamental unit of information HP-UX uses for access and storage allocation on a mass storage medium. The size of a block varies between implementations and between file systems. In order to present a more uniform interface to the user, most system calls and utilities use **block** to mean 512 bytes, independent of the actual block size of the medium. This is the meaning of **block** unless otherwise specified in the manual entry.
- (2) On media such as 9-track tape that write variable length strings of data, the size of those strings. **Block** is often used to distinguish from **record**; a block contains several records, whereas the number of records denotes the blocking factor.

b

block special file

A special file associated with a mass storage device (such as a hard disk or tape cartridge drive) that transfers data in multiple-byte blocks, rather than by series of individual bytes (see **character special file**). **Block special files** can be mounted. A **block special file** provides access to the device where hardware characteristics of the device are not visible.

boot, boot-up

The process of loading, initializing, and running an operating system.

boot area

A portion of a mass storage medium on which the volume header and a "bootstrap" program used in booting the operating system reside. The **boot area** is reserved exclusively for use by HP-UX.

boot ROM

A program residing in ROM (Read-Only Memory) that executes each time the computer is powered up and is designed to bring the computer to a desired state by means of its own action. The first few instructions of a bootstrap program are sufficient to bring the remainder of the program into the computer from an input device and initiate functions necessary for computation. The function of the boot ROM is to run tests on the computer's hardware, find all devices accessible through the computer, and then load either a specified operating system or the first operating system found according to a specific search algorithm.

bus address

A number which makes up part of the address HP-UX uses to locate a particular device. The **bus address** is determined by a switch setting on a peripheral device which allows the computer to distinguish between two devices connected to the same interface. A **bus address** is sometimes called a "device address".

character

An element used for the organization, control, or representation of text. Characters include **graphic characters** and **control characters**.

character set

A set of characters used to communicate in a native or computer language.

character special file

A special file associated with I/O devices that transfer data byte-by-byte. Other byte-mode I/O devices include printers, nine-track magnetic tape drives, and disk drives when accessed in "raw" mode (see **raw disk**). A **character special file** has no predefined structure.

child process

A new process created by a pre-existing process via the *fork(2)* system call. The new process is thereafter known to the pre-existing process as its **child process**. The pre-existing process is the **parent process** of the new process. See **parent process** and **fork**.

clock tick

A rate used within the system for scheduling and accounting. It consists of the number of intervals per second as defined by **CLK_TCK** that is used to express the value in type **clock_t**. **CLK_TCK** was previously known as the defined constant **HZ**.

coded character set

A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its corresponding bit representation. **ASCII** is a **coded character set**.

collating element

The smallest entity used in collation to determine the logical ordering of strings (that is, the **collation sequence**). To accommodate native languages, a collating element consists of either a single character, or two or more characters collating as a single entity. The current value of the **LANG** environment variable determines the current set of collating elements.

collation

The logical ordering of strings in a predefined sequence according to rules established by precedence. These rules identify a collation sequence among the collating elements and also govern the ordering of strings consisting of multiple collating elements, to accommodate native languages.

collation sequence

The ordering sequence applied to **collating elements** when they are sorted. To accommodate native languages, **collation sequence** can be thought of as the relative order of **collating elements** as set by the current value of the **LANG** environment variable. Characters can be omitted from the collation sequence, or two or more collating elements can be given the same relative order (see *string*(3C)).

command

A directive to perform a particular task. HP-UX commands are executed through a **command interpreter** called a **shell**. HP-UX supports several shells, including the POSIX shell (*sh-posix*(1)), the C shell (*cs*(1)), and the Korn shell (*ksh*(1)). See *sh*(1) for more information about supported shells. Most commands are carried out by an executable file, called a **utility**, which might take the form of a stand-alone unit of executable object code (a program) or a file containing a list of other programs to execute in a given order (a shell script). Scripts can contain references to other scripts, as well as to object-code programs. A typical **command** consists of the utility name followed by arguments that are passed to the utility. For example, in the command, **ls mydirectory**, **ls** is the utility name and **mydirectory** is an argument passed to the **ls** utility.

command interpreter

A program which reads lines of text from standard input (typed at the keyboard or read from a file), and interprets them as requests to execute other programs. A command interpreter for HP-UX is called a **shell**. See *sh*(1) and related manual entries.

Command Set 1980

See **CS/80**.

composite graphic symbol

A graphic symbol consisting of a combination of two or more other graphic symbols in a single character position, such as a diacritical mark and a basic letter.

control character

A character other than a graphic character that affects the recording, processing, transmission, or interpretation of text. In the **ASCII** character set, **control characters** are those in the range 0 through 31, and 127. Control characters can be generated by holding down the control key (which may be labeled CTRL, CONTROL, or CNTL depending on your terminal), and pressing a character key (as you would use SHIFT). These two-key sequences are often written as, for example, Control-**D**, Ctrl-**D**, or **^D**, where **^** stands for the control key.

controlling process

The session leader that establishes the connection to the **controlling terminal**. Should the terminal subsequently cease to be a controlling terminal for this session, the session leader ceases to be the controlling process.

controlling terminal

A terminal that is associated with a session. Each session can have at most one controlling terminal associated with it and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal cause signals to be sent to all processes in the foreground process group associated with the controlling terminal.

Coordinated Universal Time (UTC)

See **Epoch**.

CS/80, CS-80

A family of mass storage devices that communicate with the controlling computer by means of a series of commands and data transfer protocol referred to as the **CS/80** (Command Set 1980) command set. This command set was implemented in order to provide better forward/backward compatibility between models and generations of mass storage devices as technological advances develop. Some mass storage devices support only a subset of the full **CS/80** command set, and are usually referred to as **SS/80** (Subset 1980) devices.

crash

The unexpected shutdown of a program or system. If the operating system crashes, this is a "system crash", and requires the system to be rebooted.

current directory

See **working directory**.

current working directory

See **working directory**.

daemon

A process which runs in the background, and which is usually immune to termination instructions from a terminal. Its purpose is to perform various scheduling, clean-up, and maintenance jobs. *lpsched*(1M) is an example of a **daemon**. It exists to perform these functions for line printer jobs queued by *lp*(1). An example of a permanent **daemon** (that is, one that should never die) is *cron*(1M).

data encryption

A method for encoding information in order to protect sensitive or proprietary data. For example, HP-UX automatically encrypts all users' passwords. The encryption method used by HP-UX converts ASCII text into a base-64 representation using the alphabet `., /, 0-9, A-Z, a-z`. See *passwd*(4) for the numerical equivalents associated with this alphabet.

default search path

The sequence of directory prefixes that *sh*(1), *time*(1), and other HP-UX commands apply in searching for a file known by an relative path name (that is, a path name not beginning with a **slash** (/)). It is defined by the environment variable **PATH** (see *environ*(5)). *login*(1) sets **PATH** equal to `:/usr/bin`, which means that your working directory is the first directory searched, followed by `/usr/bin`. The search path can be redefined by modifying the value of **PATH**. This is usually done in `/etc/profile`, and/or in the **.profile** file found in the home directory.

defunct process

See **zombie process**.

delta

A term used in the **Source Code Control System** (SCCS) to describe a unit of one or more textual changes to an **SCCS file**. Each time an SCCS file is edited, changes made to the file are stored separately as a **delta**. The *get*(1) command is then used to specify which deltas are to be applied to or excluded from the SCCS file, thus yielding a particular version of the file. Contrast this with the **vi** or **ed** editor, which incorporates changes into the file immediately, eliminating any possibility of obtaining a previous version of that file. A similar capability is provided by RCS files (see *rcsintro*(5)).

demon

Improper spelling of the UNIX word **daemon**.

device

A computer peripheral or an object that appears to an application as such.

device address

See **bus address**.

device file

See **special file**.

directory

A file that provides the mapping between the names of files and their contents, and is manipulated by the operating system alone. For every file name contained in a directory, that directory contains a pointer to the file's **inode**; The pointer is called a **link**. A file can have several links appearing anywhere on the same file system. Each user is free to create as many directories as needed (using *mkdir*(1)), provided that the **parent directory** of the new directory gives the permission to do so. Once a directory has been created, it is ready to contain ordinary files and other directories. An HP-UX directory is named and behaves exactly like an ordinary file, with one exception: no user (including the superuser) is allowed to write data on the directory itself; this privilege is reserved for the HP-UX operating system.

By convention, a directory contains at least two links, `.` and `..`, referred to as **dot** and **dot-dot** respectively. `.` refers to the directory itself and `..` refers to its **parent directory**. A directory containing only `.` and `..` is considered empty.

dot

See **.** (**dot**).

dot-dot

See **..** (**dot-dot**).

dot-oh

See **.o** (**dot-oh**).

dot-oh file

See **.o** (**dot-oh**).

dot-oh format

See **.o** (**dot-oh**).

downshifting

The conversion of an uppercase character to its lowercase representation.

dynamic loader

A routine invoked at process startup time that loads shared libraries into a process's address space. The dynamic loader also resolves symbolic references between a program and the shared libraries, and initializes the shared libraries' linkage tables. See *dld.sl(5)* (PA-RISC systems) or *dld.so(5)* (Itanium®-based systems) for details.

effective group ID

Every process has an **effective group ID** that is used to determine **file access permissions**. A process's **effective group ID** is determined by the file (command) that process is executing. If that file's set-group-ID bit is set (located in the mode of the file, see **mode**), the process's **effective group ID** is set equal to the file's group ID. This makes the process appear to belong to the file's group, perhaps enabling the process to access files that must be accessed in order for the program to execute successfully. If the file's set-group-ID bit is not set, the process's **effective group ID** is inherited from the process's parent. The setting of the process's **effective group ID** lasts only as long as the program is being executed, after which the process's effective group ID is set equal to its real group ID. See **group**, **real group ID**, and **set-group-ID bit**.

effective user ID

A process has an **effective user ID** that is used to determine **file access permissions** (and other permissions with respect to system calls, if the effective user ID is 0, which means superuser). A process's effective user ID is determined by the file (command) that process is executing. If that file's set-user-ID bit is set (located in the mode of the file, see **mode**), the process's effective user ID is set equal to the file's user ID. This makes the process appear to be the file's owner, enabling the process to access files which must be accessed in order for the program to execute successfully. (Many HP-UX commands which are owned by **root**, such as **mkdir** and **mail**, have their set-user-ID bit set so other users can execute these commands.) If the file's set-user-ID bit is not set, the process's effective user ID is inherited from that process's parent. See **real user ID** and **set-user-ID bit**.

end-of-file (EOF)

- (1) The data returned when attempting to read past the logical end of a file via *stdio(3S)* routines. In this case, end-of-file is not properly a character.
- (2) The ASCII character Ctrl-D.
- (3) A character defined by *stty(1)* or *ioctl(2)* (see *termio(7)*) to act as end-of-file on your terminal. Usually this is Ctrl-D.
- (4) The return value from *read(2)* that indicates end of data.

environment

The set of defined shell variables (such as **EXINIT**, **HOME**, **PATH**, **SHELL**, **TERM**, and others) that define the conditions under which user commands run. These conditions can include user terminal characteristics, home directory, and default search path. Each shell variable setting in the current process is passed on to all **child processes** that are created, provided that each shell variable setting has been exported via the **export** command (see *sh(1)*). Unexported shell variable settings are meaningful only to the current process, and any child processes created get the default settings of certain shell variables by executing **/etc/profile**, **\$HOME/.profile**, or **\$HOME/.login**.

EOF

See **end-of-file**.

Epoch

The time period beginning at 0 hours, 0 minutes, 0 seconds, **Coordinated Universal Time (UTC)** on January 1, 1970. Increments quantify the amount of time elapsed from the Epoch to the referenced time.

Leap seconds, which occur at irregular intervals, are not reflected in the count of seconds between the Epoch and the referenced time. (Fourteen leap seconds occurred in the years 1970 through 1988.)

FIFO special file

A type of **file**. Data written to a **FIFO** is read on a first-in-first-out basis. Other characteristics are described in *open(2)*, *read(2)*, *write(2)* and *lseek(2)*.

file

A stream of bytes that can be written to and/or read from. A **file** has certain attributes, including permissions and type. File types include **regular file**, **character special file**, **block special file**, **FIFO special file**, network special file, **directory**, and **symbolic link**. Every file must have a **file name** that enables the user (and many of the HP-UX commands) to refer to the contents of the file. The system imposes no particular structure on the contents of a file, although some programs do. Files can be accessed serially or randomly (indexed by byte offset). The interpretation of file contents and structure is up to the programs that access the file.

file access mode

A characteristic of an **open file description** that determines whether the described file is open for reading, writing, or both. (See *open(2)*.)

file access permissions

Every file in the **file hierarchy** has a set of access permissions. These permissions are used in determining whether a process can perform a requested operation on the file (such as opening a file for writing). Access permissions are established when a file is created via the *open(2)* or *creat(2)* system calls, and can be changed subsequently through the *chmod(2)* call. These permissions are read by *stat(2)* or *fstat(2)*.

File access controls whether a file can be read, written, or executed. Directory files use the execute permission to control whether or not the directory can be searched.

File access permissions are interpreted by the system as they apply to three different classes of users: the **owner** of the file, the users in the file's **group**, and anyone else ("other"). Every file has an independent set of access permissions for each of these classes. When an access check is made, the system decides if permission should be granted by checking the access information applicable to the caller.

Read, write, and execute/search permissions on a file are granted to a process if any of the following conditions are met:

- The process's **effective user ID** is superuser.
- The process's **effective user ID** matches the user ID of the owner of the file and the appropriate access bit of the **owner** portion (0700) of the file mode is set.
- The process's **effective user ID** does not match the user ID of the owner of the file, and either the process's **effective group ID** matches the group ID of the file, or the group ID of the file is in the process's group access list, and the appropriate access bit of the **group** portion (070) of the file mode is set.
- The process's **effective user ID** does not match the user ID of the owner of the file, and the process's **effective group ID** does not match the group ID of the file, and the group ID of the file is not in the process's group access list, and the appropriate access bit of the "other" portion (07) of the file mode is set.

Otherwise, the corresponding permissions are denied.

file descriptor

A small unique, per-process, nonnegative integer identifier that is used to refer to a file opened for reading and/or writing. Each **file descriptor** refers to exactly one **open file description**.

A **file descriptor** is obtained through system calls such as *creat(2)*, *fcntl(2)*, *open(2)*, *pipe(2)*, or *dup(2)*. The **file descriptor** is used as an argument by calls such as *read(2)*, *write(2)*, *ioctl(2)*, and *close(2)*.

The value of a **file descriptor** has a range from 0 to one less than the system-defined maximum. The system-defined maximum is the value **NOFILE** in `<sys/param.h>`.

file group class

A process is in the **file group class** of a file if the process is not the **file owner class** and if the **effective group ID** or one of the **supplementary group IDs** of the process matches the group ID associated with the file.

file hierarchy

The collection of one or more **file systems** available on a system. All **files** in these **file systems** are organized in a single hierarchical structure in which all of the nonterminal nodes are **directories**. Because multiple **links** can refer to the same **file**, the directory is properly described as a directed graph.

file name

A string of up to 14 bytes (or 255 bytes on file systems that support long file names) used to refer to an ordinary file, special file, or directory. The byte values NUL (null) and slash (/) cannot be used as characters in a file name. Note that it is generally unwise to use *, ?, ,, [, or] as part of file names because the shell attaches special meaning to these characters (see *sh(1)*, *cs(1)*, or *ksh(1)*). Avoid beginning a file name with -, +, or =, because to some programs, these characters signify that a command argument follows. A file name is sometimes called a path name component. Although permitted, it is inadvisable to use characters that do not have a printable graphic on the hardware you commonly use, or that are likely to confuse your terminal.

file name portability

File names should be constructed from the **portable file name character set** because the use of other characters can be confusing or ambiguous in certain contexts.

file offset

The file offset specifies the position in the file where the next I/O operation begins. Each **open file description** associated with either a regular file or special file has a **file offset**. There is no file offset specified for a **pipe** or **FIFO**.

file other class

A process is in the **file other class** if the process is not in the **file owner class** or **file group class**.

file owner class

A process is in the **file owner class** if the **effective user ID** of the process matches the user ID of the file.

file permission bits

See **permission bits**.

file pointer

A data element obtained through any of the *fopen(3S)* standard I/O library routines that "points to" (refers to) a file opened for reading and/or writing, and which keeps track of where the next I/O operation will take place in the file (in the form of a byte offset relative to the beginning of the file). After obtaining the file pointer, it must thereafter be used to refer to the open file when using any of the standard I/O library routines. (See *stdio(3S)* for a list of these routines.)

file serial number

A file-system-unique identifier for a given file, also known as the file's **inode number**. Each **file serial number** identifies exactly one **inode**. **File serial numbers** are not necessarily unique across **file systems** in the **file hierarchy**.

file status flags

Part of an **open file description**. These flags can be used to modify the behavior of system calls that access the file described by the **open file description**.

file system

A collection of **files** and supporting data structures residing on a mass storage volume. A file system provides a name space for **file serial numbers** referring to those files. Refer to the System Administrator manuals supplied with your system for details concerning file system implementation and maintenance.

file times update

Each file has three associated time values that are updated when file data is accessed or modified, or when the file status is changed. These values are returned in the file characteristics structure, as described in `<sys/stat.h>`. For each function in HP-UX that reads or writes file data or changes the file status, the appropriate time-related files are noted as "marked-for-update". When an update point occurs, any marked fields are set to the current time and the update marks are cleared. One such update point occurs when the file is no longer open for any process. Updates are not performed for files on **read-only file systems**.

filter

A command that reads data from the standard input, performs a transformation on the data, and writes it to the standard output.

foreground process group

Each session that has established a connection with a controlling terminal has exactly one process group of the session as a foreground process group of that controlling terminal. The foreground process group has certain privileges when accessing its controlling terminal that are denied to background process groups. See `read(2)` and `write(2)`.

foreground process group ID

The process group ID of the foreground process group.

fork

An HP-UX system call (see `fork(2)`), which, when invoked by an existing process, causes a new process to be created. The new process is called the **child process**; the existing process is called the **parent process**. The child process is created by making an exact copy of the parent process. The parent and child processes are able to identify themselves by the value returned by their corresponding **fork** call (see `fork(2)` for details).

graphic character

A character other than a control character that has a visual representation when hand-written, printed, or displayed.

group

See **group ID**.

group ID

Associates zero or more users who must all be permitted to access the same set of files. The members of a group are defined in the files `/etc/passwd` and `/etc/loggingroup` (if it exists) via a numerical group ID that must be between zero and `UID_MAX`, inclusive. Users with identical group IDs are members of the same group. An ASCII group name is associated with each group ID in the file `/etc/group`. A group ID is also associated with every file in the **file hierarchy**, and the mode of each file contains a set of permission bits that apply only to this group. Thus, if you belong to a group that is associated with a file, and if the appropriate permissions are granted to your group in the file's mode, you can access the file. When the identity of a group is associated with a process, a group ID value is referred to as a **real group ID**, an **effective group ID**, a **supplementary group ID**, or a **saved group ID**. See also **privileged group** and **set-group-ID bit**.

group access list

A set of **supplementary group IDs** used in determining resource accessibility. Access checks are performed as described in **file access permissions**.

hardware path

A numeric string associated to a system component (bus, card, attached I/O device, and so on) and providing information related to the component location.

hierarchical directory

A directory (or file system) structure in which each directory can contain other directories as well as files.

home directory

The directory name given by the value of the environment variable **HOME**. When you first log in, `login(1)` automatically sets **HOME** to your **login directory**. You can change its value at any time. This is usually done in the `.profile` file contained in your **login directory**. Setting **HOME** does not affect your **login**

directory; it simply gives you a convenient way of referring to what is probably your most commonly used directory.

host name

A string of bytes that uniquely identifies the system in the network. The host name for your system can be viewed and/or set with the *hostname(1)* command. More information can be found in the *hostname(5)* manpage. See also **node name**.

image

The current state of your computer (or your portion of the computer, on a multiuser system) during the execution of a command. Often thought of as a "snapshot" of the state of the machine at any particular moment during execution.

init

A **system process** that performs initialization, is the ancestor of every other process in the system, and is used to start **login** processes. **init** usually has a **process ID** of **1**. See *init(1M)*.

interleave factor

A number that determines the order in which sectors on a mass storage medium are accessed. It can be optimized to make data acquisition more efficient.

inode

An **inode** is a structure that describes a file and is identified in the system by a **file serial number**. Every file or directory has associated with it an **inode**. Permissions that specify who can access the file and how are kept in a 9-bit field that is part of the **inode**. The **inode** also contains the file size, the user and group ID of the file, the number of links, and pointers to the disk blocks where the file's contents can be found. Each connection between an **inode** and its entry in one or more directories is called a **link**.

inode number

See **file serial number**.

Internal Terminal Emulator (ITE)

The "device driver" code contained in the HP-UX kernel that is associated with the computer's built-in keyboard and display or with a particular keyboard and display connected to the computer, depending on the Series and Model of system processor. See **system console** and the System Administrator manuals supplied with your system for details.

internationalization

The concept of providing software with the ability to support the **native language**, **local customs**, and **coded character set** of the user.

interrupt signal

The signal sent by **SIGINT** (see *signal(2)*). This signal generally terminates whatever program you are running. The key which sends this signal can be redefined with *ioctl(2)* or *stty(1)* (see *termio(7)*). It is often the ASCII DEL (rubout) character (the DEL key) or the BREAK key. Ctrl-C is often used instead.

intrinsic

See **system call**.

I/O redirection

A mechanism provided by the HP-UX shell for changing the source of data for standard input and/or the destination of data for standard output and standard error. See *sh(1)*.

ITE

See **Internal Terminal Emulator**.

job control

Job control allows users to selectively stop (suspend) execution of processes and continue (resume) their execution at a later time.

The user employs this facility via the interactive interface jointly supplied by the system terminal driver and certain shells (see *sh(1)*). The terminal driver recognizes a user-defined "suspend character", which causes the current foreground process group to stop and the user's job control shell to resume. The job

control shell provides commands that continue stopped process groups in either the foreground or background. The terminal driver also stops a background process group when any member of the background process group attempts to read from or write to the user's terminal. This allows the user to finish or suspend the **foreground process group** without interruption and continue the stopped **background process group** at a more convenient time.

See *stty(1)*, *sh(1)*, and related shell entries for usage and installation details, and the shell entries plus *signal(2)* and *termio(7)* for implementation details.

kernel

The HP-UX operating system. The kernel is the executable code responsible for managing the computer's resources, such as allocating memory, creating processes, and scheduling programs for execution. The kernel resides in RAM (random access memory) whenever HP-UX is running.

LANG

An environment variable used to inform a computer process of the user's requirements for **native language, local customs, and coded character set**.

legacy device special file

A special file associated with an I/O device (tape, disk, and so on), locked to a particular physical **hardware path**, containing hardware path information such as SCSI bus, target, and LUN in the device file name and minor number. See *intro(7)* for more information.

legacy hardware path

A hardware path following the legacy format conventions, that is, a series of bus-nexus addresses separated by / (slash) characters, leading to a host bus adapter (HBA). Beneath the HBA, additional address elements are separated by . (period) characters. All elements are represented in decimal. See *intro(7)* for more information.

library

A file containing a set of subroutines and variables that can be accessed by user programs. Libraries can be either archives or shared libraries. For example, `/usr/lib/libc.a` and `/usr/lib/libc.sl` are libraries containing all functions of Section 2 and all functions of Section 3 that are marked (3C) and (3S) in the *HP-UX Reference*. Similarly, `/usr/lib/libm.a` and `/usr/lib/libm.sl` are libraries containing all functions in Section 3 that are marked (3M) in the *HP-UX Reference*. See *intro(2)* and *intro(3C)*.

LIF

See **Logical Interchange Format**.

line

A sequence of text characters consisting of zero or more nonnewline characters plus a terminating newline character.

link

Link is a synonym for **directory entry**. It is an object that associates a file name with any type of file. The information constituting a **link** includes the name of the file and where the contents of that file can be found on a mass storage medium. One physical file can have several links to it. Several directory entries can associate names with a given file. If the links appear in different directories, the file may or may not have the same name in each. However, if the links appear in one directory, each link must have a unique name in that directory. Multiple links to directories are not allowed (except as created by a user with appropriate privileges). See *ln(1)*, *link(2)*, *unlink(2)*, and **symbolic link**.

Also, to prepare a program for execution; see **linker**.

link count

The number of directory entries that refer to a particular file.

linker

A program that combines one or more object programs into one program, searches libraries to resolve user program references, and builds an executable file in **a.out** format. This executable file is ready to be executed through the program loader, *exec(2)*. The linker is invoked with the *ld(1)* command. The linker is often called a **link editor**.

local customs

The conventions of a geographical area or territory for such things as date, time and currency formats.

localization

The process of adapting existing software to meet the local language, customs, and character set requirements of a particular geographical area.

Logical Interchange Format (LIF)

A standard format for mass storage implemented on many Hewlett-Packard computers to aid in media transportability. See *lif(4)* for more detail.

login

The process of gaining access to HP-UX. This consists of successful execution of the login sequence defined by *login(1)*, which varies depending on the system configuration. It requests a **login** name and possibly one or more passwords.

login directory

The directory in which you are placed immediately after you log in. This directory is defined for each user in the file `/etc/passwd`. The shell variable **HOME** is set automatically to your **login directory** by *login(1)* immediately after you log in. See **home directory**.

LUN

LUN refers to an end device, such as a disk or tape or a piece of logical storage in a disk array (mass storage term). Also known as a Logical Unit (LU).

LUN hardware path

A virtualized path that can represent multiple paths to a single mass storage device. It starts with a virtual bus-nexus (known as the **virtual root node**) with an address of 64000. Addressing beneath that virtual root node consists of a virtual bus address and a virtual LUN identifier, delimited by / (slash) characters. See *intro(7)* for more information.

lunpath hardware path

A hardware path to a LUN. It is composed of a series of bus-nexus addresses separated by / (slash) characters, leading to a host bus adapter (HBA). Beneath the HBA, additional address elements are represented in hexadecimal. The first elements represent a transport-dependent target address. The final element is a LUN address, which is the 64-bit representation of the LUN identifier reported by the target. See *intro(7)* for more information.

magic number

The first word of an **a.out** format or archive file. This word contains the system ID, which states what machine (hardware) the file will run on, and the file type (executable, sharable executable, archive, etc.).

major number

A number used exclusively to create special files that enable I/O to or from specific devices. This number indicates which device driver to use for the device. Refer to *mknod(2)* and the System Administrator manual supplied with your system for details.

message catalog

Program strings, such as program messages and prompts, are stored in a **message catalog** corresponding to a particular geographical area. Retrieval of a string from a **message catalog** is based on the value of the user's **LANG** environment variable (see **LANG**).

message queue identifier (msqid)

A unique positive integer created by a *msgget(2)* system call. Each **msqid** has a message queue and a data structure associated with it. The data structure is referred to as **msqid_ds** and contains the following members:

```

struct
ipc_perm  msg_perm;    /* operation permission */
msgqnum_t msg_qnum;   /* number of msgs on q */
msglen_t  msg_qbytes; /* max number of bytes on q */
msglen_t  msg_cbytes; /* current number of bytes on q */
pid_t     msg_lspid;  /* pid of last msgsnd operation */

```

```

pid_t      msg_lrpid; /* pid of last msgrcv operation */
time_t     msg_stime; /* last msgsnd time */
time_t     msg_rtime; /* last msgrcv time */
time_t     msg_ctime; /* last change time */
/* Times measured in secs since */
/* 00:00:00 GMT, Jan. 1, 1970 */

```

Message queue identifiers can be created using *ftok*(3C).

msg_perm is a **ipc_perm** structure that specifies the message operation permission (see below). This structure includes the following members:

```

uid_t      cuid; /* creator user id */
gid_t      cgid; /* creator group id */
uid_t      uid; /* user id */
gid_t      gid; /* group id */
mode_t     mode; /* r/w permission */

```

msg_qnum is the number of messages currently on the queue. **msg_qbytes** is the maximum number of bytes allowed on the queue. **msg_lspid** is the process id of the last process that performed a **msgsnd** operation. **msg_lrpid** is the process id of the last process that performed a **msgrcv** operation. **msg_stime** is the time of the last **msgsnd** operation, **msg_rtime** is the time of the last **msgrcv** operation, and **msg_ctime** is the time of the last *msgctl*(2) operation that changed a member of the above structure.

message operation permissions

In the *msgop*(2) and *msgctl*(2) system call descriptions, the permission required for an operation is indicated for each operation. Whether a particular process has these permissions for an object is determined by the object's permission mode bits as follows:

```

00400      Read by user
00200      Write by user
00060      Read, Write by group
00006      Read, Write by others

```

Read and Write permissions on a **msgqid** are granted to a process if one or more of the following are true:

- The process's effective user ID is superuser.
- The process's effective user ID matches **msg_perm.[c]uid** in the data structure associated with **msgqid** and the appropriate bit of the "user" portion (0600) of **msg_perm.mode** is set.
- The process's effective user ID does not match **msg_perm.[c]uid** and either the process's effective group ID matches **msg_perm.[c]gid** or one of **msg_perm.[c]gid** is in the process's group access list and the appropriate bit of the "group" portion (00060) of **msg_perm.mode** is set.
- The process's effective user ID does not match **msg_perm.[c]uid** and the process's effective group ID does not match **msg_perm.[c]gid** and neither of **msg_perm.[c]gid** is in the process's group access list and the appropriate bit of the "other" portion (06) of **msg_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

metacharacter

A character that has special meaning to the HP-UX shell, as well as to commands such as **ed**, **find**, and **grep** (see *ed*(1), *find*(1), and *grep*(1)). The set of metacharacters includes: **!**, **"**, **&**, **'**, *****, **;**, **<**, **>**, **?**, **[**, **]**, ****, and **|**. Refer to *sh*(1) and the related shell manual entries for the meaning associated with each. See also **regular expression**.

minor number

A number that is an attribute of special files, specified during their creation and used whenever they are accessed, to enable I/O to or from specific devices. This number is passed to the device driver and is used to select which device in a family of devices is to be used, and possibly some operational modes. The exact format and meaning of the **minor number** depends both on the driver and on the addressing format (legacy or agile) being used. In legacy format, the minor number encodes path information, but in agile format, the minor number is opaque and based on the WWID.

mode

A 16-bit word associated with every file in the file system, stored in the **inode**. The least-significant 12 bits of the **mode** determine the read, write, and execute permissions for the file owner, file group, and all others, and contain the set-user-ID, set-group-ID, and sticky bits. The least-significant 12 bits can be set by the *chmod*(1) command if you are the file's owner or the superuser. These 12 bits are sometimes referred to as **permission bits**. The most-significant 4 bits specify the file type for the associated file and are set as the result of *open*(2) or *mknod*(2) system calls.

mountable file system

A removable blocked file system contained on some mass storage medium with its own root directory and an independent hierarchy of directories and files. See **block special file** and *mount*(1M).

msqid

See **message queue identifier**.

Multiplexer (MUX)

Multiplexer (MUX) is a high-speed serial communication multiple port product. It combines various signals for transmission over a single channel and provides intelligent communication functions to off-load CPU serial communication processing tasks.

multiuser state

The condition of the HP-UX operating system in which terminals (in addition to the system console) allow communication between the system and its users. By convention, multiuser run level is set at state 2, which is usually defined to contain all the terminal processes and **daemons** needed in a multiuser environment. Run levels are table driven, and are specified by *init*(1M), which sets the run level by looking at the file */etc/inittab*. Do not confuse the multiuser system with the multiuser state. A multiuser system is a system which can have more than one user actively communicating with the system when it is in the multiuser state. The multiuser state removes the single-user restriction imposed by the single-user state (see **single-user state**, *inittab*(4)).

native language

A computer user's spoken or written language, such as Chinese, Dutch, English, French, German, Greek, Italian, Katakana, Korean, Spanish, Swedish, Turkish, and so on.

Network File System (NFS)

The Network File System (NFS) allows a client node to perform transparent file access over the network.

By using NFS, a client node operates on files residing on a variety of servers and server architectures, and across a variety of operating systems. File access calls on the client (such as read requests) are converted to NFS protocol requests and sent to the server system over the network. The server receives the request, performs the actual file system operation, and sends a response back to the client.

NFS operates in a stateless manner using remote procedure calls (RPC) built on top of an external data representation (XDR) protocol. The RPC protocol enables version and authentication parameters to be exchanged for security over the network.

A server grants access to a specific file system to clients by adding an entry for that file system to the server's */etc/dfs/dfstab* file.

Native Language Support (NLS)

A feature of HP-UX that provides the user with internationalized software and the application programmer with tools to develop this software.

newline character

The character with an ASCII value of 10 (line feed) used to separate lines of characters. It is represented by `\n` in the C language and in various utilities. The terminal driver normally interprets a carriage-return/line-feed sequence sent by a terminal as a single newline character (but see *tty*(7) for full details)

NLS

See **Native Language Support**.

NLSPATH

An environment variable used to indicate the search path for message catalogs (see **message catalog**).

node name

A string of bytes which uniquely identifies the system in the local network. Unlike the **host name**, the node name cannot include domain names. It can be viewed and/or set with the *uname(1)* command. The node and host names are usually set to the same value as application programs sometimes use the node and host names interchangeably.

nonspacing characters

Characters, such as a diacritical mark or accents, that are used in combination with other characters to form composite graphic symbols commonly found in non-English languages.

open file

A file that is currently associated with a file descriptor.

open file description

A record of how a process or a group of processes is accessing a file. Each **file descriptor** refers to exactly one **open file description**, but an **open file description** can be referred to by more than one file descriptor. The **file offset**, **file status flags**, and **file access modes** are attributes of an **open file description**.

ordinary file

A type of HP-UX file containing ASCII text (for example, program source), binary data (for example, executable code), etc. Ordinary files can be created by the user through I/O redirection, editors, or HP-UX commands.

orphan process

A **child process** that is left behind when a **parent process** terminates for any reason. The **init** process (see *init(1M)*) inherits (that is, becomes the effective parent of) all orphan processes.

orphaned process group

A process group in which the parent of every member is either itself a member of the group or is not a member of the group's session.

owner

The owner of a file is usually the creator of that file. However, the ownership of a file can be changed by the superuser or the current owner with the *chown(1)* command or the *chown(2)* system call. The file owner is able to do whatever he wants with his files, including remove them, copy them, move them, change their contents, etc. The owner can also change the files' modes.

parent directory

The directory one level above a directory in the **file hierarchy**. All directories except the **root directory** (*/*) have one (and only one) parent directory. The **root directory** has no parent. See also **dot** and **dot-dot**.

parent process

Whenever a new process is created by a currently-existing process (via *fork(2)*), the currently existing process is said to be the parent process of the newly created process. Every process has exactly one parent process (except the **init** process, see **init**), but each process can create several new processes with the *fork(2)* system call. The parent process ID of any process is the **process ID** of its creator.

parent process ID

A new process is created by a currently active process. The **parent process ID** of a process is the process ID of its creator for the lifetime of the creator. After the creator's lifetime has ended, the **parent process ID** is the process ID of **init**.

password

A string of ASCII characters used to verify the identity of a user. Passwords can be associated with users and groups. If a user has a password, it is automatically encrypted and entered in the second field of that user's line in the */etc/passwd* file. A user can create or change his or her own password by using the *passwd(1)* command.

path name

A sequence of directory names separated by slashes, and ending with any file name. All file names except the last in the sequence *must* be directories. If a path name begins with a **slash** (*/*), it is an **absolute**

path name; otherwise, it is a **relative path name**. A path name defines the path to be followed through the hierarchical file system in order to find a particular file.

More precisely, a path name is a null-terminated character string constructed as follows:

```
<path-name>::=<file-name>|<path-prefix><file-name>|/
<path-prefix>::=<rtprefix>|/<rtprefix>
<rtprefix>::=<dirname>/|<rtprefix><dirname>/
```

where <file-name> is a string of one or more characters other than the ASCII slash and null, and <dirname> is a string of one or more characters (other than the ASCII slash and null) that names a directory. File and directory names can consist of up to 14 characters on systems supporting short file names and up to 255 characters on systems supporting long file names.

A **slash (/)** by itself names the **root directory**. Two or more slashes in succession (////...) are treated as a single slash.

Unless specifically stated otherwise, the null or zero-length path name is treated as though it named a nonexistent file.

path name resolution

The process that resolves a path name to a particular file in a **file hierarchy**. Multiple path names can resolve to the same file, depending on whether resolution is sought in absolute or relative terms (see below). Each file name in the path name is located in the directory specified by its predecessor (for example, in the path name fragment **a/b**, file **b** is located in directory **a**). **Path name resolution** fails if this cannot be accomplished.

If the path name begins with a slash, the predecessor of the first file name in the path name is understood to be the **root directory** of the process, and the path name is referred to as an **absolute path name**. If the path name does not begin with a slash, the predecessor of the first file name of the path name is understood to be the current working directory of the process, and the path name is referred to as a **relative path name**. A path name consisting of a single slash resolves to the root directory of the process.

path prefix

A **path name** with an optional ending **slash** that refers to a **directory**.

permission bits

The nine least-significant bits of a file's **mode** are referred to as file **permission bits**. These bits determine read, write, and execute permissions for the file's **owner**, the file's **group**, and all others. The bits are divided into three parts: owner, group and other. Each part is used with the corresponding file class of processes. The bits are contained in the file mode, as described in *stat(5)*. The detailed usage of the file permission bits in access decisions is described in **file access permissions**.

persistent device special file

A device file for mass storage devices, which is associated with a LUN hardware path, and thus transparently supports **agile addressing** and multipathing. In other words, a persistent device special file is unchanged if the LUN is moved from one host bus adapter (HBA) to another, moved from one switch/hub port to another, presented via a different target port to the host, or configured with multiple hardware paths. See *intro(7)* for more information on device special files.

PIC

See **position-independent code**.

pipe

An interprocess I/O channel used to pass data between two processes. It is commonly used by the **shell** to transfer data from the standard output of one process to the standard input of another. On a command line, a pipe is signaled by a vertical bar (|). Output from the command to the left of the vertical bar is channeled directly into the standard input of the command on the right.

portable file name character set

The following set of graphical characters are portable across conforming implementations of IEEE Standard P1003.1:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
01234567890._-
```

The last three characters are the dot, underscore and hyphen characters, respectively. The hyphen should not be used as the first character of a portable file name.

position-independent code (PIC)

Object code that can run unmodified at any virtual address. Position-independent code can use PC-relative addressing modes and/or linkage tables. It is most often used in shared libraries, in which case the linkage tables are initialized by the dynamic loader. Position-independent code is generated when the **+z** or **+Z** compiler option is specified.

privileged groups

A **privileged group** is a group that has had a **setprivgrp** (see *getprivgrp(2)*) operation performed on it, giving it access to some system calls otherwise reserved for the superuser. See **appropriate privileges**.

process

An invocation of a program, or the execution of an image (see **image**). Although all commands and utilities are executed within processes, not all commands or utilities have a one-to-one correspondence with processes. Some commands (such as **cd**) execute within a process, but do not create any new processes. Others (such as in the case of **ls | wc -l**) create multiple processes. Several processes can be running the same program, but each can be different data and be in different stages of execution. A process can also be thought of as an **address space** and single thread of control that executes within that address space and its required system resources. A **process** is created by another process issuing the *fork(2)* function. The process that issues *fork(2)* is known as the **parent process** and the new process created by the *fork(2)* as the **child process**.

process 1

See **init**.

process group

Each process in the system is a member of a **process group**. This grouping permits the signaling of related processes. A newly created process joins the process group of its creator.

process group ID

Each process group in the system is uniquely identified during its lifetime by a **process group ID**, a positive integer less than or equal to **PIC_MAX**. A **process group ID** cannot be reused by the system until the process group lifetime ends.

process group leader

A **process group leader** is a process whose process ID is the same as its process group ID.

process group lifetime

A period of time that begins when a **process group** is created and ends when the last remaining process in the group leaves the group, either due to process termination or by calling the *setsid(2)* or *setpgid(2)* functions.

process ID

Each active process in the system is uniquely identified during its lifetime by a positive integer less than or equal to **PID_MAX** called a **process ID**. A process ID cannot be reused by the system until after the process lifetime ends. In addition, if there exists a process group whose process group ID is equal to that process ID, the process ID cannot be reused by the system until the process group lifetime ends.

process lifetime

After a process is created with a *fork(2)* function, it is considered active. Its thread of control and **address space** exist until it terminates. It then enters an inactive state where certain resources may be returned to the system, although some resources, such as the **process ID** are still in use. When another process executes a *wait()*, *wait3()*, or *waitpid()* function (see *wait(2)*) for an inactive process, the remaining resources are returned to the system. The last resource to be returned to the system is the process ID. At this time, the lifetime of the process ends.

program

A sequence of instructions to the computer in the form of binary code (resulting from the compilation and assembly of program source).

prompt

The characters displayed by the **shell** on the terminal indicating that the system is ready for a command. The prompt is usually a dollar sign (\$) for ordinary users (% in the C shell) and a pound sign (#) for the superuser, but you can redefine it to be any string by setting the appropriate shell variable (see *sh(1)* and related entries). See also **secondary prompt**.

quit signal

The **SIGQUIT** signal (see *signal(2)*). The quit signal is generated by typing the character defined by the teletype handler as your quit signal. (See *stty(1)*, *ioctl(2)*, and *termio(7)*.) The default is the ASCII FS character (ASCII value 28) generated by typing Ctrl-\ . This signal usually causes a running program to terminate and generates a file containing the "core image" of the terminated process. The core image is useful for debugging purposes. (Some systems do not support core images, and on those systems no such file is generated.)

radix character

The character that separates the integer part of a number from the fractional part. For example, in American usage, the **radix character** is a decimal point, while in Europe, a comma is used.

raw disk

The name given to a disk for which there exists a **character special file** that allows direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O call.

read-only file system

A characteristic of a **file system** that prevents file system modifications.

real group ID

A positive integer which is assigned to every user on the system. The association of a user and his or her **real group ID** is done in the file `/etc/passwd`. The modifier "real" is used because a user can also have an **effective group ID**. The real group ID can then be mapped to a group name in the file `/etc/group`, although it need not be. Thus, every user is a member of some group (which can be nameless), even if that group has only one member.

Every time a process creates a child process (via *fork(2)*), that process has a real group ID equal to the parent process's real group ID. This is useful for determining file access privileges within the process.

real user ID

A positive integer which is assigned to every user on the system. A real user ID is assigned to every valid **login name** in the file `/etc/passwd`. The modifier "real" is used because a user can also have an **effective user ID** (see **effective user ID**).

Every time a process creates a child process (via *fork(2)*), that process has a real user ID equal to the parent process's real user ID. This is useful for determining file access privileges within the process.

regular expression

A string of zero or more characters that selects text. All the characters contained in the string might be literal, meaning that the regular expression matches itself only; or one or more of the characters might be a **metacharacter**, meaning that a single regular expression could match several literal strings. Regular expressions are most often encountered in text editors (such as *ed(1)*, *ex(1)*, or *vi(1)*), where searches are performed for a specific piece of text, or in commands that were created to search for a particular string in a file (most notably *grep(1)*). Regular expressions are also encountered in the shell, especially when referring to file names on command lines.

regular file

A type of **file** that is a randomly accessible sequence of bytes, with no further structure imposed by the system. Its size can be extended. A regular file is also called an **ordinary file**.

relative path name

A **path name** that does not begin with a **slash (/)**. It indicates that a file's location is given relative to your current **working directory**, and that the search begins there (instead of at the **root directory**). For example, `dir1/file2` searches for the directory `dir1` in your current working directory; then `dir1` is searched for the file `file2`.

__restrict

A macro that is optionally applied to the function prototype when the application developer directly or indirectly selects C99 conformance. If the user chooses C99 conformance, the **__restrict** macro is changed to the **restrict** keyword. Otherwise, the **__restrict** macro is expanded to an empty string.

root directory

- (1) The highest level directory of the hierarchical file system, from which all other files branch. In HP-UX, the **slash (/)** character refers to the **root directory**. The root directory is the only directory in the file system that is its own **parent directory**.
- (2) Each process has associated with it a concept of a root directory for the purpose of resolving path name searches for those paths beginning with **slash (/)**. A process's root directory need not be the root directory of the root file system, and can be changed by the *chroot*(1M) command or *chroot*(2) system call. Such a directory appears to the process involved to be its own parent directory.

root volume

The mass storage volume which contains the boot area (which contains the HP-UX kernel) and the **root directory** of the HP-UX file system.

saved group ID

Every process has a saved group ID that retains the process's **effective group ID** from the last successful *exec*(2) or *setresgid*() (see *setresuid*(2)), or from the last superuser call to *setgid*() (see *setuid*(2)) or *setresuid*(2). *setgid*() permits a process to set its effective group ID to this remembered value. Consequently, a process that executes a program with the set-group-ID bit set and with a group ID of 5 (for example) can set its effective group ID to 5 at any time until the program terminates. See *exec*(2), *setuid*(2), **saved user ID**, **effective group ID**, and **set-group-ID bit**. The saved group ID is also known as the **saved set-group-ID**.

saved process group ID

Every process has a saved process group ID that retains the process's group ID from the last successful *exec*(2). See *setpgrp*(2), *termio*(7), and **process group ID**.

saved user ID

Every process has a **saved user ID** that retains the process's **effective user ID** from the last successful *exec*(2) or *setresuid*(2), or from the last superuser call to *setuid*(2). *setuid*(2) permits a process to set its effective user ID to this remembered value. Consequently, a process which executes a program with the set-user-ID bit set and with an owner ID of 5 (for example) can set its effective user ID to 5 at any time until the program terminates. See *exec*(2), *setuid*(2), **saved group ID**, **effective user ID**, and **set-user-ID bit**. The saved user ID is also known as the **saved set-user-ID**.

saved set-group-ID

See **saved group ID**.

saved set-user-ID

See **saved user ID**.

SCCS

See **Source Code Control System**.

Source Code Control System (SCCS)

A set of HP-UX commands that enables you to store changes to an **SCCS file** as separate "units" (called **deltas**). These units, each of which contains one or more textual changes to the file, can then be applied to or excluded from the SCCS file to obtain different versions of the file. The commands that make up SCCS are *admin*(1), *cdc*(1), *delta*(1), *get*(1), *prs*(1), *rmdel*(1), *sact*(1), *scsdiff*(1), *unset*(1), *val*(1), and *what*(1).

SCCS file

An ordinary text file that has been modified so the **Source Code Control System (SCCS)** can be used with it. This modification is done automatically by the *admin*(1) command. See also **delta**.

secondary prompt

One or more characters that the shell prints on the display, indicating that more input is needed. This prompt is not encountered nearly as frequently as the shell's primary prompt (see **prompt**). When it occurs, it is usually caused by an omitted right quote on a string (which confuses the shell), or when you

enter a shell programming language control-flow construct (such as a **for** construct) from the command line. By default, the shell's secondary prompt is the greater-than sign (>), but you can re-define it by setting the shell variable **PS2** appropriately in your **.profile** file. (The C shell has no secondary prompt.)

semaphore identifier (**semid**)

A unique positive integer created by a *semget(2)* system call. Each **semid** has a set of semaphores and a data structure associated with it. The data structure is referred to as **semid_ds** and contains the following members:

```

struct
ipc_perm  sem_perm; /* operation permission */
ushort    sem_nsems; /* number of sems in set */
time_t    sem_otime; /* last operation time */
time_t    sem_ctime; /* last change time */
/* Times measured in secs since */
/* 00:00:00 GMT, Jan. 1, 1970 */

```

Semaphore identifiers can be created using *ftok(3C)*.

sem_perm is a **ipc_perm** structure that specifies the semaphore operation permission (see below). This structure includes the following members:

```

uid_t     cuid; /* creator user id */
gid_t     cgid; /* creator group id */
uid_t     uid; /* user id */
gid_t     gid; /* group id */
mode_t    mode; /* r/a permission */

```

The value of **sem_nsems** is equal to the number of semaphores in the set. Each semaphore in the set is referenced by a positive integer referred to as a **sem_num**. **sem_num** values run sequentially from 0 to the value of **sem_nsems** minus 1. **sem_otime** is the time of the last *semop(2)* operation, and **sem_ctime** is the time of the last *semctl(2)* operation that changed a member of the above structure.

semaphore operation permissions

In the *semop(2)* and *semctl(2)* system call descriptions, the permission required for an operation is indicated for each operation. Whether a particular process has these permissions for an object is determined by the object's permission mode bits as follows:

```

00400    Read by user
00200    Alter by user
00060    Read, Alter by group
00006    Read, Alter by others

```

Read and Alter permissions on a **semid** are granted to a process if one or more of the following are true:

- The process's effective user ID is superuser.
- The process's effective user ID matches **sem_perm.[c]uid** in the data structure associated with **semid** and the appropriate bit of the "user" portion (0600) of **sem_perm.mode** is set.
- The process's effective user ID does not match **sem_perm.[c]uid** and the appropriate bit of the "group" portion (060) of **sem_perm.mode** is set.
- The process's effective user ID does not match **sem_perm.[c]uid** and the process's effective group ID does not match **sem_perm.[c]gid** and neither of **sem_perm.[c]gid** is in the process's group access list and the appropriate bit of the "other" portion (06) of **sem_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

semid

See **semaphore identifier**.

session

Each process group is a member of a session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its creator. A process can alter its session membership (see *setsid(2)*). A session can have multiple process groups (see *setpgid(2)*).

session leader

A process that has created a session (see *setsid(2)*).

session lifetime

The period between when a session is created and the end of the lifetime of all process groups that remain as members of the session.

set-group-ID bit

A single bit in the mode of every file in the file system. If a file is executed whose **set-group-ID bit** is set, the **effective group ID** of the process which executed the file is set equal to the **real group ID** of the owner of the file. See also **group**.

set-user-ID bit

A single bit in the mode of every file in the file system. If a file is executed whose **set-user-ID bit** is set, the **effective user ID** of the process that executed the file is set equal to the **real user ID** of the owner of the file.

shared library

An executable file that can be shared between several different programs. Code from a shared library is not linked into the program by *ld(1)*, but is instead mapped into the process's address space at run time by the dynamic loader. Shared libraries must contain position-independent code, and are created by *ld(1)*. They typically have the file name suffix **.sl**.

shared memory identifier (shmid)

A unique positive integer created by a *shmget(2)* system call. Each **shmid** has a segment of memory (referred to as a shared memory segment) and a data structure associated with it. The data structure is referred to as **shmid_ds** and contains the following members:

```

struct
ipc_perm  shm_perm;      /* operation permission struct */
size_t    shm_segsz;     /* size of segment */
pid_t     shm_cpid;      /* creator pid */
pid_t     shm_lpid;      /* pid of last operation */
shmatt_t  shm_nattch;    /* number of current attaches */
time_t    shm_atime;     /* last attach time */
time_t    shm_dtime;     /* last detach time */
time_t    shm_ctime;     /* last change time */
           /* Times measured in secs since */
           /* 00:00:00 GMT, Jan. 1, 1970 */

```

Shared memory identifiers can be created using *ftok(3C)*.

shm_perm is a **ipc_perm** structure that specifies the permission for a *shmop(2)* or *shmctl(2)* operation (see below). This structure includes the following members:

```

uid_t     cuid;          /* creator user id */
gid_t     cgid;          /* creator group id */
uid_t     uid;           /* user id */
gid_t     gid;           /* group id */
mode_t    mode;          /* r/w permission */

```

shm_segsz specifies the size of the shared memory segment. **shm_cpid** is the process id of the process that created the shared memory identifier. **shm_lpid** is the process id of the last process that performed a *shmop(2)* operation. **shm_nattch** is the number of processes that currently have this segment attached. **shm_atime** is the time of the last **shmat** operation, **shm_dtime** is the time of the last **shmdt** operation, and **shm_ctime** is the time of the last *shmctl(2)* operation that changed one of the members of the above structure.

shared memory operation permissions

In the *shmop(2)* and *shmctl(2)* system call descriptions, the permission required for an operation is indicated for each operation. Whether a particular process has the permission to perform a *shmop(2)* or *shmctl(2)* operation on an object is determined by the object's permission mode bits as follows:

