AIX バージョン 7.2

コマンド・リファレンス
第 2 巻 (d から h)

AIX バージョン 7.2

コマンド・リファレンス
第 2 巻 (d から h)

お願い

本書および本書で紹介する製品をご使用になる前に、 823 ページの『特記事項』に記載されている情報をお読みください。

本書は AIX バージョン 7.2 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： AIX Version 7.2
Commands Reference, Volume 2, d - h

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

本書は、AIX® オペレーティング・システムのコマンドについての詳細情報を網羅して、エンド・ユーザーの皆様にお届けするものです。コマンドはカテゴリー別にアルファベット順に並べてあります。また、コマンドとそれらに使用可能なフラグについての詳細な説明を加えてあります。適用できる場合は、各コマンド・リストに例を含めてあります。本書 (第 2 巻) には、先頭文字が d から h までの AIX コマンドが記載してあります。なお、本資料は、オペレーティング・システムに付属して配送されている文書 CD にも収められています。

強調表示

本書では、次の強調表示規則を使用しています。

太字	コマンド、サブルーチン、キーワード、ファイル、構造体、ディレクトリー、およびシステムによって名前が事前に定義されているその他の項目を表します。 さらに太字の強調表示は、ユーザーが選択するボタン、ラベル、およびアイコンなどのグラフィカル・オブジェクトも示します。
イタリック	ユーザーが入力する実際の名前または値のパラメーターを示します。
モノスペース	具体的なデータ値の例、表示される可能性があるテキストの例、プログラマーとして作成する可能性があるものに似たプログラム・コードの一部の例、システムからのメッセージ、またはユーザーが入力しなければならないテキストを示します。

AIX でのケース・センシティブ

AIX オペレーティング・システムでは、すべてケース・センシティブとなっています。これは、英大文字と小文字を区別するという意味です。 例えば、**ls** コマンドを使用するとファイルをリストできます。 `LS` と入力すると、システムはそのコマンドが「`is not found`」と応答します。 同様に、**FILEA**、**FiLea**、および **filea** は、同じディレクトリーにある場合でも、3 つの異なるファイル名です。 予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

Single UNIX Specification のサポート

AIX オペレーティング・システムは、UNIX ベースのオペレーティング・システムのポータビリティーに関する The Open Group の「Single UNIX Specification Version 3 (UNIX 03)」をサポートするように設計されています。 数多くの新規インターフェースが追加され、またいくつかの現行インターフェースがこの仕様を満たすように拡張されました。 UNIX 03 ポータブル・アプリケーション開発する正しい方法を決定するには、UNIX System の Web サイト (http://www.unix.org) にある The Open Group の「UNIX 03」仕様を参照してください。

d

以下の AIX コマンドは文字 d で始まります。

dacinet コマンド

目的

CAPP/EAL4+ 構成の TCP ポートにおけるセキュリティーを管理します。

構文

dacinet aclflush

dacinet aclclear *Service* | *Port*

dacinet acladd *Service* | **[-]** *addr* [*/prefix_length*] [**u:***user* | *uid* | **g:***group* | *gid*]

dacinet acldel *Service* | **[-]** *addr* [*/prefix_length*] [**u:***user* | *uid* | **g:***group* | *gid*]

dacinet aclls *Service* | *Port*

dacinet setpriv *Service* | *Port*

dacinet unsetpriv *Service* | *Port*

dacinet lspriv

説明

dacinet コマンドは、TCP ポートにおけるセキュリティーの管理に使用されます。**dacinet** の各種機能の詳細については、『サブコマンド』のセクションを参照してください。

サブコマンド

項目	説明
acladd	ACL エントリーを、**dacinet** コマンドが使用するアクセス制御リストを保持しているカーネル・テーブルに追加します。 **acladd** サブコマンドのパラメーターの構文は、次のとおりです。 **[-]***addr***[/***length***][u:***user*|*uid*| **g:***group*|*gid*]パラメーターは、次のように定義されます。 ***addr*** DNS ホスト名または IPv4 (または IPv6) アドレス。アドレスの前の "-" は、この ACL エントリーがアクセスの許可ではなくアクセスの拒否に使用されることを意味します。 ***length*** *addr* がホスト・アドレスではなくネットワーク・アドレスとして使用され、初めの *length* ビットが *addr* から取得されることを示します。 **u:*user*|*uid*** オプションのユーザー ID。*uid* を指定しない場合は、指定されたホストまたはサブネット上のすべてのユーザーにそのサービスへのアクセスが与えられます。指定した場合は、指定されたユーザーにのみアクセスが与えられます。 **g:*group*|*gid*** オプションのグループ ID。*gid* を指定しない場合は、指定されたホストまたはサブネット上のすべてのユーザーにそのサービスへのアクセスが与えられます。指定した場合は、指定されたグループにのみアクセスが与えられます。
aclclear	指定されたサービスまたはポートの ACL をクリアします。
acldel	ACL エントリーを、**dacinet** コマンドが使用するアクセス制御リストを保持しているカーネル・テーブルから削除します。 **dacinet acldel** サブコマンドは、そのエントリーを ACL に追加するのに使用されたパラメーターと正確に一致するパラメーターの指定で発行された場合にのみ、 ACL からエントリーを削除します。 **acldel** サブコマンドのパラメーターの構文は、次のとおりです。 **[-]***addr***[/***length***][u:***user*|*uid*| **g:***group*|*gid*]パラメーターは、次のように定義されます。 ***addr*** DNS ホスト名または IPv4 (または IPv6) アドレス。アドレスの前の "-" は、この ACL エントリーがアクセスの許可ではなくアクセスの拒否に使用されることを意味します。 ***length*** *addr* がホスト・アドレスではなくネットワーク・アドレスとして使用され、初めの *length* ビットが *addr* から取得されることを示します。 **u:*user*|*uid*** オプションのユーザー ID。*uid* を指定しない場合は、指定されたホストまたはサブネット上のすべてのユーザーにそのサービスへのアクセスが与えられます。指定した場合は、指定されたユーザーにのみアクセスが与えられます。 **g:*group*|*gid*** オプションのグループ ID。*gid* を指定しない場合は、指定されたホストまたはサブネット上のすべてのユーザーにそのサービスへのアクセスが与えられます。指定した場合は、指定されたグループにのみアクセスが与えられます。
aclflush	システムで定義されたすべての ACL をクリアし、すべての TCP ポートを接続要求 (ホストの root ユーザーからのものは除く) に対してアクセス不能にします。また、任意のプロセスが 1024 より大きい任意のポートへバインドできるように、特権ポートもクリアします。
aclls	指定されたサービスまたはポートの ACL をリストします。 **dacinet aclls 0** は、デフォルトの ACL をリストします。認証処理については、論理的な観点から、デフォルトの ACL がそのサービスの ACL に付加されます。その ACL 上のエントリーがそのサービスへ接続しようとしているユーザーに一致しない場合は、アクセスは拒否されます。1 つ以上のエントリーが存在する場合は、接続要求者に一致する *user*|*group@host*|*subnet* を備えたリスト上の最初のものが、そのサービスへユーザーが接続可能であるかを決定します。このようにして、グループの許可エントリーを追加する前に単にそのメンバーの拒否エントリーを追加することで、そのサービスへのアクセスを持つグループのメンバーに対してサービスを拒否することが可能になります。
lspriv	特権が永続的なものではない、特権を持つすべてのサービスまたはポートをリストします (つまり、1024 より大きいポート番号を持つ特権サービスのみをリストします)。
setpriv	指定されたサービスまたはポートを、スーパーユーザー特権を持つプロセスのみがそのポートにバインドできそのポート上のサービスを提供するような、特権状態にします。1024 より小さいポートは、永続的に特権を与えられているので無視されます。

項目	説明
unsetpriv	指定されたサービスまたはポートを、任意のプロセスがバインドできるような非特権状態にします。そのポートに特権がマークされているかどうかにかかわらず、任意のプロセスも、現行の一時ポート範囲内の任意のポートにバインドすることができます。

ファイル

項目	説明
/usr/sbin/dacinet	**dacinet** コマンドを含みます。

---

dadmin コマンド

目的

DHCP サーバーの状況を照会し、変更するのに使用します。

構文

dadmin [**-?**] [**-v**] [**-h** *Hostname*] [**-n** *interval*] [**-f**] **-d** *IpAddress* | [**-x**] **-i** | [**-x**] **-s** | **-t** **on**|**off**|*Value* | **-q** *IpAddress* | **-r** *IpAddress* | **-p** *IpAddress* | **-c** *ClientId*

説明

dadmin コマンドによって、DHCP 管理者は DHCP サーバー・データベースの状態を照会および変更することができます。このコマンドを使用すると、管理者による、IP アドレスの状況に関する、ローカルまたはリモートでの DHCP サーバーの照会、IP アドレス・プールの照会、クライアントの照会、IP アドレス・マッピングの削除、サーバーのリフレッシュ、サーバーのトレース・レベルの変更が可能になります。

dadmin コマンドは前のバージョンの DHCP サーバーと、それらのサーバーの IP アドレス状況をリストし、リフレッシュするための互換性があります。

IP アドレス情報の照会の際、**dadmin** コマンドは IP アドレスの状況を戻します。また、IP アドレスの状況に応じて、**dadmin** コマンドはリース期間、リース開始時刻、最後のリース時刻、サーバーが DNS をサポートするかどうか、この IP アドレスに関するレコード更新、およびこの IP アドレスにマップされたクライアント ID を返す場合があります。

クライアント情報の照会の際、**dadmin** コマンドは、クライアントの IP アドレスと IP アドレスの状況、クライアントに IP アドレスが与えられた最後の時刻、ホスト名、およびクライアントが使用するドメイン・ネーム、さらにサーバーがこの IP アドレスの場合に DNS、レコード更新をサポートするかどうかを戻します。

サーバーのトレース・レベルを変更するとき、**dadmin** コマンドは、トレース・マスクのフォーマットでサーバー・トレース・レベルを設定し戻します。このマスクはビット・ストリングを表しており、その中の各ビットは、サーバーが特定のログ項目をトレースするか否かを表します (オンライン文書の 『DHCP サーバー構成ファイル』を参照してください)。これらのログ項目を最も重要でないものから最も重要なものの順に挙げると、LOG_NONE、 LOG_SYSERR、 LOG_OBJERR、 LOG_PROTOCOL および LOG_PROTERR (同じ値)、 LOG_WARN、 および LOG_CONFIG (同じ値)、 LOG_EVENT、および LOG_PARSEERR (同じ値)、 LOG_ACTION、 LOG_INFM、 LOG_ACNTING、 LOG_STAT、 LOG_TRACE、 LOG_START、 および LOG_RTRACE になります。

注**:** LOG_START は使用不可にはできません。そのことは、0x0800 から 0x1FFF までのマスクの範囲を意味しています。

フラグ

項目	説明
-c *ClientId*	DHCP サーバーには知られている特定のクライアントの状況を戻します。*ClientId* は DHCP クライアントがそれ自体を識別するために使用したクライアント ID です。このフィールドは、16 進文字のみを使用するか、DHCP サーバーが使用する TYPE-STRING 表記を使用するかのいずれかの方法で指定することもできます。
-d *IpAddress*	IP アドレス *IpAddress* に関連するリース情報を削除します。その結果、アドレスは FREE 状態に移され、再度バインドに使用可能になります。
-f	**-d** フラグと一緒に使用されます。 **-f** フラグにより、アドレスの削除が強制的に行われ、プロンプトは出されません。 IP に関連するリース情報を削除します。
-h *Hostname*	宛先 DHCP サーバーの指定に使用されます。 *Hostname* は、名前であっても IP アドレスであってもかまいません。
-i	DHCP サーバーを再初期化します。このフラグは、サーバーにそのデータベースの同期をとるよう信号を送り、構成ファイルを再読み取りして始動し直します。
-n *interval*	サーバーの統計情報、要約、および要求されたインターバルを表示します。
-p *IpAddress*	サブネット内の各アドレスの状況を戻します。 *IpAddress* は、サブネットをリストに示すために使用されます。
-q *IpAddress*	特定の IP アドレスの状況を戻します。
-r *IpAddress*	IP アドレスを `Free` 状態に置きます。
-s	DHCP サーバーの構成プール内の各アドレスの状況を戻します。
-t on|off|*Value*	DHCP サーバーのトレース・レベルを変更します。トレース値は、サーバーで使用中のトレース・マスクを表す 16 進フォーマットで、報告されます。*Value* は、10 進または 16 進フォーマットで指定できます。キーワード **on** または **off** によって、トレース・マスク内で、単一ビットが一度に使用可能もしくは使用不可になります。
-v	コマンドを冗長 (verbose) モードで実行します。
-x	**dadmin** プロトコルのバージョン 1 を使用します。 **-x** フラグは、前のリリース DHCP サーバーに接続する場合に使用され、**-i** および **-s** フラグの場合にのみ有効です。 DHCPv6 サーバーに接続するときは `6` をつけ加えてください。
-?	使用法の構文を表示します。

終了状況

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

dadmin クライアントからの接続を保護するために、DHCP サーバーは、サーバー自体からの接続か、スーパーユーザーの `.rhosts` ファイルに含まれているリモート・システムからの接続だけを許可します。通常のユーザーが DHCP サーバーのアドレス・マッピングを変更できないよう、管理者は、アクセスを許可されたシステムで適正なユーザーだけが **dadmin** コマンドを実行できるように制限する必要があります。

ファイル

項目	説明
/usr/sbin/dadmin	**dadmin** コマンドが入っています。

関連資料:

141 ページの『dhcpsd デーモン』

関連情報:

.rhosts コマンド

DHCP サーバー構成ファイル

TCP/IP アドレスとパラメーターの割り当て - 動的ホスト構成プロトコル

TCP/IP デーモン

---

date コマンド

目的

日付または時刻を表示または設定します。

構文

root ユーザーとして、日付と時刻を設定する

/usr/bin/date [**-n**] [**-u**] [*Date*] [**+***FieldDescriptor* ...]

日付と時刻を表示する

/usr/bin/date [**-u**] [**+***FieldDescriptor* ...]

root ユーザーとして、時刻 (秒単位) を調整する。

/usr/bin/date [**-a** [**+** | **-**]*sss*[*.fff*]

説明

重要**:** マルチユーザー・モードでシステムが稼働している場合は、日付の変更は行わないでください。

date コマンドは、フラグを付けずに呼び出されるか、**+** (正符号) で始まるフラグ・リストを付けて呼び出されると、現在の日時と時刻を標準出力に書き出します。それ以外の場合は、現在の日付を設定します。日付と時刻を変更できるのは root ユーザーだけです。**date** コマンドは、認識できないフラグまたは入力に対しては、使用方法のメッセージを出力します。

Date パラメーターによって日付を設定する場合、次のフォーマットを使用できます。

- *mmddHHMM*[*YYyy*]
- *mmddHHMM*[*yy*]

Date パラメーターの変数は次のように定義されます。

項目	説明
mm	月を指定します。
dd	日付を指定します。
HH	24 時間クロックを使用して、1 日のうちの時間を指定します。
MM	分を指定します。
YY	西暦の最初の 2 桁を指定します。 **注:** 西暦年の最初の 2 桁を指定しなかった場合、70 から 99 までの範囲の値は、20 世紀の 1970 年から 1999 年までを指します。同様に、00 から 37 までの範囲の値は、21 世紀の 2000 年から 2037 年までの年を指します。
yy	西暦の最後の 2 桁を指定します。 **注: date** コマンドは、4 桁の年号を入力として受け付けます。例えば、 4 桁の年号が指定された場合、**date** コマンドはその年号を *YYyy* に設定しようとし、その値が範囲外 (1970 より小さい、または 2105 より大きい) である場合は失敗します。2038 から 2105 の範囲の年号の場合は、*yyyy* 形式で年号を指定してください。

年が指定されない場合は、現在の年がデフォルト値として使用されます。システムは協定世界時 (CUT) で作動します。

date コマンドの後に **+** (正符号) とフィールド・ディスクリプターを付けると、コマンドの出力を制御できます。各フィールド・ディスクリプターの前には **%** (パーセント記号) を付けなければなりません。システムは、フィールド・ディスクリプターを指定した値と置換します。リテラルの % は、%% (パーセント記号 2 つ) で入力してください。 **date** コマンドは、その他の文字を変更せずに出力にコピーします。 **date** コマンドは、文字列を常に改行文字で終わらせます。

フラグ

項目	説明
-a [**+** | **-**]*sss*[.*fff*]	*sss.fff* 秒により、遅延して時刻を調整します (*fff* は秒の少数部分を表します)。この調整は、正または負のどちらにも行うことができます。 システムのクロックは、指定された秒数によりドリフトするまでスピードアップまたはスローダウンします。
-n	ローカル・エリア・ネットワーク内の、クロックが同期化されている全システムの時刻をグローバルに設定しません。
-u	時刻を万国標準時 (CUT) で表示または設定します。

フィールド・ディスクリプター

項目	説明
%a	ロケールの曜日名の省略型を表示します。
%A	ロケールの曜日の完全名を表示します。
%b	ロケールの月名の省略型を表示します。
%B	ロケールの月の完全名を表示します。
%c	ロケールの該当する日時表記を表示します (デフォルト)。
%C	4 桁の年の最初の 2 桁を 10 進数で表示します (00-99)。年は 100 で除算されて、小数点以下は切り捨てられます。
%d	日付を 10 進数で表示します (01 から 31)。1 桁の日付の場合は、10 の位の空間を 0 で埋めて 2 桁のフィールドとします。
%D	**%m/%d/%y** に相当するフォーマットで日付を表示します。
%e	日付を 10 進数で表示します (1 から 31)。1 桁の月の場合は、10 の位の空間をスペースで埋めて 2 桁のフィールドとします。
%h	ロケールの月名の省略型を表示します (**%b** の同義語です)。
%H	時 (24 時間制) を 10 進数で表示します (00 から 23)。
%I	時 (12 時間制) を 10 進数で表示します (01 から 12)。
%j	日付を 10 進数で表示します (001 から 366)。
%k	24 時間クロックの時間を、右寄せし、スペースで充てんした数値 (0 から 23) で表示します。
%m	月を 10 進数で表示します (01 から 12)。
%M	分数を 10 進法で表示します (00 から 59)。
%n	改行文字を挿入します。
%p	ロケールの AM か PM に相当するものを表示します。

項目	説明
%r	12 時間制の時刻 (01 から 12) を、AM/PM 表記で表示します。POSIX ロケールでは、それは **%I:%M:%S %p** に相当します。
%S	秒数を 10 進数で表示します (00 から 59)。
%s	協定世界時 (CUT) で 1970 年 1 月 1 日からの秒数を表示します。
%t	<タブ> 文字を挿入します。
%T	24 時間制の時間 (00 から 23) を HH:MM:SS に相当するフォーマットで表示します。
%u	曜日を 1 から 7 までの範囲の 10 進数として表示します (日曜日 = 7)。**%w** フィールド・ディスクリプターのセクションを参照してください。
%U	年の始めから数えて何週目かということを 10 進数 (00 から 53 まで) で表示します (日曜日を週の始めの曜日とします)。新年の最初の日曜日より前にあるすべての日は、週 0 と見なされます。
%V	年の始めから何週目かを 01 から 53 までの範囲の 10 進数で表示します (月曜日が週の始めの日として使用されます)。新年の 1 月 1 日を含む週が 4 日以上であれば、その週が週 01 と見なされます。そうでない場合、その週は前年の週 53 になります。
%w	曜日を 0 から 6 までの範囲の 10 進数として表示します (日曜日 = 0)。**%u** フィールド・ディスクリプターのセクションを参照してください。
%W	月曜日を週の始めの曜日として、年の始めから何週目かを 10 進数 (00 から 53) で表示します。
%x	ロケールの適切なフォーマットで日付を表示します。
%X	ロケールの適切な時刻表示方法を表示します。
%y	西暦の下 2 桁を表示します (00 から 99)。
%Y	4 桁の西暦年を 10 進数で表示します。
%Z	時間帯名を表示します。時間帯が確定できない場合は文字は表示されません。
%%	% (パーセント記号) を表示します。

フィールド・ディスクリプターの修飾

%E と **%O** フィールド・ディスクリプターを修飾すると、異なるフォーマットまたは仕様を表すことができます。詳しくは、「ファイル参照」の『Locale Definition Source File Format』から『LC_TIME Category』を参照してください。現在のロケールで対応するキーワード (**era**、**era_year**、**era_d_fmt**、**alt_digits** の各キーワードのセクションを参照) が指定されていないか、またはサポートされていない場合は、未修飾のフィールド・ディスクリプターが使用されます。

項目	説明
%Ec	ロケールの適切な代替表示方法で日付と時刻を表示します。
%EC	基本年 (または他の期間) の名前をロケールの代替表示方法で表示します。
%Ex	ロケールの代替表示方法で日付を表示します。
%EX	ロケールの代替表示方法で時刻を表示します。
%Ey	**%EC** フィールド・ディスクリプター (年のみ) からのオフセットをロケールの代替表示方法で示します。
%EY	完全な代替表示方法で年を表示します。
%Od	ロケールの代替数値記号を使用して日付を表示します。
%Oe	ロケールの代替数値記号を使用して日付を表示します。
%OH	ロケールの代替数値記号を使用して時刻 (24 時間制) を表示します。
%OI	ロケールの代替数値記号を使用して時刻 (12 時間制) を表示します。

項目	説明
%Om	ロケールの代替数値記号を使用して月を表示します。
%OM	ロケールの代替数値記号を使用して分数を表示します。
%OS	ロケールの代替数値記号を使用して秒数を表示します。
%Ou	ロケールの代替表示方法を使用して曜日を数字で表示します (月曜 =1)。
%OU	ロケールの代替数値記号を使用して年の何週目であるかを表示します。日曜を週の最初の曜日と見なします。
%OV	ロケールの代替数値記号を使用して年の何週目であるかを表示します。月曜を週の最初の曜日と見なします。
%Ow	ロケールの代替表示方法を使用して曜日を数字で表示します (日曜 =0)。
%OW	ロケールの代替数値記号を使用して年の何週目であるかを表示します。月曜を週の最初の曜日と見なします。
%Oy	代替表示方法で年 (%C からのオフセット) を表示します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	日付が正常に書き込まれました。
>0	エラーが発生しました。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 現在の日時を表示するには、次のように入力します。