```

00400    Read by user

```

00200	Write by user
00060	Read, Write by group
00006	Read, Write by others

Read and Write permissions for a *shmop(2)* or *shmctl(2)* operation on a **shared memory identifier** (**shmid**) are granted to a process if one or more of the following are true:

- The process's effective user ID is superuser.
- The process's effective user ID matches **shm_perm.[c]uid** in the data structure associated with the **shmid** and the appropriate bit of the "user" portion (0600) of **shm_perm.mode** is set.
- The process's effective user ID does not match **shm_perm.[c]uid** and either the process's effective group ID matches **shm_perm.[c]gid** or one of **shm_perm.[c]gid** is in the process's group access list and the appropriate bit of the "group" portion (060) of **shm_perm.mode** is set.
- The process's effective user ID does not match **shm_perm.[c]uid** and the process's effective group ID does not match **shm_perm.[c]gid** and neither of **shm_perm.[c]gid** is in the process's group access list and the appropriate bit of the "other" portion (06) of **shm_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

shell

A user interface to the HP-UX operating system. A shell often functions as both a command interpreter and an interpretive programming language. A shell is automatically invoked for every user who logs in. See *sh(1)* and its related manual entries plus the tutorials supplied with your system for details.

shell program

See **shell script**.

shell script

A sequence of shell commands and shell programming language constructs stored in a file and invoked as a user command (program). No compilation is needed prior to execution because the shell recognizes the commands and constructs that make up the shell programming language. A shell script is often called a **shell program** or a **command file**. See the *Shells User Guide*.

shmid

See **shared memory identifier**.

signal

A software interrupt sent to a process, informing it of special situations or events. Also, the event itself. See *signal(2)*.

single-user state

A condition of the HP-UX operating system in which the system console provides the only communication mechanism between the system and its user. By convention, single-user state is usually specified by *init(1M)* as run-level **S** or **s**. Do not confuse **single-user state**, in which the software is limiting a multiuser system to a single-user communication, with a single-user system, which can never communicate with more than one fixed terminal. See also **multiuser state**.

slash

The literal character `/`. A **path name** consisting of a single slash resolves to the **root directory** of the process. See also **path name resolution**.

solidus

See **slash**.

source code

The fundamental high-level information (program) written in the syntax of a specified computer language. Object (machine-language) code is derived from source code. When dealing with an HP-UX shell command language, **source code** is input to the command language interpreter. The term **shell script** is synonymous with this meaning. When dealing with the C Language, **source code** is input to the *cc(1)* command. **Source code** can also refer to a collection of sources meeting any of the above conditions.

special file

A file associated with an I/O device. Often called a **device file**. Special files are read and written the same as **ordinary files**, but requests to read or write result in activation of the associated device. Due to convention and consistency, these files should always reside in the **/dev** directory. See also **file**.

special system processes

Special system processes are those which are critical to basic system operation. They include: the scheduler, the initialization process (also known as **init**) and the pager.

SS/80

See **CS/80**.

standard error

The destination of error and special messages from a program, intended to be used for diagnostic messages. The standard error output is often called **stderr**, and is automatically opened for writing on file descriptor 2 for every command invoked. By default, the user's terminal is the destination of all data written to standard error, but it can be redirected elsewhere. Unlike standard input and standard output, which are never used for data transfer in the "wrong" direction, standard error is occasionally read. This is not recommended practice, since I/O redirection is likely to break a program doing this.

standard input

The source of input data for a program. The standard input file is often called **stdin**, and is automatically opened for reading on file descriptor 0 for every command invoked. By default, the user's terminal is the source of all data read from standard input, but it can be redirected from another source.

standard output

The destination of output data from a program. The standard output file is often called **stdout**, and is automatically opened for writing on file descriptor 1 for every command invoked. By default, the user's terminal is the destination of all data written to standard output, but it can be redirected elsewhere.

stderr

See **standard error**.

stdin

See **standard input**.

stdout

See **standard output**.

stream

A term most often used in conjunction with the standard I/O library routines documented in Section 3 of this manual. A stream is simply a file pointer (declared as **FILE *stream**) returned by the *fopen(3S)* library routines. It may or may not have buffering associated with it (by default, buffering is assigned, but this can be modified with *setbuf(3S)*).

sticky bit

A single bit in the mode of every file in the file system. The sticky bit has no significance if it is set on a **regular file**.

If set on a directory, the files in that directory can be removed or renamed only by the owner of the file, the owner of the directory containing the file, or superuser. See also *chmod(2)* , *rename(2)* , *rmdir(2)* , and *unlink(2)* .

subdirectory

A directory that is one or more levels lower in the file system hierarchy than a given directory. Sometimes called a **subordinate directory**.

subordinate directory

See **subdirectory**.

Subset 1980

See **CS/80**.

superblock

A block on each file system's mass storage medium which describes the file system. The contents of the superblock vary between implementations. Refer to the system administrator manuals supplied with your system for details.

superuser

The HP-UX system administrator. This user has access to all files, and can perform privileged operations. **superuser** has a **real user ID** and **effective user ID** of 0, and, by convention, the user name of **root**.

superior directory

See **parent directory**.

supplementary group ID

A process has up to **sysconf(_SC_NGROUPS_MAX)** supplementary group IDs used in determining file access permissions, in addition to the effective group ID. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created. Note that the value returned from **sysconf(_SC_NGROUPS_MAX)** may be larger than the value of **NGROUPS_MAX** found in **<limits.h>** on certain HP-UX systems.

symbolic link

A type of file that indirectly refers to a path name. See *symlink(4)*.

system

The HP-UX operating system. See also **kernel**.

system asynchronous I/O

A method of performing I/O whereby a process informs a driver or subsystem that it wants to know when data has arrived or when it is possible to perform a write request. The driver or subsystem maintains a set of buffers through which the process performs I/O. See *ioctl(2)*, *read(2)*, *select(2)*, and *write(2)* for more information.

system call

An HP-UX operating system kernel function available to the user through a high-level language (such as FORTRAN, Pascal, or C). Also called an "intrinsic" or a "system intrinsic." The available system calls are documented in Section 2 of the *HP-UX Reference*.

system console

A keyboard and display (or terminal) given a unique status by HP-UX and associated with the special file **/dev/console**. All boot ROM error messages, HP-UX system error messages, and certain system status messages are sent to the system console. Under certain conditions (such as the single-user state), the system console provides the only mechanism for communicating with HP-UX. See the System Administrator manuals and user guides provided with your system for details on configuration and use of the system console.

system process

A **system process** is a process that runs on behalf of the system. It may have special implementation-defined characteristics.

terminal

A **character special file** that obeys the specifications of *termio(7)*.

terminal affiliation

The process by which a process group leader establishes an association between itself and a particular terminal. A terminal becomes affiliated with a process group leader (and subsequently all processes created by the process group leader, see **terminal group**) whenever the process group leader executes (either directly or indirectly) an *open(2)* or *creat(2)* system call to open a terminal. Then, *if* the process which is executing *open(2)* or *creat(2)* is a process group leader, and *if* that process group leader is not yet affiliated with a terminal, and *if* the terminal being opened is not yet affiliated with a process group, the affiliation is established (however, see *open(2)* description of **O_NOCTTY**).

An affiliated terminal keeps track of its process group affiliation by storing the process group's process group ID in an internal structure.

Two benefits are realized by terminal affiliation. First, all signals sent from the terminal are sent to all processes in the terminal group. Second, all processes in the terminal group can perform I/O to/from the generic terminal driver `/dev/tty`, which automatically selects the affiliated terminal.

Terminal affiliation is broken with a terminal group when the process group leader terminates, after which the hangup signal is sent to all processes remaining in the process group. Also, if a process (which is not a process group leader) in the terminal group becomes a process group leader via the `setpgrp(2)` system call, its terminal affiliation is broken.

See **process group**, **process group leader**, **terminal group**, and `setpgrp(2)`.

terminal device

See **terminal**.

text file

A file that contains characters organized into one or more lines. The lines cannot contain NUL characters, and none can exceed **LINE_MAX** bytes in length including the terminating newline character. Although neither the kernel nor the C language implementation distinguishes between text files and binary files (see ANSI C Standard X3-159-19xx), many utilities behave predictably only when operating on text files.

tty

Originally, an abbreviation for teletypewriter; now, generally, a **terminal**.

upshifting

The conversion of a lowercase character to its uppercase representation.

user ID

Each system user is identified by an integer known as a **user ID**, which is in the range of zero to **UID_MAX**, inclusive. Depending on how the user is identified with a process, a **user ID** value is referred to as a **real user ID**, an **effective user ID**, or a **saved user ID**.

UTC

See **Epoch**.

utility

An executable file, which might contain executable object code (that is, a **program**), or a list of **commands** to execute in a given order (that is, a **shell script**). You can write your own utilities, either as executable programs or shell scripts (which are written in the shell programming language).

volume number

Part of an address used for devices. A number whose meaning is software- and device-dependent, but which is often used to specify a particular volume on a multivolume disk drive. See the System Administrator manuals supplied with your system for details.

whitespace

One or more characters which, when displayed, cause a movement of the cursor or print head, but do not result in the display of any visible graphic. The whitespace characters in the ASCII code set are space, tab, newline, form feed, carriage return, and vertical tab. A particular command or routine might interpret some, but not necessarily all, whitespace characters as delimiters for fields, words, or command options.

working directory

Each process has associated with it the concept of a current working directory. For a shell, this appears as the directory in which you currently "reside". This is the directory in which relative path name (that is, a path name that does not begin with `/`) searches begin. It is sometimes referred to as the **current directory**, or the **current working directory**.

zombie process

The name given to a process which terminates for any reason, but whose parent process has not yet waited for it to terminate (via `wait(2)`). The process which terminated continues to occupy a slot in the process table until its parent process waits for it. Because it has terminated, however, there is no other space allocated to it either in user or kernel space. It is therefore a relatively harmless occurrence which will rectify itself the next time its parent process waits. The `ps(1)` command lists zombie processes as **defunct**.

SEE ALSO

introduction(9).

NAME

introduction - HP-UX operating system and HP-UX Reference

INTRODUCTION

HP-UX is the Hewlett-Packard Company's implementation of a UNIX® operating system that is compatible with various industry standards. It is based on the System V Release 4 operating system (SVR4) and includes important features from the Fourth Berkeley Software Distribution (4BSD).

Improvements include enhanced capabilities and other features, developed by HP to make HP-UX a very powerful, useful, and reliable operating system, capable of supporting a wide range of applications ranging from simple text processing to sophisticated engineering graphics and design. It can readily be used to control instruments and other peripheral devices. Real-time capabilities further expand the flexibility of HP-UX as a powerful tool for solving tough problems in design, manufacturing, business, and other areas where responsiveness and performance are important.

Extensive international language support enables HP-UX to interact with users in any of dozens of human languages. HP-UX interfaces easily with local area networks and resource-sharing facilities. By using industry-standard protocols, HP-UX provides flexible interaction with other computers and operating systems. Optional software products extend HP-UX capabilities into a broad range of specialized needs.

The *HP-UX Reference* is not a learning tool for beginners. It is primarily a reference tool that is most useful for experienced users of UNIX or UNIX-like systems. If you are not already familiar with UNIX or HP-UX, refer to the series of Beginner's Guides, tutorial manuals, and other learning documents supplied with your system or available separately. System implementation and maintenance details are explained in the *HP-UX System Administrator's Guide*.

OTHER MANPAGES

This introduction and the section *intro* manpages describe the "core" manpages that are delivered with HP-UX. Other manpages may be delivered separately with optional HP-UX and third-party software and may reside in the same directories as the core manpages, or in other directories.

MANPAGE ORGANIZATION

The contents of the *HP-UX Reference* and its on-line counterpart are a number of independent entries called **manpages**. These are also called **manual entries** or **reference pages**.

For convenient reference, the manpages are divided into eight specialized sections. The printed manual also has a table of contents for each volume and a composite index.

Each manpage consists of one or more printed pages, with the manpage name and section number printed in the upper corners. Manpages are arranged alphabetically within each section of the reference, except for the *intro* page at the beginning of each section. Manpages are referred to by name and section number, in the form *pagename(section)*.

The manpages are available on-line through the **man** command if the manpages are present on the system. Refer to the *man(1)* manpage in Section 1 for more information.

Each page in the printed manual has two page numbers, printed at the bottom of the page. The center page number starts over with page 1 at the beginning of each new manpage; it is placed between two dashes in normal typeface. The number printed at the outside corner on each page is the sequence number of the page within the volume. Users usually locate manpages by the alphabetic headings at the top of the page as when reading a dictionary.

Some manpages describe two or more commands or routines. In such cases, the manpage is usually named for the first command or function that appears in the NAME section. Occasionally, a manpage name appears as a group descriptor in the NAME section. In such instances, the name describes the commands or functions in more general terms. For example, the *acct(1M)* manpage with group descriptor **acct:** describes the **acctdisk**, **acctdusg**, **accton**, and other commands, while the *string(3C)* manpage with group descriptor **string:** describes many character string functions.

SECTIONS OF THE HP-UX REFERENCE

The *HP-UX Reference* contains the following sections:

Volume Table of Contents (Printed Volumes)

A complete listing of all manpages in the order they appear in each section, as well as alphabetically intermixed lists of all command, function, and feature names that are different from the manpage where they appear.

Section 1: User Commands

Programs that are usually invoked directly by users or from command language procedures (scripts).

Section 1M: System Administration Commands

Commands used for system installation and maintenance, including boot processes, crash recovery, system integrity testing, and other needs. Most commands in this section require the superuser privilege.

Section 2: System Calls

Entries into the HP-UX kernel, including the C-language interface. These topics are primarily of interest to programmers.

Section 3: Library Functions

Available subroutines that reside (in binary form) in various system libraries. These topics are primarily of interest to programmers.

Section 4: File Formats

The structure of various types of files, such as header files, primarily of interest to administrators and programmers. For example, the link editor output file format is described in *a.out(4)*. Files that are used only by a single command (such as intermediate files used by assemblers) are not described. C-language declarations corresponding to the formats in Section 4 can be found in the directories `/usr/include` and `/usr/include/sys`.

Section 5: Miscellaneous Topics

A variety of information, such as descriptions of character sets, macro packages, and kernel tunables.

Section 6 (Unused)

This section was traditionally used for games. None are shipped with HP-UX.

Section 7: Device Special Files

The characteristics of device special files (DSF) that provide the link between HP-UX and system I/O devices. The names for each topic usually refer to the type of I/O device rather than to the names of individual special files.

Section 8: System Administration Commands

Some UNIX and Linux vendors put system administration commands here. Some third party vendors install commands in this section in HP-UX.

Section 9: General Information

General introductions (such as this) and a glossary of terms used in the HP-UX environment.

This section is also used by the Driver Development Kit to store its function and structure manpages, using the section numbers 9E, 9F, and 9S.

Composite Index (Printed Manual)

An alphabetical listing of keywords and topics based on the NAME section near the beginning of each manpage as well as other information, cross-referenced to manpage names and sections. The index also contains references to built-in features in the various command interpreters ("shells").

MANPAGE FORMATS

All manpages follow an established section heading format, but not all section headings are included in each manpage. A few manpages have self-explanatory specialized headings.

NAME

Gives the names of the commands, functions, or features and briefly states the purpose.

SYNOPSIS

Summarizes the syntax of the command or program entity. A few conventions are used:

Constant-width characters indicate literal characters that should be entered exactly as they appear. These characters appear in bold in the online manpages.

Italic strings represent variable elements that should be replaced with appropriate values.

Roman square brackets ([]) indicate that the contents are optional.

Roman braces ({ }) indicate a required element, usually in a choice.

Ellipses (...) indicate that the previous element and its preceding whitespace (if any) can be repeated.

Note: An argument beginning with a dash (-), a plus sign (+), or an equal sign (=) is often defined as a command option, even if it appears in a position where a file name could appear. Therefore, it is unwise to have file names that begin with -, +, or =.

Optional subsections can include the following:

Parameters For functions, a description of the parameters in the preceding syntax.

Structure Members For structures, a description of the structure elements in the preceding syntax.

Remarks Information about special software or hardware requirements.

DESCRIPTION

Discusses the function and behavior of each entry.

Optional subsections can include the following:

Options For commands, a description of the switch arguments.

Operands For commands, a description of the nonswitch arguments and keywords.

Access Control Lists

Multithread Usage

Security Restrictions Information on restrictions and privileges required to use the item.

EXTERNAL INFLUENCES

Information on what external factors, such as environment variables, may affect system behavior.

Optional subsections can include the following:

Environment Variables The effect of language-related and other environment variables on system behavior,

International Code Set Support

Whether there is support for single- and multibyte characters,

NETWORKING FEATURES

Information under this heading is applicable only if you are using the network feature described there.

Optional subsections can include the following:

NFS Information on the network file system.

RETURN VALUE

Describes the values returned by function calls or in the return code (\$?) by commands.

DIAGNOSTICS

For commands, the diagnostic information that may be produced. Self-explanatory messages are not listed.

Optional subsections can include the following:

Errors

Warnings

ERRORS

For functions, the function error values (set in **errno**) and their corresponding error conditions.

EXAMPLES

Examples of typical usage.

WARNINGS

Potential problems and deficiencies.

DEPENDENCIES

Variations in HP-UX operation that are related to the use of specific hardware, software, or

combinations of hardware and software.

AUTHOR

Indicates the principal developer of the software documented by the manpage. Unless noted otherwise, the source of an entry is System V.

FILES

The file names that are used or affected by the program or command.

SEE ALSO

Provides references to related manpages and other documentation.

STANDARDS CONFORMANCE

For each command or subroutine entry point addressed by one or more of the following industry standards, the standard specifications to which that HP-UX component conforms.

The various standards are:

AES	OSF Application Environment Specification
ANSI C	ANSI X3.159-1989
FIPS 151-1	Federal Information Processing Standard 151-1 (National Institute of Standards and Technology)
FIPS 151-2	Federal Information Processing Standard 151-2 (National Institute of Standards and Technology)
POSIX.1	IEEE Standard 1003.1-1988 (IEEE Computer Society) (Portable Operating System Interface for Computer Environments)
POSIX.2	IEEE Standard 1003.2-1990 (IEEE Computer Society) (Portable Operating System Interface for Computer Environments)
POSIX.4	IEEE Standard 1003.1b-1993 (IEEE Computer Society) (Portable Operating System Interface for Computer Environments)
SVID2	System V Interface Definition Issue 2
SVID3	System V Interface Definition Issue 3
XPG2	X/Open Portability Guide Issue 2 (X/Open, Ltd.)
XPG3	X/Open Portability Guide Issue 3 (X/Open, Ltd.)
XPG4	X/Open Portability Guide Issue 4 (X/Open, Ltd.)
XPG4.2	X/Open Portability Guide Issue 4 Version 2 (X/Open, Ltd.)

GETTING STARTED WITH HP-UX

This is a very brief overview of how to use the HP-UX system: how to log in and log out, how to communicate through your machine, and how to run a program.

HP-UX uses **control characters** to perform certain functions. Control characters are generally shown in the form \hat{x} , such as \hat{D} for Control-D. Hold down the **Control (Ctrl)** key while you press the character key.

Note: The key names **Enter** and **Return** refer to the same key.

Logging In

To log in you must have a valid user name and password, which can be obtained from your system administrator.

When a connection has been established, the system displays **login:** on your terminal. Type your user name and press the **Enter** key. Enter your password (it is not echoed by the system) and press **Enter**.

A list of copyright notices and a message-of-the-day may greet you before the first prompt.

It is important that you type your login name with lowercase letters, if possible. If you type uppercase letters, HP-UX assumes that your terminal cannot generate lowercase letters, and treats subsequent uppercase input as lowercase.

When you log in successfully, the system starts your login shell. The default is the POSIX shell, **/usr/bin/sh**. The POSIX shell (and its predecessors, the Korn and Bourne shells) use **\$** as the default

prompt for users. The C shell uses `%`. All the shells use `#` as the default superuser prompt.

See *login(1)* for more on login, *passwd(1)* to change your password, *chsh(1)* to change your login shell.

Logging Out

You can log out of the shells by typing an **exit** command or the **eof** (end-of-file) character (see the *Special Interactive Characters* subsection below). The shell terminates and the **login:** prompt appears again. (If you are using the C, Korn, or POSIX shells, respectively, see *chsh(1)*, *ksh(1)*, or *sh-posix(1)* for information about the **ignoreeof** special command.)

How to Communicate Through Your Terminal

HP-UX gathers keyboard input characters and saves them in a buffer. The accumulated characters are not passed to the shell or other program until you type **Enter**.

HP-UX terminal input/output is full-duplex. It has full read-ahead, which means that you can type at any time, even while a program is printing on your display or terminal. Of course, if you type during output, the output display will have the input characters interspersed in it. However, whatever you type will be saved and interpreted in the correct sequence. There is a limit to the amount of read-ahead, but it is generous and not likely to be exceeded unless the system is severely overloaded or operating abnormally. When the read-ahead limit is exceeded, the system throws away *all* the saved characters.

The *stty(1)* manpage tells you how to describe the characteristics of your terminal to the system. The *profile(4)* manpage explains how to accomplish this task automatically every time you log in.

Special Interactive Characters

A number of special characters are used to control the input and output of your terminal. These characters have defaults and can be redefined with the **stty** command (see *stty(1)*). Definitions of the **stty** names are in *termio(7)* and *termiox(7)*.

Note: The system administrator can modify the system login defaults by changing the characteristics of the */dev/ttyconf* device file with the **stty** command.

stty Name	System Default At Login Character (ASCII Name; Key Names)	Common User Redefinition
eof	^D (EOT)	
erase	#	^H (BS; Backspace)
kill	@	^U (NAK), ^X (CAN)
intr	^? (DEL; Delete, Rub, Rubout)	^C (ETX)
quit	^\ (FS)	
start	^Q (DC1; X-ON)	
stop	^S (DC3; X-OFF)	

The **eof** character terminates "file" input from the terminal, as read by programs and scripts. By extension, **eof** can also terminate the shell (see the *Logging Out* subsection above).

The **erase** character erases the last character typed. Successive uses of **erase** will erase characters back to, but not beyond, the beginning of the input line.

The **kill** character deletes all characters typed before it on a terminal input line.

The **intr** character generates an interrupt signal that bypasses the input buffer. This signal generally causes whatever program you are running to terminate. It can be used to stop a long printout that you don't want. However, programs can arrange either to ignore this signal altogether, or to be notified when it happens (instead of being terminated). For example, the **vi** editor catches interrupts and stops what it is doing, instead of terminating, so that an interrupt can be used to halt an editing operation without losing the file being edited.

The **quit** character generates a quit signal that bypasses the input buffer and most program traps and causes a running program to terminate. It can cause a core dump in the current directory.

The **stop** character can be used to pause output to the terminal. It is commonly used on video terminals to suspend output to the display while you read what is already being displayed. You can then resume output by typing the **start** character. When **stop** and **start** are used to suspend or resume output, they bypass the keyboard command-line buffer and are not passed to the program. However, any other characters typed on the keyboard are saved and used as input later in the program.

The **eof**, **erase**, and **kill** characters can be used as normal text characters if you escape them with a preceding `\`, as in `\^D`. Therefore, to erase a `\`, you need two **erases**.

The **intr**, **quit**, **start**, and **stop** characters cannot be escaped on the input line.

End-of-Line and Tab Characters

Besides adapting to the speed of the terminal, HP-UX tries to be intelligent as to whether you have a terminal with a newline (line-feed) key, or whether it must be simulated with a return/line-feed character pair. In the latter case, all incoming return characters are changed to line-feed characters (the standard line delimiter), and a return/line-feed pair is echoed to the terminal. If you get into the wrong mode, use the **stty** command to correct it (see *stty(1)*).

Tab characters are used freely in HP-UX source programs. If your terminal does not have the tab function, you can arrange to have tab characters changed into spaces during output, and echoed as spaces during input. The **stty** command sets or resets this mode. By default, the system assumes that tabs are set every eight character positions. The **tabs** command (see *tabs(1)*) can set tab stops on your terminal, if the terminal supports tabs.

How to Run a Program

When you have successfully logged into HP-UX, the shell monitors input from your terminal. The shell accepts typed lines from the terminal, splits them into command names and arguments, then executes the command. The command can be the name of a shell built-in, an executable script of commands, or an executable program. There is nothing special about system-provided commands, except that they are kept in directories where the shell can find them. You can also keep commands in your own directories and arrange for the shell to find them there.

The command name is the first word on an input line to the shell; the command and its arguments are separated from one another by blanks (one or more space and/or tab characters).

When a program terminates, the shell ordinarily regains control and prompts you to indicate that it is ready for another command. The shell has many other capabilities, which are described in detail in the appropriate manpages: *sh-posix(1)* for the POSIX shell, *ksh(1)* for the Korn shell, or *cs(1)* for the C shell.

The Current Directory

HP-UX has a file system arranged in a hierarchy of directories. When the system administrator gave you a user name, he or she also created a directory for you (ordinarily with the same name as your user name, and known as your *login* or *home* directory). When you log in, that directory becomes your *current* or *working* directory, and any file name you type is assumed to be in that directory by default. Because you are the owner of this directory, you have full permission to read, write, alter, or destroy its contents. The permissions you have for other directories and files will have been granted or denied to you by their respective owners, or by the system administrator. To change the current working directory use the **cd** command (see *cd(1)*).

Path Names

To refer to files not in the current directory, you must use a path name. Full (absolute) path names begin with `/`, which is the name of the *root* directory of the whole file system. After the slash comes the name of each directory containing the next subdirectory (followed by a `/`), until finally the file name is reached (for example, `/usr/ae/filex` refers to file **filex** in directory **ae**, while **ae** is itself a subdirectory of **usr**; **usr** is a subdirectory of the root directory). See *glossary(9)* for a formal definition of **path name**.

If your current directory contains subdirectories, the path names of files in them begin with the name of the corresponding subdirectory (without a prefixed `/`). Generally, a path name can be used anywhere a file name is required.

Important commands that modify the contents of directories are **cp**, **mv**, and **rm** which respectively copy, move (that is, rename, relocate, or both), and remove files. To determine the status of files or the contents of directories, use the **ls** command. Use **mkdir** to make directories, **rmdir** to destroy them, and **mv** to rename them. See *cp(1)*, *ls(1)*, *mkdir(1)*, *mv(1)*, *rm(1)*, and *rmdir(1)*.

Writing a Program

To enter the text of a source program into an HP-UX file, use a text editing program such as **vi**, **ex**, or **ed** (see *vi(1)*, *ex(1)*, and *ed(1)*). The three principal languages available under HP-UX are C (see *cc_bundled(1)* and *cc(1)*), FORTRAN (see *f77(1)*), and aC++ (see *aCC(1)*). After the program text has been entered with the editor and written into a file (whose name has the appropriate suffix), you can give the name of that file to the appropriate language processor as an argument. Normally, the output of the language processor will

be left in a file named **a.out** in the current directory. Since the results of a subsequent compilation may also be placed in **a.out**, thus overwriting the current output, you may want to use **mv** to give the output a unique name. If the program is written in assembly language, you will probably need to link library sub-routines with it (see *ld(1)*). FORTRAN, C, and aC++ call the linker automatically.

When you have gone through this entire process without encountering any diagnostics, the resulting program can be run by giving its name to the shell in response to the prompt.

Your programs can receive arguments from the command line just as system programs do by using the *argv* and *argc* parameters. For more information, see your language's *Programmer's Guide*.

Text Processing

Almost all text is entered through a text editor. The editor preferred above all others provided with HP-UX is the **vi** editor. For batch-processing text files, the **sed** editor is very efficient. The **ex** editor is useful for handling certain situations while using **vi** but most other editors are rarely used except in various scripts.

The following editors are the same program masquerading under various names: **vi**, **view**, and **vedit** (see *vi(1)*) and **ex** and **edit** (see *ex(1)*). For information about the **sed** stream editor, see *sed(1)*. The **ed** line editor is described in *ed(1)*.

The commands most often used to display text on a terminal are **cat**, **more**, and **pr**. See *cat(1)*, *more(1)*, and *pr(1)*. The **cat** command simply copies ASCII text to the terminal, with no processing at all. The **more** command displays text on the terminal a screenful at a time, pausing for an acknowledgement from the user before continuing. The **pr** command paginates text, supplies headings, and has a facility for multicolumn output. **pr** is most commonly used in conjunction with the **lp** command (see *lp(1)*) to pipe formatted text to a line printer.

Interuser Communication

Certain commands provide interuser communication. Even if you do not plan to use them, it could be beneficial to learn about them, because someone else may direct them toward you. To communicate with another user that is currently logged in, you can use **write** to transfer text directly to that user's terminal display (if permission to do so has been granted by the other user). Otherwise, **elm**, **mailx**, or **mail** (in order of ease of use) can send a message to another user's mailbox. The user is then informed by HP-UX that mail has arrived (if currently logged in) or mail is present (when the user next logs in). Refer to *elm(1)*, *mail(1)*, *mailx(1)*, and *write(1)* for explanations of how these commands are used.

ACKNOWLEDGEMENTS

UNIX is a registered trademark of The Open Group.

SEE ALSO

cat(1), *cc_bundled(1)*, *cd(1)*, *chsh(1)*, *cp(1)*, *csh(1)*, *ed(1)*, *ex(1)*, *ksh(1)*, *ld(1)*, *login(1)*, *lp(1)*, *ls(1)*, *mail(1)*, *mailx(1)*, *man(1)*, *mkdir(1)*, *more(1)*, *mv(1)*, *passwd(1)*, *pr(1)*, *rm(1)*, *rmdir(1)*, *sed(1)*, *sh(1)*, *sh-posix(1)*, *stty(1)*, *tabs(1)*, *vi(1)*, *write(1)*, *a.out(4)*, *profile(4)*, *glossary(9)*.

The HP Technical Documentation website at: <http://docs.hp.com>.

Index

All Volumes

Index

All Volumes

Index All Volumes

Description	Entry Name(Section)
.	glossary(9)
. - execute file commands in current shell	sh-posix(1)
.	glossary(9)
.forward file	sendmail(1M)
.netrc - login information for ftp, rexec, and rexec()	netrc(4)
.o	glossary(9)
.profile file	login(1)
.profile - shell script to set up user's environment at login	profile(4)
.rhosts file	login(1)
.rhosts - security files authorizing access by remote hosts and users on local host	hosts.equiv(4)
.so's from nroff input, eliminate	soelim(1)
/dev/console - system console interface	console(7)
/dev/syscon - system console interface	console(7)
/dev/systty - system console interface	console(7)
/dev/ttyconf - file for default terminal control characters	stty(1)
/dev/zero special file	zero(7)
/etc/default/usermod - list of home directory names	usermod(4)
/etc/dfs/sharetab - shared file system table	sharetab(4)
/etc/issue identification file	issue(4)
/etc/lvmpvg - LVM physical volume group information file	lvmpvg(4)
/etc/named.conf - configuration file for Internet domain name server	named.conf(4)
/etc/protocols - protocol name database	protocols(4)
/etc/rndc.conf - rndc configuration file	rndc.conf(4)
/etc/services.window; extract window IDs of user processes from	getmemwindow(1M)
/etc/shadow file; install, update or check the	pwconv(1M)
/sbin/set_parms special initialization script	hostname(1)
/usr/lib/tztab - time zone adjustment table for date and ctime()	tztab(4)
/usr/share/lib/termcap access routines, emulate	termcap(3X)
/var/adm/userdb; display information residing in the user database,	userdbget(1M)
/var/adm/userdb; modify information in the user database,	userdbset(1M)
/var/adm/userdb, read, write or delete information in the user database,	userdb_read(3)
/var/adm/userdb, verify or fix information in the user database,	userdbck(1M)
1 KB blocks; swap chunk size in	swchunk(5)
3000-mode packed-decimal library, HP	hppac(3X)
32-bit DMA pool; the amount of memory to reserve for the	dma32_pool_size(5)
4.2 BSD-compatible process control facilities	killpg(2)
6/PWB compatibility; terminal interface for Version	sttyv6(7)
64-bit shared library with explicit load address; open an HP 9000	dlopen_pa(3C)
6; Internet Protocol Version	IPv6(7P)
: - expand shell parameters	sh-posix(1)
<term.h> - terminal capabilities	term(5)
<unctrl.h> - definition for unctrl()	unctrl(5)
<complex.h> - complex functions and macros	complex(5)
<math.h> - math functions and constants	math(5)
<pwd.h> password file format	passwd(4)
<regexp.h> - regular expression and pattern matching notation definitions	regexp(5)
<shadow.h> password file format	shadow(4)
[[- evaluate boolean expression	sh-posix(1)
[OVERFLOW] if values do not fit in fields; causes uname() system function to return	uname_overflow(5)
__data_start - last locations in program	end(3C)
__pset_rtctl() - real-time processor set control	pset_rtctl(2)
__restrict macro	glossary(9)
__text_start - last locations in program	end(3C)
__uc_get_ar() - user context access (ucontext_t)	uc_access(3)
__uc_get_ar_bsp() - user context access (ucontext_t)	uc_access(3)

Index

All Volumes

Description	Entry Name(Section)
<code>_uc_get_ar_bspstore()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_ccv()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_csd()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_ec()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_fpsr()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_lc()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_pfs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_rsc()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_ssd()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ar_unat()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_brs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_cfm()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_cr()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ed()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_frs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_grs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_ip()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_prs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_reason()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_rsebs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_rsebs64()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_get_um()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_ccv()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_csd()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_ec()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_fpsr()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_lc()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_pfs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_rsc()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_ssd()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ar_unat()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_brs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_cfm()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ed()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_frs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_grs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_ip()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_prs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_rsebs()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_rsebs64()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_uc_set_um()</code> - user context access (<code>ucontext_t</code>)	<code>uc_access(3)</code>
<code>_Exit()</code> - terminate a process	<code>exit(2)</code>
<code>_exit()</code> - terminate a process	<code>exit(2)</code>
<code>_ldcvvt(), _ldfcvt(), _ldgcvt()</code> - convert long double to string	<code>ldcvvt(3C)</code>
<code>_longjmp()</code> - restore stack environment after non-local goto	<code>setjmp(3C)</code>
<code>_pututline()</code> - update or create entry in <code>utmp</code> file	<code>getut(3C)</code>
<code>_setjmp()</code> - save stack environment for non-local goto	<code>setjmp(3C)</code>
<code>_UNW_clear()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_clearAlertCode()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_createContext()</code> - allocate and deallocate unwind library data structure	<code>_UNW_createContextForSelf(3X)</code>
<code>_UNW_createContextForSelf()</code> - allocate and deallocate unwind library data structure	<code>_UNW_createContextForSelf(3X)</code>
<code>_UNW_currentContext()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_destroyContext()</code> - allocate and deallocate unwind library data structure	<code>_UNW_createContextForSelf(3X)</code>
<code>_UNW_FR_PhysicalNumber()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_getAlertCode()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getAR()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>

Description	Entry Name(Section)
<code>_UNW_getBR()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getCFM()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getFR()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getGR()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getGR_NaT()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getIP()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getKernelSavedContext()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getPR()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_getPreds()</code> - query values in unwind library data structure	<code>_UNW_getGR(3X)</code>
<code>_UNW_GR_PhysicalNumber()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_jmpbufContext()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_PR_PhysicalNumber()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setAR()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setBR()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setCFM()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setFR()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setGR()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setGR_NaT()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setIP()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setPR()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_setPreds()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>_UNW_STACK_TRACE()</code> - produce a trace back of the procedure call stack using the unwind library	<code>U_STACK_TRACE(3X)</code>
<code>_UNW_step()</code> - manipulate values in unwind library data structure	<code>_UNW_currentContext(3X)</code>
<code>a.out</code>	<code>glossary(9)</code>
<code>a.out</code> - assembler and link editor output format	<code>a.out(4)</code>
<code>a641()</code> - convert base-64 value to long integer ASCII string	<code>a641(3C)</code>
abbreviation of function keys, enable/disable	<code>keypad(3X)</code>
ABI and context code from current context, return	<code>uwx_get_abi_context_code(3X)</code>
abort a per-process timer	<code>rmtimer(3C)</code>
<code>abort()</code> - generate an IOT fault	<code>abort(3C)</code>
about EVM; provide information	<code>evminfo(1)</code>
<code>abs()</code> , <code>labs()</code> , <code>llabs()</code> , <code>imaxabs()</code> - return integer absolute value	<code>abs(3C)</code>
absolute debugger	<code>adb(1)</code>
absolute path name	<code>glossary(9)</code>
absolute system time, add a specific time interval to the current	<code>get_expiration_time(3T)</code>
absolute value functions	<code>fabs(3M)</code>
absolute value, return integer	<code>abs(3C)</code>
accept a connection on a socket	<code>accept(2)</code>
<code>accept()</code> - accept a connection on a socket	<code>accept(2)</code>
<code>accept</code> - allow LP printer queuing requests	<code>accept(1M)</code>
access	<code>glossary(9)</code>
access and manage the pathalias database	<code>uupath(1)</code>
access and modification times, set for files	<code>utimes(2)</code>
access and update routines for user-accounting database maintained by <code>utmpd</code>	<code>getuts(3C)</code>
access checks server; NFS	<code>mountd(1M)</code>
access control facility for internet services	<code>tcpd(1M)</code>
access control language extensions; host	<code>hosts_options(5)</code>
access control library	<code>hosts_access(3)</code>
access control list (ACL), change owner and/or group in	<code>chownacl(3C)</code>
access control list (ACL), copy to another file	<code>cpacl(3C)</code>
access control list (ACL) information, get	<code>getacl(2)</code>
Access Control List (ACL) information; JFS File Systems only; set a file's	<code>acl(2)</code>
access control list (ACL) information; set	<code>setacl(2)</code>
access control list (ACL) structure, convert to string form	<code>acltostr(3C)</code>
access control list (ACL) structure, HFS file system only; convert string form to	<code>strtoacl(3C)</code>
access control list; add, modify, or delete entry	<code>setaclentry(3C)</code>
access control lists (ACLs); introduction to HFS	<code>acl(5)</code>
access control lists (ACLs); introduction to JFS	<code>aclv(5)</code>

Index

All Volumes

Description	Entry Name(Section)
access control lists (ACLs) of files, list	lsacl(1)
Access Control Lists, view or modify	swacl(1M)
access control lists; add, modify, delete, copy, or summarize	chacl(1)
access control policy switch	acps(3)
Access Control Policy Switch (ACPS); configuration file for the	acps.conf(4)
Access Control Policy Switch Configuration	acps.conf(4)
Access Control (RBAC) database files, verify the syntax of the Role-Based	rbacdbchk(1M)
access control, role-based	rbac(5)
access control; let authorized users edit files that are under	privedit(1M)
access() - determine accessibility of a file	access(2)
access device driver, SCSI direct	scsi_disk(7)
access file information; change WU-FTPd group	privatepw(1)
access groups	glossary(9)
access information in the user database, /var/adm/userdb	userdb_read(3)
access list, get group	getgroups(2)
access list, initialize group	initgroups(3C)
access list, set group	setgroups(2)
access mode	glossary(9)
access or build a binary search tree	tsearch(3C)
access path of physical volume in LVM volume group, change	pvchange(1M)
access permissions mode mask for file-creation, set	umask(1)
access permissions; change file mode	chmod(1)
access permissions; change file mode	chmod(2)
access privileges for group, list	getprivgrp(1)
access protected password database entry; trusted systems	getprpwent(3)
access rights to a file, get a user's effective	getaccess(2)
access rights to file(s), list	getaccess(1)
access routines, emulate /usr/share/lib/termcap	termcap(3X)
Access software, Data Communications and Terminal Controller Device File	ddfa(7)
access the terminfo database	tput(1)
access times, set or update file	utime(2)
access to /etc/passwd and /etc/shadow files, control	lckpwwdf(3C)
access to audio on a workstation; OBSOLETEd; control	asecure(1M)
access (ucontext_t); user context	uc_access(3)
access utmp file entry	getut(3C)
access utmpx file entry	getutx(3C)
access, modification, and/or change times of file; update	touch(1)
accessibility of a file, determine	access(2)
accessing and ordering HP-UX documentation	manuals(5)
according to type; classify characters	ctype(3C)
account validation procedures; perform PAM	pam_acct_mgmt(3)
account, authentication, session and password management PAM modules for UNIX	pam_unix(5)
account, password, and session service module for HP-UX, extended authentication,	pam_hpsec(5)
account, session, and password management PAM modules for LDAP; authentication,	pam_ldap(5)
accountable user for the current process, retrieve the	getauduser(3)
accounting and miscellaneous accounting commands; overview of	acct(1M)
accounting commands; overview of accounting and miscellaneous	acct(1M)
accounting data, disk usage by user ID	diskusg(1M)
accounting database daemon, user	utmpd(1M)
accounting file size, defines the maximum	max_acct_file_size(5)
accounting files, process, convert to ASCII text format	acctprc(1M)
accounting files, process, summarize by user ID and name	acctprc(1M)
accounting files, total, merge or add	acctmerg(1M)
accounting files; search and print process	acctcom(1M)
accounting records, manipulate	fwtmp(1M)
accounting records, per-process, command summary from	acctcms(1M)
accounting when available disk space reaches threshold, suspend and resume	acctsuspend(5)
accounting: acctcms - command summary from per-process accounting records	acctcms(1M)
accounting: acctcon1 - convert login/logoff records to per-session accounting records	acctcon(1M)
accounting: acctcon2 - convert per-session records to total accounting records	acctcon(1M)
accounting: acctmerg - merge or add total accounting files	acctmerg(1M)

Description	Entry Name(Section)
accounting; acctprc1 - convert process accounting files to ASCII text format	acctprc(1M)
accounting; acctprc2 - summarize process accounting files created by acctprc1	acctprc(1M)
accounting; daily accounting shell procedure	runacct(1M)
accounting; enable or disable process accounting	acct(2)
accounting; per-process accounting file format	acct(4)
accounts; check status of local user	userstat(1M)
acct () - enable or disable process accounting	acct(2)
acct - overview of accounting and miscellaneous accounting commands	acct(1M)
acct - per-process accounting file format	acct(4)
acctcms - command summary from per-process accounting records	acctcms(1M)
acctcom - search and print process accounting files	acctcom(1M)
acctcon1 - convert login/logoff records to per-session accounting records	acctcon(1M)
acctcon2 - convert per-session records to total accounting records	acctcon(1M)
acctdisk - create disk usage accounting records	acct(1M)
acctdusg - compute disk usage by login name	acct(1M)
acctmerg - merge or add total accounting files	acctmerg(1M)
accton - define kernel process accounting output file or disable accounting	acct(1M)
acctprc - convert process accounting files to ASCII text format	acctprc(1M)
acctprc1 - convert process accounting files to ASCII text format	acctprc(1M)
acctprc2 - summarize process accounting files created by acctprc1	acctprc(1M)
acctresume - suspend and resume accounting when available disk space reaches threshold	acctsuspend(5)
acctsh - shell procedures for system accounting	acctsh(1M)
acctsuspend - suspend and resume accounting when available disk space reaches threshold	acctsuspend(5)
acctwtmp - write utmp record and reason for writing	acct(1M)
ACL entries on JFS, sort	aclsort(3C)
ACL information; JFS File Systems only; set a file's Access Control List	acl(2)
ACL information; set access control list	setacl(2)
acl - introduction to HFS access control lists (ACLs)	acl(5)
acl () - set a file's Access Control List (ACL) information; JFS File Systems only	acl(2)
ACL structure, HFS file system only; convert string form to access control list	strtoacl(3C)
ACL, view or modify	swacl(1M)
aclentrystart - convert string form to access control list (ACL) structure, HFS file system only	strtoacl(3C)
ACLs; introduction to HFS access control lists	acl(5)
ACLs; introduction to JFS access control lists	aclv(5)
aclsort - sort ACL entries on JFS	aclsort(3C)
acltostr () - convert access control list (ACL) structure to string form	acltostr(3C)
aclv - introduction to JFS access control lists (ACLs)	aclv(5)
acos () - arccosine function	acos(3M)
acosd () - degree-valued arccosine function	acosd(3M)
acosdf () - degree-valued arccosine function (float)	acosd(3M)
acosdl () - degree-valued arccosine function (long double)	acosd(3M)
acosdq () - degree-valued arccosine function (quad)	acosd(3M)
acosdw () - degree-valued arccosine function (extended)	acosd(3M)
acosf () - arccosine function (float)	acos(3M)
acosh () - arc hyperbolic cosine function	acosh(3M)
acoshf () - arc hyperbolic cosine function (float)	acosh(3M)
acoshl () - arc hyperbolic cosine function (long double)	acosh(3M)
acoshq () - arc hyperbolic cosine function (quad)	acosh(3M)
acoshw () - arc hyperbolic cosine function (extended)	acosh(3M)
acosl () - arccosine function (long double)	acos(3M)
acosq () - arccosine function (quad)	acos(3M)
acosw () - arccosine function (extended)	acos(3M)
acpm_getenvattrs () - ACPS Service Provider Interface	acps_spi(3)
acpm_getobj () - ACPS Service Provider Interface	acps_spi(3)
acpm_getobjattrs () - ACPS Service Provider Interface	acps_spi(3)
acpm_getop () - ACPS Service Provider Interface	acps_spi(3)
acpm_getopattrs () - ACPS Service Provider Interface	acps_spi(3)
acpm_getsubattrs () - ACPS Service Provider Interface	acps_spi(3)
acpm_getsubcreds () - ACPS Service Provider Interface	acps_spi(3)
acpm_getsubid () - ACPS Service Provider Interface	acps_spi(3)

Index

All Volumes

Description	Entry Name(Section)
ACPS Application Programming Interface	acps_api(3)
ACPS Service Provider Interface	acps_spi(3)
acps.conf - configuration file for the Access Control Policy Switch (ACPS)	acps.conf(4)
ACPS; configuration file for the Access Control Policy Switch	acps.conf(4)
acps_addenvattr() - ACPS Application Programming Interface	acps_api(3)
acps_addobjattr() - ACPS Application Programming Interface	acps_api(3)
acps_addopattr() - ACPS Application Programming Interface	acps_api(3)
acps_addsubattr() - ACPS Application Programming Interface	acps_api(3)
acps_addsubcred() - ACPS Application Programming Interface	acps_api(3)
acps_api - ACPS Application Programming Interface	acps_api(3)
acps_checkauth() - ACPS Application Programming Interface	acps_api(3)
acps_end() - ACPS Application Programming Interface	acps_api(3)
acps_setobj() - ACPS Application Programming Interface	acps_api(3)
acps_setop() - ACPS Application Programming Interface	acps_api(3)
acps_setsubid() - ACPS Application Programming Interface	acps_api(3)
acps_spi - ACPS Service Provider Interface	acps_spi(3)
acps_start() - ACPS Application Programming Interface	acps_api(3)
acquire handle for credential	gss_accept_sec_context(3)
across NFS, enable swapping	remote_nfs_swap(5)
activation of a cell from nPartition; cancel online cell operation; monitor online cell operation; reset hung cell during cell activation; online	parolrad(1M)
activation; online activation of a cell from nPartition; cancel online cell operation; monitor online cell operation; reset hung cell during cell	parolrad(1M)
active processes, kill (terminate) all	killall(1M)
activity report package; system	sa1(1M)
activity reporter; system	sar(1M)
activity, print current SCCS file editing	sact(1)
Adapter, Native Agent, for SNMP	naaagt(1M)
Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus	femsutil(1M)
adb - absolute debugger	adb(1)
add a complex character and rendition to a window	add_wch(3X)
add a new group to the system	groupadd(1M)
add a new user login to the system	useradd(1M)
add a printer for use with tsm	tsm.lpadmin(1M)
add a single-byte character and rendition to a window and advance the cursor	addch(3X)
add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
add a wide-character string to a window and advance the cursor	addnwstr(3X)
add an array of complex characters and renditions to a window	add_wchnstr(3X)
add an Object Identifier (OID) to an OID set	gss_add_oid_set_member(3)
add length limited string of single-byte characters and renditions to a window	addchnstr(3X)
add new commands to system	install(1M)
add or merge total accounting files	acctmrg(1M)
add physical volumes to extend an LVM volume group	vgextend(1M)
add string of single-byte characters and renditions to a window	addchstr(3X)
add value to environment	putenv(3C)
add value to environment	setenv(3C)
add, modify, and delete entries in an LDAP directory; simple	ldapentry(1)
add, modify, delete, copy, or summarize file access control lists (ACLs)	chacl(1)
add, modify, or delete access control list entry	setaclentry(3C)
add, remove and list gsscred table entries	gsscred(1M)
add, set, remove, and retrieve a process' privileges	priv_add(3)
add_wch() - add a complex character and rendition to a window	add_wch(3X)
add_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
add_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
addch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
addchnstr() - add length limited string of single-byte characters and renditions to a window ...	addchnstr(3X)
addchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
Addition of I/O chassis; command for Online Addition/Replacement/Deletion of PCI I/O cards and Online	olrad(1M)
Addition/Replacement/Deletion of PCI I/O cards and Online Addition of I/O chassis; command for Online	olrad(1M)

Description	Entry Name(Section)
additional cursor and window coordinates, get	olrad(1M)
additional severities, define	getbegyx(3X)
addmntent () - add entry to open file system descriptor file	addsev(3C)
addnstr () - add a string of multi-byte characters without rendition to a window and advance cursor	getmntent(3X)
addnwstr () - add a wide-character string to a window and advance the cursor	addnstr(3X)
address	addnwstr(3X)
address entry; get hostname and	glossary(9)
address information and open file descriptors; displays process	getaddrinfo(3N)
address manipulation routines, Internet	pmap(1)
address manipulation routines; Internet	inet6(3N)
address mapping, physical memory	inet(3N)
address of connected peer; get	iomap(7)
address resolution display and control	getpeername(2)
address resolution protocol	arp(1M)
address router, electronic	arp(7P)
address space	pathalias(1)
address space, get information for a process's	glossary(9)
address string conversion routines, network station	pstat(2)
address to a socket; bind an	net_aton(3C)
address, symbolic information of	bind(2)
address; get socket	dladdr(3C)
address; open an HP 9000 64-bit shared library with explicit load	getsockname(2)
adds a header to the current message	dlopen_pa(3C)
adds a recipient for the current message	smfi_addheader(3N)
addsev () - define additional severities	smfi_addrcpt(3N)
addstr () - add a string of multi-byte characters without rendition to a window and advance cursor	addsev(3C)
addwstr () - add a wide-character string to a window and advance the cursor	addnstr(3X)
adjtime () - correct the time to synchronize the system clock	addnwstr(3X)
admin - create and administer SCCS files	adjtime(2)
administer and create SCCS files	admin(1)
administer disk space used for caching file systems with CacheFS	admin(1)
administer SCCS files; create and	cfsadmin(1M)
administration command for file system	admin(1)
administration command for HFS file system	fsadm(1M)
administration commands, file system configuration and binary files	fsadm_hfs(1M)
Administration Manager Daemon for IPv6; Route	fs_wrapper(5)
administration manager; system	ramd(1M)
administration: STREAMS Administrative Driver	sam(1M)
administration; local area network	sad(7)
administrator, system	lanadmin(1M)
advance () - regular expression substring comparison routines	passwd(1)
AdvanceLink server, Basic Serial and HP	regexp(3X)
advertisement daemon for IPv6, Router	pcserver(1M)
advertisement daemon; configuration file for router	rtradvd(1M)
advise system of process's expected paging behavior	rtradvd.conf(4)
advisory information; file	madvise(2)
advisory information; file	fadvise(2)
advisory or enforced lock on an open file; apply or remove an	posix_fadvise(2)
affiliation	flock(2)
affirmative responses, repetitively	glossary(9)
Agent Adapter, Native, for SNMP	yes(1)
agents; configuration file for SLP	naaagt(1M)
agile addressing	slp.conf(4)
(aid) for the current process; get the audit ID	glossary(9)
(aid) for the current process; set the audit ID	getaudit(2)
AIO async I/O operations, maximum number of	setaudit(2)
AIO async I/O, maximum size of	aio_proc_max(5)
AIO pool, maximum number of process threads allowed in	aio_iosize_max(5)
	aio_proc_threads(5)

Index

All Volumes

Description	Entry Name(Section)
AIO pool, percentage of all process threads allowed in	aio_proc_thread_pct(5)
aio() - POSIX asynchronous I/O facility	aio(5)
AIO requests and servicing threads, desirable ratio between number of pending	aio_req_per_thread(5)
AIO thread pool monitor execution (in seconds), frequency of	aio_monitor_run_sec(5)
aio_cancel() - cancel asynchronous I/O operation	aio_cancel(2)
aio_error() - return error status of asynchronous I/O operation	aio_error(2)
aio_fsync() - bring asynchronous I/O operations to synchronized state	aio_fsync(2)
aio_iosize_max - maximum size of any AIO asynchronous I/O	aio_iosize_max(5)
aio_listio_max - maximum number of asynchronous I/O operation allowed in a listio call	aio_listio_max(5)
aio_max_ops - maximum number of POSIX async I/O operations that can be queued at any time	aio_max_ops(5)
aio_monitor_run_sec - frequency of AIO thread pool monitor execution (in seconds)	aio_monitor_run_sec(5)
aio_physmem_pct - percentage of physical memory lockable for request call-back POSIX asynchronous I/O operations	aio_physmem_pct(5)
aio_prio_delta_max - greatest delta (slowdown factor) allowed in POSIX async IO request priorities	aio_prio_delta(5)
aio_proc_thread_pct - percentage of all process threads allowed in AIO pool	aio_proc_thread_pct(5)
aio_proc_threads - maximum number of process threads allowed in AIO pool	aio_proc_threads(5)
aio_read() - start asynchronous read operation	aio_read(2)
aio_reap() - wait for multiple asynchronous I/O requests	aio_reap(2)
aio_req_per_thread - desirable ratio between number of pending AIO requests and servicing threads	aio_req_per_thread(5)
aio_return() - return asynchronous I/O operation status	aio_return(2)
aio_suspend() - suspend for asynchronous I/O completion	aio_suspend(2)
aio_suspend() - wait for asynchronous I/O completion	aio_suspend(2)
aio_write() - start asynchronous write operation	aio_write(2)
alarm clock, set a process's	alarm(2)
alarm() - set a process's alarm clock	alarm(2)
alias database	elm(1)
alias - substitute command and/or file name	sh-posix(1)
alias - substitute command and/or filename	csh(1)
alias - substitute command and/or filename	ksh(1)
alias text file	elm(1)
alias: install new elm aliases for user or system	newaliases(1)
aliases - aliases file for sendmail	aliases(5)
aliases and paths; locate a program file including	which(1)
aliases file for sendmail	aliases(5)
aliases file, for mail, rebuild the database	newaliases(1M)
aliases, elm user and system, verify and display	elmaliases(1)
aliases, print system-wide sendmail	praliases(1)
aligned memory; allocate	memalign(3C)
all users over a network, write to	rwall(1M)
alloc - show dynamic memory usage	csh(1)
alloca() - allocate space from the stack	malloc(3C)
allocate a per-process timer	mktimer(3C)
allocate aligned memory	memalign(3C)
allocate and deallocate unwind library data structure	_UNW_createContextForSelf(3X)
allocate data and stack space then lock process into memory	datalock(3C)
allocate reserved space for a disk storage file	prealloc(1)
allocate transport function library structure	t_alloc(3)
allocate_fs_swapmap - determines when swapmap structures are allocated for filesystem swap	allocate_fs_swapmap(5)
allocated for filesystem swap, determines when swapmap structures are	allocate_fs_swapmap(5)
allocated physical extents, move from one LVM physical volume to other physical volumes	pvmove(1M)
allocated program regions; first locations beyond	end(3C)
allocated, system-wide limit of queued signals that can be	ksi_alloc_max(5)
allocation policy on cell-based HP-UX servers; physical memory	numa_policy(5)
allocation space of object files, print section sizes and	size(1)
allocation, change data segment space	brk(2)
allocator for main memory	malloc(3C)

Description	Entry Name(Section)
allow signals to interrupt functions	siginterrupt(2)
allowed login shells, list of	shells(4)
allowed per process, defines the maximum number of threads	max_thread_proc(5)
allowed, maximum number of system-wide System V IPC message queues (IDs)	msgmni(5)
alpha paging	kermit(1)
alphasort () - sort a directory pointer array	scandir(3C)
alter contents of and copy a (tape) file	dd(1)
alter selected characters	tr(1)
alternate stack context, set/get signal alternate stack context	sigaltstack(2)
alwaysdump - defines which classes of kernel memory pages are dumped when a kernel panic occurs	alwaysdump(5)
amount of memory to reserve for the 32-bit DMA pool; the	dma32_pool_size(5)
an array of complex characters and renditions to a window, add	add_wchnstr(3X)
analysis information; display LP spooler performance	lpana(1M)
analyzer; system configuration drift	bastille_drift(1M)
annuity () - present value factor for annuity	annuity(3M)
annuityf () - present value factor for annuity (float)	annuity(3M)
annuityl () - present value factor for annuity (long double)	annuity(3M)
annuityq () - present value factor for annuity (quad)	annuity(3M)
annuityw () - present value factor for annuity (extended)	annuity(3M)
anonymous memory region, initialize semaphore in mapped file or	msem_init(2)
anonymous region, remove semaphore in mapped file or	msem_remove(2)
another system over LAN, log in on	vt(1)
another (UNIX) system, terminal emulator; call	cu(1)
answer - phone message transcription system	answer(1)
any time, maximum number of System V IPC messages in the system at	msgtql(5)
API interfaces to support large files, non-POSIX standard	creat64(2)
APIs, PAM Service Module APIs	pam_sm(3)
Application Programming Interface; ACPS	acps_api(3)
Application Programming Interface; Generic Security Service	gssapi(5)
application versions; coordinate ELF library and	elf_version(3E)
application with privileges after performing appropriate authorization checks and optionally reauthenticating the user; invoke another	privrun(1M)
application, explicit locking of streams within a multi-thread	flockfile(3S)
application, header file for future applications	portal(5)
applications and their associated memory window ID; file containing	services.window(4)
applications on Itanium-based systems running HP-UX; emulate PA-RISC HP-UX	Aries(5)
applications, number of priority values to support for POSIX.1b realtime	rtsched_numpri(5)
apply a diff file to an original file; program to	patch(1)
apply or remove an advisory or enforced lock on an open file	flock(2)
appropriate privileges	glossary(9)
ar - archive and library maintainer for portable archives	ar(1)
ar - common archive file format	ar(4)
arbitrary-precision arithmetic language	bc(1)
arc hyperbolic cosine functions	acosh(3M)
arc hyperbolic sine functions	asinh(3M)
arc hyperbolic tangent functions	atanh(3M)
arccosine functions	acos(3M)
archive	glossary(9)
archive and library maintainer for portable archives	ar(1)
archive exchange, portable	pax(1)
archive file format, common	ar(4)
archive files; copies files and directory hierarchies; extracts, writes, and lists	pax(1)
archive format, tar tape	tar(4)
archive member access for ELF files	elf_rand(3E)
archive member header for ELF files, retrieve	elf_getarhdr(3E)
archive package, make an	shar(1)
archive symbol table, regenerate	ranlib(1)
archive symbol table; retrieve	elf_getarsym(3E)
archive the file system	backup(1M)
archive; format of cpio	cpio(4)

Index

All Volumes

Description	Entry Name(Section)
archiver; tape file	tar(1)
archives in and out; copy file	cpio(1)
archives in and out; duplicate directory trees; copy file	cpio(1)
arcsine functions	asin(3M)
arctangent functions	atan(3M)
arctangent-and-quadrant functions	atan2(3M)
area network; virtual local	VLAN(7)
argument lists and execute command; construct	xargs(1)
argument lists, variable, macros for handling	varargs(5)
argument lists, variable, macros for handling	stdarg(5)
argument vector; get option letter from	getopt(3C)
argument, varargs, formatted input conversion to a	vscanf(3S)
arguments as an expression; evaluate	expr(1)
arguments, echo (print)	echo(1)
arguments; format and print	printf(1)
arguments; print formatted	printf(1)
Aries - emulate PA-RISC HP-UX applications on Itanium-based systems running HP-UX	Aries(5)
arithmetic desk calculator	dc(1)
arithmetic language, arbitrary-precision	bc(1)
arm a per-process timer, relatively	reltimer(3C)
arp - address resolution display and control	arp(1M)
arp - address resolution protocol	arp(7P)
array of complex characters and renditions, input from a window	in_wchnstr(3X)
array of single-byte characters and renditions, input from a window	inchnstr(3X)
array of wide characters and function key codes from a terminal; get an	getn_wstr(3X)
array, sort a directory pointer	scandir(3C)
as - assembler	as(1)
as - assembler for Integrity systems	as_ia(1)
as - assembler for PA-RISC systems	as_pa(1)
as_ia - assembler for Integrity systems	as_ia(1)
as_pa - assembler for PA-RISC systems	as_pa(1)
ASA carriage control characters; interpret	asa(1)
asa - interpret ASA carriage control characters	asa(1)
ASCII	glossary(9)
ASCII file format between HP-UX and DOS formats; convert	dos2ux(1)
ASCII format, dump iconv translation tables to	dmpxlt(1)
ascii - map of ASCII character set	ascii(5)
ASCII string, convert between long integer and base-64	a64l(3C)
ASCII string, convert long integer to	ltostr(3C)
ASCII, 7-bit, translate characters to	conv(3C)
ASCII, convert binary file to, for transmission by mailer	uencode(1)
asctime() , asctime_r() - convert tm structure date and time to string	ctime(3C)
asecure - control access to audio on a workstation; OBSOLETE	asecure(1M)
Aserver - start the audio server	aserver(1M)
asin() - arcsine function	asin(3M)
asind() - degree-valued arcsine function	asind(3M)
asindf() - degree-valued arcsine function (float)	asind(3M)
asindl() - degree-valued arcsine function (long double)	asind(3M)
asindq() - degree-valued arcsine function (quad)	asind(3M)
asindw() - degree-valued arcsine function (extended)	asind(3M)
asinf() - arcsine function (float)	asin(3M)
asinh() - arc hyperbolic sine function	asinh(3M)
asinhf() - arc hyperbolic sine function (float)	asinh(3M)
asinh1() - arc hyperbolic sine function (long double)	asinh(3M)
asinhq() - arc hyperbolic sine function (quad)	asinh(3M)
asinhw() - arc hyperbolic sine function (extended)	asinh(3M)
asinl() - arcsine function (long double)	asin(3M)
asinq() - arcsine function (quad)	asin(3M)
asinw() - arcsine function (extended)	asin(3M)
ask for help on SCCS commands	sccshelp(1)
ask for user response for SD-UX	swask(1M)

Description	Entry Name(Section)
assembler	as(1)
assembler and link editor output format	a.out(4)
assembler debugger	adb(1)
assembler for Integrity systems	as_ia(1)
assembler for PA-RISC systems	as_pa(1)
assert() - verify program assertion	assert(3X)
assertion, verify program	assert(3X)
assign buffering to a stream file	setbuf(3S)
assignment database entry for a trusted system; manipulate device	getdvagnt(3)
assignment database file for a trusted system; device	devassign(4)
associated attributes; change login password and	passwd(1)
associated memory window ID; file containing applications and their	services.window(4)
Async Cancel Safe	thread_safety(5)
async I/O operations, maximum number that can be queued by any process that uses aio_reap(2)	aio_proc_max(5)
Async Signal Safe	thread_safety(5)
asynchronous disk ports that can be open at any time; maximum number	max_async_ports(5)
asynchronous I/O for the NFS version 2 client; control the number of kernel threads that perform	nfs2_max_threads(5)
asynchronous I/O for the NFS version 3 client; control the number of kernel threads that perform	nfs3_max_threads(5)
asynchronous I/O for the NFS version 4 client; control the number of kernel threads that perform	nfs4_max_threads(5)
asynchronous I/O operations; percentage of physical memory lockable for request call-back POSIX	aio_physmem_pct(5)
asynchronous I/O, error status	aio_error(2)
asynchronous I/O, initiate list of operations	lio_listio(2)
asynchronous I/O, maximum size of	aio_iosize_max(5)
asynchronous I/O, POSIX	aio(5)
asynchronous I/O, start write	aio_write(2)
asynchronous I/O, status, return	aio_return(2)
asynchronous I/O, suspend for completion	aio_suspend(2)
asynchronous I/O, synchronize	aio_fsync(2)
asynchronous I/O, wait for completion	aio_suspend(2)
asynchronous I/O, wait for multiple requests	aio_reap(2)
asynchronous serial modem line control	modem(7)
asynchronous writes	fs_async(5)
asynchronous, cancel I/O	aio_cancel(2)
asynchronous, start read	aio_read(2)
at , batch , and crontab queue description file	queuedefs(4)
at ; prototype job file for	proto(4)
at - execute commands at a later time	at(1)
atan() - arctangent function	atan(3M)
atan2() - arctangent-and-quadrant function	atan2(3M)
atan2d() - degree-valued arctangent-and-quadrant function	atan2d(3M)
atan2df() - degree-valued arctangent-and-quadrant function (float)	atan2d(3M)
atan2dl() - degree-valued arctangent-and-quadrant function (long double)	atan2d(3M)
atan2dq() - degree-valued arctangent-and-quadrant function (quad)	atan2d(3M)
atan2dw() - degree-valued arctangent-and-quadrant function (extended)	atan2d(3M)
atan2f() - arctangent-and-quadrant function (float)	atan2(3M)
atan2l() - arctangent-and-quadrant function (long double)	atan2(3M)
atan2q() - arctangent-and-quadrant function (quad)	atan2(3M)
atan2w() - arctangent-and-quadrant function (extended)	atan2(3M)
atand() - degree-valued arctangent function	atand(3M)
atandf() - degree-valued arctangent function (float)	atand(3M)
atandl() - degree-valued arctangent function (long double)	atand(3M)
atandq() - degree-valued arctangent function (quad)	atand(3M)
atandw() - degree-valued arctangent function (extended)	atand(3M)
atanf() - arctangent function (float)	atan(3M)
atanh() - arc hyperbolic tangent function	atanh(3M)
atanhf() - arc hyperbolic tangent function (float)	atanh(3M)

Index

All Volumes

Description	Entry Name(Section)
atanhl() - arc hyperbolic tangent function (long double)	atanh(3M)
atanhq() - arc hyperbolic tangent function (quad)	atanh(3M)
atanhw() - arc hyperbolic tangent function (extended)	atanh(3M)
atanl() - arctangent function (long double)	atan(3M)
atanq() - arctangent function (quad)	atan(3M)
atanw() - arctangent function (extended)	atan(3M)
atexit() - register a function to be called at program termination	atexit(3)
atof() - convert string to double-precision number	strtod(3C)
atoi() - convert string to long integer	strtol(3C)
atol() - convert string to long integer	strtol(3C)
atoll() - convert string to long integer	strtol(3C)
atomically release blocked signals and wait for interrupt	sigpause(3C)
attach a STREAMS file descriptor to an object in the file system name space	fattach(3C)
attach shared memory to data segment	shmop(2)
attempt to lock a read-write lock for writing	pthread_rwlock_wrlock(3T)
attempts to create the interface socket that MTAs use to connect to the filter	smfi_opensocket(3N)
attention button events daemon, PCI I/O hotplug	hotplug(1M)
attention LEDs (cell, cabinet and I/O chassis attention LEDs); flash/turn off	fruled(1)
attr_get() - window attribute control functions	attr_get(3X)
attr_off() - window attribute control functions	attr_get(3X)
attr_on() - window attribute control functions	attr_get(3X)
attr_set() - window attribute control functions	attr_get(3X)
attribute object, initialize or destroy thread	pthread_attr_dinit(3T)
attribute of a system complex; modify an	cplxmodify(1M)
attribute, for window, control functions	attr_get(3X)
attributes and storage formats; all objects that Software Distributor (SD) uses, their	sd(4)
attributes associated with a message queue, get	mq_getattr(2)
Attributes Configuration tool; invokes the HP-UX Security	secweb(1M)
attributes - describe an audio file	attributes(1)
attributes for group, get special	getprivgrp(1)
attributes on a binary file; set extended security	setfilesec(1M)
attributes on Integrity systems; change program's internal	chattr_ia(1)
attributes on PA-RISC systems; change program's internal	chattr_pa(1)
attributes, change RCS file	rcs(1)
attributes, set and get for pthread	pthread_attr_getdetachstate(3T)
attributes, window, set and clear	stand(3X)
attributes; change login password and associated	passwd(1)
attributes; change program's internal	chattr(1)
attributes; manage processor set	pset_getattr(2)
attributes; search or kill processes based on process name and	pgrep(1)
attroff() - restricted window attribute control functions	attroff(3X)
attron() - restricted window attribute control functions	attroff(3X)
attrset() - restricted window attribute control functions	attroff(3X)
audctl() - start or halt the auditing system and set or get audit files	audctl(2)
audevent - change or display profile, event, or system call audit status	audevent(1M)
audeventstab - define and describe audit system events	audeventstab(4)
audible signal	beep(3X)
Audio - audio tools available through HP VUE (OBSOLETE)	Audio(5)
audio control panel (OBSOLETE)	Audio(5)
audio editor (OBSOLETE)	Audio(5)
audio file and data formats (OBSOLETE)	Audio(5)
audio file; convert an	convert(1)
audio file; describe an	attributes(1)
audio file; play an	send_sound(1)
audio library (OBSOLETE)	Audio(5)
audio on a workstation; OBSOLETE; control access to	asecure(1M)
audio security (OBSOLETE)	Audio(5)
audio server; start the	aserver(1M)
audio setup (OBSOLETE)	Audio(5)
audio tools available through HP VUE (OBSOLETE)	Audio(5)
audisp - display the requested audit information	audisp(1M)

Description	Entry Name(Section)
audit files; start or halt the auditing system and set or get	audctl(2)
audit ID (aid) for the current process; get the	getaudit(2)
audit ID (aid) for the current process; set the	setaudit(2)
audit information; display requested	audisp(1M)
audit - introduction to HP-UX Auditing System	audit(5)
audit overflow monitor daemon	audomon(1M)
audit process flag for calling process; get	getaudproc(2)
audit record for self-auditing process; write	audwrite(2)
audit records; determine time interval (in secs) for flushing	diskaudit_flush_interval(5)
audit status; change or display profile, event, or system call	audevent(1M)
audit subsystem; percentage of physical memory that can be used by	audit_memory_usage(5)
audit system events, define and describe	audeventstab(4)
audit.conf - file containing event mapping information	audit.conf(4)
audit: file format and other information for auditing	audit(4)
audit: set or get audit files	audctl(2)
audit: start or halt auditing system	audctl(2)
audit; get events and system calls currently being audited	getevent(2)
audit; select users to	audusr(1M)
audit; set current events and system calls to be audited	setevent(2)
audit_memory_usage - percentage of physical memory that can be used by audit subsystem	audit_memory_usage(5)
audit_site.conf - file containing site-specific event mapping information	audit.conf(4)
audit_track_paths - enable/disable tracking of current and root directories for auditing subsystem	audit_track_paths(5)
auditing level for the current process and its decendents	setaudproc(2)
auditing on the current process; suspend or resume	audswitch(2)
auditing subsystem; enable/disable tracking of current and root directories for	audit_track_paths(5)
auditing system and set or get audit files; start or halt the	audctl(2)
Auditing System; introduction to HP-UX	audit(5)
auditing system; set or display audit trail information	audsys(1M)
auditing system; start or halt	audsys(1M)
auditing the current process as owned by a given user, start	setauduser(3)
audomon - audit overflow monitor daemon	audomon(1M)
audswitch() - suspend or resume auditing on the current process	audswitch(2)
audsys - start/halt the auditing system; set/display audit trail information	audsys(1M)
audusr - select users to audit	audusr(1M)
audwrite() - write audit record for self-auditing process	audwrite(2)
auth_destroy() - library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
authadm - non-interactive editing of the authorization information in the RBAC databases	authadm(1M)
authcap - security databases for trusted systems	authcap(4)
authck - check internal consistency of Authentication database	authck(1M)
authdes_create() - obsolete library routines for RPC	rpc_soc(3N)
authdes_getucred() - library routines for secure remote procedure calls	secure_rpc(3N)
authdes_seccreate() - library routines for secure remote procedure calls	secure_rpc(3N)
authentication and authorization; secure internet services with Kerberos	sis(5)
authentication and print request server; PC-NFS	pcnfsd(1M)
authentication database for trusted systems; protected password	prpwd(4)
Authentication database; check internal consistency of	authck(1M)
authentication file format; PPP	ppp.Auth(4)
authentication information routines for PAM	pam_set_item(3)
authentication module, configuration file for pluggable authentication module	pam.conf(4)
authentication module, pluggable	pam(3)
authentication modules; user configuration file for pluggable	pam_user.conf(4)
authentication service, modify and delete user credentials for an authentication service	pam_setcred(3)
authentication transaction routines for PAM	pam_start(3)
authentication within the PAM framework, perform	pam_authenticate(3)
authentication, account, password, and session service module for HP-UX, extended	pam_hpsec(5)
authentication, account, session and password management PAM modules for UNIX	pam_unix(5)
authentication, account, session, and password management PAM modules for LDAP	pam_ldap(5)
authnone_create() - library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
authorization and privilege information in the privrun database; noninteractive editing of a	

Index

All Volumes

Description	Entry Name(Section)
command's	cmdprivadm(1M)
authorization checks and optionally reauthenticating the user; invoke another application with privileges after performing appropriate	privrun(1M)
authorization file; EVM	evm.auth(4)
authorization file; evmdaemon	evm.auth(4)
authorization information in the RBAC databases, non-interactive editing of the	authadm(1M)
authorization; PAM module that provides user	pam_authz(5)
authorization; secure internet services with Kerberos authentication and	sis(5)
authorized users edit files that are under access control; let	privedit(1M)
authsys_create() - library routines for client side remote procedure call authentication ..	rpc_clnt_auth(3N)
authsys_default() - library routines for client side remote procedure call authentication ..	rpc_clnt_auth(3N)
authunix_create() - obsolete library routines for RPC	rpc_soc(3N)
authunix_create_default() - obsolete library routines for RPC	rpc_soc(3N)
auto_parms - initial system configuration plus DHCP support command	auto_parms(1M)
autoboot sequence	pdcc(1M)
autochanger - SCSI interfaces for medium changer device	autochanger(7)
autofs	automount(1M)
autofs - file containing parameter values for automountd daemon and automount command	autofs(4)
automatic mount points; install	automount(1M)
automatic PCI Error Recovery; time interval, in minutes, between two PCI errors at a I/O slot that will result in	pci_error_tolerance_time(5)
automount command; file containing parameter values for	autofs(4)
automount - install automatic mount points	automount(1M)
automountd - autofs mount/unmount daemon	automountd(1M)
automountd daemon; file containing parameter values for	autofs(4)
autopush - manage system database of automatically pushed STREAMS modules	autopush(1M)
available disk space reaches threshold, suspend and resume accounting when	actsuspend(5)
available through HP VUE (OBSOLETE); audio tools	Audio(5)
awk - pattern-directed scanning and processing language	awk(1)
back into input stream, push character	ungetc(3S)
back into input stream, push wide character	ungetwc(3C)
back of the procedure call stack using the unwind library, produce a trace	U_STACK_TRACE(3X)
back up files; selectively	fbackup(1M)
background batch execution	at(1)
background character and rendition using a complex character	bkgrnd(3X)
background character and rendition using a single-byte character	bkgd(3X)
background process group	glossary(9)
background processes to complete, wait for	wait(1)
backing store pointer arithmetic	uwx_add_to_bsp(3X)
backlog value of the filter, for sendmail; sets the listen	smfi_setbacklog(3N)
backspaces and reverse line-feeds, remove from text	col(1)
backup	glossary(9)
backup - backup or archive the file system	backup(1M)
backup file, create or update LVM volume group configuration	vgcfgbackup(1M)
backup, incremental file system dump	dump(1M)
backup, incremental file system dump over network	dump(1M)
banner - make posters in large letters	banner(1)
base offset for an object file, get	elf_getbase(3E)
base-10 exponential functions	exp10(3M)
base-2 exponential functions	exp2(3M)
base-64 ASCII string, convert long integer to	a64(3C)
basename, dirname - extract portions of path names	basename(1)
basename() - return final component of path name	basename(3C)
basic integer data types	inttypes(5)
Basic Serial and HP AdvanceLink server	pcserver(1M)
bastille - system lockdown tool	bastille(1M)
bastille_drift - system configuration drift analyzer	bastille_drift(1M)
batch, at, and crontab queue description file	queuedefs(4)
batch - execute commands immediately	at(1)
batch mail interface	fastmail(1)
baud rate, get terminal	baudrate(3X)

Description	Entry Name(Section)
baud rate, tty, set or get	cfspeed(3C)
baudrate () - get terminal baud rate	baudrate (3X)
bc - arbitrary-precision arithmetic language	bc (1)
bcmp () - BSD memory compare	memory (3C)
bcopy () - BSD memory copy	memory (3C)
bdf - report number of free disk blocks (Berkeley version)	bdf (1M)
bdiff - big diff	bdiff (1)
beep () - audible signal	beep (3X)
beginning of file, list first few lines at	head (1)
behalf of an NFS client, clear locks held on	clear_locks (1M)
behavior on HP-UX; UNIX standards	standards (5)
behavior, advise system of process's expected paging	madvise (2)
Bessel functions of the first kind	j0 (3M)
Bessel functions of the second kind	y0 (3M)
bg - put jobs into background	sh-posix (1)
bgets () - read stream up to next delimiter	bgets (3G)
BGP routing daemon for IPv6	bgpd (1M)
bgpd - BGP routing daemon for IPv6	bgpd (1M)
big diff	bdiff (1)
bigcrypt () - generate hashing encryption on large strings	bigcrypt (3C)
binary directories; install object files in	cpset (1M)
binary executable(s); display security attributes of	getfilesec (1M)
binary file, convert to ASCII for transmission by mailer	uuencode (1)
binary file; set extended security attributes on a	setfilesec (1M)
binary files used by file system administration commands	fs_wrapper (5)
binary files, format tracing and logging	netfmt (1M)
binary input/output to a stream file; buffered	fread (3S)
binary or object file, find the printable strings in an	strings (1)
binary program files for given name, find location of	whereis (1)
binary search routine for sorted tables	bsearch (3C)
binary search tree, manage a	tsearch (3C)
bind a driver to a device	iobind (1M)
bind address to transport endpoint (X/OPEN TLI-XTI)	t_bind (3)
bind an address to a socket	bind (2)
bind () - bind an address to a socket	bind (2)
bind process or thread to a processor set	pset_bind (2)
bind services, library routines for RPC	rpcbind (3N)
bind threads to locality domain	pthread_processor_bind_np (3T)
bind threads to processors	pthread_processor_bind_np (3T)
bind to particular Network Information Service server	ypset (1M)
binder, and transfer processes; Network Information Service (NIS) server,	ypserv (1M)
biod - NFS daemon	biod (1M)
bit bucket	null (7)
bit bucket	zero (7)
bkgd () - set or get background character and rendition using a single-byte character	bkgd (3X)
bkgrnd () - set or get background character and rendition using omplex character	bkgrnd (3X)
blank lines, reduce multiple adjacent to single blank line	ssp (1)
blank lines, remove all from file	rmnl (1)
blmode - terminal block mode interface	blmode (7)
block	glossary (9)
block count and checksum of a file, print	sum (1)
block count and checksum of a file, print	sum (1)
block mode terminal interface	blmode (7)
block size, dump file system	dumpfs (1M)
block special file	glossary (9)
block, enable or disable during read	nodelay (3X)
blocked signals, examine and change	sigprocmask (2)
blocked signals, release and atomically wait for interrupt	sigpause (3C)
blocking on input, control	notimeout (3X)
blocking status of a message queue associated with a descriptor, set	mq_setattr (2)
boot	glossary (9)

Index

All Volumes

Description	Entry Name(Section)
boot area	glossary(9)
boot device configuration table	bootconf(4)
boot programs from disk; install, update or remove	mkboot(1M)
Boot Protocol server; Internet	bootpd(1M)
boot ROM	glossary(9)
boot - run bootstrap process	boot(1M)
boot the system	reboot(2)
boot time (OBSOLETE); enable or disable System V IPC messages at	mesg(5)
boot time, enable or disable System V IPC semaphores at	sema(5)
boot variables in stable storage; display and modify	setboot(1M)
boot, primary swap, or dump volume; prepare LVM logical volume to be root,	lvlinboot(1M)
boot-up	glossary(9)
bootconf - boot device configuration table	bootconf(4)
BOOTP server, send BOOTREQUEST to	bootpquery(1M)
bootpd, command line tools for DHCP elements of	dhcptools(1M)
bootpquery - send BOOTREQUEST to BOOTP server	bootpquery(1M)
bootptab entry, get or put	getbootpent(3X)
BOOTREQUEST, send to BOOTP server	bootpquery(1M)
bootstrap and installation utility, HP-UX	hpux(1M)
bootstrap for Itanium-based systems, HP-UX	hpux.efi(1M)
bootstrap process, run	boot(1M)
border() - draw borders from single-byte characters and renditions	border(3X)
border_set() - draw borders from complex characters and renditions	border_set(3X)
borders, draw from complex characters and renditions	border_set(3X)
borders, draw from complex characters and renditions	box_set(3X)
borders, draw from single-byte characters and renditions	border(3X)
borders, draw from single-byte characters and renditions	box(3X)
box() - draw borders from single-byte characters and renditions	box(3X)
box_set() - draw borders from complex characters and renditions	box_set(3X)
break a file into multiple <i>n</i> -line pieces	split(1)
break - exit from enclosing for, select, until, or while loop	sh-posix(1)
break - exit from enclosing for/next loop	csh(1)
break - exit from enclosing for/next loop	ksh(1)
break value and file size limits, get or set	ulimit(2)
breaksw - break from switch and resume after endsw	csh(1)
brk() , sbrk() - change data segment space allocation	brk(2)
broadcast message simultaneously to all users	wall(1M)
bs - a compiler/interpreter for modest-sized programs	bs(1)
BSD pseudo terminals (ptys), maximum number of	npty(5)
BSD-4.2-compatible kill() , and signal() system calls	bsdproc(3C)
BSD-compatible process control facilities, 4.2	killpg(2)
bsearch() - binary search routine for sorted tables	bsearch(3C)
bss (uninitialized data) allocation space of object files, print section sizes and	size(1)
btlan driver; network interface management command for	nwmgr_btlan(1M)
btmp - user login record format	utmp(4)
btmps database, write records into new wtmps and	bwtmps(3C)
btmps file	login(1)
btmps - user login information	wtmps(4)
btowc() - conversion between single-byte and wide character	btowc(3C)
bufcache_max_pct - OBSOLETE kernel tunable parameter	dbc_max_pct(5)
bufcall, maximum number of outstanding STREAMS	nstrent(5)
Buffer Cache Pages used by sendfile, maximum number of	sendfile_max(5)
buffer, free storage associated with	gss_release_buffer(3)
buffer; split into fields	bufsplit(3G)
buffered binary input/output to a stream file	fread(3S)
buffered input/output standard stream file package	stdio(3S)
buffering to a stream file; assign	setbuf(3S)
buffers, flush unwritten system buffers to disk	sync(1M)
buffers, periodically flush unwritten system buffers to disk	syncer(1M)
bufpages - OBSOLETE kernel tunable parameter	dbc_max_pct(5)
bufsplit() - split buffer into fields	bufsplit(3G)

Description	Entry Name(Section)
build a makefile	mkmf(1)
build and install Network Information Service databases	ypinit(1M)
build or access a binary search tree	tsearch(3C)
Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host	femsutil(1M)
bus address	glossary(9)
bwtmptime () - write records into new wtmps and btmps database	bwtmpts(3C)
bwtmpts - write records into new wtmps and btmps database	bwtmpts(3C)
byte order, network and host, convert values between	byteorder(3N)
byte; compare memory contents with specified	memory(3C)
byte; find location of in memory	memory(3C)
byte; set contents of memory area to specified	memory(3C)
bytes in a character; get number of	mbrlen(3C)
bytes in a file; count	wc(1)
bytes on a single System V IPC message queue, maximum number of	msgmnb(5)
bytes or characters in a file; count words, lines, and	wc(1)
bytes, swap	swab(3C)
bzero () - BSD memory clear	memory(3C)
C compiler; bundled	cc_bundled(1)
C header files, generate	rpcgen(1)
C language preprocessor	cpp(1)
C language, process include and conditional instructions	cpp(1)
C library, list of pthread calls for which the stubs are provided in the	pthread_stubs(5)
C macro processor	m4(1)
C source into a file; extract error messages from	mkstr(1)
C-Kermit 8.0 communications software for serial and network connections	kermit(1)
C-like syntax; a shell (command interpreter) with	cs(1)
cabinet LEDs; flash/turn off	fruled(1)
cabs () - complex absolute value function	cabs(3M)
cabsf () - complex absolute value function (float)	cabs(3M)
cabs1 () - complex absolute value function (long double)	cabs(3M)
cabsq () - complex absolute value function (quad)	cabs(3M)
cabsw () - complex absolute value function (extended)	cabs(3M)
cache display and control, IPv6 Neighbor Discovery	ndp(1M)
Cache (DNLC); number of locks for the Directory Name Lookup	dnlc_hash_locks(5)
Cache File System statistics	cachefstat(1M)
cache in the SCSI subsystem (OBSOLETE); enable and disable use of device's write	default_disk_ir(5)
cache of recently looked-up names, get entries from system	pstat(2)
Cache Pages used by sendfile, maximum number of Buffer	sendfile_max(5)
cache that can be consumed by sequential accesses, per system-wide limit; percentage of file	fcache_seqlimit_system(5)
cache that can be consumed by sequential accesses, per-file limit; percent of file	fcache_seqlimit_file(5)
cache the Kerberos ticket-granting ticket; obtain and	kinit(1)
cache; pack files and file systems	cachefspack(1M)
cached Kerberos tickets; list	klist(1)
CacheFS file systems; mount and unmount	mount_cachefs(1M)
CacheFS; administer disk space used for caching file systems with	cfsadmin(1M)
CacheFS; check integrity of data cached with	fsck_cachefs(1M)
cachefs; packing rules file	packingrules(4)
cachefspack - pack files and file systems in the cache	cachefspack(1M)
cachefsstat - Cache File System statistics	cachefsstat(1M)
caching and hashing daemon, password and group	pwgrd(1M)
caching and hashing statistics, password and group	pwgr_stat(1M)
caching file I/O data; maximum or minimum amount of physical memory used for	filecache_max(5)
caching file systems with CacheFS; administer	cfsadmin(1M)
cacos () - complex arccosine function	cacos(3M)
cacosf () - complex arccosine function (float)	cacos(3M)
cacosh () - complex arc hyperbolic cosine function	cacosh(3M)
cacoshf () - complex arc hyperbolic cosine function (float)	cacosh(3M)
cacosh1 () - complex arc hyperbolic cosine function (long double)	cacosh(3M)

Index

All Volumes

Description	Entry Name(Section)
cacoshq () - complex arc hyperbolic cosine function (quad)	cacosh (3M)
cacoshw () - complex arc hyperbolic cosine function (extended)	cacosh (3M)
cacosl () - complex arccosine function (long double)	cacos (3M)
cacosq () - complex arccosine function (quad)	cacos (3M)
cacosw () - complex arccosine function (extended)	cacos (3M)
cal - print calendar	cal (1)
calculate default disk section sizes	disksecn (1M)
calculator, desk	dc (1)
calendar - reminder service	calendar (1)
calendar; print	cal (1)
call an initialization routine only once	pthread_once (3T)
call another (UNIX) system, terminal emulator	cu (1)
call graph execution profile data, display	gprof (1)
call stack using the unwind library, produce a trace back of the procedure	U_STACK_TRACE (3X)
call terminal- spawn getty to remote terminal	ct (1)
call-back POSIX asynchronous I/O operations; percentage of physical memory lockable for request	aiophysmem_pct (5)
callback daemon; NFS Version 4	nfs4cbd (1M)
callback for context, specify	rpc_gss_set_callback (3N)
callback function; event management (EVM)	EvmCallback (5)
callback routines for stack unwind, register	uwx_register_callbacks (3X)
callbacks for sendmail; registers a set of filter	smfi_register (3N)
calling process, suspend	napms (3X)
calling process; get audit process flag for	getaudproc (2)
calloc () - allocate memory for array	malloc (3C)
callrpc () - obsolete library routines for RPC	rpc_soc (3N)
calls currently being audited; get events and system	getevent (2)
calls for which the stubs are provided in the C library, list of pthread	pthread_stubs (5)
calls to be audited; set current events and system	setevent (2)
calls without error checking; execute link () and unlink () system	link (1M)
calls, library routines for secure remote procedure	secure_rpc (3N)
calls, system, BSD-4.2-compatible kill (), and signal ()	bsdproc (3C)
can_change_color () - color manipulation functions	can_change_color (3X)
cancel a notification request with a message queue; register or	mq_notify (2)
cancel a per-process timer	rmtimer (3C)
cancel asynchronous I/O	aiocancel (2)
cancel - cancel requests on an LP printer	lp (1)
cancel execution of a thread	pthread_cancel (3T)
cancel LP requests from spooling queue on remote system	rcancel (1M)
cancel online cell operation; monitor online cell operation; reset hung cell during cell activation; online activation of a cell from nPartition;	parolrad (1M)
cancel pending changes to complex or partition configuration data; unlock stable complex profile or	parunlock (1M)
cancel requests on an LP printer	lp (1)
Cancel Safe	thread_safety (5)
cancelability state and type, set and retrieve the current thread's	pthread_setcancelstate (3T)
cancellation cleanup handler, register or remove thread	pthread_cleanup_pop (3T)
Cancellation Points	thread_safety (5)
cancellation requests, process any pending	pthread_testcancel (3T)
capabilities, terminal, get from terminfo database	tput (1)
captainfo - convert a termcap description into a terminfo description	captainfo (1M)
card access information, network I/O	lan (7)
cards and Online Addition of I/O chassis; command for Online Addition/Replacement/Deletion of PCI I/O	olrad (1M)
carg () - complex argument function	carg (3M)
cargf () - complex argument function (float)	carg (3M)
cargl () - complex argument function (long double)	carg (3M)
cargq () - complex argument function (quad)	carg (3M)
cargw () - complex argument function (extended)	carg (3M)
carriage control characters; interpret ASA	asa (1)
case - execute commands based on pattern match	sh-posix (1)

Description	Entry Name(Section)
case - execute <i>list</i> associated with <i>pattern</i> that matches <i>word</i>	ksh(1)
case - label in a switch statement	cs(1)
casin() - complex arcsine function	casin(3M)
casinf() - complex arcsine function (float)	casin(3M)
casinh() - complex arc hyperbolic sine function	casinh(3M)
casinhf() - complex arc hyperbolic sine function (float)	casinh(3M)
casinh1() - complex arc hyperbolic sine function (long double)	casinh(3M)
casinhq() - complex arc hyperbolic sine function (quad)	casinh(3M)
casinhw() - complex arc hyperbolic sine function (extended)	casinh(3M)
casinl() - complex arcsine function (long double)	casin(3M)
casinq() - complex arcsine function (quad)	casin(3M)
casinw() - complex arcsine function (extended)	casin(3M)
cat after uncompacting Huffman coded files (see pack)	compact(1)
cat and whatis files for online manpages; create	catman(1M)
cat - concatenate, copy, and print files	cat(1)
catalog file, generate a formatted message	genscat(1)
catalog file, message, create for modification	findmsg(1)
catalog path, configure message	chnlspath(1M)
catalog, set the default message	setcat(3)
catalogs, message, find strings for inclusion in	findstr(1)
catan() - complex arctangent function	catan(3M)
catanf() - complex arctangent function (float)	catan(3M)
catanh() - complex arc hyperbolic tangent function	catanh(3M)
catanhf() - complex arc hyperbolic tangent function (float)	catanh(3M)
catanh1() - complex arc hyperbolic tangent function (long double)	catanh(3M)
catanhq() - complex arc hyperbolic tangent function (quad)	catanh(3M)
catanhw() - complex arc hyperbolic tangent function (extended)	catanh(3M)
catanl() - complex arctangent function (long double)	catan(3M)
catanq() - complex arctangent function (quad)	catan(3M)
catanw() - complex arctangent function (extended)	catan(3M)
catclose() - close message catalog for reading	catopen(3C)
categories of events; lists different	evweb_list(1)
catgets(3C) , insert calls to based on findstr(1) output	insertmsg(1)
catgets() - get an NLS program message	catgets(3C)
catman - create cat and whatis files for online manpages	catman(1M)
catopen() - open message catalog for reading	catopen(3C)
cause the calling thread to terminate	pthread_exit(3T)
cblocks for pty and tty data transfers; number of	nclist(5)
cbreak() - input mode control functions	cbreak(3X)
cbrt() - cube root function	cbrt(3M)
cbrtf() - cube root function (float)	cbrt(3M)
cbrtl() - cube root function (long double)	cbrt(3M)
cbrtq() - cube root function (quad)	cbrt(3M)
cbrtw() - cube root function (extended)	cbrt(3M)
cc - bundled C compiler	cc_bundled(1)
cc_bundled - bundled C compiler	cc_bundled(1)
ccat - uncompact and cat files using Huffman code (see pack)	compact(1)
cchar_t ; get a wide-character string and rendition from	getcchar(3X)
cchar_t from a wide-character string and rendition; set	setcchar(3X)
ccNUMA system, returns system-wide or per-process information of a	pstat_getlocality(2)
ccos() - complex cosine function	ccos(3M)
ccosf() - complex cosine function (float)	ccos(3M)
ccosh() - complex hyperbolic cosine function	ccosh(3M)
ccoshf() - complex hyperbolic cosine function (float)	ccosh(3M)
ccosh1() - complex hyperbolic cosine function (long double)	ccosh(3M)
ccoshq() - complex hyperbolic cosine function (quad)	ccosh(3M)
ccoshw() - complex hyperbolic cosine function (extended)	ccosh(3M)
ccosl() - complex cosine function (long double)	ccos(3M)
ccosq() - complex cosine function (quad)	ccos(3M)
ccosw() - complex cosine function (extended)	ccos(3M)
cd - change working directory	cd(1)

Index

All Volumes

Description	Entry Name(Section)
cd - change working directory	cs (1)
cd - change working directory	ksh (1)
cd - change working directory	sh-posix (1)
CD-ROM: background information	cdrom (4)
CD-ROM: format of a CDFS cnode	cdnode (4)
cdc - change the delta commentary of an SCCS delta	cdc (1)
CDFS cnode, format of a	cdnode (4)
CDFS file system disk blocks, report number of free	df_hfs (1M)
CDFS file systems; mount and unmount	mount_cdfs (1M)
cdnode - format of a CDFS cnode	cdnode (4)
cdrom - CD-ROM background information	cdrom (4)
ceil () - ceiling function	ceil (3M)
ceilf () - ceiling function (float)	ceil (3M)
ceiling functions	ceil (3M)
ceil1 () - ceiling function (long double)	ceil (3M)
ceilq () - ceiling function (quad)	ceil (3M)
ceilw () - ceiling function (extended)	ceil (3M)
cell activation; online activation of a cell from nPartition; cancel online cell operation; monitor online cell operation; reset hung cell during	parolrad (1M)
cell from nPartition; cancel online cell operation; monitor online cell operation; reset hung cell during cell activation; online activation of a	parolrad (1M)
cell LEDs; flash/turn off	fruled (1)
cell operation; monitor online cell operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel online	parolrad (1M)
cell operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel online cell operation; monitor online	parolrad (1M)
cell-based HP-UX servers; physical memory allocation policy on	numa_policy (5)
cells and I/O chassis; turn on/off or display current status of power for	frupower (1M)
cells; turn on/off or display current status of power	frupower (1M)
cent - Centronics-compatible interface	cent (7)
Centronics-compatible interface	cent (7)
CER (Common Error Repository); provide displaying options for HP-UX errors defined in the	emtui (1)
cerupdate - update the Common Error Repository (CER) with error metadata	cerupdate (1)
cexp () - complex exponential function	cexp (3M)
cexpf () - complex exponential function (float)	cexp (3M)
cexpl () - complex exponential function (long double)	cexp (3M)
cexpq () - complex exponential function (quad)	cexp (3M)
cexpw () - complex exponential function (extended)	cexp (3M)
cfgetispeed () - get tty input baud rate	cfspeed (3C)
cfsetispeed () - get tty output baud rate	cfspeed (3C)
cfsadmin - administer disk space used for caching file systems with CacheFS	cfsadmin (1M)
cfsetispeed () - set tty input baud rate	cfsetispeed (3C)
cfsetospeed () - set tty output baud rate	cfsetospeed (3C)
ch_rc - change system configuration file	ch_rc (1M)
chacl - add, modify, delete, copy, or summarize file access control lists (ACLs)	chacl (1)
change a user's Kerberos password	kpasswd (1)
change characteristics of physical volume in LVM volume group	pvchange (1M)
change core file settings of a process	coreadm (1M)
change core file settings of a process	coreadm (2)
change current login to a new group	newgrp (1)
change data format of and copy a (tape) file	dd (1)
change data segment space allocation	brk (2)
change default login shell	chsh (1)
change delta commentary of an SCCS delta	cdc (1)
change (delta) to an SCCS file, make a	delta (1)
change file mode access permissions	chmod (1)
change file mode access permissions	chmod (2)
change file owner or group	chown (1)
change global search path for dynamically loadable kernel modules	modpath (2)
change login name	su (1)

Description	Entry Name(Section)
change login password	passwd(1)
change login password and associated attributes	passwd(1)
change login password in Network Information System (NIS)	yppasswd(1)
change LVM logical volume characteristics	lvchange(1M)
change machine information	setuname(1M)
change or add a variable to environment	setenv(3C)
change or add value to environment	putenv(3C)
change or examine blocked signals	sigprocmask(2)
change or examine signal action	sigwait(2)
change or query stream configuration	strchg(1M)
change or reformat a text file	newform(1)
change owner and group of a file	chown(2)
change owner and/or group in access control list (ACL)	chownacl(3C)
change priority of a process	nice(2)
change priority of running processes	renice(1M)
change processor set assignment	pset_assign(2)
change program's internal attributes	chattr(1)
change program's internal attributes on Integrity systems	chattr_ia(1)
change program's internal attributes on PA-RISC systems	chattr_pa(1)
change RCS file attributes	rcs(1)
change real-time priority	rtprio(2)
change renditions of characters in a window	chgat(3X)
change root directory	chroot(2)
change root directory for a command	chroot(1M)
change selected characters	tr(1)
change service, QOP for a session	rpc_gss_set_defaults(3N)
change signal action	sigaction(2)
change state, wait for child process to	wait3(2)
change state, wait for child process to	wait4(2)
change state, wait for child process to	waitid(2)
change system configuration file	ch_rc(1M)
change the default stacksize	pthread_default_stacksize_np(3T)
change the default stacksize.	pthread_default_rsestacksize_np(3T)
change the name of a file	mv(1)
change the name of a file	rename(2)
change the signal mask of the calling thread	pthread_sigmask(3T)
change times of file; update access, modification, and/or	touch(1)
change user information used by finger command	chfn(1)
change user's secure RPC key	chkey(1)
change window ID of running program or start program in particular memory window	setmemwindow(1M)
change working directory	cd(1)
change working directory	chdir(2)
change WU-FTPD group access file information	privatepw(1)
changer device driver, SCSI media	autochanger(7)
changes NIS information	ypupdate(3C)
changes or deletes a message header	smfi_chgheader()(3N)
changes per System V IPC semop() call, maximum cumulative value	semaem(5)
changing NIS information, server for	ypupdated(1M)
channel configuration file; EVM	evmchannel.conf(4)
Channel (Fibre) Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters	fcmsutil(1M)
Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre	fcmsutil(1M)
channel manager; Event Manager	evmchmgr(1M)
channel; create an interprocess	pipe(2)
chanq_hash_locks - size of hashed pool of spinlocks protecting the channel queue hash tables	chanq_hash_locks(5)
character	glossary(9)
character and rendition to a window, add a complex	add_wch(3X)
character and rendition, complex, input from a window	in_wch(3X)
character and rendition, complex, insert into a window	ins_wch(3X)

Index

All Volumes

Description	Entry Name(Section)
character and rendition, input a single-byte from a window	inch(3X)
character and rendition, single-byte, insert into a window	insch(3X)
character back into input stream, push	ungetc(3S)
character code set, convert to another	iconv(3C)
character codeset conversion	iconv(1)
character device special file, control	ioctl(2)
character or word from a stream file; get	getc(3S)
character or word, put on a stream	putc(3S)
character rendition, write and immediately refresh the pad	pechochar(3X)
character (restartable); convert a wide-character code to a	wcrctomb(3C)
character sequences for display/keyboard, convert file data order	forder(1)
character set	glossary(9)
character special file	glossary(9)
character string and rendition from a cchar_t ; get a wide-	getchar(3X)
character string operations	string(3C)
character string operations; wide	wcstring(3C)
character string or stream file; read from with formatted input conversion	scanf(3S)
character string to a wide-character string (restartable); convert a	mbsrtowcs(3C)
character string, multi-byte, input from a window	innstr(3X)
character to a wide-character code; convert	mbrtowc(3C)
character transliteration	towctrans(3C)
character, generate printable representation of	unctrl(3X)
character, get a multi-byte character length limited string from the terminal	getnstr(3X)
character, get a multi-byte character string from the terminal	getstr(3X)
character, get a wide character from a terminal	get_wch(3X)
character, insert a wide-character string into a window	ins_nwstr(3X)
character, multi-byte, insert into a window	insnstr(3X)
character, push onto the input queue	ungetch(3X)
character, single-byte, get from the terminal	getch(3X)
character-set translation	kermit(1)
character-string login name of the user, get	cuserid(3S)
character; conversion between single-byte and wide	btowc(3C)
character; get number of bytes in	mbrlen(3C)
characteristics of a disk device, describe	diskinfo(1M)
characteristics of physical volume in LVM volume group, change	pvchange(1M)
characteristics, change LVM logical volume	lvchange(1M)
characters according to type; classify	ctype(3C)
characters and function key codes from a terminal; get an array of wide	getn_wstr(3X)
characters and renditions, an array of single-byte, input from a window	inchnstr(3X)
characters and renditions, complex, draw lines from	hline_set(3X)
characters and renditions, draw lines from single-byte	hline(3X)
characters and strings conversions; multibyte	multibyte(3C)
characters in a file, unprintable and non-ASCII, make visible or invisible	vis(1)
characters in a file; count	wc(1)
characters in a file; count words, lines, and bytes or	wc(1)
characters, alter, delete, modify, substitute, or translate	tr(1)
characters, how to type control	ascii(5)
characters, renditions of, change in a window	chgat(3X)
characters, translate to upper-case, lower-case, or 7-bit ASCII	conv(3C)
characters, wide, input a string of, from a window	innwstr(3X)
characters; classify wide	wctype(3C)
characters; interpret ASA carriage control	asa(1)
chargefee - charge fee to user based on system usage	acctsh(1M)
charmmap - symbolic translation file for localedef scripts	charmmap(4)
chassis; command for Online Addition/Replacement/Deletion of PCI I/O cards and Online Addition of I/O	olrad(1M)
chassis; turn on/off or display current status of power for cells and I/O	frupower(1M)
chatr - change program's internal attributes	chatr(1)
chatr - change program's internal attributes on Integrity systems	chatr_ia(1)
chatr - change program's internal attributes on PA-RISC systems	chatr_pa(1)
chatr_ia - change program's internal attributes on Integrity systems	chatr_ia(1)

Description	Entry Name(Section)
chattr_pa - change program's internal attributes on PA-RISC systems	chattr_pa(1)
chdir - change current working directory	csh(1)
chdir() - change working directory	chdir(2)
check if disk volume is under HP Logical Volume Manager (LVM) control	lvmchk(1M)
check if system has been converted to a trusted system	iscomsec(2)
check in RCS revisions	ci(1)
check integrity of data cached with CacheFS	fsck_cacheofs(1M)
check internal consistency of Authentication database	authchk(1M)
check memory region for validity	mvalid(3)
check nroff/troff files	checknr(1)
check on an event; perform a data integrity	EvmEventValidate(3)
check or print documents formatted with the mm macros	mm(1)
check or repair a physical volume in LVM volume group	pvck(1M)
check out RCS revisions	co(1)
check security-bulletin compliance state of HP-UX 11.x system or depot	security_patch_check(1M)
check status of local user accounts	userstat(1M)
check top wrapper configuration	tcpdchk(1)
check the /etc/shadow file; install, update or	pwconv(1M)
check the network, scatter data to	spray(3N)
check the uucp directories and permissions file	uuchek(1M)
check_patches - HP-UX 11i V3 patch check utility	check_patches(1M)
checker, file system quota consistency	quotacheck(1M)
checkers; password/group file	pwck(1M)
checking tool; named configuration file syntax	named-checkconf(1)
checking tool; zone validity	named-checkzone(1)
checking; copy HFS file system with label	volcopy_hfs(1M)
checknr - check nroff/troff files	checknr(1)
checks and optionally reauthenticating the user; invoke another application with privileges after performing appropriate authorization	privrun(1M)
checks the consistency of compartment rules for files with multiple hardlinks	vhardlinks(1M)
checksum and block count of a file, print	sum(1)
checksum and block count of a file, print	sum(1)
cheduling policy and associated parameters, get and set	pthread_getschedparam(3T)
chfn - change user information used by finger command	chfn(1)
chgat() - change renditions of characters in a window	chgat(3X)
chgrp - change group of file	chown(1)
child process	glossary(9)
child process	fork(2)
child process and process times; get	times(2)
child process to change state, wait for	wait3(2)
child process to change state, wait for	wait4(2)
child process to stop or terminate; wait for	wait(2)
child process, wait to change state	waitid(2)
children; synchronize a window with its parents or	syncok(3X)
chkey - change user's secure RPC key	chkey(1)
chmod() - change file mode access permissions	chmod(2)
chmod - change file mode access permissions	chmod(1)
chnlspath - configure message catalog path.	chnlspath(1M)
chown - change file owner	chown(1)
chown() - change owner and group of a file	chown(2)
chownacl() - change owner and/or group in access control list (ACL)	chownacl(3C)
chroot() - change root directory	chroot(2)
chroot - change root directory for a command	chroot(1M)
chsh - change default login shell	chsh(1)
chunk size in 1 KB blocks; swap	swchunk(5)
ci - check in RCS revisions	ci(1)
cimag() - complex imaginary-part function	cimag(3M)
cimagf() - complex imaginary-part function (float)	cimag(3M)
cimagl() - complex imaginary-part function (long double)	cimag(3M)
cimagq() - complex imaginary-part function (quad)	cimag(3M)
cimagw() - complex imaginary-part function (extended)	cimag(3M)

Index

All Volumes

Description	Entry Name(Section)
circuit, X.25 switched virtual, clear	clrsvc(1M)
cis () - cosine plus i times sine	cis(3M)
cisf () - cosine plus i times sine (float)	cis(3M)
cisl () - cosine plus i times sine (long double)	cis(3M)
cisq () - cosine plus i times sine (quad)	cis(3M)
cisw () - cosine plus i times sine (extended)	cis(3M)
ckconfig - verify path names of all FTP configuration files	ckconfig(1)
ckpacct - check size of process accounting file	acctsh(1M)
cksum - print file checksum and sizes	sum(1)
class driver eschgr plug-in for scsimgr; SCSI	scsimgr_eschgr(7)
class driver esdisk plug-in for scsimgr; SCSI	scsimgr_esdisk(7)
class driver estape plug-in for scsimgr; SCSI	scsimgr_estape(7)
class-dependent data translation of ELF files	elf_xlate(3E)
class-dependent object file header for elf32 or elf64 file; retrieve	elf_getehdr(3E)
class-dependent program header table for ELF files, retrieve	elf_getphdr(3E)
class-dependent section header for ELF files, retrieve	elf_getshdr(3E)
classes of kernel memory pages are not dumped when a kernel panic occurs, defines which	dontdump(5)