   ```
   date
   ```

2. 日付と時刻を設定するには、次のように入力します。

   ```
   date 0217142590
   ```

   CST をタイムゾーンとして使用するシステムでは、このコマンドにより、日付と時刻が Sat Feb 17 14:25:00 CST 1990 に設定されます。

   注**:** 日付と時刻を変更するには、root ユーザー権限がなければなりません。

3. 日付と時刻を指定したフォーマットで表示するには、次のように入力します。

   ```
   date +"%r %a %d %h %y (Julian Date: %j)"
   ```

   このコマンドにより、例 2 に示した日付は次のように表示されます。

   ```
   02:25:03 PM Fri 17 Feb 90 (Julian Date: 048)
   ```

環境変数

次の環境変数が **date** コマンドの実行に影響します。

項目	説明
LANG	*LC_ALL* とそれに対応する環境変数 (*LC_* で始まる) の両方がロケールを指定していないときは、使用するロケールを決めます。
LC_ALL	*LANG* または *LC_* で始まる環境変数を設定することによって指定されたロケール・カテゴリー値を指定変更するのに使用するロケールを決定します。
LC_CTYPE	テキスト・データのバイト・シーケンスを文字として解釈するのに使用するロケールを決めます (例えば、引数内の単一バイト文字対複数バイト文字)。
LC_MESSAGES	メッセージを書き出す際の言語を決めます。
LC_TIME	**date** コマンドによって書き込まれる日付および時刻の文字列の内容を決定します。
NLSPATH	*LC_MESSAGES* の処理のためにメッセージ・カタログのロケーションを決定します。
TZ	**-u** フラグが指定されていない場合に、時刻と日付が書き込まれるタイムゾーンを指定します。*TZ* 変数が設定されていないで、また **-u** フラグも指定されない場合、未指定システムのデフォルト・タイムゾーンが使われます。

関連情報:

localtime コマンド

time コマンド

LC_TIME カテゴリー
ロケールについて

dbts コマンド

目的

シン・サーバーをデバッグします。

構文

dbts [**-v**] *ThinServer*

説明

dbts コマンドはシン・サーバーをデバッグ・モードで始動します。このコマンドは、シン・サーバー用に作成されたデバッグ・ブート・イメージを検索して、シン・サーバーが以前にデバッグ・モードで始動されたかどうかを検査します。デバッグ・ブート・イメージが検出されなかった場合、シン・サーバーが使用している共通イメージのクローンが作成され、そのクローンからデバッグ・ブート・イメージが作成されて、シン・サーバーがデバッグ・モードにブートできるようになります。デバッグ・ブート・イメージ・クローンは次の命名規則を使用します。

{*COSI name*}_{*thin server name*}-debug

シン・サーバーがデバッグ共通イメージの使用を完了した後で、**swts** コマンドを実行してシン・サーバーを別の共通イメージに切り替える必要があります。 **rmcosi** コマンドは、**dbts** コマンドから作成されたデバッグ共通イメージを除去します。**dbts** コマンドは、NIM マスターまたはシン・サーバー上でのみ実行できます。

フラグ

項目	説明
-v	**dbts** コマンドの実行中に詳細デバッグ出力を使用可能にします。

終了状況

項目	説明
0	コマンドは正常に完了しました。
>0	エラーが発生しました。

セキュリティー

アクセス制御: **dbts** コマンドを実行するには、root 権限が必要です。

例

1. cosi1 という名前の共通イメージを使用している lobo という名前のシン・サーバーをデバッグ・モードにブートするには、次のように入力します。

   ```
   dbts lobo
   ```

   cosi1_lobo-debug という名前のデバッグ・ブート・イメージが作成され、lobo をデバッグ・モードにブートします。

位置

/usr/sbin/dbts

ファイル

項目	説明
/etc/niminfo	NIM が使用する変数が入っています。

関連情報:

lsts コマンド

mkts コマンド

nim コマンド

rmts コマンド

---

dbx コマンド

目的

プログラムをデバッグして実行する環境を提供します。

構文

dbx [**-a** *ProcessID*] [**-B** *DebugFile*] [**-c** *CommandFile*] [**-I** *Directory*] [**-E** *DebugEnvironment*] [**-p** *oldpath=newpath:...* | *pathfile*] [**-u**] [**-F**] [**-L**] [**-r**] [**-x**] [**-v**] [**-C** *CoreFile* | *ObjectFile* [*CoreFile*]]

説明

dbx コマンドは、C、C++、および Fortran の各プログラム用のシンボリック・デバッグ・プログラムを提供します。これにより、以下の操作が可能です。

- オブジェクト・ファイルとコア・ファイルの検査
- プログラム実行のための制御された環境の提供
- 選択したステートメントにブレークポイントを設定、または一度に 1 行ずつプログラムを実行
- 記号変数を使ったデバッグ、および記号変数の正しい形式の表示

ObjectFile パラメーターは、コンパイラーが生成するオブジェクト (実行可能) ファイルです。プログラムのコンパイル時に **dbx** コマンドに必要な情報を生成するには、**-g** (記号テーブル生成) フラグを使用します。

注**:** オブジェクト・ファイルをコンパイルするときは、**cc** コマンドの **-g** フラグを使用してください。**-g** フラグを使わない場合、または **strip** コマンドにより **xcoff** ファイルから記号参照が除去される場合、**dbx** コマンドの記号機能が制限されます。また、**dbx** によるデバッグを計画している実行可能プログラムの最適化には、**-O** コンパイラー・オプションを使用しないでください。最適化では、コードが再編成され、**dbx** コマンドによる実行可能プログラムのデバッグの値をさらに制限することにより、デバッグ・データを小さくします。

-c フラグを指定しないと、**dbx** コマンドはユーザーの **$HOME** ディレクトリー内に **.dbxinit** ファイルがないか調べます。それから、ユーザーの現行ディレクトリー内に **.dbxinit** ファイルがないか調べます。**.dbxinit** ファイルが現行ディレクトリー内にある場合は、そのファイルがユーザーの **$HOME** ディレクトリー内の **.dbxinit** ファイルを指定変更します。 **.dbxinit** ファイルがユーザーの **$HOME** ディレクトリー

または現行ディレクトリー内にある場合は、そのファイルのサブコマンドがデバッグ・セッションの開始時に実行されます。 **.dbxinit** ファイルを作成するにはエディターを使用してください。

ObjectFile を指定しないと、 **dbx** は検査するオブジェクト・ファイルの名前を探します。デフォルトの設定は **a.out** です。 **core** ファイルが現行ディレクトリー内に存在する場合、または *CoreFile* パラメーターが指定されている場合には、 **dbx** はプログラムに障害が発生した位置を報告します。*ObjectFile* の実行が始まるまでは、コア・イメージ内に保持されている変数、レジスター、およびメモリーを調べることができます。プログラムの実行が始まると、 **dbx** デバッグ・プログラムはコマンドの入力を促すプロンプトを表示します。

-B フラグを使用すると、始動に関するデバッグ情報を含む代替オブジェクト・ファイルまたは個別の .stab ファイルを指定できます。代替オブジェクト・ファイルは、プロセスに付加している場合にのみ指定可能です。このデバッグ情報は、実行中のプロセスのディスク・コピーからではなく、この代替オブジェクト・ファイルまたは .stab デバッグ・ファイルから読み取られます。この代替オブジェクト・ファイルは、元のオブジェクト・ファイルのストリップされていないコピーでなければなりません。それ以外の場合、これは無視されます。**-B** フラグは、デバッグ・セクションのサイズが大きい場合に使用してください。実行中はオブジェクト・ファイルのストリップされているコピーを使用し、デバッグ中はストリップされていないコピーを使用します。.stab デバッグ・ファイルは、**–bstabsplit** リンカー・オプションを使用して生成されます。**-B** フラグを stabsplit 実行可能ファイルに指定しないと、**dbx** コマンドにより実行可能ファイルのディレクトリーから対応する .stab ファイルの取得が試行されます。

式の処理

dbx プログラムは広範囲の式を表示できます。C 構文と Fortran の一部の拡張機能を使用して、**dbx** デバッグ・プログラム内の式を指定できます。

次の演算子をデバッグ・プログラムで使用することができます。

項目	説明
* (アスタリスク) または ^ (脱字記号)	間接またはポインターの参照解除を示します。
[ ] (大括弧) または **( )** (小括弧)	添え字配列式を示します。
. (ピリオド)	このフィールド参照演算子はポインターおよび構造と一緒に使用します。この演算子により C 演算子 -> (矢印) が不要になります (使用してもかまいません)。
& (アンパーサンド)	変数のアドレスを取得します。
.. (ピリオド 2 個)	配列の範囲を指定するとき、上限と下限を区切ります。例えば、**n[1..4]** のように指定します。

次のタイプの演算子をデバッグ・プログラム内の式に使用できます。

項目	説明
代数演算	**=**、**-**、*****、**/** (浮動除算)、**div** (整数除算)、**mod**、**exp** (指数)
ビット単位演算	**-**、**I**、**bitand**、**xor**、**~**. **<<**、**>>**
論理演算	**or**、**and**、**not**、**II**、**&&**
比較式	**<**、**>**、**<=**、**>=**、**< >** または **!=**、**=** または **==**
その他	**(typename),sizeof**

論理式と比較式は、**stop**、および **trace** での条件として使用可能です。

式のタイプが検査されます。式のタイプは、名前の変更またはキャスト演算子で指定変更できます。タイプの名前変更のフォーマットは、*Typename (Expression)*、*Expression* ¦*Typename*、および *(Typename) Expression* の 3 種類です。次の例は、*x* 変数が値 97 の整数である場合を示しています。

```
(dbx) print x
97
(dbx) print char (x), x ¥ char, (char) x, x
'a' 'a' 'a' 97
```

コマンド・ラインの編集

dbx コマンドには、Korn シェルの機能に似たコマンド・ライン編集機能があります。**vi** モードには **vi-like** 編集機能がありますが、 **emacs** モードには **emacs** と同じ制御があります。

これらの機能は、**dbx** サブコマンドの、 **set -o**、または **set edit** を使用するとオンにできます。 vi-style コマンド・ライン編集をオンにするには、サブコマンド **set edit vi** または **set -o vi** をタイプします。

EDITOR 環境変数を使用しても、編集モードを設定することができます。

dbx コマンドは、コマンド・ラインから入力されたコマンドのヒストリーを **.dbxhist** ヒストリー・ファイルに保管します。**DBXHISTFILE** 環境変数が設定されていない場合は、**$HOME/.dbxhist** ヒストリー・ファイルが使用されます。

デフォルトでは、**dbx** は、入力された最後の 128 コマンドのテキストを保管します。 **DBXHISTSIZE** 環境変数を使用して、この制限を増やすことができます。

フラグ

項目	説明
-a *ProcessID*	実行中のプロセスにデバッグ・プログラムを接続します。デバッグ・プログラムを接続するには、このプロセスにシグナルを送信する権限が必要です。プロセス ID を決定するには、 **ps** コマンドの使用権限が必要です。アクセス権を持っている場合、**dbx** プログラムは **ptrace** システム・コールを使用してプロセスを中断し、プロセスに **SIGTRAP** シグナルを送信します。プロセスは **SIGTRAP** シグナルを無視することができません。次にプログラムはオブジェクト・ファイルの絶対パス名を判別してシンボリック情報を読み取り、コマンドを要求するプロンプトを出します。
-B*DebugFile*	このフラグにより、始動時の代替デバッグ・ファイルを指定できます。
-c *CommandFile*	標準入力から読み取る前に、ファイル内の **dbx** サブコマンドを実行します。**$HOME** ディレクトリー内の指定ファイルが最初に処理されます。次に現行ディレクトリー内の指定ファイルが処理されます。現行ディレクトリー内のコマンド・ファイルは、 **$HOME** ディレクトリー内のコマンド・ファイルを指定変更します。指定ファイルが **$HOME** ディレクトリーにも現行ディレクトリーにも存在しなければ、警告メッセージが表示されます。**dbx** プログラムを始動すると、 **source** サブコマンドが使用できます。
-C *CoreFile*	オブジェクト・ファイルを指定しないでコア・ファイルを分析します。この場合、**dbx** コマンドはコア・ファイルに示されたオブジェクト・ファイルを使用します (そのファイルが現行ディレクトリーにあり、コア・ファイルと一致する場合)。それ以外の場合は、オブジェクト・ファイルを使用しないで進行します。このフラグは、**-r** フラグまたは **-a** フラグの後に使用すると無視されます。
-E *DebugEnvironment*	デバッグ・プログラムのための環境変数を指定します。
-p *oldpath=newpath:...* | *pathfile*	core ファイルを検査するとき、またはプロセスに付加するときのライブラリー・パスの置換を、*oldpath=newpath* のフォーマットで指定します。 *oldpath* 変数には、置換される値 (core ファイルまたはプロセスの付加時の Loader Section に格納されたとおり) を指定します。*newpath* 変数には、置換する値を指定します。*oldpath* 変数と *newpath* 変数には、完全パス、部分パス、相対パス、または絶対パスを指定できます。複数の置換値はコロンで区切ります。または、**-p** フラグでは、以前に記述されたフォーマットでのマッピングを読み取るファイル名を指定することもできます。ファイルからマッピングを読み取る場合は、1 行につき 1 マッピングのみ許可されます。プロセスに付加している場合に **-p** フラグを使用すると、デバッグ情報は置換されたパス・ファイルから読み取られます。パス・ファイルは、ライブラリーの実行中のコピーと一致しなければなりません。

項目	説明
-F	このフラグを使用すると、遅延読み取りモードがオフになり、**dbx** コマンドは起動時にすべての記号を読み取ります。デフォルトでは、遅延読み取りモードはオンなので、このコマンドは、 **dbx** セッション開始時に必要な記号テーブル情報だけを読み取ります。このモードでは、**dbx** は記号情報が読み取られていないローカル変数とタイプを読み取りません。したがって、**whereis i** などのコマンドを使用した場合に、すべての関数内のローカル変数 **i** のすべてのインスタンスが表示されるとは限りません。
-L	リンケージ・シンボルを保持します。
-I *Directory*	(大文字の i) ソース・ファイルを検索するディレクトリーのリストに、 *Directory* 変数で指定されるディレクトリーを含みます。デフォルトは、次のディレクトリー内でソース・ファイルを検索することです。 • コンパイル時にソース・ファイルが入っていたディレクトリー。このディレクトリーは、コンパイラーがソース・パスをオブジェクト内に配置した場合にのみ検索されます。 • 現行ディレクトリー。 • プログラムが現在入っているディレクトリー。
-r	即時にオブジェクト・ファイルを実行します。オブジェクト・ファイルが正常終了すると、 **dbx** デバッグ・プログラムは終了します。正常終了でない場合、デバッグ・プログラムに入り、終了理由を報告します。 **注: -r** が指定されていないと、**dbx** コマンドはプロンプトを表示し、コマンドの入力を待機します。
-u	ファイル名記号の先頭に @ (アットマーク) を付けるように、**dbx** コマンドに指示します。このフラグを使うと、未確定のシンボル名が少なくなります。
-v	**dbx** コマンドがコア・ファイルの妥当性検査を省略します。このフラグを使用すると、一部のセクションが無効な場合でも、コア・ファイルの有効なセクションを分析できます。
-x	Fortran ソース・コードで生じる記号から、**dbx** コマンドが _ (末尾の下線) を除去しないようにします。このフラグにより、**dbx** は、 xxx と xxx_ のように、下線文字以外は同じであるシンボルを区別できるようになります。

例

1. 次の例では、プロセスと同時に **dbx** デバッグ・プログラムを始動する方法について説明します。例で使っているのは、**samp.c** というプログラムです。まず、この C プログラムを下記のように **-g** オプションを使ってコンパイルし、記号テーブル参照を含むオブジェクト・ファイルを作成します。このオブジェクト・プログラムの名前は **samp** です。

   ```
   $ cc -g samp.c -o samp
   ```

   プログラム **samp** を実行すると、オペレーティング・システムはバス・エラーが発生したことを報告し、次のように現在の作業ディレクトリーにコア・イメージを書き込みます。

   ```
   $ samp
   Bus Error - core dumped
   ```

   エラーが発生した場所を判別するには、次のように入力します。

   ```
   $ dbx samp
   ```

   システムは次のメッセージを出力します。

   ```
   dbx version 3.1
   Type 'help' for help.
   reading symbolic information . . . [
   using memory image in core]
     25   x[i] = 0;
   (dbx) quit
   ```

2. この例では、**dbx** をプロセスに接続する方法を示しています。この例のプログラム名は、**looper.c** です。

```
main()
{
      int i,x[10];

      for (i = 0; i < 10;);
}
```

上記のプログラムは、**i** がまったく増分されないため、終了しません。**looper.c** を **-g** フラグを指定してコンパイルし、シンボリック・デバッグ機能を獲得します。

```
$ cc -g looper.c -o looper
```

コマンド・ラインから **looper** を実行し、実行中に **dbx** をプログラムに接続するために次のステップを実行します :

a. **dbx** を **looper** に接続するには、プロセス ID を決定しなければなりません。**looper** をバックグラウンド・プロセスとして実行しなかった場合は、X Window を別にオープンしなければなりません。この X Window から、次のように入力します。

   ```
   ps -u UserID
   ```

   ここでは、*UserID* はユーザーのログイン ID です。そのユーザーに属するアクティブ・プロセスがすべて次のように表示されます。

   ```
   PID     TTY      TIME    COMMAND
   68      console   0:04    sh
   467     lft3     10:48    looper
   ```

   この例では、**looper** のプロセス ID は 467 です。

b. **dbx** を **looper** に接続するには、次のように入力します。

   ```
   $ dbx -a 467
   ```

   システムは次のメッセージを出力します。

   ```
   Waiting to attach to process 467 . . .
   Successfully attached to /tmp/looper.
   dbx is initializing
   Type 'help' for help.
   reading symbolic information . . .

   attached in main at line 5
   5     for (i = 0; i < 10;);
   (dbx)
   ```

   これで、プロセスが **dbx** で開始されたものとして、プロセスを照会およびデバッグすることができます。

3. 実行可能ファイル **objfile** のソース・ファイルを検索するためのディレクトリーのリストにディレクトリーを追加するには、次のように入力します。

   ```
   $dbx -I /home/user/src -I /home/group/src
   objfile
   ```

   dbx が始動されると、 **use** サブコマンドを使用して、上記の処理を実行できます。**use** コマンドはディレクトリー・リストを再設定しますが、**-I** フラグはディレクトリー・リストにディレクトリーを追加します。

4. **-r** フラグを使うには、次のように入力します。

   ```
   $ dbx -r samp
   ```

   システムは次のメッセージを出力します。

```
Entering debug program . . .
dbx version 3.1
Type 'help' for help.
reading symbolic information . . .
bus error in main at line 25
  25   x[i] = 0;
(dbx) quit
```

-r フラグを使うと、コア・イメージをとらなくても、ユーザーのプロセスのメモリー状態を検査することができます。

5. デバッグ・プログラムのための環境変数を指定するには、次のように入力します。

   ```
   dbx -E LIBPATH=/home/user/lib -E LANG=Ja_JP objfile
   ```

6. プロセスに付加している場合に代替オブジェクト・ファイルとライブラリーを指定するには、次のように入力します。

   ```
   dbx –a 467 –B debug_samp –p /usr/lib/=./dir/debug_libs/
   ```

7. 始動時の個別のデバッグ・ファイルを指定するには、次のように入力します。

   ```
   dbx –B /usr/debug_samp.stab debug_samp
   ```

dbx サブコマンド

注: サブコマンドは、**dbx** デバッグ・プログラムを実行している間だけ使用できます。

項目	説明
/	現行ソース・ファイル内で前方向にパターンを検索します。
?	現行ソース・ファイル内で逆方向にパターンを検索します。
addcmd	指定されたイベント番号に **dbx** サブコマンドを追加します。
alias	**dbx** サブコマンドの別名を作成します。
assign	変数に値を代入します。
attribute	すべての属性オブジェクト、または選択した属性オブジェクトに関する情報を表示します。
call	指定したプロシージャーまたは関数に関連したオブジェクト・コードを実行します。
case	**dbx** デバッグ・プログラムが記号を解釈する方法を変更します。
catch	アプリケーション・プログラムへシグナルが送信される前に、そのシグナルのトラッピングを開始します。
clear	特定のソース行にあるすべての停止を除去します。
cleari	あるアドレスにあるすべてのブレークポイントを除去します。
condition	すべての条件変数、または選択した条件変数に関する情報を表示します。
cont	プログラムが終了するか、別のブレークポイントに行き当たるまで、現在の停止位置からアプリケーション・プログラムの実行を継続します。
corefile	コア・ファイルのハイレベル・データを表示します。
coremap	特定のアドレス・スペース領域のマッピングを表示します。
delcmd	指定されたイベント番号に関連した **dbx** サブコマンドを削除します。
delete	指定されたイベント番号とスレッドの **tskip** カウントに対応するトレースと停止を除去します。
detach	アプリケーションの実行を継続し、デバッグ・プログラムを終了します。
disable	指定のイベント番号に対応するトレースと停止を使用不可にします。
display memory	メモリーの内容を表示します。
down	現行関数をスタックの下方へ移動します。
dump	指定したプロシージャー内の変数の名前と値を表示します。
edit	指定したファイルでエディターを始動します。
enable	指定のイベント番号に対応するトレースと停止を使用可能にします。
fd	ファイル・ディスクリプター情報を表示します。
file	現行ソース・ファイルを、指定したファイルに変更します。
frame	現行関数を指定されたスタック・フレーム番号に対応する関数に変更します。
func	現行関数を指定したプロシージャーまたは関数に変更します。
goto	指定したソース行を次に実行する行にします。
gotoi	プログラム・カウンター・アドレスを変更します。
handler	pthread atfork または取り消しクリーンアップ・ハンドラーについての情報を表示します。