classification macro, floating-point	fpclassify(3M)
classify characters according to type	ctype(3C)
classify wide characters	wctype(3C)
clean-up, uucp spool directory	uucleanup(1M)
cleanup - HP-UX patch cleanup utility	cleanup(1M)
clear a window	clear(3X)
clear () - clear a window	clear(3X)
clear - clear terminal screen	clear(1)
clear from cursor to end of line	clrtoeol(3X)
clear from cursor to end of window	clrtobot(3X)
clear inode	clri(1M)
clear locks held on behalf of an NFS client	clear_locks(1M)
clear the process environment	clearenv(3C)
clear window attributes	standend(3X)
clear X.25 switched virtual circuit	clrsvc(1M)
clear_locks - clear locks held on behalf of an NFS client	clear_locks(1M)
clearenv () - clear the process environment	clearenv(3C)
clearerr () - clear I/O error on stream	ferror(3S)
clearerr_unlocked () - stream status inquiries	ferror(3S)
clearok () - terminal output control functions	clearok(3X)
client configuration file, PPPoE (Point to Point Protocol over Ethernet)	pppoe.conf(4)
client configuration information file, diskless	info(4)
client daemon process; LDAP	ldapclntd(1M)
client daemon, DHCPv6	dhcpv6clntd(1M)
Client for Dynamic Host Configuration Protocol Server	dhcpcclient(1M)
CLIENT handles, library routines for dealing with creation and manipulation of	rpc_clnt_create(3N)
client interface for requesting configuration parameters from the DHCPv6 server, DHCPv6	dhcpv6client_ui(1)
client interface; Network Information Service	ypclnt(3C)
client libraries; Kerberos	libkrb5(3)
client side, library routines for client side calls, rpc	rpc_clnt_calls(3N)
client, clear locks held on behalf of an NFS	clear_locks(1M)
client, get credentials of	rpc_gss_getcred(3N)
client, library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
client, PPPoE (Point to Point Protocol over Ethernet)	pppoe(1)
clients, directories to export to NFS	exports(4)
clnt_broadcast () - obsolete library routines for RPC	rpc_soc(3N)
clnt_call () - library routines for client side calls	rpc_clnt_calls(3N)
clnt_control () - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_create () - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_create_vers () - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_destroy () - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_dg_create () - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)

Description	Entry Name(Section)
clnt_freeres() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_geterr() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_pcreateerror() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_perrno() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_perror() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_raw_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_spccreateerror() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_sperrno() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_sperror() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_tli_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_tp_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_vc_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clntraw_create() - obsolete library routines for RPC	rpc_soc(3N)
clnttcp_create() - obsolete library routines for RPC	rpc_soc(3N)
clntudp_bufcreate() - obsolete library routines for RPC	rpc_soc(3N)
clntupd_create() - obsolete library routines for RPC	rpc_soc(3N)
clock daemon	cron(1M)
clock operations	clocks(2)
clock() - report CPU time used	clock(3C)
clock resolution, get	clocks(2)
clock tick	glossary(9)
clock ticks per second, scheduling interval in	timeslice(5)
clock time value, get	clocks(2)
clock time value, set	clocks(2)
clock, get current value of system-wide	getclock(3C)
clock, set value of system-wide	setclock(3C)
clock; correct the time to synchronize the system	adjtime(2)
clock_getres() - get clock resolution	clocks(2)
clock_gettime() - get clock time value	clocks(2)
clock_settime() - set clock time value	clocks(2)
clocks - clock operations	clocks(2)
clog() - complex logarithm function	clog(3M)
clogf() - complex logarithm function (float)	clog(3M)
clogl() - complex logarithm function (long double)	clog(3M)
clogq() - complex logarithm function (quad)	clog(3M)
clogw() - complex logarithm function (extended)	clog(3M)
clone driver: STREAMS driver	clone(7)
clone - open a major and minor device pair on a STREAMS driver	clone(7)
cloned DLPI streams allowed on the system; maximum number of	dlpi_max_clones(5)
close a crash dump descriptor	cr_close(3)
close a message queue descriptor	mq_close(2)
close a named semaphore	sem_close(2)
close a shared object	dlclose(3C)
close and open message catalog for reading	catopen(3C)
close() - close a file descriptor	close(2)
close legal user shells file	getusershell(3C)
close or flush a stream	fclose(3S)
close or open pipe I/O to or from a process	popen(3S)
close() - STREAMS enhancements to standard system calls	stream(2)
close system log file	syslog(3C)
close transport endpoint (X/OPEN TLI-XTI)	t_close(3)
close_secdef() - security defaults configuration file routines	secdef(3)
closedir() - close a currently open directory	directory(3C)
closelog() - control system log	syslog(3C)
closewtmp - overview of accounting and miscellaneous accounting commands	acct(1M)
clri - clear inode	clri(1M)
clrsvc - clear X.25 switched virtual circuit	clrsvc(1M)
clrtobot() - clear from cursor to end of window	clrtobot(3X)
clrtoeol() - clear from cursor to end of line	clrtoeol(3X)
clusters; report number of free disk	dosdf(1)
cmdprivadm - noninteractive editing of a command's authorization and privilege information in the	

Index

All Volumes

Description	Entry Name(Section)
<code>privrun</code> database	<code>cmdprivadm</code> (1M)
<code>cmp</code> - compare two files	<code>cmp</code> (1)
<code>cmpt_change()</code> - set and get process' compartment	<code>cmpt_change</code> (3)
<code>cmpt_endent()</code> - map compartment name to number or number to name	<code>cmpt_getbynum</code> (3)
<code>cmpt_get()</code> - set and get process' compartment	<code>cmpt_change</code> (3)
<code>cmpt_get_addrclid()</code> - get the compartment IDs associated with a network interfaces	<code>cmpt_get_ifcid</code> (3)
<code>cmpt_get_endpoint_cid()</code> - get the compartment IDs of socket endpoints.	<code>cmpt_get_peer_cid</code> (3)
<code>cmpt_get_ifcid()</code> - get the compartment IDs associated with a network interfaces	<code>cmpt_get_ifcid</code> (3)
<code>cmpt_get_peer_cid()</code> - get the compartment IDs of socket endpoints.	<code>cmpt_get_peer_cid</code> (3)
<code>cmpt_getbyname()</code> - map compartment name to number or number to name	<code>cmpt_getbynum</code> (3)
<code>cmpt_getbynum()</code> - map compartment name to number or number to name	<code>cmpt_getbynum</code> (3)
<code>cmpt_getent()</code> - map compartment name to number or number to name	<code>cmpt_getbynum</code> (3)
<code>cmpt_setent()</code> - map compartment name to number or number to name	<code>cmpt_getbynum</code> (3)
<code>cmpt_tune</code> - query, enable, or disable compartmentalization feature	<code>cmpt_tune</code> (1M)
<code>co</code> - check out RCS revisions	<code>co</code> (1)
code files, object, in a library, find optimum sequence for	<code>lorder</code> (1)
code set conversion, character	<code>iconv</code> (3C)
code to a character (restartable); convert a wide-character	<code>wcrtomb</code> (3C)
code widths; set and get EUC for ldterm	<code>eucset</code> (1)
code, processor-dependent (firmware)	<code>pdcc</code> (1M)
code; compress and expand files using Huffman	<code>pack</code> (1)
code; format text version of EVM status	<code>EvmStatusTextGet</code> (3)
coded character set	<code>glossary</code> (9)
codes from a terminal; get an array of wide characters and function key	<code>getn_wstr</code> (3X)
codes; Service Location Protocol (SLP) error	<code>SLPError</code> (3N)
codeset conversion	<code>iconv</code> (1)
codeset conversion routines	<code>iconv</code> (3C)
codeset conversion; character	<code>iconv</code> (1)
<code>col</code> - filter reverse line-feeds and backspaces from text	<code>col</code> (1)
collating element	<code>glossary</code> (9)
collation	<code>glossary</code> (9)
collation sequence	<code>glossary</code> (9)
collect system diagnostic messages to form error log	<code>dmesg</code> (1M)
color manipulation functions	<code>can_change_color</code> (3X)
<code>color_content()</code> - color manipulation functions	<code>can_change_color</code> (3X)
<code>color_set()</code> - window attribute control functions	<code>attr_get</code> (3X)
<code>COLS()</code> - number of columns on terminal screen	<code>COLS</code> (3X)
columns, number of, on terminal screen	<code>COLS</code> (3X)
<code>comb</code> - combine SCCS deltas	<code>comb</code> (1)
combine corresponding lines of several files or subsequent lines of one file	<code>paste</code> (1)
combine SCCS deltas	<code>comb</code> (1)
combine two LVM logical volumes into one logical volume	<code>lvmerge</code> (1M)
Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE	<code>fcmstutil</code> (1M)
<code>comm</code> - select/reject lines common to two sorted files	<code>comm</code> (1)
command	<code>glossary</code> (9)
<code>command</code> - execute command without lookup	<code>sh-posix</code> (1)
command execution, set (modify or redefine) environment for	<code>env</code> (1)
command execution, UNIX system to UNIX system	<code>uux</code> (1)
command for LAN and RDMA interfaces; network interface management	<code>nwmgr</code> (1M)
command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility	<code>fcmstutil</code> (1M)
command history for interactive programs; input editor and	<code>ied</code> (1)
command interpreter	<code>glossary</code> (9)
command interpreter (shell) with C-like syntax	<code>csh</code> (1)
Command Line Interface; display information about the Partition	<code>partition</code> (5)
command line of a process, get	<code>pstat</code> (2)
command line tool for DHCP elements of bootpd	<code>dheptools</code> (1M)
command on a remote host, execute	<code>on</code> (1)
command options, parse	<code>getopts</code> (1)

Description	Entry Name(Section)
command options; parse	getopt(1)
Command Set 1980	glossary(9)
command shells; standard and restricted POSIX.2-conformant	sh-posix(1)
command summary from per-process accounting records	acctcms(1M)
command's authorization and privilege information in the privrun database; noninteractive editing of a	cmdprivadm(1M)
command, change root directory	chroot(1M)
command, fix manpages for faster viewing with man	fixman(1M)
command, report execution time of, process accounting data and system activity	time(1)
command, run at nondefault priority	nice(1)
command, run immune to hangups	nohup(1)
command, shell, issue a	system(3S)
command; change user information used by finger	chfn(1)
command; construct argument lists and execute	xargs(1)
command; execute a simple	command(1)
command; measure time used to execute a	time(1)
command; return stream to a remote	rexec(3N)
command; time a	time(1)
commands for sharing resources across a network; file containing	dfstab(4)
commands to the Terminal Session Manager, TSM; send	tsm.command(1)
commands, file system administration configuration and binary files	fs_wrapper(5)
commands, install new	install(1M)
commands, output to the terminal	putp(3X)
commands, show last executed in reverse order	lastcomm(1)
commands: STREAMS ioctl commands	streamio(7)
commands; ask for help on SCCS	sccshelp(1)
commands; description of RCS	rcsintro(5)
commands; execute at a later time	at(1)
commands; generic device control	ioctl(5)
commentary of an SCCS delta, change delta	cdc(1)
common archive file format	ar(4)
Common Error Repository (CER); provide displaying options for HP-UX errors defined in the	emtui(1)
Common Error Repository (CER); update with error metadata	cerupdate(1)
common HP-UX terms; description of	glossary(9)
common logarithm functions	log10(3M)
common to two sorted files, reject/select lines	comm(1)
communicate interactively with another user	write(1)
communication domain protocol, local	UNIX(7P)
communication facilities, interprocess, report status	ipcs(1)
communication facilities; report status of POSIX interprocess	pipes(1)
communication identifier, create interprocess	ftok(3C)
communication; create an endpoint for	socket(2)
communications software for serial and network connections	kermit(1)
communications, Interprocess	socket(7)
compact - compact files using Huffman code (see pack)	compact(1)
compact files using Huffman code (see pack)	compact(1)
compact list of users currently on the system	users(1)
compaction; copy HFS file system with	dcopy(1M)
comparator; HP-UX installed software	sysdiff(1)
compare contents of memory with byte	memory(3C)
compare contents of two directories	dircmp(1)
compare or print out terminfo descriptions	infocmp(1M)
compare RCS revisions	rcsdiff(1)
compare sorted files; reject/select common lines	comm(1)
compare three files and find differences	diff3(1)
compare two files	cmp(1)
compare two files and find differences	diff(1)
compare two files and mark differences	diffmk(1)
compare two files and show differences side-by-side	sdiff(1)
compare two strings	string(3C)
compare two thread identifiers	pthread_equal(3T)

Index

All Volumes

Description	Entry Name(Section)
compare two versions of an SCCS file	sccsdiff(1)
comparison macro, floating-point (<)	isless(3M)
comparison macro, floating-point (<=)	islessequal(3M)
comparison macro, floating-point (<>)	islessgreater(3M)
comparison macro, floating-point (>)	isgreater(3M)
comparison macro, floating-point (>=)	isgreaterequal(3M)
comparison macro, floating-point (unordered)	isunordered(3M)
comparison routines for regular expressions	regexp(3X)
compartment IDs associated with a network interfaces; get the	cmpt_get_ifcid(3)
compartment IDs of socket endpoints.; get the	cmpt_get_peer_cid(3)
compartment name to number or number to name; map	cmpt_getbynum(3)
compartment rules for files with multiple hardlinks; checks the consistency of	vhardlinks(1M)
compartment rules; display	getrules(1M)
compartment rules; set	setrules(1M)
compartment; set and get	cmpt_change(3)
compartmentalization feature; query, enable, or disable	cmpt_tune(1M)
compartments - description of HP-UX compartments	compartments(5)
compartments - HP-UX compartments files	compartments(4)
compartments; description of HP-UX	compartments(5)
compartments; HP-UX	compartments(4)
compatibility; terminal interface for Version 6/PWB	sttyv6(7)
compile and match routines for regular expressions	regexp(3X)
compile() - regular expression compile routine	regexp(3X)
compiled terminfo file format	term(4)
compiler footprint records; summarize information from	footprints(1)
compiler/interpreter for modest-sized programs	bs(1)
compiler; bundled C	cc_bundled(1)
compilers, rpcgen; generate RPC protocols, C header files	rpcgen(1)
compilers: terminfo data base compiler	tic(1M)
compiling routines, regular expression	regcomp(3C)
complementary error functions	erf(3M)
complete, wait for background processes to	wait(1)
complex absolute value functions	cabs(3M)
complex arc hyperbolic cosine functions	cacosh(3M)
complex arc hyperbolic sine functions	casinh(3M)
complex arc hyperbolic tangent functions	catanh(3M)
complex arccosine functions	cacos(3M)
complex arcsine functions	casin(3M)
complex arctangent functions	catan(3M)
complex argument functions	carg(3M)
complex character and rendition, add to a window	add_wch(3X)
complex character and rendition, input from a window	in_wch(3X)
complex character and rendition, insert into a window	ins_wch(3X)
complex character, set or get background character and rendition using	bkgrnd(3X)
complex character, write and immediately refresh the window	echo_wchar(3X)
complex characters and renditions, add an array of, to a window	add_wchnstr(3X)
complex characters and renditions, draw borders	border_set(3X)
complex characters and renditions, draw borders from	box_set(3X)
complex characters and renditions, draw lines from	hline_set(3X)
complex characters and renditions, input an array of, from a window	in_wchnstr(3X)
complex - complex functions and macros	complex(5)
complex conjugate functions	conj(3M)
complex cosine functions	ccos(3M)
complex exponential functions	cexp(3M)
complex functions and macros	complex(5)
complex hyperbolic cosine functions	ccosh(3M)
complex hyperbolic sine functions	csinh(3M)
complex hyperbolic tangent functions	ctanh(3M)
complex imaginary-part functions	cimag(3M)
complex logarithm functions	clog(3M)
complex or partition configuration data; unlock stable complex profile or	

Description	Entry Name(Section)
cancel pending changes to	parunlock(1M)
complex power functions	cpow(3M)
complex profile or cancel pending changes to complex or partition configuration data; unlock stable	parunlock(1M)
complex projection functions	cproj(3M)
complex real-part functions	creal(3M)
complex sine functions	csin(3M)
complex square root functions	csqrt(3M)
complex tangent functions	ctan(3M)
complex: complex absolute value functions	cabs(3M)
complex; display information about a hardware partitionable	parstatus(1)
complex; modify an attribute of a system	cplxmodify(1M)
compliance state of HP-UX 11.x system or depot; check security-bulletin	security_patch_check(1M)
composite graphic symbol	glossary(9)
compound () - compound interest factor	compound(3M)
compound interest factor	compound(3M)
compoundf () - compound interest factor (float)	compound(3M)
compoundl () - compound interest factor (long double)	compound(3M)
compoundq () - compound interest factor (quad)	compound(3M)
compoundw () - compound interest factor (extended)	compound(3M)
compress, uncompress, zcat - compress or expand data	compress(1)
compress and expand files using Huffman code	pack(1)
compress or expand data	compress(1)
compressdir, uncompressdir - compress or expand files in a directory	compress(1)
compute hash value for ELF files	elf_hash(3E)
compute shortest path and route between hosts	pathalias(1)
computer system information, display	uname(1)
computer system, set node name	uname(1)
computer system; get information about	uname(2)
computer system; set node name (system name)	uname(2)
concatenate two strings	string(3C)
concatenate, copy, and print files	cat(1)
concurrency level of unbound threads, get and set	pthread_getconcurrency(3T)
condition variable attributes object, initialize or destroy	pthread_condattr_init(3T)
condition variable, unblock one or all threads waiting on a conditional variable	pthread_cond_signal(3T)
condition variable; initialize or destroy	pthread_cond_init(3T)
condition variable; wait or timed wait on a	pthread_cond_wait(3T)
condition, evaluate for true/false	test(1)
conditions on multiple file descriptors, monitor I/O	poll(7)
conditions on multiple file descriptors; monitor I/O	poll(2)
configurable path name variables, get	pathconf(2)
configurable system variables; get	sysconf(2)
Configuration and Network Services Configuration tools of HP System Management Homepage (HP SMH); launch the Network Interfaces	ncweb(1M)
configuration and status, display LAN device	lanscan(1M)
configuration backup file, create or update LVM volume group	vgcfgbackup(1M)
configuration command; NFS environment	setoncenv(1M)
configuration commands; introduction to kernel	kconfig(5)
configuration data; unlock stable complex profile or cancel pending changes to complex or partition	parunlock(1M)
configuration database, network	netconfig(4)
configuration drift analyzer; system	bastille_drift(1M)
configuration file for inetd	inetd.conf(4)
configuration file for Internet domain name server	named.conf(4)
configuration file for NIS updating	updaters(1M)
configuration file for pluggable authentication module	pam.conf(4)
configuration file for pluggable authentication modules; user	pam_user.conf(4)
configuration file for router advertisement daemon	rtradvd.conf(4)
configuration file for secure internet services	inetsvcs.conf(4)
configuration file for SLP agents	slp.conf(4)
configuration file for tcpd	tcpd.conf(4)

Index

All Volumes

Description	Entry Name(Section)
configuration file for the Access Control Policy Switch (ACPS)	acps.conf(4)
configuration file for the LDAP client daemon process	ldapclntd.conf(4)
configuration file for the name-service switch	nsswitch.conf(4)
configuration file for the SNMP agent	snmpd.conf(4)
configuration file routines, security defaults	secdef(3)
configuration file syntax checking tool; named	named-checkconf(1)
configuration file, change system	ch_rc(1M)
configuration file, evmchmgr	evmchmgr(1M)
configuration file, evmlogger	evmlogger(1M)
configuration file, NLSPATH	nlspath(4)
configuration file, PPPoE (Point to Point Protocol over Ethernet) client	pppoec.conf(4)
configuration file, PPPoE (Point to Point Protocol over Ethernet) relay	pppoerd.conf(4)
configuration file, PPPoE (Point to Point Protocol over Ethernet) server	pppoesd.conf(4)
configuration file, used by DDFSA software	pcf(4)
configuration file; EVM channel	evmchannel.conf(4)
configuration file; EVM channel manager	evmchannel.conf(4)
configuration file; EVM daemon	evmdaemon.conf(4)
configuration file; EVM logger	evmlogger.conf(4)
configuration file; evmchmgr	evmchannel.conf(4)
configuration file; ftpd	ftppaccess(4)
configuration file; Kerberos	krb5.conf(4)
configuration file; network tracing and logging	nettlgen.conf(4)
configuration file; NFS server logging	nfslog.conf(4)
configuration file; resolver	resolver(4)
configuration file; rndc	rndc.conf(4)
configuration file; Route Administration Manager Daemon (RAMD)	ramd.conf(4)
configuration file; security defaults	security(4)
configuration files used by file system administration commands	fs_wrapper(5)
configuration files, FTP	ckconfig(1)
configuration files; reload Event Manager	evmreload(1M)
configuration files; system description	system(4)
Configuration Guide; GateDaemon	gated.conf(4)
configuration information file, diskless client	info(4)
configuration information tool, multicast routing	mrinfo(1M)
configuration of the system; manage the interrupt	intctl(1M)
configuration parameters from the DHCPv6 server, DHCPv6 client interface for requesting	dhcpv6client_ui(1)
configuration pathnames; print kernel	kcpath(1M)
configuration specification file	ftpservers(4)
configuration table; boot device	bootconf(4)
Configuration tool; invokes the HP-UX Security Attributes	secweb(1M)
configuration tool; starts the HP-UX user and group account	ugweb(1M)
Configuration tools of HP System Management Homepage (HP SMH); launch the Network Interfaces Configuration and Network Services	ncweb(1M)
configuration utility for psfontpf; model script	psmsgen(1M)
configuration values, get POSIX	getconf(1)
configuration values; get string-valued	confstr(3C)
configuration, get information for a system's crash dump	pstat(2)
configuration, restore volume group	vgcfgrestore(1M)
configurations; manage kernel	kconfig(1M)
configure message catalog path.	chnlspath(1M)
configure network interface parameters	ifconfig(1M)
configure network tracing and logging subsystem database	nettlconf(1M)
configure system crash dumps	crashconf(1M)
configure system crash dumps	crashconf(2)
configure system language on multi-language systems	geocustoms(1M)
configure system swap space; manage and	swapctl(2)
configure the LP spooling system	lpadmin(1M)
configure the system to use fast symbolic links	create_fastlinks(5)
configure, software products	swinstall(1M)
configures the startup mode of the HPSMH server and of the Tomcat instance used by HPSMH	

Description	Entry Name(Section)
configures, and stops Live Dump, initiates,	smhstartconfig(1M)
confirmation from connect request (X/OPEN TLI-XTI)	livedump(1M)
confstr () - get string-valued configuration values	t_rcvconnect(3)
conj () - complex conjugate function	confstr(3C)
conjf () - complex conjugate function (float)	conj(3M)
conj1 () - complex conjugate function (long double)	conj(3M)
conjq () - complex conjugate function (quad)	conj(3M)
conjw () - complex conjugate function (extended)	conj(3M)
connect accounting records, manipulate	fwtmp(1M)
connect () - initiate a connection on a socket	connect(2)
connect request issued by a transport user (X/OPEN TLI-XTI)	t_accept(3)
connect request (X/OPEN TLI-XTI)	t_listen(3)
connected peer; get address of	getpeername(2)
connected sockets; create a pair of	socketpair(2)
connection daemon debug utility used by DDFA software, outbound	ocdebug(1M)
connection daemon used by DDFA software, outbound	ocd(1M)
connection mapper, multicast router	map-mbone(1M)
connection on a socket; accept a	accept(2)
connection on a socket; initiate a	connect(2)
connection to the EVM (Event Management) daemon	EvmConnection(5)
connection with another transport user (X/OPEN TLI-XTI)	t_connect(3)
connection with the EVM daemon; establish or destroy	EvmConnCreate(3)
connection with the EVM daemon; maintain	EvmConnCheck(3)
connection-specific data pointer for the sendmail connection; gets	smfi_getpriv(3N)
connection; control information for an EVM	EvmConnControl(3)
connection; receive data (X/OPEN TLI-XTI)	t_rcv(3)
connection; send data (X/OPEN TLI-XTI)	t_snd(3)
connections on a socket; listen for	listen(2)
connectivity, verify LAN with link-level loopback	linkloop(1M)
consistency checker, file system quota	quotacheck(1M)
consistency of Authentication database; check internal	authck(1M)
console and standard error, displays formatted message on	fmsg(3C)
console interface; system	console(7)
console - system console interface	console(7)
console, search for during boot process	pdc(1M)
constants; implementation-specific	limits(5)
constants; language information	langinfo(5)
constants; math functions and	math(5)
construct a file system (generic)	mkfs(1M)
construct a new file system	newfs(1M)
construct a new HFS file system	newfs_hfs(1M)
construct an HFS file system	mkfs_hfs(1M)
construct argument lists and execute command	xargs(1)
constructs, nroff/troff, tbl, and neqn, remove	deroff(1)
contention scope of threads, list of external options to specify the scheduling	pthread_scope_options(5)
contents of a file through a socket; send the	sendfile(2)
contents of a Large File through a socket; send the	sendfile64(2)
contents of directories; list	ls(1)
contents of two directories, compare	dircmp(1)
context access (ucontext_t); user	uc_access(3)
context code from current context, return ABI and	uwx_get_abi_context_code(3X)
context initiator and context acceptor, establish security context	gss_init_sec_context(3)
context using the RPCSEC_GSS protocol, create a security	rpc_gss_seccreate(3N)
context, specify callback for	rpc_gss_set_callback(3N)
context-sensitive softkey shell	keysh(1)
context; DEPRECATED; get and set current user	getcontext(2)
contexts; DEPRECATED; manipulate user	makecontext(2)
continue - go to next iteration of enclosing for, select, until, or while loop	sh-posix(1)
continue - resume execution of nearest while or foreach	csh(1)

Index

All Volumes

Description	Entry Name(Section)
continue - resume next iteration of enclosing for/next loop	ksh(1)
continue, resume, or suspend execution of a thread	pthread_resume_np(3T)
control a file descriptor for ELF files	elf_cntl(3E)
control a SCSI device	scsictl(1M)
control access to audio on a workstation; OBSOLETE	asecure(1M)
control address resolution	arp(1M)
control blocking on input	notimeout(3X)
control character	glossary(9)
control character device special file	ioctl(2)
control characters, how to type	ascii(5)
control characters; interpret ASA carriage	asa(1)
control checking for typeahead	typeahead(3X)
control commands; generic device	ioctl(5)
control database file for trusted systems; terminal	ttys(4)
control device driver, SCSI device	sioc_io(7)
control facilities, 4.2 BSD-compatible process	killpg(2)
control facility for internet services, access	tcpd(1M)
control function, for window refresh	touchwin(3X)
control functions for window attribute	attr_get(3X)
control functions, input mode	cbreak(3X)
control functions, restricted window attribute	attroff(3X)
control functions, terminal output	clearok(3X)
control functions, tty line	tccontrol(3C)
control functions, window refresh	is_linetouched(3X)
control information for an EVM connection	EvmConnControl(3)
control initialization; process	init(1M)
control input character delay mode	halfdelay(3X)
control list (ACL) information; set access	setacl(2)
control list (ACL) structure, HFS file system only; convert string form to access	strtoacl(3C)
control lists (ACLs); introduction to HFS access	acl(5)
control maximum resource consumption	getrlimit(2)
control operations, message	msgctl(2)
control operations, semaphore	semctl(2)
control operations, shared memory	shmctl(2)
control routines for open files	fcntl(2)
control system log	syslog(3C)
control terminal device (Bell Version 6 compatibility)	stty(2)
control tty device	tcattribute(3C)
control utility; name server	rndc(1)
control, file system	fsctl(2)
control, IPv6 Neighbor Discovery cache display and	ndp(1M)
control, uucp status inquiry and job	uustat(1)
control, version	vc(1)
control; asynchronous serial modem line	modem(7)
control; let authorized users edit files that are under access	privedit(1M)
control; memory management	memctl(3)
control; multiprocessor	mpctl(2)
control; processor set	pset_ctl(2)
controlling process	glossary(9)
controlling terminal	glossary(9)
controlling terminal interface	tty(7)
controlling terminal, generate file name of	ctermid(3S)
controls process level auditing for the current process and its decedents	setaudproc(2)
controls whether setuid and setgid bits on scripts are honored	secure_sid_scripts(5)
conv() - translate characters	conv(3C)
convenience macros, overview of stack unwind library entry points and	unwind(5)
conventions; file name suffix	suffix(5)
conversion between single-byte and wide character	btowc(3C)
conversion function; privilege name to set	priv_str_to_set(3)
conversion functions; string to NaN	nan(3M)
conversion object status; determine	mbsinit(3C)

Description	Entry Name(Section)
conversion routines, network station address string	net_aton(3C)
conversion routines; codeset	iconv(3C)
conversion, formatted input, to a varargs argument	vscanf(3S)
conversion; codeset	iconv(1)
conversion; date and time	strptime(3C)
conversions database	ftpconversions(4)
conversions; multibyte characters and strings	multibyte(3C)
convert 9-digit hash codes to compressed spelling reference list	spell(1)
convert a character string to a wide-character string (restartable)	mbsrtowcs(3C)
convert a character to a wide-character code	mbrtowc(3C)
convert a wide-character code to a character (restartable)	wctomb(3C)
convert a wide-character string to a character string	wcsrtombs(3C)
convert access control list (ACL) structure to string form	acltostr(3C)
convert an audio file	convert(1)
convert an HFS file system to allow long file names	convertfs(1M)
convert ASCII file format between HP-UX and DOS formats	dos2ux(1)
convert between long integer and base-64 ASCII string	a64l(3C)
convert binary file to ASCII for transmission by mailer	uencode(1)
convert character code set to another	iconv(3C)
convert - convert an audio file	convert(1)
convert date and time to string	ctime(3C)
convert date and time to string	strtime(3C)
convert file keyboard/display data order	forder(1)
convert file to stream	fopen(3S)
convert floating-point number to string	ecvt(3C)
convert formatted input from a window	mvscanw(3X)
convert formatted input from a window	vw_scanw(3X)
convert formatted input from a window	vwscanw(3X)
convert formatted wide-character input	fwscanf(3C)
convert login/logoff records to per-session accounting records	acctcon(1M)
convert long double floating-point number to string	ldcv(3C)
convert long integer to string	ltostr(3C)
convert per-session records to total accounting records	acctcon(1M)
convert privilege ID to privilege name	priv_getbynum(3)
convert privilege name to privilege ID	priv_getbyname(3)
convert spelling reference list words to 9-digit hash codes for spell	spell(1)
convert string data order	strord(3C)
convert string form to access control list (ACL) structure, HFS file system only	strtoacl(3C)
convert string to floating-point number	strtod(3C)
convert string to integer	strtoimax(3C)
convert string to long integer	strtol(3C)
convert tabs to spaces, and vice versa	expand(1)
convert text words to 9-digit hash codes for spell	spell(1)
convert underscores to underlining on terminal	ul(1)
convert units of measure	units(1)
convert user format date and time	getdate(3C)
convert values between host and network byte order	byteorder(3N)
convert wide character string to double-precision number	wcstod(3C)
convert wide character string to float representation	wcstof(3C)
convert wide character string to long double representation	wcstold(3C)
convert wide character string to long integer	wcstoimax(3C)
convert wide character string to long integer	wcstol(3C)
convert, reblock, translate, and copy a (tape) file	dd(1)
convert_awk - converts old sendmail.cf files to new format	convert_awk(1M)
converted trusted system, check if	iscomsec(2)
convertfs - convert an HFS file system to allow long file names	convertfs(1M)
converts old sendmail.cf files to new format	convert_awk(1M)
coordinate ELF library and application versions	elf_version(3E)
coordinate, window, transformation, define	mvdwin(3X)
Coordinated Universal Time	timezone(5)
Coordinated Universal Time (UTC)	glossary(9)

Index

All Volumes

Description	Entry Name(Section)
coordinates, get additional cursor and window coordinates	getbegyx(3X)
coordinates, get cursor and window coordinates	getyx(3X)
copies files and directory hierarchies; extracts, writes, and lists archive files;	pax(1)
copy a file into memory	copylist(3G)
copy a file system with label checking	volcopy(1M)
copy a region of window	copywin(3X)
copy access control list (ACL) to another file	cpacl(3C)
copy file archives in and out; duplicate directory trees	cpio(1)
copy file to a new or existing file	cp(1)
copy file, public UNIX system to UNIX system	uuto(1)
copy files between systems	ftp(1)
copy files to or from remote system	rcp(1)
copy HFS file system with compaction	dcopy(1M)
copy HFS file system with label checking	volcopy_hfs(1M)
copy memory to another area	memory(3C)
copy multiple files to a directory	cp(1)
copy overlapped windows	overlay(3X)
copy standard output to file; pipe fitting to	tee(1)
copy to or from EFI file	efi_cp(1M)
copy to or from LIF files	lifep(1)
copy unwritten system buffers to disk	sync(1M)
copy unwritten system buffers to disk periodically	syncer(1M)
copy, add, modify, delete, or summarize file access control lists (ACLs)	chacl(1)
copy, concatenate, and print files	cat(1)
copy, software products	swinstall(1M)
copy; UNIX system to UNIX system	uucp(1)
copydvagent() - copy device assignment structure for trusted system	getdvagent(3)
copylist() - copy a file into memory	copylist(3G)
copysign() - copysign function	copysign(3M)
copysign functions	copysign(3M)
copysignf() - copysign function (float)	copysign(3M)
copysignl() - copysign function (long double)	copysign(3M)
copysignq() - copysign function (quad)	copysign(3M)
copysignw() - copysign function (extended)	copysign(3M)
copywin() - copy a region of window	copywin(3X)
core dump, determines the inclusion of read/write shared memory in process	core_addshmem_write(5)
core dump, determines the inclusion of readable shared memory in a process	core_addshmem_read(5)
core file settings, change	coreadm(1M)
core file settings, change	coreadm(2)
core file; print a stack trace for each LWP in each process and	pstack(1)
core - format of core image file	core(4)
core image file; format	core(4)
core images of running processes; get	gcore(1)
core_addshmem_read - determines the inclusion of readable shared memory in a process core dump	core_addshmem_read(5)
core_addshmem_write - determines the inclusion of read/write shared memory in process core dump	core_addshmem_write(5)
coreadm() - change priority of a process	coreadm(1M)
coreadm() - change priority of a process	coreadm(2)
correct the time to synchronize the system clock	adjtime(2)
cos() - cosine function	cos(3M)
cosd() - cosine function of a degree argument	cosd(3M)
cosdf() - cosine function of a degree argument (float)	cosd(3M)
cosdl() - cosine function of a degree argument (long double)	cosd(3M)
cosdq() - cosine function of a degree argument (quad)	cosd(3M)
cosdw() - cosine function of a degree argument (extended)	cosd(3M)
cosf() - cosine function (float)	cos(3M)
cosh() - hyperbolic cosine function	cosh(3M)
coshf() - hyperbolic cosine function (float)	cosh(3M)
coshl() - hyperbolic cosine function (long double)	cosh(3M)
coshq() - hyperbolic cosine function (quad)	cosh(3M)

Description	Entry Name(Section)
coshw() - hyperbolic cosine function (extended)	cosh(3M)
cosine and sine of degree argument	sincosd(3M)
cosine and sine together	sincos(3M)
cosine functions	cos(3M)
cosine functions of a degree argument	cosd(3M)
cosine plus i times sine	cis(3M)
cosl() - cosine function (long double)	cos(3M)
cosq() - cosine function (quad)	cos(3M)
cosw() - cosine function (extended)	cos(3M)
cot() - cotangent function	cot(3M)
cotangent functions	cot(3M)
cotangent functions of a degree argument	cotd(3M)
cotd() - cotangent function of a degree argument	cotd(3M)
cotdf() - cotangent function of a degree argument (float)	cotd(3M)
cotdl() - cotangent function of a degree argument (long double)	cotd(3M)
cotdq() - cotangent function of a degree argument (quad)	cotd(3M)
cotdw() - cotangent function of a degree argument (extended)	cotd(3M)
cotf() - cotangent function (float)	cot(3M)
cotl() - cotangent function (long double)	cot(3M)
cotq() - cotangent function (quad)	cot(3M)
cotw() - cotangent function (extended)	cot(3M)
count adjacent repeated lines in a file	uniq(1)
count words, lines, and bytes or characters in a file	wc(1)
cp - copy file, files, or directory subtree	cp(1)
cpacl() - copy access control list (ACL) to another file	cpacl(3C)
cpio archive; format of	cpio(4)
cpio - copy file archives in and out; duplicate directory trees	cpio(1)
cpio - format of cpio archive	cpio(4)
cplxmodify - modify an attribute of a system complex	cplxmodify(1M)
cpow() - complex power function	cpow(3M)
cpowf() - complex power function (float)	cpow(3M)
cpowl() - complex power function (long double)	cpow(3M)
cpowq() - complex power function (quad)	cpow(3M)
cpoww() - complex power function (extended)	cpow(3M)
cpp - the C language preprocessor	cpp(1)
cproj() - complex projection function	cproj(3M)
cprojf() - complex projection function (float)	cproj(3M)
cprojl() - complex projection function (long double)	cproj(3M)
cprojq() - complex projection function (quad)	cproj(3M)
cprojw() - complex projection function (extended)	cproj(3M)
cpset - install object files in binary directories	cpset(1M)
CPU time used; report	clock(3C)
CPU_IS_PA_RISC() - get configurable system variables	sysconf(2)
cr_close() - close a crash dump descriptor	cr_close(3)
cr_info() - retrieve crash dump information	cr_info(3)
cr_isaddr() - validate whether physical page number was dumped	cr_isaddr(3)
cr_open() - open crash dump for reading	cr_open(3)
cr_perror() - print a libcrash error or warning message	cr_perror(3)
cr_read() - read from crash dump	cr_read(3)
cr_set_node() - set crash dump node number	cr_set_node(3)
cr_uncompress() - uncompress a file in a crash dump	cr_uncompress(3)
cr_verify() - verify integrity of crash dump	cr_verify(3)
crash	glossary(9)
crash dump access library	libcrash(5)
crash dump analyzer; invoke KWDB, the source level kernel debugger and	kwdb(1M)
crash dump configuration, get information for a system's	pstat(2)
crash dump data, manipulate	crashutil(1M)
crash dump descriptor, close	cr_close(3)
crash dump device, get information for a	pstat(2)
crash dump information, retrieve	cr_info(3)
crash dump of the operating system, save	savecrash(1M)

Index

All Volumes

Description	Entry Name(Section)
crash dump, open for reading	cr_open(3)
crash dump, read from	cr_read(3)
crash dump, uncompress a file in	cr_uncompress(3)
crash dump, verify integrity of	cr_verify(3)
crash dumps; configure system	crashconf(1M)
crash dumps; configure system	crashconf(2)
crash, patch up damaged file system (generic)	fsdb(1M)
crash, patch up damaged HFS file system	fsdb_hfs(1M)
crashconf () - configure system crash dumps	crashconf(2)
crashconf - configure system crash dumps	crashconf(1M)
crashutil - manipulate crash dump data	crashutil(1M)
creal () - complex real-part function	creal(3M)
crealf () - complex real-part function (float)	creal(3M)
creall () - complex real-part function (long double)	creal(3M)
crealq () - complex real-part function (quad)	creal(3M)
crealw () - complex real-part function (extended)	creal(3M)
creat () - create a new file or rewrite an existing one	creat(2)
creat64 () - non-POSIX standard API interfaces to support large files	creat64(2)
create a directory	mkdir(1)
create a directory file	mkdir(2)
create a directory, special, or ordinary file	mknod(2)
create a makefile	mkmf(1)
create a name for a temporary file	tmpnam(3S)
create a new file or rewrite an existing one	creat(2)
create a new key in the publickey database file	newkey(1M)
create a new partition	parcreate(1M)
create a new process	fork(2)
create a new thread of execution	pthread_create(3T)
create a pair of connected sockets	socketpair(2)
create a security context using the RPCSEC_GSS protocol	rpc_gss_seccreate(3N)
create a socket	socket(2)
create a special (device) file	mksf(1M)
create a tags file	ctags(1)
create a temporary file	tmpfile(3S)
create an endpoint for communication	socket(2)
create an interprocess channel	pipe(2)
create and administer SCCS files	admin(1)
create and destroy events	EvmEventCreate(3)
create and initialize a callback info structure for self-unwinding	uwx_self_init_info(3X)
create and initialize an unwind environment	uwx_init(3X)
create and initialize an unwind environment	uwx_init_context(3X)
create and initialize an unwind environment	uwx_set_remote(3X)
create and manage processor sets	psrset(1M)
create and manipulate event items	EvmItemGet(3)
create and monitor jobs	swjob(1M)
create backup LVM volume group configuration file	vgcgbgbackup(1M)
create cat and whatis files for online manpages	catman(1M)
create file names	glob(3C)
create interprocess communication identifier	ftok(3C)
create logical volume in LVM volume group	lvcreate(1M)
create LVM volume group	vgcreate(1M)
create message catalog file for modification	findmsg(1)
create message files for use by gettext()	mkmsgs(1)
create or destroy a thread-specific data key	pthread_key_create(3T)
create or open a message queue	mq_open(2)
create or rebuild Network Information Service databases	ypmake(1M)
create partitions for disks on an Integrity system	idisk(1M)
create physical volume for use in LVM volume group	pvcreate(1M)
create processor set	pset_create(2)
create session	setsid(2)
create session and set process group ID	setsid(2)

Description	Entry Name(Section)
create shutdown message file for ftp servers	ftpshtut(1)
create special and FIFO files	mknod(1M)
create the interface socket that MTAs use to connect to the filter; attempts to	smfi_opensocket(3N)
create zero-length file	cp(1)
create zero-length file	cat(1)
create zero-length file	null(7)
create, distribute, install, monitor, and manage software	sd(5)
create, modify, and delete event subscriptions; enables you to view,	evweb_subscribe(1)
create, remove directories in a path	mkdirp(3G)
create, windows, functions	newwin(3X)
create/open a shared memory object	shm_open(2)
create_fastlinks - configure the system to use fast symbolic links	create_fastlinks(5)
create_sysfile - create a kernel system file	create_sysfile(1M)
creates database maps for sendmail	makemap(1M)
creating PAM sessions	pam_open_session(3)
creation function, relative window	derwin(3X)
credential, acquire handle	gss_accept_sec_context(3)
credential, provide the calling application information about	gss_inquire_cred(3)
credentials of client, get	rpc_gss_getcred(3N)
cron - timed-job execution daemon	cron(1M)
crontab, batch , and at queue description file	queuedefs(4)
crontab file operations; user	crontab(1)
crontab - user crontab file operations	crontab(1)
CRT or line-printer output, format text file for	nroff(1)
crt0 - execution startup routines	crt0(3)
crt0.o - execution startup routines	crt0(3)
crt0.o() - execution startup routines for Integrity systems	crt0_ia(3)
crt0.o - execution startup routines for PA-RISC systems	crt0_pa(3)
crt0_ia - execution startup routines for Integrity systems	crt0_ia(3)
crt0_pa - execution startup routines for PA-RISC systems	crt0_pa(3)
crypt - encode and decode files	crypt(1)
crypt() - generate hashing encryption	crypt(3C)
cryptographic message integrity code (MIC), calculate and return in a token	gss_get_mic(3)
cryptographic message integrity code (MIC), check against a message to verify its integrity ..	gss_verify_mic(3)
CS/80	glossary(9)
CS/80	glossary(9)
csh - a shell (command interpreter) with C-like syntax	csh(1)
csin() - complex sine function	csin(3M)
csinf() - complex sine function (float)	csin(3M)
csinh() - complex hyperbolic sine function	csinh(3M)
csinhf() - complex hyperbolic sine function (float)	csinh(3M)
csinh1() - complex hyperbolic sine function (long double)	csinh(3M)
csinhq() - complex hyperbolic sine function (quad)	csinh(3M)
csinhw() - complex hyperbolic sine function (extended)	csinh(3M)
csinl() - complex sine function (long double)	csin(3M)
csinq() - complex sine function (quad)	csin(3M)
csinw() - complex sine function (extended)	csin(3M)
csplit - context split	csplit(1)
csqrt() - complex square root function	csqrt(3M)
csqrtf() - complex square root function (float)	csqrt(3M)
csqrt1() - complex square root function (long double)	csqrt(3M)
csqrtq() - complex square root function (quad)	csqrt(3M)
csqrtw() - complex square root function (extended)	csqrt(3M)
ct - spawn getty to remote terminal (call terminal)	ct(1)
ctags - create a tags file	ctags(1)
ctan() - complex tangent function	ctan(3M)
ctanf() - complex tangent function (float)	ctan(3M)
ctanh() - complex hyperbolic tangent function	ctanh(3M)
ctanhf() - complex hyperbolic tangent function (float)	ctanh(3M)
ctanh1() - complex hyperbolic tangent function (long double)	ctanh(3M)
ctanhq() - complex hyperbolic tangent function (quad)	ctanh(3M)

Index

All Volumes

Description	Entry Name(Section)
ctanhw() - complex hyperbolic tangent function (extended)	ctanh(3M)
ctanl() - complex tangent function (long double)	ctan(3M)
ctanq() - complex tangent function (quad)	ctan(3M)
ctanw() - complex tangent function (extended)	ctan(3M)
ctermid() - generate file name for terminal	ctermid(3S)
ctime() , ctime_r() - convert clock() date and time to string	ctime(3C)
ctime() ; time zone adjustment table for	tztab(4)
ctype - classify characters according to type	ctype(3C)
cu - call another (UNIX) system, terminal emulator	cu(1)
cube root functions	cbrt(3M)
cumulative value changes per System V IPC semop() call, maximum	semaem(5)
cur_term() - current terminal information	cur_term(3X)
current absolute system time, add a specific time interval to the	get_expiration_time(3T)
current and root directories for auditing subsystem; enable/disable tracking of	audit_track_paths(5)
current directory	glossary(9)
current erase and line kill characters	eraseschar(3X)
current events and system calls to be audited; set	setevent(2)
current host system, set or display name of	hostname(1)
current host system; get name of	gethostname(2)
current host, get an identifier for	gethostid(2)
current locale; query numeric formatting conventions	localeconv(3C)
current page size, get	getpagesize(2)
current process; get the audit ID (aid) for the	getaudit(2)
current process; set the audit ID (aid) for the	setaudit(2)
current process; suspend or resume auditing on the	audswitch(2)
current SCCS file editing activity, print	sact(1)
current state (X/OPEN TLI-XTI)	t_getstate(3)
current status of power for cells and I/O chassis; turn on/off or display	frupower(1M)
current status of the UUCP system	uusnap(1M)
current system users, list	who(1)
current terminal information	cur_term(3X)
current terminal, get verbose description of	longname(3X)
current user context; DEPRECATED; get and set	getcontext(2)
current user ID; print effective	whoami(1)
current user, find the slot in the utmpx() file of the	ttyslot(3C)
current users and processes, list	whodo(1M)
current value of system-wide clock, get	getclock(3C)
current window	curscr(3X)
current working directory	glossary(9)
current working directory, get path-name of	getcwd(3C)
current working directory, get pathname of	getwd(3C)
cur_set() - set the cursor mode	cur_set(3X)
curscr() - current window	curscr(3X)
curses - definitions for screen handling and optimization functions	curses(5)
Curses session, suspend	endwin(3X)
curses window, scroll	scroll(3X)
curses.h - definitions for screen handling and optimization functions	curses(5)
curses_intro - terminal and printer handling and optimization package	curses_intro(3X)
cursor and window coordinates, get	getyx(3X)
cursor and window coordinates, get additional	getbegyx(3X)
cursor to end of window, clear	clrtoeol(3X)
cursor, clear from it to end of line	clrtoeol(3X)
cursor, output movement commands to the terminal	mvcur(3X)
cursor, set the cursor mode	cur_set(3X)
cursor, window location functions	move(3X)
cuserid() - get character-string login name of the user	cuserid(3S)
cut - cut out selected fields of each line in a file	cut(1)
Cyclical Redundancy Check on a file	sum(1)
d_passwd - dialup security control	dialups(4)
daemon	glossary(9)
daemon configuration file; EVM	evmdaemon.conf(4)

Description	Entry Name(Section)
daemon configuration file; RAMD	ramd.conf(4)
daemon debug utility used by DDFA software, outbound connection	ocdebug(1M)
daemon for IPv6, Router Advertisement	rtradvd(1M)
daemon for IPv6; BGP routing	bgpd(1M)
daemon for IPv6; RIPng routing	ripngd(1M)
daemon for modifying Network Information Service passwd database	yppasswdd(1M)
daemon for processing system commands; pass-through	fsdaemon(1M)
daemon that responds to SNMP requests	snmpd(1M)
daemon used by DDFA software, outbound connection	ocd(1M)
daemon, DHCPv6 client	dhcpv6clientd(1M)
daemon, gateway routing	gated(1M)
daemon, kernel registry services daemon	krsd(1M)
daemon, kills the sendmail daemon	killsm(1M)
daemon, line printer daemon for LP requests from remote systems	rlpdaemon(1M)
daemon, password and group hashing and caching	pwgrd(1M)
daemon, PCI I/O hotplug (attention button) events	hotplugd(1M)
daemon, PPPoE (Point-to-Point Protocol over Ethernet) server	pppoesd(1M)
daemon, Uninterruptible Power System (UPS) monitor	ups_mond(1M)
daemon, user accounting database	utmpd(1M)
daemon; configuration file for router advertisement	rtradvd.conf(4)
daemon; connection to the EVM (Event Management)	EvmConnection(5)
Daemon; Essential Services Monitor	esmd(1M)
daemon; establish or destroy connection with the EVM	EvmConnCreate(3)
daemon; Event Manager	evmd(1M)
daemon; Internet services	inetd(1M)
daemon; IP multicast routing	mrouted(1M)
daemon; lightweight resolver	lwresd(1M)
daemon; maintain connection with the EVM	EvmConnCheck(3)
daemon; network lock	lockd(1M)
daemon; Network Time Protocol	xntpd(1M)
daemon; NFS	biод(1M)
daemon; NFS	nfsd(1M)
daemon; nfs logging	nfslogd(1M)
daemon; NFS Version 4 callback	nfs4cbd(1M)
daemon; post events to the EVM	evmpost(1)
daemon; PPP point to point protocol	pppd(1)
Daemon; Service Location Protocol	slpd(1M)
daemon; system physical environment	envd(1M)
daemon; timed-job execution	cron(1M)
daemon; UUCP over TCP/IP server	uucpd(1M)
daily accounting shell procedure	runacct(1M)
damaged file system, patch up (generic)	fsdb(1M)
damaged HFS file system, patch up	fsdb_hfs(1M)
data allocation space of object files, print section sizes and	size(1)
data and stack space, allocate then lock process into memory	datalock(3C)
data base compiler, terminfo	tic(1M)
data base of terminal-type for each tty port	ttytype(4)
data base, terminfo, de-compile	untic(1M)
Data Communications and Terminal Controller Device File Access software	ddfa(7)
data encryption	glossary(9)
data error indication (X/OPEN TLI-XTI)	t_revuderr(3)
data from a file, read	read(2)
data integrity check on an event; perform a	EvmEventValidate(3)
data link provider interface	dipi(7)
data link provider interface standard header file	dipi(4)
data link provider interface, HP specific extensions for	dipi_ext(4)
data or expedited data over a connection (X/OPEN TLI-XTI)	t_snd(3)
data order for display/keyboard, convert file	forder(1)
data order, convert string	strord(3C)
data over connection; receive (X/OPEN TLI-XTI)	t_rev(3)
data pointer for binary search tree, get	tsearch(3C)

Index

All Volumes

Description	Entry Name(Section)
data pointer for the sendmail connection; sets the private	smfi_setpriv(3N)
data returned by the stat() function	stat(5)
data segment and shared memory, attach or detach	shmop(2)
data segment for any user process; maximum size (in bytes) of the	maxdsiz(5)
data segment space allocation, change	brk(2)
data structure; allocate and deallocate unwind library	_UNW_createContextForSelf(3X)
data structure; manipulate values in unwind library	_UNW_currentContext(3X)
data structure; query values in unwind library	_UNW_getGR(3X)
data to a file, write	write(2)
data to check the network, scatter	spray(3N)
data to disk; flush kernel registry services	krs_flush(1M)
data translation of ELF files	elf_xlate(3E)
data types, basic integer	inttypes(5)
data types, system primitives	types(5)
data types; fixed-size integer	inttypes(5)
data unit from remote transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)
data unit; send to transport user (X/OPEN TLI-XTI)	t_sndudata(3)
data, expand or compress	compress(1)
data, lock in memory	plock(2)
data: manipulate crash dump data	crashutil(1M)
data; display system and user login	logins(1M)
data; display user login	listusers(1)
data; maximum or minimum amount of physical memory used for caching file I/O	filecache_max(5)
data; unlock stable complex profile or cancel pending changes to complex or partition configuration	parunlock(1M)
database and directory structure; Network Information Service	ypfiles(4)
database converter; DHCP client	dhcplib2conf(1M)
database entries (for trusted systems only); manipulate protected password	getprpwent(3)
database entry for a trusted system; manipulate device assignment	getdvagent(3)
database entry for a trusted system; manipulate system default	getprdfent(3)
database entry, manipulate terminal control	getprtcent(3)
database file for a trusted system; device assignment	devassign(4)
database file for a trusted system; system default	default(4)
database file for trusted systems; terminal control	ttys(4)
database files, verify the syntax of the Role-Based Access Control (RBAC)	rbacdbchk(1M)
database for per-user information; user	userdb(4)
database for public keys	publickey(4)
database for the mail aliases file, rebuild	newaliases(1M)
database for trusted systems; protected password authentication	prpwd(4)
database from server to local node; transfer NIS	ypxfr(1M)
database maintained by utmpd ; access and update routines for user-accounting	getuts(3C)
database maps for sendmail, creating	makemap(1M)
database subroutines (new multiple database version)	ndbm(3X)
database subroutines (old version - see also ndbm(3X))	dbm(3X)
database, /var/adm/userdb ; display information residing in the user	userdbget(1M)
database, /var/adm/userdb ; modify information in the user	userdbset(1M)
database, /var/adm/userdb , read, write or delete information in the user	userdb_read(3)
database, /var/adm/userdb , verify or fix information in the user	userdbck(1M)
database, conversions ftpd	ftpconversions(4)
database, host names	hosts(4)
database, network configuration	netconfig(4)
database, network name	networks(4)
database, pathalias, access and manage the	uupath(1)
database, protocol name	protocols(4)
database, relational, join two relations in	join(1)
database, RPC program number	rpc(4)
database, service name	services(4)
database, user-accounting	utmps(4)
database, write records into new wtmps and btmps	bwtmps(3C)
database; check internal consistency of Authentication	authck(1M)
database; daemon for modifying Network Information Service passwd	yppasswdd(1M)

Description	Entry Name(Section)
database; display protected password	getprpw(1M)
database; force propagation of Network Information Service	yppush(1M)
database; kernel packet forwarding	route(7P)
database; make a Network Information System	makedbm(1M)
database; modify protected password	modprpw(1M)
database; noninteractive editing of a command's authorization and privilege information in the privrun	cmdprivadm(1M)
databases for trusted systems; security	authcap(4)
databases, build and install Network Information Service	ypinit(1M)
databases, non-interactive editing of the authorization information in the RBAC	authadm(1M)
databases; create or rebuild Network Information Service	ypmake(1M)
databases; noninteractive editing of role-related information in RBAC	roleadm(1M)
datacomm line speed and terminal settings used by getty	gettydefs(4)
datagram protocol, Internet user	UDP(7P)
datalock() - lock process into memory after allocating data and stack space	datalock(3C)
date ; time zone adjustment table for	tztabs(4)
date and time conversion	strptime(3C)
date and time to string; convert	ctime(3C)
date and time to wide-character string; convert	wcsftime(3C)
date and time, get more precisely (Version 7 compatibility only)	ftime(2)
date and time, get the	gettimeofday(2)
date and time, set the	settimeofday(2)
date and time, set via NTP	ntpdate(1M)
date and time; convert user format	getdate(3C)
date and time; display or set	date(1)
date - display or set the date and time	date(1)
date, set	stime(2)
date/time; set system initial identity parameter	set_parms(1M)
date; convert to string	strptime(3C)
daylight() - Daylight Savings Time flag	ctime(3C)
dbc_max_pct - OBSOLETE kernel tunable parameter	dbc_max_pct(5)
dbc_min_pct - OBSOLETE kernel tunable parameter	dbc_max_pct(5)
dbm_clearerr() - reset error condition on named database	ndbm(3X)
dbm_close() - close an open database	ndbm(3X)
dbm_delete() - delete a database key and associated contents	ndbm(3X)
dbm_error() - error in reading or writing in a database	ndbm(3X)
dbm_fetch() - access a database entry under a key	ndbm(3X)
dbm_firstkey() - get first key in a database	ndbm(3X)
dbm_nextkey() - get next key in a database	ndbm(3X)
dbm_open() - open a database for access	ndbm(3X)
dbm_store() - store an entry under a key in a database	ndbm(3X)
dbmclose() - close currently open database (old single-data-base version)	dbm(3X)
dbminit() - open a single database (old single-data-base version)	dbm(3X)
dc - desk calculator	dc(1)
dcopy - copy HFS file system with compaction	dcopy(1M)
dd - convert, reblock, translate, and copy a (tape) file	dd(1)
ddfa - Data Communications and Terminal Controller Device File Access software	ddfa(7)
DDFA software and Telnet port identification feature; dedicated ports file used by	dp(4)
DDFA software, configuration file, used by	pcf(4)
DDFA software, dedicated ports parser used by	dpp(1M)
DDFA software, outbound connection daemon used by	ocd(1M)
DDS tape; initialize disk or partition	mediainit(1)
de-compile terminfo data base	untic(1M)
dead.letter file	sendmail(1M)
deallocate unwind library data structure; allocate and	_UNW_createContextForSelf(3X)
debug file system (generic)	fsdb(1M)
debug HFS file system	fsdb_hfs(1M)
debug utility used by DDFA software, outbound connection daemon	ocdebug(1M)
debugger and crash dump analyzer; invoke KWDB, the source level kernel	kwdb(1M)
debugger (generic); file system	fsdb(1M)
debugger: absolute debugger	adb(1)

Index

All Volumes

Description	Entry Name(Section)
debugger: assembler debugger	adb(1)
debugger: object code debugger	adb(1)
debugging level for the Milter library for sendmail; sets the	smfi_setdbg(3N)
debugging purposes., a feature that saves operating system state to the file system for	livedump(5)
decimal ASCII string, convert long integer to	ltostr(3C)
decimal equivalents: ASCII character set	ascii(5)
decode a file encoded by uuencode	uuencode(1)
decode files; encode and	crypt(1)
decompose floating-point number	modf(3M)
decrease physical extents allocated to LVM logical volume	lvreduce(1M)
decrypt and store secret key	keylogin(1)
decrypt files	crypt(1)
decrypt message content	gss_unwrap(3)
dedicated line, reserve for a purpose	ripoffline(3X)
dedicated ports file used by DDFA software and Telnet port identification feature	dp(4)
dedicated ports parser used by DDFA software	dpp(1M)
def_prog_mode() - save or restore program or shell terminal modes	def_prog_mode(3X)
def_shell_mode() - save terminal modes as the "shell" state	def_prog_mode(3X)
default database entry for a trusted system; manipulate system	getprdfent(3)
default database file for a trusted system; system	default(4)
default - label default in switch statement	csh(1)
default login shell; change	chsh(1)
default message catalog, set	setcat(3)
default search path	glossary(9)
default SMTP error reply code to a multi-line response; sets	smfi_setmreply(3N)
default stacksize., change the	pthread_default_rsestacksize_np(3T)
default stacksize; change	pthread_default_stacksize_np(3T)
default - system default database file for a trusted system	default(4)
default window	stdscr(3X)
default, controls whether program stacks are executable by	executable_stack(5)
default_disk_ir - enable and disable use of device's write cache in the SCSI subsystem (OBSOLETE)	default_disk_ir(5)
defaults configuration file; security	security(4)
define additional severities	addsev(3C)
define additional signal stack space	sigspace(2)
define and describe audit system events	audeventstab(4)
define character mapping	towctrans(3C)
define kernel process accounting output file or disable accounting	acct(1M)
define label for formatting routines	setlabel(3)
define the minimum priority for printing	lpsched(1M)
define what to do upon receipt of a signal	signal(2)
define window coordinate transformation	mvdwderwin(3X)
defines and manages file system stack templates	fstadm(2)
defines the maximum accounting file size	max_acct_file_size(5)
definition, display system	sysdef(1M)
definitions for screen handling and optimization functions	curses(5)
definitions, memory mapping	mman(5)
definitions, regular expression and pattern matching notation	regexp(5)
defunct process	glossary(9)
degree-valued arccosine functions	acosd(3M)
degree-valued arcsine functions	asind(3M)
degree-valued arctangent functions	atand(3M)
degree-valued arctangent-and-quadrant functions	atan2d(3M)
del_curterm() - interface to terminfo database	del_curterm(3X)
delay and insert capability, for terminal	has_ic(3X)
delay command execution	at(1)
delay mode, control input character delay mode	halfdelay(3X)
delay_output() - delay output	delay_output(3X)
delch() , mvdelch() , mvwdelch() , wdelch() - delete character from a window	delch(3X)
delete a directory	rmdir(1)
delete a file or directory	rm(1)

Description	Entry Name(Section)
delete a group from the system	groupdel(1M)
delete a node from a binary search tree	tsearch(3C)
delete a user login from the system	userdel(1M)
delete a window	delwin(3X)
delete allocated signal stack space	sigspace(2)
delete and modify user credentials for an authentication service	pam_setcred(3)
delete() - delete key and data under it (old single-data-base version)	dbm(3X)
delete entries in an LDAP directory; simple add, modify, and	ldapentry(1)
delete event subscriptions; enables you to view, create, modify, and	evweb_subscribe(1)
delete events; enables you to view and	evweb_eventviewer(1)
delete file	unlink(2)
delete information in the user database, /var/adm/userdb, read, write or	userdb_read(3)
delete or insert lines into a window	insdelln(3X)
delete secret key stored with keyserv	keylogout(1)
delete selected characters	tr(1)
delete, add, or modify delete access control list entry	setaclentry(3C)
delete, copy, add, modify, or summarize file access control lists (ACLs)	chacl(1)
delete-character features, hardware, enable or disable use of	idcok(3X)
deleteln() , wdeleteln() - delete lines in window	deleteln(3X)
deletes a message header; changes or	smfi_chgheader()(3N)
deletion, mark a credential	gss_accept_sec_context(3)
delimiter; read stream up to next	bgets(3G)
delmntent() - delete an entry from open file system descriptor file	getmntent(3X)
delscreen() - free storage associated with a screen	delscreen(3X)
delta	glossary(9)
delta (change) to an SCCS file, make a	delta(1)
delta commentary of an SCCS delta, change	cdc(1)
delta from an SCCS file; remove a	rmdel(1)
delta - make a delta (change) to an SCCS file	delta(1)
deltas; combine SCCS	comb(1)
delwin() - delete a window	delwin(3X)
demon	glossary(9)
deny or permit <i>write</i> (1) messages from other users to terminal	msg(1)
dependencies of executable files or shared libraries on Integrity systems; list dynamic	ldd_ia(1)
dependencies of executable files or shared libraries on PA-RISC systems; list dynamic	ldd_pa(1)
dependencies of executable files or shared libraries; list dynamic	ldd(1)
depot, modify software products target in root or	swmodify(1M)
depot; check security-bulletin compliance state of HP-UX 11.x system or	security_patch_check(1M)
depots and roots, register or unregister	swreg(1M)
DEPRECATED; get and set current user context;	getcontext(2)
DEPRECATED; manipulate user contexts;	makecontext(2)
deroff - remove nroff, tbl, and neqn constructs	deroff(1)
derwin() - relative window creation function	derwin(3X)
DES encryption key, generate a	makekey(1)
descend a directory hierarchy recursively, executing a function	ftw(3C)
describe an audio file	attributes(1)
describe audit system events, define and	audeventstab(4)
describe characteristics of a disk device	diskinfo(1M)
description file for at , batch , and crontab queues	queuedefs(4)
description of common HP-UX terms	glossary(9)
description of disk by its name, get	getdiskbyname(3C)
description of named defines and other specifications for namespace from HP-UX header files	stdsyms(5)
description of RCS commands	rcsintro(5)
description of signals	signal(5)
description of supported languages	lang(5)
description, Extensible Firmware Interface	efi(4)
description, logical interchange format	lif(4)
description, verbose, of current terminal	longname(3X)
descriptions; compare or print out terminfo	infocmp(1M)
descriptor file entry, get file system (BSD 4.2 compatibility only)	getfsent(3X)
descriptor file entry; get file system	getmntent(3X)

Index

All Volumes

Description	Entry Name(Section)
descriptor for ELF file; make file	elf_begin(3E)
descriptor table; get the size of the per-process file	getdtablesize(2)
descriptor to a specific slot; duplicate an open file	dup2(2)
descriptor, close a file	close(2)
descriptor, map stream pointer to file	fileno(3S)
descriptor: update an ELF descriptor	elf_update(3E)
descriptors, monitor I/O conditions on multiple file	poll(7)
descriptors; displays process address information and open file	pmap(1)
descriptors; monitor I/O conditions on multiple file	poll(2)
desk calculator	dc(1)
destinations; move requests between LP	lpsched(1M)
destroy a mutex	pthread_mutex_init(3T)
destroy a mutex attribute object	pthread_mutexattr_init(3T)
destroy a read-write lock	pthread_rwlock_init(3T)
destroy a read-write lock attribute object	pthread_rwlockattr_init(3T)
destroy a thread attribute object	pthread_attr_dinit(3T)
destroy a thread-specific data key	pthread_key_create(3T)
destroy an unnamed semaphore	sem_destroy(2)
destroy connection with the EVM daemon; establish or	EvmConnCreate(3)
destroy events; create and	EvmEventCreate(3)
destroy Kerberos tickets	kdestroy(1)
destroy or initialize a condition variable	pthread_cond_init(3T)
destroy or initialize a thread condition variable attributes object	pthread_condattr_init(3T)
destroy processor set	pset_destroy(2)
detach a name from a STREAMS-based file descriptor	fdetach(3C)
detach a STREAMS-based file descriptor	fdetach(1M)
detach a thread to reclaim its resources when it terminates	pthread_detach(3T)
detach shared memory from data segment	shmop(2)
detailed information about kernel tunable parameters; retrieve	tuneinfo2(2)
determine accessibility of a file	access(2)
determine conversion object status	mbsinit(3C)
determine current signal stack space	sigspace(2)
determine file type	file(1)
determine file type for ELF files	elf_kind(3E)
determine time interval (in secs) for flushing audit records	diskaudit_flush_interval(5)
determine whether a screen has been refreshed	isendwin(3X)
determine whether a socket is at the out-of-band mark	socketatmark(3N)
determine whether two internal names are equivalent, allow an application to compare and	gss_compare_name(3)
determine who is logged in on local network machines	rusers(1)
determined locality domain IDs	pthread_processor_bind_np(3T)
determined processor IDs	pthread_processor_bind_np(3T)
determines whether to reserve a tape device on open	st_ats_enabled(5)
devassign - trusted system device assignment database file	devassign(4)
device	glossary(9)
device address	glossary(9)
device and FIFO files, create	mknod(1M)
device assignment database entry for a trusted system; manipulate	getdvagent(3)
device assignment database file for a trusted system	devassign(4)
device configuration and status, display LAN0	lanscan(1M)
device control commands; generic	ioctl(5)
device description file format; PPP physical	ppp.Devices(4)
device driver, PS/2 keyboard and mouse	ps2(7)
device driver, SCSI direct access	scsi_disk(7)
device driver, SCSI media changer	autochanger(7)
device driver: STREAMS pass through driver	clone(7)
device driver; HP-HIL	hil(7)
device drivers	see <i>special files</i>
device drivers in the system, list	lsdev(1M)
device drivers to interact with DLPI	dlpi_drv(4)
device drivers; Small Computer System Interface	scsi(7)

Description	Entry Name(Section)
device file	glossary(9)
Device File Access software, Data Communications and Terminal Controller	ddfa(7)
device file, block mode terminal	blmode(7)
device file, write an EFI file system header on a	efi_fsinit(1M)
device files, network file system	nfs(7)
device files; special	see <i>special files</i>
device ID to file path; map	devnm(3)
device name	devnm(1M)
device numbers, header file of macros for handling	mknod(5)
device on open; determines whether to reserve a tape	st_ats_enabled(5)
device or file system for paging; enable	swapon(1M)
device special file, control character	ioctl(2)
device (special) file; make	mksf(1M)
device (special) file; remove	rmsf(1M)
device special files; introduction to	intro(7)
device's write cache in the SCSI subsystem (OBSOLETE); enable and disable use of	default_disk_ir(5)
device, control terminal (Bell Version 6 compatibility)	stty(2)
device, describe characteristics of a disk	diskinfo(1M)
device, get information for a crash dump	pstat(2)
device, who is currently using	fuser(1M)
device: STREAMS device	isastream(3C)
device; bind a driver to a	iobind(1M)
devices that can be enabled for swap; maximum	nswapdev(5)
devnm - device name	devnm(1M)
devnm() - map device ID to file path	devnm(3)
df - report number of free CDFS, HFS, or NFS file system disk blocks	df_hfs(1M)
df - report number of free file system disk blocks	df(1M)
dfstab - file containing commands for sharing resources across a network	dfstab(4)
DHCP client database converter	dhcpdb2conf(1M)
DHCP elements of bootpd, command line tool	dhcptools(1M)
DHCP support command; initial system configuration plus	auto_parms(1M)
dhcpcclient - Client for Dynamic Host Configuration Protocol Server	dhcpcclient(1M)
dhcpdb2conf - convert DHCP client database to config file parameters	dhcpdb2conf(1M)
dhcptools - command line tool for DHCP elements of bootpd	dhcptools(1M)
DHCPv6 client daemon	dhcpv6clientd(1M)
DHCPv6 client interface for requesting configuration parameters from the DHCPv6 server	dhcpv6client_ui(1)
DHCPv6 server, DHCPv6 client interface for requesting configuration parameters from the	dhcpv6client_ui(1)
dhcpv6client_ui - DHCPv6 client interface for requesting configuration parameters from the DHCPv6 server	dhcpv6client_ui(1)
dhcpv6d - Dynamic Host Configuration Protocol Server daemon for IPv6	dhcpv6d(1M)
dhcpv6db2conf - DHCPv6 client database converter	dhcpv6db2conf(1M)
dhcpv6db2conf DHCPv6 client database converter	dhcpv6db2conf(1M)
diag0 - diagnostic interface to I/O subsystem	diag0(7)
diag1 - diagnostic interface to I/O subsystem	diag1(7)
diag2 - diagnostic interface	diag2(7)
diagnostic information, dynamic linking process	dlerrno(3C)
diagnostic information, dynamic linking process	dlerror(3C)
diagnostic interface	diag2(7)
diagnostic interface to I/O subsystem	diag0(7)
diagnostic interface to I/O subsystem	diag1(7)
diagnostic messages, collect to form system error log	dmesg(1M)
diagnostic utility; SCSI management and	scsimgr(1M)
diagnostics	femsutil(1M)
dial() - establish an outgoing terminal line connection	dial(3C)
dialer description file format; PPP	ppp.Dialers(4)
dialups, d_passwd - dialup security control	dialups(4)
dialups file	login(1)
diff, diffh - differential file comparator	diff(1)
diff between sorted files; reject/select common lines	comm(1)

Index

All Volumes

Description	Entry Name(Section)
diff file to an original file; program to apply a	patch(1)
diff, big	bdiff(1)
diff3 - three-way differential file comparison	diff3(1)
differences among three files	diff3(1)
differences between files, show side-by-side	sdiff(1)
differences between large files, find	bdiff(1)
differences between two files	diff(1)
differences between two files, mark	diffmk(1)
differential file and directory comparator	diff(1)
diffmk - mark differences between files	diffmk(1)
difftime() - difference between two calendar time values	ctime(3C)
dig - domain information groper	dig(1M)
dir - format of directories	dir(4)
dircmp - directory comparison	dircmp(1)
direct access device driver, SCSI	scsi_disk(7)
direct disk device access drivers	disk(7)
directories in a path; create, remove	mkdirp(3G)
directories to export to NFS clients	exports(4)
directories; list contents of	ls(1)
directories; remove	rmdir(1)
directories; remove files or	rm(1)
directories; search for named file in named	pathfind(3G)
directory	glossary(9)
directory and file structures, statd	sm(4)
directory clean-up, uucp spool	uucleanup(1M)
directory comparator; differential file and	diff(1)
directory - directory operations	directory(3C)
directory file, remove a	rmdir(2)
directory file; make	mkdir(2)
directory file; make	mknod(2)
directory format	dir(4)
directory hierarchies; extracts, writes, and lists archive files; copies files and	pax(1)
directory hierarchy, recursively descend a, executing a function	ftw(3C)
Directory Name Lookup Cache (DNLC); number of locks for the	dnlc_hash_locks(5)
Directory Name Lookup Cache (DNLC) entries, number of	ncsize(5)
directory name; working	pwd(1)
directory names; list of home	usermod(4)
directory operations	directory(3C)
directory pointer array, sort a	scandir(3C)
directory service, Internet user name	whois(1)
directory streams and directory entries; format	dirent(5)
directory structure; Network Information Service database and	ypfiles(4)
directory subtrees; copy files and	cp(1)
directory trees; copy file archives in and out; duplicate	cpio(1)
directory trees; duplicate	cpio(1)
directory, change working	cd(1)
directory, get entries in a file-system-independent format	getdirentries(2)
directory, get pathname of current working	getwd(3C)
directory, list EFI file information or contents of an EFI	efi_ls(1M)
directory, make an EFI	efi_mkdir(1M)
directory, move directory subtree and files to another directory	mv(1)
directory, move multiple files to another directory	mv(1)
directory, remove an EFI	efi_rmdir(1M)
directory, rename directory	mv(1)
directory, scan a	scandir(3C)
directory, symbolic links between directories, create	ln(1)
directory: change root directory	chroot(2)
directory: change root directory for a command	chroot(1M)
directory: change working directory	chdir(2)
directory: compare contents of two directories	dircmp(1)
directory: copy directory subtree and files to another directory	cp(1)

Description	Entry Name(Section)
directory: copy multiple files to a directory	cp(1)
directory: expand compressed files in a directory	compress(1)
directory: get path-name of current working directory	getcwd(3C)
directory: move a directory (requires super-user)	mvdir(1M)
directory: read portable archive	pax(1)
directory: remove entry	unlink(2)
directory: scan a directory	scandir(3C)
directory: search directory tree for files	find(1)
directory: write portable archive	pax(1)
directory; make a	mkdir(1)
dirent.h - format of directory streams and directory entries	dirent(5)
dirname, basename - extract portions of path names	basename(1)
dirname() - return path name of parent directory	basename(3C)
dirs - print the directory stack	csh(1)
disable accounting or define kernel process accounting output file	acct(1M)
disable compartmentalization feature; query, enable, or	cmpt_tune(1M)
disable - disable LP printers	enable(1)
disable LCPU attribute of the default processor set; dynamically enable or	lcpu_attr(5)
disable mutex-specific or process-wide mutex handoff mode	pthread_mutexattr_getspin_np(3T)
disable option for system to dump memory using multiple dump units when a kernel panic occurs on Integrity systems; enable or	dump_concurrent_on(5)
disable or enable abbreviation of function keys	keypad(3X)
disable or enable flush on interrupt	intrflush(3X)
disable or enable process accounting	acct(2)
disable System V IPC messages at boot time (OBSOLETE); enable or	msg(5)
disable System V IPC semaphores at boot time, enable or	sema(5)
disable System V shared memory, enable or	shm(5)
disable tracking of floating-point registers	uwx_set_nofr(3X)
disable use of certain terminal capabilities	filter(3X)
disable use of device's write cache in the SCSI subsystem (OBSOLETE); enable and	default_disk_ir(5)
disable/enable block during read	nodelay(3X)
disable/enable immediate terminal refresh	immedok(3X)
disable/enable meta-keys	meta(3X)
disable/enable newline translation	nl(3X)
disable/enable queue flushing	noqiflush(3X)
disable/enable terminal echo	echo(3X)
disable/enable use of hardware insert- and delete-character features	idcok(3X)
discard file (bit bucket)	null(7)
discard input	flushinp(3X)
discipline; set terminal type, modes, speed, and line	getty(1M)
disconnect information (X/OPEN TLI-XTI)	t_rcvdis(3)
disconnect request; send user-initiated (X/OPEN TLI-XTI)	t_snddis(3)
Discovery cache display and control, IPv6 Neighbor	ndp(1M)
Discovery Protocol, NDP; Neighbor	ndp(7P)
disjointed text spaces; execution time profile for	sprofil(2)
disk accounting data, disk usage by user ID	diskusg(1M)
disk array, formatting a LUN for	format(1M)
disk blocks (Berkeley version); report number of free	bdf(1M)
disk blocks, report number of free, CDFS, HFS, or NFS file system	df_hfs(1M)
disk blocks; report number of free file system	df(1M)
disk clusters; report number of free	dosdf(1)
disk description by its name, get	getdiskbyname(3C)
disk description file	disktab(4)
disk device, describe characteristics of a	diskinfo(1M)
disk device, direct access drivers	disk(7)
disk - direct disk device access drivers	disk(7)
disk directory format	dir(4)
disk or partition DDS tape; initialize	mediainit(1)
disk quota status of specified file system, determine	fsclean(1M)
disk quotas	quota(5)
disk quotas; edit	edquota(1M)

Index

All Volumes

Description	Entry Name(Section)
disk quotas; manipulate	quotactl(2)
disk section sizes; calculate default	disksecn(1M)
disk space used for caching file systems with CacheFS; administer	cfsadmin(1M)
disk storage space, preallocate	prealloc(1)
disk storage, preallocate fast	prealloc(2)
disk usage accounting records; create	acct(1M)
disk usage and limits; display	quota(1)
disk usage by login name; compute	acct(1M)
disk usage, generate disk accounting data by user ID	diskusg(1M)
disk usage, summarize	du(1)
disk, flush unwritten system buffers to	sync(1M)
disk, get information for a	pstat(2)
disk, SCSI direct access device driver	scsi_disk(7)
disk, synchronize a file's in-core state with its state on	fsync(2)
disk: periodically flush unwritten system buffers to disk	syncer(1M)
diskaudit_flush_interval - determine time interval (in secs) for flushing audit records	diskaudit_flush_interval(5)
diskinfo - describe characteristics of a disk device	diskinfo(1M)
diskless client configuration information file	info(4)
Disks and File Systems tool of HP System Management Homepage (HP SMH); launch the	fsweb(1M)
disksecn - calculate default disk section sizes	disksecn(1M)
disktab - disk description file	disktab(4)
diskusg - generate disk accounting data by user ID	diskusg(1M)
dismount (unmount) CacheFS file systems	mount_cachefs(1M)
dismount (unmount) CDFS file systems	mount_cdfs(1M)
dismount (unmount) NFS file systems	mount_nfs(1M)
display address resolution	arp(1M)
display an EVM event	evmshow(1)
display and modify boot variables in stable storage	setboot(1M)
display and update information about top processes on system	top(1)
display call graph execution profile data	gprof(1)
display compartment rules	getrules(1M)
display current status of power for cells and I/O chassis; turn on/off or	frupower(1M)
display elm user and system aliases	elmalias(1)
display file on soft-copy terminals	pg(1)
display information about a hardware partitionable complex	parstatus(1)
display information about computer system	uname(1)
display information about LVM logical volumes	lvdisplay(1M)
display information about LVM volume groups	vgdisplay(1M)
display information about physical volumes in LVM volume group	pvdisplay(1M)
display information about the Partition Command Line Interface	partition(5)
display information residing in the user database, /var/adm/userdb	userdbget(1M)
display job information and remove jobs	swjob(1M)
display LAN device configuration and status	lanscan(1M)
display LP spooler performance analysis information	lpana(1M)
display message in standard format	pfmt(3C)
display monitor profile data	prof(1)
display name of current host system	hostname(1)
display Network Information Service domain name	domainname(1)
display (print) arguments	echo(1)
display protected password database	getprpw(1M)
display requested audit information	audisp(1M)
display security attributes of a process	getprocxsec(1M)
display security attributes of binary executable(s)	getfilexsec(1M)
display system and user login data	logins(1M)
display system definition	sysdef(1M)
display user login data	listusers(1)
display width for user and group names; get current	ug_display_width(3C)
display/keyboard data order, convert file	forder(1)
display/set audit trail information	audsys(1M)
display; format events for	EvmEventFormat(3)

Description	Entry Name(Section)
display; long user and group name enablement and	lugadmin(1M)
displayable form; dump an event in	EvmEventDump(3)
displays formatted message on standard error and console	fmtmsg(3C)
displays process address information and open file descriptors	pmap(1)
displays the last part of the mail log	mtail(1M)
distance functions, Euclidean (hypotenuse)	hypot(3M)
distribute, install, monitor, create, and manage software	sd(5)
distributed file system packages; file that registers	fstypes(4)
distribution program; remote file	rdist(1)
div() - integer division and remainder	div(3C)
divide mirrored LVM logical volume into two logical volumes	lvsplit(1M)
division and remainder; integer	div(3C)
divpage - divide pages for two-sided printing	lpfilter(1)
dladdr() - get the symbolic information for an address	dladdr(3C)
dlclose() - close a shared object	dlclose(3C)
dld.sl - dynamic loader	dld.sl(5)
dld.so - dynamic loader	dld.so(5)
dld_getenv() - explicit load of shared libraries	shl_load(3X)
dld_getenv_pa() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
dlerrno() - get error code information from dynamic linking process	dlerrno(3C)
dlerror() - get diagnostic information from dynamic linking process	dlerror(3C)
dlget() - retrieve information on loaded module (program or shared library)	dlget(3C)
dlgetfileinfo() - return file information for a library prior to loading it	dlgetfileinfo(3C)
dlgetmodinfo() - retrieve information on loaded module (program or shared library)	dlgetmodinfo(3C)
dlgetname() - retrieve name of load module	dlgetname(3C)
dlmodadd() - register information about dynamically generated functions	dlmodadd(3C)
dlmodinfo() - retrieve information on loaded module (program or shared library)	dlmodinfo(3C)
dlmodremove() - remove information registered using dlmodadd	dlmodremove(3C)
dlopen()/shl_load(); list the dynamic libraries linked into each process, including shared objects explicitly attached using	pldd(1)
dlopen() - open a shared library	dlopen(3C)
dlopen() - open a shared library on Integrity systems	dlopen_ia(3C)
dlopen() - open an HP 9000 shared library	dlopen_pa(3C)
dlopen_ia - open a shared library on Integrity systems	dlopen_ia(3C)
dlopen_pa - open an HP 9000 shared library; open an HP 9000 64-bit shared library with explicit load address	dlopen_pa(3C)
dlopene() - open a shared library on Integrity systems	dlopen_ia(3C)
dlopene() - open an HP 9000 64-bit shared library with explicit load address	dlopen_pa(3C)
dlpi - data link provider interface	dlpi(7)
DLPI streams allowed on the system; maximum number of cloned	dlpi_max_clones(5)
DLPI, data link provider interface	dlpi(7)
DLPI, data link provider interface standard header file	dlpi(4)
DLPI, HP specific extensions for DLPI	dlpi_ext(4)
dlpi.h - data link provider interface standard header file	dlpi(4)
dlpi_drv.h - header file for DLPI	dlpi_drv(4)
dlpi_ext.h - HP specific extensions for DLPI	dlpi_ext(4)
dlpi_max_clones - maximum number of cloned DLPI streams allowed on the system	dlpi_max_clones(5)
dlsetlibpath() - set the dynamic search path used to locate shared libraries	dlsetlibpath(3C)
dlsym() - get address of symbol in shared object	dlsym(3C)
DMA pool; the amount of memory to reserve for the 32-bit	dma32_pool_size(5)
dma32_pool_size - the amount of memory to reserve for the 32-bit DMA pool	dma32_pool_size(5)
dmesg - collect system diagnostic messages to form error log	dmesg(1M)
dmpxlt - dump iconv translation tables to a readable format	dmpxlt(1)
dn_comp() - resolver routines	resolver(3N)
dn_expand() - resolver routines	resolver(3N)
DNLC; number of locks for the Directory Name Lookup Cache	dnlc_hash_locks(5)
dnlc_hash_locks - number of locks for the Directory Name Lookup Cache (DNLC)	dnlc_hash_locks(5)
DNS lookup utility	host(1)
DNS service daemon for NIS	rpc.nisd_resolv(1M)
DNS update utility; Dynamic	nsupdate(1)
DNSSEC keyset signing tool	dnssec-signkey(1)

Index

All Volumes

Description	Entry Name(Section)
DNSSEC zone signing tool	dnssec-signzone(1)
DNSSEC, produces a set of DNSSEC keys	dnssec-makekeyset(1)
dnssec-keygen - key generation tool for DNSSEC	dnssec-keygen(1)
dnssec-makekeyset - produces a set of DNSSEC keys	dnssec-makekeyset(1)
dnssec-signkey - DNSSEC keyset signing tool	dnssec-signkey(1)
dnssec-signzone - DNSSEC zone signing tool	dnssec-signzone(1)
DNSSEC; key generation tool for	dnssec-keygen(1)
do nothing and return zero or non-zero exit status	true(1)
documentation, introduction to HP-UX	intro(9)
documentation; accessing and ordering HP-UX	manuals(5)
documents, format and print using the mm macros	mm(1)
documents, MM macro package for formatting	mm(5)
ddisk - perform disk accounting	acctsh(1M)
domain information groper	dig(1M)
domain name server; configuration file for Internet	named.conf(4)
domain name server; Internet	named(1M)
domain name server; send signals to the	sig_named(1M)
domain name; set or display name of Network Information Service	domainname(1)
domain protocol, local communication	UNIX(7P)
domain; get or set name of current NIS	getdomainname(2)
domainname - set or display NIS domain name	domainname(1)
dontdump - defines which classes of kernel memory pages are not dumped when a kernel panic occurs	dontdump(5)
DOS formats; convert ASCII file format between HP-UX and	dos2ux(1)
DOS interchange format	dosif(4)
dos2ux - convert ASCII file format between HP-UX and DOS formats	dos2ux(1)
dosdf - report number of free disk clusters	dosdf(1)
DOSIF - DOS interchange format	dosif(4)
dot	glossary(9)
dot-dot	glossary(9)
dot-oh	glossary(9)
dot-oh file	glossary(9)
dot-oh format	glossary(9)
double-precision number; convert wide character string to	wctod(3C)
doupdate() , refresh() , wnoutrefresh() , wrefresh() - refresh windows and lines	doupdate(3X)
downshifting	glossary(9)
dpp - dedicated ports parser used by DDFA software	dpp(1M)
drand48() , erand48() - generate double-precision pseudo-random numbers	drand48(3C)
draw borders from complex characters and renditions	border_set(3X)
draw borders from complex characters and renditions	box_set(3X)
draw borders from single-byte characters and renditions	border(3X)
draw borders from single-byte characters and renditions	box(3X)
draw lines from complex characters and renditions	hline_set(3X)
draw lines from single-byte characters and renditions	hline(3X)
drift analyzer; system configuration	bastille_drift(1M)
driver	<i>see special files</i>
driver eschgr plug-in for scsimgr; SCSI class	scsimgr_eschgr(7)
driver esdisk plug-in for scsimgr; SCSI class	scsimgr_esdisk(7)
driver estape plug-in for scsimgr; SCSI class	scsimgr_estape(7)
driver to a device; bind a	jobind(1M)
driver, block mode terminal	blmode(7)
driver, PS/2 keyboard and mouse devices	ps2(7)
driver, SCSI direct access device	scsi_disk(7)
driver, SCSI media changer device	autochanger(7)
driver, SCSI pass-through device	sioc_io(7)
Driver-Based (FCD) and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2	fcmsutil(1M)
driver: STREAMS Administrative Driver	sad(7)
driver; HP-HIL device	hil(7)
driver; HP-HIL mapped keyboard	hilkbd(7)
driver; network interface management command for btlan	nwmgr_btlan(1M)

Description	Entry Name(Section)
driver; pseudo-terminal	pty(7)
driver; STREAMS log	strlog(7)
drivers in the system, list device	lsdev(1M)
drivers to interact with DLPI, definitions of interfaces for device	dlpi_drv(4)
drivers: direct disk device access drivers	disk(7)
drivers; Small Computer System Interface device	scsi(7)
DSF; redirect the persistent device special file from one device to a different device	io_redirect_dsf(1M)
dspcat - display all or part of a message catalog	dspcat(1)
dspmsg - display a selected message from a message catalog	dspmsg(1)
dst - set daylight savings time	timezone(5)
du - summarize disk usage	du(1)
dump an event in displayable form	EvmEventDump(3)
dump analyzer; invoke KWDB, the source level kernel debugger and crash	kwdb(1M)
dump and restore protocol module, remote magnetic tape	rmt(1M)
dump configuration, get information for a system's crash	pstat(2)
dump device, get information for a crash	pstat(2)
dump file system information	dumpfs(1M)
dump file; octal and hexadecimal	od(1)
dump iconv translation tables to a readable format	dmpxlt(1)
dump - incremental file system dump (for backups)	dump(1M)
dump information contained in object files	elfdump(1)
dump information contained in SOM object files	odump(1)
dump memory using multiple dump units when a kernel panic occurs on Integrity systems; enable or disable option for system to	dump_concurrent_on(5)
dump units when a kernel panic occurs on Integrity systems; enable or disable option for system to dump memory using multiple	dump_concurrent_on(5)
dump unwritten system buffers to disk	sync(1M)
dump unwritten system buffers to disk periodically	syncer(1M)
dump volume, remove LVM logical volume link	lvrmboot(1M)
dump volume; prepare LVM logical volume to be root, boot, primary swap, or	lvlnboot(1M)
dump window to and reload window from a file	getwin(3X)
dump, crash dump access library	libcrash(5)
Dump, initiates, configures, and stops Live	livedump(1M)
dump_compress_on - selects whether the system dumps memory pages compressed or uncompressed when a kernel panic occurs	dump_compress_on(5)
dump_concurrent_on - enable or disable option for system to dump memory using multiple dump units when a kernel panic occurs on Integrity systems	dump_concurrent_on(5)
dumpfs - dump file system information	dumpfs(1M)
dumpmsg - create message catalog file for modification	findmsg(1)
dumps memory pages compressed or uncompressed when a kernel panic occurs; selects whether the system	dump_compress_on(5)
dumps; configure system crash	crashconf(1M)
dumps; configure system crash	crashconf(2)
dup() - duplicate an open file descriptor	dup(2)
dup2() - duplicate an open file descriptor to a specific slot	dup2(2)
duplicate a window	dupwin(3x)
duplicate an open file descriptor	dup(2)
duplicate an open file descriptor to a specific slot	dup2(2)
duplicate directory trees	cpio(1)
duplicate directory trees; copy file archives in and out;	cpio(1)
duplicate entries from gsscred mapping table; remove	gsscred_clean(1M)
duplicate entries in a table, eliminate	lsearch(3C)
duplicate events	evmlogger(1M)
dupwin() - duplicate a window	dupwin(3x)
dynamic dependencies of executable files or shared libraries on Integrity systems; list	ldd_ia(1)
dynamic dependencies of executable files or shared libraries on PA-RISC systems; list	ldd_pa(1)
dynamic dependencies of executable files or shared libraries; list	ldd(1)
Dynamic DNS update utility	nsupdate(1)
Dynamic Host Configuration Protocol (DHCP) client database converter	dhecpdb2conf(1M)
Dynamic Host Configuration Protocol Server daemon for IPv6	dhecpv6d(1M)
dynamic information about the system, get	pstat(2)

Index

All Volumes

Description	Entry Name(Section)
dynamic libraries linked into each process, including shared objects explicitly attached using <code>dlopen()/shl_load()</code> ; list the	pldd(1)
dynamic linking process, diagnostic information	dlerrno(3C)
dynamic linking process, diagnostic information	dlerror(3C)
dynamic loader	dld.sl(5)
dynamic loader	dld.so(5)
dynamic loader	glossary(9)
dynamically enable or disable LCPU attribute of the default processor set	lcpu_attr(5)
dynamically loadable kernel modules, change global search path	modpath(2)
dynamically loaded kernel module; get information for a	modstat(2)
e - extended line-oriented text editor	ex(1)
echo - echo (print) arguments	csh(1)
echo - echo (print) arguments	echo(1)
echo - echo (print) arguments	ksh(1)
echo - echo (print) arguments	sh-posix(1)
echo () - enable/disable terminal echo	echo(3X)
echo packets	ping(1M)
echo single-byte character and rendition to a window and refresh	echochar(3X)
echo, suppress while reading password from terminal	getpass(3C)
ECHO_REQUEST packets	ping(1M)
echo_wchar () - write a complex character and immediately refresh the window	echo_wchar(3X)
echochar () - echo single-byte character and rendition to a window and refresh	echochar(3X)
ecvt () - convert floating-point number to string	ecvt(3C)
ed - line-oriented text editor	ed(1)
edata - first address beyond initialized program data region	end(3C)
edisk - SCSI direct access device driver	scsi_disk(7)
edit - beginner's line-oriented text editor	ex(1)
edit disk quotas	edquota(1M)
edit - extended line-oriented text editor	ex(1)
edit files that are under access control; let authorized users	privedit(1M)
edit the password file using vi editor	vipw(1M)
editing activity, print current SCCS file	sact(1)
editing of a command's authorization and privilege information in the privrun database; noninteractive	cmdprivadm(1M)
editing of role-related information in RBAC databases; noninteractive	roleadm(1M)
editing of the authorization information in the RBAC databases, non-interactive	authadm(1M)
editor and command history for interactive programs; input	ied(1)
editor for Integrity systems; link	ld_ia(1)
editor for PA-RISC systems; link	ld_pa(1)
editor; extended line-oriented text	ex(1)
editor; extended screen-oriented text	vi(1)
editor; line-oriented text	ed(1)
editor; link	ld(1)
editor; screen-oriented (visual) text	vi(1)
editor; stream text	sed(1)
edquota - edit disk quotas	edquota(1M)
effective access rights to a file, get a user's	getaccess(2)
effective and real user IDs, set	setreuid(2)
effective current user ID; print	whoami(1)
effective group ID	glossary(9)
effective group IDs; get real user, effective user, real group, and	getuid(2)
effective group IDs; sets the real and	setregid(2)
effective user and group IDs; set	seteuid(2)
effective user ID	glossary(9)
effective user ID, get	getresuid(3)
effective user, real group, and effective group IDs; get real user,	getuid(2)
effective, group ID, get	getresuid(3)
effective, real, and/or saved user or group IDs, set	setresuid(2)
EFI directory, list EFI file information or contents of an	efi_ls(1M)
EFI directory, make an	efi_mkdir(1M)
EFI directory, remove an	efi_rmdir(1M)

Description	Entry Name(Section)
efi - Extensible Firmware Interface description	efi(4)
EFI file information or contents of an EFI directory, list	efi_ls(1M)
EFI file system header on a device file, write an	efi_fsinit(1M)
EFI file, copy to or from	efi_cp(1M)
EFI file, remove an	efi_rm(1M)
efi_cp - copy to or from EFI file	efi_cp(1M)
efi_fsinit - write an EFI file system header on a device file	efi_fsinit(1M)
efi_ls - list EFI file information or contents of an EFI directory	efi_ls(1M)
efi_mkdir - make an EFI directory	efi_mkdir(1M)
efi_rm - remove an EFI file	efi_rm(1M)
efi_rmdir - remove an EFI directory	efi_rmdir(1M)
egrep - search a file for a pattern	grep(1)
electronic address router	pathalias(1)
electronic mail, screen-oriented interface	elm(1)
ELF - executable and linking format object files	elf(3E)
ELF file; make file descriptor for	elf_begin(3E)
ELF files, set fill byte for	elf_fill(3E)
ELF files; get section information for	elf_getscn(3E)
ELF library and application versions; coordinate	elf_version(3E)
ELF library error handling	elf_error(3E)
elf() - object file access library	elf(3E)
elf32 or elf64 file; retrieve class-dependent object file header for	elf_getehdr(3E)
elf32_fsize() - return the size of an object file type for elf32 files	elf_fsize(3E)
elf32_getehdr() - retrieve class-dependent object file header for elf32 or elf64 file	elf_getehdr(3E)
elf32_getphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf32_getshdr() - retrieve class-dependent section header for ELF files	elf_getshdr(3E)
elf32_newehdr() - retrieve class-dependent object file header for elf32 or elf64 file	elf_getehdr(3E)
elf32_newphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf32_xlatetof() - class-dependent data translation of ELF files	elf_xlate(3E)
elf32_xlatetom() - class-dependent data translation of ELF files	elf_xlate(3E)
elf64 file; retrieve class-dependent object file header for elf32 or	elf_getehdr(3E)
elf64_fsize() - return the size of an object file type for elf64 files	elf_fsize(3E)
elf64_getehdr() - retrieve class-dependent object file header for elf32 or elf64 file	elf_getehdr(3E)
elf64_getphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf64_getshdr() - retrieve class-dependent section header for ELF files	elf_getshdr(3E)
elf64_newehdr() - retrieve class-dependent object file header for elf32 or elf64 file	elf_getehdr(3E)
elf64_newphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf64_xlatetof() - class-dependent data translation of ELF files	elf_xlate(3E)
elf64_xlatetom() - class-dependent data translation of ELF files	elf_xlate(3E)
elf_begin() - make file descriptor for ELF file	elf_begin(3E)
elf_cntl() - control a file descriptor for ELF files	elf_cntl(3E)
elf_end() - finish using an ELF object file	elf_end(3E)
elf_errmsg() - ELF library error handling	elf_error(3E)
elf_errno() - ELF library error handling	elf_error(3E)
elf_fill - set fill byte for ELF files	elf_fill(3E)
elf_flagdata() - manipulate flags for ELF files	elf_flag(3E)
elf_flagehdr() - manipulate flags for ELF files	elf_flag(3E)
elf_flagelf() - manipulate flags for ELF files	elf_flag(3E)
elf_flagphdr() - manipulate flags for ELF files	elf_flag(3E)
elf_flagscn() - manipulate flags ELF files	elf_flag(3E)
elf_flagsshdr() - manipulate flags for ELF files	elf_flag(3E)
elf_getarhdr() - retrieve archive member header for ELF files	elf_getarhdr(3E)
elf_getarsym() - retrieve archive symbol table	elf_getarsym(3E)
elf_getbase() - get the base offset for an object file	elf_getbase(3E)
elf_getdata() - manipulate section data for ELF files	elf_getdata(3E)
elf_getehdr - retrieve class-dependent object file header for elf32 or elf64 file	elf_getehdr(3E)
elf_getident() - retrieve file identification data for ELF files	elf_getident(3E)
elf_getscn() - get section information for ELF files	elf_getscn(3E)
elf_hash() - compute hash value for ELF files	elf_hash(3E)
elf_kind() - determine file type for ELF files	elf_kind(3E)
elf_ndxscn() - get section information for ELF files	elf_getscn(3E)

Index

All Volumes

Description	Entry Name(Section)
elf_newdata () - manipulate section data for ELF files	elf_getdata (3E)
elf_newscn () - get section information for ELF files	elf_getscn (3E)
elf_next () - provide sequential archive member access for ELF files	elf_next (3E)
elf_nextscn () - get section information for ELF files	elf_getscn (3E)
elf_rand () - random archive member access for ELF files	elf_rand (3E)
elf_rawdata () - manipulate section data for ELF files	elf_getdata (3E)
elf_rawfile () - retrieve uninterpreted file contents for ELF files	elf_rawfile (3E)
elf_strptr () - make a string pointer for ELF files	elf_strptr (3E))
elf_update () - update an ELF descriptor	elf_update (3E)
elf_version () - coordinate ELF library and application versions	elf_version (3E)
elfdump - dump information contained in object files	elfdump (1)
eliminate .so's from nroff input	soelim (1)
eliminate a file or directory	rm (1)
eliminate adjacent repeated lines in a file	uniq (1)
eliminate duplicate entries in a table	lsearch (3C)
eliminate multiple adjacent blank lines, reduce to single blank line	ssp (1)
elm - process electronic mail through a screen-oriented interface	elm (1)
elm user and system aliases, verify and display	elmalias (1)
elmalias - display and verify elm user and system aliases	elmalias (1)
emacs editing mode	sh-posix (1)
empty OID set, create a new	gss_create_empty_oid_set (3)
emtui - provide displaying options for HP-UX errors defined in the Common Error Repository (CER)	emtui (1)
emulate /usr/share/lib/termcap access routines	termcap (3X)
emulate PA-RISC HP-UX applications on Itanium-based systems running HP-UX	Aries (5)
Emulation module, STREAMS pty	psem (7)
emulator on an Integrity system; maximum size (in bytes) of the stack for a user process running under the PA-RISC	pa_maxssiz (5)
Emulator, number of scrollable lines used by the Internal Terminal	scroll_lines (5)
emulator; call another (UNIX) system, terminal	cu (1)
enable and disable use of device's write cache in the SCSI subsystem (OBSOLETE)	default_disk_ir (5)
enable device or file system for paging	swapon (1M)
enable - enable LP printers	enable (1)
enable high resolution timers support	hires_timeout_enable (5)
enable maximum length expansion of the system node and host names	expanded_node_host_names (5)
enable or disable block during read	nodelay (3X)
enable or disable flush on interrupt	intrflush (3X)
enable or disable immediate terminal refresh	immedok (3X)
enable or disable LCPU attribute of the default processor set; dynamically	lcpu_attr (5)
enable or disable option for system to dump memory using multiple dump units when a kernel panic occurs on Integrity systems	dump_concurrent_on (5)
enable or disable process accounting	acct (2)
enable or disable System V IPC messages at boot time (OBSOLETE)	mesg (5)
enable or disable use of hardware insert- and delete-character features	idcok (3X)
enable, or disable compartmentalization feature; query,	empt_tune (1M)
enable/disable abbreviation of function keys	keypad (3X)
enable/disable meta-keys	meta (3X)
enable/disable newline translation	nl (3X)
enable/disable queue flushing	noqiflush (3X)
enable/disable terminal echo	echo (3X)
enable/disable the NFS server's source port verification check	nfs_portmon (5)
enable/disable tracking of current and root directories for auditing subsystem	audit_track_paths (5)
enable_idds - enable intrusion detection data source	enable_idds (5)
enabled for swap; maximum number of devices that can be	nswapdev (5)
enabled for swap; maximum number of file systems that can be	nswarfs (5)
enablement and display; long user and group name	lugadmin (1M)
enables write calls to return before write operation is complete	fs_async (5)
enables/disables PCI Error Recovery	pci_ah_enable (5)
encode a binary file for transmission by mailer	uuencode (1)
encode and decode files	crypt (1)
encode/decode a binary file for transmission by mailer	uuencode (1)

Description	Entry Name(Section)
encoded uuencode file; format of an	uuencode(4)
encrypt files	crypt(1)
encrypt, attach a message integrity code (MIC) to a message and optionally	gss_wrap(3)
encryption key, generate a DES	makekey(1)
encryption keys file format; PPP	ppp.Keys(4)
encryption keys, server for storing	keyserv(1M)
encryption on large strings; generate hashing	bigcrypt(3C)
encryption, hashing, generate	crypt(3C)
encryption, password	crypt(3C)
end - first address beyond uninitialized program data region	end(3C)
end locations of allocated regions in program	end(3C)
end network host entry	gethostent(3N)
end of line, clear from cursor to end of line	clrtoeol(3X)
end part of a file, get lines from	tail(1)
end protocol entry; get, set, or	getprotoent(3N)
end service entry	getservent(3N)
end - terminate foreach or while loop	csh(1)
end-of-file	glossary(9)
endbwent () - write records into new wtmps and btmps database	bwtmps(3C)
enddvagent () - free memory and close file for trusted system	getdvagent(3)
endfsent () - close file system descriptor file	getfsent(3X)
endgrent () - close currently open group file	getgrent(3C)
endhostent () - end network host entry	gethostent(3N)
endhostent_r () - end network host entry (thread-safe)	gethostent(3N)
endmntent () - close file system descriptor file	getmntent(3X)
endnetconfig () - get /etc/netconfig entry corresponding to NETPATH component	getnetpath(3N)
endnetconfig () - get network configuration data base entry	getnetconfig(3N)
endnetent (): end network entry	getnetent(3N)
endnetgrent () - get network group entry	getnetgrent(3C)
endpoint for communication; create an	socket(2)
endpoint; manage options for a transport	t_optmgmt(3)
endprdfent () - close system default database entry for trusted system	getprdfent(3)
endprdfent () - manipulate system default database entry for a trusted system	getprdfent(3)
endprotoent () - end protocol entry	getprotoent(3N)
endprotoent_r () - end protocol entry (thread-safe)	getprotoent(3N)
endprpwent () - manipulate protected password database entries (for trusted systems only)	getprpwent(3)
endprtcent () - manipulate terminal control database entry	getprtcent(3)
endpwent () - close currently open password file	getpwent(3C)
endrpcent () - get RPC entry	getrpcent(3C)
endservent () - end service entry	getservent(3N)
endservent_r () - end service entry (thread-safe)	getservent(3N)
endspent () - close currently open secure password file	getspent(3C)
endspwent () - close currently open secure password file on trusted systems	getspwent(3X)
endspwent_r () - get secure password file entry on trusted systems	getspwent(3X)
endsw - terminate switch statement	csh(1)
endusershell () - close legal user shells file	getusershell(3C)
endutent () - close currently open utmp file	getut(3C)
endutent_r () - close currently open utmp file	getut(3C)
ENDUTSENT () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
endutsent () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
endutxent () - close currently open utmpx file	getutx(3C)
endwin () - suspend Curses session	endwin(3X)
enforced lock on an open file; apply or remove an advisory or	flock(2)
entries (for trusted systems only); manipulate protected password database	getprpwent(3)
entries from a directory, get in a file-system-independent format	getdirentries(2)
entries from name list; get	nlist(3C)
entries in a table, eliminate duplicate	lsearch(3C)
entry for a trusted system; manipulate device assignment database	getdvagent(3)
entry for a trusted system; manipulate system default database	getprdfent(3)
entry from secure password file; get	getspwent(3X)

Index

All Volumes

Description	Entry Name(Section)
entry on trusted systems; get secure password file	getspwnt(3X)
entry, get file system descriptor file (BSD 4.2 compatibility only)	getfsent(3X)
entry, manipulate terminal control database	getprtcent(3)
entry, network group, get or set	getnetgrent(3C)
entry; access utmp file	getut(3C)
entry; get hostname and address	getaddrinfo(3N)
entry; get RPC	getrpcnt(3C)
entry; write password file	putpwent(3C)
env - set environment for command execution	env(1)
envd - system physical environment daemon	envd(1M)
envelope of current sendmail message; removes a recipient from	smfi_delrept(3N)
environ - user environment variables	environ(5)
environment	glossary(9)
environment configuration command; NFS	setoncenv(1M)
environment daemon; system physical	envd(1M)
environment for command execution, set	env(1)
environment list, search for value of specified variable name	getenv(3C)
environment macros and functions, floating-point	fenv(5)
environment variable, search environment list for value of	getenv(3C)
environment variables, user	environ(5)
environment, change or add value to	putenv(3C)
environment, change or add value to	setenv(3C)
environment, clear the process	clearenv(3C)
environment, login shell script to set up user's	profile(4)
environment, print out the	printenv(1)
environment, save/restore stack for non-local goto	setjmp(3C)
environment: getting floating-point	fegetenv(3M)
environment: saving floating-point	feholdexcept(3M)
environment: setting floating-point	fesetenv(3M)
environment: updating floating-point	feupdateenv(3M)
EOF	glossary(9)
Epoch	glossary(9)
eqmem_limit - determines the maximum amount (in MB) of equivalently mapped memory which can be allocated after boot	eqmem_limit(5)
eqmemsize - determines the minimum size (in pages) of the equivalently mapped reserve pool (OBSOLETE)	eqmemsize(5)
erase character	eraseswchar(3X)
erase character, single-byte	erasechar(3X)
erase() - clear a window	clear(3X)
erase terminal screen	clear(1)
erasechar() - single-byte erase character	erasechar(3X)
eraseswchar() - current erase character	eraseswchar(3X)
erfcf() - complementary error function	erf(3M)
erf() - error function	erf(3M)
erfcf() - complementary error function (float)	erf(3M)
erfc1() - complementary error function (long double)	erf(3M)
erfcq() - complementary error function (quad)	erf(3M)
erfcw() - complementary error function (extended)	erf(3M)
erff() - error function (float)	erf(3M)
erfl() - error function (long double)	erf(3M)
erfq() - error function (quad)	erf(3M)
erfw() - error function (extended)	erf(3M)
errno() - error indicator for system calls	errno(2)
error checking; execute link() and unlink() system calls without	link(1M)
error codes on failure, get	rpc_gss_get_error(3N)
error codes; Service Location Protocol (SLP)	SLPError(3N)
error functions	erf(3M)
error handling, ELF library	elf_error(3E)
error indicator for system calls	errno(2)
error information from unit data error indication (X/OPEN TLI-XTI)	t_rcvuderr(3)
error log files, remove outdated STREAMS error log files	strclean(1M)

Description	Entry Name(Section)
error log, collect system diagnostic messages to form	dmesg(1M)
error message function (X/OPEN TLI-XTI)	t_error(3)
error message string, get PAM	pam_strerror(3)
error message; produce (X/OPEN - XTI)	t_strerror(3)
error messages from C source into a file; extract	mkstr(1)
error messages from the STREAMS log driver	strerr(1M)
error messages; write system	perror(3C)
error metadata; update the Common Error Repository (CER) with	cerupdate(1)
error or warning message, libcrash, print	cr_perror(3)
error processing with t_rcvuderr(3)	t_sndudata(3)
Error Recovery; enables/disables PCI	pci_ah_enable(5)
error reply code to a multi-line response; sets default SMTP	smfi_setmlreply(3N)
error reply code; sets the default SMTP	smfi_setreply(3N)
error status, asynchronous I/O	aio_error(2)
errors, find spelling	spell(1)
errors, library routines for server side remote procedure call errors	rpc_svc_err(3N)
eschgr plug-in for scsimgr; SCSI class driver	scsimgr_eschgr(7)
eschgr - SCSI interfaces for medium changer device	autochanger(7)
esctl/sctl ; SCSI pass-through driver	scsi_ctl(7)
esdisk plug-in for scsimgr; SCSI class driver	scsimgr_esdisk(7)
esmd - Essential Services Monitor Daemon	esmd(1M)
Essential Services Monitor Daemon	esmd(1M)
establish a subscription for event notification	EvmConnSubscribe(3)
establish an outgoing terminal line connection	dial(3C)
establish connection with another transport user (X/OPEN TLI-XTI)	t_connect(3)
establish or destroy connection with the EVM daemon	EvmConnCreate(3)
estape plug-in for scsimgr; SCSI class driver	scsimgr_estape(7)
etext - first address beyond program text region	end(3C)
Ethernet) client configuration file, PPPoE (Point to Point Protocol over	pppoe.conf(4)
Ethernet) client, PPPoE (Point to Point Protocol over	pppoe(1)
Ethernet) relay configuration file, PPPoE (Point to Point Protocol over	pppoerd.conf(4)
Ethernet) relay, PPPoE (Point to Point Protocol over	pppoerd(1M)
Ethernet) server configuration file, PPPoE (Point to Point Protocol over	pppoesd.conf(4)
Ethernet) server daemon, PPPoE (Point-to-Point Protocol over	pppoesd(1M)
EUC; set and get code widths for ldterm	eucset(1)
Euclidean distance (hypotenuse) functions	hypot(3M)
eucset - set and get EUC code widths for ldterm	eucset(1)
eval - read arguments as shell input and execute result	csh(1)
eval - read arguments as shell input and execute resulting commands	ksh(1)
eval - read arguments as shell input and execute resulting commands	sh-posix(1)
evaluate arguments as an expression	expr(1)
((- evaluate arithmetic expression	sh-posix(1)
evaluate condition for true or false	test(1)
evaluator routines; event filter	EvmFilterCreate(3)
event audit status; change or display	audevent(1M)
event filter	evmfilterfile.4
event filter evaluator routines	EvmFilterCreate(3)
event filter; EVM (Event Management)	EvmFilter(5)
event forwarding	evmlogger(1M)
event in displayable form; dump an	EvmEventDump(3)
event items; create and manipulate	EvmItemGet(3)
event logging	evmlogger(1M)
event loop; passes control to the libmilter	smfi_main(3N)
Event Management	evm.auth(4)
Event Management	evmchannel.conf(4)
Event Management	evmchmgr(1M)
Event Management	evmdaemon.conf(4)
Event Management	evmfilterfile.4
Event Management	evmlogger(1M)
Event Management	evmtemplate(4)
event management	EVM(5)

Index

All Volumes

Description	Entry Name(Section)
event management (EVM) callback function	EvmCallback(5)
Event Management (EVM) daemon; connection to the	EvmConnection(5)
Event Management (EVM) event filter; EVM	EvmFilter(5)
Event Manager channel manager	evmchmgr(1M)
Event Manager configuration files; reload	evmreload(1M)
Event Manager daemon	evmd(1M)
Event Manager filter file	evmfilterfile(4)
Event Manager logger	evmlogger(1M)
Event Manager template file	evmtemplate(4)
Event Manager; start the	evmstart(1M)
Event Manager; stop the	evmstop(1M)
event name; match EVM	EvmEventNameMatch(3)
event notification; establish a subscription for	EvmConnSubscribe(3)
event on transport endpoint; look at current event (X/OPEN TLI-XTI)	t_look(3)
event registration	evmd(1M)
event rights	evm.auth(4)
event service functions	EvmSrvStart(3)
event subscriptions; enables you to view, create, modify, and delete	evweb_subscribe(1)
event suppression	evmlogger(1M)
event template	evmtemplate(4)
event templates	evmd(1M)
event trace messages to standard output, write STREAMS event trace messages	strace(1M)
event variables; manipulate	EvmVarGet(3)
event viewer tool (a Web interface); start the HP-UX hardware	slweb(1M)
event; display an EVM	evmshow(1)
event; perform a data integrity check on an	EvmEventValidate(3)
event; post an EVM	EvmEventPost(3)
event; structure of an EVM	EvmEvent(5)
events and system calls currently being audited; get	getevent(2)
events and system calls to be audited; set current	setevent(2)
events daemon, PCI I/O hotplug (attention button)	hotplugd(1M)
events for display; format	EvmEventFormat(3)
events to and from a file; perform I/O of EVM	EvmEventRead(3)
events to the EVM daemon; post	evmpost(1)
events, audit system, define and describe	audeventstab(4)
events; create and destroy	EvmEventCreate(3)
events; enables you to view and delete	evweb_eventviewer(1)
events; lists different categories of	evweb_list(1)
events; monitor EVM	evmwatch(1)
events; retrieve stored	evmget(1)
events; sort	evmsort(1)
EVM authorization file	evm.auth(4)
EVM callback function; event management	EvmCallback(5)
EVM channel configuration file	evmchannel.conf(4)
EVM Channel Manager	evmchmgr(1M)
EVM channel manager	evmchannel.conf(4)
EVM channel manager, reconfiguration	evmreload(1M)
EVM channel manager, starting	evmstart(1M)
EVM channel manager, stopping	evmstop(1M)
evm channel manager; configuration file	evmchannel.conf(4)
EVM connection; control information for an	EvmConnControl(3)
EVM daemon	evmd(1M)
EVM daemon	evmdaemon.conf(4)
EVM daemon	evmtemplate(4)
EVM daemon configuration file	evmdaemon.conf(4)
EVM daemon, reconfiguration	evmreload(1M)
EVM daemon, starting	evmstart(1M)
EVM daemon, stopping	evmstop(1M)
EVM daemon; establish or destroy connection with the	EvmConnCreate(3)
EVM daemon; maintain connection with the	EvmConnCheck(3)
EVM daemon; post events to the	evmpost(1)

Description	Entry Name(Section)
EVM - event management	EVM(5)
EVM (Event Management) daemon; connection to the	EvmConnection (5)
EVM (Event Management) event filter	EvmFilter (5)
EVM event name; match	EvmEventNameMatch (3)
EVM event; display an	evmshow (1)
EVM event; post an	EvmEventPost (3)
EVM event; structure of an	EvmEvent (5)
EVM events to and from a file; perform I/O of	EvmEventRead (3)
EVM events; monitor	evmwatch (1)
EVM logger	evmlogger (1M)
EVM logger configuration file	evmlogger.conf (4)
EVM logger, reconfiguration	evmreload (1M)
EVM logger, starting	evmstart (1M)
EVM logger, stopping	evmstop (1M)
EVM status code; format text version of	EvmStatusTextGet (3)
evm.auth - EVM authorization file	evm.auth (4)
EVM ; provide information about	evminfo (1)
EvmCallback - event management (EVM) callback function	EvmCallback (5)
evmchannel.conf - EVM channel configuration file	evmchannel.conf (4)
evmchmgr - Event Manager channel manager	evmchmgr (1M)
evmchmgr ; configuration file	evmchannel.conf (4)
EvmConnCheck () - maintain connection with the EVM daemon	EvmConnCheck (3)
EvmConnControl () - control information for an EVM connection	EvmConnControl (3)
EvmConnCreate () - establish or destroy connection with the EVM daemon	EvmConnCreate (3)
EvmConnCreatePoster () - establish or destroy connection with the EVM daemon	EvmConnCreate (3)
EvmConnCreateSubscriber () - establish or destroy connection with the EVM daemon	EvmConnCreate (3)
EvmConnDestroy () - establish or destroy connection with the EVM daemon	EvmConnCreate (3)
EvmConnDispatch () - maintain connection with the EVM daemon	EvmConnCheck (3)
EvmConnConnection - connection to the EVM (Event Management) daemon	EvmConnection (5)
EvmConnFdGet () - establish or destroy connection with the EVM daemon	EvmConnCreate (3)
EvmConnFlush () - maintain connection with the EVM daemon	EvmConnCheck (3)
EvmConnRegistrationGet () - establish a subscription for event notification	EvmConnSubscribe (3)
EvmConnSubscribe () - establish a subscription for event notification	EvmConnSubscribe (3)
EvmConnTemplateScan () - establish a subscription for event notification	EvmConnSubscribe (3)
EvmConnWait () - maintain connection with the EVM daemon	EvmConnCheck (3)
evmd - Event Manager daemon	evmd (1M)
evmdaemon.conf - EVM daemon configuration file	evmdaemon.conf (4)
evmdaemon ; authorization file	evm.auth (4)
evmdaemon ; configuration file	evmdaemon.conf (4)
EvmEvent - structure of an EVM event	EvmEvent (5)
EvmEventCreate () - create and destroy events	EvmEventCreate (3)
EvmEventCreateVa () - create and destroy events	EvmEventCreate (3)
EvmEventDestroy () - create and destroy events	EvmEventCreate (3)
EvmEventDump () - dump an event in displayable form	EvmEventDump (3)
EvmEventDup () - create and destroy events	EvmEventCreate (3)
EvmEventFormat () - format events for display	EvmEventFormat (3)
EvmEventFormatFromTemplate () - format events for display	EvmEventFormat (3)
EvmEventNameMatch () - match EVM event name	EvmEventNameMatch (3)
EvmEventNameMatchStr () - match EVM event name	EvmEventNameMatch (3)
EvmEventPost () - post an EVM event	EvmEventPost (3)
EvmEventPostVa () - post an EVM event	EvmEventPost (3)
EvmEventRead () - perform I/O of EVM events to and from a file	EvmEventRead (3)
EvmEventValidate () - perform a data integrity check on an event	EvmEventValidate (3)