項目	説明
help	**dbx** サブコマンドまたはトピックのヘルプ情報を表示します。
ignore	アプリケーション・プログラムへシグナルが送信される前に、そのシグナルのトラッピングを停止します。
kthread	カーネル・スレッドについての情報を表示します。
limitbp	ブレークポイントを実行できる回数を制限します。
list	現行ソース・ファイルのリストを表示します。
listi	アプリケーション・プログラムの命令をリストします。
malloc	malloc サブシステムのプログラムの使用について情報を表示します。
map	アプリケーションのロード特性に関する情報を表示します。
move	次の表示行を変更します。
multproc	マルチプロセスのデバッグ機能を使用可能または使用不可にします。
mutex	すべての mutex、または選択された mutex に関する情報を表示します。
next	アプリケーション・プログラムを次のソース行まで実行します。
nexti	アプリケーション・プログラムを次のマシン・インストラクションまで実行します。
onceblock	once block についての情報を表示します。
plugin	プラグイン・サブコマンドの起動、または使用可能プラグインの名前の表示を行います。
pluginload	プラグインをロードします。
pluginunload	プラグインをアンロードします。
print	式の値を表示するかプロシージャーを実行して、そのプロシージャーの戻りコードを表示します。
printbp	ブレークポイントを実行する回数を印刷します。
proc	プロセスについての情報を表示します。
prompt	**dbx** コマンド・プロンプトを変更します。
quit	**dbx** デバッグ・プログラムを停止します。
registers	すべての汎用レジスター、システム制御レジスター、浮動小数点レジスター、および現行命令レジスターの値を表示します。
rerun	以前使用した引数でアプリケーションの実行を開始します。
resource	pthread によって所有されるか、または待機されているリソースについての情報を表示します。
return	指定したプロシージャーの復帰に戻るまで、アプリケーション・プログラムの実行を継続します。
rwlock	rwlocks に関する情報を表示します。
run	アプリケーションの実行を開始します。
screen	**dbx** コマンドによる対話のため X Window をオープンします。
set	**dbx** デバッグ・プログラムの変数値を定義します。
sh	シェルにコマンドを渡して実行させます。
skip	アプリケーション・プログラムの実行を現在の停止位置から継続します。
source	ファイルから **dbx** サブコマンドを読み取ります。
status	ブレークポイントの詳細を印刷します。また、アクティブ・トレース、**stop** サブコマンド、および残りのスレッドの **tskip** カウントを表示します。
step	ソース行を 1 行実行します。
stepi	マシン・インストラクションを 1 行実行します。
stophwp	ハードウェアの監視ポイントの停止を設定します。
stop	アプリケーション・プログラムの実行を停止します。
stopi	指定した位置に停止を設定します。
thdata	スレッド固有のデータを表示します。
thread	スレッドを表示して制御します。
tls	TLS 初期化テンプレート情報を表示します。
tm_status	*$texasr* 変数に格納される値を表示および解釈します。
tnext	スレッドを次のソース行まで実行します。
tnexti	スレッドを次のマシン・インストラクションまで実行します。
trace	トレース情報を表示します。
tracehwp	ハードウェアの監視ポイントのトレースを設定します。
tracei	トレース機能をオンにします。
tskip	スレッドのブレークポイントをスキップします。
tstep	1 ソース行のスレッドを実行します。
tstepi	1 マシン・インストラクションのスレッドを実行します。
tstop	スレッドのソース・レベル・ブレークポイント停止を設定します。
tstophwp	スレッド・レベルのハードウェア監視ポイントの停止を設定します。
tstopi	スレッドのインストラクション・レベル・ブレークポイントの停止を設定します。
ttrace	スレッドのソース・レベル・トレースを設定します。

項目	説明
ttracehwp	スレッド・レベルのハードウェア監視ポイントのトレースを設定します。
ttracei	スレッドのインストラクション・レベル・トレースを設定します。
unalias	別名を除去します。
unset	変数を削除します。
up	現行の関数をスタックの上方へ移動します。
use	ソース・ファイルの検索時に、検索されるディレクトリーのリストを設定します。
whatis	アプリケーション・プログラムのコンポーネントの宣言を表示します。
where	アクティブなプロシージャーと関数のリストを表示します。
whereis	指定した ID に名前が一致するすべての記号の完全修飾を表示します。
which	指定された ID の完全修飾を表示します。

/ サブコマンド

/ [*RegularExpression* [**/**]]

/ サブコマンドは、現行ソース・ファイル内で前方向に検索を行って、 *RegularExpression* パラメーターにより指定されたパターンを探します。引数を付けずに **/** サブコマンドを入力すると、**dbx** は以前に指定した正規表現を探して前方向に検索します。検索はファイルの終わりで先頭に折り返します。

例

1. 現行ソース・ファイル内で数値 12 を前方向に検索するには、次のように入力します。

   ```
   / 12
   ```

2. 直前の検索を繰り返すには、次のように入力します。

   ```
   /
   ```

? (検索) サブコマンドと、 **regcmp** サブルーチンを参照してください。

? サブコマンド

? [*RegularExpression* [**?**]]

? サブコマンドは、現行ソース・ファイル内で逆方向に検索を行って、 *RegularExpression* パラメーターにより指定されたパターンを探します。引数を付けずに **?**引数を無指定でサブコマンドを入力すると、 **dbx** コマンドは逆方向に検索を行って、以前指定した正規表現を探します。検索はファイルの終わりで先頭に折り返します。

例

1. 現行ソース・ファイル内で、文字 z を逆方向に検索するには、次のように入力します。

   ```
   ?z
   ```

2. 直前の検索を繰り返すには、次のように入力します。

   ```
   ?
   ```

/ (検索) サブコマンドと、 **regcmp** サブルーチンを参照してください。

addcmd サブコマンド

addcmd { *Number...* | **all** } "*commands_string*"

addcmd サブコマンドは、指定されたイベントに **dbx** サブコマンドを追加します。この指定されたイベントは、そのイベントに対応するブレークポイント、トレース・ポイント、または監視ポイントが実行される

たびに実行されます。**dbx** サブコマンドは "*commands_string*" パラメーターで指定できます。このパラメーターはセミコロン (;) で区切られた **dbx** サブコマンドのグループです。 **dbx** を追加するイベントは、*Number* パラメーターで指定できます。また、**all** フラグを使用して、すべてのイベントに **dbx** サブコマンドを追加することもできます。

フラグ

項目	説明
all	すべてのイベントに **dbx** サブコマンドを追加します。

例

1. イベント番号 1 に **where** サブコマンドを追加するには、次のように入力します。

   ```
   addcmd 1 "where"
   ```

2. イベント番号 2 に **registers** サブコマンドを追加するには、次のように入力します。

   ```
   addcmd 2 "registers"
   ```

3. イベント番号 3 に **where** サブコマンドと **registers** サブコマンドを追加するには、次のように入力します。

   ```
   addcmd 3 "where;registers"
   ```

clear サブコマンド、**delcmd** サブコマンド、**delete** サブコマンド、**disable** サブコマンド、**enable** サブコマンド、**stop** サブコマンド、**status** サブコマンド、および **trace** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』のセクションも参照してください。

alias サブコマンド

alias [*Name* [[(*Arglist*)] *String* | *Subcommand*]]

alias サブコマンドは、**dbx** サブコマンドの別名を作成します。*Name* パラメーターは、作成される別名です。*String* パラメーターは、このサブコマンドを実行した後で *Name* で参照できる一連の **dbx** サブコマンドです。パラメーターを付けずに **alias** サブコマンドを使用すると、現行の別名がすべて表示されます。

例

1. `rr` を、`rerun` の別名にするには、次のように入力します。

   ```
   alias rr rerun
   ```

2. コマンド・ラインに `printandstep` と入力したときに、 `print n` と `step` の、 2 つのサブコマンドを実行させるには、次のように入力します。

   ```
   alias printandstep "print n; step"
   ```

3. `alias` サブコマンドを、限定マクロ機能として使用することもできます。以下に例を示します。

   ```
   (dbx) alias px(n) "set $hexints; print n; unset $hexints"
   (dbx) alias a(x,y) "print symname[x]->symvalue._n_n.name.Id[y]"
   (dbx) px(126)
   0x7e
   ```

   この例では、別名 `px` は 16 進法で値を表示していますが、デバッグ環境への影響が永続することはありません。

assign サブコマンド

assign *Variable*=*Expression*

assign サブコマンドは、 *Expression* パラメーターで指定された値を *Variable* パラメーターで指定された変数に代入します。

例

1. 変数 x に値 5 を代入するには、次のように入力します。
   ```
   assign x = 5
   ```
2. 変数 x に変数 y の値を代入するには、次のように入力します。
   ```
   assign x =  y
   ```
3. 変数 z に文字値「z」を代入するには、次のように入力します。
   ```
   assign  z  =  'z'
   ```
4. 論理タイプ変数 B にブール値 false を代入するには、次のように入力します。
   ```
   assign  B  =  false
   ```
5. 文字ポインター Y に、「Hello World」という文字列を代入するには、次のように入力します。
   ```
   assign  Y  =  "Hello  World"
   ```
6. 型チェックを使用不可にするには、次のように入力して、**dbx** デバッグ・プログラム変数 $unsafeassign を設定します。
   ```
   set $unsafeassign
   ```

「変数の表示および変更」のセクションを参照してください。

attribute サブコマンド

attribute [*AttributeNumber* ...]

attribute サブコマンドは、*AttributeNumber* パラメーターにより定義されたユーザー・スレッド、mutex、または条件属性オブジェクトに関する情報を表示します。パラメーターを指定しないと、すべての属性オブジェクトが表示されます。

リストされる属性オブジェクトごとに、以下の情報が表示されます。

項目	説明
attr	属性オブジェクトのシンボル名を $a*AttributeNumber* フォーマットで示します。
obj_addr	属性オブジェクトのアドレスを示します。
type	属性オブジェクトのタイプを示します。この値には thr、mutex、または cond があり、それぞれユーザー・スレッド、mutex、および条件変数を表します。
state	属性オブジェクトの状態を示します。この値は valid または inval のいずれかです。
stack	スレッド属性オブジェクトのスタック・サイズ属性を示します。
scope	スレッド属性オブジェクトの有効範囲属性を示します。この値は、スレッドの競合有効範囲を決定し、リソース処理のために競合するスレッドのセットを定義します。値は、システムまたはプロセスの競合有効範囲を表す sys または pro です。
prio	スレッド属性オブジェクトの優先順位属性を示します。
sched	スレッド属性オブジェクトの schedpolicy 属性を示します。この属性はスケジュール・ポリシーを制御します。値は fifo、rr (ラウンドロビン)、または other です。
p-shar	mutex または条件属性オブジェクトのプロセス共用属性を示します。mutex または条件は、別のプロセスに属するスレッドがアクセスできる場合は、プロセス共用となります。値は、 yes または no です。
protocol	mutex のプロトコル属性を示します。この属性は、スレッドの優先順位に基づいて mutex を保持する効果を決定します。値は no_prio、prio、protect のいずれかです。
clock	条件属性オブジェクトのクロック属性を示します。この属性は、条件変数を待機しているスレッドがタイムアウトを指定されているとき、どのクロックを使用する必要があるかを判別します。値は realtime または monotonic とすることができます。

注**:**

1. **dbx** デバッグ・プログラムの **print** サブコマンドは、シンボルによる属性名を認識し、対応するオブジェクトの状況を表示するために使用できます。
2. 使用可能な属性は、実装される POSIX オプションに応じて異なります。

例

1. すべての属性に関する情報を表示するには、次のように入力します。

   ```
   attribute
   ```

   出力は以下のようになります。

   ```
   attr   obj_addr   type  state  stack   scope    prio
   sched p-shar
   $a1   0x200035c8  mutex valid                                 no
   $a2   0x20003628  cond  valid                                 no
   $a3   0x200037c8  thr   valid  57344    sys      126 other
   $a4   0x200050f8  thr   valid  57344    pro      126 other
   ```

2. 属性 1 と 3 に関する情報を表示するには、次のように入力します。

   ```
   attribute 1 3
   ```

   出力は以下のようになります。

   ```
   attr   obj_addr   type  state  stack   scope    prio
   sched p-shar
   $a1   0x200035c8  mutex valid                                 no
   $a3   0x200037c8  thr   valid  57344    sys      126 other
   ```

dbx コマンドの、 **condition** サブコマンド、 **mutex** サブコマンド、 **print** サブコマンド、および **thread** サブコマンドを参照してください。

また、「プログラミングの一般概念*:* プログラムの作成およびデバッグ」の『スレッドの作成』、『mutex の使用』、および『条件変数の使用』のセクションも参照してください。

call サブコマンド

call *Procedure* ([*Parameters*])

call サブコマンドは、*Procedure* パラメーターで指定されたプロシージャーを実行します。戻りコードは表示されません。パラメーターを指定すると、それらのパラメーターは実行中のプロシージャーに渡されます。

注**:** ベクトル・パラメーターを使用する機能を呼び出すのに、 **call** サブコマンドは使用できません。

例

dbx コマンドの実行中にコマンドを呼び出すには、次のように入力します。

```
(dbx) call printf("hello")
hello
```

printf は正常に処理を終了します。

case サブコマンド

case [**default** | **mixed** | **lower** | **upper**]

case サブコマンドは、**dbx** デバッグ・プログラムが記号を解釈する方法を変更します。デフォルトの記号処理は、現行の言語に基づいています。現行の言語が C、C++、または未定義であれば、記号は大文字に変換されません。現行の言語が Fortran であれば、記号は小文字に変換されます。このサブコマンドは、現行の言語と整合しない方法で記号を解釈する必要がある場合に使用します。

パラメーターを付けずに **case** サブコマンドを入力すると、現在のケース・モードが表示されます。

フラグ

項目	説明
default	現行の言語により異なります。
mixed	記号の解釈を、それらが実際に表示されるとおりに行います。
lower	記号を小文字として解釈させます。
upper	記号を大文字として解釈させます。

例

1. 現在のケース・モードを表示するには、次のように入力します。

   ```
   case
   ```

2. **dbx** に対して、シンボルを実際の表示どおりに解釈するように命令するには、次のように入力します。

   ```
   case mixed
   ```

3. **dbx** に対して、シンボルを大文字として解釈するように命令するには、次のように入力します。

   ```
   case upper
   ```

「小文字から大文字への変数の変換」のセクションを参照してください。

catch サブコマンド

catch [*SignalNumber* | *SignalName*]

catch サブコマンドは、シグナルがアプリケーション・プログラムへ送信される前に、指定したシグナルのトラッピングを開始します。このサブコマンドは、デバッグ中のアプリケーション・プログラムが割り込みなどのシグナルを処理する場合に有効です。トラップされるシグナルは、*SignalNumber* パラメーターまたは *SignalName* パラメーターを使用して、番号または名前で指定できます。シグナル名は大/小文字の区別がなく、**SIG** プレフィックスはオプションです。 *SignalNumber* パラメーターおよび *SignalName* パラメーターを指定しない場合は、デフォルトとして、**SIGHUP**、**SIGCLD**、**SIGALARM**、**SIGKILL** の各シグナルを除くすべてのシグナルがトラップされます。引数を指定していない場合は、トラップされるシグナルの現行リストが表示されます。

例

1. **dbx** コマンドがキャッチするシグナルの、現在のリストを表示するには、次のように入力します。

   ```
   catch
   ```

2. シグナル SIGALARM をトラップするには、次のように入力します。

   ```
   catch SIGALARM
   ```

ignore サブコマンド、およびシグナルの処理を参照してください。

clear サブコマンド

clear *SourceLine*

clear サブコマンドは、特定のソース行にあるすべての停止を除去します。*SourceLine* パラメーターは、以下の 2 つのフォーマットで指定できます。

- 整数として
- 後ろに : (コロン) と整数が続くファイル名文字列として

例

行 19 に設定されたブレークポイントを除去するには、次のように入力します。

```
clear 19
```

cleari サブコマンドと、 **delete** サブコマンドのセクションを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』のセクションも参照してください。

cleari サブコマンド

cleari *Address*

cleari サブコマンドは、 *Address* パラメーターで指定されたアドレスにあるすべてのブレークポイントをクリアします。

例

1. アドレス 0x100001b4 に設定されたブレークポイントを除去するには、次のように入力します。

   ```
   cleari 0x100001b4
   ```

2. main() プロシージャー・アドレスに設定されたブレークポイントを除去するには、次のように入力します。

   ```
   cleari &main
   ```

clear サブコマンドのセクション、 **delete** サブコマンドのセクション、および プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』のセクションを参照してください。

condition サブコマンド

condition [wait | **nowait** | *ConditionNumber ...* **]**

condition サブコマンドは、1 つ以上の条件変数に関する情報を表示します。 1 つ以上の *ConditionNumber* パラメーターを指定すると、 **condition** サブコマンドは指定した条件変数に関する情報を表示します。フラグまたはパラメーターを指定しないと、**condition** サブコマンドはすべての条件変数を表示します。

各条件について表示される情報は、以下のとおりです。

項目	説明
cv	条件変数のシンボル名を $c*ConditionNumber* フォーマットで示します。
obj_addr	条件変数のメモリー・アドレスを示します。
clock	条件変数のクロック属性を示します。
num_wait	条件変数を待機しているスレッドの数を示します。
waiters	条件変数を待機しているユーザー・スレッドの数を示します。

注**: dbx** デバッグ・プログラムの **print** サブコマンドは、記号条件変数名を認識するので、対応するオブジェクトの状況を表示するために使用できます。

フラグ

項目	説明
wait	待機中のスレッドのある条件変数を表示します。
nowait	待機中のスレッドのない条件変数を表示します。

例

1. すべての条件変数に関する情報を表示するには、次のように入力します。

   ```
   condition
   ```

2. 待機中のスレッドがあるすべての条件変数に関する情報を表示するには、次のように入力します。

   ```
   condition  wait
   ```

3. 条件変数 3 に関する情報を表示するには、次のように入力します。

   ```
   condition 3
   ```

   出力は以下のようになります。

   ```
   cv      obj_addr     num_wait  waiters
   $c3     0x20003290         0
   ```

attribute サブコマンド、 **mutex** サブコマンド、 **print** サブコマンド、および **thread** サブコマンドを参照してください。

また、「プログラミングの一般概念*:* プログラムの作成およびデバッグ」の『条件変数の使用』も参照してください。

cont サブコマンド

cont [*SignalNumber* | *SignalName*]

cont サブコマンドは、現在の停止位置からアプリケーション・プログラムの実行を継続し、プログラムが終了するか、別のブレークポイントに行き当たるまで、続けます。シグナルが *SignalNumber* パラメーターに指定した番号か、 *SignalName* パラメーターで指定した名前によって指定されている場合は、プログラムはそのシグナルを受信したものとして継続されます。シグナル名は、大文字、小文字を区別せず、**SIG** プレフィックスはオプションです。シグナルが指定されていないと、プログラムは停止されなかったものとして継続されます。

例

1. プログラムの実行を現在の停止位置から継続するには、次のように入力します。

   ```
   cont
   ```

2. シグナル SIGQUIT を受信したとしてプログラムの実行を継続するには、次のように入力します。

```
cont SIGQUIT
```

dbx コマンドの **detach** サブコマンド、 **dbx** コマンドの **goto** サブコマンド、 **dbx** コマンドの **next** サブコマンド、 **dbx** コマンドの **skip** サブコマンド、 **dbx** コマンドの **step** サブコマンドを参照してください。

corefile サブコマンド

corefile サブコマンドは、コア・ファイルについて、実行可能ファイル名、コア・ファイル・フォーマットのバージョン情報、使用可能データを示すフラグ、クラッシュの原因となったシグナル、およびコアをダンプしたプロセスの実行モードなど、そのヘッダーの情報を表示します。

coremap サブコマンド

coremap [*stack* | *data* | *sdata* | *mmap* | *shm* | *loader*]

coremap サブコマンドは、特定のアドレス・スペース領域のマッピングを表示します。領域名を指定しないと、**coremap** サブコマンドは使用可能なマッピングをすべて表示します。

例

1. 共用メモリー領域のマッピングを表示するには、次のように入力します。
   ```
   coremap shm
   ```
2. メモリー・マップ済み領域のマッピングを表示するには、次のように入力します。
   ```
   coremap mmap
   ```
3. ローダー・エントリーで示されたすべての領域のマッピングを表示するには、次のように入力します。
   ```
   coremap loader
   ```
4. 使用可能なマッピングをすべて表示するには、次のように入力します。
   ```
   coremap
   ```

corefile サブコマンドを参照してください。

delcmd サブコマンド

delcmd *EventNumber* { *Number...* | **all** }

delcmd サブコマンドは、指定されたイベントに関連した **dbx** サブコマンドを除去します。除去する **dbx** サブコマンドは、*Number* パラメーターで指定できます。また、**all** フラグを使用して、指定されたイベントに関連したすべての **dbx** サブコマンドを除去することもできます。 *EventNumber* パラメーターは、**dbx** サブコマンドを除去するイベントを指定します。

フラグ

項目	説明
all	指定されたイベントに関連したすべての **dbx** サブコマンドを除去します。

例

1. イベント番号 2 からすべての **dbx** サブコマンドを除去するには、次のように入力します。

   ```
   delcmd 2 all
   ```

2. イベント番号 3 から **dbx** サブコマンド番号 1 を除去するには、次のように入力します。

   ```
   delcmd 3 1
   ```

3. イベント番号 2 から **dbx** サブコマンド番号 1 および 2 を除去するには、次のように入力します。

   ```
   delcmd 2 1 2
   ```

addcmd サブコマンド、**clear** サブコマンド、**delete** サブコマンド、**disable** サブコマンド、**enable** サブコマンド、**stop** サブコマンド、**status** サブコマンド、および **trace** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』のセクションも参照してください。

delete サブコマンド

delete { *Number* ... | **all** | **tskip** [**for $t***threadnumber*]}

delete サブコマンドは、アプリケーション・プログラムおよびスレッドの **tskip** カウントからトレースと停止を除去します。除去するトレースおよび停止は、*Number* パラメーターで指定できます。また、**all** フラグを使用して、すべてのトレースおよび停止を除去することもできます。 **dbx** デバッグ・プログラムによってトレースまたは停止に関連付けられた番号を表示するときは、 **status** サブコマンドを使用します。

tskip サブコマンドを使用して選択に設定された残りの **tskip** カウントは、**tskip** フラグを使用して削除できます。 **status** サブコマンドを使用して、残ったスレッド **tskip** カウントを表示してください。スレッドが指定されていない場合は、現行スレッドが使用されます。

フラグ

項目	説明
all	トレースと停止をすべて除去します。
for $t *threadnumber*	スレッド番号を指定します。

例

1. アプリケーション・プログラムからすべてのトレースと停止を除去するには、次のように入力します。

   ```
   delete all
   ```

2. イベント番号 4 のトレースと停止を除去するには、次のように入力します。

   ```
   delete 4
   ```

3. スレッド 3 の **tskip** カウントを除去するには、次のように入力します。

   ```
   delete tskip for $t3
   ```

4. 現行スレッドの **tskip** カウントを除去するには、次のように入力します。

   ```
   delete tskip
   ```

clear サブコマンド、**cleari** サブコマンド、**status** サブコマンド、**tskip** サブコマンド、および プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』を参照してください。

detach サブコマンド

detach [*SignalNumber* | *SignalName*]

detach サブコマンドは、アプリケーション・プログラムの実行を継続し、デバッグ・プログラムを終了します。シグナルは、以下のいずれかによって指定することができます。