EvmEventWrite () - perform I/O of EVM events to and from a file	EvmEventRead (3)
EvmFilter - EVM (Event Management) event filter	EvmFilter (5)
EvmFilterCreate () - event filter evaluator routines	EvmFilterCreate (3)
EvmFilterDestroy () - event filter evaluator routines	EvmFilterCreate (3)
evmfilterfile - Event Manager filter file	evmfilterfile (4)
EvmFilterIsFile () - event filter evaluator routines	EvmFilterCreate (3)
EvmFilterReadFile () - event filter evaluator routines	EvmFilterCreate (3)
EvmFilterSet () - event filter evaluator routines	EvmFilterCreate (3)

Index

All Volumes

Description	Entry Name(Section)
EvmFilterTest() - event filter evaluator routines	EvmFilterCreate(3)
evmget - retrieve stored events	evmget(1)
evminfo - provide information about EVM	evminfo(1)
EvmItemGet() - create and manipulate event items	EvmItemGet(3)
EvmItemListFree() - create and manipulate event items	EvmItemGet(3)
EvmItemListGet() - create and manipulate event items	EvmItemGet(3)
EvmItemRelease() - create and manipulate event items	EvmItemGet(3)
EvmItemSet() - create and manipulate event items	EvmItemGet(3)
EvmItemSetVa() - create and manipulate event items	EvmItemGet(3)
evmlogger - Event Manager logger	evmlogger(1M)
evmlogger.conf - EVM logger configuration file	evmlogger.conf(4)
evmpost - post events to the EVM daemon	evmpost(1)
evmreload - reload Event Manager configuration files	evmreload(1M)
evmshow - display an EVM event	evmshow(1)
evmsort - sort events	evmsort(1)
EvmSrvMessageGet() - event service functions	EvmSrvStart(3)
EvmSrvStart() - event service functions	EvmSrvStart(3)
evmstart - start the Event Manager	evmstart(1M)
EvmStatusTextGet() - format text version of EVM status code	EvmStatusTextGet(3)
evmstop - stop the Event Manager	evmstop(1M)
evmtemplate - Event Manager template file	evmtemplate(4)
EvmVarFormat() - format events for display	EvmEventFormat(3)
EvmVarGet() - manipulate event variables	EvmVarGet(3)
EvmVarGetOpaque() - manipulate event variables	EvmVarGet(3)
EvmVarGetString() - manipulate event variables	EvmVarGet(3)
EvmVarGetType() - manipulate event variables	EvmVarGet(3)
EvmVarGetXxx() - manipulate event variables	EvmVarGet(3)
EvmVarListFree() - manipulate event variables	EvmVarGet(3)
EvmVarListGet() - manipulate event variables	EvmVarGet(3)
EvmVarRelease() - manipulate event variables	EvmVarGet(3)
EvmVarSet() - manipulate event variables	EvmVarGet(3)
EvmVarSetOpaque() - manipulate event variables	EvmVarGet(3)
EvmVarSetStringI18N() - manipulate event variables	EvmVarGet(3)
EvmVarSetXxx() - manipulate event variables	EvmVarGet(3)
evmwatch - monitor EVM events	evmwatch(1)
evweb commands; provides an overview of	evweb(1)
evweb - provides an overview of evweb commands	evweb(1)
evweb_eventviewer - enables you to view and delete events	evweb_eventviewer(1)
evweb_list - lists different categories of events	evweb_list(1)
evweb_subscribe - enables you to view, create, modify, and delete event subscriptions	evweb_subscribe(1)
ex - extended line-oriented text editor	ex(1)
examine and change blocked signals	sigprocmask(2)
examine and change signal action	sigwait(2)
examine and change the signal mask of the calling thread	pthread_sigmask(3T)
examine pending signals	sigpending(2)
examine signal action	sigaction(2)
exception flags: getting floating-point	fegetexceptflag(3M)
exception flags: setting floating-point	fesetexceptflag(3M)
exception trap enables: getting	fegettrapenable(3M)
exception trap enables: setting	fesettrapenable(3M)
exceptions: clearing floating-point	feclearexcept(3M)
exceptions: raising floating-point	feraiseexcept(3M)
exceptions: testing floating-point	fetestexcept(3M)
exchange, portable archive	pax(1)
exec - execute a file	exec(2)
exec - execute command without creating new process	csh(1)
exec - execute command without creating new process	ksh(1)
exec - execute command without creating new process	sh-posix(1)
execl() - execute a file	exec(2)
execle() - execute a file	exec(2)

Description	Entry Name(Section)
exec1p() - execute a file	exec(2)
executable and linking format object files (ELF)	elf(3E)
executable by default, controls whether program stacks are	executable_stack(5)
executable files or shared libraries on Integrity systems; list dynamic dependencies of	ldd_ia(1)
executable files or shared libraries on PA-RISC systems; list dynamic dependencies of	ldd_pa(1)
executable files or shared libraries; list dynamic dependencies of	ldd(1)
executable, prepare for faster program start-up	fastbind(1)
executable_stack - controls whether program stacks are executable by default	executable_stack(5)
execute a command; measure time used to	time(1)
execute a file	exec(2)
execute a function, descending a directory tree	ftw(3C)
execute a simple command	command(1)
execute command on a remote host	on(1)
execute command; construct argument lists and	xargs(1)
execute commands at a later time	at(1)
execute commands in background	at(1)
(- execute commands in separate shell	sh-posix(1)
execute from a remote shell	remsh(1)
execute from a remote shell	reexec(1)
execute HALGOL programs	opx25(1M)
execute link() and unlink() system calls without error checking	link(1M)
execute process with POSIX real-time priority	rtsched(1)
execute process with real-time priority	rtprio(1)
execute remote uucp or uux command requests on local system	uuxqt(1M)
execution daemon; timed-job	cron(1M)
execution of a thread; continue, resume, or suspend	pthread_resume_np(3T)
execution of commands, UNIX system to UNIX system	uux(1)
execution profile data, display call graph	gprof(1)
execution profile, prepare	smonitor(3C)
execution profile, prepare	monitor(3C)
execution server, RPC-based remote	rexd(1M)
execution server; remote	rexec(1M)
execution startup routines	crt0(3)
execution startup routines for Integrity systems	crt0_ia(3)
execution startup routines for PA-RISC systems	crt0_pa(3)
execution time profile	profil(2)
execution time profile for disjointed text spaces	sprofil(2)
execution, commands, set environment for	env(1)
execution, suspend for a time interval	sleep(1)
execution, suspend for an interval	usleep(2)
execv() - execute a file	exec(2)
execve() - execute a file	exec(2)
execvp() - execute a file	exec(2)
existing partition; modify an	parmodify(1M)
existing partition; remove an	parremove(1M)
exit - exit shell with exit status	csh(1)
exit - exit shell with exit status	ksh(1)
exit - exit shell with exit status	sh-posix(1)
exit status, do nothing and return zero or non-zero	true(1)
exit() - terminate a process	exit(2)
exit, register a function to be called at	atexit(3)
exp() - exponential function	exp(3M)
exp10() - base-10 exponential function	exp10(3M)
exp10f() - base-10 exponential function (float)	exp10(3M)
exp10l() - base-10 exponential function (long double)	exp10(3M)
exp10q() - base-10 exponential function (quad)	exp10(3M)
exp10w() - base-10 exponential function (extended)	exp10(3M)
exp2() - base-2 exponential function	exp2(3M)
exp2f() - base-2 exponential function (float)	exp2(3M)
exp2l() - base-2 exponential function (long double)	exp2(3M)
exp2q() - base-2 exponential function (quad)	exp2(3M)

Index

All Volumes

Description	Entry Name(Section)
exp2w() - base-2 exponential function (extended)	exp2(3M)
expand, unexpand - expand tabs to spaces, and vice versa	expand(1)
expand files using Huffman code; compress and	pack(1)
expand or compress data	compress(1)
expand_alias - recursively expands the sendmail aliases	expand_alias(1)
expanded_node_host_names - enable maximum length expansion of the system node and host names	expanded_node_host_names(5)
expands the sendmail aliases, recursively	expand_alias(1)
expansions, perform word	wordexp(3C)
expedited data over a connection (X/OPEN TLI-XTI)	t_snd(3)
expf() - exponential function (float)	exp(3M)
expl() - exponential function (long double)	exp(3M)
explicit load address; open an HP 9000 64-bit shared library with	dlopen_pa(3C)
explicit load of shared libraries	shl_load(3X)
explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
explicit locking of streams within a multi-thread application	flockfile(3S)
expm1() - exponential minus 1 function	expm1(3M)
expm1f() - exponential minus 1 function (float)	expm1(3M)
expm1l() - exponential minus 1 function (long double)	expm1(3M)
expm1q() - exponential minus 1 function (quad)	expm1(3M)
expm1w() - exponential minus 1 function (extended)	expm1(3M)
exponent and mantissa, split floating-point number into	frexp(3M)
exponent functions, radix-independent	ilogb(3M)
exponent functions, radix-independent	logb(3M)
exponent of a radix-independent floating-point number; scale	scalbln(3M)
exponential functions	exp(3M)
exponential minus 1 functions	expm1(3M)
export an LVM volume group and its associated logical volumes	vgexport(1M)
export - export variable names to environment of subsequent commands	ksh(1)
export - export variable names to environment of subsequent commands	sh-posix(1)
export to NFS clients, directories to	exports(4)
exportfs options to share/unshare commands; translates	exportfs(1M)
exportfs - translates exportfs options to share/unshare commands	exportfs(1M)
exports - directories to export to NFS clients	exports(4)
expq() - exponential function (quad)	exp(3M)
expr - evaluate arguments as an expression	expr(1)
Express Library; Unwind	uwX(3X)
expression matching routines, regular	regcomp(3C)
expression or string, search a file for a	grep(1)
expression, regular, and pattern matching notation definitions	regexp(5)
expression, regular, compile and match routines	regexp(3X)
expression; evaluate arguments as an	expr(1)
expw() - exponential function (extended)	exp(3M)
ext_unregister() - library routines for registering servers	rpc_svc_reg(3N)
extend a file system size (generic)	extendfs(1M)
extend an LVM volume group by adding physical volumes	vgextend(1M)
extend HFS file system size	extendfs_hfs(1M)
extended authentication, account, password, and session service module for HP-UX	pam_hpsec(5)
extended general terminal interface	termiox(7)
extended line-oriented text editor	ex(1)
extendfs - extend a file system size (generic)	extendfs(1M)
extendfs_hfs - extend HFS file system size	extendfs_hfs(1M)
Extensible Firmware Interface description	efi(4)
external options to specify the scheduling contention scope of threads, list of	pthread_scope_options(5)
extract error messages from C source into a file	mkstr(1)
extract mantissa and exponent from floating-point number	frexp(3M)
extract non-repeated lines from a file	uniq(1)
extract portions of path names	basename(1)
extract selected fields of each line in a file	cut(1)

Description	Entry Name(Section)
extract strings from C programs to implement shared strings	xstr(1)
extract window IDs of user processes from <code>/etc/services.window</code>	getmemwindow(1M)
extracts, writes, and lists archive files; copies files and directory hierarchies	pax(1)
fabs() - absolute value function	fabs(3M)
fabsf() - absolute value function (float)	fabs(3M)
fabsl() - absolute value function (long double)	fabs(3M)
fabsq() - absolute value function (quad)	fabs(3M)
fabsw() - absolute value function (extended)	fabs(3M)
facilities, 4.2 BSD-compatible process control	killpg(2)
facilities, interprocess communication, report status	ipcs(1)
facility for internet services, access control	tcpd(1M)
facility for multithreaded processes; tracing	ttrace(2)
factor, primes - factor a number, generate large primes	factor(1)
factor a number, generate large primes	factor(1)
fadvise() - file advisory information	fadvise(2)
fadvise() function; structures needed when using the	fadvise(5)
fadvise - structures needed when using the fadvise() function	fadvise(5)
fadvise.h - structures needed when using the fadvise() function	fadvise(5)
failure, get error codes on	rpc_gss_get_error(3N)
fallback mechanism	service.switch(1M)
false - do nothing and return non-zero exit status	true(1)
false/true evaluate condition for	test(1)
family, Internet protocol	inet(7F)
fast disk storage, preallocate	prealloc(2)
fast symbolic links	create_fastlinks(5)
fastbind - prepare an incomplete executable for faster program start-up	fastbind(1)
faster program start-up	fastbind(1)
faster tape I/O	ftio(1)
faster viewing with man command, fix manpages for	fixman(1M)
fastmail - quick batch mail interface	fastmail(1)
fattach() - attach a STREAMS file descriptor	fattach(3C)
fault, generate an IOT	abort(3C)
fbackup - selectively back up files	fbackup(1M)
fc - edit and execute previous command	ksh(1)
fc - edit and execute previous command	sh-posix(1)
FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based	fcmsutil(1M)
fcache_fb_policy - policy for flush behind requests from VxFS file system	fcache_fb_policy(5)
fcache_seqlimit_file - percent of file cache that can be consumed by sequential accesses, per-file limit	fcache_seqlimit_file(5)
fcache_seqlimit_system - percentage of file cache that can be consumed by sequential accesses, per system-wide limit	fcache_seqlimit_system(5)
FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2	fcmsutil(1M)
fchdir() - change working directory	chdir(2)
fchmod() - change file mode access permissions	chmod(2)
fchown() - change owner and group of a file	chown(2)
fclose() - close or flush a stream	fclose(3S)
fclose_unlocked() - close or flush a stream	fclose(3S)
fcmsutil - fibre channel diagnostic utility	fcmsutil(1M)
fcmsutil - Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters	fcmsutil(1M)
fcntl() - file control	fcntl(2)
fcntl - file control options	fcntl(5)
fcpacl() - copy access control list (ACL) to another file	cpacl(3C)
fcvt() - convert floating-point number to string	ecvt(3C)
FD_CLR() - synchronous I/O multiplexing	select(2)
FD_ISSET() - synchronous I/O multiplexing	select(2)
FD_SET() - synchronous I/O multiplexing	select(2)
FD_ZERO() - synchronous I/O multiplexing	select(2)
fdetach - detach a name from a STREAMS file descriptor	fdetach(3C)

Index

All Volumes

Description	Entry Name(Section)
fdetach - detach a STREAMS-based file descriptor	fdetach(1M)
fdim() - positive difference function	fdim(3M)
fdimf() - positive difference function (float)	fdim(3M)
fdiml() - positive difference function (long double)	fdim(3M)
fdimq() - positive difference function (quad)	fdim(3M)
fdimw() - positive difference function (extended)	fdim(3M)
fdopen() - associate a stream with an open file descriptor	fopen(3S)
feclearexcept() - clear floating-point exceptions	feclearexcept(3M)
fegetenv() - get floating-point environment	fegetenv(3M)
fegetexceptflag() - get floating-point exception flags	fegetexceptflag(3M)
fegetflushtozero() - get floating-point underflow mode	fegetflushtozero(3M)
fegetround() - get floating-point rounding mode	fegetround(3M)
fegettrapenable() - get exception trap enables	fegettrapenable(3M)
feholdexcept() - save floating-point environment	feholdexcept(3M)
feof() - check for end-of-file error on stream	feof(3S)
feof_unlocked() - stream status inquiries	feof(3S)
feraiseexcept() - raise floating-point exceptions	feraiseexcept(3M)
ferror() - check for I/O error on stream	ferror(3S)
ferror_unlocked() - stream status inquiries	ferror(3S)
fesetenv() - set floating-point environment	fesetenv(3M)
fesetexceptflag() - set floating-point exception flags	fesetexceptflag(3M)
fesetflushtozero() - set floating-point underflow mode	fesetflushtozero(3M)
fesetround() - set floating-point rounding mode	fesetround(3M)
fesettrapenable() - set exception trap enables	fesettrapenable(3M)
fetch() - access data under a key (old single-data-base version)	dbm(3X)
fetestexcept() - test floating-point exceptions	fetestexcept(3M)
feupdateenv() - update floating-point environment	feupdateenv(3M)
ff - list file names and statistics for HFS file system	ff_hfs(1M)
ff - list file names and statistics for file system	ff(1M)
fflush() - close or flush a stream	fclose(3S)
fflush_unlocked() - close or flush a stream	fclose(3S)
ffs() - BSD find first set bit	memory(3C)
fg - put jobs into foreground	sh-posix(1)
fgetacl() - get access control list (ACL) information	getacl(2)
fgetc() - get character or word from a stream file	getc(3S)
fgetgrent() - return pointer to next group	getgrent(3C)
fgetpos() - save file position indicator for a stream	fgetpos(3S)
fgetpos64() - file system API to support large files	fgetpos64(3S)
fgetpwent() - get next entry in password-file-formatted input stream	getpwent(3C)
gets() - get a string from a stream	gets(3S)
getspent() - get next secure password file entry	getspent(3C)
getspent() - get next entry in secure password-file-formatted input stream on trusted systems	getspent(3X)
getspent_r() - get secure password file entry on trusted systems	getspent(3X)
getwc() - get wide character from a stream file	getwc(3C)
fgetws() - get a wide-character string from a stream file	fgetws(3C)
fgetws_unlocked() - get a wide-character string from a stream file	fgetws(3C)
fgrep - search a file for a specific string (fast algorithm)	grep(1)
fibres channel diagnostic utility	fcmsutil(1M)
Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo	fcmsutil(1M)
Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters	fcmsutil(1M)
fields of each line in a file, cut out (extract) selected	cut(1)
FIFO file; make a	mkfifo(3C)
FIFO files, create	mknod(1M)
FIFO (named pipe) special files; make	mkfifo(1)
FIFO scheduling policy	rtsched(2)
FIFO special file	glossary(9)
file	glossary(9)
file access and modification times, set	utimes(2)

Description	Entry Name(Section)
file access and modification times, set or update	utime(2)
file access mode	glossary(9)
file access permissions	glossary(9)
File Access software, Data Communications and Terminal Controller Device	ddfa(7)
file advisory information	fadvise(2)
file advisory information	posix_fadvise(2)
file and directory comparator; differential	diff(1)
file archiver; tape	tar(1)
file archives in and out; duplicate directory trees; copy	cpio(1)
file archives; copy in and out	cpio(1)
file cache that can be consumed by sequential accesses, per system-wide limit; percentage of	fcache_seqlimit_system(5)
file cache that can be consumed by sequential accesses, per-file limit; percent of	fcache_seqlimit_file(5)
file checkers; password/group	pwck(1M)
file containing applications and their associated memory window ID	services.window(4)
file containing commands for sharing resources across a network	dfstab(4)
file containing parameter values for automountd daemon and automount command	autofs(4)
file control options for open files	fcntl(5)
file control; open-file control routines	fcntl(2)
file copy, public UNIX system to UNIX system	uuto(1)
file copy, remote	rcp(1)
file creation (permissions) mask, set and get	umask(2)
file creation, set access permissions mode mask for	umask(1)
file descriptor	glossary(9)
file descriptor for ELF file; make	elf_begin(3E)
file descriptor table; get the size of the per-process	getdtablesize(2)
file descriptor to a specific slot; duplicate an open	dup2(2)
file descriptor, map stream pointer to	fileno(3S)
file descriptor: attach a STREAMS file descriptor	fattach(3C)
file descriptor: detach a name from a STREAMS-based file descriptor	fdetach(3C)
file descriptor: detach a STREAMS-based file descriptor	fdetach(1M)
file descriptor: duplicate an open file descriptor	dup(2)
file descriptor: STREAMS device	isastream(3C)
file descriptor: STREAMS-based pipe	isastream(3C)
file descriptors per process, initial (soft) maximum number of	maxfiles(5)
file descriptors, hard maximum number of	maxfiles_lim(5)
file descriptors, monitor I/O conditions on multiple	poll(7)
file descriptors; displays process address information and open	pmap(1)
file descriptors; monitor I/O conditions on multiple	poll(2)
file - determine file type	file(1)
file distribution program; remote	rdist(1)
file editing activity, print current SCCS	sact(1)
file entry on trusted systems; get secure password	getspwent(3X)
file entry, get file system descriptor (BSD 4.2 compatibility only)	getfsent(3X)
file entry, write shadow password	putspent(3C)
file entry; access utmp	getut(3C)
file entry; get group	getgrent(3C)
file entry; write password	putpwent(3C)
file for a trusted system; device assignment database	devassign(4)
file for a trusted system; system default database	default(4)
file for at ; prototype job	proto(4)
file for ftpd ; security	ftusers(4)
file for Internet domain name server; configuration	named.conf(4)
file for router advertisement daemon; configuration	rtradvd.conf(4)
file for SLP agents; configuration	slp.conf(4)
file for the Access Control Policy Switch (ACPS); configuration	acps.conf(4)
file for trusted systems; terminal control database	ttys(4)
file format and other information for auditing	audit(4)
file format between HP-UX and DOS formats; convert ASCII	dos2ux(1)
file format for keysh softkeys	softkeys(4)

Index

All Volumes

Description	Entry Name(Section)
file format, <pwd.h> password file	passwd(4)
file format: common archive files	ar(4)
file format: compiled terminfo file format	term(4)
file format: disk description file format	disktab(4)
file format: introduction to file formats	intro(4)
file format: per-process accounting files	acct(4)
file format: text file format specification	fspec(4)
file format; core image	core(4)
file format; PPP authentication	ppp.Auth(4)
file format; PPP dialer description	ppp.Dialers(4)
file format; PPP encryption keys	ppp.Keys(4)
file format; PPP neighboring systems description	ppp.Systems(4)
file format; PPP packet filter specification	ppp.Filter(4)
file format; PPP physical device description	ppp.Devices(4)
file format; shadow password	shadow(4)
file format; translate host table to name server	hosts_to_named(1M)
file group class	glossary(9)
file header for elf32 or elf64 file; retrieve class-dependent object	elf_getehdr(3E)
file hierarchy	glossary(9)
file I/O data; maximum or minimum amount of physical memory used for caching	filecache_max(5)
file identification data for ELF files, retrieve	elf_getident(3E)
file including aliases and paths; locate a program	which(1)
file information; change WU-FTPD group access	privatepw(1)
file into memory; copy	copylist(3G)
file link: soft (symbolic) link	symlink(4)
file locking, provide semaphores and record locking on files	lockf(2)
file locks, maximum number of	nflocks(5)
file maintenance utility; Kerberos keytab	ktutil(1)
file management	kermit(1)
file name	glossary(9)
file name generation function	glob(3C)
file name of controlling terminal, generate	ctermid(3S)
file name portability	glossary(9)
file name suffix conventions	suffix(5)
file names and statistics for HFS file system, list	ff_hfs(1M)
file names for file system, list	ff(1M)
file names, convert an HFS file system to allow long	convertfs(1M)
file of a process, get information for an open	pstat(2)
file of uucp transactions; query log	uucp(1)
file offset	glossary(9)
file on remote node, get file handle for	getfh(2)
file or anonymous memory region, initialize semaphore in mapped	msem_init(2)
file or anonymous region, remove semaphore in mapped	msem_remove(2)
file other class	glossary(9)
file owner class	glossary(9)
file path; map device ID to	devnm(3)
file permission bits	glossary(9)
file perusal filter for screen viewing	more(1)
file perusal filter for soft-copy terminals	pg(1)
file pointer	glossary(9)
file pointer: move read/write file pointer	lseek(2)
file position indicator for a stream; save or restore	fgetpos(3S)
file (PSF) format; product specification	swpackage(4)
file serial number	glossary(9)
file size in words, lines, and bytes or characters	wc(1)
file size limits and break value, get or set	ulimit(2)
file status flags	glossary(9)
file status, get	fstat(2)
file structure, list processes using	fuser(1M)
file structures, statd directory and	sm(4)
file syntax checking tool; named configuration	named-checkconf(1)

Description	Entry Name(Section)
file system	glossary(9)
file system administration command	fsadm(1M)
file system administration command	fsadm_hfs(1M)
file system administration commands, configuration and binary files	fs_wrapper(5)
file system APIs to support large files	fgetpos64(3S)
file system consistency check and interactive repair	fsck(1M)
file system control	fsctl(2)
file system debugger (generic)	fsdb(1M)
file system debugger (HFS)	fsdb_hfs(1M)
file system descriptor file entry; get	getmntent(3X)
file system disk blocks, report number of free CDFS, HFS, or NFS	df_hfs(1M)
file system disk blocks; report number of free	df(1M)
file system for debugging purposes., a feature that saves operating system state to the	livedump(5)
file system for paging; enable device or	swapon(1M)
file system (generic), construct a	mkfs(1M)
file system header on a device file, write an EFI	efi_fsinit(1M)
file system incrementally, local or across network; restore	restore(1M)
file system information, dump	dumpfs(1M)
file system information; get	statvfs(2)
file system information; get	statvfsdev(3C)
file system name space: attach a STREAMS file descriptor	fattach(3C)
file system quota consistency checker	quotacheck(1M)
file system quotas; summarize	repquota(1M)
file system size (generic), extend a	extendfs(1M)
file system size, extend HFS	extendfs_hfs(1M)
file system stack templates	fstadm(2)
file system statistics, get	statfs(2)
file system statistics; cache	cachefsstat(1M)
file system statistics; get	statsfsdev(3C)
file system table, shared	sharetab(4)
file system type info, get	sysfs(2)
file system type; determine	fstyp(1M)
file system with compaction; copy HFS	dcopy(1M)
file system with label checking; copy HFS	volcopy_hfs(1M)
file system with label checking; copy a	volcopy(1M)
file system, construct an HFS	mkfs_hfs(1M)
file system, damaged, patch up (generic)	fsdb(1M)
file system, damaged, patch up (HFS)	fsdb_hfs(1M)
file system, get mounted file system statistics	ustat(2)
file system, incremental dump (for backups)	dump(1M)
file system, list file names and statistics	ff(1M)
file system, list file names and statistics for	ff(1M)
file system, list file names and statistics for HFS	ff_hfs(1M)
file system, list processes using	fuser(1M)
file system, mount an LOFS	mount_lofs(1M)
file system, restore incrementally, local or across network	restore(1M)
file system: backup or archive the file system	backup(1M)
file system: file system hierarchy	hier(5)
file system: get file system descriptor file entry (BSD 4.2 compatibility only)	getfsent(3X)
file system: mounted file system table	mnttab(4)
file system: optimize an existing HFS file system	tunefs(1M)
file system: tune an existing HFS file system	tunefs(1M)
file system; construct a new	newfs(1M)
file system; construct a new HFS	newfs_hfs(1M)
file system; mount	mount(2)
file system; mount	vfsmount(2)
file system; unmount	umount(2)
file systems that can be enabled for swap; maximum number of	nswapfs(5)
File Systems tool of HP System Management Homepage (HP SMH); launch the Disks and	fsweb(1M)
file systems unavailable for mounting by remote systems; make local NFS	unshare_nfs(1M)
file systems, keep track of remotely mounted	mount(3N)

Index

All Volumes

Description	Entry Name(Section)
file systems, mount and unmount multiple	mountall(1M)
file systems; determine the shutdown status of HFS	fsclean(1M)
file systems; mount and unmount	mount(1M)
file systems; mount and unmount CacheFS	mount_cacheofs(1M)
file systems; mount and unmount CDFS	mount_cdfs(1M)
file systems; mount and unmount HFS	mount_hfs(1M)
file systems; mount and unmount NFS	mount_nfs(1M)
file systems; static information about	fstab(4)
file that registers distributed file system packages	fstypes(4)
File through a socket; send the contents of a Large	sendfile64(2)
file through a socket; send the contents of a	sendfile(2)
file times update	glossary(9)
file transfer	kermit(1)
file transfer program	ftp(1)
file transfer program, trivial	tftp(1)
file transfer program, XMODEM-protocol	umodem(1)
file transfer protocol server	ftpd(1M)
file transfer protocol server; trivial	tftpd(1M)
file tree, walk, executing a function	ftw(3C)
file type for ELF files, determine	elf_kind(3E)
file type; determine	file(1)
file used by DDFA software and Telnet port identification feature; dedicated ports	dp(4)
file's Access Control List (ACL) information; JFS File Systems only; set a	acl(2)
file's in-core state with its state on disk, synchronize a	fsync(2)
file, change owner or group	chown(1)
file, change system configuration	ch_rc(1M)
file, configuration, for pluggable authentication module	pam.conf(4)
file, copy to or from EFI	efi_cp(1M)
file, create a kernel system	create_sysfile(1M)
file, create message catalog file for modification	findmsg(1)
file, create special and FIFO	mknod(1M)
file, determine accessibility	access(2)
file, directory, remove	rmdir(2)
file, dump window to and reload window from	getwin(3X)
file, for screen, input/output functions	scr_dump(3X)
file, generate a formatted message catalog	gencat(1)
file, get a user's effective access rights to a	getaccess(2)
file, get detailed information for an open	pstat(2)
file, get file status	stat(2)
file, get the full path name of an open	pstat(2)
file, group password	ftpgroups(4)
file, inetd configuration	inetd.conf(4)
file, inetd optional security	inetd.sec(4)
file, link existing file to a new file name	ln(1)
file, list processes using	fuser(1M)
file, LVM physical volume group information	lvmpvg(4)
file, make a symbolic link	symlink(2)
file, NLSPATH configuration	nlspath(4)
file, open for reading or writing	open(2)
file, print out mail in the incoming mailbox	prmail(1)
file, read data from	read(2)
file, remove an EFI	efi_rm(1M)
file, send the contents of a file through a socket	sendfile64(2)
file, stream: reposition or get pointer for I/O operations on a stream file	fseek(3S)
file, synchronize a mapped	msync(2)
file, tcpd configuration	tcpd.conf(4)
file, used by DDFA software, configuration	pcf(4)
file, utmpx (), of the current user, find the slot in the	ttyslot(3C)
file, validate an SCCS file	val(1)
file, write an EFI file system header on a device	efi_fsinit(1M)
file, write data	write(2)

Description	Entry Name(Section)
file: change owner and group	chown(2)
file: change the name of a file	rename(2)
file: close a file descriptor	close(2)
file: control a file descriptor for ELF files	elf_cntl(3E)
file: convert binary file to ASCII for transmission by mailer	uuencode(1)
file: copy access control list (ACL) to another file	cpacl(3C)
file: create a name for a temporary file	tmpnam(3S)
file: create a temporary file	tmpfile(3S)
file: create zero-length file	cp(1)
file: create zero-length file	cat(1)
file: create zero-length file	null(7)
file: decode a file encoded by uuencode	uuencode(1)
file: delete	unlink(2)
file: discard file (bit bucket)	null(7)
file: get the base offset for an object file	elf_getbase(3E)
file: header file for future applications	portal(5)
file: issue (/etc/issue) identification file format	issue(4)
file: null file (bit bucket)	null(7)
file: object file access library	elf(3E)
file: remove a file	remove(3C)
file: remove nroff/troff, tbl, and neqn constructs from	deroff(1)
file: return the size of an object file type for elf32 or elf64 files	elf_fsize(3E)
file: truncate a file to a specified length	truncate(2)
file: uncompress a file in a crash dump	cr_uncompress(3)
file: user accounting information file	utmpx(4)
file; /dev/zero special	zero(7)
file; apply or remove an advisory or enforced lock on an open	flock(2)
file; assign buffering to a stream	setbuf(3S)
file; change or reformat a text	newform(1)
file; compare two versions of an SCCS	sccsdiff(1)
file; convert an audio	convert(1)
file; count words, lines, and bytes or characters in a	wc(1)
file; describe an audio	attributes(1)
file; Event Manager filter	evmfilterfile(4)
file; Event Manager template	evmtemplate(4)
file; EVM authorization	evm.auth(4)
file; EVM channel configuration	evmchannel.conf(4)
file; EVM daemon configuration	evmdaemon.conf(4)
file; EVM logger configuration	evmlogger.conf(4)
file; execute	exec(2)
file; extract error messages from C source into a	mkstr(1)
file; format of an encoded uuencode	uuencode(4)
file; format of SCCS	scsfile(4)
file; install, update or check the /etc/shadow	pwconv(1M)
file; Kerberos configuration	krb5.conf(4)
file; main memory image	mem(7)
file; make a FIFO	mkfifo(3C)
file; make a special (device)	mksf(1M)
file; password	passwd(4)
file; password: get entry from secure password file	getspwent(3C)
file; perform I/O of EVM events to and from a	EvmEventRead(3)
file; pipe fitting to copy standard output to	tee(1)
file; play an audio	send_sound(1)
file; preprocess a message source	mkcatdefs(1)
file; print and summarize an SCCS	prs(1)
file; program to apply a diff file to an original	patch(1)
file; receive next message from a STREAMS file	getmsg(2)
file; remove a delta from an SCCS	rmdel(1)
file; retrieve class-dependent object file header for elf32 or elf64	elf_getehdr(3E)
file; rndc configuration	rndc.conf(4)
file; security defaults configuration	security(4)

Index

All Volumes

Description	Entry Name(Section)
file; shadow password	shadow(4)
file; update access, modification, and/or change times of	touch(1)
filecache_max - maximum amount of physical memory for caching file I/O data	filecache_max(5)
filecache_min - minimum amount of physical memory for caching file I/O data	filecache_max(5)
filename patterns, match	fnmatch(3C)
filename: detach a STREAMS-based file descriptor	fdetach(1M)
fileno() - map stream pointer to file descriptor	fileno(3S)
files and directory hierarchies; extracts, writes, and lists archive files; copies	pax(1)
files and directory subtrees; copy	cp(1)
files and file systems in the cache; pack	cachefspack(1M)
files for online manpages; create cat and whatis	catman(1M)
files for the uucp system; transfer	uucico(1M)
files for trusted systems; protected password authentication database	prpwd(4)
files in binary directories; install object	cpset(1M)
files in RCS; identify	ident(1)
files or directories; remove	rm(1)
files or shared libraries on Integrity systems; list dynamic dependencies of executable	ldd_ia(1)
files or shared libraries on PA-RISC systems; list dynamic dependencies of executable	ldd_pa(1)
files or shared libraries; list dynamic dependencies of executable	ldd(1)
files that are under access control; let authorized users edit	privedit(1M)
files using Huffman code; compress and expand	pack(1)
files with multiple hardlinks; checks the consistency of compartment rules for	vhardlinks(1M)
files, accounting: convert process accounting files to ASCII text format	acctprc(1M)
files, accounting: merge or add total accounting files	acctmerg(1M)
files, accounting: summarize process accounting files created by acctprc1	acctprc(1M)
files, C header, generate	rpcgen(1)
files, change name of a file	mv(1)
files, configuration and binary, file system administration	fs_wrapper(5)
files, Cyclical Redundancy Check on a file	sum(1)
files, find the printable strings in an object or other binary file	strings(1)
files, format and print	pr(1)
files, move directory subtree and files to another directory	mv(1)
files, move file to new location	mv(1)
files, move multiple files to another directory	mv(1)
files, non-POSIX standard API interfaces to support large	creat64(2)
files, object code: find ordering relation for files in an object code library	lorder(1)
files, object code: optimum sequence for object code files in a library, find	lorder(1)
files, overwrite file with an existing file	mv(1)
files, print checksum and block count of a file	sum(1)
files, remove outdated STREAMS error log files	strclean(1M)
files, rename directory	mv(1)
files, rename file	mv(1)
files, search a file for a string or expression	grep(1)
files, send to system log	logger(1)
files, text: format text file for CRT or line-printer output	nroff(1)
files: break a single file into multiple files	split(1)
files: check nroff/troff files	checknr(1)
files: compare two files	cmp(1)
files: compare two files and mark differences	diffmk(1)
files: compare two files and show differences side-by-side	sdiff(1)
files: compress data in a file	compress(1)
files: compress files in a directory	compress(1)
files: context split	csplit(1)
files: convert file keyboard/display data order	forder(1)
files: copy directory subtree and files to another directory	cp(1)
files: copy file to a new or existing file	cp(1)
files: copy multiple files to a directory	cp(1)
files: copy to or from remote system	rcp(1)
files: count words, lines, and bytes or characters in a file	wc(1)
files: create a tags file	ctags(1)
files: cut out (extract) selected fields of each line in a file	cut(1)

Description	Entry Name(Section)
files: differential file comparator	diff(1)
files: display file on soft-copy terminals	pg(1)
files: eliminate adjacent repeated lines in a file	uniq(1)
files: expand compressed file	compress(1)
files: find differences among three files	diff3(1)
files: find differences between two files	diff(1)
files: find (search for) files	find(1)
files: format tracing and logging binary files	netfmt(1M)
files: get first few lines in a file	head(1)
files: get lines from last part of a file	tail(1)
files: get status	fstat(2)
files: list access control lists (ACLs) of files	lsacl(1)
files: list access rights to file(s)	getaccess(1)
files: make a delta (change) to an SCCS file	delta(1)
files: make unprintable and non-ASCII characters in a file visible or invisible	vis(1)
files: merge corresponding lines of several files or subsequent lines of one file	paste(1)
files: name for a temporary file, make a	mktemp(1)
files: overwrite file with an existing file	cp(1)
files: password file, edit using vi editor	vipw(1M)
files: print checksum and block count of a file	sum(1)
files: print first few lines in a file	head(1)
files: print section sizes and allocation space of object files	size(1)
files: queue description file for at , batch , and crontab	queuedefs(4)
files: read portable archive	pax(1)
files: reduce multiple adjacent blank lines to single blank line	ssp(1)
files: remove all blank lines from file	rmnl(1)
files: remove file that is not listed in any directory	clri(1M)
files: report adjacent repeated lines in a file	uniq(1)
files: reverse the left-to-right text character sequence in each line of a file	rev(1)
files: schedule uucp transport files	uusched(1M)
files: select/reject lines common to two sorted files	comm(1)
files: split a file into multiple <i>n</i> -line pieces	split(1)
files: split file into multiple files	csplit(1)
files: strip symbol and line number information from an object file	strip(1)
files: temporary file, make a name for a	mktemp(1)
files: three-way differential file comparator	diff3(1)
files: three-way file merge	merge(1)
files: undo a previous get of an SCCS file	unget(1)
files: write portable archive	pax(1)
files; copies files and directory hierarchies; extracts, writes, and lists archive	pax(1)
files; copy to or from LIF	lifcp(1)
files; create and administer SCCS	admin(1)
files; determine file type	file(1)
files; dump file in octal or hexadecimal format	od(1)
files; format of RCS	resfile(4)
files; get section information for ELF	elf_getscn(3E)
files; HP-UX compartments	compartments(4)
files; introduction to device special	intro(7)
files; make FIFO (named pipe) special	mkfifo(1)
files; print log messages and other information on RCS	rlog(1)
files; reload Event Manager configuration	evmreload(1M)
files; search and print process accounting	acctcom(1M)
files; sort or merge	sort(1)
files; special (device)	see <i>special files</i>
files; start or halt the auditing system and set or get audit	audctl(2)
filesystem swap, determines when swapmap structures are allocated for	allocate_fs_swapmap(5)
filesystem table, remote mounted	rmtab(4)
filter	glossary(9)
filter callbacks for sendmail; registers a set of	smfi_register(3N)
filter() - disable use of certain terminal capabilities	filter(3X)
filter evaluator routines; event	EvmFilterCreate(3)

Index

All Volumes

Description	Entry Name(Section)
filter file; Event Manager	evmfilterfile(4)
filter for screen viewing; file perusal	more(1)
filter reverse line-feeds and backspaces from text	col(1)
filter specification file format; PPP packet	ppp.Filter(4)
filter, for sendmail; sets the listen backlog value of the	smfi_setbacklog(3N)
filter; attempts to create the interface socket that MTAs use to connect to the	smfi_opensocket(3N)
filter; EVM (Event Management) event	EvmFilter(5)
filter; internationalized PostScript print	psfontpf(1M)
filter; line numbering	nl(1)
filters invoked by lp interface scripts	lpfilter(1)
financial: compound interest factor	compound(3M)
financial: present value factor for annuity	annuity(3M)
find adjacent repeated lines in a file	uniq(1)
find differences among three files	diff3(1)
find differences between two files	diff(1)
find - find (search for) files	find(1)
find hyphenated words	hyphen(1)
find location of source, binary, and/or manual files for program	whereis(1)
find manpage information by keywords	man(1)
find name of a terminal	ttyname(3C)
find ordering relation for files in an object code library	lorder(1)
find spelling errors	spell(1)
find strings for inclusion in message catalogs	findstr(1)
find the printable strings in an object or other binary file	strings(1)
find the slot in the utmpx() file of the current user	ttyslot(3C)
findmsg - create message catalog file for modification	findmsg(1)
<i>findstr</i> (1) output, use to insert calls to <i>catgets</i> (3C)	insertmsg(1)
findstr - find strings for inclusion in message catalogs	findstr(1)
finger command; change user information used by	chfn(1)
finger - user information lookup program	finger(1)
fingerd - remote user information server	fingerd(1M)
finish using an ELF object file	elf_end(3E)
finish, wait for background processes to	wait(1)
finite-width output device, fold long lines for	fold(1)
finiteness macro, floating-point	isfinite(3M)
Firmware Interface description, Extensible	efi(4)
firmware (processor-dependent code)	pdc(1M)
first locations beyond allocated program regions	end(3C)
firstkey() - get first key in database (old single-data-base version)	dbm(3X)
fit in fields; causes uname() system function to return [Eoverflow] if values do not	uname_eoverflow(5)
fix damaged file system (generic)	fsdb(1M)
fix damaged HFS file system	fsdb_hfs(1M)
fix information in the user database, /var/adm/userdb, verify or	userdbck(1M)
fix manpages for faster viewing with man command	fixman(1M)
fixed-size integer data types	inttypes(5)
fixman - fix manpages for faster viewing with man command	fixman(1M)
flag for calling process; get audit process	getaudproc(2)
flash() - flash the screen	flash(3X)
flash the screen	flash(3X)
flash/turn off attention LEDs (cell, cabinet and I/O chassis attention LEDs)	fruled(1)
file system packages; file that registers distributed	fstypes(4)
floating multiply-add functions	fma(3M)
floating-point classification macros	fpclassify(3M)
floating-point classification macros	isfinite(3M)
floating-point classification macros	isinf(3M)
floating-point classification macros	isnan(3M)
floating-point classification macros	isnormal(3M)
floating-point comparison macro (unordered)	isunordered(3M)
floating-point environment macros and functions	fenv(5)
floating-point environment: getting	fegetenv(3M)

Description	Entry Name(Section)
floating-point environment: saving	fehldexcept(3M)
floating-point environment: setting	fesetenv(3M)
floating-point environment: updating	feupdateenv(3M)
floating-point exception flags: getting	fegetexceptflag(3M)
floating-point exception flags: setting	fesetexceptflag(3M)
floating-point exceptions: clearing	feclearexcept(3M)
floating-point exceptions: getting trap enables	fegettrapenable(3M)
floating-point exceptions: raising	feraiseexcept(3M)
floating-point exceptions: setting trap enables	fesettrapenable(3M)
floating-point exceptions: testing	fetestexcept(3M)
floating-point number to string, convert long double	ldcvt(3C)
floating-point number to string; convert	ecvt(3C)
floating-point number: decompose	modf(3M)
floating-point number; convert string to	strtod(3C)
floating-point number; scale exponent of a radix-independent	scalbln(3M)
floating-point quiet comparison macro (<)	isless(3M)
floating-point quiet comparison macro (<=)	islessequal(3M)
floating-point quiet comparison macro (<>)	islessgreater(3M)
floating-point quiet comparison macro (>)	isgreater(3M)
floating-point quiet comparison macro (>=)	isgreaterequal(3M)
floating-point rounding mode: getting	fegetround(3M)
floating-point rounding mode: setting	fesetround(3M)
floating-point sign-determination	signbit(3M)
floating-point underflow mode: getting	fegetflushstozero(3M)
floating-point underflow mode: setting	fesetflushstozero(3M)
floating-point: extract mantissa and exponent from floating-point number	frexp(3M)
floating-point: scale exponent of a radix-independent floating-point number	scalb(3M)
flock() - apply or remove an advisory or enforced lock on an open file	flock(2)
flockfile() , funflockfile() - explicit locking of streams within a multi-thread application	flockfile(3S)
floor() - floor function	floor(3M)
floor functions	floor(3M)
floorf() - floor function (float)	floor(3M)
floorl() - floor function (long double)	floor(3M)
floorq() - floor function (quad)	floor(3M)
floorw() - floor function (extended)	floor(3M)
flush a stream, close or	fclose(3S)
flush behind requests from VxFS file system; policy for	fcache_fb_policy(5)
flush kernel registry services data to disk	krs_flush(1M)
flush unwritten system buffers to disk	sync(1M)
flush, enable or disable on interrupt	intrflush(3X)
flushing audit records; determine time interval (in secs) for	diskaudit_flush_interval(5)
flushing queue, enable/disable	noqiflush(3X)
flushinp() - discard input	flushinp(3X)
fma() - floating multiply-add function	fma(3M)
fmaf() - floating multiply-add function (float)	fma(3M)
fmal() - floating multiply-add function (long double)	fma(3M)
fmaq() - floating multiply-add function (quad)	fma(3M)
fmax() - floating multiply-add function (extended)	fma(3M)
fmax() - maximum value function	fmax(3M)
fmaxf() - maximum value function (float)	fmax(3M)
fmaxl() - maximum value function (long double)	fmax(3M)
fmaxq() - maximum value function (quad)	fmax(3M)
fmaxw() - maximum value function (extended)	fmax(3M)
fmin() - minimum value function	fmin(3M)
fminf() - minimum value function (float)	fmin(3M)
fminl() - minimum value function (long double)	fmin(3M)
fminq() - minimum value function (quad)	fmin(3M)
fminw() - minimum value function (extended)	fmin(3M)
fmod() - remainder function	fmod(3M)
fmodf() - remainder function (float)	fmod(3M)
fmodl() - remainder function (long double)	fmod(3M)

Index

All Volumes

Description	Entry Name(Section)
fmodq() - remainder function (quad)	fmod(3M)
fmodw() - remainder function (extended)	fmod(3M)
fmt - format text	fmt(1)
fmtmsg() - displays formatted message on standard error and console	fmtmsg(3C)
fnmatch() - match filename patterns	fnmatch(3C)
fold - fold long lines for finite-width output device	fold(1)
folders by subject and sender; summarize mail	mailfrom(1)
fontdl - download fonts to printer	lpfilter(1)
footprint records; summarize information from compiler	footprints(1)
footprints - summarize information from compiler footprint records	footprints(1)
fopen() - open a named file and associate with a stream	fopen(3S)
fopen64() - file system API to support large files	fgetpos64(3S)
for a session, change service, QOP	rpc_gss_set_defaults(3N)
for - execute a do list	ksh(1)
for - execute a do list	sh-posix(1)
for which the stubs are provided in the C library, list of pthread calls	pthread_stubs(5)
force all pipes to be STREAMS-base	streampipes(5)
force process to relinquish processor	rtsched(2)
force propagation of Network Information Service database	yppush(1M)
force target process to run serially with other processes	serialize(1)
force target process to run serially with other processes	serialize(2)
foreach - initiate repetitive loop	csh(1)
foreground process group	glossary(9)
foreground process group ID	glossary(9)
foreground process group ID, get	tcgetpgrp(3C)
foreground process group ID, set	tcsetpgrp(3C)
fork	glossary(9)
fork() - create a new process	fork(2)
fork handler	pthread_atfork(3T)
form; dump an event in displayable	EvmEventDump(3)
format and print arguments	printf(1)
format and print files	pr(1)
format between HP-UX and DOS formats; convert ASCII file	dos2ux(1)
format date and time; convert user	getdate(3C)
format events for display	EvmEventFormat(3)
format - format an HP SCSI disk array LUN	format(1M)
format mathematical text for nroff	neqn(1)
format of a CDFS cnode	cdnode(4)
format of an encoded uuencode file	uuencode(4)
format of core image file	core(4)
format of cpio archive	cpio(4)
format of host access control files	hosts_access(5)
format of privileged values	privgrp(4)
format of RCS files	rcsfile(4)
format of SCCS file	scsfile(4)
format specification in text files	fspec(4)
format text	fmt(1)
format text version of EVM status code	EvmStatusTextGet(3)
format tracing and logging binary files	netfmt(1M)
format, tar tape archive	tar(4)
format: common archive file	ar(4)
format: directories	dir(4)
format: format text file for CRT or line-printer output	nroff(1)
format: per-process accounting files	acct(4)
format; display message in standard	pfmt(3C)
format; DOS interchange	dosif(4)
format; ioconfig entry	ioconfig(4)
format; PPP encryption keys file	ppp.Keys(4)
format; PPP neighboring systems description file	ppp.Systems(4)
format; PPP packet filter specification file	ppp.Filter(4)
format; product specification file (PSF)	swpackage(4)

Description	Entry Name(Section)
format; translate host table to name server file	hosts_to_named(1M)
format; user login record	utmp(4)
formats for Integrity systems; structure	nlist_ia(4)
formats for PA-RISC systems; structure	nlist_pa(4)
formats; convert ASCII file format between HP-UX and DOS	dos2ux(1)
formats; structure	nlist(4)
formatted input conversion to a varargs argument	vscanf(3S)
formatted input conversion; read from stream file or character string	scanf(3S)
formatted input from a window; convert	vw_scanw(3X)
formatted input, convert from a window	vwscanw(3X)
formatted input, convert, from a window	mvscanw(3X)
formatted message catalog file, generate a	genocat(1)
formatted message, displays on standard error and console	fmtmsg(3C)
formatted output in a window; print	vw_printw(3X)
formatted output of a varargs argument list; print	vprintf(3S)
formatted output, print in a window	vwprintw(3X)
formatted output, print in window	mvprintw(3X)
formatted output; print	printf(3S)
formatted output; print to standard output, file, or string	vwprintf(3C)
formatted read and conversion from stream file or character string	scanf(3S)
formatted wide-character output; print	fwprintf(3C)
formatter; simple text	adjust(1)
formatters: check or print documents formatted with the mm macros	mm(1)
formatters: format text file for CRT or line-printer output	nroff(1)
formatting conventions of current locale; query numeric	localeconv(3C)
formatting documents, MM macro package for	mm(5)
formatting manpages, macro package for	man(5)
formatting routines, define label for	setlabel(3)
fpathconf() - get configurable path name variables	pathconf(2)
fpclassify() - floating-point classification macro	fpclassify(3M)
fprintf() - print formatted output to a file	printf(3S)
fputc() , putc() - put character on a stream	putc(3S)
fputs() - put a string on a stream	puts(3S)
fputwc() , putwc() - put wide character on a stream	putwc(3C)
fputws() - put a wide character string on a stream file	putws(3C)
frame-buffer devices; information for raster	framebuf(7)
framebuf - information for raster frame-buffer devices	framebuf(7)
fread() - buffered binary input to a stream file	fread(3S)
frecover - selectively recover files	frecover(1M)
free a per-process timer	rmtimer(3C)
free disk clusters; report number of	dosdf(1)
free memory associated with word expansions	wordexp(3C)
free memory for library structure (X/OPEN TLI-XTI)	t_free(3)
free memory in the background is enabled, zeroing of	pagezero_daemon_enabled(5)
free memory used by an unwind environment	uwx_free(3X)
free memory used by the callback info structure	uwx_self_free_info(3X)
free memory used by the symbol cache	uwx_release_symbol_cache(3X)
free() - release allocated block of main memory	malloc(3C)
free space percentage, dump file system	dumpfs(1M)
freaddrinfo() - get hostname and address entry	getaddrinfo(3N)
freenetconfig() - get network configuration data base entry	getnetconfig(3N)
freopen() - substitute a named file in place of an already open stream	fopen(3S)
freopen64() - file system API to support large files	fgetpos64(3S)
frequency attributes; get and set mutex spin and yield	pthread_mutexattr_getspin_np(3T)
frexp() - extract mantissa and exponent from floating-point number	frexp(3M)
frexpf() - extract mantissa and exponent from floating-point number (float)	frexp(3M)
frexpl() - extract mantissa and exponent from floating-point number (long double)	frexp(3M)
frexpq() - extract mantissa and exponent from floating-point number (quad)	frexp(3M)
frexpw() - extract mantissa and exponent from floating-point number (extended)	frexp(3M)
from - who is my mail from?	from(1)
from , who is mail from	mailfrom(1)

Index

All Volumes

Description	Entry Name(Section)
fruled - flash/turn off attention LEDs (cell, cabinet and I/O chassis attention LEDs)	fruled(1)
frupower - turn on/off or display current status of power for cells and I/O chassis	frupower(1M)
fs_async - enables write calls to return before write operation is complete	fs_async(5)
fs_symlinks - maximum number of symbolic links used to resolve a path name	fs_symlinks(5)
fs_wrapper - configuration and binary files used by file system administration commands	fs_wrapper(5)
fsadm - file system administration command	fsadm(1M)
fsadm_hfs - HFS file system administration command	fsadm_hfs(1M)
fscanf() - formatted read from named input stream file	scanf(3S)
fsck command; make a lost+found directory for the	mklost+found(1M)
fsck - file system consistency check and interactive repair	fsck(1M)
fsck - HFS file system consistency check and interactive repair	fsck_hfs(1M)
fsck_cacheofs - check integrity of data cached with CacheFS	fsck_cacheofs(1M)
fsck_hfs - HFS file system consistency check and interactive repair	fsck_hfs(1M)
fsclean - determine the shutdown status of HFS file systems	fsclean(1M)
fsctl() - file system control	fsctl(2)
fsdaemon - pass-through daemon for processing system commands	fsdaemon(1M)
fsdb - file system debugger (generic)	fsdb(1M)
fsdb - HFS file system debugger	fsdb_hfs(1M)
fseek() , rewind() , ftell() - reposition a file pointer in a stream	fseek(3S)
fseek() - set position of next I/O operation on stream file	fseek(3S)
fseek_unlocked() - set position of next I/O operation on stream file, no locking of stream for multi-thread applications	fseek(3S)
fseeko() - set position of next I/O operation on stream file, non-POSIX API	fseek(3S)
fseeko64() - file system API to support large files	fgetpos64(3S)
fsetacl() - set access control list (ACL) information	setacl(2)
fsetaclentry() - add, modify, or delete access control list entry	setaclentry(3C)
fsetpos() - restore file position indicator for a stream	fgetpos(3S)
fsetpos64() - file system API to support large files	fgetpos64(3S)
fsirand : install random inode generation numbers	fsirand(1M)
fspec - format specification in text files	fspec(4)
fstab - static information about the file systems	fstab(4)
fstadm - defines and manages file system stack templates	fstadm(1M)
fstat() - get file status	fstat(2)
fstat64() - non-POSIX standard API interfaces to support large files	creat64(2)
fstatfs() , statfs() - get file system statistics	statfs(2)
fstatfsdev() - get file system statistics	statfsdev(3C)
fstatvfs() - get open file information	statvfs(2)
fstatvfs64() - non-POSIX standard API interfaces to support large files	creat64(2)
fstatvfsdev() - get file system information	statvfsdev(3C)
fstatvfsdev64() - file system API to support large files	fgetpos64(3S)
fstyp - determine file system type	fstyp(1M)
fstypes - file that registers distributed file system packages	fstypes(4)
fsweb - launch the Disks and File Systems tool of HP System Management Homepage (HP SMH) ...	fsweb(1M)
fsync() , fdatasync() - synchronize a file's in-core state with its state on disk	fsync(2)
ftell() - get offset from beginning-of-file of current byte in stream file	fseek(3S)
ftell_unlocked() - get offset from beginning-of-file of current byte in stream file, no locking of stream for multi-thread applications	fseek(3S)
ftello() - get offset from beginning-of-file of current byte in stream file, non-POSIX API	fseek(3S)
ftello64() - file system API to support large files	fgetpos64(3S)
ftime() - get date and time more precisely (Version 7 compatibility only)	ftime(2)
ftio - faster tape I/O	ftio(1)
ftok() - create interprocess communication identifier	ftok(3C)
ftp , rexec , and rexec() , login information for	netrc(4)
FTP configuration files	ckconfig(1)
ftp - file transfer program	ftp(1)
FTP server logfile	xferlog(5)
ftp servers, shutdown message	ftpsht(1)
ftp user, current process information	ftpwho(1)
ftpaccess - ftpd configuration file	ftpaccess(4)
ftpconversions - ftpd conversions database	ftpconversions(4)
ftpcount - show current number of users for each class	ftpcount(1)

Description	Entry Name(Section)
ftpd ; security file for	ftpdusers(4)
ftpd configuration file	ftpdaccess(4)
ftpd conversions database	ftpconversions(4)
ftpd - file transfer protocol server	ftpd(1M)
ftpd individual user host access file	ftphosts(4)
ftpd virtual hosting configuration specification file	ftpservers(4)
ftpgroups - group password file	ftpgroups(4)
ftphosts - ftpd individual user host access file	ftphosts(4)
ftprestart - remove shutdown message file created by ftpshut	ftprestart(1)
ftpservers - ftpd virtual hosting configuration specification file	ftpservers(4)
ftpshut - create shutdown message file for ftp servers	ftpshut(1)
ftpshut, shutdown message file	ftprestart(1)
ftpdusers - security file for ftpd	ftpdusers(4)
ftpwho - show current process information for each ftp user	ftpwho(1)
ftruncate() - truncate a file to a specified length	truncate(2)
ftruncate64() - non-POSIX standard API interfaces to support large files	creat64(2)
ftw() - walk a file tree executing a function	ftw(3C)
ftw64() - file sysmmmaptem API to support large files	fgetpos64(3S)
full name: in elm aliases	newalias(1)
full path name of an open file, get the	pstat(2)
function - define a shell function	sh-posix(1)
function key codes from a terminal; get an array of wide characters and	getn_wstr(3X)
function keys, enable/disable abbreviation of	keypad(3X)
function to be called at program termination, register a	atexit(3)
function to return [EOVERFLOW] if values do not fit in fields; causes uname() system	uname_overflow(5)
function, enhanced pad management	subpad(3X)
function, execute descending a directory tree	ftw(3C)
function, relative window creation	derwin(3X)
function, window refresh control	touchwin(3X)
function; data returned by the stat()	stat(5)
function; event management (EVM) callback	EvmCallback(5)
function; structures needed when using the fadvise()	fadvise(5)
functions and constants; math	math(5)
functions and macros; complex	complex(5)
functions for screen file input/output	scr_dump(3X)
functions from libc; subset of	libcres(5)
functions of HP 2640- and HP 2621-series terminals, handle special	hp(1)
functions of the second kind; Bessel	y0(3M)
functions that map between an interface name and index value	if_nameindex(3N)
functions, allow signals to interrupt	siginterrupt(2)
functions, floating-point environment macros and	fenv(5)
functions, for input mode control	cbreak(3X)
functions, for line update status	redrawwin(3X)
functions, for pad management	newpad(3X)
functions, query, for terminal insert and delay capability	has_ic(3X)
functions, screen initialisation functions	initscr(3X)
functions, soft label	slk_atroff(3X)
functions, terminal output control	clearok(3X)
functions, window creation functions	newwin(3X)
functions, window cursor location	move(3X)
functions, window refresh control	is_linetouched(3X)
functions; event service	EvmSrvStart(3)
functions; string to NaN conversion	nan(3M)
funlockfile() , flockfile() - explicit locking of streams within a multi-thread application	flockfile(3S)
fuser - list processes using a file or file structure	fuser(1M)
fwide() - set stream orientation	fwide(3C)
fwprintf() - print formatted wide-character output	fwprintf(3C)
fwrite() - buffered binary output to a stream file	fread(3S)
fwscanf() - convert formatted wide-character input	fwscanf(3C)
fwtmp - manipulate connect accounting records	fwtmp(1M)

Index

All Volumes

Description	Entry Name(Section)
gai_streerror() - get hostname and address entry	getaddrinfo(3N)
gamma() - log gamma function	lgamma(3M)
gamma, true, functions	tgamma(3M)
gammaf() - log gamma function (float)	lgamma(3M)
gamma1() - log gamma function (long double)	lgamma(3M)
gammaq() - log gamma function (quad)	lgamma(3M)
gammaw() - log gamma function (extended)	lgamma(3M)
gang scheduling of threads and MPI processes	gang_sched(7)
gang_sched - Gang Scheduler	gang_sched(7)
gated configuration guide	gated.conf(4)
gated - gateway routing daemon	gated(1M)
gated.conf - GateDaemon Configuration Guide	gated.conf(4)
gated; operational user interface for	gdc(1M)
GateDaemon Configuration Guide	gated.conf(4)
gateway routing daemon	gated(1M)
gateways, query RIP	ripquery(1M)
gateways; monitor OSPF	ospf_monitor(1M)
gcore - get core images of running processes	gcore(1)
gcrt0.o - execution startup routines	crt0(3)
gcrt0.o - execution startup routines for PA-RISC systems	crt0_pa(3)
gcvt() - convert floating-point number to string	ecvt(3C)
gdc - operational user interface for gated	gdc(1M)
gencat - generate a formatted message catalog file	gencat(1)
general information, introduction to HP-UX	intro(9)
general terminal interface	termio(7)
general terminal interface, extended	termiox(7)
generate a DES encryption key	makekey(1)
generate a formatted message catalog file	gencat(1)
generate a locale environment	localedef(1M)
generate an IOT fault	abort(3C)
generate C header files	rpcgen(1)
generate file name of controlling terminal	ctermid(3S)
generate file names	glob(3C)
generate hashing encryption	crypt(3C)
generate hashing encryption on large strings	bigcrypt(3C)
generate iconv translation tables	genxlt(1)
generate large primes, factor a number	factor(1)
generate path names from i-numbers	ncheck(1M)
generate permuted index	ptx(1)
generate printable representation of a character	unctrl(3X)
generate printable representation of a wide character	wunctrl(3X)
generate RPC protocols, C header files	rpcgen(1)
generate uniformly distributed pseudo-random numbers	drand48(3C)
generates and validates GSS-API tokens for kernel RPC	gssd(1M)
generation tool for DNSSEC; key	dnssec-keygen(1)
generation tool; rndc key	rndc-confgen(1)
generator, simple random-number	rand(3C)
generator, strong random number	random(7)
(generic), extend a file system size	extendfs(1M)
generic device control commands	ioctl(5)
generic file system debugger	fsdb(1M)
generic file system, construct	mkfs(1M)
Generic Security Service Application Programming Interface	gssapi(5)
genxlt - generate iconv translation tables	genxlt(1)
geocustoms - configure system language on multi-language systems	geocustoms(1M)
get a multi-byte character length limited string from the terminal	getnstr(3X)
get a multi-byte character string from the terminal	getstr(3X)
get a single-byte character from the terminal	getch(3X)
get a string from a stream	gets(3S)
get a user's effective access rights to a file	getaccess(2)
get a version of an SCCS file	get(1)

Description	Entry Name(Section)
get a wide character from a terminal	get_wch(3X)
get a wide-character string and rendition from a cchar_t	getcchar(3X)
get a wide-character string from a stream file	fgetws(3C)
get access control list (ACL) information	getacl(2)
get additional cursor and window coordinates	getbegyx(3X)
get address of connected peer	getpeername(2)
get address of symbol in shared object	dlsym(3C)
get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
get an identifier for the current host	gethostid(2)
get and set concurrency level of unbound threads	pthread_getconcurrency(3T)
get and set current user context; DEPRECATED	getcontext(2)
get and set options on sockets	getsockopt(2)
get and set the prioceiling attribute	pthread_mutexattr_getprotocol(3T)
get and set the prioceiling of a mutex	pthread_mutex_getprioceiling(3T)
get and set the protocol attribute	pthread_mutexattr_getprotocol(3T)
get and set the scheduling policy and associated parameters	pthread_getschedparam(3T)
get and set the thread-specific data associated with a key	pthread_getspecific(3T)
get attributes for pthread	pthread_attr_getdetachstate(3T)
get audit files; start or halt the auditing system and set or	audctl(2)
get audit process flag for calling process	getaudproc(2)
get character or word from a stream file	getc(3S)
get command line of a process	pstat(2)
get configurable path name variables	pathconf(2)
get configurable system variables	sysconf(2)
get core images of running processes	gcore(1)
get current display width for user and group names	ug_display_width(3C)
get current value of system-wide clock	getclock(3C)
get cursor and window coordinates	getyx(3X)
get disk description by its name	getdiskbyname(3C)
get dynamic information about the system	pstat(2)
get entries from a directory in a file-system-independent format	getdirentries(2)
get entries from name list	nlist(3C)
get entries from name list on Integrity systems	nlist_ia(3C)
get entries from name list on PA-RISC systems	nlist_pa(3C)
get entries from system cache of recently looked-up names	pstat(2)
get events and system calls currently being audited	getevent(2)
get file handle for file on remote node	getfh(2)
get file handle for file on remote node, get	getfh(2)
get file status	fstat(2)
get file system statistics	statfs(2)
get file system type info	sysfs(2)
get first few lines in a file	head(1)
get foreground process group ID	tcgetpgrp(3C)
get - get a version of an SCCS file	get(1)
getres group ID	getresuid(3)
get high resolution time	gethrtime(3C)
get hostname and address entry	getaddrinfo(3N)
get information about computer system	uname(2)
get information about resource utilization	getrusage(2)
get information for a dynamically loaded kernel module	modstat(2)
get information for a global kernel symbol	getksym(2)
get information of an I/O object	pstat(2)
get legal user shells	getusershell(3C)
get lines from last part of a file	tail(1)
get login name	logname(1)
get mounted file system statistics	ustat(2)
get name of current host system	gethostname(2)
get name of key	keyname(3X)
get name of the user's terminal or pseudo-terminal	tty(1)
get name of user logged in on this terminal	getlogin(3C)
get network entry	getnetent(3N)

Index

All Volumes

Description	Entry Name(Section)
get network group entry	getnetgrent(3C)
get network host entry	gethostent(3N)
get of an SCCS file, undo a previous	unget(1)
get option letter from argument vector	getopt(3C)
get or set background character and rendition using a complex character	bkgrnd(3X)
get or set background character and rendition using a single-byte character	bkgd(3X)
get or set name of current NIS domain	getdomainname(2)
get or set the process-shared attribute	pthread_mutexattr_getpshared(3T)
get or set the process-shared attribute	pthread_rwlockattr_getpshared(3T)
get or set the thread process-shared attribute	pthread_condattr_getpshared(3T)
get or set the type attribute	pthread_mutexattr_getpshared(3T)
get or set tty baud rate	cfspeed(3C)
get PAM error message string	pam_strerror(3)
get pathname of current working directory	getwd(3C)
get process and child process times	times(2)
get process priority	getpriority(2)
get process, process group, or parent process ID	getpid(2)
get real user, effective user, real group, and effective group IDs	getuid(2)
get RPC entry	getrpcent(3C)
get RPC port number	getrpcport(3N)
get SCCS identification information	what(1)
get section information for ELF files	elf_getscn(3E)
get secure password file entry on trusted systems	getspwent(3X)
get service entry	getservent(3N)
get signal alternate stack context	sigaltstack(2)
get socket address	getsockname(2)
get special attributes for group	getprivgrp(2)
get status information and attributes associated with a message queue	mq_getattr(2)
get supported terminal video attributes	termattrs(3X)
get symbolic link status	lstat(2)
get terminal baud rate	baudrate(3X)
get terminal name	termname(3X)
get Terminal Session Manager state information	tsm.info(1)
get the audit ID (aid) for the current process	getaudit(2)
get the base offset for an object file	elf_getbase(3E)
get the compartment IDs associated with a network interfaces	cmpt_get_ifcid(3)
get the current page size	getpagesize(2)
get the date and time	gettimeofday(2)
get the locale of a program	setlocale(3C)
get the scheduling timeslice value for PTHREAD_SCOPE_PROCESS threads with SCHED_TIMESHARE scheduling policy, set or	pthread_gettimeslice_np(3T)
get the size of the per-process file descriptor table	getdtablesize(2)
get tty device operating parameters	tcattribute(3C)
get user ID	getresuid(3)
get value of a per-process timer	gettimer(3C)
get verbose description of current terminal	longname(3X)
get wide character from a stream file	getwc(3C)
get X.25 line	getx25(1M)
get, file status	stat(2)
get, NLS program message	catgets(3C)
get, or put bootptab entry -	getbootpent(3X)
get, pointer for I/O operations on a stream file, get or reposition	fseek(3S)
get, set, or end protocol entry	getprotoent(3N)
get: data pointer for binary search tree	tsearch(3C)
get: date and time more precisely (Version 7 compatibility only)	ftime(2)
get: file size limits and break value, get or set	ulimit(2)
get: file system descriptor file entry (BSD 4.2 compatibility only)	getfsent(3X)
get: message queue	msgget(2)
get: path-name of current working directory	getcwd(3C)
get: set of semaphores	semget(2)
get: shared memory segment	shmget(2)

Description	Entry Name(Section)
get: time	time(2)
get: value of process interval timer	getitimer(2)
get_expiration_time() - add a specific time interval to the current	get_expiration_time(3T)
get_myaddress() - obsolete library routines for RPC	rpc_soc(3N)
get_resfield() - resolver routines	resolver(3N)
get_secdef_int() - security defaults configuration file routines	secdef(3)
get_secdef_str() - security defaults configuration file routines	secdef(3)
get_wch() - get a wide character from a terminal	get_wch(3X)
get_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
getaccess() - get a user's effective access rights to a file	getaccess(2)
getaccess - list access rights to file(s)	getaccess(1)
getacl(), fgetacl() - get access control list (ACL) information	getacl(2)
getacl - list access control lists for files, JFS only	getacl(1)
getaddrinfo() - get hostname and address entry	getaddrinfo(3N)
getaudit() - get the audit ID (aid) for the current process	getaudit(2)
getauditproc() - get audit process flag for calling process	getauditproc(2)
getauduser() - retrieve the accountable user for the current process	getauduser(3)
getbegyx() - get additional cursor and window coordinates	getbegyx(3X)
getbwent() - write records into new wtmps and btmps database	bwtmps(3C)
getc() - get character or word from a stream file	getc(3S)
getc_unlocked() - get character or word from a stream file	getc(3S)
getcchar() - get a wide-character string and rendition from a cchar_t	getcchar(3X)
getch() - get a single-byte character from the terminal	getch(3X)
getchar() - get character or word from standard input file	getc(3S)
getchar_unlocked() - get character or word from standard input	getc(3S)
getclock() - get current value of system-wide clock	getclock(3C)
getconf - get POSIX configuration values	getconf(1)
getcontext() - get and set current user context; DEPRECATED	getcontext(2)
getcwd() - get path-name of current working directory	getcwd(3C)
getdate() - convert user format date and time	getdate(3C)
getdate_r() - convert user format date and time	getdate(3C)
getdirentries() - get entries from a directory in a file-system-independent format	getdirentries(2)
getdiskbyname() - get disk description by its name	getdiskbyname(3C)
getdomainname() - get or set name of current NIS domain	getdomainname(2)
getdtablesize() - get the size of the per-process file descriptor table	getdtablesize(2)
getdvagent() - return pointer for device assignment database entry for trusted system	getdvagent(3)
getdvagnam() - return success or failure information for trusted system	getdvagent(3)
getegid() - get real user, effective user, real group, and effective group IDs	getuid(2)
getenv() - return value for environment name	getenv(3C)
geteuid() - get real user, effective user, real group, and effective group IDs	getuid(2)
getevent() - get events and system calls currently being audited	getevent(2)
getfh() - get file handle for file on remote node	getfh(2)
getfilexsec - display security attributes of binary executable(s)	getfilexsec(1M)
getfsent() - get next line in file system descriptor file	getfsent(3X)
getfsfile() - search descriptor file for ordinary file entry	getfsent(3X)
getfsspec() - search descriptor file for special (device) file entry	getfsent(3X)
getfstype() - search descriptor file for specified file type entry	getfsent(3X)
getgid() - get real user, effective user, real group, and effective group IDs	getuid(2)
getgrent() - get next entry in group file	getgrent(3C)
getgrgid() - get entry from group file that matches gid	getgrent(3C)
getgrgid_r() - get group file entry	getgrent(3C)
getgrnam() - get entry from group file that matches group name	getgrent(3C)
getgrnam_r() - get group file entry	getgrent(3C)
getgroups() - get group access list	getgroups(2)
gethostbyaddr() - get network host entry	gethostent(3N)
gethostbyaddr_r() - get network host entry (thread-safe)	gethostent(3N)
gethostbyname() - get network host entry	gethostent(3N)
gethostbyname_r() - get network host entry (thread-safe)	gethostent(3N)
gethostent() - get network host entry	gethostent(3N)
gethostent_r() - get network host entry (thread-safe)	gethostent(3N)
gethostid() - get an identifier for the current host	gethostid(2)

Index

All Volumes

Description	Entry Name(Section)
gethostname() - get name of current host system	gethostname(2)
gethrtime() - get high resolution time	gethrtime(3C)
getitimer() - get value of process interval timer	getitimer(2)
getksym() - get information for a global kernel symbol	getksym(2)
getlocale() - get the locale of a program	setlocale(3C)
getlocale_r() - get the locale of a program (MT-Safe)	setlocale(3C)
getlogin() - get name of user logged in on this terminal	getlogin(3C)
getlogin_r() - get name of user logged in and return name to buffer	getlogin(3C)
getmaxyx() - get additional cursor and window coordinates	getbegyx(3X)
getmemwindow - extract window IDs of user processes from /etc/services.window	getmemwindow(1M)
getmntent() - get file system descriptor file entry	getmntent(3X)
getmntent_r() - get file system descriptor file entry	getmntent(3X)
getmsg() - receive next message from a STREAMS file	getmsg(2)
getn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
getnameinfo() - get hostname and address entry	getaddrinfo(3N)
getnetbyaddr() - get network entry	getnetent(3N)
getnetbyname() - get network entry	getnetent(3N)
getnetconfig() - get network configuration data base entry	getnetconfig(3N)
getnetconfigent() - get network configuration data base entry	getnetconfig(3N)
getnetent() - get network entry	getnetent(3N)
getnetgrent() - get network group entry	getnetgrent(3C)
getnetname() - library routines for secure remote procedure calls	secure_rpc(3N)
getnetpath() - get /etc/netconfig entry corresponding to NETPATH component	getnetpath(3N)
getnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
getopt() - get option letter from argument vector	getopt(3C)
getopt - parse command options	getopt(1)
getopts - parse an argument list	sh-posix(1)
getopts - parse utility (command) options	getopts(1)
getpagesize() - get the current page size	getpagesize(2)
getparyx() - get additional cursor and window coordinates	getbegyx(3X)
getpass() - read a password from terminal while suppressing echo	getpass(3C)
getpeername() - get address of connected peer	getpeername(2)
getpgid() - get process group ID of specified process	getpid(2)
getpgrp() - 4.2 BSD-compatible process control facilities	killpg(2)
getpgrp() - get process group ID	getpid(2)
getpgrp2() - get process group ID of specified process	getpid(2)
getpid() - get process ID	getpid(2)
getpmsg() - receive next message from a STREAMS file in a priority order	getmsg(2)
getppid() - get parent process ID	getpid(2)
getprdfent() - manipulate system default database entry for a trusted system	getprdfent(3)
getprdfent() - return pointer for system default database for trusted system	getprdfent(3)
getprdfnam() - manipulate system default database entry for a trusted system	getprdfent(3)
getprdfnam() - return pointer for system default database for trusted system	getprdfent(3)
getpriority() - get process priority	getpriority(2)
getprivgrp() - get special attributes for group	getprivgrp(2)
getprivgrp - get special attributes for group	getprivgrp(1)
getprocxsec - display security attributes of a process	getprocxsec(1M)
getprotobyname() - get protocol entry	getprotoent(3N)
getprotobyname_r() - get protocol entry (thread-safe)	getprotoent(3N)
getprotobynumber() - get protocol entry	getprotoent(3N)
getprotobynumber_r() - get protocol entry (thread-safe)	getprotoent(3N)
getprotoent() - get protocol entry	getprotoent(3N)
getprotoent_r() - get protocol entry (thread-safe)	getprotoent(3N)
getprpw - display protected password database	getprpw(1M)
getprpwaid() - get protected password database audit ID (trusted systems)	getprpwent(3)
getprpwaid() - manipulate protected password database entries (for trusted systems only)	getprpwent(3)
getprpwent() - manipulate protected password database entries (for trusted systems only)	getprpwent(3)
getprpwent() - manipulate protected password database entries (for trusted systems only)	getprpwent(3)
getprpwnam() - get protected password database user name (trusted systems)	getprpwent(3)

Description	Entry Name(Section)
getprpnam() - manipulate protected password database entries (for trusted systems only) getprpwent(3)
getprpwuid() - get protected password database user ID (trusted systems) getprpwent(3)
getprpwuid() - manipulate protected password database entries (for trusted systems only) getprpwent(3)
getprtcent() - manipulate terminal control database entry getprtcent(3)
getprtcnam() - manipulate terminal control database entry getprtcent(3)
getpublickey() - retrieve public or secret key getpublickey(3M)
getpw() - get name from UID (obsolete) getpw(3C)
getpwent() - get next password file entry getpwent(3C)
getpwnam() - get password file entry matching login name getpwent(3C)
getpwnam_r() - get password file entry getpwent(3C)
getpwuid() - get password file entry matching uid getpwent(3C)
getpwuid_r() - get password file entry getpwent(3C)
getresgid() - get group ID getresuid(3)
getresuid() - get user ID getresuid(3)
getrlimit() - get system resource consumption limit getrlimit(2)
getrlimit64() - non-POSIX standard API interfaces to support large files creat64(2)
getrpcbyname() - get RPC entry getrpcnt(3C)
getrpcbynumber() - get RPC entry getrpcnt(3C)
getrpcnt() - get RPC entry getrpcnt(3C)
getrpcport() - get RPC port number getrpcport(3N)
getrules - display compartment rules getrules(1M)
getrusage() - get information about resource utilization getrusage(2)
gets connection-specific data pointer for the sendmail connection smfi_getpriv(3N)
gets() - get a string from a stream gets(3S)
gets the value of a sendmail macro smfi_getsymval(3N)
getsecretkey() - retrieve public or secret key getpublickey(3M)
getservbyname() - get service entry getservent(3N)
getservbyname_r() - get service entry (thread-safe) getservent(3N)
getservbyport() - get service entry getservent(3N)
getservbyport_r() - get service entry (thread-safe) getservent(3N)
getservent() - get service entry getservent(3N)
getservent_r() - get service entry (thread-safe) getservent(3N)
getsid() - get session ID getsid(2)
getsockname() - get socket address getsockname(2)
getsockopt() - get options on sockets getsockopt(2)
getspent() - get next secure password file entry getspent(3C)
getspnam() - get secure password file entry matching name() getspent(3C)
getspnam_r() - get secure password file entry matching name() getspent(3C)
getspwaid() - get next secure password file audit ID on trusted systems getspwent(3X)
getspwaid_r() - get secure password file entry on trusted systems getspwent(3X)
getspwent() - get next secure password file entry on trusted systems getspwent(3X)
getspwent_r() - get secure password file entry on trusted systems getspwent(3X)
getspwnam() - get secure password file entry matching login name name() on trusted systems getspwent(3X)
getspwnam_r() - get secure password file entry on trusted systems getspwent(3X)
getspwuid() - get secure password file entry matching uid() on trusted systems getspwent(3X)
getspwuid_r() - get secure password file entry on trusted systems getspwent(3X)
getstr() - get a multi-byte character string from the terminal getstr(3X)
getsubopt() - parse suboptions from a string getsubopt(3C)
gettimeofday() - get the date and time gettimeofday(2)
gettimer() - get value of a per-process timer gettimer(3C)
gettune() - get value of a kernel tunable parameter gettune(2)
gettxt() , create message files for use by mkmsgs(1)
gettxt() - read text string from message file gettxt(3C)
getty for 2-way line accessible to uucp uugetty(1M)
getty - set terminal type, modes, speed, and line discipline getty(1M)
getty to remote terminal, spawn (call terminal) ct(1)
gettydefs - speed and terminal settings used by getty gettydefs(4)
getuid() - get real user, effective user, real group, and effective group IDs getuid(2)
getusershell() - get legal user shells getusershell(3C)

Index

All Volumes

Description	Entry Name(Section)
getut - access utmp file entry	getut(3C)
getutent () - get pointer to next entry in utmp file	getut(3C)
getutent_r () - get pointer to next entry in utmp file	getut(3C)
getutid () - get pointer to entry matching ID in utmp file	getut(3C)
getutid_r () - get pointer to entry matching ID in utmp file	getut(3C)
getutline () - get pointer to entry matching line in utmp file	getut(3C)
getutline_r () - get pointer to entry matching line in utmp file	getut(3C)
getuts - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
GETUTSENT () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
getutsent () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
GETUTSID () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
getutsid () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
GETUTSLINE () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
getutslid () - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
getutx - access utmpx file entry	getutx(3C)
getutxent () - get pointer to next entry in a utmpx file	getutx(3C)
getutxid () - get pointer to entry matching id in a utmpx file	getutx(3C)
getutxline () - get pointer to entry matching line in a utmpx file	getutx(3C)
getw () - get data word (integer) from a stream file	getc(3S)
getw_unlocked () - get data word (integer) from a stream file	getc(3S)
getwc () - get wide character from a stream file	getwc(3C)
getwchar () - get wide character from a stream file	getwc(3C)
getwd () - get pathname of current working directory	getwd(3C)
getwin () - dump window to and reload window from a file	getwin(3X)
getx25 - get X.25 line	getx25(1M)
getyx () - get cursor and window coordinates	getyx(3X)
glob - echo without '\ ' escapes	csh(1)
glob () - file name generation function	glob(3C)
global kernel symbol; get information	getksym(2)
global search path for dynamically loadable kernel modules, change	modpath(2)
globfree () - free space associated with file name generation function	glob(3C)
glossary - description of common HP-UX terms	glossary(9)
gmacs editing mode	sh-posix(1)
gmtime (), gmtime_r () - convert date and time to Greenwich Mean Time	ctime(3C)
goto - continue execution on specified line	csh(1)
goto , save/restore stack environment for non-local	setjmp(3C)
gprof - display call graph execution profile data	gprof(1)
grant access to STREAMS slave pty	grantpt(3C)
grantpt () - grant access to STREAMS slave pty	grantpt(3C)
graph and display execution profile data	gprof(1)
graphic character	glossary(9)
graphics driver will not claim; PCI Vendor/Device ID that the gvid	gvid_no_claim_dev(5)
greatest delta (slow-down factor), POSIX async I/O request priorities	aio_prio_delta_max(5)
greatest delta (slowdown factor) allowed in POSIX async IO request priorities	aio_prio_delta_max(5)
Greenwich mean and local time, difference	timezone(5)
grep - search a file for a pattern (compact algorithm)	grep(1)
grget - get group information	pwget(1)
groper; domain information	dig(1M)
group	glossary(9)
group access file information; change WU-FTPD	privatepw(1)
group access list	glossary(9)
group access list: get group access list	getgroups(2)
group access list: initialize group access list	initgroups(3C)
group access list: set group access list	setgroups(2)
group account configuration tool; starts the HP-UX user and	ugweb(1M)
group and owner of a file, change	chown(2)
group and password hashing and caching statistics	pwgr_stat(1M)
group and password hashing and caching daemon	pwgrd(1M)
group and/or owner, change in access control list (ACL)	chownacl(3C)
group configuration, restore volume	vgcfgrestore(1M)
group entry, network, get or set	getnetgrent(3C)

Description	Entry Name(Section)
group file	login(1)
group file entry; get	getgrent(3C)
group file; check	pwck(1M)
group - group access and identification file, grp.h	group(4)
group ID	glossary(9)
group ID for job control, set process	setpgid(2)
group ID, create session and set process	setsid(2)
group ID, foreground process, get	tcgetpgrp(3C)
group ID, foreground process, set	tcsetpgrp(3C)
group ID, get	getresuid(3)
group ID: set real, effective, and/or saved group or user IDs	setresuid(2)
group ID; set	setuid(2)
group IDs and names; print user and	id(1)
group IDs; get real user, effective user, real group, and effective	getuid(2)
group IDs; set effective	seteuid(2)
group IDs; sets the real and effective	setregid(2)
group information, get (grget)	pwget(1)
group memberships, show	groups(1)
group name enablement and display; long user and	lugadmin(1M)
group names; get current display width for user and	ug_display_width(3C)
group of file, change	chown(1)
group on the system; modify a	groupmod(1M)
group password file	ftpgroups(4)
group privileges; HP-UX	privgrp(5)
group to the system; add a new	groupadd(1M)
group, get or set special attributes	getprivgrp(2)
group, get special attributes for	getprivgrp(1)
group, log in to a new	newgrp(1)
group; add a new	groupadd(1M)
group; create physical volume for use in LVM volume	pvcreate(1M)
group; delete from the system	groupdel(1M)
group; modify a	groupmod(1M)
groupadd - add a new group to the system	groupadd(1M)
groupdel - delete a group from the system	groupdel(1M)
groupmod - modify a group on the system	groupmod(1M)
groups - show group memberships	groups(1)
groups; scan physical volumes for LVM volume	vgscan(1M)
groups; set special privileges for	setprivgrp(1M)
grpck - password/group file checkers	pwck(1M)
gsignal() - raise a software signal	ssignal(3C)
GSS-API tokens for kernel RPC; generates and validates	gssd(1M)
gss_accept_sec_context() - establish security context	gss_accept_sec_context(3)
gss_acquire_cred() - acquire handle for credential	gss_acquire_cred(3)
gss_add_cred() - allow an application to acquire a handle for existing, named credential	gss_add_cred(3)
gss_add_oid_set_member() - add an Object Identifier (OID) to an OID set	gss_add_oid_set_member(3)
gss_canonicalize_name() - convert an internal name to an internal MN name ..	gss_canonicalize_name(3)
gss_compare_name() - allow an application to compare two internal names to determine whether they are equivalent	gss_compare_name(3)
gss_context_time() - check the number of seconds the context will remain valid	gss_context_time(3)
gss_create_empty_oid_set() - create a new, empty OID set, to which members can be added	gss_create_empty_oid_set(3)
gss_delete_sec_context(3) - delete security context	gss_delete_sec_context(3)
gss_display_name() - provide textual representation of an opaque internal name to an application	gss_display_name(3)
gss_display_status() - provides an application with the textual representation of a GSSAPI status code that can be displayed to a user or used for logging	gss_display_status(3)
gss_duplicate_name() - allow an application to create an exact duplicate of the existing internal name	gss_duplicate_name(3)
gss_export_name() - convert a mechanism name (MN) to a form suitable for direct comparison	gss_export_name(3)
gss_export_sec_context() - transfer a security context to another process on a single	

Index

All Volumes

Description	Entry Name(Section)
machine	gss_export_sec_context(3)
gss_get_mic() - calculate a cryptographic message integrity code (MIC) for a message and return in a token	gss_get_mic(3)
gss_import_name() - convert a printable name to an internal form	gss_import_name(3)
gss_import_sec_context() - transfer a security context to another process on a single machine	gss_import_sec_context(3)
gss_indicate_mechs() - allow application to determine which underlying security mechanisms are available	gss_indicate_mechs(3)
gss_init_sec_context() - establish a security context between context initiator and context acceptor	gss_init_sec_context(3)
gss_inquire_context() - obtain information about a security context	gss_inquire_context(3)
gss_inquire_cred() - provide the calling application information about a credential	gss_acquire_cred(3)
gss_inquire_cred_by_mech() - provide the calling application per-mechanism information about a credential	gss_inquire_cred_by_mech(3)
gss_inquire_mechs_for_name() - list the mechanisms that support the specified name-type	gss_inquire_mechs_for_name(3)
gss_inquire_names_for_mech() - list the name-types supported by the specified mechanism	gss_inquire_names_for_mech(3)
gss_OID_set object, free storage associated with	gss_release_oid_set(3)
gss_process_context_token() - pass a context to the security service	gss_process_context_token(3)
gss_release_buffer() - free storage associated with a buffer	gss_release_buffer(3)
gss_release_cred() - mark a credential for deletion	gss_release_cred(3)
gss_release_name() - free storage associated with an internal name allocated by a GSSAPI routine	gss_release_name(3)
gss_release_oid_set() - free storage associated with a gss_OID_set object	gss_release_oid_set(3)
gss_test_oid_set_number() - check an OID set for a specified OID	gss_test_oid_set_number(3)
gss_unwrap() - verify a message with attached message integrity code (MIC) and decrypt message content	gss_unwrap(3)
gss_verify_mic() - check a cryptographic message integrity code (MIC) against a message to verify its integrity	gss_verify_mic(3)
gss_wrap() - attach a message integrity code (MIC) to a message, and optionally encrypt	gss_wrap(3)
gss_wrap on a context, determine a token-size limit for	gss_wrap_size_limit(3)
gss_wrap_size_limit() - determine a token-size limit for gss_wrap on a context	gss_wrap_size_limit(3)
GSSAPI (Generic Security Service), shared library	libgss(4)
gssapi - Generic Security Service Application Programming Interface	gssapi(5)
GSSAPI routine, free storage associated with an internal name allocated	gss_release_name(3)
GSSAPI status code, textual representation	gss_display_status(3)
gsscred - add, remove and list gsscred table entries	gsscred(1M)
gsscred mapping table; remove duplicate entries from	gsscred_clean(1M)
gsscred table entries; add, remove and list	gsscred(1M)
gsscred_clean - remove duplicate entries from gsscred mapping table	gsscred_clean(1M)
gssd - generates and validates GSS-API tokens for kernel RPC	gssd(1M)
gtty() - control terminal device (Bell Version 6 compatibility)	stty(2)
gvid graphics driver will not claim; PCI Vendor/Device ID that the	gvid_no_claim_dev(5)
gvid_no_claim_dev - PCI Vendor/Device ID that the gvid graphics driver will not claim	gvid_no_claim_dev(5)
halfdelay() - control input character delay mode	halfdelay(3X)
HALGOL programs, execute	opx25(1M)
halt system operation	shutdown(1M)
halt the auditing system and set or get audit files; start or	audctl(2)
halt then reboot the system	reboot(1M)
halt/start the auditing system	audsys(1M)
handle physical volume size changes of an existing LVM volume group	vgmodify(1M)
handle special functions of HP 2640- and HP 2621-series terminals	hp(1)
handoff mode; disable mutex-specific or process-wide mutex	pthread_mutexattr_getspin_np(3T)
hangups, run a command immune to	nohup(1)
hard maximum number of file descriptors per process	maxfiles_lim(5)
hardlinks; checks the consistency of compartment rules for files with multiple	vhardlinks(1M)
hardware event viewer tool (a Web interface); start the HP-UX	slweb(1M)
hardware insert- and delete-character features, enable or disable use of	idcok(3X)
hardware machine model/series identification	model(4)

Description	Entry Name(Section)
hardware model information; print	model(1)
hardware partitionable complex; display information about a	parstatus(1)
hardware path	glossary(9)
has_colors() - color manipulation functions	can_change_color(3X)
has_ic() - query functions for terminal insert and delay capability	has_ic(3X)
has_il() - query functions for terminal insert and delay capability	has_ic(3X)
hash codes, convert 9-digit to or from text for spell checking	spell(1)
hash - display and set command locations	sh-posix(1)
hash search tables, manage	hsearch(3C)
hash tables, determines the size of the networking	tphashsz(5)
hash tables, size of hashed pool of spinlocks protecting the channel queue	chanq_hash_locks(5)
hash value for ELF files, compute	elf_hash(3E)
hashcheck - convert spelling reference list words to 9-digit hash codes for spell	spell(1)
hashed pool of spinlocks protecting the channel queue hash tables, size of	chanq_hash_locks(5)
hashed spinlock pool size, System V IPC	sysv_hash_locks(5)
hashing and caching statistics, password and group	pwgr_stat(1M)
hashing and caching, password and group, daemon	pwgrd(1M)
hashing encryption on large strings; generate	bigcrypt(3C)
hashing encryption, generate	crypt(3C)
hashmake - convert text words to 9-digit hash codes for spell	spell(1)
hashstat - print hash table effectiveness statistics	csh(1)
hasmntopt() - search mount option field in file system descriptor file	getmntent(3X)
havedisk() - get performance data from remote kernel	rstat(3N)
hcreate() - allocate space for new hash search table	hsearch(3C)
hdestroy() - destroy existing hash search table	hsearch(3C)
hdlpreg_hash_locks - determines size of pregion spinlock pool (OBSOLETE)	hdlpreg_hash_locks(5)
head - get first few lines in a file	head(1)
header file for future applications	portal(5)
header file of macros for handling device numbers	mknod(5)
header file, data link provider interface standard	dlpi(4)
header files, C, generate	rpcgen(1)
header files; description of named defines and other specifications for namespace from HP-UX	stdsyms(5)
header for elf32 or elf64 file; retrieve class-dependent object file	elf_getehdr(3E)
header on a device file, write an EFI file system	efi_fsinit(1M)
header to the current message; adds a	smfi_addheader(3N)
header to the current sendmail message; prepends a	smfi_inshheader(3N)
header; changes or deletes a message	smfi_chghheader(3N)
held on behalf of an NFS client, clear locks	clear_locks(1M)
help on SCCS commands; ask for	sccshelp(1)
herror() - resolver routines	resolver(3N)
hexadecimal equivalents: ASCII character set	ascii(5)
hexadecimal file dump; octal and	od(1)
HFS access control lists (ACLs); introduction to	acl(5)
HFS file system administration command	fsadm_hfs(1M)
HFS file system consistency check and interactive repair	fsck_hfs(1M)
HFS file system debugger	fsdb_hfs(1M)
HFS file system disk blocks, report number of free	df_hfs(1M)
HFS file system only; convert string form to access control list (ACL) structure,	strtoacl(3C)
HFS file system open inodes that can be in memory, maximum number of	ninode(5)
HFS file system quotas; turn on and off	quotaon(1M)
HFS file system size, extend	extendfs_hfs(1M)
HFS file system to allow long file names, convert an	convertfs(1M)
HFS file system with compaction; copy	dcopy(1M)
HFS file system with label checking; copy	volcopy_hfs(1M)
HFS file system, construct an	mkfs_hfs(1M)
HFS file system, list file names and statistics	ff_hfs(1M)
HFS file system: tune an existing file system	tunefs(1M)
HFS file system; construct a new	newfs_hfs(1M)
HFS file system; summarize ownership	quot_hfs(1M)
HFS file systems; determine the shutdown status of	fsclean(1M)

Index

All Volumes

Description	Entry Name(Section)
HFS file systems; mount and unmount	mount_hfs(1M)
HFS file systems; quota consistency checker	quotacheck_hfs(1M)
hfs_revra_per_disk - maximum HFS file system blocks to be read in one read-ahead operation when sequentially reading backwards	hfs_revra_per_disk(5)
hg - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_busywait() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_context_switch_involuntary() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_context_switch_tries() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_context_switch_voluntary() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_gethrcycles() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_gethrtime() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_getspu() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_nano_to_cycle_ratio() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_init() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_is_onRunQ() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_is_reporting() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_is_running() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_nMailboxes() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_nMailboxesInUse() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_public_remove() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
hg_setcrit() - Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
HIDS , enable intrusion detection data source	enable_ids(5)
hier - file system hierarchy	hier(5)
hierarchical directory	glossary(9)
hierarchies; extracts, writes, and lists archive files; copies files and directory	pax(1)
hierarchy, directory, recursively descend a, executing a function	ftw(3C)
hierarchy, file system	hier(5)
high resolution sleep	nanosleep(2)
high resolution time, get	gethrtime(3C)
high resolution timers support; enable	hires_timeout_enable(5)
hil - HP-HIL device driver	hil(7)
hilkbd - HP-HIL mapped keyboard driver	hilkbd(7)
hires_timeout_enable - enable high resolution timers support	hires_timeout_enable(5)
history - Display event history list	csh(1)
history for interactive programs; input editor and command	ied(1)
hline() - draw lines from single-byte characters and renditions	hline(3X)
hline_set() - draw lines from complex characters and renditions	hline_set(3X)
hold signal upon receipt	sigset(3C)
home directory	glossary(9)
home directory names; list of	usermod(4)
HOME environment variable	login(1)
Homepage (HP SMH); HP System Management	smh(1M)
Homepage (HP SMH); launch the Disks and File Systems tool of HP System Management	fsweb(1M)
Homepage server; starts or stops the HP System Management	hpsmh(1M)

Description	Entry Name(Section)
Hop-by-Hop and Destination options manipulation functions, IPv6	net6_opt_init(3N)
host access control files; format	hosts_access(5)
host access control language extensions	hosts_options(5)
host access file, ftpd	ftphosts(4)
host and network byte order, convert values between	byteorder(3N)
Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel	fcmsutil(1M)
host - DNS lookup utility	host(1)
host is Network Information System server or map master; list which	ypwhich(1)
host name	glossary(9)
host name resolution description	hostname(5)
host name; size of	nodehostnamesize(5)
host names database	hosts(4)
host names; enable maximum length expansion	expanded_node_host_names(5)
host status of local machines (RPC version), show	rup(1)
host system, set or display name of current	hostname(1)
host system; get name of current	gethostname(2)
host system; set name of current	sethostname(2)
host table to name server file format; translate	hosts_to_named(1M)
host test packets, send	ping(1M)
host, remote, execute command on a	on(1)
host2netname() - library routines for secure remote procedure calls	secure_rpc(3N)
hostname and address entry; get	getaddrinfo(3N)
hostname - host name resolution description	hostname(5)
hostname - set or display name of current host system	hostname(1)
hostname; set system initial identity parameter	set_parms(1M)
hosts - hosts name database	hosts(4)
hosts, compute shortest path and route between	pathalias(1)
hosts, remote, authorizing access on local host	hosts.equiv(4)
hosts.equiv file	login(1)
hosts.equiv - security files authorizing access by remote hosts and users on local host	hosts.equiv(4)
hosts_access() - access control library	hosts_access(3)
hosts_access - format of host access control files	hosts_access(5)
hosts_ctl() - access control library	hosts_access(3)
hosts_options - host access control language extensions	hosts_options(5)
hosts_to_named - translate host table to name server file format	hosts_to_named(1M)
hotplug (attention button) events daemon, PCI I/O	hotplugd(1M)
hotplugd - PCI I/O hotplug (attention button) events daemon	hotplugd(1M)
HP 2640- and HP 2621-series terminals, handle special functions of	hp(1)
HP 3000-mode packed-decimal library	hppac(3X)
HP 9000 64-bit shared library with explicit load address; open an	dlopen_pa(3C)
HP 9000 shared library; open an	dlopen_pa(3C)
HP AdvanceLink and Basic Serial server	pcserver(1M)
hp - handle special functions of HP 2640- and HP 2621-series terminals	hp(1)
HP SMH; launch the Disks and File Systems tool of HP System Management Homepage	fsweb(1M)
HP SMH; launch the Network Interfaces Configuration and Network Services Configuration tools of HP System Management Homepage	ncweb(1M)
HP specific extensions for DLPI	dipi_ext(4)
HP System Management Homepage (HP SMH)	smh(1M)
HP System Management Homepage server; starts or stops the	hpsmh(1M)
HP-UX 11.x system or depot; check security-bulletin compliance state of	security_patch_check(1M)
HP VUE (OBSOLETE); audio tools available through	Audio(5)
HP-HIL device driver	hil(7)
HP-HIL mapped keyboard driver	hilkbd(7)
hp-mc680x0 - is processor an HP MC680x0?	machid(1)
hp-pa - is processor an HP PA-RISC?	machid(1)
HP-UX 11i V3 patch check utility	check_patches(1M)
HP-UX and DOS formats; convert ASCII file format between	dos2ux(1)
HP-UX applications on Itanium-based systems running HP-UX; emulate PA-RISC	Aries(5)
HP-UX Auditing System; introduction to	audit(5)
HP-UX bootstrap and installation utility	hpux(1M)

Index

All Volumes

Description	Entry Name(Section)
HP-UX bootstrap for Itanium-based systems	hpux.efi(1M)
HP-UX compartments files	compartments(4)
HP-UX documentation, introduction to	intro(9)
HP-UX documentation; accessing and ordering	manuals(5)
HP-UX general information section, introduction to	intro(9)
HP-UX group privileges	privgrp(5)
HP-UX hardware event viewer tool (a Web interface); start the	slweb(1M)
HP-UX implementations; magic numbers for	magic(4)
HP-UX installed software comparator	sysdiff(1)
HP-UX kernel configuration tool (a Web interface); starts the	kcweb(1M)
HP-UX machine identification	model(4)
HP-UX operating system and <i>HP-UX Reference</i>	introduction(9)
HP-UX operating system, updates	update-ux(1M)
HP-UX patch cleanup utility	cleanup(1M)
HP-UX patch display utility	show_patches(1)
HP-UX Peripheral Device tool, part of the SMH Web interface; start the	pdweb(1M)
HP-UX privileges; description	privileges(5)
<i>HP-UX Reference</i> ; HP-UX operating system and	introduction(9)
HP-UX servers; physical memory allocation policy on cell-based	numa_policy(5)
HP-UX terms; description of common	glossary(9)
HP-UX, extended authentication, account, password, and session service module for	pam_hpsec(5)
HP-UX; emulate PA-RISC HP-UX applications on Itanium-based systems running	Aries(5)
HP-UX; UNIX standards behavior on	standards(5)
hp2686a - laserjet filter	lpfilter(1)
hp2934a - character printer filter	lpfilter(1)
hp9000s200 - is processor an HP 9000 Series 200?	machid(1)
hp9000s300 - is processor an HP 9000 Series 300?	machid(1)
hp9000s400 - is processor an HP 9000 Series 400?	machid(1)
hp9000s500 - is processor an HP 9000 Series 500?	machid(1)
hp9000s700 - is processor an HP 9000 Series 700?	machid(1)
hp9000s800 - is processor an HP 9000 Series 800?	machid(1)
hppac - HP 3000-mode packed-decimal library	hppac(3X)
HPPACADD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACMPD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACCVAD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACCVBD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACCVDA() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACCVDB() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACDIVD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACLONGDIVD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACMPYD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACNSLD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACSLD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACSRD() - HP 3000-mode packed-decimal library	hppac(3X)
HPPACSUBD() - HP 3000-mode packed-decimal library	hppac(3X)
hpsmh - starts or stops the HP System Management Homepage server	hpsmh(1M)
HPSMH; configures the startup mode of the HPSMH server and of the Tomcat instance used by	smhstartconfig(1M)
hpux - HP-UX bootstrap and installation utility	hpux(1M)
HPUX scheduling policy	rtsched(2)
hpux.efi - HP-UX bootstrap for Itanium-based systems	hpux.efi(1M)
hsearch() - hash table search routine	hsearch(3C)
htonl() - convert values between host and network byte order	byteorder(3N)
htons() - convert values between host and network byte order	byteorder(3N)
Huffman code; compress and expand files using	pack(1)
hung cell during cell activation; online activation of a cell from nPartition; cancel online cell operation; monitor online cell operation; reset	parolrad(1M)
hyperbolic cosine and hyperbolic sine together	sinhcosh(3M)
hyperbolic cosine functions	cosh(3M)
hyperbolic sine and hyperbolic cosine together	sinhcosh(3M)
hyperbolic sine functions	sinh(3M)

Description	Entry Name(Section)
hyperbolic tangent functions	tanh(3M)
hyphen - find hyphenated words	hyphen(1)
hyphenated words, find	hyphen(1)
hypot() - Euclidean distance function	hypot(3M)
hypotenuse of a right triangle	hypot(3M)
hypotf() - Euclidean distance function (float)	hypot(3M)
hypotl() - Euclidean distance function (long double)	hypot(3M)
hypotq() - Euclidean distance function (quad)	hypot(3M)
hypotw() - Euclidean distance function (extended)	hypot(3M)
i-number, list path name corresponding to	ff_hfs(1M)
i-numbers, generate path names from	ncheck(1M)
I/O card access information, network	lan(7)
I/O cards and Online Addition of I/O chassis; command for Online Addition/Replacement/Deletion of PCI	olrad(1M)
I/O chassis LEDs; flash/turn off	fruled(1)
I/O chassis; command for Online Addition/Replacement/Deletion of PCI I/O cards and Online Addition of	olrad(1M)
I/O chassis; turn on/off or display current status of power	frupower(1M)
I/O chassis; turn on/off or display current status of power for cells and	frupower(1M)
I/O conditions on multiple file descriptors, monitor	poll(7)
I/O conditions on multiple file descriptors; monitor	poll(2)
I/O data; maximum or minimum amount of physical memory used for caching file	filecache_max(5)
I/O device drivers	<i>see special files</i>
I/O device file; list a special	lssf(1M)
I/O device, who is currently using	fuser(1M)
I/O multiplexing; synchronous	select(2)
I/O of EVM events to and from a file; perform	EvmEventRead(3)
I/O on kernel memory, based on symbol name; perform	kmem(7)
I/O operations on a stream file, get or reposition pointer for	fseek(3S)
I/O operations, POSIX asynchronous, maximum allowed in a listio call	aio_listio_max(5)
I/O operations; percentage of physical memory lockable for request call-back POSIX asynchronous	aio_physmem_pct(5)
I/O pipe to or from a process, open or close	popen(3S)
I/O redirection	glossary(9)
I/O statistics; report	iostat(1)
I/O subsystem, diagnostic interface to	diag0(7)
I/O subsystem, diagnostic interface to	diag1(7)
I/O subsystem; interface for interacting with kernel	libIO(3X)
I/O system; scan	ioscan(1M)
I/O, asynchronous synchronize	aio_fsync(2)
I/O, asynchronous write	aio_write(2)
I/O, asynchronous, error status	aio_error(2)
I/O, asynchronous, POSIX	aio(5)
I/O, asynchronous, start list of operations	lio_listio(2)
I/O, cancel asynchronous	aio_cancel(2)
I/O, control character device special file	aio_ioctl(2)
I/O, POSIX asynchronous	aio(5)
I/O, read asynchronous	aio_read(2)
I/O, return asynchronous status	aio_return(2)
I/O, suspend for asynchronous completion	aio_suspend(2)
I/O, wait for asynchronous completion	aio_suspend(2)
I/O, wait for multiple asynchronous requests	aio_reap(2)
I/O; faster tape	ftio(1)
iconv() - code set conversion routine, convert character	iconv(3C)
iconv - codeset conversion	iconv(1)
iconv translation tables to a readable format, dump	dmpxlt(1)
iconv translation tables; generate	genxlt(1)
iconv_close() - code set conversion routine, deallocate conversion descriptor	iconv(3C)
iconv_open() - code set conversion routine, return conversion descriptor	iconv(3C)
ID (aid) for the current process; get the audit	getaudit(2)
ID (aid) for the current process; set the audit	setaudit(2)

Index

All Volumes

Description	Entry Name(Section)
ID for job control, set process group	setpgid(2)
ID of running program or start program in particular memory window; change window ...	setmemwindow(1M)
id - print user and group IDs and names	id(1)
ID to file path; map device	devnm(3)
ID, create session and set process group	setsid(2)
ID, foreground process group, get	tcgetpgrp(3C)
ID, foreground process group, set	tcsetpgrp(3C)
ID, get group ID	getresuid(3)
ID, get session	getsid(2)
ID, get terminal session	tcgetsid(3C)
ID, get user ID	getresuid(3)
ID, obtain the thread ID for the calling thread	pthread_self(3T)
ID; print effective current user	whoami(1)
ID; set user or group	setuid(2)
idcok () - enable or disable use of hardware insert- and delete-character features	idcok(3X)
ident - identify files in RCS	ident(1)
IDENT protocol server, TCP/IP	identd(1M)
identd - TCP/IP IDENT protocol server	identd(1M)
identification file, /etc/issue	issue(4)
identification information; get SCCS	what(1)
identification program; terminal	ttytype(1)
identifier, create interprocess communication	ftok(3C)
identifier, get for the current host	gethostid(2)
identifier, maximum number of System V IPC semaphores per	semmsl(5)
identifiers in the system, number of System V shared memory segment	shmmni(5)
identifiers, compare two thread identifiers	pthread_equal(3T)
identifiers, number of System V IPC system-wide semaphore	semmni(5)
identify files in RCS	ident(1)
identify the user of a particular TCP connection	idlookup(1)
identity parameters: hostname, date/time, root password, and networking; set system initial	set_parms(1M)
idisk - create partitions for disks on an Integrity system	idisk(1M)
idlok () - terminal output control functions	clearok(3X)
idlookup - identify the user of a particular TCP connection	idlookup(1)
IDs allowed, maximum number of system-wide System V IPC message queues	msgmni(5)
IDs and names; print user and group	id(1)
IDs of user processes from /etc/services.window ; extract window	getmemwindow(1M)
IDs, locality domain, determined	pthread_processor_bind_np(3T)
IDs, processor, determined	pthread_processor_bind_np(3T)
IDs, set real and effective user IDs	setreuid(2)
IDs; get real user, effective user, real group, and effective group	getuid(2)
IDs; set effective user and group	seteuid(2)
IDs; sets the real and effective group	setregid(2)
ied - input editor and command history for interactive programs	ied(1)
if - execute command if expression evaluates true	csh(1)
if - execute command if previous command returns exit status 0	ksh(1)
if - execute command if previous command returns exit status 0	sh-posix(1)
if values do not fit in fields; causes uname () system function to return [EOVERFLOW]	uname_eoverflow(5)
if_freenameindex () - functions that map between an interface name and index value	if_nameindex(3N)
if_indextoname () - functions that map between an interface name and index value	if_nameindex(3N)
if_nameindex () - functions that map between an interface name and index value	if_nameindex(3N)
if_nameindex () - functions that map between an interface name and index value	if_nameindex(3N)
ifconfig - configure network interface parameters	ifconfig(1M)
ignore signal	sigset(3C)
ilogb () - radix-independent exponent function	ilogb(3M)
ilogbf () - radix-independent exponent function (float)	ilogb(3M)
ilogbl () - radix-independent exponent function (long double)	ilogb(3M)
ilogbq () - radix-independent exponent function (quad)	ilogb(3M)
ilogbw () - radix-independent exponent function (extended)	ilogb(3M)
image	glossary(9)

Description	Entry Name(Section)
image file; main memory	mem(7)
images of running processes; get core	gcore(1)
imaxabs () - return long integer absolute value	abs(3C)
imaxdiv () - intmax_t integer division and remainder	div(3C)
immediate terminal refresh, enable/disable	immedok(3X)
immedok () - enable or disable immediate terminal refresh	immedok(3X)
implementation for pam_acct_mgmt ; service provider	pam_sm_acct_mgmt(3)
implementation-specific constants	limits(5)
implementations; magic numbers for HP-UX	magic(4)
import an LVM volume group onto the system	vgimport(1M)
in-core state with its state on disk, synchronize a file's	fsync(2)
in_wch () - input a complex character and rendition from a window	in_wch(3X)
in_wchnstr () - input an array of complex characters and renditions from a window	in_wchnstr(3X)
in_wchstr () - input an array of complex characters and renditions from a window	in_wchstr(3X)
inch () - input a single-byte character and rendition from a window	inch(3X)
inchstr () - input an array of single-byte characters and renditions from a window	inchstr(3X)
inchstr () - input an array of single-byte characters and renditions from a window	inchstr(3X)
include and conditional instructions, process C language	cpp(1)
including aliases and paths; locate a program file	which(1)
incoming mailbox file, print out mail in the	prmail(1)
incoming messages from other users to terminal, deny or permit <i>write</i> (1)	msg(1)
incoming telnet sessions, specifies the number of telnet device files the kernel can support for	nstrtel(5)
incomplete executable, prepare for faster program start-up	fastbind(1)
increase data segment space allocation	brk(2)
increase mirrors for LVM logical volume	lvextend(1M)
increase space for LVM logical volume	lvextend(1M)
incremental file system dump (for backups)	dump(1M)
index () - BSD portability string routine	string(3C)
index value; functions that map between an interface name and	if_nameindex(3N)
index, generate permuted	ptx(1)
individual user host access file, ftpd	ftphosts(4)
inet - Internet address manipulation routines	inet(3N)
inet - Internet protocol family	inet(7F)
inet6_opt_append () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_opt_find () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_opt_finish () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_opt_get_next () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_opt_get_set_val () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_opt_get_val () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_opt_init () - IPv6 Hop-by-Hop and Destination options manipulation functions	inet6_opt_init(3N)
inet6_rth_add () - IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
inet6_rth_getaddr () - IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
inet6_rth_init () - IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
inet6_rth_reverse () - IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
inet6_rth_segments () - IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
inet6_rth_space () - IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
inet_addr () - Internet address manipulation routines	inet(3N)
inet_lnaof () - Internet address manipulation routines	inet(3N)
inet_makeaddr () - Internet address manipulation routines	inet(3N)
inet_netof () - Internet address manipulation routines	inet(3N)
inet_network () - Internet address manipulation routines	inet(3N)
inet_ntoa () - Internet address manipulation routines	inet(3N)
inet_ntoa_r () - Internet address manipulation routines	inet(3N)
inet_nton () - Internet address manipulation routines	inet6(3N)
inet_pton () - Internet address manipulation routines	inet6(3N)
inetd configuration file	inetd.conf(4)
inetd - Internet services daemon	inetd(1M)
inetd optional security file	inetd.sec(4)

Index

All Volumes

Description	Entry Name(Section)
inetd.conf - configuration file for inetd	inetd.conf(4)
inetd.sec - optional inetd security file	inetd.sec(4)
inetsvcs.conf configuration file for secure internet services	inetsvcs.conf(4)
inetsvcs.sec - enable or disable secure internet services	inetsvcs.sec(1M)
infinity, test for	isinf(3M)
info - diskless client configuration information file	info(4)
infocmp - compare or print out terminfo descriptions	infocmp(1M)
information about a hardware partitionable complex; display	parstatus(1)
information about EVM; provide	evminfo(1)
information about kernel tunable parameters; retrieve detailed	tuneinfo2(2)
information about NIS map; query NIS server for	yppoll(1M)
information about resource utilization, get	getrusage(2)
information about the Partition Command Line Interface; display	partition(5)
information about users on remote machines, return	rnusers(3N)
information and open file descriptors; displays process address	pmap(1)
information by keywords; find manpage	man(1)
information constants; language	langinfo(5)
information file, diskless client configuration	info(4)
information file, LVM physical volume group	lvmpvg(4)
information for a dynamically loaded kernel module; get	modstat(2)
information for an EVM connection; control	EvmConnControl(3)
information for ftp , rexec , and rexec() , login	netrc(4)
information from the kernel, an infrastructure for obtaining	pstat(2)
information proper; domain	dig(1M)
information in RBAC databases; noninteractive editing of role-related	roleadm(1M)
information in the privrun database; noninteractive editing of a command's authorization and privilege	cmdprivadm(1M)
information in the user database, /var/adm/userdb , read, write or delete	userdb_read(3)
information in the user database, /var/adm/userdb ; modify	userdbset(1M)
information on current terminal	cur_term(3X)
information on loaded module (program or shared library)	dlget(3C)
information on loaded module (program or shared library)	dlgetmodinfo(3C)
information on loaded module (program or shared library)	dlmodinfo(3C)
information on RCS files; print log messages and other	rlog(1)
information residing in the user database, /var/adm/userdb ; display	userdbget(1M)
information server, remote user	fingerd(1M)
Information Service client interface; Network	ypclnt(3C)
Information Service database; force propagation of Network	yppush(1M)
Information Service (NIS) server, binder, and transfer processes; Network	ypserv(1M)
Information Service server; bind to particular Network	ypset(1M)
Information System server or map master; list which host is Network	ypwhich(1)
information used by finger command; change user	chfn(1)
information, changes NIS	ypupdate(3C)
information, current user, look up	finger(1)
information, display software product	swlist(1M)
information, get Terminal Session Manager state	tsm.info(1)
information, introduction to HP-UX general	intro(9)
information, NLS, about native languages	nl_langinfo(3C)
information; change WU-FTPD group access file	privatepw(1)
information; get locale-specific (NLS)	locale(1)
information; get SCCS identification	what(1)
information; system paging space	swapinfo(1M)
information; user login	wtmps(4)
infrastructure for obtaining information from the kernel, an	pstat(2)
init	glossary(9)
init - process control initialization	init(1M)
init process; script for the	inittab(4)
init_color() - color manipulation functions	can_change_color(3X)
init_pair() - color manipulation functions	can_change_color(3X)
initgroups() - initialize group access list	initgroups(3C)
initial (soft) maximum number of file descriptors per process	maxfiles(5)

Description	Entry Name(Section)