- 名前。*SignalName* パラメーターを使用。
- 番号。*SignalNumber* パラメーターを使用。

シグナル名は、大文字、小文字を区別せず、**SIG** プレフィックスはオプションです。

シグナルが指定されていると、プログラムはそのシグナルを受信したものとして継続されます。シグナルが指定されていないと、プログラムは停止が起こらなかったものとして継続されます。

例

1. アプリケーションの実行を継続し、**dbx** を終了するには、次のように入力します。

   ```
   detach
   ```

2. **dbx** を終了し、アプリケーションがシグナル SIGREQUEST を受信したとしてアプリケーションの実行を継続するには、次のように入力します。

   ```
   detach SIGREQUEST
   ```

『**dbx** デバッグ・プログラムの使用』を参照してください。

disable サブコマンド

disable { *Number* ... **all** }

disable サブコマンドは、デバッグ・イベントに関連するトレースと停止を使用不可にします。 *Number* パラメーターを使って使用不可にするトレースと停止を指定することも、あるいは **all** フラグを使ってすべてのトレースと停止を使用不可にすることもできます。 **dbx** デバッグ・プログラムによってトレースまたは停止に関連付けられたイベント番号を表示するときは、 **status** サブコマンドを使用します。

フラグ

項目	説明
all	トレースと停止をすべて除去します。

例

1. アプリケーション・プログラムからすべてのトレースと停止を使用不可にするには、次のように入力します。

   ```
   disable all
   ```

2. イベント番号 4 のトレースと停止を使用不可にするには、次のように入力します。

   ```
   disable 4
   ```

詳しくは、**enable subcommand**、**delete subcommand**、および **status subcommand** を参照してください。

また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』も参照してください。

display memory サブコマンド

{ *Address,Address*/ | *Address*/ [*Count*] } [*Mode*] [>*File*]

display memory サブコマンドは、以下の要素で制御されるメモリーの部分を表示します。このサブコマンドには、コマンドを始動するキーワードはありません。

表示されるメモリーの範囲は、以下のいずれかの指定により制御されます。

- 2 つの *Address* パラメーター。この場合、2 つのアドレスの間にあるすべての行が表示されます。

  または

- 表示開始位置を示す 1 つの *Address* パラメーターと、 *Address* からの表示行数を決定する *Count* パラメーター。

記号アドレスを指定するには、名前の前に & (アンパーサンド) を付けます。アドレスは、他のアドレスと演算子 + (正符号)、- (負符号)、 * (間接) からなる式で作成できます。括弧で囲まれた式は、アドレスとして解釈されます。

- メモリーの表示形式を制御するには、*Mode* パラメーターを使用します。*Mode* パラメーターのデフォルトは現行モードです。*Mode* の初期値は **X** です。以下のモードが使用できます。

項目	説明
b	1 バイトを 8 進で表示します。
c	1 バイトを 1 文字として表示します。
d	短ワードを 10 進で表示します。
D	長ワードを 10 進で表示します。
Df	倍精度の 10 進数浮動小数点数を表示します。
DDf	4 倍精度の 10 進数浮動小数点数を表示します。
f	単精度実数を表示します。
g	倍精度実数を表示します。
h	1 バイトを 16 進で表示します。
Hf	単精度の 10 進数浮動小数点数を表示します。
i	マシン・インストラクションを表示します。
lld	8 バイトの符号付き 10 進数を表示します。
llu	8 バイトの符号なし 10 進数を表示します。
llx	8 バイトの符号なし 16 進数を表示します。
llo	8 バイトの符号なし 8 進数を表示します。
o	短ワードを 8 進で表示します。
O	長ワードを 8 進で表示します。
p	アドレス/ポインターを 16 進で表示します。
q	拡張精度浮動小数点数を表示します。
s	null バイトで終了する文字列を表示します。
x	短ワードを 16 進で表示します。
X	長ワードを 16 進で表示します。

フラグ

項目	説明
>*File*	指定したファイルに出力をリダイレクトします。

例

1. アドレス 0x3fffe460 から、1 つの長ワード分のメモリーの内容を 16 進で表示するには、次のように入力します。

   ```
   0x3fffe460 / X
   ```

2. 変数 y のアドレスから、2 バイト分のメモリーの内容を文字として表示するには、次のように入力します。

   ```
   &y / 2c
   ```

3. Fortran 文字列 a_string の 6 番目から 8 番目までのエレメントを表示するには、次のように入力します。

   ```
   &a_string + 5, &a_string + 7/c
   ```

プログラミングの一般概念*:* プログラムの作成およびデバッグ の『メモリー・アドレスの検査』を参照してください。

down サブコマンド

down [*Count*]

down サブコマンドは、現行関数を *Count* レベル数だけスタックの下方へ移動します。現行関数は、名前の解決に使用します。*Count* パラメーターのデフォルト値は 1 です。

例

1. スタックの下方に 1 レベル移動するには、次のように入力します。

   ```
   down
   ```

2. スタックの下方に 3 レベル移動するには、次のように入力します。

   ```
   down 3
   ```

プログラミングの一般概念*:* プログラムの作成およびデバッグ の **up** サブコマンド、**where** サブコマンド、および『スタック・トレースの表示』を参照してください。

dump サブコマンド

dump [*Procedure* | **"PATTERN"**] [>*File*]

dump サブコマンドは、指定されたプロシージャー内のすべての変数の名前と値、または指定されたパターンに一致する変数の名前と値を表示します。*Procedure* パラメーターがピリオド (**.**) の場合は、すべてのアクティブ変数が表示されます。 *Procedure* パラメーターも **"PATTERN"** パラメーターも指定しない場合は、現行プロシージャーが使用されます。 **"PATTERN"** パラメーターは、メタ文字 *、?、および [] を使用したワイルドカード式です。 **"PATTERN"** を使用すると、グローバル・スペース内の (すべてのプロシージャーからの) 一致する記号がすべて表示されます。 >*File* フラグを使用すると、出力は指定したファイルにリダイレクトされます。

フラグ

項目	説明
>*File*	指定したファイルに出力をリダイレクトします。

例

1. 現行プロシージャー内の変数の名前と値を表示するには、次のように入力します。

   ```
   dump
   ```

2. **add_count** プロシージャー内の変数の名前と値を表示するには、次のように入力します。

   ```
   dump add_count
   ```

3. 文字 s で始まる変数の名前と値を表示するには、次のように入力します。

   ```
   dump "s*"
   ```

4. 現行プロシージャー内の変数の名前と値を、**var.list** ファイルにリダイレクトするには、次のように入力します。

   ```
   dump > var.list
   ```

プログラミングの一般概念*:* プログラムの作成およびデバッグ の『変数の表示および変更』のセクションを参照してください。

edit サブコマンド

edit [*Procedure* | *File*]

edit サブコマンドは、指定したファイルに対してエディターを開始します。ファイルを指定するには、*File* パラメーターまたは *Procedure* パラメーターを使用します。Procedure パラメーターを使用すると、そのプロシージャーを含むファイルに対してエディターが開始します。ファイルを指定しないと、エディターは現行のソース・ファイルに対して開始します。デフォルトは **vi** エディターです。 **EDITOR** 環境変数を必要なエディターの名前にリセットすると、このデフォルトをオーバーライドできます。

例

1. 現行ソース・ファイル上でエディターを始動するには、次のように入力します。

   ```
   edit
   ```

2. main.c ファイル上でエディターを始動するには、次のように入力します。

   ```
   edit main.c
   ```

3. do_count() プロシージャーが入っているファイル上のエディターを始動するには、次のように入力します。

   ```
   edit do_count
   ```

list サブコマンド、 **vi** コマンド、または **vedit** コマンドのセクションを参照してください。

enable サブコマンド

enable { *Number* ... **all** }

enable サブコマンドは、デバッグ・イベントに関連するトレースと停止を使用可能にします。 *Number* パラメーターを使って使用可能にするトレースと停止を指定することも、あるいは **all** フラグを使ってすべてのトレースと停止を使用可能にすることもできます。 **dbx** デバッグ・プログラムによってトレースまたは停止に関連付けられたイベント番号を表示するときは、 **status** サブコマンドを使用します。

フラグ

項目	説明
all	トレースと停止をすべて除去します。

例

1. アプリケーション・プログラムからすべてのトレースと停止を使用可能にするには、次のように入力します。

   ```
   enable all
   ```

2. イベント番号 4 のトレースと停止を使用可能にするには、次のように入力します。

   ```
   enable 4
   ```

詳しくは、**disable subcommand**、**delete subcommand**、**status subcommand** を参照してください。

また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』も参照してください。

fd サブコマンド

fd [**raw**] [*start* [*end*]]

fd サブコマンドは、ファイル・ディスクリプター情報を表示します。 **raw** オプションを使用すると、出力はロウの 16 進数フォーマットで表示されます。その他のオプショナル引数には、*start* および *end* インデックスがあります。インデックスが与えられないと、使用できるすべてのファイル・ディスクリプターについての情報が表示されます。1 つのインデックスを使用すると、1 つのファイル・ディスクリプターが表示され、2 つのインデックスを使用するとその範囲に含まれるファイル・ディスクリプターが表示されます。

例

1. すべてのファイル・ディスクリプターについての情報を 16 進数で表示するには、次のように入力します。

   ```
   fd raw
   ```

2. 範囲 3 から 5 のファイル・ディスクリプターについての情報を表示するには、次のように入力します。

   ```
   fd 3 5
   ```

file サブコマンド

file [*File*]

file サブコマンドは、現行ソース・ファイルを *File* パラメーターにより指定されたファイルに変更します。ただし、そのファイルへの書き込みは実行しません。 *File* パラメーターは、ファイルへの絶対パス名を指定します。*File* パラメーターにパスを指定しないと、 **dbx** プログラムは use パスを探すことによりファイルを見つけようとします。 *File* パラメーターを指定しないと、 **file** サブコマンドは現行ソース・ファイル名を表示します。また、パスが認識されていれば、**file** サブコマンドはファイルの絶対パス名または相対パス名を表示します。

例

1. 現在のソース・ファイルを、`main.c` ファイルに変更するには、次のように入力します。

   ```
   file main.c
   ```

2. 現在のソース・ファイル名を表示するには、次のように入力します。

```
file
```

func サブコマンドのセクションを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『現在のファイルまたはプロシージャーの変更』、および『現在のファイルの表示』も参照してください。

frame サブコマンド

frame [*num*]

frame サブコマンドは、現行関数を指定されたスタック・フレーム番号 *num* に対応する関数に変更します。現行関数は、名前の解決に使用します。スタック・フレームの番号付けは、現在アクティブな関数のスタック・フレームから始まります (現在アクティブな関数フレームの番号は常に `0` となります)。*n* 個のフレームがある場合、**main** 関数のフレームの番号は `n-1` になります。フレーム番号を指定しなかった場合は、現行フレームに関連した関数に関する情報が表示されます。

例

1. フレーム番号 `2` に移動するには、次のように入力します。
   ```
   frame 2
   ```
2. スタック上の現行関数を表示するには、次のように入力します。
   ```
   frame
   ```

up サブコマンドおよび **down** サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『現在のファイルまたはプロシージャーの変更』および『スタック・トレースの表示』も参照してください。

func サブコマンド

func [*Procedure*]

func サブコマンドは、現行関数を *Procedure* パラメーターにより指定されたプロシージャーまたは関数に変更します。 *Procedure* パラメーターを指定しないと、デフォルトの現行関数が表示されます。現行関数を変更すると、現行ソース・ファイルが新しい関数が入ったファイルに暗黙的に変更されます。ネーム・レゾリューションに使用される現行有効範囲も変更されます。

例

1. 現行関数を、`do_count` プロシージャーに変更するには、次のように入力します。
   ```
   func do_count
   ```
2. 現行関数名を表示するには、次のように入力します。
   ```
   func
   ```

file サブコマンドのセクションを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『現在のファイルまたはプロシージャーの変更』も参照してください。

goto サブコマンド

goto *SourceLine*

goto サブコマンドは、指定したソース行を次に実行するソース行にします。一般に、ソース行は現行ソース行と同じ関数内に入っている必要があります。この制約を無効にするには、 **$unsafegoto** フラグを指定した **set** サブコマンドを使用します。

例

次に実行させる行を行 6 に変更するには、次のように入力します。

```
goto 6
```

cont サブコマンド、 **gotoi** サブコマンド、および **set** サブコマンドを参照してください。

gotoi サブコマンド

gotoi *Address*

gotoi サブコマンドは、プログラム・カウンターのアドレスを *Address* パラメーターにより指定されたアドレスに変更します。

例

プログラム・カウンターのアドレスをアドレス `0x100002b4` に変更するには、次のように入力します。

```
gotoi 0x100002b4
```

goto サブコマンドのセクションを参照してください。

handler サブコマンド

handler { **atfork** | **cancel_cleanup** [**all** | *pthread id*] }

handler サブコマンドは、それぞれ **pthread_atfork** および **pthread_cleanup_push** を使用して登録されている atfork または取り消しクリーンアップのハンドラーについての情報を表示します。 *atfork* オプションを使用すると、`pre`、`parent`、`child` の atfork ハンドラーとして登録されているルーチンの名前が表示されます (POSIX 規則に準拠しない atfork ハンドラーの場合は、それぞれの引数とともに)。
cancel_cleanup オプションは、特定の pthread を指定するオプショナル *pthread id* パラメーター、またはすべての pthread を指定する **all** とともに指定されて、すべての登録済み取り消しクリーンアップ・ハンドラーを表示させます。どちらも指定しない場合、現行の pthread の取り消しクリーンアップ・ハンドラーがあれば、それが表示されます。

例

1. すべての登録済み atfork ハンドラーに関する情報を表示するには、次のように入力します。

   ```
   handler atfork
   ```

2. 現行 pthread の登録済み取り消しクリーンアップ・ハンドラーについての情報を表示するには、次のように入力します。

   ```
   handler cancel_cleanup
   ```

3. `$t2` と呼ばれる pthread オブジェクトの登録済み取り消しクリーンアップ・ハンドラーについての情報を表示するには、次のように入力します。

   ```
   handler cancel_cleanup 2
   ```

help サブコマンド

help [*Subcommand* | *Topic*]

help サブコマンドは、指定するパラメーターに応じて、 **dbx** サブコマンドまたはトピックに関するヘルプ情報を表示します。 *Subcommand* パラメーターを指定して **help** サブコマンドを入力すると、指定したサブコマンドの構文のステートメントと説明が表示されます。 *Topic* パラメーターを指定して **help** サブ

コマンドを入力すると、指定したトピックの詳細な説明が表示されます。 **help** サブコマンドでトピック文字列全体を提供する必要はありません。トピックの先頭から何文字かを指定すれば、**dbx** プログラムはトピックを認識できます。以下のトピックを表示することができます。

項目	説明
startup	**dbx** 始動オプションを表示します。
execution	プログラムの実行に関連する **dbx** サブコマンドを表示します。
breakpoints	ブレークポイントとトレースに関連する **dbx** サブコマンドを表示します。
ファイル	ソース・ファイルにアクセスするための **dbx** サブコマンドを表示します。
data	プログラム変数とデータにアクセスするための **dbx** サブコマンドを表示します。
machine	マシン・レベルのデバッグ用の **dbx** サブコマンドの説明を表示します。
environment	**dbx** の構成と環境を設定するための **dbx** サブコマンドを表示します。
threads	スレッド関連オブジェクトにアクセスするための **dbx** サブコマンドを表示します。
式	**dbx** の式の構文と演算子を記述します。
scope	**dbx** により異なる有効範囲からの名前の解決方法を記述します。
set_variables	**dbx** デバッグ変数とその使用方法を表示します。
usage	共通 **dbx** サブコマンドとその簡単な説明を表示します。

例

1. 使用可能なすべての **dbx** サブコマンドとトピックを表示するには、次のように入力します。

   ```
   help
   ```

2. **dbx** サブコマンド、 **list** の説明を表示するには、次のように入力します。

   ```
   help list
   ```

3. **dbx** トピック、 **set_variables** の説明を表示するには、次のように入力します。

   ```
   help set_variables
   ```

ignore サブコマンド

ignore [*SignalNumber* | *SignalName*]

ignore サブコマンドは、指定するシグナルがアプリケーション・プログラムに送信される前に、そのシグナルのトラッピングを停止します。このサブコマンドは、デバッグ中のアプリケーション・プログラムが割り込みなどのシグナルを処理する場合に有効です。

トラップされるシグナルは、以下のいずれかの方法で指定できます。

- 番号。*SignalNumber* パラメーターを使用。
- 名前。*SignalName* パラメーターを使用。

シグナル名は、大文字と小文字の区別はありません。**SIG** プレフィックスはオプションです。

SignalNumber パラメーターおよび *SignalName* パラメーターを指定すると、デフォルトとして、**SIGHUP**、**SIGCLD**、**SIGALRM**、**SIGKILL** を除くすべてのシグナルがトラップされます。 **dbx** デバッグ・プログラムは、デバッガーの外側のプロセスから発生した **SIGTRAP** シグナルを、無視することはできません。引数を指定しない場合は、現在無視されているシグナルのリストが表示されます。

例

アプリケーション・プログラムに送信されるアラーム・クロック・タイムアウト・シグナルを **dbx** で無視させるには、次のように入力します。

```
ignore alrm
```

catch サブコマンドのセクションを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『シグナルの処理』も参照してください。

kthread サブコマンド

kthread [**raw**] [**info** | **ru**] [*tid*]

kthread サブコマンドは、カーネル・スレッドについての情報を表示します。 **raw** オプションを使用すると、より読みやすいフォーマットで表示できるかどうかにかかわらず、すべての出力が 16 進数で表示されます。引数を使用しないと、すべてのカーネル・スレッドについての要約情報が印刷されます。数字スレッド ID を提供すると、**dbx** コマンドは単一スレッドについての情報を示します。また、**info** オプションを使用すると、ユーザー・スレッド構造から、スレッドについてのより詳細な出力が出されます。 **ru** オプションは、`ti_ru` データ・メンバーを表示します。この中には、リソース使用情報が入っています。

ユーザー・スレッドの詳細については、**thread subcommand** を参照してください。

例

1. 現在実行しているスレッドについての情報を検出するには、まずコマンド・ラインに以下を入力することによって、すべてのスレッドについての情報を入手してください。

   ```
   kthread
   ```

   dbx コマンドがプロセスを停止する直前に実行していた (または実行可能だった) スレッドはアスタリスクでマークされています。その出力に基づいて正しいスレッド ID を選択し、入力します。

   ```
   kthread info tid
   ```
2. すべてのスレッドについてのリソース情報を 16 進数で表示するには、次のように入力します。

   ```
   kthread raw ru
   ```

limitbp サブコマンド

limitbp (*bp1*, *Limit*) [(*bp2*, [**+**] *Limit*) ...]

limitbp サブコマンドは、ブレークポイントが指定された回数実行された場合に限りデバッグ・プログラムの実行を停止するように **dbx** コマンドに指示します。制限値の前に「**+**」文字がある場合は、そのイベントの制限値がサブコマンドに指定された制限値と、イベントが既に実行された回数の合計に変更されます。すなわち、**dbx** コマンドは、**limitbp** サブコマンドが既に実行された後に、指定された *Limit* だけブレークポイントを実行する場合にデバッグ・プログラムの実行を停止します。

例

1. ブレークポイント 1 を 10 回実行する場合にデバッグ・プログラムの実行を停止するように **dbx** コマンドに指示するには、次のように入力します。

   ```
   limitbp (1, 10)
   ```
2. ブレークポイント 1 を 15 回実行するか、またはブレークポイント 2 を 20 回実行する (あるいはその両方) 場合にデバッグ・プログラムの実行を停止するように **dbx** コマンドに指示するには、次のように入力します。

   ```
   limitbp (1, 15) (2, 20)
   ```
3. **limitbp** サブコマンドが実行された後に、ブレークポイント 1 を 20 回実行する場合にデバッグ・プログラムの実行を停止するように **dbx** コマンドに指示するには、次のように入力します。

   ```
   limitbp (1, +20)
   ```

list サブコマンド

list [*Procedure* | *SourceLine-Expression* [*,SourceLine-Expression*] | **at** *Address*]

list サブコマンドは、ソース・ファイルの指定行番号を表示します。表示される行番号を指定する方法は、以下のいずれかです。

- *Procedure* パラメーターを使用して、プロシージャーを指定する方法。

  この場合には、**list** サブコマンドは、指定したプロシージャーの数行前から、リスト・ウィンドウがいっぱいになるまで行を表示します。
- *SourceLine-Expression* パラメーターを使用して、開始ソース行番号と終了ソース行番号を指定する方法。

  SourceLine-Expression パラメーターには、有効な行番号、オプションの + (正符号) または - (負符号)、整数の順に指定する必要があります。また、$ (ドル記号) の *SourceLine* は、現在行の番号を示すのに使用でき、 @ (アットマーク) の *SourceLine* は、表示される次の行番号を示すのに使用できます。
- *$listwindow* の内部 **dbx** 変数を指定する方法。

  パラメーターの指定をしないで、**list** サブコマンドを使用すると、*$listwindow* 変数で指定された行数が、現行ソース行から始まって出力されます。デフォルトの行数を変更するには、*$listwindow* 変数を必要な行数に設定します。*$listwindow* 変数は、特殊なデバッグ・プログラム変数です。初めは *$listwindow* 変数は 10 に設定されています。

2 番目のソース行を省略すると、1 行目のみが表示されます。

最初に指定した行番号から 2 番目に指定した行番号まで (両方の行番号を含む) のすべての行が表示されます。

list サブコマンドで **at** パラメーターの後にアドレスを指定すると、**list** サブコマンドは、指定されたアドレスに対応するソース行を表示します。アドレスは、10 進または 16 進の符号なし整数、あるいは *$iar*、*$tfiar*、および *$tfhar* などのレジスターまたはデバッグ変数に対応するニーモニックとして指定できます。

例

1. 現在のファイルの、1 行目から 10 行目までを表示するには、次のように入力します。

   ```
   list 1,10
   ```
2. main プロシージャーの前後を、 10 行、すなわち、$listwindow 行分表示するには、次のように入力します。

   ```
   list main
   ```
3. 現在行の前後の 11 行を表示するには、次のように入力します。

   ```
   list $-5,$+5
   ```
4. *SourceLineExpression* の式の中で、加算および減算などの簡単な整数式を使用することもできます。以下に例を示します。

   ```
   (dbx) list $
   4 {

   (dbx) list 5
   5 char i = '4';

   (dbx) list sub
   23 char *sub(s,a,k)
   24 int a;
   ```

```
25 enum status k;  .  .  .

(dbx) move
25
(dbx) list @ -2
23 char *sub(s,a,k)
```

5. 特定のアドレスに対応するソース行を表示することができます。次に例を示します。

```
(dbx) r
[1] stopped in main at line 5
    5 int i, sum = 0;

(dbx) list at $iar
source file: "tt.c"
    5 int i, sum = 0;
    6 int last = 0;
    7
    8 scanf("%d", &last);
    9
    10 for ( i = 1; i &lt;=last; i++ ) {
    11 sum += i;
    12 }
    13 printf("sum = %d¥n", sum);
    14

(dbx) list at ($iar+16)
source file: "tt.c"
    8 scanf("%d", &last);
    9
    10 for ( i = 1; i <= last; i++ ) {
    11 sum += i;
    12 }
    13 printf("sum = %d¥n", sum);
    14
    15 return 0;
    16 }
```

edit サブコマンド、 **listi** サブコマンド、および **move** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『現在のファイルの表示』も参照してください。

listi サブコマンド

listi [*Procedure* | **at** *SourceLine* | *Address* [**,** *Address*]]

listi サブコマンドは、ソース・ファイルの指定した一連の命令を表示します。表示する命令は、以下の方法で指定します。

- *Procedure* パラメーターを指定すると、 **listi** サブコマンドは、リスト・ウィンドウがいっぱいになるまで、指定するプロシージャーの最初から命令を表示します。
- **at** *SourceLine* フラグを使用すると、**listi** サブコマンドは指定したソース行から始めて、リスト・ウィンドウがいっぱいになるまで命令の表示を続けます。 *SourceLine* 変数は、整数として、または : (コロン) と整数が続くファイル名文字列として指定できます。
- *Address* パラメーターを使用して、開始アドレスと終了アドレスを指定すると、この 2 つのアドレス間のすべての命令 (両方のアドレスの命令を含む) が表示されます。

listi サブコマンドをフラグまたはパラメーターを指定しないで使用すると、次の **$listwindow** 命令が表示されます。リスト・ウィンドウの現行サイズを変更するには、 **set $listwindow=***Value* サブコマンドを使用します。

逆アセンブリー・モード

dbx プログラムは、POWER® family または PowerPC® のどちらのアーキテクチャーの命令も逆アセンブルすることができます。デフォルト・モードでは、 **dbx** プログラムはこのプログラムが動作しているアーキテクチャーの命令を表示します。

dbx コマンドの **set** サブコマンドの **$instructionset** 変数および **$mnemonics** 変数を使用すると、デフォルトの逆アセンブリー・モードを指定変更できます。詳細については、**dbx** コマンドの **set** サブコマンドを参照してください。

フラグ

項目	説明
at *SourceLine*	リストを開始するソース行を指定します。

例

1. 次の 10 個の命令、すなわち `$listwindow` 個の命令を表示するには、次のように入力します。
   ```
   listi
   ```
2. ソース行の `10` 行目以降のマシン・インストラクションを表示するには、次のように入力します。
   ```
   listi at 10
   ```
3. `sample.c` ファイルのソース行の、 `5` 行目以降のマシン・インストラクションを表示するには、次のように入力します。
   ```
   listi at "sample.c":5
   ```
4. アドレス `0x10000400` と、 `0x10000420` の間の命令を表示するには、次のように入力します。
   ```
   listi 0x10000400, 0x10000420
   ```

list サブコマンド、および **set** サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『dbx によるマシン・レベルのデバッグ』も参照してください。

malloc サブコマンド

malloc [> *File*]

オプションを指定しない **malloc** サブコマンドは、使用可能なオプション、割り振りポリシー、およびプロセス起動以来の malloc 使用統計要約を印刷します。

malloc [**allocation** [{ *address* | *size* | *heap* | *pid* | *tid* | *time* } { "<" | "==" | ">" "!=" | "~="]} *Value*]] [> *File*]

malloc サブコマンドに **allocation** オプションを指定すると、プロセスが現在保持しているすべての割り振りのリストがソートして表示されます。オプションの `attribute RELOP value` 引数を使用すると、アクティブになっている割り振りからさらに選択を狭めることができます。

malloc [**freespace** [{ *address* | *size* | *heap* } { "<" | "==" | ">" | "!=" | "~="]} *Value*]] [> *File*]

malloc サブコマンドに **freespace** オプションを指定すると、プロセス・ヒープで使用可能なすべてのフリー・スペースのリストがソートして表示されます。オプションの `attribute RELOP value` 引数を使用すると、フリー・スペース・ノードの選択をさらに狭めることができます。

注**: ~=** 演算子は、address オプションを指定した場合にのみ使用できます。この演算子は、指定されたアドレスが属するフリー・スペース・ノードまたは割り当てノードを取り出すために使用されます。

malloc *address*

アドレスを指定した **malloc** サブコマンドにより、アドレスのノード詳細が表示されます。このアドレスは、割り当てられたノードまたはフリー・ノードの開始アドレスである必要はありません。

フラグ

項目	説明
> *File*	指定したファイルに出力をリダイレクトします。

詳しくは、「プログラミングの一般概念*:* プログラムの作成およびデバッグ」の『malloc サブシステムを使用したシステム・メモリーの割り当て』を参照してください。

map サブコマンド

map { [*Format*] [**entry** *ModuleNumber* [, *ModuleNumber*] | *Address* | *SymbolName*] [**for** **$t***threadnumber*] [**>** *File*] }

map サブコマンドは、アプリケーションのロードされた部分の特性を表示します。この情報には、ロードされた各モジュールについて、モジュール名、メンバー名、テキストの始まり、テキストの終わり、テキスト長、データの始まり、データの終わり、データ長、TLS データの始まり、TLS データの終わり、TLS データ長、およびファイル・ディスクリプターが含まれます。表示するエントリーは、次のように指定できます。

- *ModuleNumber* パラメーターを使用して、単一エントリーを指定する方法。
- コンマで区切った 2 つの *ModuleNumber* パラメーターを使用して、エントリーの範囲を指定する方法。
- *Address* パラメーターを使用して、ロードされているモジュールに解決されるアドレスを指定する方法。
- *SymbolName* パラメーターを使用して、ロードされているモジュールに解決されるシンボル名を指定する方法。

上記のいずれも指定しないで呼び出されると、map サブコマンドは、そのアプリケーションのロードされたすべての部分についての情報を表示します。

Format 引数は、ロードされたモジュールの説明を出力するモードを指定します。 *Format* 引数に指定できる値には、以下のものがあります。

項目	説明
abbr	省略出力モードを指定します。この場合、ロードされた各モジュールは 1 行で構成され、そのモジュールのエントリー番号、モジュール名、オプションとしてメンバー名を含みます。
normal	通常出力モードを指定します。この情報には、ロードされた各モジュールについて、エントリー番号、モジュール名、メンバー名、テキストの始まり、テキスト長、データの始まり、データ長、およびファイル・ディスクリプターが含まれます。ロードされたモジュールに TLS データがある場合、TLS データの始まりおよび TLS データ長も表示されます。
raw	ロウ出力モードを指定します。この場合、出力は、ロードされた各モジュールについて不定形式の 1 行で構成され、その中には、エントリー番号、モジュール名とオプションのメンバー名、テキストの始まり、テキストの終わり、テキスト長、データの始まり、データの終わり、データ長、およびファイル・ディスクリプターが、スペースで分離されたフィールドとして含まれます。ロードされたモジュールに TLS データがある場合、TLS データの始まり、TLS データの終わり、および TLS データ長も表示されます。

項目	説明
verbose	詳細出力を指定します。この情報には、ロードされた各モジュールについて、エントリー番号、モジュール名、メンバー名、テキストの始まり、テキストの終わり、テキスト長、データの始まり、データの終わり、データ長、およびファイル・ディスクリプターが含まれます。ロードされたモジュールに TLS データがある場合、TLS データの始まり、TLS データの終わり、および TLS データ長も表示されます。

Format パラメーターを指定しないと、**dbx** コマンドは **$mapformat** 内部変数の値を使用します。*Format* パラメーターの指定がなく、かつ、**$mapformat** が設定解除されている場合は、**dbx** コマンドは、ロードされたモジュールの情報を通常モードで表示します。

ロードされたモジュールに TLS データがある場合、指定されたスレッドの TLS データ情報が表示されます。スレッドが指定されていない場合は、現行スレッドが使用されます。

フラグ

項目	説明
>*File*	指定したファイルに出力をリダイレクトします。
entry *ModuleNumber* **[** *, ModuleNumber* **]**	表示するモジュールまたはモジュールの範囲を指定します。
for $t *threadnumber*	スレッド番号を指定します。

例

1. ロードされたすべてのモジュールを省略モードでリストするには、次のように入力します。

   ```
   map abbr
   ```

2. ロードされたモジュール 3 から 5 を詳細モードでリストするには、次のように入力します。

   ```
   map verbose entry 3,5
   ```

3. アドレス 0x20001000 を含むロードされたモジュールをリストするには、次のように入力します。

   ```
   map 0x20001000
   ```

4. 変数 example を含むロードされたモジュールをリストするには、次のように入力します。

   ```
   map example
   ```

5. ロードされたモジュールをスレッド 2 のモジュールの TLS データ情報とともに通常モードでリストするには、次のように入力します。

   ```
   map normal for $t2
   ```

詳しくは、**$mapformat** 内部変数を参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『dbx によるマシン・レベルのデバッグ』も参照してください。

move サブコマンド

move *SourceLine*

move サブコマンドは、次に表示される行を、 *SourceLine* パラメーターで指定した行に変更します。このサブコマンドは、@ (アットマーク) 変数の値を変更します。

SourceLine 変数は、整数として、または : (コロン) および整数が後ろに続くファイル名文字列として指定できます。

例

1. 次に表示される行を 12 行目に変更するには、次のように入力します。

   ```
   move 12
   ```

2. 次に表示される行を、 sample.c ファイルの 5 行目に変更するには、次のように入力します。

```
move "sample.c":5
```

list サブコマンドのセクションを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『現在のファイルの表示』も参照してください。

multproc サブコマンド

multproc [on | parent | child | off]

multproc サブコマンドは、fork 先と fork 元のプロセスが作成されたときに、**dbx** デバッグ・プログラムの動作を指定します。 **on** フラグを使用すると、fork の子パスをデバッグするために、新しい **dbx** セッションが作成されるように指定することができます。元の **dbx** は、親パスのデバッグを続行します。**parent** フラグと **child** フラグを使用すると、検索対象とする fork の単一パスを指定することができます。 **off** を除くすべてのフラグにより、 **dbx** が fork 元のプロセスを追跡できるようになります。 **off** フラグを指定すると、マルチプロセス・デバッグが使用不可になります。フラグを指定しないと、**multproc** サブコマンドはマルチプロセス・デバッグの状況を戻します。

dbx プログラムは、マルチプロセスのデバッグに X Window System を使用します。**dbx** プログラムは、マルチプロセスに必要な数だけウィンドウをオープンします。各子ウィンドウの名称は、子プロセスのプロセス ID (pid) です。プロセス間で切り替えるには、X Window System の取扱方法に従い、**dbx** コマンド・セッションが表示されているウィンドウをアクティブにします。使用中のシステムが X Window System をサポートしていない場合は、デバッガーが fork を実行したときに警告メッセージが表示され、**dbx** プログラムは親プロセスのデバッグのみを続行します。また、マルチプロセスのデバッグは、以下の原因でうまくいかない場合もあります。

- **dbx** プログラムが、 X Window System 環境で動作していない場合。
- X Window System は動作しているが、 **dbx** のグローバル **$xdisplay** 変数が、有効な表示名に設定されていない場合。**$xdisplay** 変数は、シェル **DISPLAY** 環境変数に対して初期化されます。 **dbx** サブコマンドの **set** *Name=Expression* を使用すると、表示名の値を変更できます。
- **/tmp** ディレクトリーが、デバッグ・プログラムに対して読み取りまたは書き込みアクセスを許可しない場合。**dbx** プログラムは、X Window 環境を制御するときに、このディレクトリー内に小さなスペースを必要とします。
- システムのリソース不足のため、新しい X Window に対応できない場合。

$xdisplay がリモート・ディスプレイに設定されていると、新しく作成した X Window が表示されないことがあります。 **$xdisplay** が正しく設定されていないと、X Window System または他のシステム・リソースにより障害の原因が表示されます。

dbx プログラムは障害のタイプを識別しませんが、サブコマンドが失敗すると次のメッセージが送信されます。

```
Warning: dbx subcommand multiproc fails. dbx
continued with multproc disabled.
```

新しく作成したウィンドウのユーザー定義設定は、 **.Xdefaults** ファイル内でアプリケーション名 **dbx_term** を付けて定義することができます。

フラグ

項目	説明
on	マルチプロセス・デバッグを使用可能にします。
off	マルチプロセス・デバッグを使用不可にします。

例

1. マルチプロセス・デバッグの状況を確認するには、次のように入力します。

   ```
   multproc
   ```

2. マルチプロセス・デバッグを使用可能にするには、次のように入力します。

   ```
   multproc on
   ```

3. マルチプロセス・デバッグを使用不可にするには、次のように入力します。

   ```
   multproc off
   ```

screen サブコマンドと、 **fork** サブルーチンを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『複数のプロセスを含むプログラムのデバッグ』も参照してください。

mutex サブコマンド

mutex [lock | **unlock** | **thnum** | **utid** | *MutexNumber ...* **]**

mutex サブコマンドは、mutex に関する情報を表示します。*MutexNumber* パラメーターを指定すると、**mutex** サブコマンドは指定した mutex に関する情報を表示します。フラグまたはパラメーターを指定しなければ、 **mutex** サブコマンドはすべての mutex に関する情報を表示します。

各 mutex について表示される情報は、以下のとおりです。

項目	説明
mutex	mutex のシンボル名を $m*MutexNumber* フォーマットで示します。
type	mutex のタイプを示します。タイプには、non-rec (非再帰的)、 recursi (再帰的)、 fast があります。
obj_addr	mutex のメモリー・アドレスを示します。
lock	mutex のロック状態を示します。mutex がロックされていれば yes、ロックされていなければ no です。
owner	mutex がロックされている場合は、その mutex を保持しているユーザー・スレッドのシンボル名を示します。
blockers	この mutex 変数でブロックされるユーザー・スレッドをリストします。

注**: dbx** デバッグ・プログラムの **print** サブコマンドは、シンボルによる mutex 名を認識し、対応するオブジェクトの状況を表示するために使用できます。

フラグ

項目	説明
lock	ロックされている mutex に関する情報を表示します。
unlock	ロックされていない mutex に関する情報を表示します。
thnum	特定のスレッドが保持するすべての mutex の情報を表示します。
utid	ユーザー・スレッド ID に一致するユーザー・スレッド ID をもつユーザー・スレッドが保持する、すべての mutex の情報を表示します。

例

1. すべての mutex に関する情報を表示するには、次のように入力します。

   ```
   mutex
   ```

2. すべてのロック状態の mutex に関する情報を表示するには、次のように入力します。

   ```
   mutex  lock
   ```

3. mutex 番号 4、5、6 に関する情報を表示するには、次のように入力します。

   ```
   mutex 4 5 6
   ```

   出力は以下のようになります。

   ```
   mutex   obj_addr         type     lock owner  blockers
   $m4    0x20003274        non-rec   no
   $m5    0x20003280        recursi   no
   $m6    0x2000328a        fast      no
   ```

4. スレッド 1 が保持するすべての mutex の情報を表示するには、次のように入力します。

   ```
   mutex thnum 1
   ```

5. 0x0001 というユーザー・スレッド ID のスレッドが保持する、すべての mutex の情報を表示します。

   ```
   mutex utid 0x0001
   ```

attribute サブコマンド、 **condition** サブコマンド、 **print** サブコマンド、 **thread** サブコマンドを参照してください。

また、「プログラミングの一般概念*:* プログラムの作成およびデバッグ」の『mutex の使用』も参照してください。

next サブコマンド

next [*Number*]

next サブコマンドは、アプリケーション・プログラムを次のソース行まで実行します。 *Number* パラメーターは、**next** サブコマンドの実行回数を指定します。*Number* パラメーターを指定しないと、**next** は 1 回だけ実行されます。

マルチスレッド・アプリケーション・プログラム内で **next** サブコマンドを使用すると、操作中にすべてのユーザー・スレッドが実行されますが、プログラムは実行中のスレッドが指定のソース行に達するまで実行を継続します。実行中のスレッドのみを進行させる場合は、 **set** サブコマンドを使用して変数 **$hold_next** を設定します。この変数を設定すると、実行中のスレッドがブロック化されているスレッドの 1 つにより保持されているロックを待機する可能性があるため、デッドロックが発生することがあります。

例

1. 次のソース行まで実行を継続するには、次のように入力します。

   ```
   next
   ```

2. 現行ソース行の次から数えて 3 番目のソース行まで実行を継続するには、次のように入力します。

   ```
   next 3
   ```

cont サブコマンド、 **goto** サブコマンド、 **nexti** サブコマンド、 **set** サブコマンド、および **step** サブコマンドを参照してください。

nexti サブコマンド

nexti [*Number*]

nexti サブコマンドは、次の命令までアプリケーション・プログラムを実行します。 *Number* パラメーターは、**nexti** サブコマンドの実行回数を指定します。 *Number* パラメーターを指定しないと、**nexti** は 1 回だけ実行されます。

マルチスレッド・アプリケーション・プログラム内で **nexti** サブコマンドを使用すると、操作中にすべてのユーザー・スレッドが実行されますが、プログラムは実行中のスレッドが指定のマシン・インストラクションに達するまで実行を継続します。実行中のスレッドのみを進行させる場合は、 **set** サブコマンドを使用して変数 **$hold_next** を設定します。この変数を設定すると、実行中のスレッドがブロック化されているスレッドの 1 つにより保持されているロックを待機する可能性があるため、デッドロックが発生することがあります。

例

1. 次のマシン・インストラクションまで実行を続行するには、次のように入力します。

   ```
   nexti
   ```

2. 現在のマシン・インストラクションの次から数えて 3 番目のマシン・インストラクションまで実行を継続するには、次のように入力します。

   ```
   nexti 3
   ```

gotoi サブコマンド、 **next** サブコマンド、 **set** サブコマンド、および、 **stepi** サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『マシン・レベルでのプログラムの実行』も参照してください。

onceblock サブコマンド

onceblock [**uninit** | **done**]

onceblock サブコマンドは、**pthread_once** ルーチンを使用して登録された初期設定コードのブロックについての情報を表示します。引数を指定しない場合は、すべての登録済み once block についての情報が表示されます。オプションの **uninit** および **done** のフラグは、それぞれ、実行されていない、または、既に実行された once block のみを表示します。ただし、数字の once ID を指定した場合は、単一の once block についての情報が表示されます。

注**:** ライブ・プロセスのデバッグ中に **onceblock** サブコマンドが機能するためには、環境変数 AIXTHREAD_ONCE_DEBUG を `ON` に設定しておく必要があります。同様に、コア・ファイルのデバッグで、プロセスの実行時にこの変数が ON になっていなかった場合は、**onceblock** サブコマンドは情報を取得することができません。

例

1. どの once block もまだ実行されていないかどうかを検出するには、次のように入力します。

   ```
   onceblock uninit
   ```

plugin サブコマンド

plugin [*Name* [*Command*]]

plugin サブコマンドは、*Command* パラメーターにより指定されたコマンドを、*Name* パラメーターで指定されたプラグインに渡します。パラメーターを指定しないと、使用可能なすべてのプラグインの名前が表示されます。

例

1. 使用可能なすべてのプラグインをリストするには、次のように入力します。

   ```
   plugin
   ```

2. 「sample」という名前のプラグインのサブコマンド「help」を開始するには、次のように入力します。

```
plugin sample help
```

3. 「xyz」という名前のプラグインのサブコマンド「interpret 0x20000688」を開始するには、次のように入力します。

```
plugin xyz interpret 0x20000688
```

pluginload サブコマンド、および **pluginunload** サブコマンドを参照してください。また、「プログラミングの一般概念」の『dbx プラグイン・フレームワーク用の開発』のセクションも参照してください。

pluginload サブコマンド

pluginload *File*

pluginload サブコマンドは、*File* パラメーターで指定されたプラグインをロードします。 *File* パラメーターでプラグインのパスを指定する必要があります。

注**:** デフォルトの **dbx** コマンドは 64 ビット・プロセスであるため、32 ビットのプラグインをロードするには **dbx32** という 32 ビット版の **dbx** コマンドを使用する必要があります。

例

「/home/user/dbx_plugins/libdbx_sample.so」にある「sample」という名前のプラグインをロードするには、次のように入力します。

```
pluginload /home/user/dbx_plugins/libdbx_sample.so
```

plugin サブコマンド、および **pluginunload** サブコマンドを参照してください。また、「プログラミングの一般概念」の『dbx プラグイン・フレームワーク用の開発』も参照してください。

pluginunload サブコマンド

pluginunload *Name*

pluginunload サブコマンドは、*Name* パラメーターで指定されたプラグインをアンロードします。

例

「sample」という名前のプラグインをアンロードするには、次のように入力します。

```
pluginunload sample
```

plugin サブコマンド、および **pluginload** サブコマンドを参照してください。また、「プログラミングの一般概念」の『dbx プラグイン・フレームワーク用の開発』も参照してください。

print サブコマンド

print *Expression ...*

print *Procedure* ([*Parameters*])

print サブコマンドは、以下のいずれかの操作を実行します。

- *Expression* パラメーターにより指定された、式のリストの値を表示します。
- *Procedure* パラメーターにより指定されたプロシージャーを実行し、そのプロシージャーの戻り値を表示します。指定されたパラメーターはプロシージャーに渡されます。

例

1. x の値と、 y の値を、2 ビット左にシフトして表示するには、次のように入力します。

   ```
   print x, y << 2
   ```

2. 引数 0 で、 sbrk ルーチンを呼び出すことにより戻される値を表示するには、次のように入力します。

   ```
   print sbrk(0)
   ```

assign サブコマンド、 **call** サブコマンド、および、 **set** サブコマンドを参照してください。

printbp サブコマンド

printbp [*bp1*] [*bp2*] ... | **all**

printbp サブコマンドは、各ブレークポイントまたはすべてのサブコマンドが実行された回数、およびブレークポイントの制限の詳細 (制限が設定されている場合) を印刷するように **dbx** コマンドに指示します。

例

1. ブレークポイント 1 が実行された回数、および設定された制限の詳細を印刷するように **dbx** コマンドに指示するには、次のように入力します。

   ```
   printbp 1
   ```

2. ブレークポイント 1 と 2 が実行された回数を印刷し、ブレークポイント 1 と 2 を実行できる回数を制限する (制限が設定されている場合) ように **dbx** コマンドに指示するには、次のように入力します。