initial system configuration plus DHCP support command	auto_parms(1M)
initial system loader	isl(1M)
initialisation functions for screen	initscr(3X)
initialization routine only once; call an	pthread_once(3T)
initialization, terminal-dependent	tset(1)
initialization; process control	init(1M)
initialize a thread attribute object	pthread_attr_init(3T)
initialize an unnamed semaphore	sem_init(2)
initialize an unwind environment	uwx_init(3X)
initialize disk or partition DDS tape	mediainit(1)
initialize group access list	initgroups(3C)
initialize or destroy a condition variable	pthread_cond_init(3T)
initialize or destroy a mutex	pthread_mutex_init(3T)
initialize or destroy a mutex attribute object	pthread_mutexattr_init(3T)
initialize or destroy a read-write lock	pthread_rwlock_init(3T)
initialize or destroy a read-write lock attribute object	pthread_rwlockattr_init(3T)
initialize or destroy a thread condition variable attributes object	pthread_condattr_init(3T)
initialize semaphore in mapped file or anonymous memory region	msem_init(2)
initialize system log file	syslog(3C)
initialize terminal based on terminal type	tset(1)
initialize the current context for self-unwinding	uwx_self_init_context(3X)
initialize, manipulate, and test signal sets	sigsetops(3C)
initiate a connection on a socket	connect(2)
initiates, configures, and stops Live Dump	livedump(1M)
initscr() - screen initialisation functions	initscr(3X)
initstate() , setstate() , random() , srandom() - generate a pseudorandom number	random(3M)
inittab - script for the init process	inittab(4)
innetgr() - get network group entry	getnetgrent(3C)
innstr() - input a multi-byte character string from a window	innstr(3X)
innwstr() - input a string of wide characters from a window	innwstr(3X)
inode	glossary(9)
inode generation numbers, install random	fsirand(1M)
inode number	glossary(9)
inode, clear	clri(1M)
input a complex character and rendition from a window	in_wch(3X)
input a multi-byte character string from a window	innstr(3X)
input a single-byte character and rendition from a window	inch(3X)
input a string of wide characters from a window	innwstr(3X)
input an array of complex characters and renditions from a window	in_wchnstr(3X)
input an array of single-byte characters and renditions from a window	inchnstr(3X)
input character, control delay mode	halfdelay(3X)
input conversion, formatted, to a varargs argument	vscanf(3S)
input conversion; formatted read from stream file or character string	scanf(3S)
input editor and command history for interactive programs	ied(1)
input mode control functions	cbreak(3X)
input queue, push a character onto	ungetch(3X)
input single line from user keyboard	line(1)
input stream, push character back into	ungetc(3S)
input stream, push wide character back into	ungetwc(3C)
input, control blocking on	notimeout(3X)
input, convert formatted, from a window	mvscanw(3X)
input, discard	flushinp(3X)
input/output functions for screen file	scr_dump(3X)
input/output to a stream file; buffered binary	fread(3S)
input/output, buffered, standard stream file package	stdio(3S)
input; read a line from standard	read(1)
inquiries, stream status	ferror(3S)
ins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
ins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
ins_wstr() - insert a wide-character string into a window	ins_wstr(3X)
insch() - insert a single-byte character and rendition into a window	insch(3X)

Index

All Volumes

Description	Entry Name(Section)
insdelln () - delete or insert lines into a window	insdelln (3X)
insert a complex character and rendition into a window	ins_wch (3X)
insert a multi-byte character into a window	insnstr (3X)
insert a single-byte character and rendition into a window	insch (3X)
insert a wide-character string into a window	ins_nwstr (3X)
insert and delay capability, for terminal	has_ic (3X)
insert calls to <i>catgets</i> (3C) based on <i>findstr</i> (1) output	insertmsg (1)
insert lines into a window	insertln (3X)
insert or delete lines into a window	insdelln (3X)
insert or remove an element in a queue	insque (3C)
insert- and delete-character features, hardware, enable or disable use of	idcok (3X)
insertln () - insert lines into a window	insertln (3X)
insertmsg - use <i>findstr</i> (1) output to insert calls to <i>catgets</i> (3C)	insertmsg (1)
insf - install special (device) files	insf (1M)
insnstr () - insert a multi-byte character into a window	insnstr (3X)
insque () - insert an element in a queue	insque (3C)
insstr () - insert a multi-byte character into a window	insstr (3X)
install automatic mount points	automount (1M)
install boot programs from disk	mkboot (1M)
install - install new commands	install (1M)
install Network Information Service databases, build and	ypinit (1M)
install new elm aliases for user or system	newalias (1)
install object files in binary directories	cpset (1M)
install random inode generation numbers	fsirand (1M)
install special (device) files	insf (1M)
install, monitor, create, distribute, and manage software	sd (5)
install, software products	swinstall (1M)
install, update or check the <i>/etc/shadow</i> file	pwconv (1M)
installation and bootstrap utility, HP-UX	hpux (1M)
installed software, configure, unconfigure, reconfigure	swconfig (1M)
instr () - input a multi-byte character string from a window	innstr (3X)
int, round to nearest functions	rint (3M)
intctl - manage the interrupt configuration of the system	intctl (1M)
integer absolute value, return	abs (3C)
integer data types	inttypes (5)
integer data types; fixed-size	inttypes (5)
integer division and remainder	div (3C)
integer to base-64 ASCII string, convert long	a64 (3C)
integer, convert string to	strtoimax (3C)
integer, convert wide character string to long	westoimax (3C)
integer, long, convert to string,	ltostr (3C)
integer; convert string to long	strtol (3C)
integrity check on an event; perform a data	EvmEventValidate (3)
integrity of data cached with CacheFS; check	fsck_cache (1M)
Integrity system; create partitions for disks on	idisk (1M)
Integrity system; maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on an	pa_maxssiz (5)
Integrity systems; assembler for	as_ia (1)
Integrity systems; change program's internal attributes on	chatr_ia (1)
Integrity systems; enable or disable option for system to dump memory using multiple dump units when a kernel panic occurs on	dump_concurrent_on (5)
Integrity systems; execution startup routines for	crto_ia (3)
Integrity systems; explicit load of shared libraries for	shl_load_ia (3X)
Integrity systems; get entries from name list on	nlist_ia (3C)
Integrity systems; link editor for	ld_ia (1)
Integrity systems; list dynamic dependencies of executable files or shared libraries on	ldd_ia (1)
Integrity systems; open a shared library on	dlopen_ia (3C)
Integrity systems; structure formats for	nlist_ia (4)
interacting with kernel I/O subsystem; interface for	libIO (3X)
interactive mail message processing system	mailx (1)
interactive programs; input editor and command history for	ied (1)

Description	Entry Name(Section)
interactively write (talk) to another user	write(1)
interactively; query name servers	nslookup(1)
interchange format; DOS	dosif(4)
interest factor, compound	compound(3M)
Interface description, Extensible Firmware	efi(4)
interface for interacting with kernel I/O subsystem	libIO(3X)
interface for stape and tape2 magnetic tape	mt(7)
interface for Version 6/PWB compatibility; terminal	sttyv6(7)
interface management command for btlan driver; network	nwmgr_btlan(1M)
interface management command for intl100 driver; network	nwmgr_intl100(1M)
interface management command for LAN and RDMA interfaces; network	nwmgr(1M)
interface management command for VLAN interface; network	nwmgr_vlan(1M)
interface name and index value; functions that map between an	if_nameindex(3N)
interface parameters, configure network	ifconfig(1M)
interface socket that MTAs use to connect to the filter; attempts to create the	smfi_opensocket(3N)
interface standard header file, data link provider	dlpi(4)
interface to terminfo database	del_curterm(3X)
interface to the TELNET protocol, user	telnet(1)
interface, block mode terminal	blmode(7)
interface, Centronics-compatible	cent(7)
interface, extended general terminal	termiox(7)
interface, terminal: system console interface special file	console(7)
Interface; ACPS Service Provider	acps_spi(3)
interface; controlling terminal	tty(7)
Interface; display information about the Partition Command Line	partition(5)
interface; general terminal	termio(7)
Interface; Generic Security Service Application Programming	gssapi(5)
interface; Network Information Service client	ypclnt(3C)
interface; system console	console(7)
Interfaces Configuration and Network Services Configuration tools of	
HP System Management Homepage (HP SMH); launch the Network	ncweb(1M)
interfaces for device drivers to interact with DLPI	dlpi_drv(4)
interfaces for medium changer device; SCSI	autochanger(7)
interfaces; get the compartment IDs associated with a network	cmpt_get_ifcid(3)
interfaces; network interface management command for LAN and RDMA	nwmgr(1M)
Interfaces; X/Open Networking	xopen_networking(7)
interleave factor	glossary(9)
interleaved paging and swapping; add swap space for	swapon(2)
Intermediate System to Intermediate System (IS-IS) routing daemon; the	isisd(1M)
internal attributes on Integrity systems; change program's	chatr_ia(1)
internal attributes on PA-RISC systems; change program's	chatr_pa(1)
internal attributes; change program's	chatr(1)
internal consistency of Authentication database; check	authchk(1M)
internal form, convert a printable name to	gss_import_name(3)
internal name allocated by a GSSAPI routine, free storage associated with	gss_release_name(3)
internal name, duplicate	gss_duplicate_name(3)
Internal Terminal Emulator (ITE)	glossary(9)
Internal Terminal Emulator (ITE), load keyboard mapping	itemap(1M)
Internal Terminal Emulator, number of scrollable lines used by the	scroll_lines(5)
internationalization	glossary(9)
Internet address manipulation routines	inet6(3N)
Internet address manipulation routines	inet(3N)
Internet Boot Protocol server	bootpd(1M)
Internet domain name server	named(1M)
Internet domain name server; configuration file for	named.conf(4)
Internet protocol family	inet(7F)
Internet Protocol, IP	IP(7P)
Internet Protocol, version 6, IP	IP6(7P)
Internet services daemon	inetd(1M)
internet services with Kerberos authentication and authorization; secure	sis(5)
Internet Transmission Control Protocol, TCP	TCP(7P)

Index

All Volumes

Description	Entry Name(Section)
Internet user datagram protocol	UDP(7P)
Internet user name directory service	whois(1)
Internet; send mail over the	sendmail(1M)
interpret ASA carriage control characters	asa(1)
interpreter/compiler for modest-sized programs	bs(1)
interpreter; command (shell) with C-like syntax	csh(1)
interprocess channel; create an	pipe(2)
interprocess communication facilities, report status	ipcs(1)
interprocess communication identifier, create	ftok(3C)
Interprocess communications	socket(7)
interrupt configuration of the system; manage the	intctl(1M)
interrupt context; limit for the percent of time a processor is allowed to spend in	intr_strobe_ics_pct(5)
interrupt functions, allowing signals to	siginterrupt(2)
interrupt signal	glossary(9)
interrupt, atomically release blocked signals and wait for	sigpause(3C)
interrupt, enable or disable flush	intrflush(3X)
interval in clock ticks per second, scheduling	timeslice(5)
interval timer, set	ualarm(2)
interval timer, set or get value of process	getitimer(2)
interval, suspend execution for a time	sleep(1)
interval, suspend execution for an interval	usleep(2)
intl100 driver; network interface management command for	nwmgr_intl100(1M)
intr_strobe_ics_pct - limit for the percent of time a processor is allowed to spend in interrupt context	intr_strobe_ics_pct(5)
intrflush() - enable or disable flush on interrupt	intrflush(3X)
intrinsic	glossary(9)
intro - introduction to command utilities and application programs	intro(1)
intro - introduction to device special files	intro(7)
intro - introduction to file formats	intro(4)
intro - introduction to HP-UX general information section	intro(9)
intro - introduction to miscellany	intro(5)
intro - introduction to subroutines and libraries	intro(3C)
intro - introduction to system maintenance commands and application programs	intro(1M)
introduction - HP-UX operating system and <i>HP-UX Reference</i>	introduction(9)
introduction to device special files	intro(7)
introduction to HFS access control lists (ACLs)	acl(5)
introduction to HP-UX	Introduction(9)
introduction to HP-UX Auditing System	audit(5)
introduction to HP-UX general information section	intro(9)
introduction to JFS access control lists (ACLs)	aclv(5)
introduction to manpages	Introduction(9)
introduction to miscellany	intro(5)
introduction to POSIX.1c threads	pthread(3T)
introduction to subroutines and libraries	intro(3C)
introduction to system calls	intro(2)
intrusion detection data source; enable	enable_ids(5)
inttypes - basic integer data types	inttypes(5)
inttypes - fixed-size integer data types	inttypes(5)
inv - make unprintable and non-ASCII characters in a file invisible	vis(1)
invoke another application with privileges after performing appropriate authorization checks and optionally reauthenticating the user	privrun(1M)
invoke KWDB, the source level kernel debugger and crash dump analyzer	kwdb(1M)
invokes the HP-UX Security Attributes Configuration tool	secweb(1M)
inwstr() - input a string of wide characters from a window	innwstr(3X)
io_block_to_char_dsfs() - interface for interacting with kernel I/O subsystem	libIO(3X)
io_block_to_raw() - interface for interacting with kernel I/O subsystem	libIO(3X)
io_char_to_block_dsfs() - interface for interacting with kernel I/O subsystem	libIO(3X)
io_dev_to_node() - interface for interacting with kernel I/O subsystem	libIO(3X)
io_dev_to_options() - interface for interacting with kernel I/O subsystem	libIO(3X)
io_end() - interface for interacting with kernel I/O subsystem	libIO(3X)

Description	Entry Name(Section)
<code>io_error()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_get_devs()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_get_legacy_mode()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_get_mapping()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_get_node_relation()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_hw_compare()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_hw_compare_ext()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_hw_path_to_node()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_hw_path_to_str()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_init()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_init_hw_path()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_is_hwpath_legacy()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_is_legacy_dev()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_is_legacy_token()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_is_option_set()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_legacy_to_new_dev()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_legacy_to_new_dsf()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_legacy_to_new_hwpath()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_mkdev()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_mkdev_ext()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_new_to_legacy_devs()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_new_to_legacy_dsfs()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_new_to_legacy_hwpath()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_node_to_hw_path()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_query()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_query_array()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_query_batch()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_raw_to_block()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_redirect_dsf</code> - redirect the persistent device special file from one device to a different device	io_redirect_dsf(1M)
<code>io_search()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_search_array()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_search_array_batch()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_str_to_hw_path()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>io_strerror()</code> - interface for interacting with kernel I/O subsystem	libIO(3X)
<code>iobind</code> - bind a driver to a device	iobind(1M)
<code>ioconfig</code> entry format	ioconfig(4)
<code>ioconfig</code> - <code>ioconfig</code> entry format	ioconfig(4)
<code>ioctl</code> commands: STREAMS	streamio(7)
<code>ioctl()</code> - control character device special file	ioctl(2)
<code>ioctl</code> - generic device control commands	ioctl(5)
<code>ioctl()</code> - STREAMS enhancements to standard system calls	stream(2)
<code>ioctl</code> : STREAMS module for converting	timod(7)
<code>iofind</code> - search for or replace legacy device special files or hardware paths	iobind(1M)
<code>ioinit</code> - test and maintain consistency between the kernel I/O data structures and the <code>ioconfig</code> files	ioinit(1M)
<code>iomap</code> - physical memory address mapping	iomap(7)
<code>ioscan</code> - scan the I/O system	ioscan(1M)
<code>iostat</code> - report I/O statistics	iostat(1)
IOT fault, generate an	abort(3C)
IP - Internet Protocol	IP(7P)
IP Multicast - Internet Protocol	IP(7P)
<code>IP</code> multicast routing daemon	mrouted(1M)
<code>ip6</code> - Internet Protocol Version 6	IPv6(7P)
<code>IPC</code> hashed spinlock pool size, System V	sysv_hash_locks(5)
<code>IPC</code> message space resource map (OBSOLETE); number of entries in the System V	msgmap(5)
<code>IPC</code> messages at boot time (OBSOLETE); enable or disable System V	msg(5)
<code>IPC</code> semaphore, maximum value of any single System V	semvmx(5)
<code>IPC</code> semaphores at boot time, enable or disable System V	sema(5)
<code>IPC</code> semaphores per identifier, maximum number of System V	semmsl(5)
<code>IPC</code> <code>semop()</code> call, maximum cumulative value changes per System V	semaem(5)
<code>IPC</code> subsystem, get information about the System V	pstat(2)

Index

All Volumes

Description	Entry Name(Section)
IPC system-wide semaphore identifiers, number of System V	semnni(5)
IPC system-wide semaphore undo structures, number of System V	semnu(5)
IPC undo entries per process, maximum number of System V	semume(5)
ipcrm - remove a message queue, semaphore set, or shared memory identifier	ipcrm(1)
ipcs - report status of interprocess communication facilities	ipcs(1)
IPMI watchdog timer expires; sets action taken if	ipmi_watchdog_action(5)
ipmi_watchdog_action - sets action taken if IPMI watchdog timer expires	ipmi_watchdog_action(5)
IPv6 Hop-by-Hop and Destination options manipulation functions	net6_opt_init(3N)
IPv6 - Internet Protocol Version 6	IPv6(7P)
ip6 - Internet Protocol Version 6	IPv6(7P)
IPv6 Neighbor Discovery cache display and control	ndp(1M)
IPv6 Routing header options manipulation functions	inet6_rth_space(3N)
IPv6, Dynamic Host Configuration Protocol Server daemon	dhcpcv6d(1M)
IPv6, Router Advertisement daemon for	rtradvd(1M)
IPv6; BGP routing daemon for	bgpd(1M)
IPv6; RIPng routing daemon for	ripngd(1M)
IPv6; Route Administration Manager Daemon for	ramd(1M)
IS-IS routing daemon; the Intermediate System to Intermediate System	isisd(1M)
is_linetouched() - window refresh control functions	is_linetouched(3X)
is_wintouched() - window refresh control functions	is_linetouched(3X)
isalnum() - character is alphanumeric	ctype(3C)
isalpha() - character is alpha	ctype(3C)
isascii() - character is 7-bit ASCII code	ctype(3C)
isastream() - determine if file descriptor refers to STREAMS device or STREAMS-based pipe	isastream(3C)
isatty() - find name of a terminal	ttyname(3C)
isblank() - character is a blank	ctype(3C)
iscntrl() - character is a control character	ctype(3C)
iscomsec - check if system has been converted to a trusted system	iscomsec(2)
isdigit() - character is a digit	ctype(3C)
isendwin() - determine whether a screen has been refreshed	isendwin(3X)
isfinite() - floating-point finiteness macro	isfinite(3M)
isgraph() - character is a visible character	ctype(3C)
isgreater() - floating-point quiet comparison macro (>)	isgreater(3M)
isgreaterequal() - floating-point quiet comparison macro (>=)	isgreaterequal(3M)
isinf() - test for infinity	isinf(3M)
isisd - the Intermediate System to Intermediate System (IS-IS) routing daemon	isisd(1M)
isl - initial system loader	isl(1M)
isless() - floating-point quiet comparison macro (<)	isless(3M)
islessequal() - floating-point quiet comparison macro (<=)	islessequal(3M)
islessgreater() - floating-point quiet comparison macro (<>)	islessgreater(3M)
islower() - character is lowercase	ctype(3C)
isnan() - test for NaN	isnan(3M)
isnormal() - test for normal value	isnormal(3M)
isprint() - character is a printing character	ctype(3C)
ispunct() - character is punctuation	ctype(3C)
isspace() - character is whitespace	ctype(3C)
issue - /etc/issue identification file	issue(4)
issue a shell command	system(3S)
isunordered() - floating-point comparison macro (unordered)	isunordered(3M)
isupper() - character is uppercase	ctype(3C)
iswalnum() - character is alphanumeric	wctype(3C)
iswalpha() - character is alpha	wctype(3C)
iswblank() - character is a blank	wctype(3C)
iswcntrl() - character is a control character	wctype(3C)
iswctype() - character has property defined by prop	wctype(3C)
iswdigit() - character is a digit	wctype(3C)
iswgraph() - character is a visible character	wctype(3C)
iswlower() - character is lowercase	wctype(3C)
iswprint() - character is a printing character	wctype(3C)
iswpunct() - character is punctuation	wctype(3C)
iswspace() - character is whitespace	wctype(3C)

Description	Entry Name(Section)
iswupper() - character is uppercase	wctype(3C)
iswxdigit() - character is a hexadecimal digit	wctype(3C)
isxdigit() - character is a hexadecimal digit	ctype(3C)
Itanium-based systems running HP-UX; emulate PA-RISC HP-UX applications on	Aries(5)
Itanium-based systems, HP-UX bootstrap for	hpux.efi(1M)
ITE (Internal Terminal Emulator)	glossary(9)
itemap - load a keyboard mapping into the Internal Terminal Emulator	itemap(1M)
items; create and manipulate event	EvmItemGet(3)
j0() , j1() , jn() - Bessel functions of the first kind	j0(3M)
j0f() , j1f() , jnf() - Bessel functions of the first kind (float)	j0(3M)
j1() - Bessel function	j0(3M)
JFS access control lists (ACLs); introduction to	aclv(5)
JFS File Systems only; set a file's Access Control List (ACL) information;	acl(2)
jn() - Bessel function	j0(3M)
job control	glossary(9)
job control, set process group ID for	setpgid(2)
job control, uucp status inquiry and	uustat(1)
job execution daemon	cron(1M)
job file for at ; prototype	proto(4)
jobs - list active jobs	csch(1)
jobs - list active jobs	ksh(1)
jobs - list active jobs	sh-posix(1)
join corresponding lines of several files or subsequent lines of one file	paste(1)
join - relational database operator	join(1)
jukebox device drivers, SCSI	autochanger(7)
justify lines left or right for NLS printing	nljust(1)
kclog - manage kernel configuration log file	kclog(1M)
kcmodule - manage kernel modules and subsystems	kcmodule(1M)
kconfig - introduction to kernel configuration commands	kconfig(5)
kconfig - manage kernel configurations	kconfig(1M)
kcpath - print kernel configuration pathnames	kcpath(1M)
kcctl - manage kernel tunable parameters	kcctl(1M)
kcweb - starts the HP-UX kernel configuration tool (a Web interface)	kcweb(1M)
kdestroy - destroy Kerberos tickets	kdestroy(1)
keep track of remotely mounted file systems	mount(3N)
Kerberos authentication and authorization; secure internet services with	sis(5)
Kerberos client libraries	libkrb5(3)
Kerberos configuration file	krb5.conf(4)
Kerberos - introduction to the Kerberos system	Kerberos(5)
Kerberos keytab file maintenance utility	ktutil(1)
Kerberos password; change a user's	kpaswd(1)
Kerberos principals; print key version numbers of	kvno(1)
Kerberos system; introduction	Kerberos(5)
Kerberos ticket-granting ticket; obtain and cache the	kinit(1)
Kerberos tickets; destroy	kdestroy(1)
Kerberos tickets; list cached	klist(1)
kermit - C-Kermit 8.0 communications software for serial and network connections: modem dialing, file transfer and management, terminal connections, character-set translation, numeric and alpha paging, and script programming	kermit(1)
kernel	glossary(9)
kernel can support for incoming telnet sessions, specifies the number of telnet device files the	nstrtel(5)
kernel configuration commands; introduction	kconfig(5)
kernel configuration log file; manage	kclog(1M)
kernel configuration pathnames; print	kcpath(1M)
kernel configuration tool (a Web interface); starts the HP-UX	kcweb(1M)
kernel configurations; manage	kconfig(1M)
kernel debugger and crash dump analyzer; invoke KWDB, the source level	kwdb(1M)
kernel definition, display system	sysdef(1M)
kernel dynamic memory allocation map, number of entries in a	nsysmap(5)
Kernel Event Manager (KEVM) subsystem attributes	sys_attr kevm(5)
kernel I/O subsystem; interface for interacting with	libIO(3X)

Index

All Volumes

Description	Entry Name(Section)
kernel logging, control	kl(1M)
kernel memory pages are dumped when a kernel panic occurs, defines which classes of	alwaysdump(5)
kernel memory pages are not dumped when a kernel panic occurs, defines which classes of	dontdump(5)
kernel memory, based on symbol name; perform I/O on	kmem(7)
kernel module, unload on demand	moduload(2)
kernel module; get information for a dynamically loaded	modstat(2)
kernel modules, change global search path for dynamically loadable	modpath(2)
kernel modules, load on demand	modload(2)
kernel modules; manage	kcmodule(1M)
kernel packet forwarding database	route(7P)
kernel panic occurs on Integrity systems; enable or disable option for system to dump memory using multiple dump units when a	dump_concurrent_on(5)
kernel panic occurs, defines which classes of kernel memory pages are dumped when a	alwaysdump(5)
kernel panic occurs; selects whether the system dumps memory pages compressed or uncompressed when a	dump_compress_on(5)
kernel registry services daemon	krsd(1M)
kernel registry services data to disk; flush	krs_flush(1M)
kernel registry services, KRS	krs(5)
kernel RPC; generates and validates GSS-API tokens for	gssd(1M)
kernel space in a lightweight manner; Mercury Library Interfaces to transfer data between user and	hg(3)
kernel statistics server	rstatd(1M)
kernel subsystems; manage	kcmodule(1M)
kernel symbol; get information for a global	getksym(2)
kernel system file, create	create_sysfile(1M)
kernel tunable parameter, get value	gettune(2)
kernel tunable parameter; set the value of a	setttune(2)
kernel tunable parameters in a transaction; sets the values of	setttune_txn(2)
kernel tunable parameters; manage	kctune(1M)
kernel tunable parameters; retrieve detailed information about	tuneinfo2(2)
kernel, an infrastructure for obtaining information from the	pstat(2)
kernel; remote, get performance data from	rstat(3N)
KEVM (Kernel Event Manager) subsystem attributes	sys_attrs_kevm(5)
key codes from a terminal; get an array of wide characters and function	getn_wstr(3X)
key generation tool for DNSSEC	dnssec-keygen(1)
key generation tool; rndc	rndc-confgen(1)
key server, storing private encryption keys	keyserv(1M)
key version numbers of Kerberos principals; print	kvno(1)
key, change user's secure RPC key	chkey(1)
key, decrypt and store secret	keylogin(1)
key, delete secret key stored with keyserv	keylogout(1)
key, generate a DES encryption	makekey(1)
key; get name of	keyname(3X)
key_decryptsession() - library routines for secure remote procedure calls	secure_rpc(3N)
key_encryptsession() - library routines for secure remote procedure calls	secure_rpc(3N)
key_gendes() - library routines for secure remote procedure calls	secure_rpc(3N)
key_name() - get name of key	keyname(3X)
key_secretkey_is_set() - library routines for secure remote procedure calls	secure_rpc(3N)
key_setsecret() - library routines for secure remote procedure calls	secure_rpc(3N)
keyboard driver; HP-HIL mapped	hilkbd(7)
keyboard mapping, loading into the Internal Terminal Emulator	itemap(1M)
keyboard, how to obtain control characters from	ascii(5)
keyboard, PS/2 device driver	ps2(7)
keyboard/display data order, convert file	forder(1)
keyenvoy(1M) - talk to the keyserv process	keyenvoy(1M)
keylogin - decrypt and store secret key	keylogin(1)
keylogout - delete secret key stored with keyserv	keylogout(1)
keyname() - get name of key	keyname(3X)
keypad() - enable/disable abbreviation of function keys	keypad(3X)
keys file format; PPP encryption	ppp.Keys(4)

Description	Entry Name(Section)
keys in Network Information Service map, print the values of selected	ypmatch(1)
keyserv process, talk to the	keyenvoy(1M)
keyserv - server for storing private encryption keys	keyserv(1M)
keyset signing tool for DNSSEC	dnssec-signkey(1)
keysh - context-sensitive softkey shell	keysh(1)
keysh softkey file format	softkeys(4)
keytab file maintenance utility; Kerberos	ktutil(1)
keywords; find manpage information by	man(1)
kill a file or directory	rm(1)
kill line character	erasewchar(3X)
kill processes based on process name and attributes	pgrep(1)
kill () - send signal to a process or a group of processes	kill(2)
kill - send signal to process; terminate process	kill(1)
kill - send termination or specified signal to a process	csh(1)
kill () system call, 4.2 BSD-compatible	bsdproc(3C)
kill - terminate job or process	ksh(1)
kill - terminate job or process	sh-posix(1)
killall - kill all active processes	killall(1M)
killchar () - single-byte line kill	erasechar(3X)
killpg () - 4.2 BSD-compatible process control facilities	killpg(2)
kills the sendmail daemon	killsm(1M)
killsm - kills the sendmail daemon	killsm(1M)
killwchar () - current line kill character	erasewchar(3X)
kinit - obtain and cache the Kerberos ticket-granting ticket	kinit(1)
klist - list cached Kerberos tickets	klist(1)
kmem - perform I/O on kernel memory, based on symbol name	kmem(7)
known systems; list uucp names of	uucp(1)
kpasswd - change a user's Kerberos password	kpasswd(1)
krb5.conf - Kerberos configuration file	krb5.conf(4)
krs - kernel registry services, KRS	krs(5)
KRS; kernel registry services,	krs(5)
krs_flush - flush kernel registry services data to disk	krs_flush(1M)
krstd - kernel registry services daemon	krstd(1M)
ksh - Korn shell command programming language	ksh(1)
ksi_alloc_max - system-wide limit of queued signals that can be allocated	ksi_alloc_max(5)
ksi_send_max - limit on number of queued signals per process	ksi_send_max(5)
ktutil - Kerberos keytab file maintenance utility	ktutil(1)
kvno - print key version numbers of Kerberos principals	kvno(1)
kwdb - invoke KWDB, the source level kernel debugger and crash dump analyzer	kwdb(1M)
KWDB, the source level kernel debugger and crash dump analyzer; invoke	kwdb(1M)
1 - list contents of directories	ls(1)
164a () - convert long integer to base-64 value ASCII string	a64l(3C)
164a_r () - convert between long integer and base-64 ASCII string	a64l(3C)
label checking; copy a file system with	volcopy(1M)
label checking; copy HFS file system with	volcopy_hfs(1M)
label, define for formatting routines	setlabel(3)
label, soft, functions	slk_attroff(3X)
labelit - copy a file system with label checking	volcopy(1M)
labelit - copy an HFS file system with label checking	volcopy_hfs(1M)
labelit - copy file systems with label checking	volcopy_hfs(1M)
labs () - return long integer absolute value	abs(3C)
LAN administration	lanadmin(1M)
LAN and RDMA interfaces; network interface management command for	nwmgr(1M)
LAN connectivity, verify with link-level loopback	linkloop(1M)
LAN device configuration and status, display	lanscan(1M)
lan - network I/O card access information	lan(7)
LAN, log in on a remote system over	vt(1)
lanadmin - local area network administration program	lanadmin(1M)
lanadmin - virtual LANs (VLANs)	lanadmin_vlan(1M)
lanadmin_vlan - virtual LANs (VLANs)	lanadmin_vlan(1M)
LANG	glossary(9)

Index

All Volumes

Description	Entry Name(Section)
lang - description of supported languages	lang(5)
langinfo - language information constants	langinfo(5)
language information constants	langinfo(5)
language macro processor	m4(1)
language on multi-language systems; configure system	geocustoms(1M)
language: arbitrary-precision arithmetic language	bc(1)
language: C language preprocessor	cpp(1)
language; pattern-directed scanning and processing	awk(1)
languages, description of supported	lang(5)
languages, NLS information about native (local)	nl_langinfo(3C)
LANs (VLANs); virtual	lanadmin_vlan(1M)
lanscan - display LAN device configuration and status	lanscan(1M)
Large File through a socket; send the contents of a	sendfile64(2)
large files, find differences between	bdiff(1)
large files, non-POSIX standard API interfaces to support	creat64(2)
large letters, make posters in	banner(1)
large strings; generate hashing encryption on	bigcrypt(3C)
last commands executed, show in reverse order	lastcomm(1)
last - indicate last logins of users and ttys	last(1)
last locations in program	end(3C)
last locations of allocated regions in program	end(3C)
last logins of users and ttys, indicate	last(1)
last part of a file, get lines from	tail(1)
lastb - indicate last bad logins of users and ttys	last(1)
lastcomm - show last commands executed in reverse order	lastcomm(1)
lastlogin - show last login date for each user	acctsh(1M)
launch policy, thread, setting, thread, setting	pthread_launch_policy(3T)
launch the Disks and File Systems tool of HP System Management Homepage (HP SMH)	fsweb(1M)
launch the Network Interfaces Configuration and Network Services Configuration tools of HP System Management Homepage (HP SMH)	ncweb(1M)
layer manager; shell	shl(1)
lc - list contents of directories	ls(1)
lckpddf () - control access to /etc/passwd and /etc/shadow file	lckpddf(3C)
LCPU attribute of the default processor set; dynamically enable or disable	lcpu_attr(5)
lcpu_attr - dynamically enable or disable LCPU attribute of the default processor set	lcpu_attr(5)
ld - link editor	ld(1)
ld - link editor for Integrity systems	ld_ia(1)
ld - link editor for PA-RISC systems	ld_pa(1)
ld_ia - link editor for Integrity systems	ld_ia(1)
ld_pa - link editor for PA-RISC systems	ld_pa(1)
LDAP client daemon process	ldapclientd(1M)
LDAP client daemon process; configuration file for the	ldapclientd.conf(4)
LDAP directory server; accessing name services from a	ldap-ux(5)
LDAP directory; simple add, modify, and delete entries in an	ldapentry(1)
LDAP-UX - accessing name services from a LDAP directory server	ldap-ux(5)
LDAP; authentication, account, session, and password management PAM modules for	pam_ldap(5)
ldapclientd - LDAP client daemon process	ldapclientd(1M)
ldapclientd.conf - configuration file for the LDAP client daemon process	ldapclientd.conf(4)
ldapentry - simple add, modify, and delete entries in an LDAP directory	ldapentry(1)
ldd - list dynamic dependencies of executable files or shared libraries	ldd(1)
ldd - list dynamic dependencies of executable files or shared libraries on Integrity systems	ldd_ia(1)
ldd - list dynamic dependencies of executable files or shared libraries on PA-RISC systems	ldd_pa(1)
ldd_ia - list dynamic dependencies of executable files or shared libraries on Integrity systems	ldd_ia(1)
ldd_pa - list dynamic dependencies of executable files or shared libraries on PA-RISC systems	ldd_pa(1)
ldecvt (), (_ldecvt ()) - convert long double to string	ldcvt(3C)
ldexp () - scale exponent of a floating-point number	ldexp(3M)
ldexpf () - scale exponent of a floating-point number (float)	ldexp(3M)

Description	Entry Name(Section)
ldexp1 () - scale exponent of a floating-point number (long double)	ldexp (3M)
ldexpq () - scale exponent of a floating-point number (quad)	ldexp (3M)
ldexpw () - scale exponent of a floating-point number (extended)	ldexp (3M)
ldfcvt (), (_ldfcvt ()) - convert long double to string	ldcv (3C)
ldgcvt (), (_ldgcvt ()) - convert long double to string	ldcv (3C)
ldiv () - long integer division and remainder	div (3C)
ldterm - STREAMS terminal line discipline module	ldterm (7)
ldterm; set and get EUC code widths for	eucset (1)
leave - remind you when you have to leave	leave (1)
leaveok () - terminal output control functions	clearok (3X)
LEDs (cell, cabinet and I/O chassis attention LEDs); flash/turn off attention	fruled (1)
left or right justify lines for NLS printing	njust (1)
left-to-right text character sequence in each line of a file, reverse the	rev (1)
legacy device special file	glossary (9)
legacy hardware path	glossary (9)
legal login shells, list of	shells (4)
legal user shells, get	getusershell (3C)
length for transmission, get maximum data	rpc_gss_max_data_length (3N)
length limited string of single-byte characters and renditions to a window, add	addchnstr (3X)
length of string; find	string (3C)
let authorized users edit files that are under access control	privedit (1M)
let - evaluate arithmetic expression	ksh (1)
let - evaluate arithmetic expression	sh-posix (1)
letter from argument vector; get option	getopt (3C)
letters, make posters in large	banner (1)
level for the Milter library for sendmail; sets the debugging	smfi_setdbg (3N)
lfind () - linear search and update	lsearch (3C)
lgamma () - log gamma function	lgamma (3M)
lgamma_r () - reentrant log gamma function	lgamma (3M)
lgammaf () - log gamma function (float)	lgamma (3M)
lgammaf_r () - reentrant log gamma function (float)	lgamma (3M)
lgammal () - log gamma function (long double)	lgamma (3M)
lgammal_r () - reentrant log gamma function (long double)	lgamma (3M)
lgammaq () - log gamma function (quad)	lgamma (3M)
lgammaq_r () - reentrant log gamma function (quad)	lgamma (3M)
lgammaw () - log gamma function (extended)	lgamma (3M)
lgammaw_r () - reentrant log gamma function (extended)	lgamma (3M)
libc administration command	libcadmin (1M)
libc interfaces	thread_safety (5)
libc; subset of functions from	libres (5)
libcadmin - libc administration command	libcadmin (1M)
libcom_err - Kerberos client libraries	libkrb5 (3)
libcom_err.sl - Kerberos client libraries	libkrb5 (3)
libcom_err.so - Kerberos client libraries	libkrb5 (3)
libcrash - crash dump access library	libcrash (5)
libcrash error or warning message, print	cr_perror (3)
libgres - subset of functions from libc	libgres (5)
libgen interfaces	thread_safety (5)
libgss () - shared library for GSSAPI (Generic Security Service)	libgss (4)
libIO - interface for interacting with kernel I/O subsystem	libIO (3X)
libk5crypto - Kerberos client libraries	libkrb5 (3)
libk5crypto.sl - Kerberos client libraries	libkrb5 (3)
libk5crypto.so - Kerberos client libraries	libkrb5 (3)
libkrb5 - Kerberos client libraries	libkrb5 (3)
libkrb5 - Kerberos client libraries	libkrb5 (3)
libkrb5.sl - Kerberos client libraries	libkrb5 (3)
libkrb5.so - Kerberos client libraries	libkrb5 (3)
libmilter event loop; passes control to the	smfi_main (3N)
libpthread interfaces	thread_safety (5)
libraries for Integrity systems; explicit load of shared	shl_load_ia (3X)
libraries for PA-RISC systems; explicit load of shared	shl_load_pa (3X)

Index

All Volumes

Description	Entry Name(Section)
libraries linked into each process, including shared objects explicitly attached using <code>dlopen()/shl_load()</code> ; list the dynamic	pldd(1)
libraries on Integrity systems; list dynamic dependencies of executable files or shared	ldd_ia(1)
libraries on PA-RISC systems; list dynamic dependencies of executable files or shared	ldd_pa(1)
libraries; explicit load of shared	shl_load(3X)
libraries; introduction to subroutines and	intro(3C)
libraries; Kerberos client	libkrb5(3)
libraries; list dynamic dependencies of executable files or shared	ldd(1)
library	glossary(9)
library and application versions; coordinate ELF	elf_version(3E)
library data structure; allocate and deallocate unwind	_UNW_createContextForSelf(3X)
library data structure; manipulate values in unwind	_UNW_currentContext(3X)
library data structure; query values in unwind	_UNW_getGR(3X)
library file, link editor and assembler	a.out(4)
library for sendmail; sets the debugging level for the Milter	smfi_setdbg(3N)
library on Integrity systems; open a shared	dlopen_ia(3C)
library routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
library routines for client side calls	rpc_clnt_calls(3N)
library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
library routines for creation and manipulation of CLIENT handles	rpc_clnt_create(3N)
library routines for dealing with creation and manipulation of CLIENT handles	rpc_clnt_create(3N)
library routines for external data representation	xdr(3N)
library routines for external data representation	xdr_admin(3N)
library routines for external data representation	xdr_complex(3N)
library routines for external data representation	xdr_simple(3N)
library routines for external data representation stream creation	xdr_create(3N)
library routines for registering servers, rpc	rpc_svc_reg(3N)
library routines for remote procedure calls	rpc(3N)
library routines for remote procedure calls, XDR	rpc_xdr(3N)
library routines for RPC bind service	rpcbind(3N)
library routines for RPC servers	rpc_svc_calls(3N)
library routines for RPC, obsolete	rpc_soc(3N)
library routines for secure remote procedure calls	secure_rpc(3N)
library routines for server side remote procedure call errors	rpc_svc_err(3N)
library routines for the creation of server handles, rpc	rpc_svc_create(3N)
library routines, RPCSEC_GSS security flavor	rpcsec_gss(3N)
library routines, SLP (Service Location Protocol)	libslp(3N)
library structure; allocate transport function	t_alloc(3)
library with explicit load address; open an HP 9000 64-bit shared	dlopen_pa(3C)
library, access control	hosts_access(3)
library, HP 3000-mode packed-decimal	hppac(3X)
library, list of pthread calls for which the stubs are provided in the C	pthread_stubs(5)
library, produce a trace back of the procedure call stack using the unwind	U_STACK_TRACE(3X)
library: crash dump access library	libcrash(5)
library: object file access library	elf(3E)
library; open a shared	dlopen(3C)
library; open an HP 9000 shared	dlopen_pa(3C)
Library; Unwind Express	uwx(3X)
libslp - SLP (Service Location Protocol) library routines	libslp(3N)
license level of operating system, display	uname(1)
LIF	glossary(9)
LIF directory, list contents of a	lifls(1)
LIF files, copy to or from	lifcp(1)
LIF files, list contents of a LIF directory	lifls(1)
LIF files, remove	lifrm(1)
LIF files, rename	lifrename(1)
LIF files; copy to or from	lifcp(1)
LIF files; write LIF volume header on file	lifninit(1)
lif - logical interchange format description	lif(4)
lifcp - copy to or from LIF files	lifcp(1)
lifninit - write LIF volume header on file	lifninit(1)

Description	Entry Name(Section)
lifs - list contents of a LIF directory	lifs(1)
lifrename - rename LIF files	lifrename(1)
lifr - remove a LIF file	lifr(1)
lightweight manner; Mercury Library Interfaces to transfer data between user and kernel space in a	hg(3)
lightweight resolver daemon	lwresd(1M)
lightweight synchronization mechanism	postwait(2)
limit - limit usage by current process	cs(1)
limit of queued signals that can be allocated, system-wide	ksi_alloc_max(5)
limit the maximum value for process IDs (PIDs)	process_id_max(5)
limit; percent of file cache that can be consumed by sequential accesses, per-file	fcache_seqlimit_file(5)
limits - implementation-specific constants	limits(5)
limits the number of processes allowed to exist simultaneously	nproc(5)
limits the number of threads allowed to run simultaneously	nkthread(5)
line	glossary(9)
line control functions, tty	tccontrol(3C)
line control; asynchronous serial modem	modem(7)
line discipline; set terminal type, modes, speed, and	getty(1M)
line from standard input; read a	read(1)
Line Interface; display information about the Partition Command	partition(5)
line kill character	erasewchar(3X)
line kill character, single-byte	erasechar(3X)
line number and symbol information, strip from an object file	strip(1)
line numbering filter	nl(1)
line printer	<i>see printer</i>
line printer daemon for LP requests from remote systems	rlpdaemon(1M)
line printer device files	lp(7)
line - read one line from user input	line(1)
line speed, datacomm, and terminal settings used by getty	gettydefs(4)
line update status functions	redrawwin(3X)
line, dedicated, reserve for a purpose	ripoffline(3X)
line, get X.25	getx25(1M)
line, single, input from user keyboard	line(1)
line-feeds, remove multiple from output	ssp(1)
line-feeds, reverse, and backspaces, remove from text	col(1)
line-oriented text editor	ed(1)
line-oriented text editor	ex(1)
line-oriented text editor; extended	ex(1)
line-printer or CRT output, format text file for	nroff(1)
linear table search with optional update	lsearch(3C)
lines common to two sorted files, reject/select	comm(1)
lines in a file, cut out (extract) selected fields of	cut(1)
lines in a file, report adjacent repeated	uniq(1)
lines in a file; count	wc(1)
LINES - number of lines on terminal screen	LINES(3X)
lines used by the Internal Terminal Emulator, number of scrollable	scroll_lines(5)
lines, and bytes or characters in a file; count words,	wc(1)
lines, delete or insert into a window	insdelln(3X)
lines, draw from complex characters and renditions	hline_set(3X)
lines, draw from single-byte characters and renditions	hline(3X)
lines, insert into a window	insertln(3X)
lines, justify left or right for NLS printing	nljust(1)
lines, long, fold for finite-width output device	fold(1)
lines, merge corresponding lines of several files or subsequent lines of one file	paste(1)
lines, number of, on terminal screen	LINES(3X)
lines, reduce multiple adjacent blank to single blank line	ssp(1)
lines, remove preprocessor	unifdef(1)
link	glossary(9)
link() and unlink() system calls without error checking; execute	link(1M)
link count	glossary(9)

Index

All Volumes

Description	Entry Name(Section)
link directories using symbolic links	ln(1)
link editor	ld(1)
link editor and assembler output format	a.out(4)
link editor for Integrity systems	ld_ia(1)
link editor for PA-RISC systems	ld_pa(1)
link editor, find correct ordering of object code files for single pass	lorder(1)
link - execute link() system call without error checking	link(1M)
link existing file to new file name	ln(1)
link() - link to a file	link(2)
link provider interface, data	dlpi(7)
link status, get symbolic	lstat(2)
link to a file, make a symbolic	symlink(2)
link to root, primary swap, or dump volume, remove LVM logical volume	lvrmboot(1M)
link, file, symbolic (soft)	symlink(4)
link, symbolic, read value of	readlink(2)
link-level loopback to verify LAN connectivity	linkloop(1M)
linked into each process, including shared objects explicitly attached using dlopen()/shl_load(); list the dynamic libraries	pldd(1)
linker	glossary(9)
linking process, diagnostic information for dynamic linking	dlerrno(3C)
linking process, diagnostic information for dynamic linking	dlerror(3C)
linkloop - verify LAN connectivity with link-level loopback	linkloop(1M)
lio_listio() - start a list of asynchronous I/O operations	lio_listio(2)
list access control lists (ACLs) of files	lsacl(1)
list access control lists for files, JFS only	getacl(1)
list access privileges for group	getprivgrp(1)
list access rights to file(s)	getaccess(1)
list (ACL) structure, HFS file system only; convert string form to access control	strtoacl(3C)
list cached Kerberos tickets	klist(1)
list contents of a LIF directory	lifs(1)
list contents of directories	ls(1)
list current system users	who(1)
list device drivers in the system	lsdev(1M)
list dynamic dependencies of executable files or shared libraries	ldd(1)
list dynamic dependencies of executable files or shared libraries on Integrity systems	ldd_ia(1)
list dynamic dependencies of executable files or shared libraries on PA-RISC systems	ldd_pa(1)
list EFI file information or contents of an EFI directory	efi_ls(1M)
list entries in sendmail mail queue	sendmail(1M)
list file names and statistics for file system	ff(1M)
list file names and statistics for HFS file system	ff_hfs(1M)
list first few lines in a file	head(1)
list gsscred table entries; add, remove and	gsscred(1M)
list lines from last part of a file	tail(1)
list NFS security modes	nfssec.conf(4)
list of allowed login shells	shells(4)
list of home directory names	usermod(4)
list of pthread calls for which the stubs are provided in the C library	pthread_stubs(5)
list path name corresponding to i-number	ff_hfs(1M)
list processes using a file or file structure	fuser(1M)
list spooled uucp transactions grouped by transaction	uuls(1M)
list the dynamic libraries linked into each process, including shared objects explicitly attached using dlopen()/shl_load()	pldd(1)
list users currently on the system	users(1)
list uucp names of known systems	uucp(1)
list which host is Network Information System server or map master	ypwhich(1)
list, get group access	getgroups(2)
list, initialize group access	initgroups(3C)
list; get entries from name	nlist(3C)
listen backlog value of the filter, for sendmail; sets the	smfi_setbacklog(3N)
listen for connections on a socket	listen(2)
listen() - listen for connections on a socket	listen(2)

Description	Entry Name(Section)
lists (ACLs); introduction to HFS access control	acl(5)
lists (ACLs); introduction to JFS access control	aclv(5)
lists and execute command; construct argument	xargs(1)
lists archive files; copies files and directory hierarchies; extracts, writes, and	pax(1)
lists different categories of events	evweb_list(1)
lists owners of outgoing network connections	owners(1M)
listusers - display user login data	listusers(1)
Live Dump, initiates, configures, and stops	livedump(1M)
livedump - a feature that saves operating system state to the file system for debugging purposes.	livedump(5)
livedump - initiates, configures, and stops Live Dump	livedump(1M)
ls - list contents of directories	ls(1)
llabs () - return long long integer absolute value	abs(3C)
lldiv () - long long integer division and remainder	div(3C)
llrint () - round to nearest long long function	llrint(3M)
llrintf () - round to nearest long long function (float)	llrint(3M)
llrintl () - round to nearest long long function (long double)	llrint(3M)
llrintq () - round to nearest long long function (quad)	llrint(3M)
llrintw () - round to nearest long long function (extended)	llrint(3M)
llround () - round to long long function	llround(3M)
llroundf () - round to long long function (float)	llround(3M)
llroundl () - round to long long function (long double)	llround(3M)
llroundq () - round to long long function (quad)	llround(3M)
llroundw () - round to long long function (extended)	llround(3M)
ln - link files and directories	ln(1)
load a kernel configuration from a system file	mk kernel(1M)
load address; open an HP 9000 64-bit shared library with explicit	dlopen_pa(3C)
load exponent of a floating-point number	ldexp(3M)
load kernel modules on demand	modload(2)
load module information for current context	uwx_get_module_info(3X)
load module, retrieve name	dlgetname(3C)
load of shared libraries for Integrity systems; explicit	shl_load_ia(3X)
load of shared libraries for PA-RISC systems; explicit	shl_load_pa(3X)
load of shared libraries; explicit	shl_load(3X)
load operating system	boot(1M)
loaded module (program or shared library)	dlget(3C)
loaded module (program or shared library)	dlgetmodinfo(3C)
loaded module (program or shared library)	dlmodinfo(3C)
loader, dynamic	dld.sl(5)
loader, dynamic	dld.so(5)
loader, initial system	isl(1M)
local area network administration	lanadmin(1M)
local area network; virtual	VLAN(7)
local communication domain protocol	UNIX(7P)
local customs	glossary(9)
local host, authorizing access from remote hosts and users	hosts.equiv(4)
local machines (RPC version), show host status of	rup(1)
local machines, show status of	ruptime(1)
local machines, show who is logged in on	rwho(1)
local (native) languages, NLS information about	nl_langinfo(3C)
local network machines, determine who is logged in on	rusers(1)
local network packet routing; system support for	routing(7)
local node; transfer NIS database from server to	ypxfr(1M)
local or across network; restore file system incrementally,	restore(1M)
local resource unavailable for mounting by remote systems; make	unshare(1M)
local time, difference between Universal (Greenwich mean) and	timezone(5)
locale definition file; format and semantics	localedef(4)
locale environment; generate	localedef(1M)
locale - get locale-specific (NLS) information	locale(1)
locale of a program, get or set the	setlocale(3C)
locale-specific (NLS) information; get	locale(1)
localeconv () - query numeric formatting conventions of current locale	localeconv(3C)

Index

All Volumes

Description	Entry Name(Section)
localedef - format and semantics of locale definition file	localedef(4)
localedef - generate a locale environment	localedef(1M)
localedef scripts; symbolic translation file for	charmap(4)
locality domain IDs, determined	pthread_processor_bind_np(3T)
locality domain, bind threads to	pthread_processor_bind_np(3T)
locality domain, control, on which a specific process executes	mpsched(1)
locality domain, how many available	pthread_processor_bind_np(3T)
localization	glossary(9)
localtime() , localtime_r() - convert date and time to local timezone	ctime(3C)
locate a program file including aliases and paths	which(1)
locate source, binary, and/or manual files for program	whereis(1)
location functions, window cursor	move(3X)
location of byte in memory; find	memory(3C)
Location Protocol Daemon; Service	slpd(1M)
Location Protocol (SLP) error codes; Service	SLPError(3N)
locations beyond allocated program regions; first	end(3C)
locations in program; last	end(3C)
lock a POSIX semaphore	sem_wait(2)
lock a POSIX semaphore without blocking	sem_wait(2)
lock a semaphore	msem_lock(2)
lock access to /etc/passwd and /etc/shadow files	lckpwwd(3C)
lock daemon; network	lockd(1M)
lock on an open file; apply or remove an advisory or enforced	flock(2)
lock or attempt to lock a read-write lock for reading	pthread_rwlock_rdlock(3T)
lock or attempt to lock a read-write lock for writing	pthread_rwlock_wrlock(3T)
lock or try to lock a mutex	pthread_mutex_lock(3T)
lock process address space	mlockall(2)
lock process into memory after allocating data and stack space	datalock(3C)
lock process, text, or data in memory	plock(2)
lock - protect terminal from use by others	lock(1)
lock segment in memory	mlock(2)
lock terminal against use by others	lock(1)
lockable for request call-back POSIX asynchronous I/O operations; percentage of physical memory	aio_phymem_pct(5)
lockd - network lock daemon	lockd(1M)
lockdown tool; system	bastille(1M)
lockf() - provide semaphores and record locking on files	lockf(2)
lockf64() - non-POSIX standard API interfaces to support large files	creat64(2)
locking of streams within a multi-thread application, explicit	flockfile(3S)
locking on files, provide semaphores and record	lockf(2)
locks for the Directory Name Lookup Cache (DNLC); number of	dnlc_hash_locks(5)
locks held on behalf of an NFS client, clear	clear_locks(1M)
LOFS file system, mount	mount_lofs(1M)
log driver; receive error messages from the STREAMS log driver	strerr(1M)
log driver; STREAMS	strlog(7)
log file of uucp transactions; query	uucp(1)
log files, remove outdated STREAMS error log files	strclean(1M)
log gamma functions	lgamma(3M)
log in on another system over LAN	vt(1)
log in to system	login(1)
log messages and other information on RCS files; print	rlog(1)
log() - natural logarithm function	log(3M)
log system messages	syslogd(1M)
log, error, collect system diagnostic messages to form	dmesg(1M)
log, mail, displays the last part of	mtail(1M)
log10() - common logarithm function	log10(3M)
log10f() - common logarithm function (float)	log10(3M)
log10l() - common logarithm function (long double)	log10(3M)
log10q() - common logarithm function (quad)	log10(3M)
log10w() - common logarithm function (extended)	log10(3M)
log1p() - natural logarithm of one-plus-argument function	log1p(3M)

Description	Entry Name(Section)
log1pf () - natural logarithm of one-plus-argument function (float)	log1p (3M)
log1pl () - natural logarithm of one-plus-argument function (long double)	log1p (3M)
log1pq () - natural logarithm of one-plus-argument function (quad)	log1p (3M)
log1pw () - natural logarithm of one-plus-argument function (extended)	log1p (3M)
log2 () - logarithm base two function	log2 (3M)
log2f () - logarithm base two function (float)	log2 (3M)
log2l () - logarithm base two function (long double)	log2 (3M)
log2q () - logarithm base two function (quad)	log2 (3M)
log2w () - logarithm base two function (extended)	log2 (3M)
log; control system	syslog (3C)
logarithm base two functions	log2 (3M)
logb () - radix-independent exponent function	logb (3M)
logbf () - radix-independent exponent function (float)	logb (3M)
logbl () - radix-independent exponent function (long double)	logb (3M)
logbq () - radix-independent exponent function (quad)	logb (3M)
logbw () - radix-independent exponent function (extended)	logb (3M)
logf () - natural logarithm function (float)	log (3M)
logfile, FTP server logfile	xferlog (5)
logfile, viewing and saving SAM	samlog viewer (1)
logged in on local machines, show who is	rwho (1)
logged in on local network machines, determine who is	rusers (1)
logged in users' accounting information file	utmpx (4)
logger configuration file; EVM	evmlogger.conf (4)
logger - make entries in the system log	logger (1)
logger; Event Manager	evmlogger (1M)
logging and tracing administration manager; network	nettladm (1M)
logging and tracing binary files, format	netfmt (1M)
logging and tracing, configure subsystem database	nettlconf (1M)
logging and tracing; control network	nettl (1M)
logging configuration file; network tracing and	nettlgen.conf (4)
logging configuration file; NFS server	nfslog.conf (4)
logging daemon; nfs	nfslogd (1M)
Logical Interchange Format	glossary (9)
logical interchange format description	lif (4)
logical volume characteristics (LVM), change	lvchange (1M)
logical volume in LVM volume group, create	lvcreate (1M)
logical volume into two logical volumes; split mirrored LVM	lvsplit (1M)
logical volume (LVM), decrease physical extents allocated to	lvreduce (1M)
logical volume (LVM), stripe, increase space, increase mirrors	lvextend (1M)
Logical Volume Manager (LVM)	lvm (7)
Logical Volume Manager (LVM) control; check if disk volume is under	lvmchk (1M)
logical volume mirrors in LVM volume groups, synchronize stale	vgsync (1M)
logical volume to be root, boot, primary swap, or dump volume; prepare LVM	lvlnboot (1M)
logical volume, get information for a	pstat (2)
logical volume; merge two LVM logical volumes into one	lvmerge (1M)
logical volumes from LVM volume group, remove	lvremove (1M)
logical volumes (LVM), synchronize stale mirrors in	lvsync (1M)
logical volumes; split mirrored LVM logical volume into two	lvsplit (1M)
login	glossary (9)
login data; display system and user	logins (1M)
login data; display user	listusers (1)
login directory	login (1)
login directory	glossary (9)
login environment, shell script to set up user's	profile (4)
login from the system; delete a user	userdel (1M)
login information for ftp , rexec , and rexec ()	netrc (4)
login information; user	wtmps (4)
login name	login (1)
login name of the user, get character-string	cuserid (3S)
login name of user, obtain	logname (3C)
login name, change	su (1)

Index

All Volumes

Description	Entry Name(Section)
login name, get	logname(1)
login on the system; modify a user	usermod(1M)
login password and associated attributes; change	passwd(1)
login password in Network Information System (NIS), change	yppasswd(1)
login password, change	passwd(1)
login record format; user	utmp(4)
login server; remote	rlogind(1M)
login shell; change default	chsh(1)
login shells, list of allowed	shells(4)
login - sign on; start terminal session	login(1)
login - terminate login shell	csh(1)
login to remote system; test for successful	uucp(1)
login to the system; add a new user	useradd(1M)
login/logoff records, convert to per-session accounting records	acctcon(1M)
login; remote	rlogin(1)
logingroup file	login(1)
logingroup - group access and identification file, grp.h	group(4)
logins - display login data	logins(1M)
logins - display system and user login data	logins(1M)
logins of users and ttys, indicate last	last(1)
logl () - natural logarithm function (long double)	log(3M)
LOGNAME environment variable	login(1)
logname - get login name	logname(1)
logname () - return login name of user	logname(3C)
logout - terminate login shell	csh(1)
logq () - natural logarithm function (quad)	log(3M)
logw () - natural logarithm function (extended)	log(3M)
long double floating-point number to string, convert	ldcvt(3C)
long file names, convert an HFS file system to allow	convertfs(1M)
long files, find differences between	bdiff(1)
long integer to base-64 ASCII string, convert	a64l(3C)
long integer to string, convert	ltostr(3C)
long integer, convert wide character string to	wctoimax(3C)
long integer; convert wide character string to	westol(3C)
long lines, fold for finite-width output device	fold(1)
long user and group name enablement and display	lugadmin(1M)
longjmp () - restore stack environment after non-local goto	setjmp(3C)
longname () - get verbose description of current terminal	longname(3X)
look up current user information	finger(1)
looked-up names, get entries from system cache of recently	pstat(2)
Lookup Cache (DNLC); number of locks for the Directory Name	dnlc_hash_locks(5)
lookup sources	service.switch(1M)
lookup utility; DNS	host(1)
loop; passes control to the libmilter event	smfi_main(3N)
loopback, link-level, to verify LAN connectivity	linkloop(1M)
lorder - find ordering relation for an object code library	lorder(1)
lost+found directory for the fsck command; make a	mklost+found(1M)
lower-case, translate characters to	conv(3C)
lowercase, translate wide characters to	wconv(3C)
LP destinations; move requests between	lpsched(1M)
lp interface scripts; filters invoked by	lpfilter(1)
lp - line printer device files	lp(7)
lp - print requests on an LP printer	lp(1)
LP printer, allow or prevent queuing requests	accept(1M)
LP printer, print/alter/cancel requests	lp(1)
LP printers, enable/disable	enable(1)
LP request scheduler; start the	lpsched(1M)
LP request scheduler; stop the	lpsched(1M)
LP requests, cancel from spooling queue on remote system	rcancel(1M)
LP requests: daemon for LP requests from remote systems	rlpdaemon(1M)
LP requests: send LP request to a remote system	rlp(1M)

Description	Entry Name(Section)
LP spooler performance analysis information; display	lpana(1M)
LP spooler requests on a remote system; print status of	rlpstat(1M)
LP spooling system; configure the	lpadmin(1M)
LP subsystem; report status information of the	lpstat(1)
lpadmin - configure the LP spooling system	lpadmin(1M)
lpalt - alter requests on an LP printer	lp(1)
lpana - display LP spooler performance analysis information	lpana(1M)
lpfence - define the minimum priority for printing	lpsched(1M)
lpfilter - filters invoked by lp interface scripts	lpfilter(1)
lpmove - move requests between LP destinations	lpsched(1M)
lprpp - laserjet filter	lpfilter(1)
lpsched - start the LP request scheduler	lpsched(1M)
lpshut - stop the LP request scheduler	lpsched(1M)
lpstat - report status information of the LP subsystem	lpstat(1)
lrand48() , nrand48() - generate long-integer pseudo-random numbers	drand48(3C)
lrint() - round to nearest long int function	lrint(3M)
lrintf() - round to nearest long int function (float)	lrint(3M)
lrintl() - round to nearest long int function (long double)	lrint(3M)
lrintq() - round to nearest long int function (quad)	lrint(3M)
lrintw() - round to nearest long int function (extended)	lrint(3M)
lround() - round to long int function	lround(3M)
lroundf() - round to long int function (float)	lround(3M)
lroundl() - round to long int function (long double)	lround(3M)
lroundq() - round to long int function (quad)	lround(3M)
lroundw() - round to long int function (extended)	lround(3M)
ls - list contents of directories	ls(1)
lsacl - list access control lists (ACLs) of files	lsacl(1)
lsdev - list device drivers in the system	lsdev(1M)
lsearch() , lfind() - linear search and update	lsearch(3C)
lseek() - move read/write file pointer; seek	lseek(2)
lseek64() - non-POSIX standard API interfaces to support large files	creat64(2)
lsf - list contents of directories	ls(1)
lsr - list contents of directories	ls(1)
lssf - list a special (I/O device) file	lssf(1M)
lstat() - get symbolic link status	lstat(2)
lstat64() - non-POSIX standard API interfaces to support large files	creat64(2)
lsx - list contents of directories	ls(1)
lsync() - update disk	sync(2)
ltoa() - convert long integer to ASCII decimal	ltostr(3C)
ltoa_r() - convert long integer to ASCII decimal (MT-Safe)	ltostr(3C)
ltostr() - convert long integer to string	ltostr(3C)
ltostr_r() - convert long integer to string (MT-Safe)	ltostr(3C)
lugadmin - long user and group name enablement and display	lugadmin(1M)
LUN	glossary(9)
LUN hardware path	glossary(9)
lunpath hardware path	glossary(9)
lvchange - change LVM logical volume characteristics	lvchange(1M)
lvcreate - create logical volume in LVM volume group	lvcreate(1M)
lvdisplay - display information about LVM logical volumes	lvdisplay(1M)
lvextend - stripe, increase space, increase mirrors for LVM logical volume	lvextend(1M)
lvlnboot - prepare LVM logical volume to be root, boot, primary swap, or dump volume	lvlnboot(1M)
LVM logical volume characteristics, change	lvchange(1M)
LVM logical volume into two logical volumes; split mirrored	lvsplit(1M)
LVM logical volume link to root, primary swap, or dump volume, remove	lvrmboot(1M)
LVM (Logical Volume Manager)	lvm(7)
lvm - Logical Volume Manager (LVM)	lvm(7)
LVM logical volume to be root, boot, primary swap, or dump volume; prepare	lvlnboot(1M)
LVM logical volume, decrease physical extents allocated to	lvreduce(1M)
LVM logical volume, stripe, increase space, increase mirrors	lvextend(1M)
LVM logical volumes, synchronize stale mirrors in	lvsync(1M)
LVM logical volumes; display information about	lvdisplay(1M)

Index

All Volumes

Description	Entry Name(Section)
LVM logical volumes; merge two into one logical volume	lvmerge(1M)
LVM physical volume group information file	lvmpvg(4)
LVM physical volume to other physical volumes, move allocated physical extents from	pvmove(1M)
LVM volume group and its associated logical volumes; export	vgexport(1M)
LVM volume group availability; set	vgchange(1M)
LVM volume group configuration backup file, create or update	vgcfgbackup(1M)
LVM volume group definition, remove from the system	vgremove(1M)
LVM volume group onto the system; import	vgimport(1M)
LVM volume group, change characteristics and access path of physical volume in	pvchange(1M)
LVM volume group, check or repair a physical volume in	pvck(1M)
LVM volume group, create logical volume in	lvcreate(1M)
LVM volume group, extend by adding physical volumes	vgextend(1M)
LVM volume group, remove logical volumes from	lvremove(1M)
LVM volume group, remove physical volume	pvremove(1M)
LVM volume group; create	vgcreate(1M)
LVM volume group; create physical volume for use in	pvcreate(1M)
LVM volume group; display information about physical volumes in	pvdisplay(1M)
LVM volume group; handle physical volume size changes and modify configuration parameters of an existing	vgmodify(1M)
LVM volume group; remove physical volumes from	vgreduce(1M)
LVM volume groups, synchronize stale logical volume mirrors	vgsync(1M)
LVM volume groups; display information about	vgdisplay(1M)
LVM volume groups; scan physical volumes for	vgscan(1M)
LVM, split mirrored logical volume into two logical volumes	lvsplit(1M)
lvchk - check if disk volume is under HP Logical Volume Manager (LVM) control	lvchk(1M)
lvmerge - merge two LVM logical volumes into one logical volume	lvmerge(1M)
lvmpvg - LVM physical volume group information file	lvmpvg(4)
lvreduce - decrease physical extents allocated to LVM logical volume	lvreduce(1M)
lvremove - remove logical volumes from LVM volume group	lvremove(1M)
lvrmboot - remove LVM logical volume link to root, primary swap, or dump volume	lvrmboot(1M)
lvsplit - split mirrored LVM logical volume into two logical volumes	lvsplit(1M)
lvsync - synchronize stale mirrors in LVM logical volumes	lvsync(1M)
LWP in a process, get information for a thread or	pstat(2)
LWP in each process and core file; print a stack trace for each	pstack(1)
lwresd - lightweight resolver daemon	lwresd(1M)
m4 - macro processor	m4(1)
machid - provide truth value about processor type	machid(1)
machine identification, display	uname(1)
machine information, change	setuname(1M)
machine information; print	machinfo(1)
machine number, display	uname(1)
machine-dependent values	values(5)
machines (RPC version), show host status of local	rup(1)
machines, determine who is logged in on local network	rusers(1)
machines, return information about users on remote	rnusers(3N)
machines, show status of local	ruptime(1)
machines, show who is logged in on local	rwho(1)
machines, write to specified remote	rwall(3N)
machinfo - print machine information	machinfo(1)
macro package for formatting documents, MM	mm(5)
macro package for formatting manpages	man(5)
macro processor for C, Ratfor and other programming languages	m4(1)
macro; gets the value of a sendmail	smfi_getsymval(3N)
macros and functions, floating-point environment	fenv(5)
macros for handling device numbers, header file of	mknod(5)
macros for handling variable argument lists	varargs(5)
macros for handling variable argument lists	stdarg(5)
macros, check or print documents formatted using mm	mm(1)
macros; complex functions and	complex(5)
madvise () - advise system of process's expected paging behavior	madvise(2)
magic - magic numbers for HP-UX implementations	magic(4)

Description	Entry Name(Section)
magic number	glossary(9)
magic numbers for HP-UX implementations	magic(4)
magnetic tape dump and restore protocol module, remote	rmt(1M)
magnetic tape interface for stape, tape2 and	mt(7)
magnetic tape manipulating program	mt(1)
mail aliases file, rebuild database	newaliases(1M)
MAIL environment variable	login(1)
mail file	login(1)
mail folder, read mail from specified	readmail(1)
mail folders by subject and sender; summarize	mailfrom(1)
mail folders, summarize by subject and sender	mailfrom(1)
mail in the incoming mailbox file, print out	prmail(1)
mail interface, batch	fastmail(1)
mail log, displays the last part of	mtail(1M)
mail message processing system; interactive	mailx(1)
mail queue, printing	mailq(1)
mail queue; list entries in sendmail	sendmail(1M)
mail - send mail to users or read mail	mail(1)
mail traffic statistics, print	mailstats(1)
mail vacation response	vacation(1)
mail - who is mine from?	from(1)
mail, notify users of new	newmail(1)
mail, read from specified mail folder	readmail(1)
mail, screen-oriented interface	elm(1)
mail; send mail to users or read	mail(1)
mail; send over the Internet	sendmail(1M)
mailbox file, print out mail in the incoming	prmail(1)
mailboxes, notify users of new mail in	newmail(1)
mailer, convert binary file to ASCII for transmission by	uuencode(1)
mailer; encode/decode a binary file for transmission by	uuencode(1)
mailfrom - summarize mail folders by subject and sender	mailfrom(1)
mailq - prints the mail queue	mailq(1)
mailstats - print mail traffic statistics	mailstats(1)
mailx - interactive mail message processing system	mailx(1)
main memory allocator	malloc(3C)
main memory image file	mem(7)
maintain connection with the EVM daemon	EvmConnCheck(3)
maintain, update, and regenerate groups of programs	make(1)
maintainer, archive and library, for portable archives	ar(1)
maintenance utility; Kerberos keytab file	ktutil(1)
major and minor device pair: STREAMS driver	clone(7)
major number	glossary(9)
make a delta (change) to an SCCS file	delta(1)
make a directory	mkdir(1)
make a directory file	mkdir(2)
make a directory, special, or ordinary file	mknod(2)
make a FIFO file	mkfifo(3C)
make a file system (generic)	mkfs(1M)
make a lost+found directory for the fsck command	mklost+found(1M)
make a lost+found directory for the fsck command	mklost+found(1M)
make a makefile	mkmf(1)
make a name for a temporary file	mktemp(1)
make a Network Information System database	makedbm(1M)
make a new file system	newfs(1M)
make a new HFS file system	newfs_hfs(1M)
make a special (device) file	mkfsf(1M)
make a string pointer for ELF files	elf_strptr(3E)
make a symbolic link to a file	symlink(2)
make an EFI directory	efi_mkdir(1M)
make FIFO (named pipe) special files	mkfifo(1)
make file descriptor for ELF file	elf_begin(3E)

Index

All Volumes

Description	Entry Name(Section)
make local NFS file systems available for mounting by remote systems	share_nfs(1M)
make local NFS file systems unavailable for mounting by remote systems	unshare_nfs(1M)
make local resource available for mounting by remote systems	share(1M)
make local resource unavailable for mounting by remote systems	unshare(1M)
make - maintain, update, and regenerate groups of programs	make (1)
make posters in large letters	banner(1)
make typescript of terminal session	script(1)
make unprintable and non-ASCII characters in a file visible or invisible	vis(1)
makecontext () - manipulate user contexts; DEPRECATED	makecontext (2)
makedbm - make a Network Information System database	makedbm (1M)
makefile; make a	mkmf(1)
makekey - generate a DES encryption key	makekey (1)
makemap - creates database maps for sendmail	makemap (1M)
mallinfo () - display memory space usage	malloc (3C)
malloc () - allocate block of main memory	malloc (3C)
mallopt () - control memory space allocation	malloct (3C)
man command, fix manpages for faster viewing with	fixman (1M)
man - find manpage information by keywords; print out a manpage	man (1)
man - macros for formatting manpages	man (5)
manage a binary search tree	tsearch(3C)
manage and configure system swap space	swaptl(2)
manage hash search tables	hsearch(3C)
manage kernel configurations	kconfig(1M)
manage kernel modules and subsystems	kcmodule(1M)
manage kernel tunable parameters	kctune(1M)
manage options for a transport endpoint	t_optmgmt(3)
manage processor sets; create and	psrset(1M)
manage system database of automatically pushed STREAMS modules	autopush(1M)
manage the interrupt configuration of the system	intctl(1M)
manage the pathalias database, access and	uupath(1)
manage, monitor, create, distribute, and install software	sd(5)
management and diagnostic utility; SCSI	scsingr(1M)
management command for LAN and RDMA interfaces; network interface	nwmgr(1M)
Management) daemon; connection to the EVM (Event	EvmConnection(5)
Management) event filter; EVM (Event	EvmFilter(5)
management (EVM) callback function; event	EvmCallback(5)
management functions, for pad	newpad(3X)
Management Homepage (HP SMH); HP System	smh(1M)
management; signal	sigset(3C)
Manager filter file; Event	evmfilterfile(4)
Manager state information, get Terminal Session	tsm.info(1)
Manager template file; Event	evmtemplate(4)
manager, shell layer	shl(1)
Manager, Terminal Session	tsm(1)
manager; shell layer	shl(1)
manager; system administration	sam(1M)
manages file system stack templates	fstadm(2)
manipulate connect accounting records	fwtmp(1M)
manipulate crash dump data	crashutil(1M)
manipulate device assignment database entry for a trusted system	getdvagent(3)
manipulate event items; create and	EvmItemGet(3)
manipulate event variables	EvmVarGet(3)
manipulate flags for ELF files	elf_flag(3E)
manipulate protected password database entries (for trusted systems only)	getprpwent(3)
manipulate routing tables manually	route(1M)
manipulate section data for ELF files	elf_getdata(3E)
manipulate system default database entry for a trusted system	getprdfent(3)
manipulate terminal control database entry	getprtcent(3)
manipulate user contexts; DEPRECATED	makecontext(2)
manipulate values in unwind library data structure	_UNW_currentContext(3X)
manipulate, initialize, and test signal sets	sigsetops(3C)

Description	Entry Name(Section)
manipulation routines, Internet address	inet6(3N)
manpage information by keywords; find	man(1)
manpage, macro package for formatting	man(5)
manpage; print out a	man(1)
manpages for faster viewing with man command, fix	fixman(1M)
manpages; create cat and whatis files for online	catman(1M)
manpages; introduction to	Introduction(9)
mantissa and exponent, split floating-point number into	frexp(3M)
manual entries, macro package for formatting	man(5)
manual entry files for given name, find location of	wheris(1)
manually manipulate routing tables	route(1M)
manuals - accessing and ordering HP-UX documentation	manuals(5)
map between an interface name and index value; functions that	if_nameindex(3N)
map compartment name to number or number to name	cmpt_getbynum(3)
map device ID to file path	devnm(3)
map master; list which host is Network Information System server or	ypwhich(1)
map mechanism, QOP strings to non-string values	rpc_gss_mech_to_oid(3N)
map of ASCII character set	ascii(5)
map pages of memory	mmap(2)
map security file, NIS	securenets(4)
map stream pointer to file descriptor	fileno(3S)
map, number of entries in a kernel dynamic memory allocation	nsysmap(5)
map, print all values in a Network Information Service	ypcat(1)
map, print the values of selected keys in Network Information Service	ypmatch(1)
map-mbone - multicast router connection mapper	map-mbone(1M)
map; query NIS server for information about NIS	yppoll(1M)
mapped file or anonymous memory region, initialize semaphore in	msem_init(2)
mapped file or anonymous region, remove semaphore in	msem_remove(2)
mapped file, synchronize a	msync(2)
mapped keyboard driver; HP-HIL	hilkbd(7)
mapped memory which can be allocated after boot; determines the maximum amount (in MB) of equivalently	eqmem_limit(5)
mapped region, unmap a	munmap(2)
mapper, multicast router connection	map-mbone(1M)
mapper, universal addresses to RPC program number	rpcbind(1M)
mapping definitions, memory	mman(5)
mapping table; remove duplicate entries from gsscred	gsscred_clean(1M)
mapping, physical memory address	iomap(7)
maps, creates database maps for sendmail	makemap(1M)
mark a thread as detached to reclaim its resources when it terminates	pthread_detach(3T)
mark differences between two files	diffmk(1)
mask for file creation, set and get permissions	umask(2)
mask for file-creation, set access permissions mode	umask(1)
Mass Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel	fcmutil(1M)
master and slave pty, STREAMS, unlocking	unlockpt(3C)
master driver, STREAMS	tels(7)
master pty driver, STREAMS	ptm(7)
master; list which host is Network Information System server or map	ypwhich(1)
match EVM event name	EvmEventNameMatch(3)
match filename patterns	fnmatch(3C)
match routines for regular expressions	regexp(3X)
matching routines, regular expression	regcomp(3C)
math.h - floating-point environment macros and functions	fenv(5)
math: absolute value functions	fabs(3M)
math: arc hyperbolic cosine functions	acosh(3M)
math: arc hyperbolic sine functions	asinh(3M)
math: arc hyperbolic tangent functions	atanh(3M)
math: arccosine functions	acos(3M)
math: arcsine functions	asin(3M)
math: arctangent functions	atan(3M)

Index

All Volumes

Description	Entry Name(Section)
math: arctangent-and-quadrant functions	atan2(3M)
math: base-10 exponential functions	exp10(3M)
math: base-2 exponential functions	exp2(3M)
math: Bessel functions of the first kind	j0(3M)
math: Bessel functions of the second kind	y0(3M)
math: ceiling functions	ceil(3M)
math: common logarithm functions	log10(3M)
math: compound interest (growth) factor	compound(3M)
math: copysign functions	copysign(3M)
math: cosine functions	cos(3M)
math: cosine functions of a degree argument	cosd(3M)
math: cotangent functions	cot(3M)
math: cotangent functions of a degree argument	cotd(3M)
math: cube root functions	cbrt(3M)
math: decompose floating-point number	modf(3M)
math: degree-valued arccosine functions	acosd(3M)
math: degree-valued arcsine functions	asind(3M)
math: degree-valued arctangent functions	atand(3M)
math: degree-valued arctangent-and-quadrant functions	atan2d(3M)
math: error and complementary error functions	erf(3M)
math: Euclidean distance (hypotenuse) functions	hypot(3M)
math: exponential functions	exp(3M)
math: exponential minus 1 functions	expm1(3M)
math: extract mantissa and exponent from floating-point number	frexp(3M)
math: floating multiply-add functions	fma(3M)
math: floating-point classification macro	fpclassify(3M)
math: floating-point comparison macro (unordered)	isunordered(3M)
math: floating-point finiteness macro	isfinite(3M)
math: floating-point quiet comparison macro (<)	isless(3M)
math: floating-point quiet comparison macro (<=)	islessequal(3M)
math: floating-point quiet comparison macro (<>)	islessgreater(3M)
math: floating-point quiet comparison macro (>)	isgreater(3M)
math: floating-point quiet comparison macro (>=)	isgreaterequal(3M)
math: floating-point sign-determination	signbit(3M)
math: floating-point test for infinity	isinf(3M)
math: floating-point test for NaN	isnan(3M)
math: floating-point test for normal value	isnormal(3M)
math: floor functions	floor(3M)
math: functions and constants	math(5)
math: hyperbolic cosine functions	cosh(3M)
math: hyperbolic sine and hyperbolic cosine together	sinhcosh(3M)
math: hyperbolic sine functions	sinh(3M)
math: hyperbolic tangent functions	tanh(3M)
math: load exponent of a floating-point number	ldexp(3M)
math: log gamma functions	lgamma(3M)
math: logarithm base two functions	log2(3M)
math: maximum value functions	fmax(3M)
math: minimum value functions	fmin(3M)
math: natural logarithm functions	log(3M)
math: natural logarithm of one-plus-argument functions	log1p(3M)
math: next representable floating-point values	nextafter(3M)
math: positive difference functions	fdim(3M)
math: power functions	pow(3M)
math: present value factor for annuity	annuity(3M)
math: pseudorandom number generation functions	random(3M)
math: radix-independent exponent functions	ilogb(3M)
math: radix-independent exponent functions	logb(3M)
math: reciprocal square root functions	rsqrt(3M)
math: remainder functions	fmod(3M)
math: remainder functions	remainder(3M)
math: remainder functions with quotient	remquo(3M)

Description	Entry Name(Section)
math: round functions	round(3M)
math: round to long int functions	lround(3M)
math: round to long long functions	llround(3M)
math: round to nearest integer functions	rint(3M)
math: round to nearest long int functions	lrint(3M)
math: round to nearest long long functions	llrint(3M)
math: scale exponent of a floating-point number	ldexp(3M)
math: scale exponent of a radix-independent floating-point number	scalb(3M)
math: scale exponent of a radix-independent floating-point number	scalbn(3M)
math: sine and cosine of degree argument	sincos(3M)
math: sine and cosine together	sincos(3M)
math: sine functions	sin(3M)
math: sine functions of degree argument	sind(3M)
math: square root functions	sqrt(3M)
math: tangent functions	tan(3M)
math: tangent functions of a degree argument	tand(3M)
math: true gamma functions	tgamma(3M)
math: truncation functions	trunc(3M)
mathematical text for nroff, format	neqn(1)
max_acct_file_size - defines the maximum accounting file size	max_acct_file_size(5)
max_async_ports - maximum number of asynchronous disk ports that can be open at any time	max_async_ports(5)
max_mem_window - maximum number of group-private 32-bit shared memory configurable by users	max_mem_window(5)
max_thread_proc - defines the maximum number of threads allowed per process	max_thread_proc(5)
maxdsiz - maximum size (in bytes) of the data segment for any user process	maxdsiz(5)
maxdsiz_64bit - maximum size (in bytes) of the data segment for any user process	maxdsiz(5)
maxfiles - initial (soft) maximum number of file descriptors per process	maxfiles(5)
maxfiles_lim - hard maximum number of file descriptors per process	maxfiles_lim(5)
maximum accounting file size, defines the	max_acct_file_size(5)
maximum data length for transmission, get	rpc_gss_max_data_length(3N)
maximum HFS file system blocks to be read in one read-ahead operation when sequentially reading backwards	hfs_revra_per_disk(5)
maximum number of BSD pseudo terminals (ptys)	npty(5)
maximum number of bytes on a single System V IPC message queue	msgmnb(5)
maximum number of cloned DLPI streams allowed on the system	dspi_max_clones(5)
maximum number of file locks	nlocks(5)
maximum number of group-private 32-bit shared memory configurable by users	max_mem_window(5)
maximum number of HFS file system open inodes that can be in memory	ninode(5)
maximum number of open CDFS files (system-wide)	ncdnode(5)
maximum number of open files, system-wide (OBSOLETE)	nfile(5)
maximum number of outstanding STREAMS bufcall	nstrevent(5)
maximum number of POSIX async I/O operations that can be queued at any time	aio_max_ops(5)
maximum number of POSIX asynchronous I/O operations allowed in a listio call	aio_listio_max(5)
maximum number of STREAMS modules in a single stream	nstrpush(5)
maximum number of STREAMS-based pseudo terminals (pts)	nstrpty(5)
maximum number of symbolic links used to resolve a path name	fs_symlinks(5)
maximum number of System V IPC messages in the system at any time	msgtql(5)
maximum number of system-wide System V IPC message queues (IDs) allowed	msgmni(5)
maximum number of threads allowed per process, defines the	max_thread_proc(5)
maximum number of user processes per user, limits the	maxuprc(5)
maximum or minimum amount of physical memory used for caching file I/O data	filecache_max(5)
maximum size (in bytes) of the data segment for any user process	maxdsiz(5)
maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on an Integrity system	pa_maxssiz(5)
maximum size (in bytes) of the stack for any user process	maxssiz(5)
maximum size (in bytes) of the text segment for any user process	maxtsiz(5)
maximum size of streams message control in bytes	strctlsz(5)
maximum size of streams message data in bytes	strmsgsz(5)
maximum System V IPC message size in bytes (OBSOLETE)	msgmax(5)
maximum value functions	fmax(3M)

Index

All Volumes

Description	Entry Name(Section)
maxrsestsz - maximum size (in bytes) of the RSE stack for any user process	maxrsestsz(5)
maxrsestsz_64bit - maximum size (in bytes) of the RSE stack for any user process	maxrsestsz(5)
maxssiz - maximum size (in bytes) of the stack for any user process	maxssiz(5)
maxssiz_64bit - maximum size (in bytes) of the stack for any user process	maxssiz(5)
maxtsiz - maximum size (in bytes) of the text segment for any user process	maxtsiz(5)
maxtsiz_64bit - maximum size (in bytes) of the text segment for any user process	maxtsiz(5)
maxuprc - limits the maximum number of user processes per user	maxuprc(5)
maxvgs - OBSOLETE kernel tunable parameter	maxvgs(5)
mblen() - number of bytes in the multibyte character	multibyte(3C)
mbrlen() - get number of bytes in a character	mbrlen(3C)
mbrtowc() - convert a character to a wide-character code	mbrtowc(3C)
mbsinit() - determine conversion object status	mbsinit(3C)
mbsrtowcs() - convert a character string to a wide-character string (restartable)	mbsrtowcs(3C)
mbstowcs() - convert sequence of multibyte characters	multibyte(3C)
mbtowc() - number of bytes in the multibyte character	multibyte(3C)
mc - media changer manipulation utility	mc(1M)
mcrt0.o - execution startup routines	crt0(3)
mcrt0.o - execution startup routines for PA-RISC systems	crt0_pa(3)
measure time used to execute a command	time(1)
measure, convert units of	units(1)
mech - security mechanism and quality of protection (QOP) files	mech(4)
mechanism name (MN), convert for direct comparison	gss_export_name(3)
mechanisms and RPC version, get information on	rpc_gss_get_mechanisms(3N)
media changer device driver, SCSI	autochanger(7)
media changer manipulation utility	mc(1M)
mediainit - initialize disk or partition DDS tape	mediainit(1)
mem - main memory image file	mem(7)
memalign() - allocate aligned memory	memalign(3C)
memberships, show group	groups(1)
memccpy() - copy bytes from memory to another memory location	memory(3C)
memchr() - find first occurrence of byte in memory area	memory(3C)
memcmp() - compare contents of memory with byte	memory(3C)
memcntl() - memory management control	memcntl(3)
memcpy() - copy bytes from memory to another memory location	memory(3C)
memmove() - move memory contents	memory(3C)
memory address mapping, physical	iomap(7)
memory allocation map, number of entries in a kernel dynamic	nsysmap(5)
memory allocation policy on cell-based HP-UX servers; physical	numa_policy(5)
memory allocator; main	malloc(3C)
memory control operations, shared	shmctl(2)
memory for library structure (X/OPEN TLI-XTI)	t_free(3)
memory image file; main	mem(7)
memory in a process core dump, determines the inclusion of readable shared	core_addshmem_read(5)
memory in process core dump, determines the inclusion of read/write shared	core_addshmem_write(5)
memory in the background is enabled, zeroing of free	pagezero_daemon_enabled(5)
memory lockable for request call-back POSIX asynchronous I/O operations; percentage of physical	aio_physmem_pct(5)
memory management control	memcntl(3)
memory mapping definitions	mman(5)
memory mapping; set or check protection of	mprotect(2)
memory - memory operations	memory(3C)
memory operations	memory(3C)
memory operations based on wide-character	wmemory(3C)
memory pages are not dumped when a kernel panic occurs, defines which classes of kernel	dontdump(5)
memory pages compressed or uncompressed when a kernel panic occurs; selects whether the system dumps	dump_compress_on(5)
memory region, initialize semaphore in mapped file or anonymous	msem_init(2)
memory region; check for validity	mvalid(3)
memory segment identifiers in the system, number of System V shared	shmmni(5)
memory segment, get information for a System V shared	pstat(2)

Description	Entry Name(Section)
memory segment, get shared	shmget(2)
memory segment, maximum size (in bytes) for a System V shared	shmmx(5)
memory segments per process, maximum number of System V shared	shmseg(5)
memory statistics, report virtual	vmstat(1)
memory subsystem, get information about the virtual	pstat(2)
memory that can be used by audit subsystem; percentage of physical	audit_memory_usage(5)
memory to reserve for the 32-bit DMA pool; the amount of	dma32_pool_size(5)
memory used by an unwind environment, free	uwx_free(3X)
memory used for caching file I/O data; maximum or minimum amount of physical	filecache_max(5)
memory using multiple dump units when a kernel panic occurs on Integrity systems; enable or disable option for system to dump	dump_concurrent_on(5)
memory which can be allocated after boot; determines the maximum amount (in MB) of equivalently mapped	eqmem_limit(5)
memory window ID; file containing applications and their associated	services.window(4)
memory window; change window ID of running program or start program in particular ...	setmemwindow(1M)
memory, based on symbol name; perform I/O on kernel	kmem(7)
memory, enable or disable System V shared	shm(5)
memory, lock process address space in	mlockall(2)
memory, lock process into after allocating data and stack space	datalock(3C)
memory, lock process, text, data, stack, or shared library in	plock(2)
memory, lock segment in	mlock(2)
memory, unlock segment	munlock(2)
memory, unlock virtual address space	munlockall(2)
memory; allocate aligned	memalign(3C)
memory; copy a file into	copylist(3G)
memory; map pages of	mmap(2)
memorymap () - display contents of memory allocator	malloc(3C)
memset () - set area in memory to contain a specified byte	memory(3C)
Mercury Library Interfaces to transfer data between user and kernel space in a lightweight manner	hg(3)
merge and/or sort files	sort(1)
merge corresponding lines of several files or subsequent lines of one file	paste(1)
merge files; sort or	sort(1)
merge or add total accounting files	acctmerg(1M)
merge RCS revisions	rcsmerge(1)
merge - three-way file merge	merge(1)
merge two LVM logical volumes into one logical volume	lvmerge(1M)
mesg - enable or disable System V IPC messages at boot time (OBSOLETE)	mesg(5)
mesg - permit or deny <i>write(1)</i> messages from other users to terminal	mesg(1)
message catalog	glossary(9)
message catalog file, create for modification	findmsg(1)
message catalog file, extract messages from	findmsg(1)
message catalog file, generate a formatted	gencat(1)
message catalog for reading; open and close	catopen(3C)
message catalog path, configure	chnlspath(1M)
message catalog, set the default	setcat(3)
message catalog; display a selected message from a	dspmsg(1)
message catalog; display all or part of	dspcat(1)
message catalogs, find strings for inclusion in	findstr(1)
message control operations	msgctl(2)
message file, read text string from	gettxt(3C)
message files, create for use by <i>gettxt()</i>	mkmsg(1)
message from a message catalog; display a selected	dspmsg(1)
message from a socket; receive a	recv(2)
message from a socket; send a	send(2)
message header; changes or deletes a	smfi_chgheader()(3N)
message in standard format; display	pfmt(3C)
message integrity code (MIC), verify and decrypt message content	gss_unwrap(3)
message on a stream; send a	putmsg(2)
message operation permissions	glossary(9)
message operations	msgop(2)

Index

All Volumes

Description	Entry Name(Section)
message processing system; interactive mail	mailx(1)
message queue descriptor, close	mq_close(2)
message queue identifier (msqid)	glossary(9)
message queue identifier, remove	ipcrm(1)
message queue, get	msgget(2)
message queue, get information for a POSIX	pstat(2)
message queue, receive a message	mq_receive(2)
message queue, send a message	mq_send(2)
message queue, set the blocking status	mq_setattr(2)
message queue, unlink	mq_unlink(2)
message queue; create or open a	mq_open(2)
message queue; register or cancel a notification request with a	mq_notify(2)
message queues, report status	ipcs(1)
message queues; report status	pips(1)
message source file; preprocess a	mkcatdefs(1)
message transcription system	answer(1)
message using the given reason; quarantines the sendmail	smfi_quarantine(3N)
message, broadcast simultaneously to all users	wall(1M)
message, NLS program, get an	catgets(3C)
message, print libcrash error or warning message	cr_perror(3)
message, send or receive message queue message	msgop(2)
message; adds a header to the current	smfi_addheader(3N)
message; adds a recipient for the current	smfi_addrcpt(3N)
messages and other information on RCS files; print log	rlog(1)
messages from C source into a file; extract error	mkstr(1)
messages from other users to terminal, deny or permit <i>write</i> (1)	mesg(1)
messages to system log, send	logger(1)
messages, diagnostic, collect to form system error log	dmesg(1M)
messages; log system	syslogd(1M)
messages; write system error	perror(3C)
meta () - enable/disable meta-keys	meta(3X)
meta-keys, enable/disable	meta(3X)
metacharacter	glossary(9)
metric system, convert units to or from	units(1)
microloader	dld.so(5)
Militer for sendmail; starts an orderly shutdown of the	smfi_stop(3N)
Militer library for sendmail; sets the debugging level for the	smfi_setdbg(3N)
MIME (Multipurpose Internet Mail Extensions)	elm(1)
minimum amount of physical memory used for caching file I/O data; maximum or	filecache_max(5)
minimum priority for printing; define the	lpsched(1M)
minimum value functions	fmin(3M)
minor number	glossary(9)
mirrored LVM logical volume into two logical volumes; split	lvsplit(1M)
mirrors for LVM logical volume, increase	lvextend(1M)
mirrors in LVM logical volumes, synchronize stale	lvsync(1M)
mirrors in LVM volume groups, synchronize stale logical volume	vgsync(1M)
miscellaneous accounting commands; overview of accounting and	acct(1M)
miscellany, introduction to	intro(5)
mk_kernel - load a kernel configuration from a system file	mk_kernel(1M)
mkboot - install, update or remove boot programs from disk	mkboot(1M)
mkcatdefs - preprocess a message source file	mkcatdefs(1)
mkdir - make a directory	mkdir(1)
mkdir () - make a directory file	mkdir(2)
mkdirp () - create directories in a path	mkdirp(3G)
mkfifo () - make a FIFO file	mkfifo(3C)
mkfifo - make FIFO (named pipe) special files	mkfifo(1)
mkfs - construct a file system (generic)	mkfs(1M)
mkfs - construct an HFS file system	mkfs_hfs(1M)
mkfs_hfs - construct an HFS file system	mkfs_hfs(1M)
mklost+found - make a lost+found directory for the fsck command	mklost+found(1M)
mkmf - make a makefile	mkmf(1)

Description	Entry Name(Section)
mkmsgs - create message files for use by gettxt()	mkmsgs(1)
mknod - create special and FIFO files	mknod(1M)
mknod() - make a directory, special, or ordinary file	mknod(2)
mknod.h - header file of macros for handling device numbers	mknod(5)
mkssf - make a special (device) file	mkssf(1M)
mkstemp() - make a unique file name	mktemp(3C)
mkstr - extract error messages from C source into a file	mkstr(1)
mktemp - make a name for a temporary file	mktemp(1)
mktemp() - make a unique file name	mktemp(3C)
mktime() - convert time into calendar time value	ctime(3C)
mktimer() - allocate a per-process timer	mktimer(3C)
mkupath, uupath - access and manage the pathalias database	uupath(1)
mlock() - lock segment of process address space in memory	mlock(2)
mlockall() - lock all process address space in memory	mlockall(2)
MM macro package for formatting documents	mm(5)
mm - print or check documents formatted with the mm macros	mm(1)
mm - the MM macro package for formatting documents	mm(5)
mmap() - map pages of memory	mmap(2)
mmap64() - non-POSIX standard API interfaces to support large files	creat64(2)
mnttab - mounted file system table	mnttab(4)
mode	glossary(9)
mode mask for file-creation, set access permissions	umask(1)
mode, place system in single-user	shutdown(1M)
mode, set the cursor mode	 curs_set(3X)
model - HP-UX machine identification	model(4)
model information; print hardware	model(1)
model - print hardware model information	model(1)
model script configuration utility for psfontpf	psmsgen(1M)
modem - asynchronous serial modem line control	modem(7)
modem capability database	terminfo(4)
modem dialing	kermit(1)
modem line control; asynchronous serial	modem(7)
modes, speed, and line discipline; set terminal type,	getty(1M)
modes; save or restore program or shell terminal	 def_prog_mode(3X)
modest-sized programs, compiler/interpreter for	bs(1)
modf() - decompose floating-point number	modf(3M)
modff() - decompose floating-point number (float)	modf(3M)
modfl() - decompose floating-point number (long double)	modf(3M)
modfq() - decompose floating-point number (quad)	modf(3M)
modfw() - decompose floating-point number (extended)	modf(3M)
modification times, set or update file	utime(2)
modification, and/or change times of file; update access,	touch(1)
modification, create message catalog file for	 findmsg(1)
modify a group on the system	groupmod(1M)
modify a user login on the system	usermod(1M)
modify access control lists for files (JFS only)	setacl(1)
modify an attribute of a system complex	 cplxmodify(1M)
modify an existing partition	 parmodify(1M)
modify and delete user credentials for an authentication service	 pam_setcred(3)
modify boot variables in stable storage; display and	 setboot(1M)
modify configuration parameters of an existing LVM volume group	 vgmodify(1M)
modify environment for command execution	 env(1)
modify information in the user database, /var/adm/userdb	 userdbset(1M)
modify or view Access Control Lists	 swacl(1M)
modify protected password database	 modprpw(1M)
modify selected characters	 tr(1)
modify the Volume Group ID (VGID) on a given set of physical devices	 vgchgid(1M)
modify, add, or delete access control list entry	 setaclentry(3C)
modify, and delete entries in an LDAP directory; simple add,	 ldapentry(1)
modify, and delete event subscriptions; enables you to view, create,	 evweb_subscribe(1)
modify, delete, copy, add, or summarize file access control lists (ACLs)	 chacl(1)

Index

All Volumes

Description	Entry Name(Section)
modifying Network Information Service passwd database; daemon for	yppasswdd(1M)
modload () - load kernel modules on demand	modload(2)
modpath () - change global search path for dynamically loadable kernel modules	modpath(2)
modprpw - modify protected password database	modprpw(1M)
modstat () - get information for a dynamically loaded kernel module	modstat(2)
module for HP-UX, extended authentication, account, password, and session service	pam_hpsec(5)
module, Emulation for STREAMS pty	ptem(7)
module, Packet Mode, for STREAM pty	pckt(7)
module, PAM user policy definition service	pam_updbe(5)
module, remote magnetic tape dump and restore protocol	rmt(1M)
module; get information for a dynamically loaded kernel	modstat(2)
modules for LDAP; authentication, account, session, and password management PAM	pam_ldap(5)
modules; user configuration file for pluggable authentication	pam_user.conf(4)
moduload () - unload a kernel module on demand	moduload(2)
monacct - create periodic accounting summary files	acctsh(1M)
monetary value to string; convert	strfmon(3C)
Monitor Daemon; Essential Services	esmd(1M)
monitor EVM events	evmwatch(1)
monitor execution (in seconds), frequency of AIO thread pool	aio_monitor_run_sec(5)
monitor I/O conditions on multiple file descriptors	poll(2)
monitor I/O conditions on multiple file descriptors	poll(7)
monitor online cell operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel online cell operation;	parolrad(1M)
monitor OSPF gateways	ospf_monitor(1M)
monitor () - prepare execution profile	monitor(3C)
monitor profile data, display	prof(1)
monitor uucp subnetwork activity	uusub(1M)
monitor, create, distribute, install, and manage software	sd(5)
more - file perusal filter for screen viewing	more(1)
motd file	login(1)
mount a file system	vfsmount(2)
mount and unmount CDFS file systems	mount_cdfs(1M)
mount and unmount file systems	mount(1M)
mount and unmount HFS file systems	mount_hfs(1M)
mount and unmount remote NFS resources	mount_nfs(1M)
mount () - mount a file system	mount(2)
mount - mount an LOFS file system	mount_lofs(1M)
mount - mount and unmount CDFS file systems	mount_cdfs(1M)
mount - mount CacheFS file systems	mount_cachefs(1M)
mount - mount CDFS file systems	mount_cdfs(1M)
mount - mount file systems	mount(1M)
mount - mount HFS file systems	mount_hfs(1M)
mount - mount remote NFS resources	mount_nfs(1M)
mount multiple file systems	mountall(1M)
mount points; install automatic	automount(1M)
mount request server; NFS	mountd(1M)
mount_cachefs - mount and unmount CacheFS file systems	mount_cachefs(1M)
mount_cdfs - mount and unmount CDFS file systems	mount_cdfs(1M)
mount_hfs - mount and unmount HFS file systems	mount_hfs(1M)
mount_nfs - mount and unmount remote NFS resources	mount_nfs(1M)
mountable file system	glossary(9)
mountall - mount multiple file systems	mountall(1M)
mountd - NFS mount requests and access checks server	mountd(1M)
mounted file system statistics, get	ustat(2)
mounted file system table	mnttab(4)
mounted file system, list processes using	fuser(1M)
mounted file systems, keep track of remotely	mount(3N)
mounting by remote systems; make local NFS file systems unavailable for	unshare_nfs(1M)
mounting by remote systems; make local resource available for	share(1M)
mounting by remote systems; make local resource unavailable for	unshare(1M)
mounts, show all remote	showmount(1M)

Description	Entry Name(Section)
mouse, PS/2 device driver	ps2(7)
move a directory (requires super-user)	mvdir(1M)
move allocated physical extents from one LVM physical volume to other physical volumes	pvmmove(1M)
move file to new location	mv(1)
move files between systems	ftp(1)
move multiple files to another directory	mv(1)
move read/write file pointer; seek	lseek(2)
move requests between LP destinations;	lpsched(1M)
move window	mvwin(3X)
move() - window cursor location functions	move(3X)
movement commands cursor, output to the terminal	mvcur(3X)
mpctl() - multiprocessor control	mpctl(2)
mprotect() - set or check protection of memory mapping	mprotect(2)
mpsched - control processor or locality domain on which a specific process executes	mpsched(1)
mq_close() - close a message queue descriptor	mq_close(2)
mq_getattr() - get status information and attributes associated with a message queue	mq_getattr(2)
mq_notify() - register or cancel a notification request with a message queue	mq_notify(2)
mq_open() - create or open a message queue	mq_open(2)
mq_receive() - receive a message from a message queue	mq_receive(2)
mq_send() - send a message to a message queue	mq_send(2)
mq_setattr() - set the blocking status of a message queue associated with descriptor	mq_setattr(2)
mq_unlink() - unlink a message queue	mq_unlink(2)
mrand48() , jrand48() - generate signed long-integer pseudo-random numbers	drand48(3C)
mrinfo - multicast routing configuration information tool	mrinfo(1M)
mouted - IP multicast routing daemon	mouted(1M)
msem_init() - initialize semaphore in mapped file or anonymous memory region	msem_init(2)
msem_lock() - lock a semaphore	msem_lock(2)
msem_remove - remove semaphore in mapped file or anonymous region	msem_remove(2)
msem_unlock - unlock a semaphore	msem_unlock(2)
msgctl() - message control operations	msgctl(2)
msgget() - get message queue	msgget(2)
msgmap - number of entries in the System V IPC message space resource map (OBSOLETE)	msgmap(5)
msgmax - maximum System V IPC message size in bytes (OBSOLETE)	msgmax(5)
msgrcv() - receive message from message queue	msgop(2)
msgseg - number of System V IPC message segments in the system (OBSOLETE)	msgseg(5)
msgsnd() - send message to message queue	msgop(2)
msgssz - number of bytes in a System V IPC message segment (OBSOLETE)	msgssz(5)
msqid (message queue identifier)	glossary(9)
msync - synchronize a mapped file	msync(2)
mt - magnetic tape interface for stape and tape2	mt(7)
mt - magnetic tape manipulating program	mt(1)
MTA that a sendmail operation is still in progress; notifies the	smfi_progress(3N)
mtail - displays the last part of the mail log	mtail(1M)
MTAs use to connect to the filter; attempts to create the interface socket that	smfi_opensocket(3N)
multi-byte character length limited string, get from the terminal	getnstr(3X)
multi-byte character string, get from the terminal	getstr(3X)
multi-byte character string, input from a window	innstr(3X)
multi-byte character, insert into a window	insnstr(3X)
multi-byte characters, add a string of, without rendition to a window and advance cursor	addnstr(3X)
multi-language systems; configure system language on	geocustoms(1M)
multi-thread application, explicit locking of streams within a	flockfile(3S)
multibyte characters and strings conversions	multibyte(3C)
multibyte() - multibyte characters and strings conversions	multibyte(3C)
multicast router connection mapper	map-mbone(1M)
multicast routing configuration information tool	mrinfo(1M)
multicast routing daemon; IP	mouted(1M)
multiple adjacent blank lines, reduce to single blank line	ssp(1)
multiple dump units when a kernel panic occurs on Integrity systems; enable or disable option for system to dump memory using	dump_concurrent_on(5)
multiple file descriptors, monitor I/O conditions on	poll(7)
multiple file descriptors; monitor I/O conditions on	poll(2)

Index

All Volumes

Description	Entry Name(Section)
multiple file systems, mount and unmount	mountall(1M)
multiple files, split file into	csplit(1)
multiple line-feeds, remove from output	ssp(1)
multiple <i>n</i> -line pieces, split a file into	split(1)
Multiplexer (MUX)	glossary(9)
multiplexing; synchronous I/O	select(2)
multiprocessor control	mpctl(2)
Multipurpose Internet Mail Extensions (MIME)	elm(1)
multithreaded processes; tracing facility	ttrace(2)
multiuser state	glossary(9)
munlock() - unlock segment of process virtual address space	munlock(2)
munlockall() - unlock entire process virtual address space	munlockall(2)
munmap() - unmap a mapped region	munmap(2)
mutex attribute object, initialize or destroy	pthread_mutexattr_init(3T)
mutex handoff mode; disable mutex-specific or process-wide	pthread_mutexattr_getspin_np(3T)
mutex spin and yield frequency attributes; get and set	pthread_mutexattr_getspin_np(3T)
mutex, get and set the prioceiling of	pthread_mutex_getprioceiling(3T)
mutex, lock or try to lock	pthread_mutex_lock(3T)
mutex, unlock	pthread_mutex_unlock(3T)
mutex-specific or process-wide mutex handoff mode; disable	pthread_mutexattr_getspin_np(3T)
mutex; initialize or destroy	pthread_mutex_init(3T)
mv - move or rename files and directories	mv(1)
mvadd_wch() - add a complex character and rendition to a window	add_wch(3X)
mvadd_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
mvadd_wchstr() - add an array of complex characters and renditions to a window	add_wchstr(3X)
mvaddch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
mvaddchnstr() - add length limited string of single-byte characters and renditions to a window	addchnstr(3X)
mvaddchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
mvaddnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvaddnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvaddstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvaddwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvalid - check memory region for validity	mvalid(2)
mvchgat() - change renditions of characters in a window	chgat(3X)
mvcur() - output cursor movement commands to the terminal	mvcur(3X)
mvdelch() - delete character from a window	delch(3X)
mvderwin() - define window coordinate transformation	mvderwin(3X)
mvdir - move a directory (requires super-user)	mvdir(1M)
mvget_wch() - get a wide character from a terminal	get_wch(3X)
mvget_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvgetch() - get a single-byte character from the terminal	getch(3X)
mvgetn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvgetnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
mvgetstr() - get a multi-byte character string from the terminal	getstr(3X)
mvhline() - draw lines from single-byte characters and renditions	hline(3X)
mvhline_set() - draw lines from complex characters and renditions	hline_set(3X)
mvin_wch() - input a complex character and rendition from a window	in_wch(3X)
mvin_wchnstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvin_wchstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvinch() - input a single-byte character and rendition from a window	inch(3X)
mvinchnstr() - input an array of single-byte characters and renditions from a window	inchnstr(3X)
mvinchstr() - input an array of single-byte characters and renditions from a window	inchnstr(3X)
mvinnstr() - input a multi-byte character string from a window	innstr(3X)
mvinnwstr() - input a string of wide characters from a window	innwstr(3X)
mvins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
mvins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
mvins_wstr() - insert a wide-character string into a window	insnwstr(3X)
mvinsch() - insert a single-byte character and rendition into a window	insch(3X)

Description	Entry Name(Section)
mvinsnstr() - insert a multi-byte character into a window	insnstr(3X)
mvinsstr() - insert a multi-byte character into a window	insnstr(3X)
mvinstr() - input a multi-byte character string from a window	innstr(3X)
mvlnwstr() - input a string of wide characters from a window	innwstr(3X)
mvprintw() - print formatted output in window	mvprintw(3X)
mvscanw() - convert formatted input from a window	mvscanw(3X)
mvvline() - draw lines from single-byte characters and renditions	hline(3X)
mvvline_set() - draw lines from complex characters and renditions	hline_set(3X)
mvwadd_wch() - add a complex character and rendition to a window	add_wch(3X)
mvwadd_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
mvwadd_wchstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
mvwaddch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
mvwaddchnstr() - add length limited string of single-byte characters and renditions to a window	addchnstr(3X)
mvwaddchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
mvwaddnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvwaddnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvwaddstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvwaddwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvwchgat() - change renditions of characters in a window	chgat(3X)
mvwdelch() - delete character from a window	delch(3X)
mvwget_wch() - get a wide character from a terminal	get_wch(3X)
mvwget_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvwgetch() - get a single-byte character from the terminal	getch(3X)
mvwgetn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvwgetnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
mvwgetstr() - get a multi-byte character string from the terminal	getstr(3X)
mvwhline() - draw lines from single-byte characters and renditions	hline(3X)
mvwhline_set() - draw lines from complex characters and renditions	hline_set(3X)
mvwin() - move window	mvwin(3X)
mvwin_wch() - input a complex character and rendition from a window	in_wch(3X)
mvwin_wchnstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvwin_wchstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvwinch() - input a single-byte character and rendition from a window	inch(3X)
mvwinchnstr() - input an array of single-byte characters and renditions from a window	inchstr(3X)
mvwinchstr() - input an array of single-byte characters and renditions from a window	inchstr(3X)
mvwinnstr() - input a multi-byte character string from a window	innstr(3X)
mvwinnwstr() - input a string of wide characters from a window	innwstr(3X)
mvwins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
mvwins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
mvwins_wstr() - insert a wide-character string into a window	ins_nwstr(3X)
mvwinsch() - insert a single-byte character and rendition into a window	insch(3X)
mvwinsnstr() - insert a multi-byte character into a window	insnstr(3X)
mvwinsstr() - insert a multi-byte character into a window	insnstr(3X)
mvwinstr() - input a multi-byte character string from a window	innstr(3X)
mvwinwstr() - input a string of wide characters from a window	innwstr(3X)
mvvprintw() - print formatted output in window	mvprintw(3X)
mvvscanw() - convert formatted input from a window	mvscanw(3X)
mvvline() - draw lines from single-byte characters and renditions	hline(3X)
mvvline_set() - draw lines from complex characters and renditions	hline_set(3X)
naaagt - Native Agent Adapter for SNMP	naaagt(1M)
name and index value; functions that map between an interface	if_nameindex(3N)
name database, network	networks(4)
name database, service	services(4)
name enablement and display; long user and group	lugadmin(1M)
name for a temporary file, make	mktemp(1)
name list on Integrity systems; get entries from	nlist_ia(3C)
name list on PA-RISC systems; get entries from	nlist_pa(3C)
name list (symbol table) of object code file, print	nm(1)

Index

All Volumes

Description	Entry Name(Section)
name list; get entries from	nlist(3C)
Name Lookup Cache (DNLC); number of locks for the Directory	dnlc_hash_locks(5)
name of a file, change	mv(1)
name of a slave pty, get the	ptsname(3C)
name of an open file, get the full path	pstat(2)
name of current host system, set or display	hostname(1)
name of current host system; get	gethostname(2)
name of current host system; set	sethostname(2)
name of current NIS domain; get or set	getdomainname(2)
name of device	devnm(1M)
name of file owner or group, change	chown(1)
name of key; get	keyname(3X)
name of operating system, display	uname(1)
name of the user's terminal or pseudo-terminal, get	tty(1)
name resolution description; host	hostname(5)
name server control utility	rndc(1)
name server file format; translate host table to	hosts_to_named(1M)
name server; configuration file for Internet domain	named.conf(4)
name server; Internet domain	named(1M)
name server; send signals to the domain	sig_named(1M)
name servers interactively; query	nslookup(1)
Name Service Switch backend libraries; query	nsquery(1)
name services from a LDAP directory server; accessing	ldap-ux(5)
name space: attach a STREAMS file descriptor	fattach(3C)
name suffix conventions; file	suffix(5)
name, change login	su(1)
name, get disk description by its	getdiskbyname(3C)
name, get login	logname(1)
name, user, PAM routine to retrieve	pam_get_user(3)
name-service switch; configuration file for	nsswitch.conf(4)
name-to-address translation, generic transport	netdir(3N)
name-types supported by the specified mechanism, list	gss_inquire_names_for_mech(3)
name: change the name of a file	rename(2)
name: create a name for a temporary file	tmpnam(3S)
name: detach a name from a STREAMS-based file descriptor	fdetach(3C)
name: find name of a terminal	ttyname(3C)
name: get character-string representation of user login name	cuserid(3S)
name: get name from UID (obsolete)	getpw(3C)
name: obtain user login name	logname(3C)
name; match EVM event	EvmEventNameMatch(3)
name; working directory	pwd(1)
named configuration file syntax checking tool	named-checkconf(1)
named credential, allow an application to acquire a handle for existing	gss_add_cred(3)
named directories; search for named file in	pathfind(3G)
named - Internet domain name server	named(1M)
named pipe, special files; make FIFO	mkfifo(1)
named pipes, create	mknod(1M)
named semaphore, get information for a POSIX	pstat(2)
named-checkconf - named configuration file syntax checking tool	named-checkconf(1)
named-checkzone - zone validity checking tool	named-checkzone(1)
named.conf - configuration file for Internet domain name server	named.conf(4)
names and statistics for HFS file system, list file	ff_hfs(1M)
names database, host	hosts(4)
names from i-numbers, generate path	ncheck(1M)
names of known systems; list uucp	uucp(1)
names, convert an HFS file system to allow long file	convertfs(1M)
names, get entries from system cache of recently looked-up	pstat(2)
names: extract portions of path names	basename(1)
names; list of home directory	usermod(4)
names; print user and group IDs and	id(1)
NaN conversion functions; string to	nan(3M)

Description	Entry Name(Section)
nan() - string to NaN conversion function	nan(3M)
NaN, test for	isnan(3M)
nanf() - string to NaN conversion function (float)	nan(3M)
nanl() - string to NaN conversion function (long double)	nan(3M)
nanosleep() - high resolution sleep	nanosleep(2)
nanq() - string to NaN conversion function (quad)	nan(3M)
nanw() - string to NaN conversion function (extended)	nan(3M)
napms() - suspend the calling process	napms(3X)
Native Agent Adapter, for SNMP	naaagt(1M)
native language	glossary(9)
Native Language Support (NLS)	glossary(9)
native languages, NLS information about	nl_langinfo(3C)
natural logarithm functions	log(3M)
natural logarithm of one-plus-argument functions	log1p(3M)
nbuf - OBSOLETE kernel tunable parameter	dbc_max_pct(5)
nc_perror() - get network configuration data base entry	getnetconfig(3N)
nc_spperror() - get network configuration data base entry	getnetconfig(3N)
ncdnode - maximum number of open CDFS files (system-wide)	ncdnode(5)
ncheck - generate path names from i-numbers	ncheck(1M)
nclist - number of cblocks for pty and tty data transfers	nclist(5)
ncsize - number of Directory Name Lookup Cache (DNLC) entries	ncsize(5)
ncweb - launch the Network Interfaces Configuration and Network Services Configuration tools of HP System Management Homepage (HP SMH)	ncweb(1M)
ndd - network tuning	ndd(1M)
ndp - IPv6 Neighbor Discovery cache display and control	ndp(1M)
NDP; Neighbor Discovery Protocol,	ndp(7P)
nearbyint() - round to nearest integer function	rint(3M)
nearbyintf() - round to nearest integer function (float)	rint(3M)
nearbyintl() - round to nearest integer function (long double)	rint(3M)
nearbyintq() - round to nearest integer function (quad)	rint(3M)
nearbyintw() - round to nearest integer function (extended)	rint(3M)
Neighbor Discovery cache display and control, IPv6	ndp(1M)
Neighbor Discovery Protocol, NDP	ndp(7P)
neighboring systems description file format; PPP	ppp.Systems(4)
neqn - format mathematical text for nroff	neqn(1)
neqn, tbl, and nroff/troff constructs, remove	deroff(1)
net_aton() - network station address string conversion routines	net_aton(3C)
net_ntoa() - network station address string conversion routines	net_aton(3C)
netconfig - network configuration database	netconfig(4)
netdir() - generic transport name-to-address translation	netdir(3N)
netdir_free() - generic transport name-to-address translation	netdir(3N)
netdir_getbyaddr() - generic transport name-to-address translation	netdir(3N)
netdir_getbyname() - generic transport name-to-address translation	netdir(3N)
netdir_options() - generic transport name-to-address translation	netdir(3N)
netdir_perror() - generic transport name-to-address translation	netdir(3N)
netdir_spperror() - generic transport name-to-address translation	netdir(3N)
netfmt - format tracing and logging binary files	netfmt(1M)
netgroup - list of network groups	netgroup(4)