   ```
   printbp 1, 2
   ```

3. すべてのブレークポイントが実行された回数、およびブレークポイントの制限の詳細 (該当する場合) を印刷するように **dbx**コマンドに指示するには、次のように入力します。

   ```
   printbp all
   ```

proc サブコマンド

proc [**raw**] [**cred** | **cru** | **ru** | **sigflags** | **signal**]

proc サブコマンドは、このプロセスについての情報を表示します。 **raw** オプションを使用すると、もっと読みやすい方式で値を解釈することはせず、出力をロウ 16 進数で表示します。追加の引数を指定せずに **proc** サブコマンドを使用すると、このプロセスについての一般情報が、ユーザー・プロセス・データ構造に保管されているとおりに出力されます。 **cred** オプションは、pi_cred データ・メンバーの内容を表示します。これは、プロセスのクリデンシャルを記述するものです。 **cru** と **ru** のオプションは、データ・メンバー pi_cru と pi_ru をそれぞれ表示します。これらには、リソース使用情報が入っています。 **sigflags** と **signal** のオプションは、pi_sigflags と pi_signal のデータ・メンバー内に入っているとおりに、現行のシグナルの状況と登録済みのシグナル・ハンドラーに関する情報を表示します。

例

1. 現行プロセス (またはコア・ファイル) についてのリソース使用について、ロウ 16 進数で表示するには、次のように入力します。

   ```
   proc raw ru
   ```

2. シグナル・ハンドラーの情報を表示するには、次のように入力します。

   ```
   proc signal
   ```

prompt サブコマンド

prompt [**"***String***"**]

prompt サブコマンドは、 **dbx** コマンド・プロンプトを *String* パラメーターにより指定された文字列に変更します。

例

プロンプトを dbx> に変更するには、次のように入力します。

```
prompt "dbx>"
```

プログラミングの一般概念: プログラムの作成およびデバッグ の『新規 dbx プロンプトの定義』を参照してください。

quit サブコマンド

quit

quit サブコマンドは、 **dbx** デバッグ・セッションで実行しているすべてのプロセスを終了します。

detach サブコマンドを参照してください。

registers サブコマンド

registers [ALL | *$t*threadnumber ...] [**>***File*]

registers サブコマンドは、汎用レジスター、システム制御レジスター、浮動小数点レジスター、ベクトル・レジスター、および現行命令レジスターの値を表示します。

- 汎用レジスターは、 **$r***Number* 変数により示されます。ここでは、*Number* パラメーターはレジスターの番号を示します。

  注**:** レジスターの値は、**0xdeadbeef** 16 進値に設定される場合があります。 **0xdeadbeef** 16 進値は、プロセスの初期設定時に汎用レジスターに割り当てられる初期値です。
- 浮動小数点レジスターは、 **$fr***Number* 変数により示されます。デフォルトでは、浮動小数点レジスターは表示されません。浮動小数点レジスターを表示するには、**dbx** サブコマンドの **unset $noflregs** を使用します。
- ベクトル・レジスターは、**$vr***Number* 変数により示されます。**$novregs** 内部変数は、ベクトル・レジスターを表示するかどうかを制御します。 **$novregs** 変数はデフォルトで設定され、ベクトル・レジスターは表示されません。 **$novregs** が設定されず、ベクトル・レジスターが有効 (ベクトル対応プロセッサーでのプログラムのデバッグ、またはベクトル・レジスター状態を含むコア・ファイルの分析) の場合は、すべてのベクトル・レジスター (vr0–vr31、vrsave、vscr) が表示されます。ベクトル・レジスターは、タイプ別に参照することもできます。例えば、**$vrNf** (float)、**$vrNs** (short)、および **$vrNc** (char) のベクトル・レジスター変数と **print** および **assign** サブコマンドを併用して、タイプ別にベクトル・レジスターを表示および設定することができます。
- ベクトル・スカラー・レジスターは、**$vsr***Number* 変数により示されます。デフォルトでは、ベクトル・スカラー・レジスターは表示されません。(ベクトル・スカラー対応プロセッサーでのプログラムのデバッグ、またはベクトル・スカラー・レジスター状態を含むコア・ファイルの分析において) ベクトル・スカラー・レジスターが有効な場合に必ずベクトル・スカラー・レジスターを表示するには、**$novsregs** 変数を設定解除します。ベクトル・スカラー・レジスターは従来の浮動小数点レジスターおよびベクトル・レジスターのスーパーセットであるため、デバッグ変数 **$novsregs** を設定解除すると、

ベクトル・スカラー・レジスター状態が有効な場合には常に **$noflregs** および **$novsregs** より優先します。**registers** サブコマンドは、ベクトル・スカラー・レジスターを従来のレジスター別名と一緒に中括弧で囲んで表示します。浮動小数点レジスターの別名は、下位 64 ビットにのみ対応します。ベクトル・スカラー・レジスターは、ベクトル・レジスターと同様にタイプ別に参照することもできます。例えば、**$vsrNf** (float)、**$vsrNs** (short)、**$vsrNc** (char)、**$vsrNg** (double) および **$vsrNll** (long long) のベクトル・スカラー・レジスター変数と print および assign サブコマンドを併用して、タイプ別にベクトル・スカラー・レジスターを表示および設定することができます。

- マルチスレッド環境では、オプション ALL を指定すると、使用可能なすべてのスレッドのレジスターの詳細が表示されます。各スレッドのレジスターの詳細を表示するには、registers サブコマンドを使用してスレッド番号を指定します。registers サブコマンドをオプションなしで使用すると、現行スレッドのレジスターが表示されます。

注**:** 現行スレッドがカーネル・モードになっている場合は、 **registers** サブコマンドはレジスターを表示できません。

フラグ

項目	説明
>*File*	指定したファイルに出力をリダイレクトします。

set サブコマンド、および **unset** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『マシン・レジスターの使用』も参照してください。

例

スレッド $t1、$t2、および $t3 のレジスターの詳細を表示するには、次のように入力します。

```
registers $t1 $t2 $t3
```

set サブコマンド、および **unset** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『マシン・レジスターの使用』も参照してください。

rerun サブコマンド

rerun [*Arguments*] [**<** *File*] [**>** *File*] [**> >** *File*] [**2>** *File*] [**2> >** *File*] [**>&** *File*] [**> >&** *File*]

rerun サブコマンドは、オブジェクト・ファイルの実行を開始します。 *Arguments* は、コマンド・ラインの引数として渡されます。*Arguments* パラメーターを指定しないと、最後に実行した **run** サブコマンドまたは **rerun** サブコマンドの引数が再使用されます。

フラグ

項目	説明
<*File*	入力が *File* から受け取られるように、入力をリダイレクトします。
>*File*	出力を *File* にリダイレクトします。
> >*File*	リダイレクトされた出力を *File* に追加します。
2>*File*	標準エラーを *File* にリダイレクトします。
2> >*File*	リダイレクトされた標準エラーを *File* に追加します。
>&*File*	出力および標準エラーを *File* にリダイレクトします。
> >&*File*	出力および標準エラーを *File* に追加します。

run サブコマンドを参照してください。

resource サブコマンド

resource { **owner** | **waiter** } [**all** | *pthread id*]

resource サブコマンドは、pthread が現在保持している、または待機しているリソースについての情報を表示します。 1 番目の引数は必須であり、リソースを所有している pthread、またはリソースを待機している pthread のどちらを表示したいかを示します。 2 番目の引数は、すべての pthread を指示するため、または特定の pthread を指示するために使用できます。どちらも指定しないと、現行の pthread がある場合は、それに関係する情報のみが表示されます。

注**: resource** サブコマンドは、いくつかのデバッグ用環境変数を `ON` に設定して実行しているデバッグ・プロセスにのみ役立ちます。これらの環境変数の中には、 AIXTHREAD_MUTEX_DEBUG、 AIXTHREAD_COND_DEBUG、 AIXTHREAD_RWLOCK_DEBUG、 AIXTHREAD_READ_OWNER および AIXTHREAD_WAITLIST_DEBUG があります。ライブ・プロセスのデバッグ中にこれらの変数がオンになっていない場合、またはデバッガー・コア・ファイルの生成時にこれらの変数がオンになっていなかった場合は、**resource** サブコマンドは、より少ない情報しか検索できないか、または情報をまったく検索できません。これらの機能は、使用するとパフォーマンスが低下するおそれがあるので、デバッグの目的にだけ活動化するようにお勧めします。

例

1. 現行の pthread が何らかのリソースを保持しているか確認するには、次のように入力します。
   ```
   resource owner
   ```
2. いずれかの pthreads が待機しているリソースを表示するには、次のように入力します。
   ```
   resource waiter all
   ```

return サブコマンド

return [*Procedure*]

return サブコマンドは、 *Procedure* パラメーターにより指定されたプロシージャーに戻るまで、アプリケーション・プログラムを実行させます。 *Procedure* パラメーターを指定しないと、現行プロシージャーが戻ったときに実行が停止します。

例

1. 呼び出し側のルーチンまで実行を継続するには、次のように入力します。
   ```
   return
   ```
2. `main` プロシージャーまで実行を継続するには、次のように入力します。
   ```
   return main
   ```

rwlock サブコマンド

rwlock [read | write | *RwlockNumber*....]

rwlock サブコマンドは、rwlock に関する情報を表示します。 *RwlockNumber* パラメーターを指定すると、 **rwlock** サブコマンドは指定された rwlock に関する情報を表示します。フラグまたはパラメーターが指定されない場合、**rwlock** サブコマンドはすべての rwlock に関する情報を表示します。

各 **rwlock** に関する情報を、以下に示します。

項目	説明
rwl	rwlock のシンボル名を $rw *RwlockNumber* フォーマットで示します。
flag_value	フラグ値を示します。
owner	rwlock のオーナーを示します。
status	rwlock の保持者を示します。値は、read (リーダーが保持する場合)、write (ライターが保持する場合)、free (空きの場合) です。
wsleep[#]	書き込みでのスレッドのブロッキングを示します。 # は、書き込みでのスレッドのブロッキングの合計数を示します。
rsleep[#]	読み取りでのスレッドのブロッキングを示します。 # は、読み取りでのスレッドのブロッキングの合計数を示します。

注**: dbx** デバッグ・プログラムの **print** サブコマンドは、シンボルによる rwlock 名を認識し、対応するオブジェクトの状況を表示するために使用できます。

フラグ

項目	説明
read	状況が読み取りモードにあるすべての rwlock に関する情報を表示します。
write	状況が書き込みモードにあるすべての rwlock に関する情報を表示します。

例

1. すべての rwlock に関する情報を表示するには、次のように入力します。

   ```
   rwlock
   ```

   出力は以下のようになります。

   ```
   rwl     flag_value    owner status
   $rw1        1          $t1    write
           rsleeps[     0]:
           wsleeps[     0]:
   ```

2. 書き込みモードにあるすべての rwlock に関する情報を表示するには、次のように入力します。

   ```
   rwlock write
   ```

   出力は以下のようになります。

   ```
   rwl     flag_value    owner status
   $rw1        1          $t1    write
           rsleeps[     0]:
           wsleeps[     0]:
   ```

attribute サブコマンド、 **condition** サブコマンド、 **mutex** サブコマンド、 **print** サブコマンド、**thread** サブコマンドを参照してください。

run サブコマンド

run [*Arguments*] [**<***File*] [**>***File*] [**> >***File*] [**2>***File*] [**2> >***File*] [**>&***File*] [**> >&***File*]

run サブコマンドは、オブジェクト・ファイルを開始します。*Arguments* は、コマンド・ラインの引数として渡されます。

フラグ

項目	説明
<*File*	入力が *File* から受け取られるように、入力をリダイレクトします。
>*File*	出力を *File* にリダイレクトします。
2>*File*	標準エラーを *File* にリダイレクトします。
> >*File*	リダイレクトされた出力を *File* に追加します。
2> >*File*	リダイレクトされた標準エラーを *File* に追加します。
>&*File*	出力および標準エラーを *File* にリダイレクトします。
> >&*File*	出力および標準エラーを *File* に追加します。

例

引数 `blue` と `12` を指定してアプリケーションを実行するには、次のように入力します。

```
run blue 12
```

rerun サブコマンドを参照してください。

screen サブコマンド

screen

screen サブコマンドは、 **dbx** コマンドとの対話用に X Window をオープンします。プロセスを開始したウィンドウでの操作を引き続き行うことができます。

screen サブコマンドは、 **dbx** デバッグ・プログラムが X Window System 環境で動作している間に実行する必要があります。 **screen** サブコマンドを X Window 対応でない環境で実行すると、**dbx** プログラムにより警告メッセージが表示され、**screen** サブコマンドを実行しなかった場合と同様にデバッグ処理が再開されます。**screen** サブコマンドは、以下の状況でも成功しないことがあります。

- **dbx** プログラムが、 X Window System 環境で動作していない場合。
- X Window System は動作しているが、 **dbx** のグローバル **$xdisplay** 変数が、有効な表示名に設定されていない場合。**$xdisplay** 変数は、**DISPLAY** 環境変数に初期化されます。 **dbx** サブコマンドの **set** *Name=Expression* は、ディスプレイ名の値を変更します。
- X Window System は動作していても、**TERM** 環境変数が、新しいウィンドウを開始するのに有効なコマンド名に設定されていない場合。
- **/tmp** ディレクトリーが、プログラムに対して読み取りまたは書き込みアクセスを許可しない場合。**dbx** プログラムは、screen コマンドの実行時にこのディレクトリー内に小さなスペースを必要とします。
- システムのリソース不足のため、新しい X Window に対応できない場合。

dbx プログラムは、障害のタイプを識別しませんが、次のメッセージを送信します。

```
Warning: dbx subcommand screen fails. dbx
continues.
```

$xdisplay がリモート・ディスプレイに設定されていると、新しく作成した X Window が表示されないことがあります。 **$xdisplay** が正しく設定されていないと、 X Window System または他のシステム・リソースにより問題点が報告されます。

新しく作成したウィンドウのユーザー定義設定は、 **.Xdefaults** ファイル内でアプリケーション名 **dbx_term** を付けて定義することができます。

例

dbx コマンドによる対話のために X Window をオープンするには、次のように入力します。

```
screen
```

プログラミングの一般概念: プログラムの作成およびデバッグ の『プログラム出力からの dbx 出力の分離』および*AIX Version 6.1 AIXwindows Programming Guide*『AIXwindows Overview』を参照してください。

set サブコマンド

set [*Variable*=*Expression*]

set サブコマンドは、**dbx** デバッグ・プログラムの変数値を定義します。この値は *Expression* パラメーターにより指定され、プログラム変数は *Variable* パラメーターにより指定されます。変数の名前は、デバッグ中のプログラムに含まれる名前と競合してはなりません。変数は、他のコマンドに含まれる対応する式に展開されます。引数を指定せずに **set** サブコマンドを使用すると、現在設定されている変数が表示されます。

変数	説明
$catchbp	次のコマンドの実行中にブレークポイントをキャッチします。
$codepage	プログラム内での文字の解釈に使用するコード・セットを指定します。有効なコード・ページを使用して指定すると、すべての文字が指定のコード・セットから読み取られ、現在の環境で使用されているコード・セットに変換されます。
$compact_bt_ident	スタック・トレースに出力できる ID 名の文字数の限度を指定します。指定される限度は、4 から 128 までの範囲の正整数であることが必要です。この変数が、限度となる値を指定せずに設定された場合、出力できるデフォルトの文字数は 8 文字です。 この変数が設定され、ID 名が指定された限度より 4 文字以上長かった場合、**dbx** コマンドはスタック・トレースに元の ID 名のうちの指定された文字数を出力し、その直後に 3 つのピリオド (...) を続けます。 例えば、ID 名が *variable_example* (長さが 16 文字) で、指定された限度が 7 の場合、ID 名は `variabl...` として出力されます。ただし、ID 名が *variable_1* (長さが 10 文字) で、指定された限度が 7 の場合、**dbx** コマンドは、ID 名を 7 文字とそれに続く 3 つのピリオドに短縮しません。この ID 名は、`variable_1` として出力されます。
$compact_bt_string	スタック・トレースに出力できる関数引数ストリングの文字数の限度を指定します。指定される限度は、4 から 128 までの範囲の正整数であることが必要です。この変数が、限度となる値を指定せずに設定された場合、出力できるデフォルトの文字数は 8 文字です。 この変数が設定され、ストリングが指定された限度より 4 文字以上長かった場合、**dbx** コマンドはスタック・トレースに元のストリングのうちの指定された文字数を出力し、その直後に 3 つのピリオド (...) を続けます。 例えば、ストリングが *string_example* (長さが 14 文字) で、指定された限度が 5 の場合、ストリングは `strin...` として出力されます。ただし、ストリングが 8 文字の長さの *string_1* で、指定された限度が 5 の場合、**dbx** コマンドは、このストリングを 5 文字とそれに続く 3 つのピリオドに短縮しません。このストリングは、`string_1` として出力されます。
$deferevents	据え置きイベント機能をオンにします。
$display_address_name	メンバーの変数 ID、および **dbx** コマンドを使用する一連のメモリー・アドレスを検査する際にその ID が占有するメモリー・アドレスを表示します。
$expandunions	可変レコードまたは共用体の各部分の値を表示します。
$frame	スタック・トレースを実行し、ローカル変数にアクセスするために、 **$frame** の値により指定されたアドレスが指し示すスタック・フレームを使用します。

変数	説明
$hexchars	文字を 16 進数値で表示します。
$hexin	アドレスを 16 進数で解釈します。
$hexints	整数を 16 進数で表示します。
$hexstrings	文字ポインターを 16 進数で表示します。
$hold_next	**cont**、**next**、**nexti**、および **step** の各サブコマンド実行中に、実行中のスレッドを除くすべてのスレッドを保留します。この変数を設定すると、実行中のスレッドがブロック化されているスレッドの 1 つにより保持されているロックを待機する可能性があるため、デッドロックが発生することがあります。
$ignoreifhandler	ユーザーのプログラムが、登録済みハンドラーを持つシグナルを受信したときに、停止しません。
$ignoreload	ユーザーのプログラムが **load**、 **unload**、または **loadbind** サブルーチンを実行したときに、停止しません。
$ignorenonbptrap	ユーザーのプログラムが、非ブレークポイント・トラップ命令を検出し、登録済み **SIGTRAP** ハンドラーを持っているときに、停止しません。
$instructionset	デフォルトの逆アセンブリー・モードを指定変更します。次のリストに、*Expression* パラメーターに使用できる値を示します。

"default"
dbx プログラムが実行されているアーキテクチャーを指定します。

"com" PowerPC と POWER family の各アーキテクチャーの共通論理積モードの命令セットを指定します。**dbx** プログラムのデフォルトは、POWER プロセッサー・ベース・ニーモニックです。

"pwr" POWER family アーキテクチャーの命令セットおよびニーモニックを指定します。

"pwrx" AIX 5.1 およびそれ以前のバージョンについて、POWER family アーキテクチャーの POWER2 インプリメンテーション用の命令セットおよびニーモニックを指定します。

"pwr6" PowerPC アーキテクチャーの POWER6® インプリメンテーション用の命令セットおよびニーモニックを指定します。

"pwr7" PowerPC アーキテクチャーの POWER7® インプリメンテーション用の命令セットおよびニーモニックを指定します。

"pwr8" PowerPC アーキテクチャーの POWER8® インプリメンテーション用の命令セットおよびニーモニックを指定します。

"601" AIX 5.1 およびそれ以前のバージョンについて、PowerPC 601 RISC マイクロプロセッサー用の命令セットおよびニーモニックを指定します。

"603" AIX 5.1 およびそれ以前のバージョンについて、PowerPC 603 RISC マイクロプロセッサー用の命令セットおよびニーモニックを指定します。

"604" PowerPC 604™ RISC マイクロプロセッサーの命令セットおよびニーモニックを指定します。

"970" PowerPC 970 マイクロプロセッサーの命令セットおよびニーモニックを指定します。

"ppc" オプションの命令を除く、POWER プロセッサー・ベース・アーキテクチャーに定義された命令セットおよびニーモニックを指定します。これらの命令は、AIX 5.1 およびそれ以前のバージョンで、PowerPC 601 RISC マイクロプロセッサー以外のすべての POWER プロセッサー・ベースのインプリメンテーションで使用できます。

"any" 有効な POWER プロセッサー・ベース命令または POWER family命令を指定します。オーバーラップする命令セットの場合、デフォルトは POWER プロセッサー・ベース・ニーモニックとなります。

Expression パラメーターに値を指定しないと、 **dbx** プログラムはデフォルトの逆アセンブリー・モードを使用します。

変数 説明

$java 設定すると、以下の変数も設定されて、**dbx** コマンドが Java アプリケーションをデバッグするモードになります。設定解除すると、以下の変数も設定解除されます。

$ignorenonbptrap
Java Just-In-Time (JIT) コンパイラーで生成されたトラップ命令の通知を抑止します。

$listwindow パラメーターを指定しないで **list** サブコマンドを使用したときに、関数の前後をリストする際の行数およびリストの行数を指定します。デフォルトは 10 行です。

$mapaddrs マッピング・アドレスを開始します。**$mapaddrs** を設定解除すると、アドレスのマッピングが停止します。

$mapformat **map** サブコマンドのためのデフォルトの出力モードを指定します。

"abbr" 省略出力モードを指定します。この場合、ロードされた各モジュールは 1 行で構成され、そのモジュールのエントリー番号、モジュール名、オプションとしてメンバー名を含みます。

"normal"
通常出力モードを指定します。この情報には、ロードされた各モジュールについて、エントリー番号、モジュール名、メンバー名、テキストの始まり、テキスト長、データの始まり、データ長、およびファイル・ディスクリプターが含まれます。ロードされたモジュールに TLS データがある場合、TLS データの始まり、および TLS データ長も表示されます。

"raw" ロウ出力モードを指定します。この場合、出力は、ロードされた各モジュールについて不定形式の 1 行で構成され、その中には、エントリー番号、モジュール名とオプションのメンバー名、テキストの始まり、テキストの終わり、テキスト長、データの始まり、データの終わり、データ長、およびファイル・ディスクリプターが、スペースで分離されたフィールドとして含まれます。ロードされたモジュールに TLS データがある場合、TLS データの始まり、TLS データの終わり、および TLS データ長も表示されます。

"verbose"
詳細出力を指定します。この情報には、ロードされた各モジュールについて、エントリー番号、モジュール名、メンバー名、テキストの始まり、テキストの終わり、テキスト長、データの始まり、データの終わり、データ長、およびファイル・ディスクリプターが含まれます。ロードされたモジュールに TLS データがある場合、TLS データの始まり、TLS データの終わり、および TLS データ長も表示されます。

Expression パラメーターに値を指定しないと、 **dbx** プログラムは通常出力モードを使用します。

$mnemonics 逆アセンブル時に **dbx** プログラムにより使用されるニーモニックのセットを変更します。

"default"
指定された命令セットに最も近いニーモニックを指定します。

"pwr" POWER family アーキテクチャーのニーモニックを指定します。

"ppc" オプションの命令を除く、POWER プロセッサー・ベース・アーキテクチャー・ブックに定義されたニーモニックを指定します。

Expression パラメーターに値を指定しないと、 **dbx** プログラムは指定した命令セットに最も近いニーモニックを指定します。

$noargs *where*、*up*、*down*、および *dump* などのサブコマンドから引数を省略します。

$noflregs **registers** サブコマンドから浮動小数点レジスターの表示を省略します。

$novregs **registers** サブコマンドからベクトル・レジスターの表示を省略します。

$novsregs **registers** サブコマンドからベクトル・スカラー・レジスターの表示を省略します。

$octint アドレスを 8 進数で解釈します。

$octints 整数を 8 進数で表示します。

変数	説明
$pretty	C および C++ の複合データ構造 (strut、union、array) の値を、**print** サブコマンドを使用して、*pretty* プリント 形式で表示します。 **"on"** pretty プリントを、1 行ごとに値を持ち、それぞれの値の静的有効範囲を字下げで表すように指定します。 **"verbose"** pretty プリントを、1 行ごとに値を持ち、それぞれの値の静的有効範囲を修飾名で表すように指定します。修飾名は、その値が関連付けられている外部ブロックのリストをドットで区切ったものです。 **"off"** pretty プリントをオフに指定します。この値はデフォルトです。
$print_dynamic	print/dump コマンドを使用して動的タイプの C++ オブジェクトを表示します。デフォルトでは、この変数は設定されていません。
$repeat	コマンドが入力されなかった場合に、前に入力されたコマンドを反復します。
$sigblock	ユーザーのプログラムへのシグナルをブロック化します。
$show_vft	仮想関数テーブルを表示し、同時に print/dump コマンドを使用して C++ オブジェクトを印刷します。デフォルトでは、これは設定されていません。
$stack_details	**where** サブコマンドによって表示される各アクティブ関数またはプロシージャーのフレーム番号およびレジスター・セットを表示します。
$stepignore	デバッグ情報が使用できない別のルーチンを呼び出すソース行で **step/tstep** サブコマンドを実行するときの **dbx** コマンドの動作方法を制御します。この変数を使用すると、**step /tstep** サブコマンドは、デバッグ情報が使用できない大規模なルーチンをステップオーバーすることができます。次のリストに、*Expression* パラメーターに使用できる値を示します。 **"function"** **dbx** コマンドの **next /tnext** サブコマンドの関数を実行します。この値はデフォルト値です。 **"module"** デバッグ情報が使用できないロード・モジュール (システム・ライブラリーなど) に **next/tnext** サブコマンドの関数がある場合は、その関数を実行します。 **"none"** ソース情報が使用できる命令に達するまで、**dbx** コマンドの **stepi/tstepi** サブコマンドの関数をバックグラウンドで実行します。ソース情報が使用できる命令に達すると、**dbx** は実行を停止した位置を表示します。
$trace_good_transaction	**dbx** コマンドに、トランザクション・メモリー (TM) のトランザクションが正常に完了するたびに次のメッセージを表示するよう指示します。 `Process {PID} may have performed a transaction - $texasr, $tfiar, $tfhar are valid and may be inspected` この変数はデフォルトでは使用不可になっているので、トランザクションが成功しても報告されません。
$thcomp	**$thcomp** が設定されているとき、 thread サブコマンド **th-** が表示する情報は圧縮フォーマットで表示されます。
$unsafeassign	**assign** ステートメントの指定範囲内の精密タイプ検査をオフにします。 **$unsafeassign** 変数が設定されている場合でも、 **assign** ステートメントの指定範囲内に異なるサイズのストレージ・タイプは設定されません。
$unsafebounds	配列の添え字検査をオフにします。
$unsafecall	サブルーチンまたは関数呼び出しに渡される引数に対する精密な型チェックをオフにします。
$unsafegoto	**goto** サブコマンドの宛先検査をオフにします。
$vardim	バウンダリーが不明である配列を表示するときに使用するサイズを指定します。デフォルト値は 10 です。
$xdisplay	**multproc** サブコマンドまたは **screen** サブコマンドで使用する X Window System のディスプレイ名を指定します。デフォルト値は、シェルの **DISPLAY** 変数の値です。

$unsafe 変数を指定すると、 **dbx** デバッグ・プログラムのエラー検出の有用性が制限されます。

例

1. 表示する行数のデフォルト値を 20 に変更するには、次のように入力します。

   ```
   set $listwindow=20
   ```

2. assign サブコマンドの型チェックを使用不可にするには、次のように入力します。

   ```
   set $unsafeassign
   ```

3. POWER7 プロセッサーのマシン・インストラクションを逆アセンブルするには、次のように入力します。

   ```
   set $instructionset="pwr7"
   ```

4. IBM-eucCN コード・セットでエンコードされた文字列を表示するには、次のように入力します。

   ```
   set $codepage="IBM-eucCN"
   ```

5. スタック・トレースに表示される ID に 4 文字の限度、ストリングに 12 文字の限度を指定するには、次のコマンドを入力します。

   ```
   set $compact_bt_ident=6
   set $compact_bt_string=12
   ```

   long_identifier、long_variable_name_str、recursive_fun などの ID、および this_is_a_really_long_string などのストリングを使用するスタック・トレースは、次の出力のようになります。

   ```
   long_i...(a = 11, long_v... = "this_is_a_re..."), line 3 in "example.c"
   recurs...(), line 13 in "example.c"
   ```

unset サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『特殊なデバッグ・プログラム変数による印刷出力の変更』も参照してください。

set edit [vi, emacs] または **set -o [vi, emacs]** サブコマンド

set サブコマンドと **-o** または **edit** オプションを一緒に使用して、行編集モードの 1 つをオンにすることができます。 **set-o vi** または **set edit vi** コマンドを指定すると、 *vi* 行エディターの入力モードに入ります。 **set -o emacs** または **set edit emacs** コマンドを指定すると、 *emacs* 行エディターの入力モードに入ります。

例

1. vi 行エディターをオンにするには、次のように入力します。

   ```
   set-o vi
   ```

   または

   ```
   set edit vi
   ```

sh サブコマンド

sh [*Command*]

sh サブコマンドは、 *Command* パラメーターにより指定されたコマンドをシェルに渡して実行します。 **SHELL** 環境変数は、使用するシェルを決定します。デフォルトは **sh** シェルです。引数を指定しないと、シェルに制御が渡されます。

例

1. ls コマンドを実行するには、次のように入力します。

   ```
   sh ls
   ```

2. シェルにエスケープするには、次のように入力します。

```
sh
```

3. SHELL 環境変数を使用するには、次のように入力します。

```
sh echo $SHELL
```

プログラミングの一般概念: プログラムの作成およびデバッグ の『dbx からのシェル・コマンドの実行』を参照してください。

skip サブコマンド

skip [*Number*]

skip サブコマンドは、現在の停止位置からアプリケーション・プログラムの実行を継続します。*Number* パラメーターの値に等しいブレークポイントの数がスキップされ、次のブレークポイントに達したとき、またはプログラムが終了したときに実行が停止します。*Number* パラメーターを指定しないと、デフォルト値の 1 が採用されます。

例

2 番目のブレークポイントを検出するまで実行を継続するには、次のように入力します。

```
skip 1
```

cont サブコマンドのセクションを参照してください。

source サブコマンド

source *File*

source サブコマンドは、*File* パラメーターにより指定されたファイルから **dbx** サブコマンドを読み取ります。

例

cmdfile ファイルの **dbx** サブコマンドを読み取るには、次のように入力します。

```
source cmdfile
```

プログラミングの一般概念: プログラムの作成およびデバッグ の『ファイルからの dbx サブコマンドの読み取り』を参照してください。

status サブコマンド

status [**more**] [>*File*]

status サブコマンドは、残ったスレッド **tskip** カウント (**tskip** サブコマンドを使用して設定されたもの) のほかに、すべてのユーザー定義のブレークポイント、トレース・ポイント、および監視ポイントを表示します。さらにパラメーターが指定されている場合は、**status** サブコマンドは、ブレークポイント、トレース・ポイント、および監視ポイントに関連した **dbx** サブコマンドも表示します。 **status** サブコマンドでは、使用可能イベントはイベント番号を大括弧 ([]) で囲んで、使用不可イベントはイベント番号をピリオド (..) で囲んで、また据え置きイベントはイベント番号を不等号括弧 (<>) で囲んでリストされます。

> フラグは、**status** サブコマンドの出力を *File* パラメーターで指定されたファイルに送ります。

フラグ

項目	説明
>*File*	出力を *File* にリダイレクトします。

例

1. すべてのユーザー定義のブレークポイント、トレース・ポイント、監視ポイント、ならびに残りのスレッド **tskip** カウントを表示するには、次のように入力します。

   ```
   status
   ```

   出力は以下のようになります。

   ```
   [1] stop at 13
   [2] stop at 14
   .3. stop at 15
   .4. stop at 16
   [5] stop at 17 ( count = 0, limit = 3 )
   <6> stop at 18 if g > 10
   <7> stop in func

   Remaining tskip counts:
   tskip 2 for $t1
   tskip 1 for $t5
   ```

   上記の出力例では、イベント 3 と 4 が使用不可であり、イベント 6 と 7 が据え置きです。

2. すべてのユーザー定義のブレークポイント、トレース・ポイント、および監視ポイントを、関連した **dbx** サブコマンドと共に表示するには、次のように入力します。

   ```
   status more
   ```

   出力は以下のようになります。

   ```
   [1] stop at 13
       [1] where
   .2. stop at 14
       [1] where
       [2] registers
   <3> stop at 15 if g > 10
       [1] where; registers
   ```

dbx コマンドの **addcmd** サブコマンド、 **clear** サブコマンド、 **delete** サブコマンド、 **delcmd** サブコマンド、 **tskip** サブコマンド、 **stop** サブコマンド、および **trace** サブコマンドを参照してください。

また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』も参照してください。

step サブコマンド

step [*Number*]

step サブコマンドは、アプリケーション・プログラムのソース行を実行します。実行する行数は、*Number* パラメーターで指定します。*Number* パラメーターを指定しないと、デフォルト値の 1 が採用されます。

マルチスレッド・アプリケーション・プログラムで **step** サブコマンドを使用すると、操作中にすべてのユーザー・スレッドが実行されますが、プログラムは実行中のスレッドが指定のソース行に達するまで実行を継続します。実行中のスレッドのみを進行させる場合は、 **set** サブコマンドを使用して変数 **$hold_next** を設定します。この変数を設定すると、実行中のスレッドがブロック化されているスレッドの 1 つにより保持されているロックを待機する可能性があるため、デッドロックが発生することがあります。

注**: step** サブコマンドの動作を制御するには、**set** サブコマンドの **$stepignore** 変数を使用します。**$stepignore** 変数を使用すると、 **step** サブコマンドはデバッグ情報が使用できない大規模なルーチンへステップオーバーすることができます。

例

1. ソース行の実行を 1 行継続するには、次のように入力します。

   ```
   step
   ```

2. ソース行の実行を 5 行継続するには、次のように入力します。

   ```
   step 5
   ```

3. **dbx** プログラムが次のサンプル・コードに示すような **printf** 関数に単一ステップで進まないようにするには、

   ```
   60 printf ("hello world ¥n");
   ```

   次のように入力します。

   ```
   set $stepignore="function"; step
   ```

cont サブコマンド、 **goto** サブコマンド、 **next** サブコマンド、 **set** サブコマンド、 **stepi** サブコマンドを参照してください。

stepi サブコマンド

stepi [*Number*]

stepi サブコマンドは、アプリケーション・プログラムの命令を実行します。実行する命令の数は、*Number* パラメーターで指定します。*Number* パラメーターを省略すると、デフォルト値の 1 が採用されます。

stepi サブコマンドをマルチスレッド・アプリケーション・プログラムで使用すると、実行中のスレッドのみを進行させます。他のすべてのユーザー・スレッドは停止されたままです。

例

1. マシン・インストラクションの実行を 1 つ継続するには、次のように入力します。

   ```
   stepi
   ```

2. マシン・インストラクションの実行を 5 つ継続するには、次のように入力します。

   ```
   stepi 5
   ```

gotoi サブコマンド、 **nexti** サブコマンド、 **step** サブコマンドを参照してください。

stop サブコマンド

stop { [*Variable*] [**at** *SourceLine* | **in** *Procedure* | **on load** ["*ModuleName*"]] [**if** *Condition*]} ["{ "*Limit*" }"]

stop サブコマンドは、一定の条件が満たされると、アプリケーション・プログラムの実行を停止します。アプリケーション・プログラムが停止するのは、以下のような場合です。

- **if** *Condition* フラグを使用しているときに、*Condition* が真である場合。
- **in** *Procedure* フラグを使用しているときに、*Procedure* が呼び出された場合。
- *Variable* パラメーターを指定しているときに、*Variable* が変更された場合。
- **at** *SourceLine* フラグを使用しているときに、*SourceLine* 行番号に達した場合。

SourceLine 変数は、整数として、または : (コロン) および整数が後ろに続くファイル名文字列として指定できます。

- **on load** フラグが使用され、*ModuleName* パラメーターが指定されているときに、その *ModuleName* ロード・モジュールがロードまたはアンロードされる場合。

  オプションの *ModuleName* 変数を使用すると、単一のモジュール名を、または、以下のフォーマットでモジュール名とメンバー名のペアを以下のように指定することができます。

  ModuleName(*MemberName*)
- **on load** フラグが使用され、*ModuleName* パラメーターが指定されていないときに、いずれかのロード・モジュールがロードまたはアンロードされる場合。

指定された回数だけ条件を無視するように **dbx** コマンドに指示する場合は、*Limit* パラメーターを設定できます。すなわち、*Limit* パラメーターは、デバッグ・プログラムの実行を停止する前に、指定された条件を満たす必要がある回数を指定します。

これらのコマンドの実行後、**dbx** デバッグ・プログラムは、コマンドの結果として作成したイベントを報告するメッセージを表示します。このメッセージには、コマンドの解釈と共にブレークポイントに関連付けられているイベント ID が含まれます。解釈の構文は、コマンドと一致していない場合があります。次に例を示します。

```
stop in main
[1] stop in main
stop at 19 if x == 3
[2] stop at "hello.c":19 if x = 3
stop in func
<3> stop in func
stop g
<4> stop g
stop in getdata {3}
[5] stop in getdata ( count = 0, limit = 3 )
```

大括弧 ([]) 内の番号は、ブレークポイントに関連付けられているイベント ID です。**dbx** デバッグ・プログラムは、イベント番号を各 **stop** サブコマンドと関連付けます。プログラムが 1 つのイベントの結果として停止させられると、どのイベントがプログラムを停止させたかを示すために、イベント ID が現在行と一緒に表示されます。不等号括弧 (<>) 内の番号は、据え置きイベントのイベント ID です。据え置きイベントとは、ブレークポイント、トレース・ポイント、または監視ポイントが関連付けられておらず、現在メモリーにロードされていないシンボルが入力コマンドに含まれていると必ず生成されるイベントです。大括弧 ([]) 内に表示される通常のイベントも、対応するモジュールがアンロードされると必ず据え置きイベントに変換されます。据え置きイベントに対応するモジュールがメモリーにロードされると、据え置きイベントは通常のイベントに変換され、対応するブレークポイント、トレース・ポイント、または監視ポイントが作成されます。ユーザーが作成するイベントは、**dbx** コマンドにより作成された内部イベントと共存するため、イベント番号が常に連番になっているとは限りません。

イベントの作成後に制限をイベントと関連付けるには、**limitbp** サブコマンドを使用します。イベントと関連付けられた制限を表示するには、**printbp** サブコマンドを使用できます。

これらの番号を表示する場合は、**status** サブコマンドを使用します。出力を **status** から任意のファイルにリダイレクトすることができます。 **stop** サブコマンドをオフにするには、**delete** または **clear** サブコマンドを使用します。あるいは、**enable** または **disable** サブコマンドを使用します。指定されたイベント番号に **dbx** サブコマンドを追加するには **addcmd** サブコマンドを使用し、指定されたイベント番号から関連した **dbx** サブコマンドを削除するには **delcmd** サブコマンドを使用します。

マルチスレッド・アプリケーション・プログラムでは、ユーザー・スレッドが 1 つでもブレークポイントに達すると、すべてのユーザー・スレッドが一時停止されます。 例 9 に示すような条件を指定した場合を除き、ソース行または関数を実行するユーザー・スレッドは、そのソース行または関数に設定されたブレークポイントにヒットします。以下の別名は、自動的に条件を指定します。

- **bfth** (*Function, ThreadNumber*)
- **blth** (*SourceLine, ThreadNumber*)

ThreadNumber は、**thread** サブコマンドで表示される、シンボル・スレッド名の番号部分です (例えば、5 はスレッド名 $t5 の *ThreadNumber* です)。これらの別名は、実際には以下の例のように展開されるサブコマンドを生成するマクロです。

```
stopi at &Function if ($running_thread == ThreadNumber)
stop at SourceLine if ($running_thread == ThreadNumber)
```

フラグ

項目	説明
at *SourceLine*	行番号を指定します。
if *Condition*	条件 (例えば、真) を指定します。
in *Procedure*	呼び出されるプロシージャーを指定します。
on load *ModuleName*	ロード・モジュールをモニターすることを指定します。

例

1. `main` プロシージャーの、最初のステートメントで実行を停止するには、次のように入力します。

   ```
   stop in main
   ```

2. 変数 `x` の値が、実行を開始してから `12` 行目で変更された場合に実行を停止するには、次のように入力します。

   ```
   stop x at 12
   ```

3. ファイル `sample.c` の、 `5` 行目で実行を停止するには、次のように入力します。

   ```
   stop at "sample.c":5
   ```

4. **dbx** コマンドが、 `func1` 内のサブルーチンを実行するたびに、 `x` の値を調べるには、次のように入力します。

   ```
   stop in func1 if x = 22
   ```

5. **dbx** コマンドが、 `func1` の実行を開始するたびに、 `x` の値を調べるには、次のように入力します。

   ```
   stopi at &func1 if x = 22
   ```

6. *Variable* の値が変わったときに、プログラムを停止させるには、次のように入力します。

   ```
   stop Variable
   ```

7. *Condition* が真であると評価されたときに、常にプログラムを停止させるには、次のように入力します。

   ```
   stop if (x > y) and (x < 2000)
   ```

8. 次の例では、アクティブ・イベントの表示方法および除去方法を示しています。

   ```
   status
   [1] stop in main
   [2] stop at "hello.c":19 if x = 3
   delete 1
   status
   [2] stop at "hello.c":19 if x = 3
   clear 19
   status
   (dbx)
   ```

delete コマンドは、イベント ID によりイベントを削除します。 **clear** コマンドは、行番号に基づいてブレークポイントを削除します。

9. スレッド `$t5` で実行されるときにだけ、 `func1` の先頭にブレークポイントを配置するには、以下の等価のコマンドのどちらかを入力します。

   ```
   stopi at &func1 if ($running_thread == 5)
   ```

   または

   ```
   bfth(func1, 5)
   ```

10. いずれかのモジュールがロードまたはアンロードされたときにプログラムを停止するには、次のように入力します。

    ```
    stop on load
    ```

11. モジュール `Module` がロードまたはアンロードされるときにプログラムを停止するには、次のように入力します。

    ```
    stop on load "Module"
    ```

12. モジュール `Module` のメンバー `Member` がロードまたはアンロードされるときにプログラムを停止するには、次のように入力します。

    ```
    stop on load "Module(Member)"
    ```

13. 関数 `getdata` 内のプログラムが 3 回目に呼び出されたときにこれを停止するには、次のように入力します。

    ```
    stop in getdata {3}
    ```

addcmd サブコマンド、**clear** サブコマンド、**delete** サブコマンド、**delcmd** サブコマンド、**disable** サブコマンド、**enable** サブコマンド、**limitbp** サブコマンド、**printbp** サブコマンド、**status** サブコマンド、**stopi** サブコマンド、および **trace** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『ブレークポイントの設定および削除』も参照してください。

stophwp サブコマンド

stophwp *Address Size*

stophwp サブコマンドは、ハードウェアの監視ポイントの停止を指定されたメモリー領域に設定します。プログラムは、その領域の内容が変化したときに停止します。

注**:**

1. **stophwp** サブコマンドが成功するかどうかは、ハードウェアに依存します。この機能は、POWER630 および POWER4 以降でのみ使用可能です。
2. 単一の監視ポイントのみ設定できるというハードウェアの制限により、 **stophwp** および **tracehwp** で別のハードウェア監視ポイント・イベントを作成しようとすると、アクティブな監視ポイント・イベントは矛盾するものとして動作することになります。そのため、新しいイベントを作成する前に、前のイベントは削除する必要があります。また、アクティブなソフトウェアの監視ポイントの存在 (**stop** および **trace** サブコマンドの呼び出しにより作成された) はハードウェアの監視ポイントのパフォーマンス向上を打ち消すため、これらのタイプのイベントも矛盾するものとして動作するので、ハードウェアの監視ポイントを作成する前に削除する必要があります。

例

1. アドレス 0x200004e8 で始まる 4 バイト・メモリー領域の内容が変化するときにプログラムを停止するには、次のように入力します。

```
stophwp 0x200004e8 4
```

tracehwp サブコマンドを参照してください。

stopi サブコマンド

stopi { [*Address*] [**at** *Address* | **in** *Procedure*] [**if** *Condition*]}

stopi サブコマンドは、指定した位置に停止を設定します。

- **if** *Condition* フラグを指定している場合、条件に真を指定するとプログラムは停止します。
- *Address* パラメーターを指定している場合、 *Address* の内容が変更されるとプログラムは停止します。
- **at** *Address* フラグを指定している場合、指定したアドレスに停止が設定されます。
- **in** *Procedure* フラグを指定している場合、 *Procedure* が呼び出されるとプログラムは停止します。

フラグ

項目	説明
if *Condition*	条件 (例えば、真) を指定します。
in *Procedure*	呼び出されるプロシージャーを指定します。
at *Address*	マシン・インストラクションのアドレスを指定します。

例

1. アドレス 0x100020f0 で実行を停止するには、次のように入力します。
   ```
   stopi at 0x100020f0
   ```
2. アドレス 0x100020f0 の内容が変更されたときに、実行を停止するには、次のように入力します。
   ```
   stopi 0x100020f0
   ```
3. アドレス 0x100020f0 の内容が、スレッド $t1 により変更されたときに実行を停止するには、次のように入力します。
   ```
   stopi 0x200020f0 if ($running_thread == 1)
   ```

stop サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『dbx によるマシン・レベルのデバッグ』も参照してください。

thdata サブコマンド

thdata [*$tthreadnumber* [all | *key1* ...] ...] | [all]

thdata サブコマンドは、**pthread_key_create()** 関数を使用して作成された各種キーに関連するスレッド固有データを印刷します。 **thdata** サブコマンドは、次のように使用できます。

コマンド	アクション
thdata [all]	すべての使用可能なスレッドのキーすべてに関連するスレッド固有データを印刷します。
thdata $t1 [all]	*$t1* スレッドのすべてのキーに関連付けられたスレッド固有データを印刷します。
thdata $t1 key1 key2	*$t1* スレッドのキー *key1* および *key2* に関連付けられたスレッド固有データを印刷します。
thdata $t1 key1 key2 $t2 key1	*$t1* スレッドのキー *key1* および *key2* に関連付けられたスレッド固有データ、および *$t2* スレッドのキー *key1* に関連付けられたスレッド固有データを印刷します。

例

1. 現行スレッドに対してすべての使用可能なキーに関連付けられたデータを印刷するには、次のように入力します。

```
(dbx) thdata $t1
Thread  :  1
 Key : 1 Data pointer : 0x200f7a28
 Key : 2  Data pointer : 0x200f7aa8
 Key : 3 Data pointer : 0x200f7ac4
(dbx)
```

2. 複数のスレッドと複数のキーに対して、関連付けられたデータを印刷するには、次のように入力します。

```
(dbx) thdata $t1 2 3 $t2
Thread  :  1
 Key : 2  Data pointer : 0x200f7aa8
 Key : 3 Data pointer : 0x200f7ac4
Thread : 2
 Key : 2  Data pointer : 0x200f7b24
 Key : 3 Data pointer : 0x200f7ba4
(dbx)
```

「プログラミングの一般概念: プログラムの作成およびデバッグ」の『スレッド固有データ』を参照してください。

thread サブコマンド

選択したスレッドを表示する

thread { [**info**] [**-**] [*ThreadNumber* ...] } | **current** | **run** | **susp** | **term** | **wait**

個別のスレッドを選択する

thread current [-] *ThreadNumber*

スレッドを保留または解放する

thread { **hold** | **unhold** } [-] [*ThreadNumber* ...]