netname2host() - library routines for secure remote procedure calls	secure_rpc(3N)
netname2user() - library routines for secure remote procedure calls	secure_rpc(3N)
NETPATH component, get /etc/netconfig entry corresponding to	getnetpath(3N)
netrc - login information for ftp , rexec , and rexec()	netrc(4)
netstat - show network status	netstat(1)
nettl - control network tracing and logging	nettl(1M)
nettladm - network tracing and logging administration manager	nettladm(1M)
nettlconf - configure tracing and logging commands	nettlconf(1M)
nettlgen.conf - network tracing and logging configuration file	nettlgen.conf(4)
network and host byte order, convert values between	byteorder(3N)
network connections	kermit(1)
network connections, outgoing, list owners of	owners(1M)
network entry, get or set	getnetent(3N)

Index

All Volumes

Description	Entry Name(Section)
network file system device files	nfs(7)
Network File System (NFS)	glossary(9)
Network File System statistics	nfsstat(1M)
network group entry, get or set	getnetgrent(3C)
network groups; list of	netgroup(4)
network host entry, get, set, or end	gethostent(3N)
network I/O card access information	lan(7)
Network Information Service client interface	ypclnt(3C)
Network Information Service client interface	ypclnt(3C)
Network Information Service databases, build and install	ypinit(1M)
Network Information Service database and directory structure	ypfiles(4)
Network Information Service database; force propagation of	yppush(1M)
Network Information Service databases; create or rebuild	ypmake(1M)
Network Information Service map, print all values in a	ypcat(1)
Network Information Service map, print the values of selected keys in	ypmatch(1)
Network Information Service (NIS) server, binder, and transfer processes	ypserv(1M)
Network Information Service passwd database; daemon for modifying	yppasswdd(1M)
Network Information Service server; bind to particular	ypset(1M)
Network Information Service, update user password in	yppasswd(3N)
Network Information Service: NIS map, query NIS server for information about an	yppoll(1M)
Network Information System database; make a	makedbm(1M)
Network Information System (NIS), change login password in	yppasswd(1)
Network Information System server or map master; list which host is	ypwhich(1)
network interface management command for LAN and RDMA interfaces	nwmgr(1M)
network interface management command for VLAN interface	nwmgr_vlan(1M)
network interface management command for btlan driver	nwmgr_btlan(1M)
network interface management command for intl100 driver	nwmgr_intl100(1M)
network interface parameters, configure	ifconfig(1M)
Network Interfaces Configuration and Network Services Configuration tools of HP System Management Homepage (HP SMH); launch the	ncweb(1M)
network interfaces; get the compartment IDs associated with a	cmpt_get_ifcid(3)
network lock daemon	lockd(1M)
network name database	networks(4)
network packet routing; system support for local	routing(7)
network rwall server	rwalld(1M)
Network Services Configuration tools of HP System Management Homepage (HP SMH); launch the Network Interfaces Configuration and	ncweb(1M)
network station address string conversion routines	net_aton(3C)
network status monitor	statd(1M)
network status, show	netstat(1)
network test packets, send	ping(1M)
Network Time Protocol (NTP), query program	ntpq(1M)
Network Time Protocol (NTP), set time and date	ntpdate(1M)
Network Time Protocol (NTP) daemon	xntpd(1M)
network tracing and logging administration manager	nettladm(1M)
network tracing and logging configuration file	nettlgen.conf(4)
network tracing and logging; control	nettl(1M)
network tuning	nnd(1M)
network username server	rusersd(1M)
network, monitor uucp subnetwork activity	uusub(1M)
network, remote backup over	dump(1M)
network, restore file system incrementally across	restore(1M)
network, scatter data to check the	spray(3N)
network, show status	netstat(1)
network, write to all users over a	rwall(1M)
network; file containing commands for sharing resources across a	dfstab(4)
network; restore file system incrementally, local or across	restore(1M)
network; virtual local area	VLAN(7)
networking hash tables, determines the size of the	tphashsz(5)
Networking Interfaces; X/Open	xopen_networking(7)
networking; set system initial identity parameters	set_parms(1M)

Description	Entry Name(Section)
networks - network name database	networks(4)
new commands, install	install(1M)
new file system; construct a	newfs(1M)
new group to the system; add a	groupadd(1M)
new HFS file system; construct a	newfs_hfs(1M)
new key, creating in publickey database file	newkey(1M)
new partition; create a	parcreate(1M)
new process, create a	fork(2)
new user login to the system; add a	useradd(1M)
newaliases - install new elm aliases for user or system	newaliases(1)
newaliases - rebuilds the database for the mail aliases file	newaliases(1M)
newform - change or reformat a text file	newform(1)
newfs - construct a new file system	newfs(1M)
newfs - construct a new HFS file system	newfs_hfs(1M)
newfs_hfs - construct a new HFS file system	newfs_hfs(1M)
newgrp - equivalent to exec newgrp	cs(1)
newgrp - equivalent to exec newgrp	ksh(1)
newgrp - equivalent to exec newgrp	sh-posix(1)
newgrp - switch to a new group	newgrp(1)
newkey - create a new key in the publickey database file	newkey(1M)
newline character	glossary(9)
newline translation, enable/disable	nl(3X)
newmail - notify users of new mail in mailboxes	newmail(1)
newpad() - pad management functions	newpad(3X)
news - print news items	news(1)
newterm() - screen initialisation functions	initscr(3X)
newwin() - window creation functions	newwin(3X)
next delimiter; read stream up to	bgets(3G)
next representable floating-point values	nextafter(3M)
nextafter() - next representable floating-point value	nextafter(3M)
nextafterf() - next representable floating-point value (float)	nextafter(3M)
nextafterl() - next representable floating-point value (long double)	nextafter(3M)
nextafterq() - next representable floating-point value (quad)	nextafter(3M)
nextafterw() - next representable floating-point value (extended)	nextafter(3M)
nextkey() - get next key in database (old single-data-base version)	dbm(3X)
nexttoward() - next representable floating-point value	nextafter(3M)
nexttowardf() - next representable floating-point value (float)	nextafter(3M)
nexttowardl() - next representable floating-point value (long double)	nextafter(3M)
nexttowardq() - next representable floating-point value (quad)	nextafter(3M)
nexttowardw() - next representable floating-point value (extended)	nextafter(3M)
nfile - maximum number of open files, system-wide (OBSOLETE)	nfile(5)
nflocks - maximum number of file locks	nflocks(5)
NFS client, clear locks held on behalf of an	clear_locks(1M)
NFS clients, directories to export to	exports(4)
NFS daemon	biod(1M)
NFS daemon	nfsd(1M)
NFS environment configuration command	setonenv(1M)
nfs - file containing parameter values for NFS-related daemons	nfs(4)
NFS file system disk blocks, report number of free	df_hfs(1M)
NFS file system, determine which processes are using	fuser(1M)
NFS file systems unavailable for mounting by remote systems; make local	unshare_nfs(1M)
nfs logging daemon	nfslogd(1M)
nfs - network file system device files	nfs(7)
NFS resources; mount and unmount remote	mount_nfs(1M)
NFS security modes; list	nfssec.conf(4)
NFS security modes; overview of	nfssec(5)
NFS server logging configuration file	nfslog.conf(4)
NFS server; mount requests and access checks	mountd(1M)
NFS statistics	nfsstat(1M)
NFS user and group id mapping daemon	nfsmapid(1M)
NFS version 2 client; control the number of kernel threads that perform asynchronous I/O for	

Index

All Volumes

Description	Entry Name(Section)
NFS version 3 client; control the number of kernel threads that perform asynchronous I/O for	nfs2_max_threads(5)
.....	nfs3_max_threads(5)
NFS version 3 client; control the number of read-ahead operations queued by when sequentially accessing a file	nfs3_nra(5)
NFS version 3 clients; control the logic block size used by	nfs3_bsize(5)
NFS version 3 read, write, readdir, or readdirplus request over TCP; control the data portion size of a	nfs3_max_transfer_size_cots(5)
.....	nfs3_max_transfer_size(5)
NFS version 3 read, write, readdir, or readdirplus request; control the data portion size of a	nfs3_max_transfer_size(5)
NFS version 3 readdirplus functionality on the NFS server; turn on or off	nfs3_do_readdirplus(5)
NFS Version 4 callback daemon	nfs4cbd(1M)
NFS version 4 client; control the number of kernel threads that perform asynchronous I/O for	nfs4_max_threads(5)
NFS version 4 client; control the number of read-ahead operations queued by when sequentially accessing a file	nfs4_nra(5)
NFS version 4 clients; control the logic block size used by	nfs4_bsize(5)
NFS version 4 read, write, readdir, or readdirplus request over TCP; control the data portion size	nfs4_max_transfer_size_cots(5)
.....	nfs4_max_transfer_size(5)
NFS version 4 read, write, readdir, or readdirplus request; control the size of the data portion	nfs4_max_transfer_size(5)
NFS, enable swapping across	remote_nfs_swap(5)
NFS-related daemons; file containing parameter values	nfs(4)
nfs2_max_threads - control the number of kernel threads that perform asynchronous I/O for the NFS version 2 client	nfs2_max_threads(5)
nfs2_nra - control the number of read-ahead operations queued by the NFS version 2 client when sequentially accessing a file	nfs2_nra(5)
nfs3_bsize - control the logic block size used by NFS version 3 clients	nfs3_bsize(5)
nfs3_do_readdirplus - turn on or off NFS version 3 readdirplus functionality on the NFS server	nfs3_do_readdirplus(5)
.....	nfs3_do_readdirplus(5)
nfs3_jukebox_delay - control the length of time the NFS version 3 client waits before re-transmitting request after receiving NFS3ERR_JUKEBOX error	nfs3_jukebox_delay(5)
nfs3_max_threads - control the number of kernel threads that perform asynchronous I/O for the NFS version 3 client	nfs3_max_threads(5)
nfs3_max_transfer_size - control the data portion size of a NFS version 3 read, write, readdir, or readdirplus request	nfs3_max_transfer_size(5)
nfs3_max_transfer_size_cots - control the data portion size of a NFS version 3 read, write, readdir, or readdirplus request over TCP	nfs3_max_transfer_size_cots(5)
nfs3_nra - control the number of read-ahead operations queued by the NFS version 3 client when sequentially accessing a file	nfs3_nra(5)
NFS3ERR_JUKEBOX error; control the length of time the NFS version 3 client waits before re-transmitting request after receiving	nfs3_jukebox_delay(5)
nfs4_bsize - control the logic block size used by NFS version 4 clients	nfs4_bsize(5)
nfs4_max_threads - control the number of kernel threads that perform asynchronous I/O for the NFS version 4 client	nfs4_max_threads(5)
nfs4_max_transfer_size - control the size of the data portion of a NFS version 4 read, write, readdir, or readdirplus request	nfs4_max_transfer_size(5)
nfs4_max_transfer_size_cots - control the data portion size of a NFS version 4 read, write, readdir, or readdirplus request over TCP	nfs4_max_transfer_size_cots(5)
nfs4_nra - control the number of read-ahead operations queued by the NFS version 4 client when sequentially accessing a file	nfs4_nra(5)
nfs4cbd - NFS Version 4 callback daemon	nfs4cbd(1M)
nfs_portmon - enable/disable the NFS server's source port verification check	nfs_portmon(5)
nfsd - NFS daemon	nfsd(1M)
nfslog.conf - NFS server logging configuration file	nfslog.conf(4)
nfslogd - nfs logging daemon	nfslogd(1M)
nfsmapid - NFS user and group id mapping daemon	nfsmapid(1M)
nfssec - overview of NFS security modes	nfssec(5)
nfssec.conf - list NFS security modes	nfssec.conf(4)
nfsstat - Network File System statistics	nfsstat(1M)
nftw() - walk a file tree executing a function	nftw(3C)

Description	Entry Name(Section)
nftw2 () - walk a file tree executing a function	ftw (3C)
nftw64 () - file sysmmmaptem API to support large files	fgetpos64 (3S)
nice - alter command priority	csh (1)
nice () - change priority of a process	nice (2)
nice - run a command at nondefault priority	nice (1)
nice value	renice (1M)
nice value of a thread; get or set	pthread_get_nice_np (3T)
ninode - maximum number of HFS file system open inodes that can be in memory	ninode (5)
NIS database from server to local node; transfer	ypxfr (1M)
NIS domain name; set or display	domainname (1)
NIS domain; get or set name of current	getdomainname (2)
NIS information, changes	ypupdate (3C)
NIS information, server for changing	ypupdated (1M)
NIS map security file	securenets (4)
NIS map, updates to	udpublickey (1M)
NIS map; query NIS server for information about	yppoll (1M)
NIS (Network Information Service) server, binder, and transfer processes	ypserv (1M)
NIS (Network Information System), change login password in	yppasswd (1)
NIS server for information about NIS map; query	yppoll (1M)
NIS server, binder, and transfer processes; Network Information Service	ypserv (1M)
NIS updating, configuration file for	updaters (1M)
nkthread - limits the number of threads allowed to run simultaneously	nkthread (5)
nl () - enable/disable newline translation	nl (3X)
nl - line numbering filter	nl (1)
nl_langinfo () - obtain NLS string form of local language variable	nl_langinfo (3C)
nlist () - get entries from name list	nlist (3C)
nlist () - get entries from name list on Integrity systems	nlist_ia (3C)
nlist () - get entries from name list on PA-RISC systems	nlist_pa (3C)
nlist - structure formats	nlist (4)
nlist - structure formats for Integrity systems	nlist_ia (4)
nlist - structure formats for PA-RISC systems	nlist_pa (4)
nlist64 () - get entries from name list	nlist (3C)
nlist64 () - get entries from name list on Integrity systems	nlist_ia (3C)
nlist64 () - get entries from name list on PA-RISC systems	nlist_pa (3C)
nlist64 - structure formats	nlist (4)
nlist64 - structure formats for Integrity systems	nlist_ia (4)
nlist64 - structure formats for PA-RISC systems	nlist_pa (4)
nlist_ia - get entries from name list on Integrity systems	nlist_ia (3C)
nlist_ia - structure formats for Integrity systems	nlist_ia (4)
nlist_pa - get entries from name list on PA-RISC systems	nlist_pa (3C)
nlist_pa - structure formats for PA-RISC systems	nlist_pa (4)
nljust - justify lines left or right for NLS printing	nljust (1)
NLS information; get locale-specific	locale (1)
NLS (Native Language Support)	glossary (9)
NLS, description of supported languages	lang (5)
NLS, get an NLS program message	catgets (3C)
NLS, information about native languages	nl_langinfo (3C)
NLS: justify lines left or right for NLS printing	nljust (1)
NLSPATH	glossary (9)
NLSPATH configuration file	nlspath (4)
nlspath - NLSPATH configuration file	nlspath (4)
nm - print name list of common object file	nm (1)
nocbreak () - input mode control functions	cbreak (3X)
node from a binary search tree, delete a	tsearch (3C)
node name	glossary (9)
node name (system name); set	uname (2)
node name, display/set	uname (1)
node name; size of	nodehostnamesize (5)
node number, set crash dump	cr_set_node (3)
node, get information about an SCA system	pstat (2)
node; transfer NIS database from server to local	ypxfr (1M)

Index

All Volumes

Description	Entry Name(Section)
nodehostnamesize - size of node name and host name	nodehostnamesize(5)
nodelay() - enable or disable block during read	nodelay(3X)
noecho() - enable/disable terminal echo	echo(3X)
nohup - ignore hangups during command execution	cs(1)
nohup - run a command immune to hangups	nohup(1)
non-ASCII characters in a file, make visible or invisible	vis(1)
non-interactive editing of the authorization information in the RBAC databases	authadm(1M)
non-local goto, save/restore stack environment for	setjmp(3C)
non-POSIX standard API interfaces to support large files	creat64(2)
nondefault priority, run a command at	nice(1)
noninteractive editing of a command's authorization and privilege information in the privrun database	cmdprivadm(1M)
noninteractive editing of role-related information in RBAC databases	roleadm(1M)
nonl() - enable/disable newline translation	nl(3X)
nonshadow, convert passwords from shadow to	pwunconv(1M)
nonspacing characters	glossary(9)
nop (do nothing) and return zero or non-zero exit status	true(1)
noqiflush() - enable/disable queue flushing	noqiflush(3X)
noraw() - input mode control functions	cbreak(3X)
normal value, test for	isnormal(3M)
notification request with a message queue; register or cancel a	mq_notify(2)
notification; establish a subscription for event	EvmConnSubscribe(3)
notifies the MTA that a sendmail operation is still in progress	smfi_progress(3N)
notify - notify user of change in job status	cs(1)
notify users of new mail in mailboxes	newmail(1)
notify you when it is time to leave	leave(1)
notimeout() - control blocking on input	notimeout(3X)
nPartition ; cancel online cell operation; monitor online cell operation; reset hung cell during cell activation; online activation of a cell from	parolrad(1M)
nproc - limits the number of processes allowed to exist simultaneously	nproc(5)
npty - maximum number of pseudo terminals (ptys)	npty(5)
nroff , format mathematical text for	neqn(1)
nroff , preprocess tables for	tbl(1)
nroff - format text	nroff(1)
nroff input, eliminate .so's from	soelim(1)
nroff/troff files, check	checknr(1)
nroff/troff , tbl , and neqn constructs, remove	deroff(1)
nslookup - query name servers interactively	nslookup(1)
nsquery - query the Name Service Switch backend libraries	nsquery(1)
nsswitch.conf - configuration file for the name-service switch	nsswitch.conf(4)
NSTREVENT - maximum number of outstanding STREAMS bufcalls	nstrevent(5)
nstrpty - maximum number of STREAMS-based pseudo-teletypes (pts)	nstrpty(5)
NSTRPUSH - maximum number of STREAMS modules in a single stream	nstrpush(5)
NSTRSCHED - number of STREAMS scheduler daemons to run	nstrsched(5)
nstrtel - specifies the number of telnet device files the kernel can support for incoming telnet sessions	nstrtel(5)
nsupdate - Dynamic DNS update utility	nsupdate(1)
nswapdev - maximum number of devices that can be enabled for swap	nswapdev(5)
nswapfs - maximum number of file systems that can be enabled for swap	nswapfs(5)
nsysmap - number of entries in a kernel dynamic memory allocation map	nsysmap(5)
nsysmap64 - number of entries in a kernel dynamic memory allocation map	nsysmap(5)
ntohl() - convert values between host and network byte order	byteorder(3N)
ntohs() - convert values between host and network byte order	byteorder(3N)
NTP daemon	xntpd(1M)
NTP (Network Time Protocol), set time and date	ntpdate(1M)
NTP query program; special	xntpd(1M)
NTP, query program	ntp(1M)
ntpdate - set time and date via NTP	ntpdate(1M)
ntpq - Network Time Protocol query program	ntpq(1M)
null - null file	null(7)
nulladm - create empty file owned by adm with mode 664	acctsh(1M)

Description	Entry Name(Section)
numa_policy - physical memory allocation policy on cell-based HP-UX servers	numa_policy(5)
number generator, strong random	random(7)
number of BSD pseudo terminals (ptys), maximum	npty(5)
number of Buffer Cache Pages used by sendfile, maximum	sendfile_max(5)
number of bytes in a character; get	mbrlen(3C)
number of bytes on a single System V IPC message queue, maximum	msgmnb(5)
number of columns on terminal screen	COLS(3X)
number of devices that can be enabled for swap; maximum	nswapdev(5)
number of file systems that can be enabled for swap; maximum	nswapfs(5)
number of free disk clusters; report	dosdf(1)
number of lines on terminal screen	LINES(3X)
number of locks for the Directory Name Lookup Cache (DNLC)	dnlc_hash_locks(5)
number of processors available, determine	pthread_processor_bind_np(3T)
number of queued signals per process, limit on	ksi_send_max(5)
number of STREAMS scheduler daemons to run	nstrsched(5)
number of STREAMS-based pseudo terminals (pts), maximum	nstrpty(5)
number of System V IPC messages in the system at any time, maximum	msgtql(5)
number of System V IPC semaphores per identifier, maximum	semmsl(5)
number of System V IPC undo entries per process, maximum	semume(5)
number of System V shared memory segments per process, maximum	shmseg(5)
number of system-wide System V IPC message queues (IDs) allowed, maximum	msgmni(5)
number of telnet device files the kernel can support for incoming telnet sessions, specifies the	nstrtel(5)
number of threads allowed per process, defines the maximum	max_thread_proc(5)
number of user processes per user, limits the maximum	maxuprc(5)
number of users for each class	ftpcount(1)
number to string, convert long double floating-point	ldcvt(3C)
number to string; convert floating-point	ecvt(3C)
number; scale exponent of a radix-independent floating-point	scalbln(3M)
numbering filter; line	nl(1)
numbers for HP-UX implementations; magic	magic(4)
numbers of Kerberos principals; print key version	kvno(1)
numbers, generate uniformly distributed pseudo-random	drand48(3C)
numbers, inode generation, install random	fsirand(1M)
numeric paging	kermit(1)
nwmgr - network interface management command for LAN and RDMA interfaces	nwmgr(1M)
nwmgr - network interface management command for VLAN interface	nwmgr_vlan(1M)
nwmgr - network interface management command for btlan driver	nwmgr_btlan(1M)
nwmgr - network interface management command for intl100 driver	nwmgr_intl100(1M)
nwmgr_btlan - network interface management command for btlan driver	nwmgr_btlan(1M)
nwmgr_intl100 - network interface management command for intl100 driver	nwmgr_intl100(1M)
nwmgr_vlan - network interface management command for VLAN interface	nwmgr_vlan(1M)
O_LARGEFILE() - non-POSIX standard API interfaces to support large files	creat64(2)
object code debugger	adb(1)
object code file, print symbol table (name list) for	nm(1)
object code files in a library, find optimum sequence for	lorder(1)
object file access library	elf(3E)
object file header for elf32 or elf64 file; retrieve class-dependent	elf_getehdr(3E)
object file, ELF, finish using	elf_end(3E)
object file, link editor and assembler	a.out(4)
object file, strip symbol and line number information from	strip(1)
object files in binary directories; install	cpset(1M)
object files, print section sizes and allocation space of	size(1)
object files: dump information contained in	elfdump(1)
object files: dump information contained in	odump(1)
object or binary file, find the printable strings in an	strings(1)
objects explicitly attached using dlopen()/shl_load() ; list the dynamic libraries linked into each process, including shared	pldd(1)
objects that Software Distributor (SD) uses, their attributes and storage formats; all	sd(4)
(OBSOLETE); enable and disable use of device's write cache in the SCSI subsystem	default_disk_ir(5)
(OBSOLETE); number of bytes in a System V IPC message segment	msgsz(5)
obsolete library routines for RPC	rpc_soc(3N)

Index

All Volumes

Description	Entry Name(Section)
(OBSOLETE); audio tools available through HP VUE	Audio(5)
(OBSOLETE); control access to audio on a workstation;	asecure(1M)
obtain and cache the Kerberos ticket-granting ticket	kinit(1)
obtain source information from ELF files	uwx_find_source_info(3X)
obtain symbolic information from ELF files	uwx_find_symbol(3X)
obtain the thread ID for the calling thread	pthread_self(3T)
obtaining information from the kernel, an infrastructure for	pstat(2)
ocd - outbound connection daemon used by DDFA software	ocd(1M)
ocdebug - outbound connection daemon debug utility used by DDFA software	ocdebug(1M)
octal and hexadecimal file dump	od(1)
octal equivalents: ASCII character set	ascii(5)
od - octal and hexadecimal file dump	od(1)
od - octal file dump	od(1)
odump - dump information contained in SOM object files	odump(1)
offset for an object file, get base	elf_getbase(3E)
OID set, add an Object Identifier (OID)	gss_add_oid_set_member(3)
OLA/R/D functions	rad(1M)
olrad - command for Online Addition/Replacement/Deletion of PCI I/O cards and Online Addition of I/O chassis	olrad(1M)
olrad, rad features have been moved to	rad(1M)
on - execute command on a remote host	on(1)
on/off or display current status of power for cells and I/O chassis; turn	frupower(1M)
once; call an initialization routine only	pthread_once(3T)
onintr - specify shell's treatment of interrupts	csh(1)
online activation of a cell from nPartition; cancel online cell operation; monitor online cell operation; reset hung cell during cell activation	parolrad(1M)
Online Addition of I/O chassis; command for Online Addition/Replacement/Deletion of PCI I/O cards and	olrad(1M)
Online Addition/Replacement/Deletion of PCI I/O cards and Online Addition of I/O chassis; command for	olrad(1M)
online cell operation; monitor online cell operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel	parolrad(1M)
online cell operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel online cell operation; monitor	parolrad(1M)
online manpages; create cat and whatis files for	catman(1M)
opaque internal name, provide textual representation to an application	gss_display_name(3)
open a message queue; create or	mq_open(2)
open a pseudo-terminal master device	posix_openpt(3C)
open a shared library	dlopen(3C)
open a shared library on Integrity systems	dlopen_ia(3C)
open an HP 9000 64-bit shared library with explicit load address	dlopen_pa(3C)
open an HP 9000 shared library	dlopen_pa(3C)
open and close message catalog for reading	catopen(3C)
open crash dump for reading	cr_open(3)
open file	glossary(9)
open file description	glossary(9)
open file descriptor to a specific slot; duplicate an	dup2(2)
open file descriptor, duplicate an	dup(2)
open file descriptors; displays process address information and	pmap(1)
open file of a process, get information for an	pstat(2)
open file, get detailed information for an	pstat(2)
open file, get the full path name of an	pstat(2)
open file; apply or remove an advisory or enforced lock on an	flock(2)
open files, file control options for	fcntl(5)
open() - open file for reading or writing	open(2)
open or close pipe I/O to or from a process	popen(3S)
open or reopen a stream file	fopen(3S)
Open Shortest Path First (OSPF)	ospf_monitor(1M)
open() - STREAMS enhancements to standard system calls	stream(2)
open-file control routines	fcntl(2)
open/create a named semaphore	sem_open(2)

Description	Entry Name(Section)
open64 () - non-POSIX standard API interfaces to support large files	creat64 (2)
open ; determines whether to reserve a tape device on	st_ats_enabled (5)
open_secdef () - security defaults configuration file routines	secdef (3)
opendir () - open a directory and associated directory stream for access	directory (3C)
openlog () - control system log	syslog (3C)
operating system and <i>HP-UX Reference</i> ; HP-UX	introduction (9)
operating system information, display	uname (1)
operating system name, display	uname (1)
operating system state to the file system for debugging purposes., a feature that saves	livedump (5)
operating system, load (reboot)	boot (1M)
operating system, save a crash dump of	savecrash (1M)
operating system, updates HP-UX	update-ux (1M)
operation; monitor online cell operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel online cell	parolrad (1M)
operation; reset hung cell during cell activation; online activation of a cell from nPartition; cancel online cell operation; monitor online cell	parolrad (1M)
operational user interface for gated	gdc (1M)
operations on a stream file, get or reposition pointer for I/O	fseek (3S)
operations, clock	clocks (2)
operations, message control	msgctl (2)
operations, semaphore control	semctl (2)
operations, shared memory control	shmctl (2)
operations; directory	directory (3C)
operations; memory	memory (3C)
operations; percentage of physical memory lockable for request call-back POSIX asynchronous I/O	aio_physmem_pct (5)
operations; real-time scheduling	rtsched (2)
operations; timer	timers (2)
optarg - get option letter from argument vector	getopt (3C)
opterr - get option letter from argument vector	getopt (3C)
optimization functions; definitions for screen handling and	curses (5)
optimization package; terminal and printer handling and	curses_intro (3X)
optimize an existing HFS file system	tunefs (1M)
optimum sequence for object code files in a library, find	lorder (1)
optind - get option letter from argument vector	getopt (3C)
option for system to dump memory using multiple dump units when a kernel panic occurs on Integrity systems; enable or disable	dump_concurrent_on (5)
option letter from argument vector; get	getopt (3C)
options for a non-serial printer, set printing	slp (1)
options for a terminal port; set the	stty (1)
options for a transport endpoint; manage	t_optmgmt (3)
options on sockets; get and set	getsockopt (2)
options to specify the scheduling contention scope of threads, list of external	pthread_scope_options (5)
options, file control for open files	fcntl (5)
options, parse suboptions from a string	getsubopt (3C)
options, parse utility (command)	getopts (1)
options; parse command	getopt (1)
optopt - get option letter from argument vector	getopt (3C)
opx25 - execute HALGOL programs	opx25 (1M)
order of data, convert string	strord (3C)
ordering HP-UX documentation; accessing and	manuals (5)
ordering relation for files in an object code library, find	lorder (1)
ordinary file	glossary (9)
ordinary file; make	mknod (2)
orientation of a stream	orientation (5)
orientation - orientation of a stream	orientation (5)
orientation; set stream	fwide (3C)
original file; program to apply a diff file to an	patch (1)
orphan process	glossary (9)
orphaned process group	glossary (9)

Index

All Volumes

Description	Entry Name(Section)
osdd - print or check documents formatted with the mm macros	mm (1)
OSPF gateways; monitor	ospf_monitor (1M)
ospf_monitor - monitor OSPF gateways	ospf_monitor (1M)
other processes; force target process to run serially with	serialize (1)
other processes; force target process to run serially with	serialize (2)
OTHER scheduling policy	rtsched (2)
out of office mail response	vacation (1)
out-of-band mark; determine whether a socket is at the	socketmark (3N)
outbound connection daemon debug utility used by DDFA software	ocdebug (1M)
outbound connection daemon used by DDFA software	ocd (1M)
outgoing network connections, list owners of	owners (1M)
outgoing terminal line connection; establish	dial (3C)
output attributes to terminal	vidattr (3X)
output commands to the terminal	putp (3X)
output control functions, terminal	clearok (3X)
output cursor movement commands to the terminal	mvcur (3X)
output device, finite-width, fold long lines for	fold (1)
output first few lines in a file	head (1)
output (format and print) files	pr (1)
output format, link editor and assembler	a.out (4)
output to file; pipe fitting to copy standard	tee (1)
output, formatted, print in window	mvprintw (3X)
output, formatted; print to standard output, file, or string	vwprintf (3C)
output/input, buffered, standard stream file package	stdio (3S)
over TCP/IP server daemon; UUCP	uucpd (1M)
overlapped windows, copy	overlay (3X)
overlay () - copy overlapped windows	overlay (3X)
overview of accounting and miscellaneous accounting commands	acct (1M)
overview of ewweb commands; provides an	ewweb (1)
overview of NFS security modes	nfsec (5)
overview of stack unwind library entry points and convenience macros	unwind (5)
overview of various system shells	sh (1)
overwrite () - copy overlapped windows	overlay (3X)
overwrite file with an existing file	cp (1)
overwrite file with an existing file	mv (1)
owner	glossary (9)
owner and group of a file, change	chown (2)
owner and/or group, change in access control list (ACL)	chownacl (3C)
owner of file, change	chown (1)
owners - lists owners of outgoing network connections	owners (1M)
ownership, summarize file system	quot (1M)
PA-RISC emulator on an Integrity system; maximum size (in bytes) of the stack for a user process running under the	pa_maxssiz (5)
PA-RISC HP-UX applications on Itanium-based systems running HP-UX; emulate	Aries (5)
PA-RISC systems; assembler for	as_pa (1)
PA-RISC systems; change program's internal attributes on	chatr_pa (1)
PA-RISC systems; execution startup routines for	crto_pa (3)
PA-RISC systems; explicit load of shared libraries for	shl_load_pa (3X)
PA-RISC systems; get entries from name list on	nlist_pa (3C)
PA-RISC systems; link editor for	ld_pa (1)
PA-RISC systems; list dynamic dependencies of executable files or shared libraries on	ldd_pa (1)
PA-RISC systems; structure formats for	nlist_pa (4)
pa_maxssiz - maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on an Integrity system	pa_maxssiz (5)
pa_maxssiz_32bit - maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on an Integrity system	pa_maxssiz (5)
pa_maxssiz_64bit - maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on an Integrity system	pa_maxssiz (5)
pack - compress files using Huffman code	pack (1)
pack files and file systems in the cache	cachefspack (1M)
packages; file that registers distributed file system	fstypes (4)

Description	Entry Name(Section)
packed-decimal library, HP 3000-mode	hppac(3X)
packet filter specification file format; PPP	ppp.Filter(4)
packet forwarding database; kernel	route(7P)
Packet Mode module for STREAMS pty	pckt(7)
packet routing; system support for local network	routing(7)
packets, echo	ping(1M)
packets, ECHO_REQUEST	ping(1M)
packets, spray	spray(1M)
packing rules file; cachefs	packingrules(4)
packingrules - packing rules file for cachefs	packingrules(4)
pad management functions	newpad(3X)
pad, enhanced, management function	subpad(3X)
pad, refresh immediately after writing a character rendition	pechochar(3X)
page - file perusal filter for screen viewing	more(1)
page number, physical, validate whether dumped	cr_isaddr(3)
page size, get the current	getpagesize(2)
page size; minimum (in kilobytes) of system-selected	vps_pagesize(5)
pages are not dumped when a kernel panic occurs, defines which classes of kernel memory	dontdump(5)
pages of memory; map	mmap(2)
Pages used by sendfile, maximum number of Buffer Cache	sendfile_max(5)
pagezero_daemon_enabled - zeroing of free memory in the background is enabled	pagezero_daemon_enabled(5)
paging behavior, advise system of process's expected	madvise(2)
paging space information; system	swapinfo(1M)
paging; add swap space for interleaved	swapon(2)
paging; enable device or file system for	swapon(1M)
pair of connected sockets; create a	socketpair(2)
pair_content() - color manipulation functions	can_change_color(3X)
PAM	<i>see Pluggable Authentication Modules</i>
PAM account validation procedures; perform	pam_acct_mgmt(3)
PAM module that provides user authorization	pam_authz(5)
PAM modules for LDAP; authentication, account, session, and password management	pam_ldap(5)
PAM() - Pluggable Authentication Module	pam(3)
pam - Pluggable Authentication Module	pam(3)
PAM routine to retrieve user name	pam_get_user(3)
PAM routines to maintain module specific state	pam_set_data(3)
PAM Service Module APIs	pam_sm(3)
PAM session creation and termination operations, perform	pam_open_session(3)
PAM user policy definition service module	pam_updbe(5)
PAM, authentication information routines for PAM	pam_set_item(3)
PAM, get error message string	pam_strerror(3)
PAM, perform authentication within the PAM framework	pam_authenticate(3)
PAM, perform password related functions within the PAM framework	pam_chauthtok(3)
PAM, service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
PAM, service provider implementation for pam_authenticate()	pam_sm_authenticate(3)
PAM, service provider implementation for pam_chauthtok()	pam_sm_chauthtok(3)
PAM, service provider implementation for pam_open_session() and pam_close_session()	pam_sm_open_session(3)
PAM, service provider implementation for pam_setcred()	pam_sm_setcred(3)
pam.conf - configuration file for pluggable authentication module	pam.conf(4)
PAM; authentication transaction routines for	pam_start(3)
pam_acct_mgmt ; service provider implementation for	pam_sm_acct_mgmt(3)
pam_acct_mgmt() - perform PAM account validation procedures	pam_acct_mgmt(3)
pam_authenticate() - perform authentication within the PAM framework	pam_authenticate(3)
pam_authz - PAM module that provides user authorization	pam_authz(5)
pam_chauthtok() - perform password related functions within the PAM framework	pam_chauthtok(3)
pam_close_session() - perform PAM session creation and termination operations	pam_open_session(3)
pam_end() - authentication transaction routines for PAM	pam_start(3)
pam_get_data() - PAM routines to maintain module specific state	pam_set_data(3)
pam_get_item() - authentication information routines for PAM	pam_set_item(3)

Index

All Volumes

Description	Entry Name(Section)
pam_get_user() - PAM routine to retrieve user name	pam_get_user(3)
pam_hpsec - extended authentication, account, password, and session service module for HP-UX	pam_hpsec(5)
pam_ldap - authentication, account, session, and password management PAM modules for LDAP	pam_ldap(5)
pam_open_session() - perform PAM session creation and termination operations	pam_open_session(3)
pam_set_data() - PAM routines to maintain module specific state	pam_set_data(3)
pam_set_item() - authentication information routines for PAM	pam_set_item(3)
pam_setcred() - modify and delete user credentials for an authentication service	pam_setcred(3)
pam_sm() - PAM Service Module APIs	pam_sm(3)
pam_sm_acct_mgmt() - service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
pam_sm_acct_mgmt() - service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
pam_sm_authenticate() - service provider implementation for pam_authenticate()	pam_sm_authenticate(3)
pam_sm_chauthtok() - service provider implementation for pam_chauthtok()	pam_sm_chauthtok(3)
pam_sm_close_session() - service provider implementation for pam_close_session()	pam_sm_close_session(3)
pam_sm_open_session() - service provider implementation for pam_open_session()	pam_sm_open_session(3)
pam_sm_setcred() - service provider implementation for pam_setcred()	pam_sm_setcred(3)
pam_start() - authentication transaction routines for PAM	pam_start(3)
pam_strerror() - get PAM error message string	pam_strerror(3)
pam_unix - authentication, account, session and password management PAM modules for UNIX ..	pam_unix(5)
pam_updbe - user policy definition service module, PAM	pam_updbe(5)
pam_user.conf - user configuration file for pluggable authentication modules	pam_user.conf(4)
panic occurs on Integrity systems; enable or disable option for system to dump memory using multiple dump units when a kernel	dump_concurrent_on(5)
panic occurs, defines which classes of kernel memory pages are not dumped when a kernel	dontdump(5)
panic occurs; selects whether the system dumps memory pages compressed or uncompressed when a kernel	dump_compress_on(5)
parallel remote commands; return streams to	prcmd(3N)
parameter values for automountd daemon and automount command; file containing	autofs(4)
parameter, get value of kernel tunable parameter	gettune(2)
parameter; set the value of a kernel tunable	settone(2)
parameters in a transaction; sets the values of kernel tunable	settone_txn(2)
parameters, configure network interface	ifconfig(1M)
parameters, display system	sysdef(1M)
parameters; retrieve detailed information about kernel tunable	tuneinfo2(2)
parcreate - create a new partition	parcreate(1M)
parent directory	glossary(9)
parent process	glossary(9)
parent process	fork(2)
parent process ID	glossary(9)
parent process ID, get	getpid(2)
parents or children; synchronize a window with its	syncok(3X)
parmodify - modify an existing partition	parmodify(1M)
parolrad - online activation of a cell from nPartition; cancel online cell operation; monitor online cell operation; reset hung cell during cell activation	parolrad(1M)
parremove - remove an existing partition	parremove(1M)
parse command options	getopt(1)
parse suboptions from a string	getsubopt(3C)
parse utility (command) options	getopts(1)
parser used by DDFA software, dedicated ports	dpp(1M)
parstatus - display information about a hardware partitionable complex	parstatus(1)
particular Network Information Service server; bind to	ypset(1M)
Partition Command Line Interface; display information about the	partition(5)
partition configuration data; unlock stable complex profile or cancel pending changes to complex or	parunlock(1M)
partition DDS tape; initialize disk or	mediainit(1)
partition - display information about the Partition Command Line Interface	partition(5)

Description	Entry Name(Section)
partition; create a new	parcreate(1M)
partition; modify an existing	parmodify(1M)
partition; remove an existing	parremove(1M)
partitionable complex; display information about a hardware	parstatus(1)
parunlock - unlock stable complex profile or cancel pending changes to complex or partition configuration data	parunlock(1M)
pass-through daemon for processing system commands	fsdaemon(1M)
pass-through device driver, SCSI	sioc_io(7)
pass-through driver (esctl/sctl); SCSI	scsi_ctl(7)
passes control to the libmilter event loop	smfi_main(3N)
passwd - change login password and associated attributes	passwd(1)
passwd database; daemon for modifying Network Information Service	yppasswdd(1M)
passwd file	login(1)
passwd file; change default login shell	chsh(1)
passwd - password file, <pwd.h> format	passwd(4)
password	glossary(9)
password and associated attributes; change login	passwd(1)
password and group hashing and caching statistics	pwgr_stat(1M)
password and group hashing and caching daemon	pwgrd(1M)
password authentication database for trusted systems; protected	prpwd(4)
password database entries (for trusted systems only); manipulate protected	getprpwent(3)
password database; display protected	getprpw(1M)
password database; modify protected	modprpw(1M)
password encryption function	crypt(3C)
password entries; access shadow	getspent(3C)
password expiration	passwd(1)
password file	passwd(4)
password file entry on trusted systems; get secure	getspwent(3X)
password file entry, write shadow	putspent(3C)
password file entry; get	getpwent(3C)
password file entry; write	putpwent(3C)
password file, <pwd.h> format	passwd(4)
password file, edit using vi editor	vipw(1M)
password file, group	ftpgroups(4)
password file, grp.h for user group access and identification	group(4)
password file; check	pwck(1M)
password file; shadow	shadow(4)
password generation	passwd(1)
password in Network Information Service, update user	yppasswd(3N)
password in Network Information System (NIS), change login	yppasswd(1)
password information, get (pwget)	pwget(1)
password management PAM modules for LDAP; authentication, account, session, and	pam_ldap(5)
password related functions within the PAM framework, perform	pam_chauthtok(3)
password, and session service module for HP-UX, extended authentication, account,	pam_hpsec(5)
password, authentication, account, and session management PAM modules for UNIX	pam_unix(5)
password, change login	passwd(1)
password, read from terminal while suppressing echo	getpass(3C)
password/group file checkers	pwck(1M)
password; change a user's Kerberos	kpasswd(1)
passwords from shadow to nonshadow, convert	pwunconv(1M)
paste - merge corresponding lines of several files or subsequent lines of one file	paste(1)
patch check utility; HP-UX 11i V3	check_patches(1M)
patch cleanup utility; HP-UX	cleanup(1M)
patch display utility; HP-UX	show_patches(1)
patch - program to apply a diff file to an original file	patch(1)
patch up damaged file system (generic)	fsdb(1M)
patch up damaged HFS file system	fsdb_hfs(1M)
path and route between hosts, compute shortest	pathalias(1)
PATH environment variable	login(1)
path name	glossary(9)
path name component	basename(3C)

Index

All Volumes

Description	Entry Name(Section)
path name corresponding to i-number, list	ff_hfs(1M)
path name of an open file, get the full	pstat(2)
path name of parent directory	basename(3C)
path name resolution	glossary(9)
path name variables, get configurable	pathconf(2)
path names from i-numbers, generate	ncheck(1M)
path names of all FTP configuration files	ckconfig(1)
path names, check	pathchk(1)
path names, extract portions of	basename(1)
path prefix	glossary(9)
path-name of current working directory, get	getcwd(3C)
path; create, remove directories in a	mkdirp(3G)
path; map device ID to file	devnm(3)
pathalias database, access and manage the	uupath(1)
pathalias - electronic address router	pathalias(1)
pathchk - check path names	pathchk(1)
pathconf() - get configurable path name variables	pathconf(2)
pathfind() - search for named file in named directories	pathfind(3G)
pathname of current working directory, get	getwd(3C)
pathname; resolve	realpath(3X)
pathnames; print kernel configuration	kcpath(1M)
paths; locate a program file including aliases and	which(1)
pattern matching and regular expression notation definitions	regex(5)
pattern-directed scanning and processing language	awk(1)
patterns, match filename	fnmatch(3C)
pause execution for a time interval	sleep(1)
pause() - suspend process until signal	pause(2)
pax - extracts, writes, and lists archive files; copies files and directory hierarchies	pax(1)
pax - portable archive exchange	pax(1)
PC-NFS authentication and print request server	pcnfsd(1M)
pcat - expand (unpack) and cat Huffman-coded file	pack(1)
pcf - port configuration file, used by DDFA software	pcf(4)
PCI Error Recovery; enables/disables	pci_ah_enable(5)
PCI Error Recovery; time interval, in minutes, between two PCI errors at a I/O slot that will result in automatic	pci_error_tolerance_time(5)
PCI errors at a I/O slot that will result in automatic PCI Error Recovery; time interval, in minutes, between two	pci_error_tolerance_time(5)
PCI I/O cards and Online Addition of I/O chassis; command for Online Addition/Replacement/Deletion of	olrad(1M)
PCI I/O hotplug (attention button) events daemon	hotplugd(1M)
PCI Vendor/Device ID that the gvid graphics driver will not claim	gvid_no_claim_dev(5)
pci_ah_enable - enables/disables PCI Error Recovery	pci_ah_enable(5)
pci_error_tolerance_time - time interval, in minutes, between two PCI errors at a I/O slot that will result in automatic PCI Error Recovery	pci_error_tolerance_time(5)
pckt - Packet Mode module for STREAMS pty	pckt(7)
pclose() - terminate pipe I/O to or from a process	popen(3S)
pcnfsd - PC-NFS authentication and print request server	pcnfsd(1M)
pcserver - Basic Serial and HP AdvanceLink server	pcserver(1M)
pdcc - processor-dependent code (firmware)	pdcc(1M)
pdp11 - is processor a PDP 11?	machid(1)
pdweb - start the HP-UX Peripheral Device tool, part of the SMH Web interface	pdweb(1M)
pecho_wchar() - write a character rendition and immediately refresh the pad	pechochar(3X)
pechochar() - write a character rendition and immediately refresh the pad	pechochar(3X)
peer; get address of connected	getpeername(2)
pending cancellation requests, process any	pthread_testcancel(3T)
pending signals, examine	sigpending(2)
per identifier, maximum number of System V IPC semaphores	semmsl(5)
per system-wide limit; percentage of file cache that can be consumed by sequential accesses,	fcache_seqlimit_system(5)
per-file limit; percent of file cache that can be consumed by sequential accesses,	fcache_seqlimit_file(5)

Description	Entry Name(Section)
per-mechanism information about a credential, provide the calling application ..	gss_inquire_cred_by_mech(3)
per-process accounting file format	acct(4)
per-process file descriptor table; get the size of the	getdtablesize(2)
per-process information of a ccNUMA system, returns system-wide or	pstat_getlocality(2)
per-process timer, allocate a	mktimer(3C)
per-process timer, free a	rmtimer(3C)
per-process timer, get value of a	gettimer(3C)
per-process timer, relatively arm a	reltimer(3C)
per-session accounting records, convert login/logoff records to	acctcon(1M)
per-session records, convert to total accounting records	acctcon(1M)
per-user information; user database for	userdb(4)
percent of file cache that can be consumed by sequential accesses, per-file limit	fcache_seqlimit_file(5)
percentage of all process threads allowed in AIO pool	aio_proc_thread_pct(5)
percentage of file cache that can be consumed by sequential accesses, per system-wide limit	fcache_seqlimit_system(5)
percentage of physical memory lockable for request call-back POSIX asynchronous I/O	aio_phymem_pct(5)
percentage of physical memory that can be used by audit subsystem	audit_memory_usage(5)
perform a data integrity check on an event	EvmEventValidate(3)
perform authentication within the PAM framework	pam_authenticate(3)
perform Cyclical Redundancy Check on a file	sum(1)
perform I/O of EVM events to and from a file	EvmEventRead(3)
perform I/O on kernel memory, based on symbol name	kmem(7)
perform PAM account validation procedures	pam_acct_mgmt(3)
perform PAM session creation and termination operations	pam_open_session(3)
perform password related functions within the PAM framework	pam_chauthtok(3)
perform word expansions	wordexp(3C)
performance analysis information; display LP spooler	lpana(1M)
performance data from remote kernel, get	rstat(3N)
performing appropriate authorization checks and optionally reauthenticating the user; invoke another application with privileges after	privrun(1M)
Peripheral Device tool, part of the SMH Web interface; start the HP-UX	pdweb(1M)
permission bits	glossary(9)
permissions file, check the uucp directories and	uuccheck(1M)
permissions mask for file creation, set and get	umask(2)
permissions mode mask for file-creation, set access	umask(1)
permissions; change file mode access	chmod(1)
permissions; change file mode access	chmod(2)
permit or deny <i>write</i> (1) messages from other users to terminal	mesg(1)
permuted index, generate	ptx(1)
perorr() - write system error messages	perorr(3C)
persistent device special file	glossary(9)
persistent device special file; redirect from one device to a different device	io_redirect_dsf(1M)
perusal filter for screen viewing; file	more(1)
peruse file on soft-copy terminals	pg(1)
pfdat_hash_locks - OBSOLETE kernel tunable parameter	pfdat_hash_locks(5)
pfiles - displays process address information and open file descriptors	pmmap(1)
pfmt() - display message in standard format	pfmt(3C)
pg - file perusal filter for soft-copy terminals	pg(1)
pgrep - search processes based on process name and attributes	pgrep(1)
phone message transcription system	answer(1)
physical device description file format; PPP	ppp.Devices(4)
physical environment daemon; system	envd(1M)
physical extents allocated to LVM logical volume, decrease	lvreduce(1M)
physical extents, move from one LVM physical volume to other physical volumes	pvmove(1M)
physical memory address mapping	iomap(7)
physical memory allocation policy on cell-based HP-UX servers	numa_policy(5)
physical memory lockable for request call-back POSIX asynchronous I/O operations; percentage of	aio_phymem_pct(5)
physical memory that can be used by audit subsystem; percentage of	audit_memory_usage(5)
physical memory used for caching file I/O data; maximum or minimum amount of	filecache_max(5)

Index

All Volumes

Description	Entry Name(Section)
physical page number, validate whether dumped	cr_isaddr(3)
physical volume for use in LVM volume group; create	pvcreate(1M)
physical volume group information file (LVM)	lvmpvg(4)
physical volume in LVM volume group, change characteristics and access path	pvchange(1M)
physical volume in LVM volume group, check or repair a	pvck(1M)
physical volume in LVM volume group, remove	pvremove(1M)
physical volume (LVM) to other physical volumes, move allocated physical extents from	pvmove(1M)
physical volumes for LVM volume groups; scan	vgscan(1M)
physical volumes from an LVM volume group; remove	vgreduce(1M)
physical volumes, extend an LVM volume group by adding	vgextend(1M)
physical_io_buffers - total physical I/O buffers	physical_io_buffers(5)
PIC	glossary(9)
PIDs; limit the maximum value for	process_id_max(5)
PIDs; specify minimum value	process_id_min(5)
ping - send echo packets to a network host	ping(1M)
pipcrm - remove a POSIX message queue, semaphore name	pipcrm(1)
pipcs - report status of POSIX interprocess communication facilities	pipcs(1)
pipe	glossary(9)
pipe() - create an interprocess channel	pipe(2)
pipe fitting to copy standard output to file	tee(1)
pipe I/O to or from a process, open or close	popen(3S)
pipe: STREAMS-based pipe	isastream(3C)
pipes to be STREAMS-base, force all	streampipes(5)
pipes, create named	mknod(1M)
pkill - kill processes based on process name and attributes	pgrep(1)
play an audio file	send_sound(1)
pldd - list the dynamic libraries linked into each process, including shared objects explicitly attached using dlopen()/shl_load()	pldd(1)
plock() - lock process, text, data, stack, or shared library in memory	plock(2)
plotdvr - plotter driver for lp	lpfilter(1)
plotter	<i>see printer</i>
plug-in for scsimgr; SCSI class driver eschgr	scsimgr_eschgr(7)
plug-in for scsimgr; SCSI class driver esdisk	scsimgr_esdisk(7)
plug-in for scsimgr; SCSI class driver estape	scsimgr_estape(7)
pluggable authentication module	pam(3)
pluggable authentication module, configuration file	pam.conf(4)
Pluggable Authentication Modules	login(1)
pluggable authentication modules; user configuration file for	pam_user.conf(4)
pmmap - displays process address information and open file descriptors	pmmap(1)
pmmap_getmaps() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_getport() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_rmtcall() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_set() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_unset() - obsolete library routines for RPC	rpc_soc(3N)
pnoutrefresh() - pad management functions	newpad(3X)
Point Protocol over Ethernet relay, PPPoE (Point to	pppoerd(1M)
point to point protocol daemon; PPP	pppd(1)
Point to Point Protocol over Ethernet client configuration file, PPPoE	pppoe.conf(4)
Point to Point Protocol over Ethernet client, PPPoE	pppoe(1)
Point to Point Protocol over Ethernet relay configuration file, PPPoE	pppoerd.conf(4)
Point to Point Protocol over Ethernet server configuration file, PPPoE	pppoesd.conf(4)
Point-to-Point Protocol over Ethernet server daemon, PPPoE	pppoesd(1M)
pointer array, sort a directory	scandir(3C)
pointer for binary search tree, get data	tsearch(3C)
pointer for I/O operations on a stream file, get or reposition	fseek(3S)
pointer, file, move read/write	lseek(2)
pointer, stream, map to file descriptor	fileno(3S)
pointer, string, for ELF files, make	elf_strptr(3E)
points and convenience macros, overview of stack unwind library entry	unwind(5)
points; install automatic mount	automount(1M)
policy for flush behind requests from VxFS file system	fcache_fb_policy(5)

Description	Entry Name(Section)
poll() - monitor I/O conditions on multiple file descriptors	poll(2)
poll - monitor I/O conditions on multiple file descriptors	poll(7)
pool size, System V IPC hashed spinlock	sysv_hash_locks(5)
pool; the amount of memory to reserve for the 32-bit DMA	dma32_pool_size(5)
popd - pop directory stack	csh(1)
popen() - initiate pipe I/O to or from a process	popen(3S)
port number, RPC, get	getrpcport(3N)
port socket, return a reserved	rcmd(3N)
port; set the options for a terminal	stty(1)
portable archive exchange	pax(1)
portable archives, library and archive maintainer for	ar(1)
portable file name character set	glossary(9)
portal - header file for future applications	portal(5)
portions of path names, extract	basename(1)
ports file used by DDFA software and Telnet port identification feature; dedicated	dp(4)
ports parser used by DDFA software, dedicated	dpp(1M)
position-independent code (PIC)	glossary(9)
positive difference functions	fdim(3M)
POSIX async I/O operations, maximum number of, that can be queued at any time	aio_max_ops(5)
POSIX async I/O request priorities, greatest delta (slowdown factor) allowed	aio_prio_delta_max(5)
POSIX asynchronous I/O	aio(5)
POSIX asynchronous I/O operations, maximum allowed in a listio call	aio_listio_max(5)
POSIX asynchronous I/O operations, percentage of physical memory lockable for request call-back	aio_physmem_pct(5)
POSIX asynchronous I/O operations; percentage of physical memory lockable for request call-back	aio_physmem_pct(5)
POSIX configuration values, get	getconf(1)
POSIX interprocess communication facilities; report status	pipcs(1)
POSIX message queue, get information for a	pstat(2)
POSIX named semaphore, get information for a	pstat(2)
POSIX queue name, remove	pipcrm(1)
POSIX real-time priority, execute process with	rtsched(1)
POSIX realtime extensions	aio(5)
POSIX realtime extensions	aio_cancel(2)
POSIX realtime extensions	aio_error(2)
POSIX realtime extensions	aio_fsync(2)
POSIX realtime extensions	aio_read(2)
POSIX realtime extensions	aio_return(2)
POSIX realtime extensions	aio_suspend(2)
POSIX realtime extensions	aio_write(2)
POSIX realtime extensions	lio_listio(2)
POSIX realtime extensions	mlock(2)
POSIX realtime extensions	mlockall(2)
POSIX realtime extensions	munlock(2)
POSIX realtime extensions	munlockall(2)
POSIX semaphore, close a named semaphore	sem_close(2)
POSIX semaphore, destroy an unnamed semaphore	sem_destroy(2)
POSIX semaphore, initialize an unnamed semaphore	sem_init(2)
POSIX semaphore, lock a semaphore	sem_wait(2)
POSIX semaphore, lock a semaphore without blocking	sem_wait(2)
POSIX semaphore, open/create a named semaphore	sem_open(2)
POSIX semaphore, read	sem_getvalue(2)
POSIX semaphore, unlink a named semaphore	sem_unlink(2)
POSIX semaphore, unlock a semaphore	sem_post(2)
POSIX.1b realtime applications, number of priority values to support for	rtsched_numpri(5)
POSIX.1c threads	pthread(3T)
POSIX.2-conformant command shells; standard and restricted	sh-posix(1)
posix_fadvise() - file advisory information	posix_fadvise(2)
posix_fadvise64() - file advisory information	posix_fadvise(2)
posix_openpt() - open a pseudo-terminal master device	posix_openpt(3C)
post an EVM event	EvmEventPost(3)

Index

All Volumes

Description	Entry Name(Section)
post events to the EVM daemon	evmpost(1)
posters, make in large letters	banner(1)
PostScript print filter; internationalized	psfontpf(1M)
postwait - lightweight synchronization mechanism	postwait(2)
pow() - power function	pow(3M)
power for cells and I/O chassis; turn on/off or display current status of	frupower(1M)
power functions	pow(3M)
power_onoff - timed, automatic system power on, and power off	power_onoff(1M)
powf() - power function (float)	pow(3M)
powl() - power function (long double)	pow(3M)
powlln() - power function (double,long long)	pow(3M)
powllnf() - power function (float,long long)	pow(3M)
powllnl() - power function (long double,long long)	pow(3M)
powllnq() - power function (quad,long long)	pow(3M)
powllnw() - power function (extended,long long)	pow(3M)
powm() - power function (double,int)	pow(3M)
powmf() - power function (float,int)	pow(3M)
powml() - power function (long double,int)	pow(3M)
powmq() - power function (quad,int)	pow(3M)
powmw() - power function (extended,int)	pow(3M)
powq() - power function (quad)	pow(3M)
poww() - power function (extended)	pow(3M)
PPP authentication file format	ppp.Auth(4)
PPP dialer description file format	ppp.Dialers(4)
PPP encryption keys file format	ppp.Keys(4)
PPP neighboring systems description file format	ppp.Systems(4)
PPP packet filter specification file format	ppp.Filter(4)
PPP physical device description file format	ppp.Devices(4)
PPP point to point protocol daemon	pppd(1)
ppp.Auth - PPP authentication file format	ppp.Auth(4)
ppp.Devices - PPP physical device description file format	ppp.Devices(4)
ppp.Dialers - PPP dialer description file format	ppp.Dialers(4)
ppp.Filter - PPP packet filter specification file format	ppp.Filter(4)
ppp.Keys - PPP encryption keys file format	ppp.Keys(4)
ppp.Keys - ppp encryption keys file format	ppp.Keys(4)
ppp.Systems - PPP neighboring systems description file format	ppp.Systems(4)
pppd - PPP point to point protocol daemon	pppd(1)
PPPoE (Point to Point Protocol over Ethernet) client	pppoe(1)
PPPoE (Point to Point Protocol over Ethernet) client configuration file	pppoe.conf(4)
PPPoE (Point to Point Protocol over Ethernet) relay	pppoerd(1M)
PPPoE (Point to Point Protocol over Ethernet) relay configuration file	pppoerd.conf(4)
PPPoE (Point to Point Protocol over Ethernet) server configuration file	pppoesd.conf(4)
PPPoE (Point-to-Point Protocol over Ethernet) server daemon	pppoesd(1M)
pppoe - PPPoE (Point to Point Protocol over Ethernet) client	pppoe(1)
pppoe.conf - PPPoE (Point to Point Protocol over Ethernet) client configuration file	pppoe.conf(4)
pppoerd - PPPoE (Point to Point Protocol over Ethernet) relay	pppoerd(1M)
pppoerd.conf - PPPoE (Point to Point Protocol over Ethernet) relay configuration file	pppoerd.conf(4)
pppoesd - PPPoE (Point-to-Point Protocol over Ethernet) server daemon	pppoesd(1M)
pppoesd.conf - PPPoE (Point to Point Protocol over Ethernet) server configuration file	pppoesd.conf(4)
pr - format and print files	pr(1)
praliases - print system-wide sendmail aliases	praliases(1)
prcmd() - return streams to parallel remote commands	prcmd(3N)
prctmp - print session record file created by acctcon1	acctsh(1M)
prdaily - print daily accounting report	acctsh(1M)
pread() - read contiguous data from a position in a file	pread(2)
pread() - read from file	read(2)
pread64() - non-POSIX standard API interfaces to support large files	creat64(2)
prealloc - preallocate disk storage	prealloc(1)

Description	Entry Name(Section)
prealloc() - preallocate fast disk storage	prealloc(2)
prealloc64() - non-POSIX standard API interfaces to support large files	creat64(2)
preallocate fast disk storage	prealloc(2)
preallocate space for a disk storage file	prealloc(1)
prefresh() - pad management functions	newpad(3X)
prepare an incomplete executable for faster program start-up	fastbind(1)
prepare execution profile	monitor(3C)
prepare LVM logical volume to be root, boot, primary swap, or dump volume	lvlboot(1M)
prepends a header to the current sendmail message	smfi_insheder(3N)
preprocess a message source file	mkcatdefs(1)
preprocess tables for nroff	tbl(1)
preprocessor lines, remove	unifdef(1)
preprocessor, C language	cpp(1)
present value factor for annuity	annuity(3M)
preset contents of memory area to specified byte	memory(3C)
prevent terminal use by others	lock(1)
previous get of an SCCS file, undo a	unget(1)
PRI_HPUX_TO_POSIX() - return POSIX process priority	rtsched(2)
PRI_POSIX_TO_HPUX() - return HP-UX process priority	rtsched(2)
primary swap volume, remove LVM logical volume link	lvrmboot(1M)
primary swap, or dump volume; prepare LVM logical volume to be root, boot,	lvlboot(1M)
primes, factor - factor a number, generate large primes	factor(1)
primitive system data types	types(5)
principal name to a server, send a	rpc_gss_set_svc_name(3N)
principal names at server, get	rpc_gss_get_principal_name(3N)
principals; print key version numbers of Kerberos	kvno(1)
print a libcrash error or warning message	cr_perror(3)
print a stack trace for each LWP in each process and core file	pstack(1)
print all values in a Network Information Service map	ypcat(1)
print and summarize an SCCS file	prs(1)
print arguments; format and	printf(1)
print calendar	cal(1)
print checksum and block count of a file	sum(1)
print checksum and block count of a file	sum(1)
print current SCCS file editing activity	sact(1)
print (echo) arguments	echo(1)
print effective current user ID	whoami(1)
print files, format and	pr(1)
print filter; internationalized PostScript	psfontpf(1M)
print first few lines in a file	head(1)
print formatted arguments	printf(1)
print formatted output	printf(3S)
print formatted output in a window	vw_printf(3X)
print formatted output in a window	vwprintf(3X)
print formatted output in window	mvprintf(3X)
print formatted output of a varargs argument list	vprintf(3S)
print formatted wide-character output	fwprintf(3C)
print hardware model information	model(1)
print key version numbers of Kerberos principals	kvno(1)
print list of current system users	who(1)
print log messages and other information on RCS files	rlog(1)
print mail traffic statistics	mailstats(1)
print name list of object code file	nm(1)
print news items	news(1)
print or check documents formatted with the mm macros	mm(1)
print out a manpage	man(1)
print out mail in the incoming mailbox file	prmail(1)
print out terminfo descriptions; compare or	infocmp(1M)
print out the environment	printenv(1)
print - output from shell	ksh(1)
print - output from shell	sh-posix(1)

Index

All Volumes

Description	Entry Name(Section)
print process accounting files; search and	acctcom(1M)
print request server; PC-NFS	pcnfsd(1M)
print requests on an LP printer	lp(1)
print section sizes and allocation space of object files	size(1)
print status of LP spooler requests on a remote system	rlpstat(1M)
print symbol table for object code file	nm(1)
print system-wide sendmail aliases	praliases(1)
print the values of selected keys in Network Information Service map	ypmatch(1)
print user and group IDs and names	id(1)
print, copy, and concatenate files	cat(1)
printable representation of a character, generate	unctrl(3X)
printable representation of a wide character, generate	wunctrl(3X)
printable strings in an object or other binary file, find the	strings(1)
printenv - print out the environment	printenv(1)
printer capability database	terminfo(4)
printer daemon for LP requests from remote systems	rlpdaemon(1M)
printer device files, line	lp(7)
printer for use with tsm ; add or remove a	tsm.lpadmin(1M)
printer handling and optimization package; terminal and	curses_intro(3X)
printer, LP, allow or prevent queuing requests	accept(1M)
printer, LP, print/alter/cancel requests	lp(1)
printer, set printing options for a non-serial	slp(1)
printers, LP, enable/disable	enable(1)
printf - format and print arguments	printf(1)
printf() - print formatted output to standard output	printf(3S)
printing options for a non-serial printer, set	slp(1)
printing; define the minimum priority for	lpsched(1M)
prints the mail queue	mailq(1)
prints the process tree hierarchy	ptree(1)
printstat - check status of serial printer	lpfilter(1)
printw() - print formatted output in window	mvprintw(3X)
prioceling attribute, get or set	pthread_mutexattr_getprotocol(3T)
priority for printing; define the minimum	lpsched(1M)
priority of a process, change	nice(2)
priority of a thread; sets scheduling	pthread_setschedprio(3T)
priority of running processes, alter	renice(1M)
priority values to support for POSIX.1b realtime applications, number of	rtsched_numpri(5)
priority, get or set process	getpriority(2)
priority, POSIX real-time, execute process with	rtsched(1)
priority, real-time, execute process with	rtprio(1)
priority, run a command at nondefault	nice(1)
priv_add - add, set, remove, and retrieve a process' privileges	priv_add(3)
priv_add_effective() - add, set, remove, and retrieve a process' privileges	priv_add(3)
priv_addset() - privilege manipulation operations for checking and debugging purposes	privileges(3)
priv_delset() - privilege manipulation operations for checking and debugging purposes	privileges(3)
priv_get() - add, set, remove, and retrieve a process' privileges	priv_add(3)
priv_getbyname() - convert privilege name to privilege ID	priv_getbyname(3)
priv_getbynum() - convert privilege ID to privilege name	priv_getbynum(3)
priv_ismember() - privilege manipulation operations for checking and debugging purposes	privileges(3)
priv_isobserved() - privilege manipulation operations for checking and debugging purposes	privileges(3)
priv_remove() - add, set, remove, and retrieve a process' privileges	priv_add(3)
priv_set_effective() - add, set, remove, and retrieve a process' privileges	priv_add(3)
priv_set_to_str() - privilege name to set conversion function	priv_str_to_set(3)
priv_str_to_set() - privilege name to set conversion function	priv_str_to_set(3)
private data pointer for the sendmail connection; sets the	smfi_setpriv(3N)
private encryption key storage, server for	keyserv(1M)
privatepw - change WU-FTPD group access file information	privatepw(1)

Description	Entry Name(Section)
privedit - let authorized users edit files that are under access control	privedit(1M)
privgrp () - format of privileged values	privgrp(4)
privgrp - HP-UX group privileges	privgrp(5)
privgrp, get special attributes for group	getprivgrp(1)
privilege group, get or set special attributes	getprivgrp(2)
privilege ID to privilege name; convert	priv_getbyname(3)
privilege ID; convert privilege name to	priv_getbyname(3)
privilege information in the privrun database; noninteractive editing of a command's authorization and	cmdprivadm(1M)
privilege manipulation operations for checking and debugging purposes	privileges(3)
privilege name to privilege ID; convert	priv_getbyname(3)
privilege name to set conversion function	priv_str_to_set(3)
privilege name; convert privilege ID to	priv_getbyname(3)
privileged groups	glossary(9)
privileged values; format of	privgrp(4)
privileges after performing appropriate authorization checks and optionally reauthenticating the user; invoke another application with	privrun(1M)
privileges - description of HP-UX privileges	privileges(5)
privileges for group, list access	getprivgrp(1)
privileges for groups; set special	setprivgrp(1M)
privileges - privilege manipulation operations for checking and debugging purposes	privileges(3)
privileges; add, set, remove, and retrieve a process'	priv_add(3)
privileges; HP-UX group	privgrp(5)
privrun database; noninteractive editing of a command's authorization and privilege information in the	cmdprivadm(1M)
privrun - invoke another application with privileges after performing appropriate authorization checks and optionally reauthenticating the user	privrun(1M)
privset_add () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_add_effective () - add, set, remove, and retrieve a process' privileges	priv_add(3)
privset_alloc () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_copy () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_del () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_empty () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_fill () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_free () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_get () - add, set, remove, and retrieve a process' privileges	priv_add(3)
privset_intersect () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_inverse () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_isempty () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_isequal () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_isfull () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_ismember () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_issubset () - privilege manipulation operations for checking and debugging purposes	privileges(3)
privset_remove () - add, set, remove, and retrieve a process' privileges	priv_add(3)
privset_set_effective () - add, set, remove, and retrieve a process' privileges	priv_add(3)
privset_subtract () - privilege manipulation operations for checking and debugging purposes	privileges(3)

Index

All Volumes

Description	Entry Name(Section)
privset_union() - privilege manipulation operations for checking and debugging purposes	
PRM	privileges(3)
PRM	see <i>Process Resource Manager</i>
PRM	see <i>Process Resource Manager</i>
prmail - print out mail in the incoming mailbox file	prmail(1)
procedure call stack using the unwind library, produce a trace back of the	U_STACK_TRACE(3X)
procedure calls, library routines for secure remote	secure_rpc(3N)
procedures; perform PAM account validation	pam_acct_mgmt(3)
process	glossary(9)
process 1	glossary(9)
process accounting files; search and print	acctcom(1M)
process accounting, daily accounting shell procedure	runacct(1M)
process accounting, enable or disable	acct(2)
process address information and open file descriptors; displays	pmap(1)
process and child process times; get	times(2)
process and core file; print a stack trace for each LWP in each	pstack(1)
process any pending cancellation requests	pthread_testcancel(3T)
process C language include and conditional instructions	cpp(1)
process control facilities, 4.2 BSD-compatible	killpg(2)
process control initialization	init(1M)
process core dump, determines the inclusion of readable shared memory in a	core_addshmem_read(5)
process environment, clear the	clearenv(3C)
process flag for calling process; get audit	getaudproc(2)
process group	glossary(9)
process group ID	glossary(9)
process group ID for job control, set	setpgid(2)
process group ID, create session and set	setsid(2)
process group ID, foreground, get	tcgetpgrp(3C)
process group ID, foreground, set	tcsetpgrp(3C)
process group ID, get	getpid(2)
process group ID; set	setpgrp(2)
process group ID; set	setsid(2)
process group leader	glossary(9)
process group lifetime	glossary(9)
process ID	glossary(9)
process ID, get	getpid(2)
process IDs (PIDs); limit the maximum value for	process_id_max(5)
process IDs (PIDs); specify minimum value	process_id_min(5)
process information for each ftp user	ftpwho(1)
process interval timer, set or get value of	getitimer(2)
process lifetime	glossary(9)
process mail through screen-oriented interface	elm(1)
process name and attributes; search or kill processes based on	pgrep(1)
process or group of processes; send a signal to	kill(2)
process or thread to a processor set; bind	pset_bind(2)
process priority, get or set	getpriority(2)
Process Resource Manager	acctcom(1M)
Process Resource Manager	cron(1M)
Process Resource Manager	login(1)
Process Resource Manager	pgrep(1)
Process Resource Manager	ps(1)
process running under the PA-RISC emulator on an Integrity system; maximum size (in bytes) of the stack for a user	pa_maxssiz(5)
process status; report	ps(1)
process threads allowed in AIO pool, maximum number of	aio_proc_threads(5)
process to run serially with other processes; force target	serialize(1)
process to run serially with other processes; force target	serialize(2)
process to stop or terminate; wait for child	wait(2)
process' privileges; add, set, remove, and retrieve a	priv_add(3)
process's address space, get information for a	pstat(2)

Description	Entry Name(Section)
process's alarm clock, set	alarm(2)
process's expected paging behavior, advise system of	madvise(2)
process, change priority of a	nice(2)
process, child, wait to change state	wait3(2)
process, child, wait to change state	wait4(2)
process, child, wait to change state	waitid(2)
process, create a new	fork(2)
process, defines the maximum number of threads allowed per	max_thread_proc(5)
process, execute, with POSIX real-time priority	rtsched(1)
process, execute, with real-time priority	rtprio(1)
process, get command line of a	pstat(2)
process, get information for a	pstat(2)
process, get information for a thread or LWP in a	pstat(2)
process, get information for an open file of a	pstat(2)
process, limit on number of queued signals per	ksi_send_max(5)
process, lock address space in memory	mlockall(2)
process, lock in memory	plock(2)
process, lock into memory after allocating data and stack space	datalock(3C)
process, lock segment in memory	mlock(2)
process, maximum number of System V IPC undo entries per	semume(5)
process, maximum number of System V shared memory segments per	shmseg(5)
process, open or close pipe I/O to or from a	popen(3S)
process, send signal to	kill(1)
process, spawn new (use fork() instead)	vfork(2)
process, suspend the calling process	napms(3X)
process, suspend until signal	pause(2)
process, terminate	kill(1)
process, unlock memory segment	munlock(2)
process, unlock virtual address space	munlockall(2)
process-shared attribute, get or set	pthread_condattr_getpshared(3T)
process-shared attribute, get or set	pthread_mutexattr_getpshared(3T)
process-shared attribute, get or set	pthread_rwlockattr_getpshared(3T)
process-wide mutex handoff mode; disable mutex-specific or	pthread_mutexattr_getspin_np(3T)
process; controls process level auditing	setaudproc(2)
process; display security attributes of a	getprocxsec(1M)
process; get audit process flag for calling	getaudproc(2)
process; get the audit ID (aid) for the current	getaudit(2)
process; maximum size (in bytes) of the stack for any user	maxssiz(5)
process; queue a signal to a	sigqueue(2)
process; set the audit ID (aid) for the current	setaudit(2)
process; suspend or resume auditing on the current	audswitch(2)
process; terminate a	exit(2)
process; write audit record for self-auditing	audwrite(2)
process_id_max - limit the maximum value for process IDs (PIDs)	process_id_max(5)
process_id_min - specify minimum value for process IDs (PIDs)	process_id_min(5)
processes and users, list current	whodo(1M)
processes based on process name and attributes; search or kill	pgrep(1)
processes from /etc/services.window ; extract window IDs of user	getmemwindow(1M)
processes per user, limits the maximum number of user	maxuprc(5)
processes to complete, wait for background	wait(1)
processes using a file or file structure, list	fuser(1M)
processes, alter priority of running	renice(1M)
processes, kill all active	killall(1M)
processes; force target process to run serially with other	serialize(1)
processes; force target process to run serially with other	serialize(2)
processes; Network Information Service (NIS) server, binder, and transfer	ypserv(1M)
processes; spawn	init(1M)
processing language; pattern-directed scanning and	awk(1)
processor for C, Ratfor and other programming language macros	m4(1)
processor IDs, determined	pthread_processor_bind_np(3T)
processor initialization	pd(1M)

Index

All Volumes

Description	Entry Name(Section)
processor self test	pdc(1M)
processor set assignment; change	pset_assign(2)
processor set attributes; manage	pset_getattr(2)
processor set control	pset_ctl(2)
processor set control; real-time	_pset_rtctl(2)
processor set, get information for a	pstat(2)
processor set; bind process or thread to a	pset_bind(2)
processor set; create	pset_create(2)
processor set; destroy	pset_destroy(2)
processor sets; create and manage	psrset(1M)
processor type; provide truth value about	machid(1)
processor, control, on which a specific process executes	mpsched(1)
processor, get information for a	pstat(2)
processor-dependent code (firmware)	pdc(1M)
processors, bind threads to	pthread_processor_bind_np(3T)
processors, how many available	pthread_processor_bind_np(3T)
produce a trace back of the procedure call stack using the unwind library	U_STACK_TRACE(3X)
produces a set of DNSSEC keys	dnssec-makekeyset(1)
product specification file (PSF) format	swpackage(4)
products; verify software	swverify(1M)
prof - display monitor profile data	prof(1)
profil() - execution time profile	profil(2)
profile audit status; change or display	audevent(1M)
profile data, display call graph execution	gprof(1)
profile data, display monitor	prof(1)
profile file	login(1)
profile of execution, prepare	monitor(3C)
profile or cancel pending changes to complex or ppartition configuration data; unlock stable complex	parunlock(1M)
profile - shell script to set up user's environment at login	profile(4)
profile, execution time	profil(2)
profile, prepare execution	smonitor(3C)
program	glossary(9)
program assertion, verify	assert(3X)
program file including aliases and paths; locate a	which(1)
program for SCCS commands	sccs(1)
program loaded module	dlget(3C)
program loaded module	dlgetmodinfo(3C)
program loaded module	dlmodinfo(3C)
program message, get an NLS	catgets(3C)
program number database, RPC	rpc(4)
program or shell terminal modes; save or restore	def_prog_mode(3X)
program regions; first locations beyond allocated	end(3C)
program stacks are executable by default, controls whether	executable_stack(5)
program start-up, prepare for faster	fastbind(1)
program termination, register a function to be called at	atexit(3)
program to apply a diff file to an original file	patch(1)
program's internal attributes on Integrity systems; change	chatr_ia(1)
program's internal attributes on PA-RISC systems; change	chatr_pa(1)
program's internal attributes; change	chatr(1)
program, get or set the locale of a	setlocale(3C)
program; last locations in	end(3C)
program; remote file distribution	rdist(1)
program; special NTP query	xntpdc(1M)
program; terminal identification	ttytype(1)
Programming Interface; Generic Security Service Application	gssapi(5)
programming language macro processor	m4(1)
programs, a compiler/interpreter for modest-sized	bs(1)
programs, HALGOL, execute	opx25(1M)
programs; input editor and command history for interactive	ied(1)
programs; maintain, update, and regenerate groups of	make(1)

Description	Entry Name(Section)
prompt	glossary(9)
propagation of Network Information Service database; force	yppush(1M)
protect terminal from use by others	lock(1)
protected password authentication database for trusted systems	prpwd(4)
protected password database entries (for trusted systems only); manipulate	getprpwd(3)
protected password database; display	getprpw(1M)
protected password database; modify	modprpw(1M)
proto - prototype job file for at	proto(4)
protocol daemon; PPP point to point	pppd(1)
protocol address (X/OPEN XTI)	t_getprotaddr(3)
protocol attribute, get or set	pthread_mutexattr_getprotocol(3T)
Protocol Daemon; Service Location	slpd(1M)
protocol entry; get, set, or end	getprotoent(3N)
protocol family, Internet	inet(7F)
protocol module, remote magnetic tape dump and restore	rmt(1M)
protocol name database	protocols(4)
Protocol over Ethernet) client configuration file, PPPoE (Point to Point	pppoe.conf(4)
Protocol over Ethernet) client, PPPoE (Point to Point	pppoe(1)
Protocol over Ethernet) relay configuration file, PPPoE (Point to Point	pppoerd.conf(4)
Protocol over Ethernet) relay, PPPoE (Point to Point	pppoerd(1M)
Protocol over Ethernet) server configuration file, PPPoE (Point to Point	pppoerd.conf(4)
Protocol over Ethernet) server daemon, PPPoE (Point-to-Point	pppoesd(1M)
protocol server, for TCP/IP IDENT	identd(1M)
protocol server; file transfer	ftpd(1M)
Protocol server; Internet Boot	bootpd(1M)
protocol server; TELNET	telnetd(1M)
protocol server; trivial file transfer	tftpd(1M)
Protocol (SLP) error codes; Service Location	SLPError(3N)
Protocol Version 6; Internet	IPv6(7P)
protocol, address resolution	arp(7P)
protocol, create a security context using the RPCSEC_GSS	rpc_gss_seccreate(3N)
protocol, Internet user datagram	UDP(7P)
Protocol, IP Internet	IP(7P)
protocol, local communication domain	UNIX(7P)
Protocol, NDP; Neighbor Discovery	ndp(7P)
Protocol, TCP Internet Transmission Control	TCP(7P)
protocol, user interface to the TELNET	telnet(1)
protocol-specific service information (X/OPEN TLI-XTI)	t_getinfo(3)
protocols - protocol name database	protocols(4)
prototype job file for at	proto(4)
provide displaying options for HP-UX errors defined in the Common Error Repository (CER)	emtui(1)
provide information about EVM	evminfo(1)
provide semaphores and record locking on files	lockf(2)
provide sequential archive member access for ELF files	elf_next(3E)
provide truth value about processor type	machid(1)
provided in the C library, list of pthread calls for which the stubs are	pthread_stubs(5)
provider implementation for pam_acct_mgmt ; service	pam_sm_acct_mgmt(3)
Provider Interface; ACPS Service	acps_spi(3)
prpwd - protected password authentication database for trusted systems	prpwd(4)
prs - print and summarize an SCCS file	prs(1)
prtacct - print any total accounting (tacct) file	acctsh(1M)
ps - report process status	ps(1)
PS/2 keyboard and mouse device driver	ps2(7)
ps2 - PS/2 keyboard and mouse device driver	ps2(7)
ps2kbd - PS/2 keyboard device file	ps2(7)
ps2mouse - PS/2 mouse device file	ps2(7)
pselect() - synchronous I/O multiplexing	select(2)
pset_assign() - change processor set assignment	pset_assign(2)
pset_bind() - bind process or thread to a processor set	pset_bind(2)
pset_create() - create a processor set	pset_create(2)
pset_ctl() - processor set control	pset_ctl(2)

Index

All Volumes

Description	Entry Name(Section)
pset_destroy() - destroy processor set	pset_destroy(2)
pset_getattr() - get processor set attributes	pset_getattr(2)
pset_setattr() - set processor set attributes	pset_getattr(2)
pseudo terminals (pts), maximum number of STREAMS-based	nstrpty(5)
pseudo terminals (ptys), maximum number of BSD	npty(5)
pseudo-random numbers, generate uniformly distributed	drand48(3C)
pseudo-terminal driver	pty(7)
pseudo-terminal master device; open	posix_openpt(3C)
pseudo-terminal master driver, STREAMS	ptm(7)
pseudo-terminal slave driver, for STREAMS	pts(7)
pseudo-terminal slave driver, for STREAMS	tels(7)
pseudo-terminal, get name of user's terminal or	tty(1)
pseudo-terminal, Packet Mode module for STREAMS pty	pkct(7)
pseudo-terminal, STREAMS Emulation module	ptem(7)
pseudorandom number generation functions	random(3M)
PSF format; product specification file	swpackage(4)
psfontpf - internationalized PostScript print filter	psfontpf(1M)
psfontpf; model script configuration utility for	psmsgen(1M)
psmsgen - model script configuration utility for psfontpf	psmsgen(1M)
psrset - create and manage processor sets	psrset(1M)
pstack - print a stack trace for each LWP in each process and core file	pstack(1)
pstat - an infrastructure for obtaining information from the kernel	pstat(2)
pstat_getcommandline() - get command line of a process	pstat(2)
pstat_getcrashdev() - get information for a crash dump device	pstat(2)
pstat_getcrashinfo() - get information for a system's crash dump configuration	pstat(2)
pstat_getdisk() - get information for a disk	pstat(2)
pstat_getdynamic() - get dynamic information about the system	pstat(2)
pstat_getfile2() - get information for an open file of a process	pstat(2)
pstat_getfiledetails() - get detailed information for an open file	pstat(2)
pstat_getio() - get information about an I/O object	pstat(2)
pstat_getipc() - get information about the System V IPC subsystem	pstat(2)
pstat_getlocality() - returns system-wide information of a ccNUMA system	pstat_getlocality(2)
pstat_getlvp() - get information for a logical volume	pstat(2)
pstat_getlwp() - get information for a thread or LWP in a process	pstat(2)
pstat_getmpathname() - get entries from system cache of recently looked-up names	pstat(2)
pstat_getnode() - get information about an SCA system node	pstat(2)
pstat_getpathname() - get the full path name of an open file	pstat(2)
pstat_getpmq() - get information for a POSIX message queue	pstat(2)
pstat_getproc() - get information for a process	pstat(2)
pstat_getprocessor() - get information for a processor	pstat(2)
pstat_getproclocality() - returns per-process information of a ccNUMA system	pstat_getlocality(2)
pstat_getprocvm() - get information for a process's address space	pstat(2)
pstat_getpsem() - get information for a POSIX named semaphore	pstat(2)
pstat_getpset() - get information for a processor set	pstat(2)
pstat_getsem() - get information for a System V semaphore set	pstat(2)
pstat_getshm() - get information for a System V shared memory segment	pstat(2)
pstat_getsocket() - get detailed information for a socket	pstat(2)
pstat_getstable() - get information from the system's stable storage area	pstat(2)
pstat_getstatic() - get information about the system	pstat(2)
pstat_getstream() - get detailed information for a stream	pstat(2)
pstat_getswap() - get information for a swap area	pstat(2)
pstat_getvminfo() - get information about the virtual memory subsystem	pstat(2)
ptem - STREAMS pty (pseudo-terminal) Emulation module	ptem(7)
pthread calls for which the stubs are provided in the C library, list of	pthread_stubs(5)
pthread() - introduction to POSIX.1c threads	pthread(3T)
pthread register fork handler	pthread_atfork(3T)
pthread_atfork() - register fork handler	pthread_atfork(3T)
pthread_attr_destroy() - destroy a thread attribute object	pthread_attr_destroy(3T)
pthread_attr_getdetachstate() - get the detachstate attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getguardsize() - get the guardsize attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getinheritsched() - get the inheritsched attribute	pthread_attr_getdetachstate(3T)

Description	Entry Name(Section)
pthread_attr_getprocessor_np() - get the processor and binding_type attributes	pthread_attr_getdetachstate(3T)
pthread_attr_getrsestacksize_np() - get the rse stack size attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getschedparam() - get the schedparam attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getschedpolicy() - get the schedpolicy attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getscope() - get the contention scope attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getstack() - get stacksize and stackaddr attributes	pthread_attr_getdetachstate(3T)
pthread_attr_getstackaddr() - get the stackaddr attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getstacksize() - get the stacksize attribute	pthread_attr_getdetachstate(3T)
pthread_attr_init() - initialize a thread attribute object	pthread_attr_init(3T)
pthread_attr_setdetachstate() - set the detachstate attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setguardsize() - set the guardsize attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setinheritsched() - set the inheritsched attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setprocessor_np() - set the processor and binding_type attributes	pthread_attr_getdetachstate(3T)
pthread_attr_setrsestacksize_np() - set the rse stack size attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setschedparam() - set the schedparam attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setschedpolicy() - set the schedpolicy attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setscope() - set the contention scope attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setstack() - set stacksize and stackaddr attributes	pthread_attr_getdetachstate(3T)
pthread_attr_setstackaddr() - set the stackaddr attribute	pthread_attr_getdetachstate(3T)
pthread_attr_setstacksize() - set the stacksize attribute	pthread_attr_getdetachstate(3T)
pthread_cancel() - cancel execution of a thread	pthread_cancel(3T)
pthread_cleanup_pop() - remove a thread cancellation cleanup handler	pthread_cleanup_pop(3T)
pthread_cleanup_push() - register a thread cancellation cleanup handler	pthread_cleanup_pop(3T)
pthread_cond_broadcast() - unblock all threads waiting on a condition variable	pthread_cond_signal(3T)
pthread_cond_destroy() - destroy a condition variable	pthread_cond_init(3T)
pthread_cond_init() - initialize a condition variable	pthread_cond_init(3T)
pthread_cond_signal() - unblock one thread waiting on a condition variable	pthread_cond_signal(3T)
pthread_cond_timedwait() - wait or timed wait on a condition variable	pthread_cond_wait(3T)
pthread_cond_wait() - wait or timed wait on a condition variable	pthread_cond_wait(3T)
pthread_condattr_destroy() - destroy a thread condition variable attributes object	pthread_condattr_init(3T)
pthread_condattr_getpshared() - get the thread process-shared attribute	pthread_condattr_getpshared(3T)
pthread_condattr_init() - initialize a thread condition variable attributes object	pthread_condattr_init(3T)
pthread_condattr_setpshared() - set the thread process-shared attribute	pthread_condattr_getpshared(3T)
pthread_continue() - continue execution of a thread	pthread_resume_np(3T)
pthread_create() - create a new thread of execution	pthread_create(3T)
pthread_default_rsestacksize_np() - change the default stacksize.	pthread_default_rsestacksize_np(3T)
pthread_default_stacksize_np() - change the default stacksize	pthread_default_stacksize_np(3T)
pthread_detach() - mark a thread as detached to reclaim its resources when terminate	pthread_detach(3T)
pthread_equal() - compare two thread identifiers	pthread_equal(3T)
pthread_exit() - cause the calling thread to terminate	pthread_exit(3T)
pthread_get_nice_np() - get the nice value of a thread	pthread_get_nice_np(3T)
pthread_getconcurrency() - get concurrency level of unbound threads	pthread_getconcurrency(3T)
pthread_getschedparam() - get the scheduling policy and associated parameters	pthread_getschedparam(3T)
pthread_getspecific() - get the thread-specific data associated with a key	pthread_getspecific(3T)
pthread_gettimeslice_np() - set or get the scheduling timeslice value for threads with SCHED_TIMESHARE scheduling policy	pthread_gettimeslice_np(3T)
pthread_join() - wait for the termination of a specified thread	pthread_join(3T)
pthread_key_create() - create a thread-specific data key	pthread_key_create(3T)
pthread_key_delete() - destroy a thread-specific data key	pthread_key_create(3T)
pthread_kill() - send a signal to a thread	pthread_kill(3T)
pthread_launch_policy() - setting thread launch policy	pthread_launch_policy(3T)
pthread_ldom_bind_np() - bind a thread to a locality domain	pthread_processor_bind_np(3T)
pthread_ldom_id_np() - obtain locality domain IDs	pthread_processor_bind_np(3T)

Index

All Volumes

Description	Entry Name(Section)
<code>pthread_mutex_destroy()</code> - destroy a mutex	<code>pthread_mutex_init(3T)</code>
<code>pthread_mutex_disable_handoff_np()</code> - disable process-wide mutex handoff mode	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutex_getprioceiling()</code> - get the prioceiling of a mutex	<code>pthread_mutex_getprioceiling(3T)</code>
<code>pthread_mutex_getyieldfreq_np()</code> - get mutex yield frequency attribute	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutex_init()</code> - initialize a mutex	<code>pthread_mutex_init(3T)</code>
<code>pthread_mutex_lock()</code> - lock a mutex	<code>pthread_mutex_lock(3T)</code>
<code>pthread_mutex_setprioceiling()</code> - set the prioceiling of a mutex	<code>pthread_mutex_getprioceiling(3T)</code>
<code>pthread_mutex_setyieldfreq_np()</code> - set mutex yield frequency attributes	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutex_trylock()</code> - try to lock a mutex	<code>pthread_mutex_lock(3T)</code>
<code>pthread_mutex_unlock()</code> - unlock a mutex	<code>pthread_mutex_unlock(3T)</code>
<code>pthread_mutexattr_destroy()</code> - destroy a mutex attribute object	<code>pthread_mutexattr_init(3T)</code>
<code>pthread_mutexattr_disable_handoff_np()</code> - disable mutex-specific handoff mode	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutexattr_getprioceiling()</code> - get the prioceiling attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_getprotocol()</code> - get the protocol attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_getpshared()</code> - get the process-shared attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutexattr_getspin_np()</code> - get mutex spin attribute	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutexattr_gettype()</code> - get the type attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutexattr_init()</code> - initialize a mutex attribute object	<code>pthread_mutexattr_init(3T)</code>
<code>pthread_mutexattr_setprioceiling()</code> - set the prioceiling attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_setprotocol()</code> - set the protocol attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_setpshared()</code> - set the process-shared attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutexattr_setspin_np()</code> - set mutex spin attribute	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutexattr_settype()</code> - set the type attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_num_ldomprocs_np()</code> - determine processors on locality domain	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_num_ldoms_np()</code> - determine number of locality domain	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_num_processors_np()</code> - return how many processors are available	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_once()</code> - call an initialization routine only once	<code>pthread_once(3T)</code>
<code>pthread_processor_bind_np()</code> - bind threads to processors	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_processor_id_np()</code> - determine processor IDs	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_pset_bind_np()</code> - bind a thread to processor set	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_resume_np()</code> - resume execution of a thread	<code>pthread_resume_np(3T)</code>
<code>pthread_rwlock_destroy()</code> - destroy a read-write lock	<code>pthread_rwlock_init(3T)</code>
<code>pthread_rwlock_init()</code> - initialize a read-write lock	<code>pthread_rwlock_init(3T)</code>
<code>pthread_rwlock_rdlock()</code> - lock a read-write lock for reading	<code>pthread_rwlock_rdlock(3T)</code>
<code>pthread_rwlock_tryrdlock()</code> - attempt to lock a read-write lock for reading	<code>pthread_rwlock_rdlock(3T)</code>
<code>pthread_rwlock_trywrlock()</code> - attempt to lock a read-write lock for writing	<code>pthread_rwlock_wrlock(3T)</code>
<code>pthread_rwlock_unlock()</code> - unlock a read-write lock	<code>pthread_rwlock_unlock(3T)</code>
<code>pthread_rwlock_wrlock()</code> - lock a read-write lock for writing	<code>pthread_rwlock_wrlock(3T)</code>
<code>pthread_rwlockattr_destroy()</code> - destroy a read-write lock attribute object	<code>pthread_rwlockattr_init(3T)</code>
<code>pthread_rwlockattr_getpshared()</code> - get the process-shared attribute	<code>pthread_rwlockattr_getpshared(3T)</code>
<code>pthread_rwlockattr_init()</code> - initialize a read-write lock attribute object	<code>pthread_rwlockattr_init(3T)</code>
<code>pthread_rwlockattr_setpshared()</code> - set the process-shared attribute	<code>pthread_rwlockattr_getpshared(3T)</code>
<code>pthread_scope_options</code> - list of external options to specify the scheduling contention scope of threads	<code>pthread_scope_options(5)</code>
<code>PTHREAD_SCOPE_PROCESS</code> threads with <code>SCHED_TIMESHARE</code> scheduling policy, set or get the scheduling timeslice value for	<code>pthread_gettimeslice_np(3T)</code>
<code>pthread_self()</code> - obtain the thread ID for the calling thread	<code>pthread_self(3T)</code>
<code>pthread_set_nice_np()</code> - set the nice value of a thread	<code>pthread_get_nice_np(3T)</code>
<code>pthread_setcancelstate()</code> - set and retrieve the current thread's cancelability state	<code>pthread_setcancelstate(3T)</code>
<code>pthread_setcanceltype()</code> - set and retrieve the current thread's cancelability type	<code>pthread_setcancelstate(3T)</code>

Description	Entry Name(Section)
pthread_setconcurrency() - set concurrency level of unbound threads	pthread_getconcurrency(3T)
pthread_setschedparam() - set the scheduling policy and associated parameters	pthread_getschedparam(3T)
pthread_setschedprio() - sets scheduling priority of a thread	pthread_setschedprio(3T)
pthread_setspecific() - set the thread-specific data associated with a key	pthread_getspecific(3T)
pthread_settimeslice_np() - set or get the scheduling timeslice value for threads with SCHED_TIMESHARE scheduling policy	pthread_gettimeslice_np(3T)
pthread_sigmask() - examine and change the signal mask of the calling thread	pthread_sigmask(3T)
pthread_spu_to_ldom_np() - determine ID of locality domain specified by spu	pthread_processor_bind_np(3T)
pthread_stubs - list of pthread calls for which the stubs are provided in the C library	pthread_stubs(5)
pthread_suspend() - suspend execution of a thread	pthread_resume_np(3T)
pthread_testcancel() - process any pending cancellation requests	pthread_testcancel(3T)
ptm - STREAMS master pty (pseudo-terminal) driver	ptm(7)
ptree - prints the process tree hierarchy	ptree(1)
pts - STREAMS slave pty (pseudo-terminal) driver	pts(7)
ptsname() - get the name of a slave pty	ptsname(3C)
ptx - generate permuted index	ptx(1)
pty and tty data transfers; number of cblocks for	nclist(5)
pty - get the name of the user's pseudo-terminal	tty(1)
pty master driver, STREAMS	ptm(7)
pty - pseudo-terminal driver	pty(7)
pty, get the name of a slave	ptsname(3C)
ptys, maximum number of pseudo terminals	npty(5)
ptys, maximum number of STREAMS-based pseudo terminals	nstrpty(5)
public key, database	publickey(4)
public or secret key, retrieve	getpublickey(3M)
public UNIX system to UNIX system file copy	uuto(1)
publickey database file, creating new key in	newkey(1M)
publickey database file, updates to	udpublickey(1M)
publickey - database for public keys	publickey(4)
publickey() - retrieve public or secret key	getpublickey(3M)
push a character onto the input queue	ungetch(3X)
push character back into input stream	ungetc(3S)
push wide character back into input stream	ungetwc(3C)
pushd - push directory stack	csh(1)
pushed STREAMS modules; manage system database	autopush(1M)
put a string on a stream	puts(3S)
put character or word on a stream	putc(3S)
put wide character on a stream	putwc(3C)
put word or character on a stream	putc(3S)
putc() , fputc() - put character on a stream	putc(3S)
putchar() - put character on stream <i>standard output</i>	putc(3S)
putdvagname() - add or rewrite device assignment database entry for trusted system	getdvagent(3)
putenv() - change or add value to environment	putenv(3C)
putp() - output commands to the terminal	putp(3X)
putpmsg() - send a message on a stream	putmsg(2)
putpmsg() - send a message on a stream in different priority bands	putmsg(2)
putprdfname() - lock system default database entry for trusted system	getprdfent(3)
putprdfname() - manipulate system default database entry for a trusted system	getprdfent(3)
putprdfname() - put default control entry for system default database entry for trusted system ..	getprdfent(3)
putprpwnam() - manipulate protected password database entries (for trusted systems only)	getprpwent(3)
putprpwnam() - manipulate protected password database entry (trusted systems)	getprpwent(3)
putprtcnam() - manipulate terminal control database entry	getprtcent(3)
putpwent() - write password file entry	putpwent(3C)
puts() - put a string on a stream	puts(3S)
putspent - shadow password file entry, write	putspent(3C)
pututline() - update or create entry in utmp file	getut(3C)
pututline_r() - update or create entry in utmp file	getut(3C)
PUTUTSLINE() - access and update routines for user-accounting database maintained by utmpd	getuts(3C)

Index

All Volumes

Description	Entry Name(Section)
pututslne() - access and update routines for user-accounting database maintained by utmpd	getuts(3C)
pututxline() - update or create entry in a utmpx file	getutx(3C)
putw() - put word (integer) on a stream	putc(3S)
putwc() , fputwc() - put wide character on a stream	putwc(3C)
putwchar() - put wide character on stream <i>standard output</i>	putwc(3C)
putwin() - dump window to and reload window from a file	getwin(3X)
putws() - put a wide character string on a stream file	putws(3C)
pvchange - change characteristics and access path of physical volume in LVM volume group	pvchange(1M)
pvck - check or repair a physical volume in LVM volume group	pvck(1M)
pvcreate - create physical volume for use in LVM volume group	pvcreate(1M)
pvdisplay - display information about physical volumes in LVM volume group	pvdisplay(1M)
pvmove - move allocated physical extents from one LVM physical volume to other physical volumes	pvmove(1M)
pvremove - remove an LVM physical volume	pvremove(1M)
pw_getukid() - lightweight synchronization mechanism	postwait(2)
pw_getvmax() - lightweight synchronization mechanism	postwait(2)
pw_post() - lightweight synchronization mechanism	postwait(2)
pw_postv() - lightweight synchronization mechanism	postwait(2)
pw_wait() - lightweight synchronization mechanism	postwait(2)
pwck - password/group file checkers	pwck(1M)
pwconv - install, update or check the /etc/shadow file	pwconv(1M)
pwd - print current working directory	ksh(1)
pwd - print current working directory	sh-posix(1)
pwd - working directory name	pwd(1)
pwget - get password information	pwget(1)
pwgr_stat - password and group hashing and caching statistics	pwgr_stat(1M)
pwgrd - password and group hashing and caching daemon	pwgrd(1M)
pwrite() - write contiguous data to a position in a file	pwrite(2)
pwrite64() - non-POSIX standard API interfaces to support large files	creat64(2)
pwunconv - convert passwords from shadow to nonshadow	pwunconv(1M)
qiflush() - enable/disable queue flushing	noqiflush(3X)
QOP files; security mechanism and quality of protection	mech(4)
QOP for a session, change service,	rpc_gss_set_defaults(3N)
qop - security mechanism and quality of protection (QOP) files	mech(4)
QOP strings to non-string values, map mechanism,	rpc_gss_mech_to_oid(3N)
qsort() - quicker sort	qsort(3C)
quality of protection (QOP) files; security mechanism and	mech(4)
quarantines the sendmail message using the given reason	smfi_quarantine(3N)
query functions for terminal insert and delay capability	has_ic(3X)
query log file of uucp transactions	uucp(1)
query name servers interactively	nslookup(1)
query NIS server for information about NIS map	yppoll(1M)
query numeric formatting conventions of current locale	localeconv(3C)
query program, Network Time Protocol	ntp(1M)
query program; special NTP	xntpd(1M)
query RIP gateways	ripquery(1M)
query stream configuration	strchg(1M)
query the Name Service Switch backend libraries	nsquery(1)
query the terminfo database	tput(1)
query values in unwind library data structure	_UNW_getGR(3X)
query, enable, or disable compartmentalization feature	cmpt_tune(1M)
queue a signal to a process	sigqueue(2)
queue description file for at , batch , and crontab	queuedefs(4)
queue flushing, enable/disable	noqiflush(3X)
queue, get information for a POSIX message	pstat(2)
queue, input, push a character onto	ungetch(3X)
queue, insert or remove an element	insque(3C)
queue, maximum number of bytes on a single System V IPC message	msgmb(5)
queue, prints the mail queue	mailq(1)
queue; create or open a message	mq_open(2)
queue; register or cancel a notification request with a message	mq_notify(2)
queued signals per process, limit on number of	ksi_send_max(5)

Description	Entry Name(Section)
queued signals that can be allocated, system-wide limit of	ksi_alloc_max(5)
queuedefs - queue description file for at , batch , and crontab	queuedefs(4)
queues (IDs) allowed, maximum number of system-wide System V IPC message	msgmni(5)
quick batch mail interface	fastmail(1)
quicker sort	qsort(3C)
quiet comparison macro, floating-point (<)	isless(3M)
quiet comparison macro, floating-point (<=)	islessequal(3M)
quiet comparison macro, floating-point (< >)	islessgreater(3M)
quiet comparison macro, floating-point (>)	isgreater(3M)
quiet comparison macro, floating-point (>=)	isgreaterequal(3M)
quit signal	glossary(9)
quot - summarize file system ownership	quot(1M)
quot - summarize ownership on an HFS file system	quot_hfs(1M)
quota consistency checker for HFS file systems	quotacheck_hfs(1M)
quota consistency checker, file system	quotacheck(1M)
quota - disk quotas	quota(5)
quota - display disk usage and limits	quota(1)
quota server, remote	rquotad(1M)
quota status of specified file system, determine disk	fsclean(1M)
quotacheck - file system quota consistency checker	quotacheck(1M)
quotacheck - quota consistency checker for HFS file systems	quotacheck_hfs(1M)
quotactl () - manipulate disk quotas	quotactl(2)
quotaoff - turn HFS file system quotas off	quotaon(1M)
quotaon - turn HFS file system quotas on	quotaon(1M)
quotas, file system; turn on and off	quotaon(1M)
quotas; disk	quota(5)
quotas; edit disk	edquota(1M)
quotas; manipulate disk	quotactl(2)
quotas; summarize file system	repquota(1M)
rad - rad features have been moved to olrad	rad(1M)
radix character	glossary(9)
radix-independent exponent functions	ilogb(3M)
radix-independent exponent functions	logb(3M)
radix-independent floating-point number, scale exponent of	scalb(3M)
radix-independent floating-point number; scale exponent of a	scalbln(3M)
raise a software signal	ssignal(3C)
raise () - send signal to executing program	kill(2)
RAM monitor; the Route Administration Manager	ram_monitor(1M)
ram_monitor - the Route Administration Manager (RAM) monitor	ram_monitor(1M)
RAMD configuration file; Route Administration Manager Daemon	ramd.conf(4)
ramd - Route Administration Manager Daemon for IPv6	ramd(1M)
ramd.conf - Route Administration Manager Daemon (RAMD) configuration file	ramd.conf(4)
rand () - generate successive random numbers	rand(3C)
rand (), srand (), initstate (), setstate () - generate a pseudorandom number	random(3M)
random archive member access for ELF files	elf_rand(3E)
random inode generation numbers, install	fsirand(1M)
random number generation functions	random(3M)
random number generator, strong	random(7)
random - strong random number generator	random(7)
random-number generator, simple	rand(3C)
ranlib (1) - regenerate archive symbol table	ranlib(1)
rarpd - Reverse Address Resolution Protocol client	rarpd(1M)
rarpd - Reverse Address Resolution Protocol daemon	rarpd(1M)
raster frame-buffer devices; information for	framebuf(7)
Ratfor macro processor	m4(1)
ratio between number of pending AIO requests and servicing threads, desirable	aio_req_per_thread(5)
raw disk	glossary(9)
raw () - input mode control functions	cbreak(3X)
RBAC database files, verify the syntax of the Role-Based Access Control	rbacdbchk(1M)
RBAC databases, non-interactive editing of the authorization information in the	authadm(1M)

Index

All Volumes

Description	Entry Name(Section)
RBAC databases; noninteractive editing of role-related information in	roleadm(1M)
RBAC - role-based access control	rbac(5)
rbacdbchk - verify the syntax of the Role-Based Access Control (RBAC) database files	rbacdbchk(1M)
rc - general purpose sequencer invoked upon entering new run level	rc(1M)
rcsequencer , invoked upon entering new run level	rc(1M)
rc.config - file containing system configuration information	rc.config(4)
rc.config.d - the location of files containing system configuration variable assignments	rc.config(4)
rcancel - remove requests from line printer spooling queue on remote system	rcancel(1M)
rcmd() - execute a command on a remote host	rcmd(3N)
rcmd_af() - execute a command on a remote host	rcmd(3N)
rcp - remote file copy	rcp(1)
rccs - change RCS file attributes	rccs(1)
RCS commands; description of	rcsintro(5)
RCS file attributes, change	rccs(1)
RCS files; format of	rcsfile(4)
RCS files; print log messages and other information on	rlog(1)
RCS revisions; check in	ci(1)
RCS revisions; check out	co(1)
RCS revisions; compare	rcsdiff(1)
RCS revisions; merge	rcsmerge(1)
RCS: change RCS file attributes	rccs(1)
RCS; identify files in	ident(1)
rcsdiff - compare RCS revisions	rcsdiff(1)
rcsfile - format of RCS files	rcsfile(4)
rcsintro - description of RCS commands	rcsintro(5)
rcsmerge - merge RCS revisions	rcsmerge(1)
rdc - user interface for Routing Administration Manager (RAMD)	rdc(1M)
RDMA interfaces; network interface management command for LAN and	nwmgr(1M)
rdpd - router discovery protocol daemon	rdpd(1M)
rdump - incremental file system dump (for backups)	dump(1M)
read a line from standard input	read(1)
read a NaT bit from current frame's context	uwx_get_nat(3X)
read a POSIX semaphore	sem_getvalue(2)
read a register from current frame's context	uwx_get_reg(3X)
read from crash dump	cr_read(3)
read from stream file or character string with formatted input conversion,	scanf(3S)
read - input and parse a line	ksh(1)
read - input and parse a line	sh-posix(1)
read mail; send mail to users or	mail(1)
read one line from user input	line(1)
read password from terminal while suppressing echo	getpass(3C)
read - read a line from standard input	read(1)
read() - read contiguous data from a file	read(2)
read() - read from file	read(2)
read real-time priority	rtprio(2)
read stream up to next delimiter	bgets(3G)
read() - STREAMS enhancements to standard system calls	stream(2)
read text string from message file	getttx(3C)
read value of a symbolic link	readlink(2)
read, enable or disable block during	nodelay(3X)
read, start asynchronous	aio_read(2)
read, write or delete information in the user database, /var/adm/userdb	userdb_read(3)
read-only file system	glossary(9)
read-write lock attribute object, initialize or destroy	pthread_rwlockattr_init(3T)
read-write lock for reading, lock or attempt to lock	pthread_rwlock_rdlock(3T)
read-write lock for writing, lock or attempt to lock	pthread_rwlock_wrlock(3T)
read-write lock, unlock	pthread_rwlock_unlock(3T)
read-write lock; initialize or destroy	pthread_rwlock_init(3T)
read/write file pointer, move	lseek(2)
read/write shared memory in process core dump, determines the inclusion of	core_addshmem_write(5)
readable format, dump iconv translation tables to	dmpxlt(1)

Description	Entry Name(Section)
readable shared memory in a process core dump, determines the inclusion of	core_addshmem_read(5)
readdir() - get pointer to current entry in open directory	directory(3C)
readdir_r() - initialize a dirent structure	directory(3C)
reading, open file for	open(2)
readlink() - read value of a symbolic link	readlink(2)
readmail - read mail from specified mail folder	readmail(1)
readonly - mark names as unmodifiable	ksh(1)
readonly - mark names as unmodifiable	sh-posix(1)
reads and writes: STREAMS module	tirdwr(7)
readv() - read from file	read(2)
readv() - read noncontiguous data from a file	read(2)
readv() - STREAMS enhancements to standard system calls	stream(2)
real and effective group IDs; sets the	setregid(2)
real and effective user IDs, set	setreuid(2)
real group ID	glossary(9)
real group, and effective group IDs; get real user, effective user,	getuid(2)
real user ID	glossary(9)
real user ID, get	getresuid(3)
real user, effective user, real group, and effective group IDs; get	getuid(2)
real, effective, and/or saved user or group IDs, set	setresuid(2)
real, group ID, get	getresuid(3)
real-time priority, execute process with	rtprio(1)
real-time processor set control	_pset_rctl(2)
real-time scheduling operations	rtsched(2)
realloc() - change size of allocated memory block	malloc(3C)
realpath() - resolve pathname	realpath(3X)
realtime applications, number of priority values to support for POSIX.1b	rtsched_numpri(5)
realtime extensions	aio_reap(2)
realtime, POSIX extensions	aio(5)
realtime, POSIX extensions	aio_cancel(2)
realtime, POSIX extensions	aio_error(2)
realtime, POSIX extensions	aio_fsync(2)
realtime, POSIX extensions	aio_read(2)
realtime, POSIX extensions	aio_return(2)
realtime, POSIX extensions	aio_suspend(2)
realtime, POSIX extensions	aio_write(2)
realtime, POSIX extensions	lio_listio(2)
realtime, POSIX extensions	mlock(2)
realtime, POSIX extensions	mlockall(2)
realtime, POSIX extensions	munlock(2)
realtime, POSIX extensions	munlockall(2)
reauthenticating the user; invoke another application with privileges after performing appropriate authorization checks and optionally	privrun(1M)
reblock, convert, translate and copy a (tape) file	dd(1)
reboot() - boot the system	reboot(2)
reboot - reboot the system	reboot(1M)
reboot system	boot(1M)
reboot system automatically after shutting system down	shutdown(1M)
reboots, evaluate time between	last(1)
rebuild Network Information Service databases; create or	ypmake(1M)
rebuilds the database for the mail aliases file	newaliases(1M)
receipt of a signal, define what to do upon	signal(2)
receive a message from a message queue	mq_receive(2)
receive a message from a socket	recv(2)
receive error messages from the STREAMS log driver	strerr(1M)
receive message from message queue	msgop(2)
recipient for the current message; adds a	smfi_addrcpt(3N)
recipient from envelope of current sendmail message; removes a	smfi_delrcpt(3N)
reciprocal square root functions	rsqrt(3M)
recognized login shells, list of	shells(4)
reconfigure, unconfigure, configure installed software	swconfig(1M)

Index

All Volumes

Description	Entry Name(Section)
reconfiguring, EVM channel manager	evmreload(1M)
reconfiguring, EVM daemon	evmreload(1M)
reconfiguring, EVM logger	evmreload(1M)
record for self-auditing process; write audit	auditwrite(2)
record format; user login	utmp(4)
record locking and semaphores on files, provide	lockf(2)
recover files selectively from backup media	frecover(1M)
Recovery; enables/disables PCI Error	pci_ah_enable(5)
recursively descend a directory hierarchy, executing a function	ftw(3C)
recursively expands the sendmail aliases	expand_alias(1)
recv() - receive a message from a socket	recv(2)
recvfrom() - receive a message from a socket	recv(2)
recvmsg() - receive a message from a socket	recv(2)
red - line-oriented text editor	ed(1)
red - restricted line-oriented text editor	ed(1)
redefine default login shell	chsh(1)
redefine environment for command execution	env(1)
redirect the persistent device special file from one device to a different device	io_redirect_dsf(1M)
redrawwin() - line update status functions	redrawwin(3X)
reduce multiple adjacent blank lines to single blank line	ssp(1)
reentrant log gamma functions	lgamma(3M)
reference pages, macro package for formatting	man(5)
reformat a text file; change or	newform(1)
reformat and copy a (tape) file	dd(1)
refresh control function, for window	touchwin(3X)
refresh control functions for window	is_linetouched(3X)
refresh() - refresh windows and lines	doupdate(3X)
refresh the pad immediately after writing a character rendition	pechochar(3X)
refresh the window immediately after writing a complex character	echo_wchar(3X)
refresh, determine whether a screen has been refreshed	isendwin(3X)
regcomp() - regular expression matching routines	regcomp(3C)
regenerate groups of programs; maintain, update, and	make(1)
regerror() - regular expression matching routines	regcomp(3C)
regexexec() - regular expression matching routines	regcomp(3C)
regfree() - regular expression matching routines	regcomp(3C)
region of window, copy a	copywin(3X)
region, initialize semaphore in mapped file or anonymous memory	msem_init(2)
region, remove semaphore in mapped file or anonymous	msem_remove(2)
region, unmap a mapped	munmap(2)
region_hash_locks - OBSOLETE kernel tunable parameter	region_hash_locks(5)
regions; first locations beyond allocated program	end(3C)
register a function to be called at program termination	atexit(3)
register callback routines for stack unwind	uwx_register_callbacks(3X)
register custom allocate and free callbacks	uwx_register_alloc_cb(3X)
register fork handler	pthread_atfork(3T)
register information about dynamically generated functions	dlmodadd(3C)
register or cancel a notification request with a message queue	mq_notify(2)
register or remove a thread cancellation cleanup handler	pthread_cleanup_pop(3T)
registerrpc() - obsolete library routines for RPC	rpc_soc(3N)
registers a set of filter callbacks for sendmail	smfi_register(3N)
registers distributed file system packages; file that	fstypes(4)
registry services data to disk; flush kernel	krs_flush(1M)
registry services, KRS; kernel	krs(5)
regular expression	glossary(9)
regular expression and pattern matching notation definitions	regexp(5)
regular expression compile and match routines	regexp(3X)
regular expression matching routines	regcomp(3C)
regular file	glossary(9)
rehash - recompute internal hash table	csh(1)
reinitialize the context at a signal frame	uwx_self_do_context_frame(3X)
reject - prevent LP printer queuing requests	accept(1M)

Description	Entry Name(Section)
reject/select lines common to two sorted files	comm(1)
relational database, join two relations in	join(1)
relative path name	glossary(9)
relative window creation function	derwin(3X)
relatively arm a per-process timer	reltimer(3C)
relay configuration file, PPPoE (Point to Point Protocol over Ethernet)	pppoerd.conf(4)
relay, PPPoE (Point to Point Protocol over Ethernet)	pppoerd(1M)
release blocked signals and atomically wait for interrupt	sigpause(3C)
release level of operating system, display	uname(1)
release memory used by an unwind environment	uwx_free(3X)
release of operating system, display	uname(1)
reload Event Manager configuration files	evmreload(1M)
reload window from a file	getwin(3X)
reltimer() - relatively arm a per-process timer	reltimer(3C)
remainder functions	fmod(3M)
remainder functions	remainder(3M)
remainder functions with quotient	remquo(3M)
remainder() - remainder function	remainder(3M)
remainder; integer division and	div(3C)
remainderf() - remainder function (float)	remainder(3M)
remainderl() - remainder function (long double)	remainder(3M)
remainderq() - remainder function (quad)	remainder(3M)
remainderw() - remainder function (extended)	remainder(3M)
remind you when you have to leave	leave(1)
reminder service	calendar(1)
remote backup over network	dump(1M)
remote commands; return streams to parallel	prcmd(3N)
remote execution server	rexecd(1M)
remote execution server, RPC-based	rexd(1M)
remote file copy	rcp(1)
remote file distribution program	rdist(1)
remote host, execute a command on a	rcmd(3N)
remote host, execute command on a	on(1)
remote hosts and users, authorizing access on local host	hosts.equiv(4)
remote incremental file system dump (for backups)	dump(1M)
remote incremental file system restore	restore(1M)
remote kernel; get performance data from	rstat(3N)
remote login	rlogin(1)
remote login server	rlogind(1M)
remote machines, return information about users on	rnusers(3N)
remote machines, write to specified	rwall(3N)
remote magnetic tape dump and restore protocol module	rmt(1M)
remote mounted filesystem table	rmtab(4)
remote mounts, show all	showmount(1M)
remote NFS resources; mount and unmount	mount_nfs(1M)
remote node, get file handle for file on	getfh(2)
remote procedure calls, library routines for secure	secure_rpc(3N)
remote quota server	rquotad(1M)
remote shell server	remshd(1M)
remote shell; execute from a	remsh(1)
remote shell; execute from a	rexec(1)
remote system over LAN, log in on a	vt(1)
remote system, send LP request to	rlp(1M)
remote system; print status of LP spooler requests on a	rlpstat(1M)
remote system; test for successful login to	uucp(1)
remote systems, cancel LP spooling requests sent to	rcancel(1M)
remote systems, daemon for LP requests from	rlpdaemon(1M)
remote systems; make local resource available for mounting by	share(1M)
remote systems; make local resource unavailable for mounting by	unshare(1M)
remote terminal, spawn getty to (call terminal)	ct(1)
remote transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)

Index

All Volumes

Description	Entry Name(Section)
remote user communication server	talkd(1M)
remote user information server	fingerd(1M)
remote user, verify as a local user	rcmd(3N)
remote uucp or uux command requests, execute on local system	uuxqt(1M)
remote_nfs_swap - enable swapping across NFS	remote_nfs_swap(5)
remotely mounted file systems, keep track of	mount(3N)
remove a delta from an SCCS file	rmdel(1)
remove a directory file	rmdir(2)
remove a file or directory	rm(1)
remove a LIF file	lifrm(1)
remove a message queue, semaphore set, or shared memory identifier	ipcrm(1)
remove a POSIX message queue, semaphore name	pipcrm(1)
remove a printer for use with tsm ; add or	tsm.lpadmin(1M)
remove a special (device) file	rmsf(1M)
remove all blank lines from file	rmnl(1)
remove an advisory or enforced lock on an open file; apply or	flock(2)
remove an EFI directory	efi_rmdir(1M)
remove an EFI file	efi_rm(1M)
remove an existing partition	parremove(1M)
remove and list gsscred table entries; add,	gsscred(1M)
remove and unconfigure software products	swremove(1M)
remove boot programs from disk	mkboot(1M)
remove directories	rmdir(1)
remove directory entry	unlink(2)
remove duplicate entries from gsscred mapping table	gsscred_clean(1M)
remove extra new-line characters from file	rmnl(1)
remove file that is not listed in any directory	clri(1M)
remove files or directories	rm(1)
remove information registered using dlmodadd	dlmodremove(3C)
remove logical volumes from LVM volume group	lvremove(1M)
remove LVM logical volume link to root, primary swap, or dump volume	lvrmboot(1M)
remove LVM volume group definition from the system	vgremove(1M)
remove multiple line-feeds from output	ssp(1)
remove nroff/troff , tbl , and neqn constructs	deroff(1)
remove or insert an element in a queue	insque(3C)
remove or register a thread cancellation cleanup handler	pthread_cleanup_pop(3T)
remove outdated STREAMS error log files	strclean(1M)
remove physical volume in LVM volume group	pvremove(1M)
remove physical volumes from an LVM volume group	vgreduce(1M)
remove preprocessor lines	unifdef(1)
remove () - remove a file	remove(3C)
remove reverse line-feeds and backspaces from text	col(1)
remove semaphore in mapped file or anonymous region	msem_remove(2)
remove shutdown message file created by ftpsht	ftprestart(1)
remove symbol and line number information from an object file	strip(1)
remove, and retrieve a process' privileges; add, set,	priv_add(3)
remove, create directories in a path	mkdirp(3G)
removes a recipient from envelope of current sendmail message	smfi_delrcpt(3N)
remque () - remove an element in a queue	insque(3C)
remquo () - remainder function with quotient	remquo(3M)
remquof () - remainder function with quotient (float)	remquo(3M)
remquol () - remainder function with quotient (long double)	remquo(3M)
remquoq () - remainder function with quotient (quad)	remquo(3M)
remquow () - remainder function with quotient (extended)	remquo(3M)
remsh - execute from a remote shell	remsh(1)
remshd - remote shell server	remshd(1M)
rename () - change the name of a file	rename(2)
rename directory	mv(1)
rename file	mv(1)
rename LIF files	lifrename(1)
renditions and characters, draw lines from single-byte	hline(3X)

Description	Entry Name(Section)
renditions of characters in a window, change	chgat(3X)
renice - alter priority of running processes	renice(1M)
reopen a stream file	fopen(3S)
repair damaged file system (generic)	fsdb(1M)
repair damaged HFS file system	fsdb_hfs(1M)
repair or check a physical volume in LVM volume group	pvck(1M)
repeat - execute command more than once	csch(1)
repeated (adjacent) lines in a file, count, extract, or eliminate	uniq(1)
repetitively affirmative responses	yes(1)
replace selected characters	tr(1)
replaces the data in the sendmail message body	smfi_replacebody(3N)
reply code to a multi-line response; sets default SMTP error	smfi_setmlreply(3N)
report adjacent repeated lines in a file	uniq(1)
report CPU time used	clock(3C)
report disk usage	du(1)
report I/O statistics	iostat(1)
report number of free CDFS, HFS, or NFS file system disk blocks	df_hfs(1M)
report number of free disk clusters	dosdf(1)
report number of free file system disk blocks	df(1M)
report process status	ps(1)
report RPC information	rpcinfo(1M)
report status information of the LP subsystem	lpstat(1)
report status of interprocess communication facilities	ipcs(1)
report status of POSIX interprocess communication facilities	pipcs(1)
report virtual memory statistics	vmstat(1)
reporter; system activity	sar(1M)
reposition or get pointer for I/O operations on a stream file	fseek(3S)
repquota - summarize file system quotas	repquota(1M)
representation of an opaque internal name, convert an internal name to an internal MN name	gss_canonicalize_name(3)
request call-back POSIX asynchronous I/O operations; percentage of physical memory lockable for	aio_physmem_pct(5)
request scheduler; start the LP	lpsched(1M)
request scheduler; stop the LP	lpsched(1M)
request server	automountd(1M)
request with a message queue; register or cancel a notification	mq_notify(2)
request; receive confirmation (X/OPEN TLI-XTI)	t_rcvconnect(3)
request_init() - access control library	hosts_access(3)
request_set() - access control library	hosts_access(3)
requests between LP destinations; move	lpsched(1M)
requests on a remote system; print status of LP spooler	rlpstat(1M)
requests on an LP printer	lp(1)
requests, daemon that responds to SNMP	snmpd(1M)
requests, LP	<i>see LP spooler requests</i>
res_init() - resolver routines	resolver(3N)
res_mkquery() - resolver routines	resolver(3N)
res_query() - resolver routines	resolver(3N)
res_search() - resolver routines	resolver(3N)
res_send() - resolver routines	resolver(3N)
reserve a line for a dedicated purpose	ripoffline(3X)
reserve a tape device on open; determines whether to	st_ats_enabled(5)
reserve disk space	prealloc(1)
reserve for the 32-bit DMA pool; the amount of memory to	dma32_pool_size(5)
reserved port socket, return a	rcmd(3N)
reset hung cell during cell activation; online activation of a cell from nPartition; cancel online cell operation; monitor online cell operation;	parolrad(1M)
reset - terminal-dependent initialization	tset(1)
reset_prog_mode() - restore shell terminal modes to "program" state	def_prog_mode(3X)
reset_shell_mode() - restore terminal modes to "shell" state	def_prog_mode(3X)
resetty() - save/restore terminal mode	resetty(3X)
residing in the user database, /var/adm/userdb; display information	userdbget(1M)

Index

All Volumes

Description	Entry Name(Section)
resolution protocol, address	arp(7P)
resolution time, get high	gethrtime(3C)
resolv.conf - resolver configuration file	resolver(4)
resolve pathname	realpath(3X)
resolver configuration file	resolver(4)
resolver daemon; lightweight	lwresd(1M)
resolver - resolver configuration file	resolver(4)
resolver - resolver routines	resolver(3N)
resolver routines	resolver(3N)
resource available for mounting by remote systems; make local	share(1M)
resource consumption; control maximum	getrlimit(2)
resource unavailable for mounting by remote systems; make local	unshare(1M)
resource utilization, get information	getusage(2)
resources across a network; file containing commands for sharing	dfstab(4)
resources; mount and unmount remote NFS	mount_nfs(1M)
resources; share, unshare multiple	shareall(1M)
respond to vt requests from other systems	vtdaemon(1M)
response, ask for user response for SD-UX	swask(1M)
responses, repetitively affirmative	yes(1)
restartable; convert a character string to a wide-character string	mbsrtowcs(3C)
restartterm() - interface to terminfo database	del_curterm(3X)
restore file position indicator for a stream; save or	fgetpos(3S)
restore file system incrementally	restore(1M)
restore program or shell terminal modes; save or	def_prog_mode(3X)
restore - restore file system incrementally, local or across network	restore(1M)
restore shell terminal modes to "program" state	def_prog_mode(3X)
restore signal action	sigset(3C)
restore terminal mode	resetty(3X)
restore terminal modes to "shell" state	def_prog_mode(3X)
restore volume group configuration	vgcfgrestore(1M)
restore/save stack environment for non-local goto	setjmp(3C)
restrict keyword	glossary(9)
restricted mailer (send only)	mail(1)
restricted POSIX.2-conformant command shells; standard and	sh-posix(1)
restricted shell for sendmail	smrsh(1M)
restricted window attribute control functions	atloff(3X)
resume accounting when available disk space reaches threshold, suspend and	acctsuspend(5)
resume auditing on the current process; suspend or	audswitch(2)
resume execution of a thread	pthread_resume_np(3T)
resvport() - return a reserved port socket	rcmd(3N)
resvport_af() - return a reserved port socket	rcmd(3N)
retrieve a process' privileges; add, set, remove, and	priv_add(3)
retrieve and set the current thread's cancelability state and type	pthread_setcancelstate(3T)
retrieve archive member header for ELF files	elf_getarhdr(3E)
retrieve archive symbol table	elf_getarsym(3E)
retrieve capabilities from the terminfo database	tigetflag(3X)
retrieve class-dependent object file header for elf32 or elf64 file	elf_getehdr(3E)
retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
retrieve class-dependent section header for ELF files	elf_getshdr(3E)
retrieve crash dump information	cr_info(3)
retrieve detailed information about kernel tunable parameters	tuneinfo2(2)
retrieve file identification data for ELF files	elf_getident(3E)
retrieve information on loaded module (program or shared library)	dlget(3C)
retrieve information on loaded module (program or shared library)	dlgetmodinfo(3C)
retrieve information on loaded module (program or shared library)	dlmodinfo(3C)
retrieve name of load module	dlgetname(3C)
retrieve SD product from new SD media	swgettools(1M)
retrieve stored events	evmget(1)
retrieve uninterpreted file contents for ELF files	elf_rawfile(3E)
return "I am not here" indication	vacation(1)
return [EOVERFLOW] if values do not fit in fields; causes uname() system function to	

Description	Entry Name(Section)
return ABI and context code from current context	uname_eoverflow(5)
return asynchronous I/O status	uwx_get_abi_context_code(3X) aio_return(2)
return character back into input stream	ungetc(3S)
return file information for a library prior to loading it	dlgetfileinfo(3C)
return how many processors are available	pthread_processor_bind_np(3T)
return HP-UX process priority	rtsched(2)
return information about users on remote machines	rnusers(3N)
return integer absolute value	abs(3C)
return load module information for current context	uwx_get_module_info(3X)
return maximum for scheduling policy	rtsched(2)
return minimum for scheduling policy	rtsched(2)
return POSIX process priority	rtsched(2)
return scheduling parameters	rtsched(2)
return scheduling policy	rtsched(2)
return - shell function return to invoking script	ksh(1)
return - shell function return to invoking script	sh-posix(1)
return source information for current frame	uwx_get_source_info(3X)
return start address of current function	uwx_get_funcstart(3X)
return stream to a remote command	rexec(3N)
return streams to parallel remote commands	prcmd(3N)
return symbolic information for current frame	uwx_get_sym_info(3X)
return the size of an object file type for elf32 or elf64 files	elf_fsize(3E)
return wide character back into input stream	ungetwc(3C)
returned by the stat() function; data	stat(5)
rev - reverse the text character sequence in each line of a file	rev(1)
Reverse Address Resolution Protocol client	rarp(1M)
Reverse Address Resolution Protocol daemon	rarpd(1M)
reverse line-feeds and backspaces, remove from text	col(1)
reverse order, show last commands executed in	lastcomm(1)
reverse - reverse printer pages for collating	lpfilter(1)
reverse the left-to-right text character sequence in each line of a file	rev(1)
Revision Control System	see RCS
Revision Control System	see RCS
revisions; check in RCS	ci(1)
revisions; check out RCS	co(1)
revisions; compare RCS	rcsdiff(1)
revisions; merge RCS	rcsmerge(1)
rewind legal user shells file	getusershell(3C)
rewind() - set position of next I/O operation on stream file	fseek(3S)
rewind_unlocked() - set position of next I/O operation on stream file, no locking of stream for multi-thread applications	fseek(3S)
rewinddir() - reset position of named directory stream to beginning of directory	directory(3C)
rewrite an existing file	creat(2)
rexcd - RPC-based remote execution server	rexcd(1M)
rexec , and rexec() , login information for ftp ,	netrc(4)
rexec() , login information for ftp , rexec , and	netrc(4)
rexec - execute from a remote shell	rexec(1)
rexec_af() - return stream to a remote command	rexec(3N)
rexecd - remote execution server	rexecd(1M)
right or left justify lines for NLS printing	nljust(1)
right triangle, hypotenuse of a	hypot(3M)
rights to a file, get a user's effective access	getaccess(2)
rights, access, to file(s), list	getaccess(1)
rintx() - BSD portability string routine	string(3C)
rint() - round to nearest integer function	rint(3M)
rintf() - round to nearest integer function (float)	rint(3M)
rintl() - round to nearest integer function (long double)	rint(3M)
rintq() - round to nearest integer function (quad)	rint(3M)
rintw() - round to nearest integer function (extended)	rint(3M)
RIP gateways, query	ripquery(1M)

Index

All Volumes

Description	Entry Name(Section)
RIPng routing daemon for IPv6	ripngd(1M)
ripngd - RIPng routing daemon for IPv6	ripngd (1M)
ripoffline () - reserve a line for a dedicated purpose	ripoffline (3X)
ripquery - query RIP gateways	ripquery (1M)
rksh - restricted Korn shell command programming language	rksh (1)
rlog - print log messages and other information on RCS files	rlog (1)
rlogin - remote login	rlogin (1)
rlogind - remote login server	rlogind (1M)
rlp - send LP line printer request to a remote system	rlp (1M)
rlpdaemon - line printer daemon for LP requests from remote systems	rlpdaemon (1M)
rlpstat - print status of LP spooler requests on a remote system	rlpstat (1M)
rm - remove files or directories	rm (1)
rmail - restricted mailer (send only)	rmail (1)
rmboot - remove boot programs from disk	mkboot (1M)
rmdel - remove a delta from an SCCS file	rmdel (1)
rmdir () - remove a directory file	rmdir (2)
rmdir - remove directories	rmdir (1)
rmdirp () - remove directories in a path	mkdirp (3G)
rmnl - remove extra new-line characters from file	rmnl (1)
rmsf - remove a special (device) file	rmsf (1M)
rmt - remote magnetic tape protocol module	rmt (1M)
rmtab - remote mounted filesystem table	rmtab (4)
rmtimer () - free a per-process timer	rmtimer (3C)
rndc configuration file	rndc.conf (4)
rndc key generation tool	rndc-confgen (1)
rndc - name server control utility	rndc (1)
rndc-confgen - rndc key generation tool	rndc-confgen (1)
rndc.conf - rndc configuration file	rndc.conf (4)
rng - strong random number generator	random (7)
rnusers () : return information about users on remote machines	rnusers (3N)
role-based access control	rbac (5)
Role-Based Access Control (RBAC) database files, verify the syntax of the	rbacdbchk (1M)
role-related information in RBAC databases; noninteractive editing of	roleadm (1M)
roleadm - noninteractive editing of role-related information in RBAC databases	roleadm (1M)
root directories for auditing subsystem; enable/disable tracking of current and	audit_track_paths (5)
root directory	glossary (9)
root directory, change	chroot (2)
root directory, change for a command	chroot (1M)
root password; set system initial identity parameter	set_parms (1M)
root volume	glossary (9)
root volume, remove LVM logical volume link	lvrmboot (1M)
root, boot, primary swap, or dump volume; prepare LVM logical volume to be	lvlnboot (1M)
root, target, modify software products in depot or	swmodify (1M)
roots and depots, register or unregister	swreg (1M)
round functions	round (3M)
round () - round function	round (3M)
round to long int functions	lround (3M)
round to long long functions	llround (3M)
round to nearest integer functions	rint (3M)
round to nearest long int functions	lrint (3M)
round to nearest long long functions	llrint (3M)
roundf () - round function (float)	round (3M)
rounding mode: getting floating-point	fegetround (3M)
rounding mode: setting floating-point	fesetround (3M)
roundl () - round function (long double)	round (3M)
roundq () - round function (quad)	round (3M)
roundw () - round function (extended)	round (3M)
Route Administration Manager Daemon (RAMD) configuration file	ramd.conf (4)
Route Administration Manager Daemon for IPv6	ramd (1M)
Route Administration Manager (RAM) monitor; the	ram_monitor (1M)
route and path between hosts, compute shortest	pathalias (1)

Description	Entry Name(Section)
route - kernel packet forwarding database	route(7P)
route - manipulate routing tables manually	route(1M)
router Advertisement daemon for IPv6	rtradvd(1M)
router advertisement daemon; configuration file for	rtradvd.conf(4)
router connection mapper, multicast	map-mbone(1M)
router, electronic address	pathalias(1)
routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
routine for sorted tables, binary search	bsearch(3C)
routine only once; call an initialization	pthread_once(3T)
routine to retrieve user name, PAM	pam_get_user(3)
routines for client side calls	rpc_clnt_calls(3N)
routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
routines for dealing with creation and manipulation of CLIENT handles	rpc_clnt_create(3N)
routines for external data representation	xdr(3N)
routines for external data representation	xdr_admin(3N)
routines for external data representation	xdr_complex(3N)
routines for external data representation	xdr_simple(3N)
routines for external data representation stream creation	xdr_create(3N)
routines for Integrity systems; execution startup	crto_ia(3)
routines for PA-RISC systems; execution startup	crto_pa(3)
routines for PAM, authentication information	pam_set_item(3)
routines for PAM; authentication transaction	pam_start(3)
routines for registering servers, rpc	rpc_svc_reg(3N)
routines for remote procedure calls, XDR	rpc_xdr(3N)
routines for remote procedure calls, rpc	rpc(3N)
routines for RPC servers	rpc_svc_calls(3N)
routines for secure remote procedure calls, library	secure_rpc(3N)
routines for server side remote procedure call errors	rpc_svc_err(3N)
routines for the creation of server handles, rpc	rpc_svc_create(3N)
routines for user-accounting database maintained by utmpd ; access and update	getuts(3C)
routines to maintain module specific state, PAM	pam_set_data(3)
routines, define label for formatting	setlabel(3)
routines, emulate /usr/share/lib/termcap access	termcap(3X)
routines, Internet address manipulation	inet6(3N)
routines, library routines for RPC bind service	rpcbind(3N)
routines, network station address string conversion	net_aton(3C)
routines, obsolete library routines for RPC	rpc_soc(3N)
routines, resolver	resolver(3N)
routines, RPCSEC_GSS security flavor library	rpcsec_gss(3N)
routines, security defaults configuration file	secdef(3)
routines, SLP (Service Location Protocol) library	libslp(3N)
routines; codeset conversion	iconv(3C)
routines; event filter evaluator	EvmFilterCreate(3)
routines; execution startup	crto(3)
Routing Administration Manager (RAMD); user interface for	rdc(1M)
routing daemon for IPv6; BGP	bgpd(1M)
routing daemon for IPv6; RIPng	ripngd(1M)
routing daemon, gateway	gated(1M)
routing daemon; IP multicast	mrouted(1M)
routing daemon; the Intermediate System to Intermediate System (IS-IS)	isisd(1M)
Routing header options manipulation functions, IPv6	inet6_rth_space(3N)
routing - system support for local network packet routing	routing(7)
routing tables; manually manipulate	route(1M)
routing, multicast, configuration information tool	mrinfo(1M)
RPC entry; get	getrpc(3C)
RPC information, report	rpcinfo(1M)
rpc - library routines for remote procedure calls	rpc(3N)
RPC port number, get	getrpcport(3N)
RPC program number mapper, universal addresses to	rpcbind(1M)
RPC protocols, generate C header files	rpcgen(1)
rpc - RPC program number database	rpc(4)

Index

All Volumes

Description	Entry Name(Section)
RPC servers	automountd(1M)
RPC version, get information on mechanisms and	rpc_gss_get_mechanisms(3N)
rpc, CLIENT handles, library routines for dealing with creation and manipulation of	rpc_clnt_create(3N)
rpc, library routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
RPC, library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
rpc, library routines for client side calls	rpc_clnt_calls(3N)
rpc, library routines for registering servers	rpc_svc_reg(3N)
rpc, library routines for remote procedure calls	rpc(3N)
rpc, library routines for RPC bind service	rpcbind(3N)
rpc, library routines for RPC servers	rpc_svc_calls(3N)
rpc, library routines for the creation of server handles	rpc_svc_create(3N)
rpc, obsolete library routines for RPC	rpc_soc(3N)
RPC, secure, change user's key	chkey(1)
RPC-based remote execution server	rexd(1M)
rpc.lockd - network lock daemon	lockd(1M)
rpc.mountd - NFS mount requests and access checks server	mountd(1M)
rpc.nisd_resolv - DNS service daemon for NIS	rpc.nisd_resolv(1M)
rpc.pcnfsd - PC-NFS authentication and print request server	pcnfsd(1M)
rpc.sprayd - spray server	sprayd(1M)
rpc.statd - network status monitor	statd(1M)
rpc.yppasswdd - daemon for modifying Network Information Service passwd database	yppasswdd(1M)
rpc.yppupdated, yppupdated, - server for changing NIS information	yppupdated(1M)
RPC; generates and validates GSS-API tokens for kernel	gssd(1M)
rpc_broadcast() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_broadcast_exp() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_call() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_clnt_auth() - library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
rpc_clnt_calls() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_clnt_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
rpc_control() - library routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
rpc_createerr() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
rpc_gss_get_error() - get error codes on failure	rpc_gss_get_error(3N)
rpc_gss_get_mech_info() - get information on mechanisms and RPC version	rpc_gss_get_mechanisms(3N)
rpc_gss_get_mechanisms() - get information on mechanisms and RPC version	rpc_gss_get_mechanisms(3N)
rpc_gss_get_principal_name() - get principal names at server	rpc_gss_get_principal_name(3N)
rpc_gss_get_versions() - get information on mechanisms and RPC version	rpc_gss_get_mechanisms(3N)
rpc_gss_getcred() - get credentials of client	rpc_gss_getcred(3N)
rpc_gss_is_installed() - get information on mechanisms and RPC version	rpc_gss_get_mechanisms(3N)
rpc_gss_max_data_length() - get maximum data length for transmission ..	rpc_gss_max_data_length(3N)
rpc_gss_mech_to_oid() - map mechanism, QOP strings to non-string values	rpc_gss_mech_to_oid(3N)
rpc_gss_qop_to_num() - map mechanism, QOP strings to non-string values	rpc_gss_mech_to_oid(3N)
rpc_gss_seccreate() - create a security context using the RPCSEC_GSS protocol	rpc_gss_seccreate(3N)
rpc_gss_set_callback() - specify callback for context	rpc_gss_set_callback(3N)
rpc_gss_set_defaults() - change service, QOP for a session	rpc_gss_set_defaults(3N)
rpc_gss_set_svc_name() - send a principal name to a server	rpc_gss_set_svc_name(3N)
rpc_gss_svc_max_data_length() - get maximum data length for transmission	rpc_gss_max_data_length(3N)
rpc_reg() - library routines for registering servers	rpc_svc_reg(3N)
rpc_soc() - obsolete library routines for RPC	rpc_soc(3N)
rpc_svc_calls() - library routines for RPC servers	rpc_svc_calls(3N)
rpc_svc_create() - library routines for the creation of server handles	rpc_svc_create(3N)
rpc_svc_err() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
rpc_svc_reg() - library routines for registering servers	rpc_svc_reg(3N)
rpc_xdr() - XDR library routines for remote procedure calls	rpc_xdr(3N)
rpcb_getaddr() - library routines for RPC bind service	rpcbind(3N)
rpcb_getmaps() - library routines for RPC bind service	rpcbind(3N)
rpcb_gettime() - library routines for RPC bind service	rpcbind(3N)
rpcb_rmtcall() - library routines for RPC bind service	rpcbind(3N)

Description	Entry Name(Section)
rpcb_set () - library routines for RPC bind service	rpcbind (3N)
rpcb_unset () - library routines for RPC bind service	rpcbind (3N)
rpcbind () - library routines for RPC bind service	rpcbind (3N)
rpcbind - universal addresses to RPC program number mapper	rpcbind (1M)
rpcgen - generate RPC protocols, C header files	rpcgen (1)
rpcinfo - report RPC information	rpcinfo (1M)
RPCSEC_GSS protocol, create a security context using the	rpc_gss_seccreate (3N)
rpcsec_gss () - RPCSEC_GSS security flavor library routines	rpcsec_gss (3N)
RPCSEC_GSS security flavor library routines	rpcsec_gss (3N)
rquotad - remote quota server	rquotad (1M)
RR scheduling policy	rtsched (2)
RR2 scheduling policy	rtsched (2)
rrestore - restore file system incrementally, local or across network	restore (1M)
RSE stack for any user process, maximum size (in bytes) of the	maxrsesssiz (5)
rsh - standard and restricted POSIX.2-conformant command shells	sh-posix (1)
rsqrt () - reciprocal square root function	rsqrt (3M)
rsqrtf () - reciprocal square root function (float)	rsqrt (3M)
rsqrtl () - reciprocal square root function (long double)	rsqrt (3M)
rsqrtq () - reciprocal square root function (quad)	rsqrt (3M)
rsqrtw () - reciprocal square root function (extended)	rsqrt (3M)
rstat () - get performance data from remote kernel	rstat (3N)
rstatd - kernel statistics server	rstatd (1M)
rtprio () - change or read real-time priority	rtprio (2)
rtprio - execute process with real-time priority	rtprio (1)
RTPRIO scheduling policy	rtsched (2)
rtradvd - Router Advertisement daemon for IPv6	rtradvd (1M)
rtradvd.conf - configuration file for router advertisement daemon	rtradvd.conf (4)
rtsched - execute process with POSIX real-time priority	rtsched (1)
rtsched - real-time scheduling operations	rtsched (2)
rtsched_numpri - number of priority values to support for POSIX.1b realtime applications	rtsched_numpri (5)
run a command at nondefault priority	nice (1)
run a command immune to hangups	nohup (1)
run daily accounting	runacct (1M)
run level	init (1M)
run serially with other processes; force target process to	serialize (1)
run serially with other processes; force target process to	serialize (2)
run-level s , place system in	shutdown (1M)
runacct - run daily accounting	runacct (1M)
running HP-UX; emulate PA-RISC HP-UX applications on Itanium-based systems	Aries (5)
running processes; get core images of	gcore (1)
running program or start program in particular memory window; change window ID of ...	setmemwindow (1M)
rup : show host status of local machines (RPC version)	rup (1)
ruptime - show status of local machines	ruptime (1)
ruserok () - verify a remote user as a local user	rcmd (3N)
rusers : determine who is logged in on local network machines	rusers (1)
rusers (): return information about users on remote machines	rnusers (3N)
rusersd - network username server	rusersd (1M)
rwall (): write to specified remote machines	rwall (3N)
rwall server, network	rwalld (1M)
rwall - write to all users over a network	rwall (1M)
rwalld - network rwall server	rwalld (1M)
rwho - show who is logged in on local machines	rwho (1)
rwhod - system status server	rwhod (1M)
sa1 - collect and output or store system activity report data	sa1 (1M)
sa2 - write daily system activity report in binary file	sa1 (1M)
sact - print current SCCS file editing activity	sact (1)
sad - STREAMS Administrative Driver	sad (7)
sadc - collect and output or store system activity report data	sa1 (1M)
SAM logfile, tool for viewing and saving	samlog_viewer (1)
sam - system administration manager	sam (1M)

Index

All Volumes

Description	Entry Name(Section)
samlog_viewer - tool for viewing and saving the SAM logfile	samlog_viewer(1)
sar - system activity reporter	sar(1M)
save a crash dump of the operating system	savecrash(1M)
save or restore file position indicator for a stream	fgetpos(3S)
save or restore program or shell terminal modes	def_prog_mode(3X)
save terminal modes as the "shell" state	def_prog_mode(3X)
save/restore stack environment for non-local goto	setjmp(3C)
save/restore terminal mode	resetty(3X)
savecrash - save a crash dump of the operating system	savecrash(1M)
saved group ID	glossary(9)
saved process group ID	glossary(9)
saved set-group-ID	glossary(9)
saved set-user-ID	glossary(9)
saved user ID	glossary(9)
saved user ID, get	getresuid(3)
saved, group ID, get	getresuid(3)
saved, real, and/or effective user or group IDs, set	setresuid(2)
saves operating system state to the file system for debugging purposes., a feature that	livedump(5)
savetty() - save/restore terminal mode	resetty(3X)
saving, viewing SAM logfile tool	samlog_viewer(1)
sbrk() - increase data segment space allocation	brk(2)
SCA system node, get information about an	pstat(2)
scalb() - scale exponent of a radix-independent floating-point number	scalb(3M)
scalbf() - scale exponent of a radix-independent floating-point number (float)	scalb(3M)
scalbl() - scale exponent of a radix-independent floating-point number (long double)	scalb(3M)
scalbln() - scale exponent of a radix-independent floating-point number	scalbln(3M)
scalblnf() - scale exponent of a radix-independent floating-point number	scalbln(3M)
scalblnl() - scale exponent of a radix-independent floating-point number	scalbln(3M)
scalblnq() - scale exponent of a radix-independent floating-point number	scalbln(3M)
scalblnw() - scale exponent of a radix-independent floating-point number	scalbln(3M)
scalbn() - scale exponent of a radix-independent floating-point number	scalbn(3M)
scalbnf() - scale exponent of a radix-independent floating-point number (float)	scalbn(3M)
scalbnl() - scale exponent of a radix-independent floating-point number (long double)	scalbn(3M)
scalbnq() - scale exponent of a radix-independent floating-point number (quad)	scalbn(3M)
scalbnw() - scale exponent of a radix-independent floating-point number (extended)	scalbn(3M)
scalbq() - scale exponent of a radix-independent floating-point number (quad)	scalb(3M)
scalbw() - scale exponent of a radix-independent floating-point number (extended)	scalb(3M)
scale exponent of a floating-point number	ldexp(3M)
scale exponent of a radix-independent floating-point number	scalb(3M)
scale exponent of a radix-independent floating-point number	scalbln(3M)
scale exponent of a radix-independent floating-point number	scalbn(3M)
scan a directory	scandir(3C)
scan physical volumes for LVM volume groups	vgscan(1M)
scan the I/O system	ioscan(1M)
scandir() - scan a directory	scandir(3C)
scanf() - formatted read from standard input stream file	scanf(3S)
scanning and processing language; pattern-directed	awk(1)
scanw() - convert formatted input from a window	mvscanw(3X)
scatter data to check the network	spray(3N)
SCCS commands, utility program for	scs(1)
SCCS commands; ask for help on	sccshelp(1)
SCCS delta, change delta commentary of	cdc(1)
SCCS deltas; combine	comb(1)
SCCS file	glossary(9)
SCCS file; compare two versions of an	sccsdiff(1)
SCCS file; format of	sccsfile(4)
SCCS file; get a version of an	get(1)
SCCS file; print and summarize an	prs(1)
SCCS file; remove a delta from an	rmdel(1)
SCCS files; create and administer	admin(1)
SCCS identification information; get	what(1)

Description	Entry Name(Section)
SCCS (Source Code Control System)	glossary(9)
sccs - utility program for SCCS commands	sccs(1)
SCCS: make a delta (change) to an SCCS file	delta(1)
SCCS: print current SCCS file editing activity	sact(1)
SCCS: undo a previous get of an SCCS file	unget(1)
SCCS: validate an SCCS file	val(1)
sccsdiff - compare two versions of an SCCS file	sccsdiff(1)
sccsfile - format of SCCS file	sccsfile(4)
sccshelp - ask for help on SCCS commands	sccshelp(1)
SCHED_FIFO scheduling policy	rtsched(2)
sched_get_priority_max() - return maximum for scheduling policy	rtsched(2)
sched_get_priority_min() - return minimum for scheduling policy	rtsched(2)
sched_getparam() - return scheduling parameters	rtsched(2)
sched_getscheduler() - return scheduling policy	rtsched(2)
SCHED_HPUX scheduling policy	rtsched(2)
SCHED_OTHER scheduling policy	rtsched(2)
SCHED_RR scheduling policy	rtsched(2)
SCHED_RR2 scheduling policy	rtsched(2)
sched_rr_get_interval() - update execution time limit	rtsched(2)
SCHED_RTPRIO scheduling policy	rtsched(2)
sched_setparam() - set scheduling parameters	rtsched(2)
sched_setscheduler() - set scheduling policy	rtsched(2)
sched_thread_affinity - adjust scheduler thread affinity	sched_thread_affinity(5)
SCHED_TIMESHARE scheduling policy	rtsched(2)
SCHED_TIMESHARE scheduling policy, set or get the scheduling timeslice value for PTHREAD_SCOPE_PROCESS threads with	pthread_gettimeslice_np(3T)
sched_yield() - force process to relinquish processor	rtsched(2)
schedule uucp transport files	uusched(1M)
scheduler daemons to run, number of STREAMS	nstrsched(5)
scheduler thread affinity; adjust	sched_thread_affinity(5)
scheduler; start the LP request	lpsched(1M)
scheduler; stop the LP request	lpsched(1M)
scheduling contention scope of threads, list of external options to specify the	pthread_scope_options(5)
scheduling operations; real-time	rtsched(2)
scheduling policy	rtsched(2)
scheduling policy, set or get the scheduling timeslice value for PTHREAD_SCOPE_PROCESS threads with SCHED_TIMESHARE	pthread_gettimeslice_np(3T)
scheduling priority	renice(1M)
scheduling priority of a thread; sets	pthread_setschedprio(3T)
schgr - SCSI interfaces for medium changer device	autochanger(7)
scr_dump() - screen file input/output functions	scr_dump(3X)
scr_init() - screen file input/output functions	scr_dump(3X)
scr_restore() - screen file input/output functions	scr_dump(3X)
scr_set() - screen file input/output functions	scr_dump(3X)
screen file input/output functions	scr_dump(3X)
screen handling and optimization functions; definitions for	curses(5)
screen initialisation functions	initscr(3X)
screen size information, specify source	use_env(3X)
screen viewing; file perusal filter for	more(1)
screen, clear terminal	clear(1)
screen, determine if it has been refreshed	isendwin(3X)
screen, free storage associated with a screen	delscreen(3X)
screen, number of columns	COLS(3X)
screen, number of lines on	LINES(3X)
screen-oriented mail interface	elm(1)
screen-oriented (visual) text editor	vi(1)
screen: flash the screen	flash(3X)
screens, switch between	set_term(3X)
script configuration utility for psfontpf; model	psmsgen(1M)
script for the init process	inittab(4)

Index

All Volumes

Description	Entry Name(Section)
script - make typescript of terminal session	script (1)
script programming	kermit (1)
script to set up user's environment at login, shell	profile (4)
scripts; symbolic translation file for localedef	charmap (4)
scr1() - scroll the window, enhanced curses	scr1 (3X)
scroll a curses window	scroll (3X)
scroll() - scroll a curses window	scroll (3X)
scroll the window, enhanced curses	scr1 (3X)
scroll_lines - number of scrollable lines used by the Internal Terminal Emulator	scroll_lines (5)
scrollable lines used by the Internal Terminal Emulator (ITE), number of	scroll_lines (5)
scrollok() - terminal output control functions	clearok (3X)
SCSI class driver eschgr plug-in for scsimgr	scsimgr_eschgr (7)
SCSI class driver esdisk plug-in for scsimgr	scsimgr_esdisk (7)
SCSI class driver estape plug-in for scsimgr	scsimgr_estape (7)
SCSI device drivers	scsi (7)
SCSI device; control	scsictl (1M)
SCSI direct access device driver	scsi_disk (7)
SCSI interfaces for medium changer device	autochanger (7)
SCSI management and diagnostic utility	scsimgr (1M)
SCSI pass-through device driver	sioc_io (7)
SCSI pass-through driver (esctl/sctl)	scsi_ctl (7)
SCSI sequential access device driver	scsi_tape (7)
scsi - Small Computer System Interface device drivers	scsi (7)
SCSI subsystem (OBSOLETE); enable and disable use of device's write cache in the	default_disk_ir (5)
scsi_ctl - SCSI pass-through driver (esctl/sctl)	scsi_ctl (7)
scsi_disk - SCSI direct access device driver	scsi_disk (7)
scsi_max_qdepth - maximum number of I/Os that target will queue up for execution (OBSOLETE)	scsi_max_qdepth (5)
scsi_maxphys - maximum allowed length of an I/O on all SCSI devices (OBSOLETE)	scsi_maxphys (5)
scsi_tape - SCSI sequential access device driver	scsi_tape (7)
scsictl - control a SCSI device	scsictl (1M)
scsimgr - SCSI management and diagnostic utility	scsimgr (1M)
scsimgr; SCSI class driver eschgr plug-in for	scsimgr_eschgr (7)
scsimgr; SCSI class driver esdisk plug-in for	scsimgr_esdisk (7)
scsimgr; SCSI class driver estape plug-in for	scsimgr_estape (7)
scsimgr_eschgr - SCSI class driver eschgr plug-in for scsimgr	scsimgr_eschgr (7)
scsimgr_esdisk - SCSI class driver esdisk plug-in for scsimgr	scsimgr_esdisk (7)
scsimgr_estape - SCSI class driver estape plug-in for scsimgr	scsimgr_estape (7)
sctl/esctl ; SCSI pass-through driver	scsi_ctl (7)
sd - all objects that Software Distributor (SD) uses, their attributes and storage formats	sd (4)
sd - create and monitor jobs	swjob (1M)
sd - create, distribute, install, monitor, and manage software	sd (5)
SD product from new SD media	swgettools (1M)
SD uses, their attributes and storage formats; all objects that Software Distributor	sd (4)
sdiff - compare two files and show differences side-by-side	sdiff (1)
sdisk - SCSI direct access device driver	scsi_disk (7)
search a file for a string or expression	grep (1)
search and print process accounting files	acctcom (1M)
search directory tree for files	find (1)
search environment list for value of specified variable name	getenv (3C)
search for files	find (1)
search for named file in named directories	pathfind (3G)
search for or replace legacy device special files or hardware paths	iofind (1M)
search or kill processes based on process name and attributes	pgrep (1)
search path for dynamically loadable kernel modules, change	modpath (2)
search physical volumes for LVM volume groups	vgscan (1M)
search routine, binary, for sorted tables	bsearch (3C)
search table for entry; optional update if missing	lsearch (3C)
search tables, hash, manage	hsearch (3C)
search tree, manage a binary	tsearch (3C)
secdef - security defaults configuration file routines	secdef (3)

Description	Entry Name(Section)
second kind; Bessel functions of the	y0(3M)
second, scheduling interval in clock ticks per	timeslice(5)
secondary prompt	glossary(9)
secret key, decrypt and store	keylogin(1)
secret key, delete key stored with keyserv	keylogout(1)
secret key, retrieve public or	getpublickey(3M)
section data for ELF files, manipulate	elf_getdata(3E)
section information for ELF files; get	elf_getscn(3E)
section sizes and allocation space of object files, print	size(1)
secure internet services with Kerberos authentication and authorization	sis(5)
secure internet services, configuration file	inetsvcs.conf(4)
secure internet services, enable or disable	inetsvcs_sec(1M)
secure password file entry on trusted systems; get	getspwent(3X)
secure password file; get entry from	getspwent(3X)
secure remote procedure calls, library routines for	secure_rpc(3N)
secure RPC key, change user's	chkey(1)
secure_rpc - library routines for secure remote procedure calls	secure_rpc(3N)
secure_sid_scripts - controls whether setuid and setgid bits on scripts are honored	secure_sid_scripts(5)
securenets (4) - NIS map security file	securenets(4)
securetty file	login(1)
Security Attributes Configuration tool; invokes the HP-UX	secweb(1M)
security attributes of a process; display	getprocxsec(1M)
security attributes of binary executable(s); display	getfilexsec(1M)
security attributes on a binary file; set extended	setfilexsec(1M)
security-bulletin compliance state of HP-UX 11.x system or depot; check	security_patch_check(1M)
security context using the RPCSEC_GSS protocol, create a	rpc_gss_seccreate(3N)
security context, delete	gss_delete_sec_context(3)
security context, establish	gss_accept_sec_context(3)
security context, establish between context initiator and context acceptor	gss_init_sec_context(3)
security context, obtain information about	gss_inquire_context(3)
security context, transfer to another process on a single machine	gss_export_sec_context(3)
security context, transfer to another process on a single machine	gss_import_sec_context(3)
security databases for trusted systems	authcap(4)
security defaults configuration file	security(4)
security defaults configuration file routines	secdef(3)
security file for ftpd	ftpusers(4)
security file, inetd optional	inetd.sec(4)
security file, map NIS	securenets(4)
security files authorizing access by remote hosts and users on local host	hosts.equiv(4)
security flavor library routines, RPCSEC_GSS	rpcsec_gss(3N)
security mechanism and quality of protection (QOP) files	mech(4)
security modes; list NFS	nfssec.conf(4)
security modes; overview of NFS	nfssec(5)
security - security defaults configuration file	security(4)
Security Service Application Programming Interface; Generic	gssapi(5)
Security Service, GSSAPI shared library	libgss(4)
security service, pass a context to	gss_process_context_token(3)
security_patch_check - check security-bulletin compliance state of HP-UX 11.x system or depot	security_patch_check(1M)
secweb - invokes the HP-UX Security Attributes Configuration tool	secweb(1M)
sed - stream text editor	sed(1)
seek; move read/write file pointer	lseek(2)
seekdir () - set position of next readdir () operation on named directory stream	directory(3C)
setreuid () - set real and effective user IDs	setreuid(2)
segment for any user process; maximum size (in bytes) of the data	maxdsiz(5)
segment for any user process; maximum size (in bytes) of the text	maxtsiz(5)
segment identifiers in the system, number of System V shared memory	shmmni(5)
segment, get information for a System V shared memory	pstat(2)
segment, get shared memory	shmget(2)
segment, maximum size (in bytes) for a System V shared memory	shmmax(5)

Index

All Volumes

Description	Entry Name(Section)
select - execute a shell select list	sh-posix(1)
select method of handling signal	sigset(3C)
select() - STREAMS enhancements to standard system calls	stream(2)
select() - synchronous I/O multiplexing	select(2)
select users to audit	audusr(1M)
select/reject lines common to two sorted files	comm(1)
selectable page size; maximum (in kilobytes) of user	vps_chatr_ceiling(5)
selected characters, alter, delete, modify, substitute, translate	tr(1)
selected fields of each line in a file, cut out (extract)	cut(1)
selected keys in Network Information Service map, print the values of	ypmatch(1)
selectively back up files	fbackup(1M)
selectively recover files	frecover(1M)
selects whether the system dumps memory pages compressed or uncompressed when a kernel panic occurs	dump_compress_on(5)
self-auditing process; write audit record for	auditwrite(2)
sem_close() - close a named semaphore	sem_close(2)
sem_destroy() - destroy an unnamed semaphore	sem_destroy(2)
sem_getvalue() - read a POSIX semaphore	sem_getvalue(2)
sem_init() - initialize an unnamed semaphore	sem_init(2)
sem_open() - open/create a named semaphore	sem_open(2)
sem_post() - unlock a POSIX semaphore	sem_post(2)
sem_trywait() - lock a POSIX semaphore without blocking	sem_wait(2)
sem_unlink() - unlink a named semaphore	sem_unlink(2)
sem_wait() - lock a POSIX semaphore	sem_wait(2)
sema - enable or disable System V IPC semaphores at boot time	sema(5)
semaem - maximum cumulative value changes per System V IPC semop() call	semaem(5)
semaphore control operations	semctl(2)
semaphore identifier (semid)	glossary(9)
semaphore identifiers, number of System V IPC system-wide	semnmi(5)
semaphore in mapped file or anonymous memory region, initialize	msem_init(2)
semaphore in mapped file or anonymous region, remove	msem_remove(2)
semaphore name	pipcrm(1)
semaphore operation permissions	glossary(9)
semaphore operations	semop(2)
semaphore set identifier, remove	ipcrm(1)
semaphore set, get information for a System V	pstat(2)
semaphore undo structures, number of System V IPC system-wide	semnmu(5)
semaphore, get information for a POSIX named	pstat(2)
semaphore, maximum value of any single System V IPC	semvmx(5)
semaphore, unlock a	msem_unlock(2)
semaphore; lock a	msem_lock(2)
semaphores and record locking on files, provide	lockf(2)
semaphores at boot time, enable or disable System V IPC	sema(5)
semaphores, get set of	semget(2)
semaphores, number of System V system-wide	semnms(5)
semaphores, report status	ipcs(1)
semaphores; report status	pipcs(1)
semctl() - semaphore control operations	semctl(2)
semget() - get set of semaphores	semget(2)
semid (semaphore identifier)	glossary(9)
semnmi - number of System V IPC system-wide semaphore identifiers	semnmi(5)
semnms - number of System V system-wide semaphores	semnms(5)
semnmu - number of System V IPC system-wide semaphore undo structures	semnmu(5)
semnmsl - maximum number of System V IPC semaphores per identifier	semnmsl(5)
semop() call, maximum cumulative value changes per System V IPC	semaem(5)
semop() - semaphore operations	semop(2)
semtimedop() - semaphore operations	semop(2)
semume - maximum number of System V IPC undo entries per process	semume(5)
semvmx - maximum value of any single System V IPC semaphore	semvmx(5)
send a message from a socket	send(2)
send a message on a stream	putmsg(2)

Description	Entry Name(Section)
send a message simultaneously to all users	wall(1M)
send a message to a message queue	mq_send(2)
send a signal to a process or a group of processes	sigsend(2)
send a signal to a thread	pthread_kill(3T)
send BOOTREQUEST to BOOTP server	bootpquery(1M)
send commands to the Terminal Session Manager, TSM	tsm.command(1)
send LP line printer request to a remote system	rlp(1M)
send mail to users or read mail	mail(1)
send message to message queue	msgop(2)
send() - send a message from a socket	send(2)
send signal to process	kill(1)
send signals to slpd	slpdc(1M)
send signals to the domain name server	sig_named(1M)
send test packets	ping(1M)
send the contents of a file through a socket	sendfile(2)
send the contents of a Large File through a socket	sendfile64(2)
send_sound - play an audio file	send_sound(1)
sender; summarize mail folders by subject and	mailfrom(1)
sendfile() - send the contents of a file through a socket	sendfile(2)
sendfile, maximum number of Buffer Cache Pages used by	sendfile_max(5)
sendfile64() - send the contents of a Large File through a socket	sendfile64(2)
sendfile_max - maximum number of Buffer Cache Pages used by sendfile	sendfile_max(5)
sendmail aliases, print system-wide	praliases(1)
sendmail aliases, recursively expands	expand_alias(1)
sendmail connection timeout value of a filter; sets the	smfi_settimeout(3N)
sendmail connection; sets the private data pointer for the	smfi_setpriv(3N)
sendmail daemon, killing it	killsm(1M)
sendmail database maps, creating	makemap(1M)
sendmail macro; gets the value of a	smfi_getsymval(3N)
sendmail message body; replaces the data in the	smfi_replacebody(3N)
sendmail message using the given reason; quarantines the	smfi_quarantine(3N)
sendmail message; prepends a header to the current	smfi_inshdr(3N)
sendmail message; removes a recipient from envelope of current	smfi_delrcpt(3N)
sendmail operation is still in progress; notifies the MTA that a	smfi_progress(3N)
sendmail - send mail over the Internet	sendmail(1M)
sendmail, registers a set of filter callbacks	smfi_register(3N)
sendmail, sets the debugging level for the Milter library for sendmail	smfi_setdbg(3N)
sendmail, sets the listen backlog value of the filter	smfi_setbacklog(3N)
sendmail, starts an orderly shutdown of the Milter	smfi_stop(3N)
sendmail.cf files, convert to new format	convert_awk(1M)
sendmail; aliases file for	aliases(5)
sendmail; restricted shell for	smrsh(1M)
sendmail; sets the socket for filter to communicate with	smfi_setconn(3N)
sendmsg() - send a message from a socket	send(2)
sendto() - send a message from a socket	send(2)
separate a file into multiple <i>n</i> -line pieces	split(1)
separate floating-point number into mantissa and exponent	frexp(3M)
separate mirrored LVM logical volume into two logical volumes	lvsplit(1M)
sequence for object code files in a library, find optimum	lorder(1)
sequential access device driver; SCSI	scsi_tape(7)
sequential accesses, per system-wide limit; percentage of file cache that can be consumed by	fcache_seqlimit_system(5)
sequential accesses, per-file limit; percent of file cache that can be consumed by	fcache_seqlimit_file(5)
sequential archive member access for ELF files, provide	elf_next(3E)
Serial and HP AdvanceLink server, Basic	pcserver(1M)
serial connections	kermit(1)
serial modem line control; asynchronous	modem(7)
serialize() - force target process to run serially with other processes	serialize(2)
serialize - force target process to run serially with other processes	serialize(1)
serially with other processes; force target process to run	serialize(1)

Index

All Volumes

Description	Entry Name(Section)
serially with other processes; force target process to run	serialize(2)
server configuration file, PPPoE (Point to Point Protocol over Ethernet)	pppoesd.conf(4)
server control utility; name	rncd(1)
server daemon, PPPoE (Point-to-Point Protocol over Ethernet)	pppoesd(1M)
server daemon; UUCP over TCP/IP	uucpd(1M)
server file format; translate host table to name	hosts_to_named(1M)
server for changing NIS information	ypupdated(1M)
server for information about NIS map; query NIS	ypoll(1M)
server for storing private encryption keys	keyserv(1M)
server or map master; list which host is Network Information System	ypwhich(1)
server to local node; transfer NIS database from	ypxfr(1M)
server, Basic Serial and HP AdvanceLink	pcserver(1M)
server, binder, and transfer processes; Network Information Service (NIS)	ypserv(1M)
server, FTP server logfile	xferlog(5)
server, get principal names at	rpc_gss_get_principal_name(3N)
server, kernel statistics	rstatd(1M)
server, network rwall	rwalld(1M)
server, network username	rusersd(1M)
server, remote quota	rquotad(1M)
server, remote user information	fingerd(1M)
server, RPC-based remote execution	rexid(1M)
server, send a principal name to a	rpc_gss_set_svc_name(3N)
server, send BOOTREQUEST to BOOTP	bootquery(1M)
server, spray	sprayd(1M)
server; bind to particular Network Information Service	ypset(1M)
server; configuration file for Internet domain name	named.conf(4)
server; file transfer protocol	ftpd(1M)
server; Internet Boot Protocol	bootpd(1M)
server; Internet domain name	named(1M)
server; remote execution	rexc(1M)
server; remote login	rlogind(1M)
server; remote shell	remshd(1M)
server; send signals to the domain name	sig_named(1M)
server; spray	sprayd(1M)
server; start the audio	aserver(1M)
server; system status	rwhod(1M)
server; TELNET protocol	telnetd(1M)
server; trivial file transfer protocol	tftpd(1M)
servers	automountd(1M)
servers interactively; query name	nslookup(1)
servers, library routines for registering servers, rpc	rpc_svc_reg(3N)
servers, library routines for RPC servers	rpc_svc_calls(3N)
servers, library routines for server side remote procedure call errors	rpc_svc_err(3N)
servers, library routines for the creation of server handles, rpc	rpc_svc_create(3N)
Service Application Programming Interface; Generic Security	gssapi(5)
Service client interface; Network Information	ypclnt(3C)
Service database; force propagation of Network Information	yppush(1M)
service entry, get, set, or end	getservent(3N)
service functions; event	EvmSrvStart(3)
Service Location Protocol Daemon	slpd(1M)
Service Location Protocol library routines, SLP	libslp(3N)
Service Location Protocol (SLP) error codes	SLPError(3N)
service module APIs, PAM	pam_sm(3)
service module for HP-UX, extended authentication, account, password, and session	pam_hpsec(5)
service module, PAM user policy definition	pam_updbe(5)
service name database	services(4)
Service (NIS) server, binder, and transfer processes; Network Information	ypserv(1M)
service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
service provider implementation for pam_authenticate()	pam_sm_authenticate(3)
service provider implementation for pam_chauthtok()	pam_sm_chauthtok(3)
service provider implementation for pam_open_session() and pam_close_session()	

Description	Entry Name(Section)
service provider implementation for <code>pam_setcred()</code>	<code>pam_sm_open_session</code> (3)
Service Provider Interface; ACPS	<code>pam_sm_setcred</code> (3)
Service Provider Interface; ACPS	<code>acps_spi</code> (3)
service rights	<code>evm.auth</code> (4)
Service server; bind to particular Network Information	<code>ypset</code> (1M)
service switch	<code>service.switch</code> (1M)
Service Type Syntax, SLP	<code>slp_syntax</code> (7)
service <code>vt</code> requests from other systems	<code>vtdaemon</code> (1M)
service, QOP for a session, change	<code>rpc_gss_set_defaults</code> (3N)
service, reminder	<code>calendar</code> (1)
service.switch - indicate lookup sources and fallback mechanism	<code>service.switch</code> (1M)
services daemon, PCI I/O hotplug (attention button)	<code>hotplugd</code> (1M)
services daemon; Internet	<code>inetd</code> (1M)
Services Monitor Daemon; Essential	<code>esmd</code> (1M)
services - service name database	<code>services</code> (4)
services with Kerberos authentication and authorization; secure internet	<code>sis</code> (5)
services.window - file containing applications and their associated memory window ID	<code>services.window</code> (4)
session	<code>glossary</code> (9)
session creation and termination operations, PAM	<code>pam_open_session</code> (3)
session ID, get	<code>getsid</code> (2)
session ID, get terminal	<code>tcgetsid</code> (3C)
session leader	<code>glossary</code> (9)
session lifetime	<code>glossary</code> (9)
Session Manager state information, get Terminal	<code>tsm.info</code> (1)
Session Manager, Terminal	<code>tsm</code> (1)
session service module for HP-UX, extended authentication, account, password, and	<code>pam_hpsec</code> (5)
session, and password management PAM modules for LDAP; authentication, account,	<code>pam_ldap</code> (5)
session, authentication, account, and password management PAM modules for UNIX	<code>pam_unix</code> (5)
session, change service, QOP for a	<code>rpc_gss_set_defaults</code> (3N)
session, create and set process group ID	<code>setsid</code> (2)
session; make typescript of terminal	<code>script</code> (1)
session; start terminal	<code>login</code> (1)
set a file's Access Control List (ACL) information; JFS File Systems only	<code>acl</code> (2)
set a process's alarm clock	<code>alarm</code> (2)
set access control list (ACL) information	<code>setacl</code> (2)
set access permissions mode mask for file-creation	<code>umask</code> (1)
set and clear window attributes	<code>standend</code> (3X)
set and get concurrency level of unbound threads	<code>pthread_getconcurrency</code> (3T)
set and get process' compartment	<code>cmpt_change</code> (3)
set and get the scheduling policy and associated parameters	<code>pthread_getschedparam</code> (3T)
set and get the thread-specific data associated with a key	<code>pthread_getspecific</code> (3T)
set and retrieve the current thread's cancelability state and type	<code>pthread_setcancelstate</code> (3T)
set and/or get signal alternate stack context	<code>sigaltstack</code> (2)
set attributes for pthread	<code>pthread_attr_getdetachstate</code> (3T)
set <code>cchar_t</code> from a wide-character string and rendition	<code>setcchar</code> (3X)
set compartment rules	<code>setrules</code> (1M)
set control; processor	<code>pset_ctl</code> (2)
set crash dump node number	<code>cr_set_node</code> (3)
set current events and system calls to be audited	<code>setevent</code> (2)
set current user context; DEPRECATED; get and	<code>getcontext</code> (2)
set effective user and group IDs	<code>seteuid</code> (2)
set environment for command execution	<code>env</code> (1)
set extended security attributes on a binary file	<code>setfilexsec</code> (1M)
set file access and modification times	<code>utimes</code> (2)
set file access and modification times	<code>utime</code> (2)
set fill byte for ELF files	<code>elf_fill</code> (3E)
set foreground process group ID	<code>tcsetpgrp</code> (3C)
set group access list	<code>setgroups</code> (2)
set name of current host system	<code>hostname</code> (1)
set name of current host system	<code>sethostname</code> (2)

Index

All Volumes

Description	Entry Name(Section)
set name of current NIS domain; get or	getdomainname(2)
set network entry	getnetent(3N)
set network group entry	getnetgrent(3C)
set network host entry	gethostent(3N)
set Network Information Service domain name	domainname(1)
set node name	uname(1)
set node name (system name)	uname(2)
set of semaphores, get	semget(2)
set options on sockets; get and	getsockopt(2)
set or get audit files; start or halt the auditing system and	audctl(2)
set or get background character and rendition using a complex character	bkgrnd(3X)
set or get background character and rendition using a single-byte character	bkgd(3X)
set or get the scheduling timeslice value for PTHREAD_SCOPE_PROCESS threads with SCHED_TIMESHARE scheduling policy	pthread_gettimeslice_np(3T)
set or get the thread process-shared attribute	pthread_condattr_getpshared(3T)
set or get tty baud rate	cfspeed(3C)
set printing options for a non-serial printer	slp(1)
set process group ID for job control	setpgid(2)
set process group ID, create session and	setsid(2)
set process priority	getpriority(2)
set real and effective user IDs	setreuid(2)
set real, effective, and/or saved user or group IDs	setresuid(2)
set scheduling parameters	rtsched(2)
set scheduling policy	rtsched(2)
set service entry	getservent(3N)
set - set/define flags and arguments	csh(1)
set - set/define options and arguments	ksh(1)
set - set/define options and arguments	sh-posix(1)
set signal alternate stack context	sigaltstack(2)
set special attributes for group	getprivgrp(2)
set special privileges for groups	setprivgrp(1M)
set system initial identity parameters: hostname, date/time, root password, and networking	set_parms(1M)
set system log file priority mask	syslog(3C)
set system name	uname(1)
set tabs on a terminal	tabs(1)
set terminal type, modes, speed and line discipline for 2-way line	uugetty(1M)
set terminal type, modes, speed, and line discipline	getty(1M)
set the audit ID (aid) for the current process	setaudit(2)
set the blocking status of a message queue associated with a descriptor	mq_setattr(2)
set the cursor mode	curs_set(3X)
set the date and time	settimeofday(2)
set the default message catalog	setcat(3)
set the dynamic search path used to locate shared libraries	dldsetlibpath(3C)
set the interval timer	ualarm(2)
set the locale of a program	setlocale(3C)
set the options for a terminal port	stty(1)
set the prioceiling attribute	pthread_mutexattr_getprotocol(3T)
set the prioceiling of a mutex	pthread_mutex_getprioceiling(3T)
set the process-shared attribute	pthread_mutexattr_getpshared(3T)
set the process-shared attribute	pthread_rwlockattr_getpshared(3T)
set the protocol attribute	pthread_mutexattr_getprotocol(3T)
set the type attribute	pthread_mutexattr_getpshared(3T)
set time and date	stime(2)
set tty device operating parameters	tcattribute(3C)
set user or group IDs	setuid(2)
set value of process interval timer	getitimer(2)
set value of system-wide clock	setclock(3C)
set, or end protocol entry; get,	getprotoent(3N)
set, remove, and retrieve a process' privileges; add,	priv_add(3)
set-group-ID bit	glossary(9)

Description	Entry Name(Section)
set-user-ID bit	glossary(9)
set/display audit trail information	audsys(1M)
set: file creation (permissions) mask, set and get	umask(2)
set: file size limits and break value, get or set	ulimit(2)
set_curterm() - interface to terminfo database	del_curterm(3X)
set_parms - set system initial identity parameters: hostname, date/time, root password, and networking	set_parms(1M)
set_parms special initialization script	hostname(1)
set_resfield() - resolver routines	resolver(3N)
set_term() - switch between screens	set_term(3X)
setacl - modify access control lists for files (JFS only)	setacl(1)
setacl() - set access control list (ACL) information	setacl(2)
setaclentry() - add, modify, or delete access control list entry	setaclentry(3C)
setauditid() - set the audit ID (aid) for the current process	setaudit(2)
setaudproc() - controls process level auditing for the current process and its decedents	setaudproc(2)
setauduser() - start auditing the current process as owned by a given user	setauduser(3)
setboot - display and modify boot variables in stable storage	setboot(1M)
setbuf() - assign buffering to a stream file	setbuf(3S)
setbwtent() - write records into new wtmps and btmps database	bwtmps(3C)
setcat() - set the default message catalog	setcat(3)
setcchar() - set cchar_t from a wide-character string and rendition	setcchar(3X)
setclock() - set value of system-wide clock	setclock(3C)
setcontext() - get and set current user context; DEPRECATED	getcontext(2)
setdomainname() - get or set name of current NIS domain	getdomainname(2)
setdvagent() - set device assignment database entry for trusted system	getdvagent(3)
setegid() - set effective group IDs	seteuid(2)
setenv() - change or add a variable to environment	setenv(3C)
setenv - define environment variable	cs(1)
seteuid() - set effective user IDs	seteuid(2)
setevent() - set current events and system calls to be audited	setevent(2)
setfilexsec - set extended security attributes on a binary file	setfilexsec(1M)
setfsent() - open and rewind file system descriptor file	getfsent(3X)
setgid bits on scripts are honored; controls whether setuid and	secure_sid_scripts(5)
setgid() - set group ID	setuid(2)
setgrent() - rewind pointer to first entry in group file	getgrent(3C)
setgroups() - set group access list	setgroups(2)
sethostent() - set network host entry	gethostent(3N)
sethostent_r() - set network host entry (thread-safe)	gethostent(3N)
sethostname() - set name of current host system	sethostname(2)
setitimer() - set value of process interval timer	getitimer(2)
setjmp() - save stack environment for non-local goto	setjmp(3C)
setlabel() - define label for formatting routines	setlabel(3)
setlinebuf() - assign buffering to a stream file	setbuf(3S)
setlocale() - set the locale of a program	setlocale(3C)
setlocale_r() - set the locale of a program (MT-Safe)	setlocale(3C)
setlogmask() - control system log	syslog(3C)
setmemwindow - change window ID of running program or start program in particular memory window	setmemwindow(1M)
setmntent() - open a file system descriptor file	getmntent(3X)
setnetconfig() - get /etc/netconfig entry corresponding to NETPATH component	getnetpath(3N)
setnetconfig() - get network configuration data base entry	getnetconfig(3N)
setnetent() : set network entry	getnetent(3N)
setnetgrent() - get network group entry	getnetgrent(3C)
setoncnv - NFS environment configuration command	setoncnv(1M)
setpgid() - set process group ID for job control	setpgid(2)
setppgrp() - 4.2 BSD-compatible process control facilities	killpg(2)
setppgrp() - set process group ID	setppgrp(2)
setppgrp2() - set process group ID for job control	setppgrp(2)
setppgrp3() - create session and set process group ID	setsid(2)
setprdfent() - manipulate system default database entry for a trusted system	getprdfent(3)

Index

All Volumes

Description	Entry Name(Section)
setprdfent () - rewind default control file for system default database for trusted system	getprdfent (3)
setpriority () - set process priority	getpriority (2)
setprivgrp () - set special attributes for group	getprivgrp (2)
setprivgrp - set special privileges for groups	setprivgrp (1M)
setprotoent () - set protocol entry	getprotoent (3N)
setprotoent_r () - set protocol entry (thread-safe)	getprotoent (3N)
setprpwent () - manipulate protected password database entries (for trusted systems only)	getprpwent (3)
setprpwent () - set protected password database entry (trusted systems)	getprpwent (3)
setprtcent () - manipulate terminal control database entry	getprtcent (3)
setpwent () - rewind pointer to beginning of password file	getpwent (3C)
setregid () - sets the real and effective group IDs	setregid (2)
setresgid () - set real, effective, and/or saved group IDs	setresuid (2)
setresuid () - set real, effective, and/or saved user IDs	setresuid (2)
setrlimit () - set system resource consumption limit	getrlimit (2)
setrlimit64 () - non-POSIX standard API interfaces to support large files	creat64 (2)
setrpcent () - get RPC entry	getrpcent (3C)
setrules - set compartment rules	setrules (1M)
sets action taken if IPMI watchdog timer expires	ipmi_watchdog_action (5)
sets default SMTP error reply code to a multi-line response	smfi_setmlreply (3N)
sets scheduling priority of a thread	pthread_setschedprio (3T)
sets the debugging level for the Milter library for sendmail	smfi_setdbg (3N)
sets the default SMTP error reply code	smfi_setreply (3N)
sets the listen backlog value of the filter for sendmail	smfi_setbacklog (3N)
sets the private data pointer for the sendmail connection	smfi_setpriv (3N)
sets the real and effective group IDs	setregid (2)
sets the sendmail connection timeout value of a filter	smfi_settimeout (3N)
sets the socket for filter to communicate with sendmail	smfi_setconn (3N)
sets the values of kernel tunable parameters in a transaction	settune_txn (2)
sets; create and manage processor	psrset (1M)
setscrreg () - terminal output control functions	clearok (3X)
setservent () - set service entry	getservent (3N)
setservent_r () - set service entry (thread-safe)	getservent (3N)
setsid () - create session and set process group ID	setsid (2)
setsockopt () - set options on sockets	getsockopt (2)
setspent () - rewind pointer to beginning of secure password file	getspent (3C)
setspwent () - rewind pointer to beginning of secure password file on trusted systems	getspwent (3X)
setspwent_r () - get secure password file entry on trusted systems	getspwent (3X)
setstate (), random (), srandom (), initstate () - generate a pseudorandom number	random (3M)
settimeofday () - set the date and time	settimeofday (2)
settings, core file change	coreadm (1M)
settings, core file change	coreadm (2)
settings, terminal, and datacomm line speed used by getty	gettydefs (4)
settune () - set value of a kernel tunable parameter	settune (2)
settune_txn () - sets the values of kernel tunable parameters in a transaction	settune_txn (2)
setuid and setgid bits on scripts are honored; controls whether	secure_sid_scripts (5)
setuid () - set user ID	setuid (2)
setuname - change machine information	setuname (1M)
setuname () - set node name (system name)	uname (2)
setupterm () - interface to terminfo database	del_curterm (3X)
setusershell () - rewind legal user shells file	getusershell (3C)
setutent () - reset input stream to beginning of utmp file	getut (3C)
setutent_r () - reset input stream to beginning of utmp file	getut (3C)
SETUTSENT () - access and update routines for user-accounting database maintained by utmpd	getuts (3C)
setutsent () - access and update routines for user-accounting database maintained by utmpd	getuts (3C)
setutxent () - reset input stream to beginning of utmpx file	getutx (3C)
setvbuf () - assign buffering to a stream file	setbuf (3S)
severities, define additional	addsev (3C)
sfinger - utility programs for TCP Wrappers	tryfrom (1)
sh - overview of various system shells	sh (1)
sh - standard and restricted POSIX.2-conformant command shells	sh-posix (1)

Description	Entry Name(Section)
sh-posix - standard and restricted POSIX.2-conformant command shells	sh-posix(1)
shadow password entries; access	getspent(3C)
shadow password file	shadow(4)
shadow - shadow password file	shadow(4)
shadow to nonshadow, convert passwords from	pwunconv(1M)
shar - make a shell archive package	shar(1)
share - make local NFS file systems available for mounting by remote systems	share_nfs(1M)
share - make local resource available for mounting by remote systems	share(1M)
share, unshare multiple resources	shareall(1M)
share/unshare commands; translates exportfs options to	exportfs(1M)
share_nfs - make local NFS file systems available for mounting by remote systems	share_nfs(1M)
shareall - share, unshare multiple resources	shareall(1M)
shared file system table	sharetab(4)
shared libraries for Integrity systems; explicit load of	shl_load_ia(3X)
shared libraries for PA-RISC systems; explicit load of	shl_load_pa(3X)
shared libraries on Integrity systems; list dynamic dependencies of executable files or	ldd_ia(1)
shared libraries on PA-RISC systems; list dynamic dependencies of executable files or	ldd_pa(1)
shared libraries programs, prepare for faster program start-up	fastbind(1)
shared libraries; explicit load of	shl_load(3X)
shared libraries; list dynamic dependencies of executable files or	ldd(1)
shared library	glossary(9)
shared library for GSSAPI (Generic Security Service)	libgss(4)
shared library loaded module	dlget(3C)
shared library loaded module	dlgetmodinfo(3C)
shared library loaded module	dlmodinfo(3C)
shared library on Integrity systems; open a	dlopen_ia(3C)
shared library with explicit load address; open an HP 9000 64-bit	dlopen_pa(3C)
shared library, lock in memory	plock(2)
shared library; open a	dlopen(3C)
shared library; open an HP 9000	dlopen_pa(3C)
shared memory and data segment, attach or detach	shmop(2)
shared memory control operations	shmctl(2)
shared memory identifier (shmid)	glossary(9)
shared memory identifier, remove	ipcrm(1)
shared memory in a process core dump, determines the inclusion of readable	core_addshmem_read(5)
shared memory in process core dump, determines the inclusion of read/write	core_addshmem_write(5)
shared memory operation permissions	glossary(9)
shared memory segment identifiers in the system, number of System V	shmmni(5)
shared memory segment, get	shmget(2)
shared memory segment, get information for a System V	pstat(2)
shared memory segment, maximum size (in bytes) for a System V	shmmax(5)
shared memory segments per process, maximum number of System V	shmseg(5)
shared memory segments, report status	ipcs(1)
shared memory, enable or disable System V	shm(5)
shared memory, enable or disable System V	shm(5)
shared object, close	dlclose(3C)
shared object, get address of symbol	dlsym(3C)
shared object, return file information	dlgetfileinfo(3C)
shared object, set the dynamic search path	dlsetlibpath(3C)
shared objects explicitly attached using dlopen() / shl_load() ; list the dynamic libraries linked into each process, including	pldd(1)
shared tape administration	st(1M)
sharetab - shared file system table	sharetab(4)
sharing resources across a network; file containing commands for	dfstab(4)
shell	glossary(9)
shell archive package, make a	shar(1)
shell (command interpreter) with C-like syntax	csh(1)
shell command, issue a	system(3S)
SHELL environment variable	login(1)
shell for sendmail; restricted	smrsh(1M)
shell layer manager	shl(1)

Index

All Volumes

Description	Entry Name(Section)
shell layer manager	shl(1)
shell procedures for system accounting	acctsh(1M)
shell program	glossary(9)
shell script	glossary(9)
shell script to set up user's environment at login	profile(4)
shell server; remote	remshd(1M)
shell terminal modes; save or restore program or	def_prog_mode(3X)
shell, context-sensitive softkey	keysh(1)
shell; change default login	chsh(1)
shell; execute from a remote	remsh(1)
shell; execute from a remote	rexec(1)
shells - list of allowed login shells	shells(4)
shells, get legal user	getusershell(3C)
shells, list of allowed login	shells(4)
shells, overview of various system	sh(1)
shells; standard and restricted POSIX.2-conformant command	sh-posix(1)
shift - shift <i>argv</i> members one position to left	csh(1)
shift - shift <i>argv</i> members one position to left	ksh(1)
shift - shift <i>argv</i> members one position to left	sh-posix(1)
shl - shell layer manager	shl(1)
shl_definesym() - explicit load of shared libraries	shl_load(3X)
shl_definesym() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_definesym() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_findsym() - explicit load of shared libraries	shl_load(3X)
shl_findsym() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_findsym() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_get() - explicit load of shared libraries	shl_load(3X)
shl_get() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_get() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_get_r() - explicit load of shared libraries	shl_load(3X)
shl_get_r() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_get_r() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_gethandle() - explicit load of shared libraries	shl_load(3X)
shl_gethandle() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_gethandle() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_gethandle_r() - explicit load of shared libraries	shl_load(3X)
shl_gethandle_r() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_gethandle_r() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_getsymbols() - explicit load of shared libraries	shl_load(3X)
shl_getsymbols() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_getsymbols() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_load() - explicit load of shared libraries	shl_load(3X)
shl_load() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_load() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_load_ia - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_load_pa - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shl_unload() - explicit load of shared libraries	shl_load(3X)
shl_unload() - explicit load of shared libraries for Integrity systems	shl_load_ia(3X)
shl_unload() - explicit load of shared libraries for PA-RISC systems	shl_load_pa(3X)
shm_open() - create/open a shared memory object	shm_open(2)
shm_unlink() - unlink a shared memory object	shm_unlink(2)
shmat() - attach shared memory to data segment	shmop(2)
shmctl() - shared memory control operations	shmctl(2)
shmdt() - detach shared memory from data segment	shmop(2)
shmем - enable or disable System V shared memory	shmем(5)
shmget() - get shared memory segment	shmget(2)
shmид (shared memory identifier)	glossary(9)
shmmax - maximum size (in bytes) for a System V shared memory segment	shmmax(5)
shmmni - number of System V shared memory segment identifiers in the system	shmmni(5)
shmseg - maximum number of System V shared memory segments per process	shmseg(5)
shortest path and route between hosts, compute	pathalias(1)

Description	Entry Name(Section)
show all remote mounts	showmount(1M)
show current number of users for each class	ftpcount(1)
show current process information for each ftp user	ftpwho(1)
show disk usage	du(1)
show file differences side-by-side	sdiff(1)
show group memberships	groups(1)
show how long system has been up	uptime(1)
show who is logged in on local machines	rwho(1)
show_patches - HP-UX patch display utility	show_patches(1)
showmount - show all remote mounts	showmount(1M)
shut down a socket	shutdown(2)
shut down and reboot the system	reboot(1M)
shutacct - turn off accounting for system shutdown	acctsh(1M)
shutdown message file created by ftpshut	ftprestart(1)
shutdown message file for ftp servers	ftpshut(1)
shutdown of the Milter for sendmail; starts an orderly	smfi_stop(3N)
shutdown() - shut down a socket	shutdown(2)
shutdown status of HFS file systems; determine the	fsclean(1M)
shutdown - terminate all processing	shutdown(1M)
sig_named - send signals to the domain name server	sig_named(1M)
sigaction() - examine and change signal action	sigaction(2)
sigaddset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sigaltstack() - set and/or get signal alternate stack context	sigaltstack(2)
sigdelset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sigemptyset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sigfillset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sighold() - signal management	signal(2)
sigold() - signal management	sigset(3C)
SIGHUP	evmreload(1M)
sigignore() - signal management	signal(2)
sigignore() - signal management	sigset(3C)
siginterrupt() - allow signals to interrupt functions	siginterrupt(2)
sigismember() - initialize, manipulate, and test signal sets	sigsetops(3C)
sign on	login(1)
sign on; start terminal session	login(1)
sign-determination, floating-point	signbit(3M)
signal	glossary(9)
signal() - 4.2 BSD-compatible process control facilities	killpg(2)
signal action, examine and change	sigwait(2)
signal action; examine and change	sigaction(2)
signal action; restore	sigset(3C)
signal alternate stack context, set and/or get	sigaltstack(2)
signal - description of signals	signal(5)
signal management	sigset(3C)
signal mask, examine and change, of the calling thread	pthread_sigmask(3T)
signal received; suspend calling process until	sigset(3C)
signal sets, initialize, manipulate, and test	sigsetops(3C)
signal() - specify what to do upon receipt of a signal	signal(2)
signal stack space, define, delete, or get amount of	sigspace(2)
signal() - STREAMS enhancements to standard system calls	stream(2)
signal() system call, 4.2 BSD-compatible	bsdproc(3C)
signal to a process; queue a	sigqueue(2)
signal to a thread, send	pthread_kill(3T)
signal to process	kill(1)
signal to process or group of processes; send	kill(2)
signal upon receipt; hold	sigset(3C)
signal, audible	beep(3X)
signal, define what to do upon receipt of a	signal(2)
signal, raise a software	ssignal(3C)
signal, send to a process or a group of processes	sigsend(2)
signal, SIGHUP	evmreload(1M)

Index

All Volumes

Description	Entry Name(Section)
signal, suspend process until	pause(2)
signal, wait for a	sigsuspend(2)
signal.h - description of signals	signal(5)
signal; ignore	sigset(3C)
signal; select method of handling	sigset(3C)
signals allowed to interrupt functions	siginterrupt(2)
signals per process, limit on number of queued	ksi_send_max(5)
signals that can be allocated, system-wide limit of queued	ksi_alloc_max(5)
signals to the domain name server; send	sig_named(1M)
signals, blocked, examine and change	sigprocmask(2)
signals, examine pending	sigpending(2)
signals, release blocked and atomically wait for interrupt	sigpause(3C)
signals; description of	signal(5)
signbit() - sign-determination	signbit(3M)
signgam() - sign of gamma from log gamma function	lgamma(3M)
signing tool; DNSSEC zone	dnsec-sigzone(1)
sigpause() - atomically release blocked signals and wait for interrupt	sigpause(3C)
sigpending() - examine pending signals	sigpending(2)
sigprocmask() - examine and change blocked signals	sigprocmask(2)
sigqueue() - queue a signal to a process	sigqueue(2)
sigrelse() - signal management	signal(2)
sigrelse() - signal management	sigset(3C)
sigsend() - send a signal to a process	sigsend(2)
sigsendset() - send a signal to a group of processes	sigsend(2)
sigset() - signal management	signal(2)
sigset() - signal management	sigset(3C)
sigsetjmp() - save signal mask if savemask is non-zero	setjmp(3C)
sigspace() - define or delete additional signal stack space	sigspace(2)
sigsuspend() - wait for a signal	sigsuspend(2)
sigtimedwait() - synchronously accept a signal	sigwait(2)
sigvec() - 4.2 BSD-compatible process control facilities	killpg(2)
sigwait() - synchronously accept a signal	sigwait(2)
sigwaitinfo() - synchronously accept a signal	sigwait(2)
simple add, modify, and delete entries in an LDAP directory	ldapentry(1)
simple command; execute a	command(1)
simple text formatter	fmt(1)
sin() - sine function	sin(3M)
sincos() - both sine and cosine	sincos(3M)
sincosd() - both sine and cosine of degree argument	sincosd(3M)
sincosdf() - both sine and cosine of degree argument (float)	sincosd(3M)
sincosdl() - both sine and cosine of degree argument (long double)	sincosd(3M)
sincosdq() - both sine and cosine of degree argument (quad)	sincosd(3M)
sincosdw() - both sine and cosine of degree argument (extended)	sincosd(3M)
sincosf() - both sine and cosine (float)	sincos(3M)
sincosl() - both sine and cosine (long double)	sincos(3M)
sincosq() - both sine and cosine (quad)	sincos(3M)
sincosw() - both sine and cosine (extended)	sincos(3M)
sind() - sine function of degree argument	sind(3M)
sindf() - sine function of degree argument (float)	sind(3M)
sindl() - sine function of degree argument (long double)	sind(3M)
sindq() - sine function of degree argument (quad)	sind(3M)
sindw() - sine function of degree argument (extended)	sind(3M)
sine and cosine of degree argument	sincosd(3M)
sine and cosine together	sincos(3M)
sine functions	sin(3M)
sine functions of degree argument	sind(3M)
sinf() - sine function (float)	sin(3M)
single stream, maximum number of STREAMS modules	nstrpush(5)
single System V IPC message queue, maximum number of bytes on a	msgmnb(5)
single-byte and wide character; conversion between	btowc(3C)
single-byte character and rendition, add, to a window and advance the cursor	addch(3X)

Description	Entry Name(Section)
single-byte character and rendition, echo to a window and refresh	echochar(3X)
single-byte character and rendition, input from a window	inch(3X)
single-byte character and rendition, insert into a window	insch(3X)
single-byte character, get from the terminal	getch(3X)
single-byte character, set or get background character or rendition, using	bkgd(3X)
single-byte characters and renditions to a window, add length limited string of	addchnstr(3X)
single-byte characters and renditions to a window, add string of	addchstr(3X)
single-byte characters and renditions, array of, input from a window	inchnstr(3X)
single-byte characters and renditions, draw borders	border(3X)
single-byte characters and renditions, draw borders	box(3X)
single-byte characters and renditions, draw lines from	hline(3X)
single-byte terminal environment query functions	erasechar(3X)
single-user mode	init(1M)
single-user mode, place system in	shutdown(1M)
single-user state	glossary(9)
sinh() - hyperbolic sine function	sinh(3M)
sinhcosh() - both hyperbolic sine and hyperbolic cosine	sinhcosh(3M)
sinhcoshf() - both hyperbolic sine and hyperbolic cosine (float)	sinhcosh(3M)
sinhcoshl() - both hyperbolic sine and hyperbolic cosine (long double)	sinhcosh(3M)
sinhcoshq() - both hyperbolic sine and hyperbolic cosine (quad)	sinhcosh(3M)
sinhcoshw() - both hyperbolic sine and hyperbolic cosine (extended)	sinhcosh(3M)
sinhf() - hyperbolic sine function (float)	sinh(3M)
sinhl() - hyperbolic sine function (long double)	sinh(3M)
sinhq() - hyperbolic sine function (quad)	sinh(3M)
sinhw() - hyperbolic sine function (extended)	sinh(3M)
sinl() - sine function (long double)	sin(3M)
sinq() - sine function (quad)	sin(3M)
sinw() - sine function (extended)	sin(3M)
sioc_io - SCSI pass-through interface	sioc_io(7)
sis - secure internet services with Kerberos authentication and authorization	sis(5)
size (generic), extend a file system	extendfs(1M)
size in 1 KB blocks; swap chunk	swchunk(5)
size (in bytes) for a System V shared memory segment, maximum	shmmax(5)
size (in bytes) of the RSE stack for any user process, maximum	maxrressiz(5)
size (in bytes) of the stack for a user process running under the PA-RISC emulator on an Integrity system; maximum	pa_maxssiz(5)
size (in bytes) of the stack for any user process; maximum	maxssiz(5)
size information of screen, specify source	use_env(3X)
size of file in words, lines, and bytes or characters	wc(1)
size of node name and host name	nodehostnamesize(5)
size of the networking hash tables, determines the	tphashsz(5)
size of the per-process file descriptor table; get the	getdtablesize(2)
size - print section sizes and allocation space of object files	size(1)
size, extend HFS file system	extendfs_hfs(1M)
size, return for elf32 or elf64 files	elf_fsize(3E)
slash	glossary(9)
slave and master pty, STREAMS, unlocking	unlockpt(3C)
slave (pseudo-terminal) driver, STREAMS	tels(7)
slave pty (pseudo-terminal) driver, STREAMS	pts(7)
slave pty, get the name of a	ptsname(3C)
slave pty, STREAMS, granting access	grantpt(3C)
sleep - suspend execution for a time interval	sleep(1)
sleep() - suspend execution for interval	sleep(3C)
sleep; high resolution	nanosleep(2)
slk_attr_off() - soft label functions	slk_attr(3X)
slk_attr_on() - soft label functions	slk_attr(3X)
slk_attr_set() - soft label functions	slk_attr(3X)
slk_attr(3X) - soft label functions	slk_attr(3X)
slk_attron() - soft label functions	slk_attr(3X)
slk_attrset() - soft label functions	slk_attr(3X)
slk_clear() - soft label functions	slk_attr(3X)

Index

All Volumes

Description	Entry Name(Section)
slk_color() - soft label functions	slk_attroff(3X)
slk_init() - soft label functions	slk_attroff(3X)
slk_label() - soft label functions	slk_attroff(3X)
slk_noutrefresh() - soft label functions	slk_attroff(3X)
slk_refresh() - soft label functions	slk_attroff(3X)
slk_restore() - soft label functions	slk_attroff(3X)
slk_set() - soft label functions	slk_attroff(3X)
slk_touch() - soft label functions	slk_attroff(3X)
slk_wset() - soft label functions	slk_attroff(3X)
slot in the utmpx() file of the current user, find	ttyslot(3C)
SLP agents; configuration file for	slp.conf(4)
SLP error codes; Service Location Protocol	SLPError(3N)
SLP Protocol Daemon	sldap(1M)
SLP (Service Location Protocol) library routines	libslp(3N)
SLP Service Type Syntax	slp_syntax(7)
slp - set printing options for a non-serial printer	slp(1)
SLP static registration file	slp.reg(4)
slp.conf - configuration file for SLP agents	slp.conf(4)
slp.reg - SLP static registration file	slp.reg(4)
slp_syntax - SLP Service Type Syntax	slp_syntax(7)
slpClose() - SLP (Service Location Protocol) library routines	libslp(3N)
sldap - Service Location Protocol Daemon	sldap(1M)
sldap - SLP Protocol Daemon	sldap(1M)
sldapc - send signals to SLP daemon	sldapc(1M)
SLPDelAttrs() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPDereg() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPError - Service Location Protocol (SLP) error codes	SLPError(3N)
SLPEscape() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPFindAttrs() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPFindScopes() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPFindSrvs() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPFindSrvTypes() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPFree() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPGetProperty() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPGetRefreshInterval() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPOpen() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPParseSrvURL() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPReg() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPSetProperty() - SLP (Service Location Protocol) library routines	libslp(3N)
SLPUnescape() - SLP (Service Location Protocol) library routines	libslp(3N)
slweb - start the HP-UX hardware event viewer tool (a Web interface)	slweb(1M)
sm - statd directory and file structures	sm(4)
sm.bak - statd directory and file structures	sm(4)
Small Computer System Interface device drivers	scsi(7)
Smart Card Login	login(1)
smfi_addheader() - adds a header to the current message	smfi_addheader(3N)
smfi_addrcpt() - adds a recipient for the current message	smfi_addrcpt(3N)
smfi_chgheader() - changes or deletes a message header	smfi_chgheader(3N)
smfi_delrcpt() - removes a recipient from envelope of current sendmail message	smfi_delrcpt(3N)
smfi_getpriv() - gets connection-specific data pointer for the sendmail connection	smfi_getpriv(3N)
smfi_getsymval() - gets the value of a sendmail macro	smfi_getsymval(3N)
smfi_inshheader() - prepends a header to the current sendmail message	smfi_inshheader(3N)
smfi_main() - passes control to the libmilter event loop	smfi_main(3N)
smfi_opensocket() - attempts to create the interface socket that MTAs use to connect to the filter	smfi_opensocket(3N)
smfi_progress() - notifies the MTA that a sendmail operation is still in progress	smfi_progress(3N)
smfi_quarantine() - quarantines the sendmail message using the given reason	smfi_quarantine(3N)
smfi_register() - registers a set of filter callbacks for sendmail	smfi_register(3N)
smfi_replacbody() - replaces the data in the sendmail message body	smfi_replacbody(3N)
smfi_setbacklog() - sets the listen backlog value of the filter for sendmail	smfi_setbacklog(3N)

Description	Entry Name(Section)
smfi_setconn() - sets the socket for filter to communicate with sendmail	smfi_setconn(3N)
smfi_setdbg() - sets the debugging level for the Milter library for sendmail	smfi_setdbg(3N)
smfi_setmlreply() - sets default SMTP error reply code to a multi-line response	smfi_setmlreply(3N)
smfi_setpriv() - sets the private data pointer for the sendmail connection	smfi_setpriv(3N)
smfi_setreply() - sets the default SMTP error reply code	smfi_setreply(3N)
smfi_settimeout() - sets the sendmail connection timeout value of a filter	smfi_settimeout(3N)
smfi_stop() - starts an orderly shutdown of the Milter for sendmail	smfi_stop(3N)
smh - HP System Management Homepage (HP SMH)	smh(1M)
SMH Web interface; start the HP-UX Peripheral Device tool, part of the	pdweb(1M)
SMH; HP System Management Homepage. HP	smh(1M)
SMH; launch the Disks and File Systems tool of HP System Management Homepage	fsweb(1M)
smhstartconfig - configures the startup mode of the HPSMH server and of the Tomcat instance used by HPSMH	smhstartconfig(1M)
smonitor() - prepare execution profile	smonitor(3C)
smrsh - restricted shell for sendmail	smrsh(1M)
SMTP error reply code to a multi-line response; sets default	smfi_setmlreply(3N)
SMTP error reply code; sets the default	smfi_setreply(3N)
snapshot of the UUCP system	uusnap(1M)
SNMP agent, configuration file for the	snmpd.conf(4)
SNMP requests, daemon that responds to	snmpd(1M)
SNMP, Native Agent Adapter	naaagt(1M)
snmpd - daemon that responds to SNMP requests	snmpd(1M)
snmpd.conf - configuration file for the SNMP agent	snmpd.conf(4)
snprintf() - print formatted output to a string	printf(3S)
socketmark() - determine whether a socket is at the out-of-band mark	socketmark(3N)
socket address; get	getsockname(2)
socket at out-of-band mark; determine whether	socketmark(3N)
socket() - create an endpoint for communication	socket(2)
socket endpoints.; get the compartment IDs of	cmpt_get_peer_cid(3)
socket for filter to communicate with sendmail; sets the	smfi_setconn(3N)
socket - Interprocess communications	socket(7)
socket that MTAs use to connect to the filter; attempts to create the interface	smfi_opensocket(3N)
socket, get detailed information for a	pstat(2)
socket, return a reserved port	rcmd(3N)
socket; accept a connection on a	accept(2)
socket; bind an address to a	bind(2)
socket; initiate a connection on a	connect(2)
socket; listen for connections on a	listen(2)
socket; receive a message from a	recv(2)
socket; send a message from	send(2)
socket; send a message from a	send(2)
socket; send the contents of a file through a	sendfile(2)
socket; send the contents of a Large File through a	sendfile64(2)
socket; shut down a	shutdown(2)
socketpair() - create a pair of connected sockets	socketpair(2)
sockets; create a pair of connected	socketpair(2)
sockets; get and set options on	getsockopt(2)
soelim - eliminate .so's from nroff input	soelim(1)
soft label functions	slk_attroff(3X)
soft (symbolic) file link	symlink(4)
soft-copy terminals, peruse file on	pg(1)
softkey file format for keysh	softkeys(4)
softkey shell, context-sensitive	keysh(1)
softkeys - keysh softkey file format	softkeys(4)
softpower - determine if softpower hardware is installed	softpower(1M)
softpower hardware	softpower(1M)
Software Distributor (SD) uses, their attributes and storage formats; all objects that	sd(4)
software for serial and network connections	kermit(1)
software products, display information about	swlist(1M)
software products, install, configure, and copy	swinstall(1M)

Index

All Volumes

Description	Entry Name(Section)
software products, modify in target root or depot	swmodify(1M)
software products, package into target depot or tape	swpackage(1M)
software products, remove and unconfigure	swremove(1M)
software products; verify	swverify(1M)
software signal, raise a	ssignal(3C)
software, outbound connection daemon debug utility used by DDFA	ocdebug(1M)
solidus	glossary(9)
sort a directory pointer array	scandir(3C)
sort and embellish uusnap output	uusnaps(1M)
sort events	evmsort(1)
sort or merge files	sort(1)
sort - sort or merge files	sort(1)
sort, quicker	qsort(3C)
sort, topological	tsort(1)
sorted files, reject/select lines common to two	comm(1)
sorted tables, binary search routine for	bsearch(3C)
source code	glossary(9)
Source Code Control System	see <i>SCCS</i>
Source Code Control System (SCCS)	glossary(9)
source - define source for command input	csh(1)
source file; preprocess a message	mkcatdefs(1)
source information from ELF files	uwxfind_source_info(3X)
source of screen size information, specify	use_env(3X)
source port verification check; enable/disable the NFS server's	nfs_portmon(5)
source program files for given name, find location of	whereis(1)
space allocation, change data segment	brk(2)
space for LVM logical volume, increase	lvextend(1M)
space for signal stack, define, delete, or get amount of	sigspace(2)
space information; system paging	swapinfo(1M)
space, stack and data, allocate then lock process into memory	datalock(3C)
spaces, convert to tabs and vice versa	expand(1)
spawn getty to remote terminal (call terminal)	ct(1)
spawn new process (use fork() instead)	vfork(2)
spawn processes	init(1M)
special and FIFO files, create	mknod(1M)
special attributes for group, get	getprivgrp(1)
special (device) file; make	mksf(1M)
special (device) file; remove	rmsf(1M)
special (device) files; install	insf(1M)
special file	glossary(9)
special file, control character device	ioctl(2)
special file; /dev/zero	zero(7)
special file; make	mknod(2)
special files: line printer device files	lp(7)
special files: system console interface	console(7)
special files; controlling terminal interface	tty(7)
special files; general terminal interface	termio(7)
special files; introduction to device	intro(7)
special files; list all device drivers available in the system	lsdev(1M)
special files; make FIFO (named pipe)	mkfifo(1)
special files; pseudo-terminal driver	pty(7)
special functions of HP 2640- and HP 2621-series terminals, handle	hp(1)
special (I/O device) file; list	lssf(1M)
special initialization script, /sbin/set_parms	hostname(1)
special NTP query program	xntpd(1M)
special privileges for groups; set	setprivgrp(1M)
special system processes	glossary(9)
specific time interval to the current absolute system time, add a	get_expiration_time(3T)
specification file format; PPP packet filter	ppp.Filter(4)
specification file (PSF) format; product	swpackage(4)
specification, format, in text files	fspec(4)

Description	Entry Name(Section)
specified name-type, list the mechanisms that support	gss_inquire_mechs_for_name(3)
specified remote machines, write to	rwall(3N)
specify minimum value for process IDs (PIDs)	process_id_min(5)
specify source of screen size information	use_env(3X)
specify what to do upon receipt of a signal	signal(2)
speed, and line discipline; set terminal type, modes,	getty(1M)
speed, datacomm line, and terminal settings used by getty	gettydefs(4)
spell - find spelling errors	spell(1)
spellin - convert 9-digit hash codes to compressed spelling reference list	spell(1)
spelling errors, find	spell(1)
spin attribute; get and set mutex	pthread_mutexattr_getspin_np(3T)
spinlock pool size, System V IPC hashed	sysv_hash_locks(5)
spinlocks protecting the channel queue hash tables, size of hashed pool of	chanq_hash_locks(5)
split a file into multiple <i>n</i> -line pieces	split(1)
split buffer into fields	bufsplit(3G)
split file into multiple files	csplit(1)
split mirrored LVM logical volume into two logical volumes	lvsplit(1M)
split - split a file into multiple <i>n</i> -line pieces	split(1)
spool directory clean-up, uucp	uuclean(1M)
spool directory clean-up, uucp	uucleanup(1M)
spooled uucp transactions grouped by transaction; list	uuls(1M)
spooler performance analysis information; display LP	lpana(1M)
spooler requests on a remote system; print status of LP	rlpstat(1M)
spooler, LP	<i>see LP spooler</i>
spooling	<i>see printer</i>
spooling system; configure the LP	lpadmin(1M)
spray : scatter data to check the network	spray(3N)
spray packets	spray(1M)
spray server	sprayd(1M)
spray - spray packets	spray(1M)
sprayd - spray server	sprayd(1M)
sprintf() - print formatted output to a string	printf(3S)
sprofil() - execution time profile for disjointed text spaces	sprofil(2)
sqrt() - square root function	sqrt(3M)
sqrtf() - square root function (float)	sqrt(3M)
sqrtl() - square root function (long double)	sqrt(3M)
sqrtq() - square root function (quad)	sqrt(3M)
sqrtw() - square root function (extended)	sqrt(3M)
square root functions	sqrt(3M)
srand() - reset random-number generator to random starting point	rand(3C)
srand48() , seed48() , lcong48() - initialize pseudo-random number generator	drand48(3C)
srandom() , initstate() , setstate() , random() - generate a pseudorandom number	random(3M)
SS/80	glossary(9)
sscanf() - formatted read from character string	scanf(3S)
ssignal() - raise a software signal and perform an action	ssignal(3C)
ssp - remove multiple line-feeds from output	ssp(1)
st - shared tape administration	st(1M)
st_ats_enabled - determines whether to reserve a tape device on open	st_ats_enabled(5)
stable storage area, get information from the system's	pstat(2)
stable storage; display and modify boot variables	setboot(1M)
stable storage; display and modify boot variables in	setboot(1M)
stack and data space, allocate then lock process into memory	datalock(3C)
stack context, set/get signal alternate stack context	sigaltstack(2)
stack environment, save/restore for non-local goto	setjmp(3C)
stack for a user process running under the PA-RISC emulator on an Integrity system; maximum size (in bytes) of the	pa_maxssiz(5)
stack space for signals, define, delete, or get amount of	sigspace(2)
stack templates, defines and manages file system	fstadm(2)
stack trace for each LWP in each process and core file; print a	pstack(1)
Stack unwind library	uwX(3X)
stack using the unwind library, produce a trace back of the procedure call	U_STACK_TRACE(3X)

Index

All Volumes

Description	Entry Name(Section)
stack, lock in memory	plock(2)
stacks are executable by default, controls whether program	executable_stack(5)
stacksize., change the default	pthread_default_rsestacksize_np(3T)
stacksize; change default	pthread_default_stacksize_np(3T)
stale logical volume mirrors in LVM volume groups, synchronize	vgsync(1M)
stale mirrors in LVM logical volumes, synchronize	lvsync(1M)
standard and restricted POSIX.2-conformant command shells	sh-posix(1)
standard buffered input/output stream file package	stdio(3S)
standard error	glossary(9)
standard error and console, displays formatted message on	fntmsg(3C)
standard format; display message in	pfmt(3C)
standard input	glossary(9)
standard input to system log, send	logger(1)
standard input; read a line from	read(1)
standard output	glossary(9)
standard output to file; pipe fitting to copy	tee(1)
standard POSIX.2-conformant command shell	sh-posix(1)
standard structures and symbolic constants	unistd(5)
standards behavior on HP-UX; UNIX	standards(5)
standards - UNIX standards behavior on HP-UX	standards(5)
standend() - set and clear window attributes	standend(3X)
standout() - set and clear window attributes	standend(3X)
stape, magnetic tape interface for tape2 and	mt(7)
start address of current function	uwx_get_funcstart(3X)
start auditing the current process as owned by a given user	setauduser(3)
start of file, list first few lines at	head(1)
start or halt the auditing system and set or get audit files	audctl(2)
start program in particular memory window; change window ID of running program or ...	setmemwindow(1M)
start terminal session	login(1)
start terminal session; sign on;	login(1)
start the audio server	aserver(1M)
start the Event Manager	evmstart(1M)
start the HP-UX hardware event viewer tool (a Web interface)	slweb(1M)
start the HP-UX Peripheral Device tool, part of the SMH Web interface	pdweb(1M)
start the LP request scheduler	lpsched(1M)
start/halt the auditing system	audsys(1M)
start_color() - color manipulation functions	can_change_color(3X)
starting, EVM channel manager	evmstart(1M)
starting, EVM daemon	evmstart(1M)
starting, EVM logger	evmstart(1M)
starts an orderly shutdown of the Milter for sendmail	smfi_stop(3N)
starts or stops the HP System Management Homepage server	hpsmh(1M)
starts the HP-UX kernel configuration tool (a Web interface)	kcweb(1M)
starts the HP-UX user and group account configuration tool	ugweb(1M)
startup mode of the HPSMH server and of the Tomcat instance used by HPSMH; configures the	smhstartconfig(1M)
startup routines for Integrity systems; execution	crt0_ia(3)
startup routines for PA-RISC systems; execution	crt0_pa(3)
startup routines; execution	crt0(3)
startup - start accounting process at system startup	acctsh(1M)
stat - data returned by the stat() function	stat(5)
stat() function; data returned by the	stat(5)
stat() - get file status	stat(2)
stat.h - data returned by the stat() function	stat(5)
stat64() - non-POSIX standard API interfaces to support large files	creat64(2)
statd directory and file structures	sm(4)
statd - network status monitor	statd(1M)
state information, get Terminal Session Manager	tsm.info(1)
state of HP-UX 11.x system or depot; check security-bulletin compliance	security_patch_check(1M)
state - statd directory and file structures	sm(4)
state to the file system for debugging purposes., a feature that saves operating system	livedump(5)

Description	Entry Name(Section)
state with its state on disk, synchronize a file's in-core	fsync(2)
state, PAM routines to maintain module specific state	pam_set_data(3)
statfs() , fstatfs() - get file system statistics	statfs(2)
statfsdev() - get file system statistics	statfsdev(3C)
static information about the file systems	fstab(4)
station address string conversion routines, network	net_aton(3C)
statistics for file system, list	ff(1M)
statistics for HFS file system, list file names and	ff_hfs(1M)
statistics server, kernel	rstatd(1M)
statistics, get file system	statfs(2)
statistics, get mounted file system	ustat(2)
statistics, Network File System	nfsstat(1M)
statistics, print mail traffic	mailstats(1)
statistics, report virtual memory	vmstat(1)
statistics; get file system	statfsdev(3C)
statistics; report I/O	iostat(1)
status code, GSSAPI textual representation	gss_display_status(3)
status code; format text version of EVM	EvmStatusTextGet(3)
status information and attributes associated with a message queue, get	mq_getattr(2)
status information of the LP subsystem; report	lpstat(1)
status inquiries, stream	ferror(3S)
status inquiries, stream	ferror(3S)
status inquiry and job control, uucp	uustat(1)
status monitor; network	statd(1M)
status of HFS file systems; determine the shutdown	fsclan(1M)
status of interprocess communication facilities, report	ipcs(1)
status of local machines, show	ruptime(1)
status of local user accounts; check	userstat(1M)
status of LP spooler requests on a remote system; print	rlpstat(1M)
status of power for cells and I/O chassis; turn on/off or display current	frupower(1M)
status server; system	rwhod(1M)
status, asynchronous I/O error	aio_error(2)
status, current, of the UUCP system	uusnap(1M)
status, display LAN device configuration and	lanscan(1M)
status, exit, do nothing and return zero or non-zero	true(1)
status, get file	stat(2)
status, get file status	fstat(2)
status, get symbolic link	lstat(2)
status, host, of local machines (RPC version), show	rup(1)
status, line update status functions	redrawwin(3X)
status; report process	ps(1)
statvfs() - get mounted file system information	statvfs(2)
statvfs64() - non-POSIX standard API interfaces to support large files	creat64(2)
statvfsdev() - get file system information	statvfsdev(3C)
statvfsdev64() - file system API to support large files	fgetpos64(3S)
stdarg argument list; convert formatted wide-character input	vwscanf(3S)
stdarg.h - macros for handling variable argument list	stdarg(5)
stderr	glossary(9)
stdin	glossary(9)
stdio() - standard buffered input/output stream file package	stdio(3S)
stdout	glossary(9)
stdscr() - default window	stdscr(3X)
stdsyms - description of named defines and other specifications for namespace from HP-UX header files	stdsyms(5)
step one frame	uwx_step(3X)
step over one inline call	uwx_step_inline(3X)
step() - regular expression string comparison routine	regexp(3X)
sticky bit	glossary(9)
stime() - set time and date	stime(2)
stop or terminate; wait for child process to	wait(2)
stop system operation	shutdown(1M)

Index

All Volumes

Description	Entry Name(Section)
stop the Event Manager	evmstop(1M)
stop the LP request scheduler	lpsched(1M)
stop then reboot the system	reboot(1M)
stopping, EVM channel manager	evmstop(1M)
stopping, EVM daemon	evmstop(1M)
stopping, EVM logger	evmstop(1M)
stops Live Dump, initiates, configures, and	livedump(1M)
stops the HP System Management Homepage server; starts or	hpsmh(1M)
storage area, get information from the system's stable	pstat(2)
storage associated with a screen	delscreen(3X)
storage formats; all objects that Software Distributor (SD) uses, their attributes and	sd(4)
Storage Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass	fcmutil(1M)
storage, preallocate disk	prealloc(1)
storage, preallocate fast disk	prealloc(2)
storage; display and modify boot variables in stable	setboot(1M)
store() - store data under a key (old single-data-base version)	dbm(3X)
stored events; retrieve	evmget(1)
strace - write STREAMS event trace messages to standard output	strace(1M)
strcasemp() , strncasemp() - compare two strings	string(3C)
strcat() , strncat() - append string 2 to string 1	string(3C)
strchg - change or query stream configuration	strchg(1M)
strchr() , strrchr() - get pointer to character in string	string(3C)
strclean - remove outdated STREAMS error log files	strclean(1M)
strcmp() , strncmp() - compare two strings	string(3C)
strcoll() - process string of text tokens	string(3C)
strconf - query stream configuration	strchg(1M)
strcpy() , strncpy() - copy string 2 to string 1	string(3C)
strcspn() , strspn() - find length of matching substrings	string(3C)
STRCTLSZ - maximum size of streams message control in bytes	strctlsz(5)
strdup() - determine length of a string	string(3C)
stream	glossary(9)
stream configuration, change or query	strchg(1M)
stream creation, library routines for external data representation	xdr_create(3N)
stream file or character string; read from with formatted input conversion	scanf(3S)
stream file package, standard buffered input/output	stdio(3S)
stream file, get or reposition pointer for I/O operations on a	fseek(3S)
stream file; assign buffering to a	setbuf(3S)
stream file; buffered binary input/output to	fread(3S)
stream file; get a wide-character string from	fgetws(3C)
stream file; get character or word from a	getc(3S)
stream file; get wide character from	getwc(3C)
stream orientation; set	fwide(3C)
stream pointer, map to file descriptor	fileno(3S)
stream status inquiries	ferror(3S)
stream status inquiries	ferror(3S)
stream() - STREAMS enhancements to standard system calls	stream(2)
stream text editor	sed(1)
stream to a remote command; return	rexec(3N)
stream up to next delimiter; read	bgets(3G)
stream, close or flush a	fclose(3S)
stream, get detailed information for a	pstat(2)
stream, orientation of	orientation(5)
stream, push character back into input	ungetc(3S)
stream, push wide character back into input	ungetwc(3C)
stream, put wide character on a	putwc(3C)
stream, put word or character on a	putc(3S)
stream, return to a remote command	rcmd(3N)
stream; get a string from	gets(3S)
stream; save or restore file position indicator for a	fgetpos(3S)
stream; send a message on a	putmsg(2)

Description	Entry Name(Section)
streamio - STREAMS ioctl commands	streamio(7)
streampipes - force all pipes to be STREAMS-based	streampipes(5)
STREAMS Administrative Driver	sad(7)
streams allowed on the system; maximum number of cloned DLPI	dlpi_max_clones(5)
STREAMS bufcall, maximum number of outstanding	nstrent(5)
STREAMS enhancements to system calls	stream(2)
STREAMS error log files, remove outdated files	strclean(1M)
STREAMS event trace messages to standard output, write	strace(1M)
STREAMS file descriptor	fattach(3C)
STREAMS file; receive next message	getmsg(2)
STREAMS ioctl commands	streamio(7)
STREAMS log driver	strlog(7)
STREAMS log driver; receive error messages	strerr(1M)
STREAMS master pty (pseudo-terminal) driver	ptm(7)
streams message control, maximum size in bytes	strctlsz(5)
streams message data, maximum size in bytes	strmsgsz(5)
STREAMS module for converting ioctl() calls into Transport Interface messages	timod(7)
STREAMS module for reads and writes by Transport Interface	tirdwr(7)
STREAMS module, terminal line discipline	ldterm(7)
STREAMS modules in a single stream, maximum number	nstrpush(5)
STREAMS modules; manage system database of automatically pushed STREAMS modules	autopush(1M)
STREAMS pass through device driver to open a major and minor device pair on a STREAMS driver	clone(7)
STREAMS pty master/slave pair, unlocking	unlockpt(3C)
STREAMS pty (pseudo-terminal) Emulation module	ptem(7)
STREAMS pty, get the name of a slave	ptsname(3C)
STREAMS pty, Packet Mode module	pckt(7)
STREAMS scheduler daemons to run, number of	nstrsched(5)
STREAMS slave pty driver	pts(7)
STREAMS slave pty, granting access to	grantpt(3C)
STREAMS Telnet slave driver	tels(7)
STREAMS terminal line discipline module	ldterm(7)
streams to parallel remote commands; return	prcmd(3N)
STREAMS verification tool	strvf(1M)
streams within a multi-thread application, explicit locking of	flockfile(3S)
STREAMS, change or query stream configuration	strchg(1M)
STREAMS-base, force all pipes to be	streampipes(5)
STREAMS-based file descriptor	fdetach(1M)
STREAMS-based file descriptor	fdetach(3C)
STREAMS-based pipe	isastream(3C)
STREAMS-based pseudo terminals (pts), maximum number of	nstrpty(5)
STREAMS: attach a STREAMS file descriptor	fattach(3C)
STREAMS: detach a name from a STREAMS-based file descriptor	fdetach(3C)
STREAMS: detach a STREAMS-based file descriptor	fdetach(1M)
STREAMS: determine if file descriptor refers to STREAMS device or STREAMS-based pipe	isastream(3C)
strerr - receive error messages from the STREAMS log driver	strerr(1M)
strerror() - write system error messages	perror(3C)
strerror_r() - write system error messages	perror(3C)
strfmon() - convert monetary value to string	strfmon(3C)
strftime() - convert date and time to string	strftime(3C)
string conversion routines, network station address	net_aton(3C)
string data order, convert	strord(3C)
string form to access control list (ACL) structure, HFS file system only; convert	strtoacl(3C)
string form, convert access control list (ACL) structure to	acltostr(3C)
string from a stream file; get a wide-character	fgetws(3C)
string from a stream; get	gets(3S)
string of single-byte characters and renditions to a window, add	addchstr(3X)
string of wide characters, input from a window	innwstr(3X)
string of wide-character, insert into a window	ins_nwstr(3X)
string operations; character	string(3C)
string operations; wide character	wcstring(3C)
string or expression, search a file for a	grep(1)

Index

All Volumes

Description	Entry Name(Section)
string pointer for ELF files, make	elf_strptr(3E))
string (restartable); convert a character string to a wide-character	mbsrtowcs(3C)
string to a wide-character string (restartable); convert a character	mbsrtowcs(3C)
string to floating-point number; convert	strtod(3C)
string to integer, convert	strtoimax(3C)
string to long integer; convert	strtol(3C)
string to NaN conversion functions	nan(3M)
string, convert between long integer and base-64 ASCII	a64l(3C)
string, convert long double floating-point number to	ldcv(3C)
string, convert long integer to	ltostr(3C)
string, get a multi-byte character length limited string from the terminal	getnstr(3X)
string, get a multi-byte character string from the terminal	getstr(3X)
string, parse suboptions from a	getsubopt(3C)
string-valued configuration values; get	confstr(3C)
string; convert date and time to	ctime(3C)
string; convert date and time to	strftime(3C)
string; convert floating-point number to	ecvt(3C)
string; convert monetary value to	strfmon(3C)
strings and characters conversions; multibyte	multibyte(3C)
strings - find the printable strings in an object, or other binary, file	strings(1)
strings, find for inclusion in message catalogs	findstr(1)
strings; generate hashing encryption on large	bigcrypt(3C)
strip nroff/troff, tbl, and neqn constructs from a file	deroff(1)
strip - strip symbol and line number information from an object file	strip(1)
strip symbol and line number information from an object file	strip(1)
stripe LVM logical volume	lvextend(1M)
strlen() - determine length of a string	string(3C)
strlog - STREAMS log driver	strlog(7)
STRMSGSZ - maximum size of streams message data in bytes	strmsgsz(5)
strong random number generator	random(7)
strord() - convert string data order	strord(3C)
strprbk() - find occurrence of character from string 2 in string 1	string(3C)
strptime() - date and time conversion	strptime(3C)
strrstr() - process string of text tokens	string(3C)
strspn() , strcspn() - find length of matching substrings	string(3C)
strstr() - process string of text tokens	string(3C)
strtoacl() - convert string form to access control list (ACL) structure, HFS file system only	strtoacl(3C)
strtoacl_r() - convert string form to access control list (ACL) structure, HFS file system only	strtoacl(3C)
strtoaclpatt() - convert string form to access control list (ACL) structure, HFS file system only	strtoacl(3C)
strtoaclpatt_r() - convert string form to access control list (ACL) structure, HFS file system only	strtoacl(3C)
strtod() - convert string to floating-point number (double)	strtod(3C)
strtof() - convert string to floating-point number (float)	strtod(3C)
strtoimax() - convert string to integer	strtoimax(3C)
strtok() - process string of text tokens	string(3C)
strtok_r() - process string of text tokens	string(3C)
strtol() - convert string to long integer	strtol(3C)
strtold() - convert string to floating-point number (long double)	strtod(3C)
strtoll() - convert string to long integer	strtol(3C)
strtoq() - convert string to floating-point number (quad)	strtod(3C)
strtoul() - convert string to long integer	strtol(3C)
strtoull() - convert string to long integer	strtol(3C)
strtoimax() - convert string to integer	strtoimax(3C)
strtow() - convert string to floating-point number (extended)	strtod(3C)
structure formats	nlist(4)
structure formats for Integrity systems	nlist_ia(4)
structure formats for PA-RISC systems	nlist_pa(4)
structure of an EVM event	EvmEvent(5)

Description	Entry Name(Section)
structure, HFS file system only; convert string form to access control list (ACL)	strtoacl(3C)
structure; allocate and deallocate unwind library data	_UNW_createContextForSelf(3X)
structure; allocate transport function library	t_alloc(3)
structure; manipulate values in unwind library data	_UNW_currentContext(3X)
structure; query values in unwind library data	_UNW_getGR(3X)
structures needed when using the fadvise() function	fadvise(5)
structures, number of System V IPC system-wide semaphore undo	semnu(5)
structures, statd directory and file	sm(4)
strvf - STREAMS verification tool	strvf(1M)
strxfrm() - process string of text tokens	string(3C)
stty() - control terminal device (Bell Version 6 compatibility)	stty(2)
stty - set the options for a terminal port	stty(1)
stty - terminal interface for Version 6/PWB compatibility	sttyv6(7)
sttyv6 - terminal interface for Version 6/PWB compatibility	sttyv6(7)
stubs are provided in the C library, list of pthread calls for which the	pthread_stubs(5)
su - switch user	su(1)
subdirectory	glossary(9)
subject and sender; summarize mail folders by	mailfrom(1)
suboptions, parse from a string	getsubopt(3C)
subordinate directory	glossary(9)
subpad() - enhanced pad management function	subpad(3X)
subroutine call graph execution profile data, display	gprof(1)
subroutines and libraries; introduction to	intro(3C)
subroutines, database (new multiple database version)	ndbm(3X)
subroutines, database (old version - see also ndbm(3X))	dbm(3X)
subscription for event notification; establish a	EvmConnSubscribe(3)
subscriptions; enables you to view, create, modify, and delete event	evweb_subscribe(1)
Subset 1980	glossary(9)
subset of functions from libc	libcres(5)
substitute selected characters	tr(1)
subsystem (OBSOLETE); enable and disable use of device's write cache in the SCSI	default_disk_ir(5)
subsystem, get information about the System V IPC	pstat(2)
subsystem, get information about the virtual memory	pstat(2)
subsystem; interface for interacting with kernel I/O	libIO(3X)
subsystem; percentage of physical memory that can be used by audit	audit_memory_usage(5)
subsystem; report status information of the LP	lpstat(1)
subtrees; copy files and directory	cp(1)
subwin() - window creation functions	newwin(3X)
successful login to remote system; test for	uucp(1)
suffix conventions; file name	suffix(5)
suffix - file name suffix conventions	suffix(5)
sum - print checksum and block count of a file	sum(1)
summarize an SCCS file; print and	prs(1)
summarize disk usage	du(1)
summarize file system ownership	quot(1M)
summarize file system quotas	repquota(1M)
summarize information from compiler footprint records	footprints(1)
summarize mail folders by subject and sender	mailfrom(1)
summarize, add, modify, delete, or copy file access control lists (ACLs)	chacl(1)
superblock	glossary(9)
superior directory	glossary(9)
superuser	passwd(1)
superuser	glossary(9)
superuser, change login name to	su(1)
supplementary group ID	glossary(9)
support for local network packet routing; system	routing(7)
support for POSIX.1b realtime applications, number of priority values to	rtsched_numpri(5)
supported terminal video attributes, get	termattrs(3X)
suppress echo while reading password from terminal	getpass(3C)
suspend calling process until signal received	sigset(3C)
suspend Curses session	endwin(3X)

Index

All Volumes

Description	Entry Name(Section)
suspend execution for a time interval	sleep(1)
suspend execution for an interval	usleep(2)
suspend execution for interval	sleep(3C)
suspend execution of a thread	pthread_resume_np(3T)
suspend for asynchronous I/O completion	aio_suspend(2)
suspend foreground until background processes are finished	wait(1)
suspend or resume auditing on the current process	audswitch(2)
suspend process until signal	pause(2)
suspend the calling process	napms(3X)
svc_auth_reg() - library routines for registering servers	rpc_svc_reg(3N)
svc_control() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_destroy() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_dg_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_dg_enablecache() - library routines for RPC servers	rpc_svc_calls(3N)
svc_done() - library routines for RPC servers	rpc_svc_calls(3N)
svc_exit() - library routines for RPC servers	rpc_svc_calls(3N)
svc_fd_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_fd_negotiate_ucred() - library routines for RPC servers	rpc_svc_calls(3N)
svc_fds() - obsolete library routines for RPC	rpc_soc(3N)
svc_fdset() - library routines for RPC servers	rpc_svc_calls(3N)
svc_freeargs() - library routines for RPC servers	rpc_svc_calls(3N)
svc_getargs() - library routines for RPC servers	rpc_svc_calls(3N)
svc_getcaller() - obsolete library routines for RPC	rpc_soc(3N)
svc_getreq() - obsolete library routines for RPC	rpc_soc(3N)
svc_getreq_common() - library routines for RPC servers	rpc_svc_calls(3N)
svc_getreq_poll() - library routines for RPC servers	rpc_svc_calls(3N)
svc_getreqset() - library routines for RPC servers	rpc_svc_calls(3N)
svc_getrpcaller() - library routines for RPC servers	rpc_svc_calls(3N)
svc_pollset() - library routines for RPC servers	rpc_svc_calls(3N)
svc_raw_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_reg() - library routines for registering servers	rpc_svc_reg(3N)
svc_register() - obsolete library routines for RPC	rpc_soc(3N)
svc_run() - library routines for RPC servers	rpc_svc_calls(3N)
svc_sendreply() - library routines for RPC servers	rpc_svc_calls(3N)
svc_tli_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_tp_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_unreg() - library routines for registering servers	rpc_svc_reg(3N)
svc_unregister() - obsolete library routines for RPC	rpc_soc(3N)
svc_vc_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svcerr_auth() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_decode() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_noproc() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_noprog() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_progvers() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_systemerr() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_weakauth() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcfid_create() - obsolete library routines for RPC	rpc_soc(3N)
svcrow_create() - obsolete library routines for RPC	rpc_soc(3N)
svctcp_create() - obsolete library routines for RPC	rpc_soc(3N)
svcudp_bufcreate() - obsolete library routines for RPC	rpc_soc(3N)
svcudp_create() - obsolete library routines for RPC	rpc_soc(3N)
swab() - swap bytes	swab(3C)
swacl - view or modify Access Control Lists	swacl(1M)
swagent - perform software management tasks as the agent of an SD command	swagentd(1M)
swagentd - serve local or remote software management tasks	swagentd(1M)
swap area, get information for a	pstat(2)
swap bytes	swab(3C)
swap space for interleaved paging and swapping; add	swapon(2)
swap space information, system	swapinfo(1M)
swap space; manage and configure system	swapctl(2)

Description	Entry Name(Section)
swap the left-to-right text character sequence in each line of a file	rev(1)
swap volume, remove LVM logical volume link	lvrmboot(1M)
swap, determines when swapmap structures are allocated for filesystem	allocate_fs_swapmap(5)
swap, or dump volume; prepare LVM logical volume to be root, boot, primary	lvlnboot(1M)
swap; maximum number of devices that can be enabled for	nswapdev(5)
swap; maximum number of file systems that can be enabled for	nswapfs(5)
swapcontext() - manipulate user contexts; DEPRECATED	makecontext(2)
swapctl() - manage and configure system swap space	swapctl(2)
swapinfo - system paging space information	swapinfo(1M)
swapmap structures are allocated for filesystem swap, determines when	allocate_fs_swapmap(5)
swapmem_on - OBSOLETE kernel tunable parameter	swapmem_on(5)
swapon() - add swap space for interleaved paging and swapping	swapon(2)
swapon - enable device or file system for paging	swapon(1M)
swapping across NFS, enable	remote_nfs_swap(5)
swapping; add swap space for interleaved	swapon(2)
swapping; enable device or file system	swapon(1M)
swask - ask for user response for SD-UX	swask(1M)
swchunk - swap chunk size in 1 KB blocks	swchunk(5)
swconfig - configure, unconfigure, reconfigure installed software	swconfig(1M)
swcopy - copy software products for subsequent installation or distribution	swinstall(1M)
swgettools - utility for retrieving the SD product from new SD media	swgettools(1M)
swinstall - install and configure software products	swinstall(1M)
switch between screens	set_term(3X)
switch - define switch statement	csh(1)
switch to a new group	newgrp(1)
switched virtual circuit, clear X.25	clrsvc(1M)
swjob - display job information and remove jobs	swjob(1M)
swlist - display information about software products	swlist(1M)
swmodify - modify software products in a target root or depot	swmodify(1)
swpackage - package software products into a target depot or tape	swpackage(1M)
swprintf() - print formatted wide-character output	fwprintf(3C)
swreg - register or unregister depots and roots	swreg(1M)
swremove - unconfigure and remove software products	swremove(1M)
swscanf() - convert formatted wide-character input	fwscanf(3C)
swverify - verify software products	swverify(1M)
symbol and line number information, strip from an object file	strip(1)
symbol name; perform I/O on kernel memory, based on	kmem(7)
symbol table for object code file, print	nm(1)
symbol table, regenerate archive	ranlib(1)
symbol table; retrieve archive	elf_getarsym(3E)
symbolic information for an address	dladdr(3C)
symbolic information from ELF files	uwx_find_symbol(3X)
symbolic link	glossary(9)
symbolic link status, get	lstat(2)
symbolic link to a file, make a	symlink(2)
symbolic link, read value of	readlink(2)
symbolic links between files or directories, create	ln(1)
symbolic links used to resolve a path name, maximum number of	fs_symlinks(5)
symbolic (soft) file link	symlink(4)
symbolic translation file for localedef scripts	charmap(4)
symlink() - make symbolic link to a file	symlink(2)
symlink - symbolic file link	symlink(4)
sync - flush unwritten system buffers to disk	sync(1M)
sync() - update disk	sync(2)
syncer - periodically sync for file system integrity	syncer(1M)
synchronization mechanism, lightweight	postwait(2)
synchronize a file's in-core state with its state on disk	fsync(2)
synchronize a mapped file	msync(2)
synchronize a window with its parents or children	syncok(3X)
synchronize asynchronous I/O	aio_fsync(2)
synchronize stale logical volume mirrors in LVM volume groups	vgfsync(1M)

Index

All Volumes

Description	Entry Name(Section)
synchronize stale mirrors in LVM logical volumes	lvsync(1M)
synchronize the system clock; correct the time to	adjtime(2)
synchronize transport library for transport endpoint (X/OPEN TLI-XTI)	t_sync(3)
synchronous I/O multiplexing	select(2)
synchronous writes	fs_async(5)
syncok() - synchronize a window with its parents or children	syncok(3X)
syntax checking tool; named configuration file	named-checkconf(1)
syntax of the Role-Based Access Control (RBAC) database files, verify the	rbacdbchk(1M)
syntax; a shell (command interpreter) with C-like	csh(1)
sys_attrs_kevm - KEVM (Kernel Event Manager) subsystem attributes	sys_attrs_kevm(5)
syscon - system console interface	console(7)
sysconf() - get configurable system variables	sysconf(2)
sysdef - display system definition	sysdef(1M)
sysdiff - HP-UX installed software comparator	sysdiff(1)
sysfs() - get file system type info	sysfs(2)
syslog() - control system log	syslog(3C)
syslogd - log system messages	syslogd(1M)
system	glossary(9)
system accounting; shell procedures for	acctsh(1M)
system activity reporter	sar(1M)
system administration manager	sam(1M)
system administrator	passwd(1)
system alias database	elm(1)
system alias text file	elm(1)
system alias: install new elm aliases	newalias(1)
system aliases, elm , verify and display	elmalias(1)
system and set or get audit files; start or halt the auditing	audctl(2)
system and user login data; display	logins(1M)
system asynchronous I/O	glossary(9)
system at any time, maximum number of System V IPC messages in the	msgctl(5)
system buffers, flush unwritten buffers to disk	sync(1M)
system buffers, periodically flush unwritten buffers to disk	syncer(1M)
system cache of recently looked-up names, get entries from	pstat(2)
system call	glossary(9)
system call audit status; change or display	audevent(1M)
system calls currently being audited; get events and	getevent(2)
system calls to be audited; set current events and	setevent(2)
system calls without error checking; execute link() and unlink()	link(1M)
system calls, BSD-4.2-compatible kill() , and signal()	bsdproc(3C)
system calls, STREAMS	stream(2)
system calls; introduction to	intro(2)
system clock; correct the time to synchronize the	adjtime(2)
system complex; modify an attribute of a	cplxmodify(1M)
system configuration drift analyzer	bastille_drift(1M)
system configuration file, change	ch_rc(1M)
system console	glossary(9)
system console interface	console(7)
system copy; UNIX system to UNIX	uucp(1)
system crash dumps; configure	crashconf(1M)
system crash dumps; configure	crashconf(2)
system database of automatically pushed STREAMS modules; manage	autopush(1M)
system debugger (generic); file	fsdb(1M)
system default database entry for a trusted system; manipulate	getprdfent(3)
system default database file for a trusted system	default(4)
system definition, display	sysdef(1M)
system description configuration files	system(4)
system descriptor file entry; get file	getmntent(3X)
system diagnostic messages, collect to form error log	dmesg(1M)
system dumps memory pages compressed or uncompressed when a kernel panic occurs; selects whether the	dump_compress_on(5)
system error messages; write	perror(3C)

Description	Entry Name(Section)
system file, create a kernel	create_sysfile(1M)
system for paging; enable device or file	swapon(1M)
system function to return [EOVERFLOW] if values do not fit in fields; causes uname ()	uname_overflow (5)
system (generic), construct a file	mkfs(1M)
system information, display	uname(1)
system () - issue a shell command	system(3S)
system language on multi-language systems; configure	geocustoms(1M)
system loader, initial	isl(1M)
system lockdown tool	bastille(1M)
system log, make entries in	logger(1)
system log; control	syslog(3C)
system maintenance commands and application programs, introduction to	intro(1M)
System Management Homepage (HP SMH); HP	smh(1M)
System Management Homepage (HP SMH); launch the Disks and File Systems tool of HP	fsweb(1M)
System Management Homepage (HP SMH); launch the Network Interfaces Configuration and Network Services Configuration tools of HP	ncweb(1M)
System Management Homepage server; starts or stops the HP	hpsmh(1M)
system messages; log	syslogd(1M)
system name, display/set	uname(1)
system name; set	uname(2)
system node names; enable maximum length expansion	expanded_node_host_names(5)
system node, get information about an SCA	pstat(2)
system of process's expected paging behavior, advise	madvise(2)
system only; convert string form to access control list (ACL) structure, HFS file	strtoacl(3C)
system or depot; check security-bulletin compliance state of HP-UX 11.x	security_patch_check(1M)
system over LAN, log in on another	vt(1)
system paging space information	swapinfo(1M)
system physical environment daemon	envd(1M)
system process	glossary(9)
system scheduling priority	renice(1M)
System server or map master; list which host is Network Information	ypwhich(1)
system shells, overview of various	sh(1)
system size (generic), extend a file	extendfs(1M)
system size, extend HFS file	extendfs_hfs(1M)
system startup and shutdown, timed	power_onoff(1M)
system status server	rwhod(1M)
system support for local network packet routing	routing(7)
system swap space; manage and configure	swaptctl(2)
system - system description configuration files	system(4)
system to dump memory using multiple dump units when a kernel panic occurs on Integrity systems; enable or disable option for	dump_concurrent_on(5)
system to UNIX system command execution, UNIX	uux(1)
system to UNIX system copy; UNIX	uucp(1)
system to UNIX system file copy, public UNIX	uuto(1)
system up time, show	uptime(1)
system users, list current	who(1)
System V IPC message segment (OBSOLETE); number of bytes in a	msgsz(5)
System V IPC message segments in the system (OBSOLETE); number of	msgseg(5)
System V IPC message size in bytes (OBSOLETE); maximum	msgmax(5)
System V IPC message space resource map (OBSOLETE); number of entries in the	msgmap(5)
System V IPC messages at boot time (OBSOLETE); enable or disable	mesg(5)
System V IPC semaphore, maximum value of any single	semvmx(5)
System V IPC semaphores at boot time, enable or disable	sema(5)
System V IPC semaphores per identifier, maximum number of	semmsl(5)
System V IPC semop () call, maximum cumulative value changes per	semaem(5)
System V IPC subsystem, get information about the	pstat(2)
System V IPC system-wide semaphore identifiers, number of	semmsl(5)
System V IPC system-wide semaphore undo structures, number of	semmsl(5)
System V IPC undo entries per process, maximum number of	semume(5)
System V semaphore set, get information for a	pstat(2)

Index

All Volumes

Description	Entry Name(Section)
System V shared memory segment identifiers in the system, number of	shmmni(5)
System V shared memory segment, get information for a	pstat(2)
System V shared memory segment, maximum size (in bytes) for a	shmmax(5)
System V shared memory segments per process, maximum number of	shmseg(5)
System V shared memory, enable or disable	shmem(5)
System V system-wide semaphores, number of	semms(5)
system variables; get configurable	sysconf(2)
system with compaction; copy HFS file	dcopy(1M)
system with label checking; copy a file	volcopy(1M)
system with label checking; copy HFS file	volcopy_hfs(1M)
system's crash dump configuration, get information for a	pstat(2)
system's stable storage area, get information from the	pstat(2)
system, boot	reboot(2)
system, construct an HFS file	mkfs_hfs(1M)
system, display information	uname(1)
system, get dynamic information about the	pstat(2)
system, get information about the	pstat(2)
system, list users currently on the	users(1)
system, number of System V shared memory segment identifiers in the	shmmni(5)
system, send LP request to remote	rlp(1M)
system, set node name	uname(1)
system, set or display name of current host	hostname(1)
system, terminal emulator; call another (UNIX)	cu(1)
system-calls error indicator	errno(2)
system-selectable page size; maximum (in kilobytes) of	vps_ceiling(5)
system-wide clock, get current value of	getclock(3C)
system-wide clock, set value of	setclock(3C)
system-wide limit; percentage of file cache that can be consumed by sequential accesses, per	fcache_seqlimit_system(5)
system-wide or per-process information of a ccNUMA system, returns	pstat_getlocality(2)
system-wide semaphore identifiers, number of System V IPC	semmni(5)
system-wide semaphore undo structures, number of System V IPC	semnu(5)
system-wide semaphores, number of System V	semms(5)
system-wide sendmail aliases, print	praliases(1)
system; add a new group to the	groupadd(1M)
system; add a new user login to the	useradd(1M)
system; check if system has been converted to a trusted	iscomsec(2)
system; configure the LP spooling	lpadmin(1M)
system; construct a new file	newfs(1M)
system; construct a new HFS file	newfs_hfs(1M)
system; delete a group from the	groupdel(1M)
system; delete a user login from the	userdel(1M)
system; log in to	login(1)
system; manage the interrupt configuration of the	intctl(1M)
system; manipulate device assignment database entry for a trusted	getdvagent(3)
system; manipulate system default database entry for a trusted	getprdfent(3)
system; modify a group on the	groupmod(1M)
system; mount a file	vfsmount(2)
system; print status of LP spooler requests on a remote	rlpstat(1M)
system; system default database file for a trusted	default(4)
system; test for successful login to remote	uucp(1)
system; transfer files for the uucp	uucico(1M)
system; trusted system device assignment database file	devassign(4)
systems description file format; PPP neighboring	ppp.Systems(4)
systems running HP-UX; emulate PA-RISC HP-UX applications on Itanium-based	Aries(5)
systems; assembler for Integrity	as_ia(1)
systems; assembler for PA-RISC	as_pa(1)
systems; change program's internal attributes on Integrity	chatr_ia(1)
systems; configure system language on multi-language	geocostoms(1M)
systems; determine the shutdown status of HFS file	fsclean(1M)
systems; enable or disable option for system to dump memory using multiple dump units when a kernel panic	

Description	Entry Name(Section)
occurs on Integrity	dump_concurrent_on(5)
systems; execution startup routines for Integrity	crt0_ia(3)
systems; execution startup routines for PA-RISC	crt0_pa(3)
systems; explicit load of shared libraries for Integrity	shl_load_ia(3X)
systems; explicit load of shared libraries for PA-RISC	shl_load_pa(3X)
systems; get entries from name list on PA-RISC	nlist_pa(3C)
systems; get secure password file entry on trusted	getspwnt(3X)
systems; link editor for Integrity	ld_ia(1)
systems; link editor for PA-RISC	ld_pa(1)
systems; list dynamic dependencies of executable files or shared libraries on Integrity	ldd_ia(1)
systems; list dynamic dependencies of executable files or shared libraries on PA-RISC	ldd_pa(1)
systems; list uucp names of known	uucp(1)
systems; make local resource available for mounting by remote	share(1M)
systems; make local resource unavailable for mounting by remote	unshare(1M)
systems; mount and unmount CDFS file	mount_cdfs(1M)
systems; mount and unmount HFS file	mount_hfs(1M)
systems; open a shared library on Integrity	dlopen_ia(3C)
systems; protected password authentication database files for trusted	prpwd(4)
systems; security databases for trusted	authcap(4)
systems; structure formats for Integrity	nlist_ia(4)
systems; structure formats for PA-RISC	nlist_pa(4)
systems; terminal control database file for trusted	ttys(4)
sys tty - system console interface	console(7)
sysv_hash_locks - System V IPC hashed spinlock pool size	sysv_hash_locks(5)
t_accept() - accept a connect request issued by a transport user (X/OPEN TLI-XTI)	t_accept(3)
t_alloc() - allocate transport function library structure	t_alloc(3)
t_bind() - bind address to transport endpoint (X/OPEN TLI-XTI)	t_bind(3)
t_close() - close transport endpoint (X/OPEN TLI-XTI)	t_close(3)
t_connect() - establish connection with another transport user (X/OPEN TLI-XTI)	t_connect(3)
t_error() - error message function (X/OPEN TLI-XTI)	t_error(3)
t_free() - free memory for library structure (X/OPEN TLI-XTI)	t_free(3)
t_getinfo() - get protocol-specific service information (X/OPEN TLI-XTI)	t_getinfo(3)
t_getprotaddr() - get protocol address (X/OPEN XT I)	t_getprotaddr(3)
t_getstate() - get current state (X/OPEN TLI-XTI)	t_getstate(3)
t_listen() - listen for connect request (X/OPEN TLI-XTI)	t_listen(3)
t_look() - look at current event on transport endpoint (X/OPEN TLI-XTI)	t_look(3)
t_open() - establish transport endpoint (X/OPEN TLI-XTI)	t_open(3)
t_optmgmt() - manage options for a transport endpoint	t_optmgmt(3)
t_rcv() - receive normal or expedited data sent over connection (X/OPEN TLI-XTI)	t_rcv(3)
t_rcvconnect() - receive confirmation from connect request (X/OPEN TLI-XTI)	t_rcvconnect(3)
t_rcvdis() - retrieve information from disconnect (X/OPEN TLI-XTI)	t_rcvdis(3)
t_rcvrel() - acknowledge receipt of release indication at transport endpoint (X/OPEN TLI-XTI)	t_rcvrel(3)
t_rcvudata() - receive data unit from remote transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)
t_rcvuderr() - receive error information from unit data error indication (X/OPEN TLI-XTI)	t_rcvuderr(3)
t_snd() - send data or expedited data over a connection (X/OPEN TLI-XTI)	t_snd(3)
t_snddis() - send user-initiated disconnect request (X/OPEN TLI-XTI)	t_snddis(3)
t_sndrel() - initiate orderly release at transport endpoint (X/OPEN TLI-XTI)	t_sndrel(3)
t_sndudata() - send data unit to transport user (X/OPEN TLI-XTI)	t_sndudata(3)
t_strerror() - produce error message string (X/OPEN - XT I)	t_strerror(3)
t_sync() - synchronize transport library for transport endpoint (X/OPEN TLI-XTI)	t_sync(3)
t_unbind() - disable transport endpoint (X/OPEN TLI-XTI)	t_unbind(3)
table entries; add, remove and list gsscred	gsscred(1M)
table preprocessor for nroff	tbl(1)
table to name server file format; translate host	hosts_to_named(1M)
table, eliminate duplicate entries in a	lsearch(3C)
table, linear search for entry; optional update if missing	lsearch(3C)
table, mounted file system	mnttab(4)
table, shared file system	sharetab(4)
table, symbol, for object code file, print	nm(1)
table; remove duplicate entries from gsscred mapping	gsscred_clean(1M)
table; retrieve archive symbol	elf_getarsym(3E)

Index

All Volumes

Description	Entry Name(Section)
tables, binary search routine for sorted	bsearch(3C)
tables, determines the size of the networking hash	tcphashsz(5)
tables, hash search, manage	hsearch(3C)
tables; generate iconv translation	genxlt(1)
tables; manually manipulate routing	route(1M)
tabs - set tabs on a terminal	tabs(1)
tabs, convert to spaces and vice versa	expand(1)
TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for	fcmsutil(1M)
TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage Utility Command for TACHYON TL	fcmsutil(1M)
taddr2uaddr () - generic transport name-to-address translation	netdir(3N)
tags file, create a	ctags(1)
tail - get lines from last part of a file	tail(1)
tails the mail log	mtail(1M)
talk - talk to another user	talk(1)
talk to another user	talk(1)
talk to the keyserver process	keyenvoy(1M)
talkd - remote user communication server	talkd(1M)
tan () - tangent function	tan(3M)
tand () - tangent function of a degree argument	tand(3M)
tandf () - tangent function of a degree argument (float)	tand(3M)
tandl () - tangent function of a degree argument (long double)	tand(3M)
tandq () - tangent function of a degree argument (quad)	tand(3M)
tandw () - tangent function of a degree argument (extended)	tand(3M)
tanf () - tangent function (float)	tan(3M)
tangent functions	tan(3M)
tangent functions of a degree argument	tand(3M)
tanh () - hyperbolic tangent function	tanh(3M)
tanhf () - hyperbolic tangent function (float)	tanh(3M)
tanh1 () - hyperbolic tangent function (long double)	tanh(3M)
tanhq () - hyperbolic tangent function (quad)	tanh(3M)
tanhw () - hyperbolic tangent function (extended)	tanh(3M)
tanl () - tangent function (long double)	tan(3M)
tanq () - tangent function (quad)	tan(3M)
tanw () - tangent function (extended)	tan(3M)
tape archive format, tar	tar(4)
tape device on open; determines whether to reserve a	st_ats_enabled(5)
tape dump and restore protocol module, remote magnetic	rmt(1M)
tape file archiver	tar(1)
tape files: convert, reblock, translate, and copy	dd(1)
tape I/O; faster	ftio(1)
tape manipulating program; magnetic	mt(1)
tape, package software products into	swpackage(1M)
tape, shared tape administration	st(1M)
tape2, magnetic tape interface for stape and	mt(7)
tape; initialize disk or partition DDS	mediainit(1)
tar tape archive format	tar(4)
tar - tape file archiver	tar(1)
target depot, package software products into	swpackage(1M)
target process to run serially with other processes; force	serialize(1)
target process to run serially with other processes; force	serialize(2)
target root, modify software products in depot or	swmodify(1M)
tbl - table preprocessor for nroff	tbl(1)
tbl, nroff/troff, and neqn constructs, remove	deroff(1)
tcdrain (): tty line control function	tcccontrol(3C)
tcflow (): tty line control function	tcccontrol(3C)
tcflush (): tty line control function	tcccontrol(3C)
tcgetattr () - get tty device operating parameters	tcattribute(3C)
tcgetpgrp (): get foreground process group ID	tcgetpgrp(3C)
tcgetsid () - get terminal session ID	tcgetsid(3C)

Description	Entry Name(Section)
TCP connection, identify user	idlookup(1)
TCP - Internet Transmission Control Protocol	TCP(7P)
tcp wrapper configuration	tcpdchk(1)
tcp wrapper service requests, evaluate	tcpdmatch(1)
TCP Wrappers, utility programs for	tryfrom(1)
TCP/IP IDENT protocol server	identd(1M)
TCP/IP server daemon; UUCP over	uucpd(1M)
tcpd - access control facility for internet services	tcpd(1M)
tcpd configuration file	tcpd.conf(4)
tcpd.conf - configuration file for tcpd	tcpd.conf(4)
tcpdchk - check tcp wrapper configuration	tcpdchk(1)
tcpdmatch - evaluate tcp wrapper service requests	tcpdmatch(1)
tcphashsz - determines the size of the networking hash tables	tcphashsz(5)
tcsendbreak() : tty line control function	tccontrol(3C)
tcsetattr() - set tty device operating parameters	tcattribute(3C)
tcsetpgrp() - set foreground process group ID	tcsetpgrp(3C)
tdelete() - delete a node from a binary search tree	tsearch(3C)
tee - pipe fitting to copy standard output to file	tee(1)
telldir() - get current location of named directory stream	directory(3C)
telm - STREAMS Telnet (pseudo-terminal) driver	tels(7)
Telnet drivers, STREAMS	tels(7)
Telnet port identification feature; dedicated ports file used by DDFA software and	dp(4)
TELNET protocol server	telnetd(1M)
TELNET protocol, user interface to the	telnet(1)
telnet - user interface to the TELNET protocol	telnet(1)
telnetd - TELNET protocol server	telnetd(1M)
tels - STREAMS Telnet (pseudo-terminal) driver	tels(7)
template file; Event Manager	evmtemplate(4)
templates, defines and manages file system stack	fstadm(2)
tmpnam() - create a name for a temporary file	tmpnam(3S)
temporary file, create a	tmpfile(3S)
temporary file, create a name for	tmpnam(3S)
temporary file, make a name for a	mktemp(1)
term - format of compiled terminfo file	term(4)
TERM - terminal capabilities	term_c(4)
term - terminal capabilities	term_c(4)
term.h - terminal capabilities	term_c(4)
term_attrs() - get supported terminal video attributes	termattrs(3X)
term_c - terminal capabilities	term_c(4)
termattrs() - get supported terminal video attributes	termattrs(3X)
termcap description, convert into a terminfo description	captoinfo(1M)
termcap - emulate /usr/share/lib/termcap access routines	termcap(3X)
terminal	glossary(9)
terminal affiliation	glossary(9)
terminal and printer handling and optimization package	curses_intro(3X)
terminal baud rate, get	baudrate(3X)
terminal block mode interface	blmode(7)
terminal capabilities	term_c(4)
terminal capabilities, disable use of	filter(3X)
terminal capabilities, get from terminfo database	tput(1)
terminal capability database	terminfo(4)
terminal connection, set terminal type, modes, speed and line discipline for 2-way line	ugetty(1M)
terminal connections	kermit(1)
terminal control database entry, manipulate	getprtcent(3)
terminal control database file for trusted systems	ttys(4)
Terminal Controller Device File Access software, Data Communications and	ddfa(7)
terminal device	glossary(9)
terminal device, control (Bell Version 6 compatibility)	stty(2)
terminal driver; pseudo-	pty(7)
terminal echo, enable/disable	echo(3X)
terminal emulator, keyboard mapping	itemap(1M)

Index

All Volumes

Description	Entry Name(Section)
Terminal Emulator, number of scrollable lines used by the Internal	scroll_lines(5)
terminal emulator; call another (UNIX) system,	cu(1)
terminal identification program	ttytype(1)
terminal insert and delay capability, query functions	has_ic(3X)
terminal interface device file	tty(7)
terminal interface for Version 6/PWB compatibility	sttyv6(7)
terminal interface, extended general	termiox(7)
terminal interface; controlling	tty(7)
terminal interface; general	termio(7)
terminal line connection; establish an outgoing	dial(3C)
terminal line discipline module, STREAMS	ldterm(7)
terminal mode, save/restore	resetty(3X)
terminal modes; save or restore program or shell	def_prog_mode(3X)
terminal name, get	termname(3X)
terminal or pseudo-terminal, get name of user's	tty(1)
terminal output control functions	clearok(3X)
terminal port; set the options for a	stty(1)
terminal refresh, immediate, enable/disable	immedok(3X)
terminal screen, clear	clear(1)
terminal screen, number of columns	COLS(3X)
terminal screen, number of lines	LINES(3X)
terminal session ID, get	tcgetsid(3C)
Terminal Session Manager	tsm(1)
Terminal Session Manager state information, get	tsm.info(1)
Terminal Session Manager, TSM; send commands to the	tsm.command(1)
terminal session; make typescript of	script(1)
terminal session; sign on; start	login(1)
terminal session; start	login(1)
terminal settings and datacomm line speed used by getty	gettydefs(4)
terminal type, modes, speed and line discipline, set for 2-way line	uugetty(1M)
terminal type, modes, speed, and line discipline, set	getty(1M)
terminal type, modes, speed, and line discipline; set	getty(1M)
terminal video attributes, get supported ones	termattr(3X)
terminal, convert underscores to underlining on	ul(1)
terminal, deny or permit <i>write</i> (1) messages from other users to	mesg(1)
terminal, find name of	ttyname(3C)
terminal, generate file name of controlling	ctermid(3S)
terminal, get a multi-byte character length limited string from	getnstr(3X)
terminal, get a multi-byte character string	getstr(3X)
terminal, get a single-byte character	getch(3X)
terminal, get a wide character from	get_wch(3X)
terminal, get name	termname(3X)
terminal, get name of user logged in on this terminal	getlogin(3C)
terminal, get verbose description of	longname(3X)
terminal, information on current terminal	cur_term(3X)
terminal, initialize based on terminal type	tset(1)
terminal, lock against use by others	lock(1)
terminal, output attributes	vidattr(3X)
terminal, output commands to	putp(3X)
terminal, output cursor movement commands to	mvcur(3X)
terminal, read password from while suppressing echo	getpass(3C)
terminal, remote, spawn getty to (call terminal)	ct(1)
terminal, set tabs on a	tabs(1)
terminal-dependent initialization	tset(1)
terminal-type data base for each tty port	ttytype(4)
terminal: spawn getty to (call) remote terminal	ct(1)
terminal; get an array of wide characters and function key codes from a	getn_wstr(3X)
terminal; pseudo-terminal driver	pty(7)
terminals, HP 2640- and HP 2621-series, handle special functions of	hp(1)
terminals, VT320, VT100, Wyse60	swinstall(1M)
terminals, VT320, VT100, Wyse60	swremove(1M)

Description	Entry Name(Section)
terminate a per-process timer	rmtimer(3C)
terminate a process	exit(2)
terminate a process	kill(1)
terminate all active processes	killall(1M)
terminate all system processing	shutdown(1M)
terminate, cause the calling thread to terminate	pthread_exit(3T)
terminate, wait for background processes to	wait(1)
terminate; wait for child process to stop or	wait(2)
terminating PAM sessions	pam_open_session(3)
termination of a specified thread, wait for	pthread_join(3T)
termination, register a function to be called at program	atexit(3)
terminfo data base compiler	tic(1M)
terminfo data base, de-compile	untic(1M)
terminfo database; interface to	del_curterm(3X)
terminfo database; retrieve capabilities from	tigetflag(3X)
terminfo de-compiler	untic(1M)
terminfo description, convert from a termcap description	captainfo(1M)
terminfo descriptions; compare or print out	infocmp(1M)
terminfo - printer, terminal, and modem capability database	terminfo(4)
termio - general terminal interface	termio(7)
termios - general terminal interface	termio(7)
termiox - extended general terminal interface	termiox(7)
termname () - get terminal name	termname(3X)
terms; description of common HP-UX	glossary(9)
test and maintain consistency between the kernel I/O data structures and the ioconfig files	ioinit(1M)
test - evaluate condition for true or false	test(1)
test - evaluate conditional expression	csh(1)
test - evaluate conditional expression	ksh(1)
test - evaluate conditional expression	sh-posix(1)
test for successful login to remote system	uucp(1)
test packets, send	ping(1M)
test, initialize, and manipulate signal sets	sigsetops(3C)
text allocation space of object files, print section sizes and	size(1)
text editor	see <i>editor</i>
text editor; extended line-oriented	ex(1)
text editor; line-oriented	ed(1)
text editor; screen-oriented (visual)	vi(1)
text editor; stream	sed(1)
text file	glossary(9)
text file for CRT or line-printer output, format	nroff(1)
text file format specification	fspec(4)
text file; change or reformat a	newform(1)
text formatter	fmt(1)
text formatter; simple	adjust(1)
text pattern-directed scanning and processing language	awk(1)
text processors: remove preprocessor lines	unifdef(1)
text processors: reverse the left-to-right text character sequence in each line of a file	rev(1)
text segment for any user process; maximum size (in bytes) of the	maxtsiz(5)
text string, read from message file	gettxt(3C)
text version of EVM status code; format	EvmStatusTextGet(3)
text, lock in memory	plock(2)
tfind () - get data pointer for binary search tree	tsearch(3C)
tftp - trivial file transfer program	tftp(1)
tftpd - trivial file transfer protocol server	tftpd(1M)
tgamma () - true gamma function	tgamma(3M)
tgammaf () - true gamma function (float)	tgamma(3M)
tgammal () - true gamma function (long double)	tgamma(3M)
tgammaq () - true gamma function (quad)	tgamma(3M)
tgammaw () - true gamma function (extended)	tgamma(3M)
tgetent () - emulate /usr/share/lib/termcap access routines	termcap(3X)
tgetent () - get compiled terminfo data base entry into buffer	termcap(3X)

Index

All Volumes

Description	Entry Name(Section)
tgetflag() - emulate /usr/share/lib/termcap access routines	termcap(3X)
tgetflag() - get availability of compiled boolean terminal capability	termcap(3X)
tgetnum() - emulate /usr/share/lib/termcap access routines	termcap(3X)
tgetnum() - get numeric value of compiled terminal capability	termcap(3X)
tgetstr() - emulate /usr/share/lib/termcap access routines	termcap(3X)
tgetstr() - get string value of compiled terminal capability	termcap(3X)
tgoto() - emulate /usr/share/lib/termcap access routines	termcap(3X)
tgoto() - get compiled terminal cursor addressing string	termcap(3X)
the amount of memory to reserve for the 32-bit DMA pool	dma32_pool_size(5)
thread attribute object, initialize or destroy	pthread_attr_init(3T)
thread cancellation cleanup handler, register or remove	pthread_cleanup_pop(3T)
thread condition variable attributes object, initialize or destroy	pthread_condattr_init(3T)
thread ID for the calling thread, obtain	pthread_self(3T)
thread identifiers, compare	pthread_equal(3T)
thread launch policy, setting	pthread_launch_policy(3T)
thread of execution, create	pthread_create(3T)
thread or LWP in a process, get information for a	pstat(2)
thread process-shared attribute, get or set	pthread_condattr_getpshared(3T)
thread to a processor set; bind process or	pset_bind(2)
thread, cancel execution of	pthread_cancel(3T)
Thread-Safe	thread_safety(5)
thread-safe, get, set, or end network host entry	gethostent(3N)
thread-safe, get, set, or end protocol entry	getprotoent(3N)
thread-safe, get, set, or end service entry	getservent(3N)
thread-specific data associated with a key, get or set	pthread_getspecific(3T)
thread-specific data key; create or destroy	pthread_key_create(3T)
thread; get or set the nice value	pthread_get_nice_np(3T)
thread; sets scheduling priority of a	pthread_setschedprio(3T)
thread_safety - list of libc, libpthread and libgen interfaces that are: Not Thread-Safe, Cancellation Points, Cancel Safe, Async Signal Safe, or Async Cancel Safe	thread_safety(5)
threads allowed per process, defines the maximum number of	max_thread_proc(5)
threads allowed to run simultaneously; limits the number of	nkthread(5)
threads waiting on a condition variable, unblock	pthread_cond_signal(3T)
threads, desirable ratio between number of pending AIO requests and servicing	aio_req_per_thread(5)
threads, list of external options to specify the scheduling contention scope of	pthread_scope_options(5)
threads, POSIX.1c introduction	pthread(3T)
three-way differential file comparison	diff3(1)
three-way file merge	merge(1)
tic - terminfo data base compiler	tic(1M)
ticket-granting ticket; obtain and cache the Kerberos	kinit(1)
ticket; obtain and cache the Kerberos ticket-granting	kinit(1)
tickets; destroy Kerberos	kdestroy(1)
tickets; list cached Kerberos	klist(1)
ticks per second, scheduling interval in clock	timeslice(5)
tigetflag() - retrieve capabilities from the terminfo database	tigetflag(3X)
tigetnum() - retrieve capabilities from the terminfo database	tigetflag(3X)
tigetstr() - retrieve capabilities from the terminfo database	tigetflag(3X)
time, times - print summary of time used by processes	ksh(1)
time a command	time(1)
time a command; report process accounting data and system activity	timex(1)
time and date conversion	strptime(3C)
time and date, get more precisely (Version 7 compatibility only)	ftime(2)
time and date, set via NTP	ntpdate(1M)
time and date; display or set	date(1)
time between reboots, evaluate	last(1)
time delay; execute commands after	at(1)
time() - get time	time(2)
time interval (in secs) for flushing audit records; determine	diskaudit_flush_interval(5)
time interval, suspend execution for a	sleep(1)
time - measure time used to execute a command	time(1)

Description	Entry Name(Section)
time - print elapsed time used by a pipeline	sh-posix(1)
time - print summary of time used by shell and children	csh(1)
time profile, execution	profil(2)
time - time a command	time(1)
time to leave, notify you when it is	leave(1)
time to string; convert date and	ctime(3C)
time to synchronize the system clock; correct the	adjtime(2)
time to wide-character string; convert date and	wcsftime(3C)
time used; report CPU	clock(3C)
time zone adjustment table for date and ctime()	tztzab(4)
time, get	time(2)
time, get high resolution	gethrtime(3C)
time, get the date and	gettimeofday(2)
time, maximum number of System V IPC messages in the system at any	msgctl(5)
time, set	stime(2)
time, set the date and	settimeofday(2)
time; convert to string	strptime(3C)
time; convert user format date and	getdate(3C)
timed wait on a condition variable; wait or	pthread_cond_wait(3T)
timed, automatic system power on, and power off	power_onoff(1M)
timed-job execution daemon	cron(1M)
timeout() - control blocking on input	notimeout(3X)
timeout value of a filter; sets the sendmail connection	smfi_settimeout(3N)
timer expires; sets action taken if IPMI watchdog	ipmi_watchdog_action(5)
timer operations	timers(2)
timer, allocate a per-process	mktimer(3C)
timer, free a per-process	rmtimer(3C)
timer, get value of a per-process	gettimer(3C)
timer, relatively arm a per-process	reltimer(3C)
timer, set or get value of process interval	getitimer(2)
timer, set the interval timer	ualarm(2)
timer_create() - create timer	timers(2)
timer_delete() - delete timer	timers(2)
timer_getoverrun() - return timer expiration count	timers(2)
timer_gettime() - store timer expiration and reload value	timers(2)
timer_settime() - set timer expiration	timers(2)
timers - timer operations	timers(2)
times() - get process and child process times	times(2)
times of file; update access, modification, and/or change	touch(1)
times - print summary of time used by processes	sh-posix(1)
times, file access and modification, set or update	utime(2)
times, set file access and modification times	utimes(2)
times; get process and child process	times(2)
TIMESHARE scheduling policy	rtsched(2)
timeslice - scheduling interval in clock ticks per second	timeslice(5)
timex - time a command; report process accounting data and system activity	timex(1)
timezone - difference between Universal (Greenwich mean) and local time	timezone(5)
timezone() - difference between UTC and local timezone	ctime(3C)
timod - STREAMS module for converting ioctl() calls into Transport Interface messages	timod(7)
tirdwr - STREAMS module for reads and writes by Transport Interface users	tirdwr(7)
TLI function; accept a connect request issued by a transport user	t_accept(3)
TLI function; acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
TLI function; bind address to transport endpoint	t_bind(3)
TLI function; close transport endpoint	t_close(3)
TLI function; disable transport endpoint	t_unbind(3)
TLI function; error message function	t_error(3)
TLI function; establish connection with another transport user	t_connect(3)
TLI function; establish transport endpoint	t_open(3)
TLI function; free library structure	t_free(3)
TLI function; get current state	t_getstate(3)
TLI function; get protocol-specific service information	t_getinfo(3)

Index

All Volumes

Description	Entry Name(Section)
TLI function; initiate orderly release at transport endpoint	t_sndrel(3)
TLI function; listen for connect request	t_listen(3)
TLI function; look at current event on transport endpoint	t_look(3)
TLI function; receive confirmation from connect request	t_rcvconnect(3)
TLI function; receive data over connection	t_rev(3)
TLI function; receive data unit from remote transport provider user	t_rcvudata(3)
TLI function; receive error information from unit data error indication	t_revuderr(3)
TLI function; retrieve disconnect information	t_revdis(3)
TLI function; send data or expedited data over a connection	t_snd(3)
TLI function; send data unit to transport user	t_sndudata(3)
TLI function; send user-initiated disconnect request	t_snddis(3)
TLI function; synchronize transport library for transport endpoint	t_sync(3)
tmpfile() - create a temporary file	tmpfile(3S)
tmpfile64() - file system API to support large files	fgetpos64(3S)
tmpnam() - create a name for a temporary file	tmpnam(3S)
toascii() - translate characters to 7-bit ASCII	conv(3C)
tokens for kernel RPC; generates and validates GSS-API	gssd(1M)
tolower() _tolower - translate characters to lower-case	conv(3C)
tool for DNSSEC; key generation	dnssec-keygen(1)
tool of HP System Management Homepage (HP SMH); launch the Disks and File Systems	sigweb(1M)
tool; DNSSEC zone signing	dnssec-signzone(1)
tool; named configuration file syntax checking	named-checkconf(1)
tool; rndc key generation	rndc-confgen(1)
tool; system lockdown	bastille(1M)
tool; zone validity checking	named-checkzone(1)
tools available through HP VUE (OBSOLETE); audio	Audio(5)
top - display and update information about top processes on system	top(1)
top processes on system; display and update information about	top(1)
topological sort	tsort(1)
total accounting files, merge or add	acctmerge(1M)
total accounting records, convert per-session records to	acctcon(1M)
touch - update access, modification, and/or change times of file	touch(1)
touchline() - window refresh control functions	is_linetouched(3X)
touchwin() - window refresh control function	touchwin(3X)
toupper() _toupper - translate characters to upper-case	conv(3C)
towctrans() - character transliteration	towctrans(3C)
tolower() - translate wide characters to lowercase	wconv(3C)
toupper() - translate wide characters to uppercase	wconv(3C)
tparm() - retrieve capabilities from the terminfo database	tigetflag(3X)
tput - query the terminfo database	tput(1)
tputs() - decode terminal string padding information	termcap(3X)
tputs() - emulate /usr/share/lib/termcap access routines	termcap(3X)
tputs() - output commands to the terminal	putp(3X)
tr - translate selected characters	tr(1)
trace and log command, configure subsystem database	nettlconf(1M)
trace back of the procedure call stack using the unwind library, produce a	U_STACK_TRACE(3X)
trace for each LWP in each process and core file; print a stack	pstack(1)
trace messages to standard output, write STREAMS event trace messages	strace(1M)
tracing and logging administration manager; network	nettladm(1M)
tracing and logging binary files, format	netfmt(1M)
tracing and logging configuration file; network	nettlgen.conf(4)
tracing and logging; control network	nettl(1M)
tracing facility for multithreaded processes	ttrace(2)
tracking of current and root directories for auditing subsystem; enable/disable	audit_track_paths(5)
traffic statistics, print mail	mailstats(1)
transaction routines for PAM; authentication	pam_start(3)
transactions; query log file of uucp	uucp(1)
transcription system	answer(1)
transfer data between user and kernel space in a lightweight manner; Mercury Library Interfaces to	hg(3)
transfer files between systems	ftp(1)

Description	Entry Name(Section)
transfer files for the uucp system	uucico(1M)
transfer files using XMODEM-protocol	umodem(1)
transfer NIS database from server to local node	ypxfr(1M)
transfer processes; Network Information Service (NIS) server, binder, and	ypserv(1M)
transfer program, trivial file	tftp(1)
transfer protocol server; file	ftpd(1M)
transfer protocol server; trivial file	tftpd(1M)
translate character code to another code set	iconv(3C)
translate characters to upper-case, lower-case, or 7-bit ASCII	conv(3C)
translate host table to name server file format	hosts_to_named(1M)
translate selected characters	tr(1)
translate wide characters to uppercase or lowercase	wconv(3C)
translate, convert, reblock and copy a (tape) file	dd(1)
translates exportfs options to share/unshare commands	exportfs(1M)
translation file for localedef scripts; symbolic	charmap(4)
translation tables to a readable format, dump iconv	dmpxlt(1)
translation tables; generate iconv	genxlt(1)
translation, class-dependent data, of ELF files	elf_xlate(3E)
translation, generic transport name-to-address	netdir(3N)
transliteration; character	towctrans(3C)
transmission by mailer; encode/decode a binary file for	uencode(1)
transmission, get maximum data length for	rpc_gss_max_data_length(3N)
transport endpoint; acknowledge receipt of release (X/OPEN TLI-XTI)	t_rcvrel(3)
transport endpoint; disable (X/OPEN TLI-XTI)	t_unbind(3)
transport endpoint; establish (X/OPEN TLI-XTI)	t_open(3)
transport endpoint; initiate orderly release (X/OPEN TLI-XTI)	t_sndrel(3)
transport endpoint; manage options for a	t_optmgmt(3)
transport endpoint; synchronize transport library (X/OPEN TLI-XTI)	t_sync(3)
transport files, schedule uucp	uused(1M)
transport function library structure; allocate	t_alloc(3)
Transport Interface messages: STREAMS module for converting ioctl()	timod(7)
Transport Interface: STREAMS module for reads and writes	tirdwr(7)
transport library; synchronize (X/OPEN TLI-XTI)	t_sync(3)
transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)
transport user; accept connect request (X/OPEN TLI-XTI)	t_accept(3)
transport user; establish connection (X/OPEN TLI-XTI)	t_connect(3)
transport user; send data unit (X/OPEN TLI-XTI)	t_sndudata(3)
trap enables: getting	fegettrapenable(3M)
trap enables: setting	fesettrapenable(3M)
trap - trap specified signal	ksh(1)
trap - trap specified signal	sh-posix(1)
traverse a binary search tree	tsearch(3C)
traverse (walk) a file tree, executing a function	ftw(3C)
tree hierarchy; prints the process	ptree(1)
tree, manage a binary search	tsearch(3C)
tree, search directory tree for files	find(1)
tree, walk a file, executing a function	ftw(3C)
trees; copy file archives in and out; duplicate directory	cpio(1)
triangle, right, hypotenuse of a	hypot(3M)
trigonometric arc hyperbolic cosine functions	acosh(3M)
trigonometric arc hyperbolic sine functions	asinh(3M)
trigonometric arc hyperbolic tangent functions	atanh(3M)
trigonometric arccosine functions	acos(3M)
trigonometric arcsine functions	asin(3M)
trigonometric arctangent functions	atan(3M)
trigonometric arctangent-and-quadrant functions	atan2(3M)
trigonometric complex absolute value functions	cabs(3M)
trigonometric complex arc hyperbolic cosine functions	cacosh(3M)
trigonometric complex arc hyperbolic sine functions	casinh(3M)
trigonometric complex arc hyperbolic tangent functions	catanh(3M)
trigonometric complex arccosine functions	cacos(3M)

Index

All Volumes

Description	Entry Name(Section)
trigonometric complex arcsine functions	casin(3M)
trigonometric complex arctangent functions	catan(3M)
trigonometric complex argument functions	carg(3M)
trigonometric complex conjugate functions	conj(3M)
trigonometric complex cosine functions	ccos(3M)
trigonometric complex exponential functions	cexp(3M)
trigonometric complex hyperbolic cosine functions	ccosh(3M)
trigonometric complex hyperbolic sine functions	csinh(3M)
trigonometric complex hyperbolic tangent functions	ctanh(3M)
trigonometric complex imaginary-part functions	cimag(3M)
trigonometric complex logarithm functions	clog(3M)
trigonometric complex power functions	cpow(3M)
trigonometric complex projection functions	cproj(3M)
trigonometric complex real-part functions	creal(3M)
trigonometric complex sine functions	csin(3M)
trigonometric complex square root functions	csqrt(3M)
trigonometric complex tangent functions	ctan(3M)
trigonometric cosine functions	cos(3M)
trigonometric cosine functions of a degree argument	cosd(3M)
trigonometric cosine plus i times sine	cis(3M)
trigonometric cotangent functions	cot(3M)
trigonometric cotangent functions of a degree argument	cotd(3M)
trigonometric degree-valued arccosine functions	acod(3M)
trigonometric degree-valued arcsine functions	asind(3M)
trigonometric degree-valued arctangent functions	atand(3M)
trigonometric degree-valued arctangent-and-quadrant functions	atan2d(3M)
trigonometric error and complementary error functions	erf(3M)
trigonometric hyperbolic cosine functions	cosh(3M)
trigonometric hyperbolic sine and hyperbolic cosine together	sinhcosh(3M)
trigonometric hyperbolic sine functions	sinh(3M)
trigonometric hyperbolic tangent functions	tanh(3M)
trigonometric sine and cosine of degree argument	sincod(3M)
trigonometric sine and cosine together	sincos(3M)
trigonometric sine functions	sin(3M)
trigonometric sine functions of degree argument	sind(3M)
trigonometric tangent functions	tan(3M)
trigonometric tangent functions of a degree argument	tand(3M)
trivial file transfer program	tftp(1)
trivial file transfer protocol server	tftpd(1M)
troff/nroff files, check	checknr(1)
troff/nroff, tbl, and neqn constructs, remove	deroff(1)
true - do nothing and return zero exit status	true(1)
true gamma functions	tgamma(3M)
true/false evaluate condition for	test(1)
trunc() - truncation function	trunc(3M)
truncate() - truncate a file to a specified length	truncate(2)
truncate64() - non-POSIX standard API interfaces to support large files	creat64(2)
truncation functions	trunc(3M)
truncf() - truncation function (float)	trunc(3M)
trunc1() - truncation function (long double)	trunc(3M)
truncq() - truncation function (quad)	trunc(3M)
truncw() - truncation function (extended)	trunc(3M)
trusted system, check if converted	iscomsec(2)
trusted system; device assignment database file for a	devassign(4)
trusted system; manipulate device assignment database entry for a	getdvagent(3)
trusted system; manipulate system default database entry for a	getprdfent(3)
trusted system; system default database file for a	default(4)
trusted systems; get secure password file entry on	getspwnt(3X)
trusted systems; manipulate protected password database entries	getprpwnt(3)
trusted systems; protected password authentication database for	prpwd(4)
trusted systems; security databases for	authcap(4)

Description	Entry Name(Section)
trusted systems; terminal control database file for	ttys(4)
truth value about processor type; provide	machid(1)
tryfrom - utility programs for TCP Wrappers	tryfrom(1)
tsearch() - build and access a binary search tree	tsearch(3C)
tset - terminal-dependent initialization	tset(1)
tsm ; add or remove a printer for use with	tsm.lpadmin(1M)
tsm - Terminal Session Manager	tsm(1)
tsm.command - send commands to the Terminal Session Manager, TSM	tsm.command(1)
tsm.info - get Terminal Session Manager state information	tsm.info(1)
tsm.lpadmin - add or remove a printer for use with tsm	tsm.lpadmin(1M)
TSM; send commands to the Terminal Session Manager,	tsm.command(1)
tsort - topological sort	tsort(1)
trace request, wait for	ttrace_wait(2)
ttrace() - tracing facility for multithreaded processes	ttrace(2)
ttrace_wait - wait for ttrace() request	ttrace_wait(2)
tty	glossary(9)
tty baud rate, set or get	cfspeed(3C)
tty - controlling terminal interface	tty(7)
tty device operating parameters, get or set	tcattribute(3C)
TTY - get the name of the user's terminal or pseudo-terminal	tty(1)
tty line control functions	tcontrol(3C)
TTY port, terminal-type data base for each	ttytype(4)
TTYCONF - file for default terminal control characters	stty(1)
TTYNAME() , ISATTY() - find name of a terminal	ttyname(3C)
ttys and users, indicate last logins of	last(1)
ttys - terminal control database file for trusted systems	ttys(4)
ttyslot() - find the slot in the utmpx() file of the current user	ttyslot(3C)
TTYTYPE - data base of terminal-type for each TTY port	ttytype(4)
TTYTYPE - terminal identification program	ttytype(1)
tunable parameter, get value of kernel	gettune(2)
tunable parameter; set the value of a kernel	settune(2)
tunable parameters, display system	sysdef(1M)
tunable parameters; manage kernel	kctune(1M)
tunable parameters; retrieve detailed information about kernel	tuneinfo2(2)
tunefs - tune up an existing HFS file system	tunefs(1M)
tuneinfo2() - retrieve detailed information about kernel tunable parameters	tuneinfo2(2)
tuning, network	nnd(1M)
turn on/off or display current status of power for cells and I/O chassis	frupower(1M)
turnacct - turn process accounting on or off	acctsh(1M)
tutorial: asynchronous writes	fs_async(5)
tutorial: synchronous writes	fs_async(5)
twalk() - traverse a binary search tree	tsearch(3C)
two files, compare	cmp(1)
two logical volumes; split mirrored LVM logical volume into	lvsplit(1M)
two sorted files, reject/select lines common to	comm(1)
two versions of an SCCS file; compare	sccsdiff(1)
type attribute, get or set	pthread_mutexattr_getpshared(3T)
type control characters, how to	ascii(5)
type, modes, speed, and line discipline; set terminal	getty(1M)
type; classify characters according to	ctype(3C)
type; determine file	file(1)
typeahead() - control checking for typeahead	typeahead(3X)
typeahead, control checking for	typeahead(3X)
types - primitive system data types	types(5)
types; fixed-size integer data	inttypes(5)
typescript of terminal session; make	script(1)
typeset - control leading blanks and parameter handling	ksh(1)
typeset - control leading blanks and parameter handling	sh-posix(1)
tzname() - name of local timezone	ctime(3C)
tzset() - initialize timezone() , daylight() , and tzname() using TZ variable	ctime(3C)
tztab - time zone adjustment table for date and ctime()	tztab(4)

Index

All Volumes

Description	Entry Name(Section)
u370 - is processor an IBM 370?	machid(1)
u3b - is processor a U3B?	machid(1)
u3b10 - is processor a U3B10?	machid(1)
u3b2 - is processor a U3B2?	machid(1)
u3b5 - is processor a U3B5?	machid(1)
U_STACK_TRACE() - produce a trace back of the procedure call stack using the unwind library	U_STACK_TRACE(3X)
uaddr2taddr() - generic transport name-to-address translation	netdir(3N)
ualarm() - set the interval timer	ualarm(2)
uc_access - user context access (ucontext_t)	uc_access(3)
(ucontext_t); user context access	uc_access(3)
udp - Internet user datagram protocol	UDP(7P)
udppublickey - updates the publickey database file and NIS map	udppublickey(1M)
ug_display_width() - get current display width for user and group names	ug_display_width(3C)
ugweb - starts the HP-UX User and Group Account Configuration tool	ugweb(1M)
UID , get name from (obsolete)	getpw(3C)
ul - do underlining on terminal	ul(1)
ulckpword() - control access to /etc/passwd and /etc/shadow files	lckpword(3C)
uld.so - microloader	dld.so(5)
ulimit() - get or set file size limits and break value	ulimit(2)
ulimit - set size or time limits	ksh(1)
ulimit - set size or time limits	sh-posix(1)
ultoa() - convert unsigned long integer to ASCII decimal	ltostr(3C)
ultoa_r() - convert unsigned long integer to ASCII decimal (MT-Safe)	ltostr(3C)
ltostr() - convert unsigned long integer to string	ltostr(3C)
ltostr_r() - convert unsigned long integer to string (MT-Safe)	ltostr(3C)
umask - set access permissions mode mask for file-creation	umask(1)
umask() - set and get file creation (permissions) mask	umask(2)
umask - set permissions mask for creating new files	csh(1)
umask - set permissions mask for creating new files	ksh(1)
umask - set permissions mask for creating new files	sh-posix(1)
umodem - XMODEM-protocol file transfer program	umodem(1)
umount - mount and unmount CDFS file systems	mount_cdfs(1M)
umount() - unmount a file system	umount(2)
umount - unmount CacheFS file systems	mount_cachefs(1M)
umount - unmount CDFS file systems	mount_cdfs(1M)
umount - unmount file systems	mount(1M)
umount - unmount HFS file systems	mount_hfs(1M)
umount - unmount remote NFS resources	mount_nfs(1M)
umount2() - unmount a file system	umount(2)
umountall - unmount multiple file systems	mountall(1M)
unalias - discard specified alias	csh(1)
unalias - discard specified alias	ksh(1)
unalias - discard specified alias	sh-posix(1)
uname - display information about computer system; set node name (system name)	uname(1)
uname() - get information about computer system	uname(2)
uname() system function to return [EOVERFLOW] if values do not fit in fields; causes	uname_eoverflow(5)
uname_eoverflow - causes uname() system function to return [EOVERFLOW] if values do not fit in fields	uname_eoverflow(5)
unavailable for mounting by remote systems; make local resource	unshare(1M)
unbiased exponent functions	ilogb(3M)
unblock one or all threads waiting on a conditional variable	pthread_cond_signal(3T)
uncompact previously compacted Huffman coded files (see pack)	compact(1)
uncompact - uncompact Huffman coded files (see pack)	compact(1)
uncompile terminfo data base	untic(1M)
uncompress , compress , zcat - compress or expand data	compress(1)
uncompress a file in a crash dump	cr_uncompress(3)
uncompressdir , compressdir - compress or expand files in a directory	compress(1)
unconfigure, reconfigure, configure installed software	swconfig(1M)
unctrl() - generate printable representation of a character	unctrl(3X)

Description	Entry Name(Section)
underflow mode: getting floating-point	fegetflushzero(3M)
underflow mode: setting floating-point	fesetflushzero(3M)
underlining on terminal, convert underscores to	ul(1)
underlying security mechanisms, allow application to determine which are available	gss_indicate_mechs(3)
underscores, convert to underlining on terminal	ul(1)
undial() - establish an outgoing terminal line connection	dial(3C)
undo a previous get of an SCCS file	unget(1)
undo entries per process, maximum number of System V IPC	semume(5)
undo structures, number of System V IPC system-wide semaphore	semnu(5)
unexpand, expand - expand tabs to spaces, and vice versa	expand(1)
unget - undo a previous get of an SCCS file	unget(1)
unget_wch() - push a character onto the input queue	ungetch(3X)
ungetc() - push character back into input stream	ungetc(3S)
ungetch() - push a character onto the input queue	ungetch(3X)
ungetwc() - push wide character back into input stream	ungetwc(3C)
ungetwc_unlocked() - unlocked version of ungetwc()	ungetwc(3C)
unhash - disable use of internal hash tables	csh(1)
unifdef - remove preprocessor lines	unifdef(1)
uninterpreted file contents, retrieve for ELF files	elf_rawfile(3E)
Uninterruptible Power System monitor configuration file	ups_conf(4)
Uninterruptible Power System (UPS), monitor daemon	ups_mond(1M)
uniq - report adjacent repeated lines in a file	uniq(1)
unique file name; make	mktemp(3C)
unistd - standard structures and symbolic constants	unistd(5)
unistd.h - standard structures and symbolic constants	unistd(5)
unit data error indication (X/OPEN TLI-XTI)	t_revuderr(3)
units - convert units of measure	units(1)
units of measure, convert	units(1)
Universal (Greenwich mean) and local time, difference between	timezone(5)
UNIX - local communication domain protocol	UNIX(7P)
UNIX standards behavior on HP-UX	standards(5)
UNIX system to UNIX system command execution	uux(1)
UNIX system to UNIX system copy	uucp(1)
UNIX system to UNIX system file copy, public	uuto(1)
UNIX system, terminal emulator; call another	cu(1)
unlink a message queue	mq_unlink(2)
unlink a named semaphore	sem_unlink(2)
unlink a shared memory object	shm_unlink(2)
unlink - execute unlink() system call without error checking	link(1M)
unlink - remove directory entry; delete file	unlink(2)
unlink() system calls without error checking; execute link() and	link(1M)
unload a kernel module on demand	moduload(2)
unlock a mutex	pthread_mutex_unlock(3T)
unlock a POSIX semaphore	sem_post(2)
unlock a read-write lock	pthread_rwlock_unlock(3T)
unlock a semaphore	msem_unlock(2)
unlock a STREAMS pty master/slave pair	unlockpt(3C)
unlock access to /etc/passwd and /etc/shadow files	lckpwwdf(3C)
unlock memory segment	munlock(2)
unlock process virtual address space	munlockall(2)
unlock stable complex profile or cancel pending changes to complex or partition configuration data	parunlock(1M)
unlockable_mem - OBSOLETE kernel tunable parameter	unlockable_mem(5)
unlockpt() - unlock a STREAMS pty master and slave pair	unlockpt(3C)
unmap a mapped region	munmap(2)
unmount a file system	umount(2)
unmount CacheFS file systems	mount_cachefs(1M)
unmount CDFS file systems	mount_cdfs(1M)
unmount file systems	mount(1M)
unmount HFS file systems; mount and	mount_hfs(1M)
unmount multiple file systems	mountall(1M)

Index

All Volumes

Description	Entry Name(Section)
unmount remote NFS resources; mount and	mount_nfs(1M)
unpack - expand Huffman-coded files	pack(1)
unprintable characters in a file, make visible or invisible	vis(1)
unregister or register depots and roots	swreg(1M)
unset - remove definition/setting of flags and arguments	csh(1)
unset - remove definition/setting of options and arguments	ksh(1)
unset - remove definition/setting of options and arguments	sh-posix(1)
unsetenv () - deletes an environment variable	unsetenv(3C)
unsetenv - remove variable from environment	csh(1)
unsetenv () - removes an environment variable	unsetenv(3C)
unshare - make local resource unavailable for mounting by remote systems	unshare(1M)
unshare multiple resources; share,	shareall(1M)
unshare_nfs - make local NFS file systems unavailable for mounting by remote systems	unshare_nfs(1M)
unshareall - share, unshare multiple resources	shareall(1M)
unsigned long integer to string, convert	ltostr(3C)
utic - terminfo de-compiler	utic(1M)
until - execute commands until expression is nonzero	ksh(1)
until - execute commands until expression is nonzero	sh-posix(1)
untouchwin () - window refresh control functions	is_linetouched(3X)
unwind environment, create and initialize	uwx_init(3X)
Unwind Express Library	uwx(3X)
unwind library data structure; allocate and deallocate	_UNW_createContextForSelf(3X)
unwind library data structure; manipulate values in	_UNW_currentContext(3X)
unwind library data structure; query values in	_UNW_getGR(3X)
unwind library, produce a trace back of the procedure call stack using the	U_STACK_TRACE(3X)
unwind - overview of stack unwind library entry points and convenience macros	unwind(5)
unwind.h - overview of stack unwind library entry points and convenience macros	unwind(5)
unwritten system buffers, flush to disk	sync(1M)
unwritten system buffers, periodically flush to disk	syncer(1M)
update access, modification, and/or change times of file	touch(1)
update an ELF descriptor	elf_update(3E)
update backup LVM volume group configuration file	vgcgbbackup(1M)
update boot programs from disk	mkboot(1M)
update disk	sync(2)
update execution time limit	rtsched(2)
update file access and modification times	utime(2)
update information about top processes on system	top(1)
update or check the /etc/shadow file; install,	pwconv(1M)
update routines for user-accounting database maintained by utmpd ; access and	getuts(3C)
update status, line, functions	redrawwin(3X)
update table if entry missing after search	lsearch(3C)
update the Common Error Repository (CER) with error metadata	cerupdate(1)
update user password in Network Information Service	yppasswd(3N)
update utility; Dynamic DNS	nsupdate(1)
update, and regenerate groups of programs; maintain,	make(1)
update-ux - updates the HP-UX operating system	update-ux(1M)
updatewdb () - write records into new wtmps and btmps database	bwtmps(3C)
updaters - configuration file for NIS updating	updaters(1M)
updates the HP-UX operating system	update-ux(1M)
updates the publickey database file and NIS map	udpublickey(1M)
updating, configuration file for NIS	updaters(1M)
upper-case, translate characters to	conv(3C)
uppercase, translate wide characters to	wconv(3C)
UPS monitor configuration file	ups_conf(4)
UPS, monitor daemon	ups_mond(1M)
ups_conf - HP PowerTrust UPS monitor configuration file	ups_conf(4)
ups_conf - Uninterruptible Power System (UPS) monitor configuration file	ups_conf(4)
ups_mond - HP PowerTrust monitor daemon	ups_mond(1M)
ups_mond - HP PowerTrust Uninterruptible Power System monitor daemon	ups_mond(1M)
ups_mond - Uninterruptible Power System monitor daemon	ups_mond(1M)

Description	Entry Name(Section)
upshifting	glossary(9)
uptime - show how long system has been up	uptime(1)
urandom - strong random number generator	random(7)
usage, summarize disk	du(1)
use in LVM volume group; create physical volume for	pvccreate(1M)
use of device's write cache in the SCSI subsystem (OBSOLETE); enable and disable	default_disk_ir(5)
use, disable, of certain terminal capabilities	filter(3X)
use_env() - specify source of screen size information	use_env(3X)
used by finger command; change user information	chfn(1)
used by sendfile, maximum number of Buffer Cache Pages	sendfile_max(5)
used by the Internal Terminal Emulator, number of scrollable lines	scroll_lines(5)
user accounting database daemon	utmpd(1M)
user accounting information file	utmpx(4)
user accounting, daily accounting shell procedure	runacct(1M)
user alias database	elm(1)
user alias text file	elm(1)
user alias: install new elm aliases	newalias(1)
user aliases, elm , verify and display	elmalias(1)
user and group account configuration tool; starts the HP-UX	ugweb(1M)
user and group id mapping daemon; NFS	nfsmapid(1M)
user and group IDs and names; print	id(1)
user and group name enablement and display; long	lugadmin(1M)
user and group names; get current display width for	ug_display_width(3C)
user authorization; PAM module that provides	pam_authz(5)
user configuration file for pluggable authentication modules	pam_user.conf(4)
user context access (ucontext_t)	uc_access(3)
user context; DEPRECATED; get and set current	getcontext(2)
user contexts; DEPRECATED; manipulate	makecontext(2)
user credentials for an authentication service, modify and delete	pam_setcred(3)
user database for per-user information	userdb(4)
user database, /var/adm/userdb ; display information residing in the	userdbget(1M)
user database, /var/adm/userdb ; modify information in the	userdbset(1M)
user database, /var/adm/userdb , read, write or delete information in the	userdb_read(3)
user database, /var/adm/userdb , verify or fix information in the	userdbck(1M)
user datagram protocol, Internet	UDP(7P)
user environment variables	environ(5)
user format date and time; convert	getdate(3C)
user group access and identification file, grp.h	group(4)
user ID	glossary(9)
user ID, get	getresuid(3)
user ID; print effective current	whoami(1)
user ID; set	setuid(2)
user IDs, set real and effective user IDs	setreuid(2)
user IDs; set effective	seteuid(2)
user information lookup program	finger(1)
user information server, remote	fingerd(1M)
user information used by finger command; change	chfn(1)
user interface for gated; operational	gdc(1M)
user interface for Routing Administration Manager (RAMD)	rdc(1M)
user interface to the TELNET protocol	telnet(1)
user login data; display	listusers(1)
user login data; display system and	logins(1M)
user login from the system; delete a	userdel(1M)
user login information	wtmps(4)
user login name, get character-string representation of	cuserid(3S)
user login name, obtain	logname(3C)
user login on the system; modify a	usermod(1M)
user login record format	utmp(4)
user login to the system; add a new	useradd(1M)
user name directory service, Internet	whois(1)
user name, PAM routine to retrieve	pam_get_user(3)

Index

All Volumes

Description	Entry Name(Section)
user name: in elm aliases	newaliases(1)
user of a particular TCP connection, identify	idlookup(1)
user or group IDs, set real, effective, and/or saved	setresuid(2)
user password in Network Information Service, update	yppasswd(3N)
user policy definition service module, PAM	pam_updbe(5)
user process, maximum size (in bytes) of the RSE stack for any	maxrsessiz(5)
user process; maximum size (in bytes) of the data segment for any	maxdsiz(5)
user process; maximum size (in bytes) of the stack for any	maxssiz(5)
user process; maximum size (in bytes) of the text segment for any	maxtsiz(5)
user processes from /etc/services.window ; extract window IDs of	getmemwindow(1M)
user processes per user, limits the maximum number of	maxuprc(5)
user processes using a file or file structure	fuser(1M)
user selectable page size; maximum (in kilobytes) of	vps_chatr_ceiling(5)
user shells, get legal	getusershell(3C)
user's effective access rights to a file, get a	getaccess(2)
user's Kerberos password; change a	kpasswd(1)
user's login environment, shell script to set up	profile(4)
user's terminal or pseudo-terminal, get name of	tty(1)
user, ask for user response for SD-UX	swask(1M)
user, change login name to another	su(1)
user, change user's secure RPC key	chkey(1)
user, communicate interactively with another	write(1)
user, current, find the slot in the utmpx() file of the	ttyslot(3C)
user, get name of user logged in on this terminal	getlogin(3C)
user, limits the maximum number of user processes per	maxuprc(5)
user-accounting database	utmps(4)
user-accounting database maintained by utmpd ; access and update routines for	getuts(3C)
user2netname() - library routines for secure remote procedure calls	secure_rpc(3N)
user: print list of current system users	who(1)
useradd - add a new user login to the system	useradd(1M)
userdb - user database for per-user information	userdb(4)
userdb_delete() - delete information in the user database, /var/adm/userdb	userdb_read(3)
userdb_read() - read information in the user database, /var/adm/userdb	userdb_read(3)
userdb_write() - write information in the user database, /var/adm/userdb	userdb_read(3)
userdbck - verify or fix information in the user database, /var/adm/userdb	userdbck(1M)
userdbget - display information residing in the user database, /var/adm/userdb	userdbget(1M)
userdbset - modify information in the user database, /var/adm/userdb	userdbset(1M)
userdel - delete a user login from the system	userdel(1M)
usermod - list of home directory names	usermod(4)
usermod - list of home directory names	usermod(4)
usermod - modify a user login on the system	usermod(1M)
username server, network	rusersd(1M)
users and processes, list current	whodo(1M)
users and ttys, indicate last logins of	last(1)
users - compact list of users currently on the system	users(1)
users currently on the system, list	users(1)
users edit files that are under access control; let authorized	privedit(1M)
users for each class	ftpcount(1)
users on remote machines, return information about	rnusers(3N)
users over a network, write to all	rwall(1M)
users to audit; select	audusr(1M)
users, notify of new mail in mailboxes	newmail(1)
users, remote, authorizing access on local host	hosts.equiv(4)
users: list current users and what they are doing	whodo(1M)
userstat - check status of local user accounts	userstat(1M)
using Huffman code; compress and expand files	pack(1)
using the unwind library, produce a trace back of the procedure call stack	U_STACK_TRACE(3X)
usleep() - suspend execution for an interval	usleep(2)

Description	Entry Name(Section)
ustat () - get mounted file system statistics	ustat (2)
UTC (Coordinated Universal Time)	glossary (9)
utility	glossary (9)
Utility Command for TACHYON TL, TACHYON XL2, FCD Driver-Based and FC/GigE Combo Fibre Channel Host Bus Adapters; Fibre Channel Mass Storage	femsutil (1M)
utility for psfontpf; model script configuration	psmsgen (1M)
utility options, parse	getopts (1)
utility program for SCCS commands	sccs (1)
utility; DNS lookup	host (1)
utility; Dynamic DNS update	nsupdate (1)
utility; Kerberos keytab file maintenance	ktutil (1)
utility; name server control	rndc (1)
utility; SCSI management and diagnostic	scsimgr (1M)
utime () - set or update file access and modification times	utime (2)
utimes () - set file access and modification times	utimes (2)
utmp file entry; access	getut (3C)
utmp () file of the current user, find the slot in the	ttyslot (3C)
utmp record; write and include reason for writing	acct (1M)
utmp - user login record format	utmp (4)
utmp2wtmp - overview of accounting and miscellaneous accounting commands	acct (1M)
utmpd , get login name of user from	getlogin (3C)
utmpd ; access and update routines for user-accounting database maintained by	getuts (3C)
utmpd - user accounting database daemon	utmpd (1M)
utmpname () - change name of utmp file being examined	getut (3C)
utmpname_r () - change name of utmp file being examined	getut (3C)
utmps file	login (1)
utmps - user-accounting database	utmps (4)
utmpx file entry; access	getutx (3C)
utmpx - user accounting information file	utmpx (4)
uuccheck - check the uucp directories and permissions file	uuccheck (1M)
uucico - transfer files for the uucp system	uucico (1M)
uuclean - uucp spool directory clean-up	uuclean (1M)
uucleanup - uucp spool directory clean-up	uucleanup (1M)
uucp names of known systems; list	uucp (1)
uucp or uux command requests from remote, execute on local system	uuxqt (1M)
UUCP over TCP/IP server daemon	uucpd (1M)
uucp spool directory clean-up	uuclean (1M)
uucp spool directory clean-up	uucleanup (1M)
uucp status inquiry and job control	uustat (1)
uucp subnetwork activity, monitor	uusub (1M)
uucp system; transfer files for the	uucico (1M)
uucp transactions grouped by transaction; list spooled	uuls (1M)
uucp transactions; query log file of	uucp (1)
uucp - UNIX system to UNIX system copy	uucp (1)
uucp: check the uucp directories and permissions file	uuccheck (1M)
uucp: schedule uucp transport files	uusched (1M)
uucp: set terminal type, modes, speed and line discipline for 2-way line	uugetty (1M)
uucp: show snapshot of the UUCP system	uusnap (1M)
uucp: uucleanup - uucp spool directory clean-up	uuclean (1M)
uucpd - server for supporting UUCP over TCP/IP networks	uucpd (1M)
uucpd - UUCP over TCP/IP server daemon	uucpd (1M)
uudecode - decode a file encoded by uuencode	uudecode (1)
uuencode - encode a binary file for transmission by mailer	uuencode (1)
uuencode file; format of an encoded	uuencode (4)
uuencode - format of an encoded uuencode file	uuencode (4)
uugetty - set terminal type, modes, speed and line discipline for 2-way line	uugetty (1M)
uulog - query log file of uucp transactions	uucp (1)
uuls - list spooled uucp transactions grouped by transaction	uuls (1M)
uname - list uucp names of known systems	uucp (1)
uupath , mtuupath - access and manage the pathalias database	uupath (1)
uupick - accept or reject files sent by uuto	uuto (1)

Index

All Volumes

Description	Entry Name(Section)
uusched - schedule uucp transport files	uusched (1M)
uusnap output, sort and embellish	uusnaps (1M)
uusnap - show snapshot of the UUCP system	uusnap (1M)
uusnaps - sort and embellish uusnap output	uusnaps (1M)
uustat - uucp status inquiry and job control	uustat (1)
uusub - monitor uucp subnetwork activity	uusub (1M)
uuto - public UNIX system to UNIX system file copy	uuto (1)
uutry - test for successful login to remote system	uucp (1)
uux or uucp command requests from remote, execute on local system	uuxqt (1M)
uux - UNIX system to UNIX system command execution	uux (1)
uuxqt - execute remote uucp or uux command requests on local system	uuxqt (1M)
uwx () - Unwind Express Library	uwx (3X)
uwx_add_to_esp - backing store pointer arithmetic	uwx_add_to_esp (3X)
uwx_find_source_info - obtain source information from ELF files	uwx_find_source_info (3X)
uwx_find_symbol - obtain symbolic information from ELF files	uwx_find_symbol (3X)
uwx_free - free memory used by an unwind environment	uwx_free (3X)
uwx_get_abi_context_code - return ABI and context code from current context	uwx_get_abi_context_code (3X)
uwx_get_funcstart - return start address of current function	uwx_get_funcstart (3X)
uwx_get_module_info - return load module information for current context	uwx_get_module_info (3X)
uwx_get_nat - read a NaT bit from current frame's context	uwx_get_nat (3X)
uwx_get_reg - read a register from current frame's context	uwx_get_reg (3X)
uwx_get_source_info - return source information for current frame	uwx_get_source_info (3X)
uwx_get_sym_info - return symbolic information for current frame	uwx_get_sym_info (3X)
uwx_init - create and initialize an unwind environment	uwx_init (3X)
uwx_init_context - create and initialize an unwind environment	uwx_init_context (3X)
uwx_register_alloc_cb - register custom allocate and free callbacks	uwx_register_alloc_cb (3X)
uwx_register_callbacks - register callback routines for stack unwind	uwx_register_callbacks (3X)
uwx_release_symbol_cache - free memory used by the symbol cache	uwx_release_symbol_cache (3X)
uwx_self_do_context_frame - reinitialize the context at a signal frame	uwx_self_do_context_frame (3X)
uwx_self_free_info - free memory used by the callback info structure	uwx_self_free_info (3X)
uwx_self_init_context - initialize the current context for self-unwinding	uwx_self_init_context (3X)
uwx_self_init_info - create and initialize a callback info structure for self-unwinding	uwx_self_init_info (3X)
uwx_set_nofr - disable tracking of floating-point registers	uwx_set_nofr (3X)
uwx_set_remote - create and initialize an unwind environment	uwx_set_remote (3X)
uwx_step - step one frame	uwx_step (3X)
uwx_step_inline - step over one inline call	uwx_step_inline (3X)
ux2dos - convert ASCII file format between HP-UX and DOS formats	dos2ux (1)
V IPC message queue, maximum number of bytes on a single System	msgmnb (5)
V IPC message queues (IDs) allowed, maximum number of system-wide System	msgmni (5)
V IPC messages in the system at any time, maximum number of System	msgtql (5)
vacation - return "I am not here" indication	vacation (1)
val - validate an SCCS file	val (1)
validate an SCCS file	val (1)
validate whether physical page number was dumped	cr_isaddr (3)
validates GSS-API tokens for kernel RPC; generates and	gssd (1M)
validation procedures; perform PAM account	pam_acct_mgmt (3)
validity checking tool; zone	named-checkzone (1)
validity; check memory region	mvalid (3)
valloc () - allocate space on boundary aligned to sysconf value	malloc (3C)
value changes per System V IPC semop () call, maximum cumulative	semaem (5)
value of a per-process timer, get	gettimer (3C)
value of a symbolic link, read	readlink (2)
value of any single System V IPC semaphore, maximum	semvmx (5)
value of process interval timer, set or get	getitimer (2)
value of system-wide clock, get current	getclock (3C)
value of system-wide clock, set	setclock (3C)
value, change or add to environment	putenv (3C)
value, change or add to environment	setenv (3C)
value, get or set file size limits and break	ulimit (2)

Description	Entry Name(Section)
value, return integer absolute	abs(3C)
values do not fit in fields; causes uname() system function to return [E_OVERFLOW] if	uname_eoverflow(5)
values in a Network Information Service map, print all	ypcat(1)
values in unwind library data structure; manipulate	_UNW_currentContext(3X)
values in unwind library data structure; query	_UNW_getGR(3X)
values - machine-dependent values	values(5)
values of selected keys in Network Information Service map, print the	ypmatch(1)
values to support for POSIX.1b realtime applications, number of priority	rtsched_numpri(5)
values, convert between host and network byte order	byteorder(3N)
values, get POSIX configuration	getconf(1)
values; machine-dependent	values(5)
varargs argument list; print formatted output of a	vprintf(3S)
varargs argument, formatted input conversion to a	vscanf(3S)
varargs.h - macros for handling variable argument list	varargs(5)
variable argument list macros	varargs(5)
variable argument list macros	stdarg(5)
variable, environment, search environment list for value of	getenv(3C)
variables in stable storage; display and modify boot	setboot(1M)
variables, configurable path name, get	pathconf(2)
variables, environment, print value of	printenv(1)
variables, user environment	environ(5)
variables; manipulate event	EvmVarGet(3)
vax - is processor a VAX?	machid(1)
vc - version control	vc(1)
vector; get option letter from argument	getopt(3C)
vedit - beginner's screen-oriented text editor	vi(1)
verbose description of current terminal, get	longname(3X)
verification tool, STREAMS	strvf(1M)
verifier, file system quota consistency	quotacheck(1M)
verify elm user and system aliases	elmalias(1)
verify integrity of crash dump	cr_verify(3)
verify LAN connectivity with link-level loopback	linkloop(1M)
verify or fix information in the user database, /var/adm/userdb	userdbck(1M)
verify path names of all FTP configuration files	ckconfig(1)
verify program assertion	assert(3X)
verify software products	swverify(1M)
verify the syntax of the Role-Based Access Control (RBAC) database files	rbacdbchk(1M)
Version 6/PWB compatibility; terminal interface for	sttyv6(7)
Version 6; Internet Protocol	IPv6(7P)
version control	vc(1)
version level of operating system, display	uname(1)
version numbers of Kerberos principals; print key	kvno(1)
version of an SCCS file; get	get(1)
version of EVM status code; format text	EvmStatusTextGet(3)
version, get information on mechanisms and RPC	rpc_gss_get_mechanisms(3N)
versions of an SCCS file; compare two	sccsdiff(1)
versions; coordinate ELF library and application	elf_version(3E)
vfork() - spawn new process (use fork() instead)	vfork(2)
vfprintf() - print formatted output of a varargs argument list	vprintf(3S)
vfprintf() - formatted input conversion to a varargs argument	vscanf(3S)
vfsmount() - mount a file system	vfsmount(2)
vwprintf() - print formatted output to a file	vwprintf(3C)
vscanf() - convert formatted wide-character input of a stdarg argument list	vscanf(3S)
vgcgbgbackup - create LVM volume group configuration backup file	vgcgbgbackup(1M)
vgcgrestore - restore volume group configuration	vgcgrestore(1M)
vgchange - set LVM volume group availability	vgchange(1M)
vgchgid - modify the Volume Group ID (VGID) on a given set of physical devices	vgchgid(1M)
vgcreate - create LVM volume group	vgcreate(1M)
vgdisplay - display information about LVM volume groups	vgdisplay(1M)
vgexport - export an LVM volume group and its associated logical volumes	vgexport(1M)

Index

All Volumes

Description	Entry Name(Section)
vgextend - extend an LVM volume group by adding physical volumes	vgextend(1M)
VGID, modify the Volume Group ID (VGID) on a given set of physical devices	vgchgid(1M)
vgimport - import an LVM volume group onto the system	vgimport(1M)
vgmodify - handle physical volume size changes and modify configuration parameters of an existing LVM volume group	vgmodify(1M)
vgreduce - remove physical volumes from an LVM volume group	vgreduce(1M)
vgremove - remove LVM volume group definition from the system	vgremove(1M)
vgscan - scan physical volumes for LVM volume groups	vgscan(1M)
vgsync - synchronize stale logical volume mirrors in LVM volume groups	vgsync(1M)
vhardlinks - checks the consistency of compartment rules for files with multiple hardlinks	vhardlinks(1M)
vi edit on the password file	vipw(1M)
vi editing mode	sh-posix(1)
vi - extended screen-oriented text editor	vi(1)
vid_attr() - output attributes to terminal	vidattr(3X)
vid_puts() - output attributes to terminal	vidattr(3X)
vidattr() - output attributes to terminal	vidattr(3X)
video attributes, terminal, get supported	termattrs(3X)
vidputs() - output attributes to terminal	vidattr(3X)
view and delete events; enables you to	evweb_eventviewer(1)
view or modify Access Control Lists	swacl(1M)
view - read-only screen-oriented text editor	vi(1)
view, create, modify, and delete event subscriptions; enables you to	evweb_subscribe(1)
viewer tool (a Web interface); start the HP-UX hardware event	slweb(1M)
viewing, saving SAM logfile tool	samlog_viewer(1)
viewing; file perusal filter for screen	more(1)
vipw - edit the password file	vipw(1M)
virtual circuit, X.25 switched, clear	clrsvc(1M)
virtual hosting configuration specification file	ftpservers(4)
virtual LANs (VLANs)	lanadmin_vlan(1M)
virtual local area network	VLAN(7)
virtual memory statistics, report	vmstat(1)
virtual memory subsystem, get information about the	pstat(2)
virtual terminal requests from other systems, respond to	vtdaemon(1M)
vis - make unprintable and non-ASCII characters in a file visible	vis(1)
(visual) text editor; screen-oriented	vi(1)
VLAN interface; network interface management command for	nwmgr_vlan(1M)
VLAN - virtual local area network	VLAN(7)
VLANs; virtual LANs	lanadmin_vlan(1M)
vline() - draw lines from single-byte characters and renditions	hline(3X)
vline_set() - draw lines from complex characters and renditions	hline_set(3X)
vmstat - report virtual memory statistics	vmstat(1)
volcopy - copy a file system with label checking	volcopy(1M)
volcopy - copy file systems with label checking	volcopy_hfs(1M)
volcopy - copy HFS file system with label checking	volcopy_hfs(1M)
volcopy_hfs - copy HFS file system with label checking	volcopy_hfs(1M)
volume for use in LVM volume group; create physical	pvcreate(1M)
volume group availability; set LVM	vgchange(1M)
volume group configuration, restore	vgfgrestore(1M)
volume group definition (LVM), remove from the system	vgremove(1M)
Volume Group ID (VGID), modify, on a given set of physical devices	vgchgid(1M)
volume group information file, LVM physical	lvmpvg(4)
volume group (LVM), change characteristics and access path of physical volume in	pvchange(1M)
volume group (LVM), check or repair a physical volume in	pvck(1M)
volume group (LVM), create logical volume in	lvcreate(1M)
volume group (LVM), extend by adding physical volumes	vgextend(1M)
volume group (LVM), remove logical volumes from	lvremove(1M)
volume group (LVM), remove physical volume	pvremove(1M)
volume group (LVM); create	vgcreate(1M)
volume group (LVM) configuration backup file, create or update	vgcfgbackup(1M)
volume group; create physical volume for use in LVM	pvcreate(1M)

Description	Entry Name(Section)
volume group; handle physical volume size changes and modify configuration parameters of an existing LVM	vgmodify(1M)
volume group; remove physical volumes from an LVM	vgreduce(1M)
volume groups (LVM), synchronize stale logical volume mirrors	vgsync(1M)
volume groups (LVM); display information about	vgdisplay(1M)
volume groups (LVM); scan physical volumes for	vgscan(1M)
volume header on LIF file; write	lifinit(1)
volume in LVM volume group, create logical	lvcreate(1M)
volume into two logical volumes; split mirrored LVM logical	lvsplit(1M)
volume mirrors in LVM volume groups, synchronize stale logical	vgsync(1M)
volume number	glossary(9)
volume to be root, boot, primary swap, or dump volume; prepare LVM logical	lvlnboot(1M)
volume, get information for a logical	pstat(2)
volume; prepare LVM logical volume to be root, boot, primary swap, or dump	lvlnboot(1M)
volumes for LVM volume groups; scan physical	vgscan(1M)
volumes from LVM volume group, remove logical	lvremove(1M)
volumes; split mirrored LVM logical volume into two logical	lvsplit(1M)
vpfmt() - display message in standard format	vpfmt(3C)
vprintf() - print formatted output of a varargs argument list	vprintf(3S)
vps_ceiling - maximum (in kilobytes) of system-selectable page size	vps_ceiling(5)
vps_chattr_ceiling - maximum (in kilobytes) of user selectable page size	vps_chattr_ceiling(5)
vps_pagesize - minimum (in kilobytes) of system-selected page size	vps_pagesize(5)
vscanf() - formatted input conversion to a varargs argument	vscanf(3S)
vsprintf() - print formatted output of a varargs argument list	vprintf(3S)
vprintf() - print formatted output of a varargs argument list	vprintf(3S)
vsscanf() - formatted input conversion to a varargs argument	vscanf(3S)
vsprintf() - print formatted output to a string	vwprintf(3C)
vwscanf() - convert formatted wide-character input of a stdarg argument list	vwscanf(3S)
vt - log in on another system over LAN	vt(1)
vt requests from other systems, respond to	vtdaemon(1M)
VT100 terminal	swinstall(1M)
VT100 terminal	swremove(1M)
VT320 terminal	swinstall(1M)
VT320 terminal	swremove(1M)
vtdaemon - respond to vt requests	vtdaemon(1M)
VUE (OBSOLETE); audio tools available through HP	Audio(5)
vw_printw() - print formatted output in a window	vw_printw(3X)
vw_scanw() - convert formatted input from a window	vwprintf(3X)
vwprintf() - print formatted output to standard output	vwprintf(3C)
vwprintw() - print formatted output in a window	vwprintw(3X)
vwscanf() - convert formatted wide-character input of a stdarg argument list	vwscanf(3S)
vwscanw() - convert formatted input from a window	vwprintf(3X)
VxFS file system; policy for flush behind requests from	fcache_fb_policy(5)
w - show how long system has been up	uptime(1)
wadd_wch() - add a complex character and rendition to a window	add_wch(3X)
wadd_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
wadd_wchstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
waddch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
waddchnstr() - add length limited string of single-byte characters and renditions to a window	addchnstr(3X)
waddchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
waddnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
waddnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
waddstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
waddwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
wait for a signal	sigsuspend(2)
wait for asynchronous I/O completion	aio_reap(2)
wait for asynchronous I/O completion	aio_suspend(2)
wait for child process to change state	wait3(2)
wait for child process to change state	wait4(2)

Index

All Volumes

Description	Entry Name(Section)
wait for child process to change state	waitid(2)
wait for child process to stop or terminate	wait(2)
wait for interrupt, atomically release blocked signals and	sigpause(3C)
wait for multiple asynchronous I/O requests	aio_reap(2)
wait for the termination of a specified thread	pthread_join(3T)
wait for trace request	ttrace_wait(2)
wait or timed wait on a condition variable	pthread_cond_wait(3T)
wait - wait for background processes	csh(1)
wait - wait for background processes to complete	wait(1)
wait - wait for child process	ksh(1)
wait - wait for child process	sh-posix(1)
wait () - wait for child process to stop or terminate	wait(2)
wait3 () - wait for child process to change state	wait3(2)
wait4 () - wait for child process to change state	wait3(2)
wait4 () - wait for child process to change state	wait4(2)
waitid () - wait for child process to change state	waitid(2)
waitpid () - wait for child process to stop or terminate	wait(2)
walk a file tree, executing a function	ftw(3C)
wall - write message to all users	wall(1M)
watchdog timer expires; sets action taken if IPMI	ipmi_watchdog_action(5)
wattr_get () - window attribute control functions	attr_get(3X)
wattr_off () - window attribute control functions	attr_get(3X)
wattr_on () - window attribute control functions	attr_get(3X)
wattr_set () - window attribute control functions	attr_get(3X)
wattroff () - restricted window attribute control functions	attroff(3X)
wattron () - restricted window attribute control functions	attroff(3X)
wattrset () - restricted window attribute control functions	attroff(3X)
wc - count words, lines, and bytes or characters in a file	wc(1)
wcgat () - change renditions of characters in a window	chgat(3X)
wclear () - clear a window	clear(3X)
wclrtobot () - clear from cursor to end of window	clrtobot(3X)
wclrtoeol () - clear from cursor to end of line	clrtoeol(3X)
wcolor_set () - window attribute control functions	attr_get(3X)
wconv () - translate wide characters	wconv(3C)
wcrtomb () - convert a wide-character code to a character (restartable)	wcrtomb(3C)
wcscat () - append wide string 2 to wide string 1	wcstring(3C)
wcchr () - get pointer to wide character in wide string	wcstring(3C)
wcscmp () - compare two wide strings	wcstring(3C)
wcscoll () - process wide string of text tokens	wcstring(3C)
wcscpy () - copy wide string 2 to wide string 1	wcstring(3C)
wcscspn () - find length of match wide substrings	wcstring(3C)
wcsftime () - convert date and time to wide-character string	wcsftime(3C)
wcslen () - determine length of wide string	wcstring(3C)
wcsncat () - append wide string 2 to wide string 1	wcstring(3C)
wcsncmp () - compare two wide strings	wcstring(3C)
wcsncpy () - copy wide string 2 to wide string 1	wcstring(3C)
wcspbrk () - find occurrence of wide character from wide string 2 in wide string 1	wcstring(3C)
wcsrchr () - get pointer to wide character in wide string	wcstring(3C)
wcsrtombs () - convert a wide-character string to a character string	wcsrtombs(3C)
wcsspn () - find length of matching wide substrings	wcstring(3C)
wcssstr () - locate first occurrence of wide-character string	wcstring(3C)
wcstod () - convert wide character string to double-precision number	wcstod(3C)
wcstof () - convert wide character string to float representation	wcstof(3C)
wcstoimax () - convert wide character string to long integer	wcstoimax(3C)
wcstok () - process wide string of text tokens	wcstring(3C)
wcstol () - convert wide character string to long integer	wcstol(3C)
wcstold () - convert wide character string to long double representation	wcstod(3C)
wcstoll () - convert wide character string to long long integer	wcstol(3C)
wcstombs () - convert sequence of codes corresponding to multibyte characters	multibyte(3C)
wcstoul () - convert wide character string to long integer	wcstol(3C)
wcstoull () - convert wide character string to long long integer	wcstol(3C)

Description	Entry Name(Section)
wcstoumax() - convert wide character string to long integer	wcstoumax(3C)
wcswcs() - process wide string of text tokens	wcswcs(3C)
wcswidth() - return number of columns required for wide character	wcswidth(3C)
wcsxfrm() - process wide string of text tokens	wcsxfrm(3C)
wctob() - conversion between wide character and single-byte	wctob(3C)
wctomb() - number of bytes needed to represent multibyte character	wctomb(3C)
wctrans() - define character mapping	wctrans(3C)
wctype() - classify wide characters	wctype(3C)
wcursyncup() - synchronize a window with its parents or children	wcursyncup(3C)
wcwidth() - return number of columns required for wide character	wcwidth(3C)
wdelch() - delete character from a window	wdelch(3X)
wdeleteln() , deleteln() - delete lines in window	wdeleteln(3X)
Web interface; start the HP-UX Peripheral Device tool, part of the SMH	pdweb(1M)
wecho_wchar() - write a complex character and immediately refresh the window	wecho_wchar(3X)
wechochar() - echo single-byte character and rendition to a window and refresh	wechochar(3X)
werase() - clear a window	werase(3X)
wget_wch() - get a wide character from a terminal	wget_wch(3X)
wget_wstr() - get an array of wide characters and function key codes from a terminal	wget_wstr(3X)
wgetch() - get a single-byte character from the terminal	wgetch(3X)
wgetn_wstr() - get an array of wide characters and function key codes from a terminal	wgetn_wstr(3X)
wgetnstr() - get a multi-byte character length limited string from the terminal	wgetnstr(3X)
wgetstr() - get a multi-byte character string from the terminal	wgetstr(3X)
what - get SCCS identification information	what(1)
whatis files for online manpages; create cat and	catman(1M)
whence - define interpretation of name as a command	whence(1)
whence - define interpretation of name as a command	sh-posix(1)
whereis - locate source, binary, and/or manual files for program	whereis(1)
which - locate a program file including aliases and paths	which(1)
while - execute commands while expression is non-zero	while(1)
while - execute commands while expression is nonzero	ksh(1)
while - execute commands while expression is nonzero	sh-posix(1)
whitespace	whitespace(9)
whline() - draw lines from single-byte characters and renditions	whline(3X)
whline_set() - draw lines from complex characters and renditions	whline_set(3X)
who is logged in on local machines, show	rwho(1)
who is logged in on local network machines, determine	users(1)
who is my mail from?	from(1)
who - who is using the system	who(1)
whoami - print effective current user ID	whoami(1)
whodo - which users are doing what	whodo(1M)
whois - Internet user name directory service	whois(1)
wide character back into input stream, push	ungetwc(3C)
wide character from a stream file; get	getwc(3C)
wide character string operations	wcstring(3C)
wide character string to a double-precision number; convert a	wcstod(3C)
wide character string to long integer, convert	wcstoumax(3C)
wide character string to long integer; convert	wcstol(3C)
wide character, generate printable representation of	wunctrl(3X)
wide character, get from a terminal	get_wch(3X)
wide character, put on a stream	putwc(3C)
wide character; conversion between single-byte and	btowc(3C)
wide characters and function key codes from a terminal; get an array of	getn_wstr(3X)
wide characters, string of, input from a window	innwstr(3X)
wide characters, translate to uppercase or lowercase	wconv(3C)
wide characters; classify	wctype(3C)
wide-character code to a character (restartable); convert a	wcrtomb(3C)
wide-character code; convert a character to	mbrtowc(3C)
wide-character input of a stdarg argument list; convert formatted	vwscanf(3S)
wide-character input; convert formatted	fwscanf(3C)
wide-character output; print formatted	fwprintf(3C)
wide-character string and rendition from a cchar_t ; get a	getcchar(3X)

Index

All Volumes

Description	Entry Name(Section)
wide-character string and rendition; set <code>cchar_t</code> from a	<code>setcchar(3X)</code>
wide-character string from a stream file; get	<code>fgetws(3C)</code>
wide-character string (restartable); convert a character string to a	<code>mbsrtowcs(3C)</code>
wide-character string to a character string; convert	<code>wcrtombs(3C)</code>
wide-character string, insert into a window	<code>ins_nwstr(3X)</code>
wide-character string; convert date and time to	<code>wcsftime(3C)</code>
wide-characters; find, compare, set, or copy in memory	<code>wmemory(3C)</code>
width for user and group names; get current display	<code>ug_display_width(3C)</code>
<code>win_wch()</code> - input a complex character and rendition from a window	<code>in_wch(3X)</code>
<code>win_wchnstr()</code> - input an array of complex characters and renditions from a window	<code>in_wchnstr(3X)</code>
<code>win_wchstr()</code> - input an array of complex characters and renditions from a window	<code>in_wchnstr(3X)</code>
<code>winch()</code> - input a single-byte character and rendition from a window	<code>inch(3X)</code>
<code>winchnstr()</code> - input an array of single-byte characters and renditions from a window	<code>inchstr(3X)</code>
<code>winchstr()</code> - input an array of single-byte characters and renditions from a window	<code>inchstr(3X)</code>
window and cursor coordinates, get additional	<code>getbegyx(3X)</code>
window attribute control functions	<code>attr_get(3X)</code>
window attribute control functions, restricted	<code>attroff(3X)</code>
window attributes, set and clear	<code>standend(3X)</code>
window coordinate transformation, define	<code>mvderwin(3X)</code>
window creation function	<code>derwin(3X)</code>
window creation functions	<code>newwin(3X)</code>
window cursor location functions	<code>move(3X)</code>
window ID of running program or start program in particular memory window; change ...	<code>setmemwindow(1M)</code>
window ID; file containing applications and their associated memory	<code>services.window(4)</code>
window IDs of user processes from <code>/etc/services.window</code> ; extract	<code>getmemwindow(1M)</code>
window refresh control function	<code>touchwin(3X)</code>
window refresh control functions	<code>is_linetouched(3X)</code>
window refreshed after echo single-byte character and rendition	<code>echochar(3X)</code>
window with its parents or children; synchronize a	<code>syncok(3X)</code>
window, change renditions of characters in a window	<code>chgat(3X)</code>
window, clear	<code>clear(3X)</code>
window, clear from cursor to end of window	<code>clrtoeol(3X)</code>
window, convert formatted input	<code>vwscanw(3X)</code>
window, convert formatted input from	<code>mvscanw(3X)</code>
window, copy a region of window	<code>copywin(3X)</code>
window, current	<code>curscr(3X)</code>
window, default	<code>stdscr(3X)</code>
window, delete	<code>delwin(3X)</code>
window, delete or insert lines into	<code>insdelln(3X)</code>
window, dump to and reload from a file	<code>getwin(3X)</code>
window, duplicate	<code>dupwin(3X)</code>
window, get cursor and window coordinates	<code>getyx(3X)</code>
window, input a complex character and rendition from	<code>in_wch(3X)</code>
window, input a multi-byte character string from	<code>innstr(3X)</code>
window, input a single-byte character and rendition from	<code>inch(3X)</code>
window, input a string of wide characters from	<code>innwstr(3X)</code>
window, input an array of complex characters and renditions from	<code>in_wchnstr(3X)</code>
window, insert a complex character and rendition into	<code>ins_wch(3X)</code>
window, insert a multi-byte character into	<code>insnstr(3X)</code>
window, insert a single-byte character and rendition into	<code>insch(3X)</code>
window, insert a wide-character string into	<code>ins_nwstr(3X)</code>
window, insert lines into	<code>insertln(3X)</code>
window, move	<code>mvwin(3X)</code>
window, print formatted output	<code>vwprintw(3X)</code>
window, print formatted output in	<code>mvprintw(3X)</code>
window, refresh immediately after writing a complex character	<code>echo_wchar(3X)</code>
window, scroll a curses window	<code>scroll(3X)</code>
window, scroll, enhanced curses	<code>scr1(3X)</code>
window; convert formatted input from a	<code>vw_scanw(3X)</code>
window; print formatted output in a	<code>vw_printw(3X)</code>
windows, copy overlapped windows	<code>overlay(3X)</code>

Description	Entry Name(Section)
winnstr() - input a multi-byte character string from a window	innstr(3X)
winnwstr() - input a string of wide characters from a window	innwstr(3X)
wins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
wins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
wins_wstr() - insert a wide-character string into a window	ins_wstr(3X)
winsch() - insert a single-byte character and rendition into a window	insch(3X)
winsdelln() - delete or insert lines into a window	insdelln(3X)
winsertln() - insert lines into a window	insertln(3X)
winsnstr() - insert a multi-byte character into a window	insnstr(3X)
winsstr() - insert a multi-byte character into a window	insnstr(3X)
winstr() - input a multi-byte character string from a window	innstr(3X)
winnwstr() - input a string of wide characters from a window	innwstr(3X)
wmemchr() - find a wide-character in memory	wmemory(3C)
wmemcmp() - compare wide-characters in memory	wmemory(3C)
wmemcpy() - copy wide-characters in memory	wmemory(3C)
wmemmove() - copy wide-characters in memory with overlapping areas	wmemory(3C)
wmemset() - set wide-characters in memory	wmemory(3C)
wmove() - window cursor location functions	move(3X)
wnoutrefresh() - refresh windows and lines	doupdate(3X)
word expansions, perform	wordexp(3C)
word from a stream file; get character or	getc(3S)
word or character, put on a stream	putc(3S)
wordexp() - perform word expansions	wordexp(3C)
wordfree() - free memory associated with word expansions	wordexp(3C)
words in a file; count	wc(1)
words, find hyphenated	hyphen(1)
words, lines, and bytes or characters in a file; count	wc(1)
working directory	glossary(9)
working directory name	pwd(1)
working directory, change	cd(1)
working directory, change	chdir(2)
working directory, get path-name of current	getcwd(3C)
working directory, get pathname of current	getwd(3C)
workstation; OBSOLETE; control access to audio on a	asecure(1M)
wprintf() - print formatted wide-character output	fwprintf(3C)
wprintw() - print formatted output in window	mvwprintw(3X)
wrapper service requests, evaluate tcp	tcpdmatch(1)
Wrappers, utility programs for TCP	tryfrom(1)
wredrawln() - line update status functions	redrawwin(3X)
wrefresh() - refresh windows and lines	doupdate(3X)
write(1) messages from other users to terminal, deny or permit	mesg(1)
write a character rendition and immediately refresh the pad	pechochar(3X)
write a complex character and immediately refresh the window	echo_wchar(3X)
write a message simultaneously to all users	wall(1M)
write a null-terminated string on a stream	puts(3S)
write a null-terminated wide string on a stream	putws(3C)
write an EFI file system header on a device file	efi_fsinit(1M)
write audit record for self-auditing process	audwrite(2)
write cache in the SCSI subsystem (OBSOLETE); enable and disable use of device's	default_disk_ir(5)
write - interactively write (talk) to another user	write(1)
write LIF volume header on file	lifinit(1)
write message onto system log file	syslog(3C)
write or delete information in the user database, /var/adm/userdb, read,	userdb_read(3)
write password file entry	putpwent(3C)
write records into new wtmps and btmps database	bwtmps(3C)
write() - STREAMS enhancements to standard system calls	stream(2)
write STREAMS event trace messages to standard output	strace(1M)
write to all users over a network	rwall(1M)
write to specified remote machines	rwall(3N)
write() - write contiguous data to a file	write(2)
write, asynchronous start	aio_write(2)

Index

All Volumes

Description	Entry Name(Section)
write/read file pointer, move	lseek(2)
writes, and lists archive files; copies files and directory hierarchies; extracts,	pax(1)
writev() - STREAMS enhancements to standard system calls	stream(2)
writev() - write noncontiguous data to a file	write(2)
writing, open file for	open(2)
wscanf() - convert formatted wide-character input	fwscanf(3C)
wscanw() - convert formatted input from a window	mvscanw(3X)
wscrl() - scroll the window, enhanced cursors	scrl(3X)
wsetscrreg() - terminal output control functions	clearok(3X)
wstandend() - set and clear window attributes	standend(3X)
wstandout() - set and clear window attributes	standend(3X)
wsyncdown() - synchronize a window with its parents or children	syncok(3X)
wsyncup() - synchronize a window with its parents or children	syncok(3X)
wtimeout() - control blocking on input	notimeout(3X)
wtmp - user login record format	utmp(4)
wtmpfix - manipulate connect accounting records	fwtmp(1M)
wtmps and btmps database, write records into new	bwtmps(3C)
wtmps file	login(1)
wtmps - user login information	wtmps(4)
wtouchln() - window refresh control functions	is_linetouched(3X)
WU-FTPD group access file information; change	privatepw(1)
wunctrl() - generate printable representation of a wide character	wunctrl(3X)
wvline() - draw lines from single-byte characters and renditions	hline(3X)
wvline_set() - draw lines from complex characters and renditions	hline_set(3X)
Wyse60 terminal	swinstall(1M)
Wyse60 terminal	swremove(1M)
X.25 line, get	getx25(1M)
X.25 switched virtual circuit, clear	clrsvc(1M)
X/Open Networking Interfaces	xopen_networking(7)
X/OPEN Transport Interface - XTI; accept a connect request issued by a transport user	t_accept(3)
X/OPEN Transport Interface - XTI; acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
X/OPEN Transport Interface - XTI; bind address to transport endpoint	t_bind(3)
X/OPEN Transport Interface - XTI; close transport endpoint	t_close(3)
X/OPEN Transport Interface - XTI; disable transport endpoint	t_unbind(3)
X/OPEN Transport Interface - XTI; error message function	t_error(3)
X/OPEN Transport Interface - XTI; establish connection with another transport user	t_connect(3)
X/OPEN Transport Interface - XTI; establish transport endpoint	t_open(3)
X/OPEN Transport Interface - XTI; free library structure	t_free(3)
X/OPEN Transport Interface - XTI; get current state	t_getstate(3)
X/OPEN Transport Interface - XTI; get protocol address	t_getprotaddr(3)
X/OPEN Transport Interface - XTI; get protocol-specific service information	t_getinfo(3)
X/OPEN Transport Interface - XTI; initiate orderly release at transport endpoint	t_sndrel(3)
X/OPEN Transport Interface - XTI; listen for connect request	t_listen(3)
X/OPEN Transport Interface - XTI; look at current event on transport endpoint	t_look(3)
X/OPEN Transport Interface - XTI; produce error message string	t_strerror(3)
X/OPEN Transport Interface - XTI; receive confirmation from connect request	t_rcvconnect(3)
X/OPEN Transport Interface - XTI; receive data over connection	t_rcv(3)
X/OPEN Transport Interface - XTI; receive data unit from remote transport provider user	t_rcvudata(3)
X/OPEN Transport Interface - XTI; receive error information from unit data error indication	t_rcvuderr(3)
X/OPEN Transport Interface - XTI; retrieve disconnect information	t_rcvdis(3)
X/OPEN Transport Interface - XTI; send data or expedited data over a connection	t_snd(3)
X/OPEN Transport Interface - XTI; send data unit to transport user	t_sndudata(3)
X/OPEN Transport Interface - XTI; send user-initiated disconnect request	t_snddis(3)
X/OPEN Transport Interface - XTI; synchronize transport library for transport endpoint	t_sync(3)
X/OPEN Transport Layer Interface - TLI; accept a connect request issued by a transport user	t_accept(3)
X/OPEN Transport Layer Interface - TLI; acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
X/OPEN Transport Layer Interface - TLI; bind address to transport endpoint	t_bind(3)
X/OPEN Transport Layer Interface - TLI; close transport endpoint	t_close(3)
X/OPEN Transport Layer Interface - TLI; disable transport endpoint	t_unbind(3)

Description	Entry Name(Section)
X/OPEN Transport Layer Interface - TLI; error message function	t_error(3)
X/OPEN Transport Layer Interface - TLI; establish connection with another transport user	t_connect(3)
X/OPEN Transport Layer Interface - TLI; establish transport endpoint	t_open(3)
X/OPEN Transport Layer Interface - TLI; free library structure	t_free(3)
X/OPEN Transport Layer Interface - TLI; get current state	t_getstate(3)
X/OPEN Transport Layer Interface - TLI; get protocol-specific service information	t_getinfo(3)
X/OPEN Transport Layer Interface - TLI; initiate orderly release at transport endpoint	t_sndrel(3)
X/OPEN Transport Layer Interface - TLI; listen for connect request	t_listen(3)
X/OPEN Transport Layer Interface - TLI; look at current event on transport endpoint	t_look(3)
X/OPEN Transport Layer Interface - TLI; receive confirmation from connect request	t_rcvconnect(3)
X/OPEN Transport Layer Interface - TLI; receive data over connection	t_rev(3)
X/OPEN Transport Layer Interface - TLI; receive data unit from remote transport provider user ..	t_rcvudata(3)
X/OPEN Transport Layer Interface - TLI; receive error information from unit data error indication	t_rcvuderr(3)
X/OPEN Transport Layer Interface - TLI; retrieve disconnect information	t_revdis(3)
X/OPEN Transport Layer Interface - TLI; send data or expedited data over a connection	t_snd(3)
X/OPEN Transport Layer Interface - TLI; send data unit to transport user	t_sndudata(3)
X/OPEN Transport Layer Interface - TLI; send user-initiated disconnect request	t_snddis(3)
X/OPEN Transport Layer Interface - TLI; synchronize transport library for transport endpoint	t_sync(3)
xargs - construct argument lists and execute command	xargs(1)
xd - hexadecimal file dump	od(1)
xd - octal and hexadecimal file dump	od(1)
xdr - library routines for external data representation	xdr(3N)
XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr, library routines for external data representation	xdr(3N)
xdr, library routines for external data representation	xdr_admin(3N)
xdr, library routines for external data representation	xdr_complex(3N)
xdr, library routines for external data representation	xdr_simple(3N)
xdr, library routines for external data representation stream creation	xdr_create(3N)
xdr_accepted_reply() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_accepted_reply() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_admin() - library routines for external data representation	xdr_admin(3N)
xdr_array() - library routine for external data representation	xdr_complex(3N)
xdr_authsys_parms() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_authsys_parms() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_authunix_parms() - write noncontiguous data to a file	rpc_soc(3N)
xdr_bool() - library routines for external data representation	xdr_simple(3N)
xdr_bytes() - library routine for external data representation	xdr_complex(3N)
xdr_callhdr() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_callhdr() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_callmsg() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_callmsg() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_char() - library routines for external data representation	xdr_simple(3N)
xdr_complex() - library routine for external data representation	xdr_complex(3N)
xdr_control() - library routines for external data representation	xdr_admin(3N)
xdr_create() - library routines for external data representation stream creation	xdr_create(3N)
xdr_destroy() - library routines for external data representation stream creation	xdr_create(3N)
xdr_double() - library routines for external data representation	xdr_simple(3N)
xdr_enum() - library routines for external data representation	xdr_simple(3N)
xdr_float() - library routines for external data representation	xdr_simple(3N)
xdr_free() - library routines for external data representation	xdr_simple(3N)
xdr_getpos() - library routines for external data representation	xdr_admin(3N)
xdr_hyper() - library routines for external data representation	xdr_simple(3N)
xdr_inline() - library routines for external data representation	xdr_admin(3N)
xdr_int() - library routines for external data representation	xdr_simple(3N)
xdr_long() - library routines for external data representation	xdr_simple(3N)
xdr_longlong_t() - library routines for external data representation	xdr_simple(3N)
xdr_opaque() - library routine for external data representation	xdr_complex(3N)
xdr_opaque_auth() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_opaque_auth() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_pointer() - library routine for external data representation	xdr_complex(3N)

Index

All Volumes

Description	Entry Name(Section)
xdr_quadruple() - library routines for external data representation	xdr_simple(3N)
xdr_reference() - library routine for external data representation	xdr_complex(3N)
xdr_rejected_reply() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_rejected_reply() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_replymsg() - write noncontiguous data to a file	rpc_xdr(3N)
xdr_replymsg() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_setpos() - library routines for external data representation	xdr_admin(3N)
xdr_short() - library routines for external data representation	xdr_simple(3N)
xdr_simple() - library routines for external data representation	xdr_simple(3N)
xdr_sizeof() - library routines for external data representation	xdr_admin(3N)
xdr_string() - library routine for external data representation	xdr_complex(3N)
xdr_u_char() - library routines for external data representation	xdr_simple(3N)
xdr_u_hyper() - library routines for external data representation	xdr_simple(3N)
xdr_u_int() - library routines for external data representation	xdr_simple(3N)
xdr_u_long() - library routines for external data representation	xdr_simple(3N)
xdr_u_longlong_t() - library routines for external data representation	xdr_simple(3N)
xdr_u_short() - library routines for external data representation	xdr_simple(3N)
xdr_union() - library routine for external data representation	xdr_complex(3N)
xdr_vector() - library routine for external data representation	xdr_complex(3N)
xdr_void() - library routines for external data representation	xdr_simple(3N)
xdr_wrapstring() - library routine for external data representation	xdr_complex(3N)
xdrmem_create() - library routines for external data representation stream creation	xdr_create(3N)
xdrrec_create() - library routines for external data representation stream creation	xdr_create(3N)
xdrrec_endofrecord() - library routines for external data representation	xdr_admin(3N)
xdrrec_eof() - library routines for external data representation	xdr_admin(3N)
xdrrec_readbytes() - library routines for external data representation	xdr_admin(3N)
xdrrec_skiprecord() - library routines for external data representation	xdr_admin(3N)
xdrstdio_create() - library routines for external data representation stream creation	xdr_create(3N)
xferlog - FTP server logfile	xferlog(5)
XMODEM -protocol file transfer program	umodem(1)
xntpd - Network Time Protocol daemon	xntpd(1M)
xntpd - special NTP query program	xntpd(1M)
xopen_networking - X/Open Networking Interfaces	xopen_networking(7)
xprt_register() - library routines for registering servers	rpc_svc_reg(3N)
xstr - extract strings from C programs to implement shared strings	xstr(1)
xstab - directories to export to NFS clients	exports(4)
XTI function; accept a connect request issued by a transport user	t_accept(3)
XTI function; acknowledge receipt of orderly release indication at transport endpoint	t_revrel(3)
XTI function; bind address to transport endpoint	t_bind(3)
XTI function; close transport endpoint	t_close(3)
XTI function; disable transport endpoint	t_unbind(3)
XTI function; error message function	t_error(3)
XTI function; establish connection with another transport user	t_connect(3)
XTI function; establish transport endpoint	t_open(3)
XTI function; free library structure	t_free(3)
XTI function; get current state	t_getstate(3)
XTI function; get protocol address	t_getprotaddr(3)
XTI function; get protocol-specific service information	t_getinfo(3)
XTI function; initiate orderly release at transport endpoint	t_sndrel(3)
XTI function; listen for connect request	t_listen(3)
XTI function; look at current event on transport endpoint	t_look(3)
XTI function; produce error message string	t_strerror(3)
XTI function; receive confirmation from connect request	t_rcvconnect(3)
XTI function; receive data over connection	t_rcv(3)
XTI function; receive data unit from remote transport provider user	t_revudata(3)
XTI function; receive error information from unit data error indication	t_rcvuderr(3)
XTI function; retrieve disconnect information	t_revdis(3)
XTI function; send data or expedited data over a connection	t_snd(3)
XTI function; send data unit to transport user	t_sndudata(3)
XTI function; send user-initiated disconnect request	t_snddis(3)
XTI function; synchronize transport library for transport endpoint	t_sync(3)

Description	Entry Name(Section)
y0() - Bessel functions of the second kind	y0(3M)
y0f() - Bessel functions of the second kind (float)	y0(3M)
y1() - Bessel functions of the second kind	y0(3M)
y1f() - Bessel functions of the second kind (float)	y0(3M)
yes - repetitively affirmative responses	yes(1)
yield frequency attribute; get and set mutex	pthread_mutexattr_getspin_np(3T)
yn() - Bessel functions of the second kind	y0(3M)
ynf() - Bessel functions of the second kind (float)	y0(3M)
yp_all() - Network Information Service client interface	ypclnt(3C)
yp_bind() - Network Information Service client interface	ypclnt(3C)
yp_first() - Network Information Service client interface	ypclnt(3C)
yp_get_default_domain() - Network Information Service client interface	ypclnt(3C)
yp_master() - Network Information Service client interface	ypclnt(3C)
yp_match() - Network Information Service client interface	ypclnt(3C)
yp_next() - Network Information Service client interface	ypclnt(3C)
yp_order() - Network Information Service client interface	ypclnt(3C)
yp_unbind() - Network Information Service client interface	ypclnt(3C)
ypbind - Network Information Service (NIS) server, binder, and transfer processes	yppserv(1M)
ypcat - print values in Network Information Service map	ypcat(1)
ypclnt() - Network Information Service client interface	ypclnt(3C)
yperr_string() - Network Information Service client interface	ypclnt(3C)
ypfiles - Network Information Service database and directory structure	ypfiles(4)
ypinit - build and install Network Information Service databases	ypinit(1M)
ypmake - create or rebuild Network Information Service databases	ypmake(1M)
ypmatch - print the values of selected keys in Network Information Service map	ypmatch(1)
yppasswd() - update user password in Network Information Service	yppasswd(3N)
yppasswd - change login password in Network Information System (NIS)	yppasswd(1)
yppasswdd - daemon for modifying Network Information Service passwd database	yppasswdd(1M)
yppoll - query NIS server for information about NIS map	yppoll(1M)
ypprot_err() - Network Information Service client interface	ypclnt(3C)
yppush - force propagation of Network Information Service database	yppush(1M)
yppserv - Network Information Service (NIS) server, binder, and transfer processes	yppserv(1M)
ypset - bind to particular Network Information Service server	ypset(1M)
ypupdate() - changes NIS information	ypupdate(3C)
ypupdated, rpc.yppupdated - server for changing NIS information	ypupdated(1M)
ypwhich - list which host is Network Information System server or map master	ypwhich(1)
ypxfr - transfer NIS database from server to local node	ypxfr(1M)
ypxfr_1perday - transfer NIS database from server to local node	ypxfr(1M)
ypxfr_1perhour - transfer NIS database from server to local node	ypxfr(1M)
ypxfr_2perday - transfer NIS database from server to local node	ypxfr(1M)
ypxfrd - Network Information Service (NIS) server, binder, and transfer processes	yppserv(1M)
zcat, compress, uncompress - compress or expand data	compress(1)
zero - /dev/zero special file	zero(7)
zero-length file, create	cp(1)
zero-length file, create	cat(1)
zero-length file, create	null(7)
zeroing of free memory in the background is enabled	pagezero_daemon_enabled(5)
zombie process	glossary(9)
zone signing tool; DNSSEC	dnssec-signzone(1)
zone validity checking tool	named-checkzone(1)
{ - execute commands in same shell	sh-posix(1)

Notes