表示されたオプションのヘルプを表示する

thread { **help**}

thread サブコマンドは、ユーザー・スレッドを表示および制御します。

thread サブコマンドの最初のフォーマットは、2 つのフォーマットで情報を表示することができます。 **thread** サブコマンドが **th** ならば、表示される情報は最初のフォーマットになります。 **thread** サブコマンドが **th -** ならば、表示される情報は 2 番目のフォーマットになります。 パラメーターを指定しないと、すべてのユーザー・スレッドに関する情報が表示されます。 1 つ以上の *ThreadNumber* を指定すると、それに対応するユーザー・スレッドの情報が表示されます。 **thread** サブコマンドがスレッドを表示するとき、現行のスレッド行の前に **>** が置かれます。実行中のスレッドが現行スレッドと同じでない場合は、そのスレッド行の前には ***** が表示されます。 **thread** サブコマンドによって両方のフォーマットで表示される情報を、以下に説明します。

最初のフォーマットで **thread** サブコマンドによって表示される情報を、次に示します。

項目	説明
thread	ユーザー・スレッドのシンボル名を $t*ThreadNumber* フォーマットで示します。
state-k	カーネル・スレッドの状態を示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。この値は、それぞれ実行中、待機中、延期、終了を示す run、wait、 susp、term のいずれかです。
wchan	カーネル・スレッドが待機中またはスリープ中のイベントを示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。
state-u	ユーザー・スレッドの状態を示します。可能な状態は、running、blocked、terminated のいずれかです。
k-tid	カーネル・スレッドの ID を示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。
mode	ユーザー・スレッドが停止されるモード (カーネルまたはユーザー) を示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。
held	ユーザー・スレッドが保留されているかどうかを示します。
scope	ユーザー・スレッドの競合範囲を示します。これは、システムまたはプロセスの競合範囲についてそれぞれ sys または pro のいずれかを指定できます。
function	ユーザー・スレッド関数の名前を示します。

2 番目のフォーマットで **thread** サブコマンドによって表示される情報を、以下に示します。デフォルトでは、**thread** サブコマンド **th -** の場合、情報は長形式で表示されます。

項目	説明
thread	ユーザー・スレッドのシンボル名を $t*ThreadNumber* フォーマットで示します。

カーネル・スレッド関連の情報

項目	説明
tid	ユーザー・スレッド ID を示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。
pri	カーネル・スレッドの優先順位を示します。
sched	カーネル・スレッドのスケジューリング・ポリシーを示します。この値には、fif、oth、rr、for fifo、その他 (other)、またはラウンドロビン (round robin) スケジューリング・ポリシーがあります。
state	カーネル・スレッドの状態を示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。この値には、実行中、待機中、延期中、またはゾンビを示す、run、wait、susp、または zomb があります。

ユーザー・スレッド関連の情報

項目	説明
tid	ユーザー・スレッド ID を示します。
pri	userl スレッドの優先順位を示します。
sched	ユーザー・スレッドのスケジューリング・ポリシーを示します。この値には、fif、oth、rr、for fifo、その他 (other)、またはラウンドロビン (round robin) スケジューリング・ポリシーがあります。
state	ユーザー・スレッドの状態を示します。この値には、実行中、作成中、延期、ブロック済み、実行可能、または終了の状態があります。
state	ユーザーの状態を 16 進数で示します。
フラグ	pthread フラグの値を 16 進数で示します。
wchan	カーネル・スレッドが待機中またはスリープ中のイベントを示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。
mode	ユーザー・スレッドが停止されるモード (カーネルまたはユーザー) を示します (ユーザー・スレッドがカーネル・スレッドに付加されている場合)。
held	ユーザー・スレッドが保留されているかどうかを示します。
scope	ユーザー・スレッドの競合範囲を示します。この値は、システムまたはプロセスの競合範囲を表す sys または pro です。

項目	説明
cancelation	**pending** 取り消しが保留中であるかどうかを示します。 **state** 取り消しのモードおよび状態を示します。 取り消しが保留中でなく、かつ状態とモードがそれぞれ使用可と据え置きになっていると、取り消しは **ed** で表され、取り消しの状態とモードが使用可と非同期になっていると、 **ea** で表され、モードが使用可でないと **d** で表されます。 取り消しが保留中で、かつ取り消しの状態とモードがそれぞれ使用可と据え置きになっていると、取り消しは **ED** で表され、取り消しの状態とモードが使用可と非同期になっていると、**EA** で表され、モードが使用可でない場合は **D** で表されます。

項目	説明
joinable	スレッドが結合可能かどうかを示します。
boosted	スレッドの格上げ値を示します。
function	ユーザー・スレッド関数の名前を示します。
cursig	現行シグナル値を示します。

オプション・セット $thcomp が設定されると、情報は以下の例に示すように圧縮形式で表示されます。

```
m           mode           (k)ernel (u)ser
k           k-state        (r)unning (w)aiting (s)uspended (z)ombie
u           u-state        (r)unning (R)unnable (s)uspended (t)erminated

                           (b)locked (c)reating
h           held           (yes) (n)o
s           scope          (s)ystem (p)rocess
c           cancellation   not pending:  (e)nabled & (d)eferred,
                                         (e)nabled & (a)sync, (d)isabled
                           pending    :  (E)nabled & (D)eferred,
                                         (E)nabled & (A)sync, (D)isabled
j           joinable       (yes) (n)o
b           boosted        value of boosted field in pthread structure
plk         kernel thread  (oth)er  (fif)o  (rr)-> round-robin
            policy
plu         user thread    (oth)er  (fif)o  (rr)-> round-robin
            policy
prk         kernel thread  hex number
            policy
pru         user thread    hex number
            policy
k-tid                      kernel thread id in hex
u-tid                      pthread id in hex
fl                         value of flags field in pthread structure in hex
sta                        value of state field in pthread structure in hex
cs                         value of the current signal
wchan                      event for which thread is waiting
function                   function name
```

現行スレッドを選択する場合は、**thread** サブコマンドの 2 番目のフォーマットが使用されます。 **dbx** デバッグ・プログラムの **print**、**registers**、および **where** の各サブコマンドは、いずれも現行スレッドのコンテキストに沿って機能します。現行スレッドがカーネル・モードになっている場合は、 **registers** サブコマンドはレジスターを表示できません。

thread サブコマンドの 3 番目のフォーマットは、スレッドの実行の制御に使用します。スレッドは、**hold** フラグを使用して保留にするか、 **unhold** フラグを使用して解放することができます。保留にされているスレッドは、解放されるまで再開されません。

注**: dbx** デバッグ・プログラムの **print** サブコマンドは、シンボルによるスレッド名を認識し、対応するオブジェクトの状況を表示するために使用できます。

フラグ

項目	説明
現行	*ThreadNumber* パラメーターを指定しないと、現行スレッドが表示されます。 *ThreadNumber* パラメーターを指定すると、指定したユーザー・スレッドが現行スレッドとして選択されます。
help	**th -** コマンドを使用したとき表示されるスレッド・オプションに関する情報をすべて表示します。
hold	*ThreadNumber* パラメーターを指定しないと、すべてのユーザー・スレッドが保留にされ、表示されます。1 つ以上の *ThreadNumber* パラメーターを指定すると、指定したユーザー・スレッドが保留にされ、表示されます。
unhold	*ThreadNumber* パラメーターを指定しないと、以前に保留にされたすべてのユーザー・スレッドが解放され、表示されます。 1 つ以上の *ThreadNumber* を指定すると、指定したユーザー・スレッドが解放され、表示されます。
info	*ThreadNumber* パラメーターを指定しないと、すべてのユーザー・スレッドの長形式のリストが表示されます。 1 つ以上の *ThreadNumber* パラメーターを指定すると、指定したユーザー・スレッドの長いフォーマットのリストが表示されます。 上記のフラグはすべて [-] オプションを取ります。このオプションを指定すると、 `set $thcomp` オプションが設定されない限り、スレッド情報は 2 番目の形式と長形式で表示されます。
run	`run` 状態のスレッドを表示します。
susp	`susp` 状態のスレッドを表示します。
term	`term` 状態のスレッドを表示します。
wait	`wait` 状態のスレッドを表示します。

例

1. 待機状態のスレッドに関する情報を表示するには、次のように入力します。

   ```
   thread  wait
   ```

   出力は以下のようになります。

   ```
   thread  state-k   wchan state-u   k-tid mode held scope function
    $t1     wait            running   17381   u   no   pro  main
    $t3     wait            running    8169   u   no   pro  iothread
   ```

2. 複数の、任意のスレッドについて情報を表示するには、次のように入力します。

   ```
   thread 1 3 4
   ```

   出力は以下のようになります。

   ```
   thread  state-k   wchan state-u   k-tid mode held scope function
    $t1     wait           running   17381   u   no   pro  main
    $t3     wait           running    8169   u   no   pro  iothread
   >$t4     run            running    9669   u   no   pro  save_thr
   ```

3. スレッド 4 を現行スレッドにするには、次のように入力します。

   ```
   thread  current 4
   ```

4. スレッド番号 2 を保留にするには、次のように入力します。

   ```
   thread  hold 2
   ```

5. 待機状態のスレッドについて情報を 2 番目のフォーマットで表示するには、次のように入力します。

   ```
   thread wait -
   ```

   出力は以下のようになります。

```
thread m k u h s c  j b  kpl  upl  kpr upr k_tid   u_tid   fl  sta wchan  function
*$t1   u r w n p ed y 0  oth  oth  61   1  0043e5  000001  51  004        main
 $t3   u r w n p ed y 0  oth  oth  61   1  001fe9  000102  51  004        iothread
>$t4   u r r n p ed y 0  oth  oth  61   1  0025c5  000203  50  064        save_thr
```

6. 複数の、任意のスレッドについて情報を 2 番目のフォーマットで表示するには、次のように入力します。

```
thread - 1 2 3
```

出力は以下のようになります。

```
thread m k u h s c  j b  kpl  upl  kpr upr k_tid   u_tid   fl  sta wchan function
*$t1   u r w n p ed y 0  oth  oth  61   1  0043e5  000001  51  004       main
 $t3   u r w n p ed y 0  oth  oth  61   1  00fe9   000102  51  004       iothread
>$t4   u r r n p ed y 0  oth  oth  61   1  0025c5  000203  50  064       save_thr
```

attribute サブコマンド、 **condition** サブコマンド、 **mutex** サブコマンド、 **print** サブコマンド、 **registers** サブコマンド、 **where** サブコマンドを参照してください。

また、「プログラミングの一般概念: プログラムの作成およびデバッグ」の『スレッドの作成』も参照してください。

tls サブコマンド

tls map

tls サブコマンドは、ロードされた TLS モジュールごとに TLS 初期化テンプレートの始まりおよび長さを表示するために使用するフラグとして、フラグを 1 つだけ取ります。

tm_status サブコマンド

```
tm_status
```

tm_status サブコマンドは、*$texasr* 変数 (トランザクション例外およびサマリー・レジスター) の内容を表示し、トランザクション失敗の原因および種類を判別するためにその内容を解釈します。

例

$texasr 変数に格納された値を表示および解釈するには、次のコマンドを入力します。

```
(dbx) tm_status
```

以下の例のような出力が表示されます。

```
REGISTER : TEXASR = 0x100000018C000001
Bit(s)   |Field                    |Meaning
_____________________________________________________________________
0-7      Failure Code               TM_SIG_DELIVERED | Failed due to signal delivery
7        Failure Persistent         Failure is transient
31       Abort                      Execution of TM instruction caused Abort
32       Suspended                  Failure while in Suspended State
34-35    Privilege                  During Failure process-thread privilege state was 0
36       Failure                    Summary Failure recording has been performed
37       TFIAR (in)exact            TFIAR is exact
38       Rollback Only Transaction  non-ROT tbegin. initiated
52-63    Transaction Level          1
```

tnext サブコマンド

tnext [*Number*]

tnext サブコマンドは、実行中のスレッドを次のソース行まで実行します。 *Number* パラメーターは、**tnext** サブコマンドの実行回数を指定します。 *Number* パラメーターを指定しない場合は、**tnext** は 1 回だけ実行されます。このサブコマンドは、システム・スコープ・スレッドでのみ開始できます。

この操作時にすべてのスレッドが実行されます。この操作時にブレークポイントをキャッチするには、`$catchbp` **dbx** 変数を設定します。 `$catchbp` 変数が設定されている場合、別のスレッドのブレークポイントに達すると、残りの回数については **tnext** サブコマンドは繰り返されません。

例

1. 実行中のスレッドの実行を次のソース行まで継続するには、次のように入力します。

   ```
   tnext
   ```

2. 実行中のスレッドの実行を現行ソース行から数えて 3 番目のソース行まで継続するには、次のように入力します。

   ```
   tnext 3
   ```

tnexti サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

tnexti サブコマンド

tnexti [*Number*]

tnexti サブコマンドは、実行中のスレッドを次の命令まで実行します。 *Number* パラメーターは、**tnexti** サブコマンドの実行回数を指定します。 *Number* パラメーターを指定しない場合は、**tnexti** は 1 回だけ実行されます。このサブコマンドは、システム・スコープ・スレッドでのみ開始できます。

この操作時にすべてのスレッドが実行されます。この操作時にブレークポイントをキャッチするには、`$catchbp` **dbx** 変数を設定します。 `$catchbp` 変数が設定されている場合、別のスレッドのブレークポイントに達すると、残りの回数については **tnexti** サブコマンドは繰り返されません。

例

1. 実行中のスレッドの実行を次のマシン・インストラクションまで継続するには、次のように入力します。

   ```
   tnexti
   ```

2. 実行中のスレッドの実行を現行マシン・インストラクションから数えて 3 番目のマシン・インストラクションまで継続するには、次のように入力します。

   ```
   tnexti 3
   ```

tnext サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

trace サブコマンド

trace [*SourceLine* | *Expression* **at** *SourceLine* | *Procedure* | [*Variable*] [**at** *SourceLine* | **in** *Procedure*] | **on load** *ModuleName*] [**if** *Condition*]

trace サブコマンドは、プログラム実行時に、指定したプロシージャー、関数、ソース行、式、または変数に関するトレース情報を表示します。 *SourceLine* 変数は、整数として、または : (コロン) および整数が後ろに続くファイル名文字列として指定できます。条件も指定できます。 **dbx** デバッグ・プログラムは、番号と各 **trace** サブコマンドを関連付けます。これらの番号を表示する場合は、**status** サブコマンドを使用

します。トレースをオフにするには、**delete** サブコマンドを使用します。それぞれ、**enable** および **disable** サブコマンドを使用して、トレースを使用可能にすることおよび使用不可にすることができます。

trace サブコマンドは、デバッグされるプロセスによってモジュールがロードまたはアンロードされるときにトレース情報を表示できます。オプションの *ModuleName* パラメーターを使用すると、単一のモジュール名を、または、以下のフォーマットでモジュール名とメンバー名のペアを以下のように指定することができます。

ModuleName(*MemberName*)

ModuleName パラメーターを指定しないで **on load** フラグを使用すると、**dbx** コマンドはすべてのモジュールのロードとアンロードをトレースします。

デフォルトでは、トレースはプロセス・ベースで行われます。スレッド・ベースのトレースを行うには、例 8 に示す条件でスレッドを指定します。

フラグ

項目	説明
at *SourceLine*	トレースされる式を検索するソース行を指定します。
if *Condition*	トレースを開始する条件を指定します。トレースは、 **if** *Condition* が真のときにのみ開始されます。
in *Procedure*	トレースされるプロシージャーまたは変数の検索に使用するプロシージャーを指定します。
on load *ModuleName*	モニターするロード・モジュールを指定します。

例

1. printf プロシージャーに対する各呼び出しをトレースするには、次のように入力します。

   ```
   trace printf
   ```

2. hello.c ファイルの、 22 行目の各実行をトレースするには、次のように入力します。

   ```
   trace "hello.c":22
   ```

3. main プロシージャー内の、 *x* 変数の変更をトレースするには、次のように入力します。

   ```
   trace x in main
   ```

4. データ・アドレス 0x2004000 をトレースするには、次のように入力します。

   ```
   set $A=0x2004000
   trace $A
   ```

   注**: tracei** サブコマンドは、アドレスをトレースするように設計されています。

5. 指定した *Procedure* がアクティブな場合は、ソース行の表示を制限することができます。トレース情報を作成するタイミングを制御する場合は、オプションの *Condition* を指定することもできます。以下に例を示します。

   ```
   (dbx) trace in sub2
   [1] trace in sub2
   (dbx) run
   trace in hellosub.c:  8  printf("%s",s);
   trace in hellosub.c:  9  i = '5';
   trace in hellosub.c:  10  }
   ```

6. プロシージャーが呼び出されるたびに、あるいは戻されるたびに、メッセージを表示することができます。プロシージャーが呼び出される場合は、表示される情報には渡されたパラメーターと呼び出し側のルーチンの名前が含まれます。プロシージャーが戻される場合は、情報には *Procedure* の戻り値が含まれます。以下に例を示します。

```
(dbx) trace sub
[1] trace sub
(dbx) run
calling sub(s = "hello", a = -1, k = delete) from function main
returning "hello" from sub
```

7. プログラムが指定ソース行に達したときに、 *Expression* の値を表示することができます。行番号とファイルが表示されますが、ソース行は表示されません。以下に例を示します。

```
(dbx) trace x*17 at "hellosub.c":8 if (x > 0)
[1] trace x*17 at "hellosub.c":8 if x > 0
(dbx) run
at line 8 in file "hellosub.c": x*17 = 51

(dbx) trace x
[1] trace x
initially (at line 4 in "hello.c"):  x = 0
after line 17 in "hello.c":  x = 3
```

8. スレッド `$t1` により作成された、 *x* 変数の変更をトレースするには、次のように入力します。

```
(dbx) trace x if ($running_thread == 1)
```

9. すべてのモジュールのロードまたはアンロードをトレースするには、次のコマンドを入力します。

```
trace on load
```

10. モジュール `Module` のロードまたはアンロードをトレースするには、次のコマンドを入力します。

```
trace on load "Module"
```

11. モジュール `Module` のメンバー `Member` のロードまたはアンロードをトレースするには、次のコマンドを入力します。

```
trace on load "Module(Member)"
```

tracei サブコマンドも参照してください。

tracehwp サブコマンド

tracehwp *Address Size*

tracehwp サブコマンドは、ハードウェアの監視ポイントの停止を指定されたメモリー領域に設定します。**dbx** デバッグ・プログラムは、その領域の内容が変化したときにトレース情報を出力します。

注**:**

1. **tracehwp** サブコマンドが成功するかどうかは、ハードウェアに依存します。この機能は、POWER630 および POWER4 以降でのみ使用可能です。
2. 単一の監視ポイントのみ設定できるというハードウェアの制限により、 **stophwp** および **tracehwp** で別のハードウェア監視ポイント・イベントを作成しようとすると、アクティブな監視ポイント・イベントは矛盾するものとして動作することになります。そのため、新しいイベントを作成する前に、前のイベントは削除する必要があります。また、アクティブなソフトウェアの監視ポイントの存在 (**stop** および **trace** サブコマンドの呼び出しにより作成された) はハードウェアの監視ポイントのパフォーマンス向上を打ち消すため、これらのタイプのイベントも矛盾するものとして動作するので、ハードウェアの監視ポイントを作成する前に削除する必要があります。

例

1. アドレス `0x200004e8` で始まる 4 バイト・メモリー領域の内容が変化するたびにトレースするには、次のコマンドを入力します。

```
tracehwp 0x200004e8 4
```

stophwp サブコマンドを参照してください。

tracei サブコマンド

tracei [[*Address*] [**at** *Address* | **in** *Procedure*] | *Expression* **at** *Address*] [**if** *Condition*]

tracei サブコマンドは、以下の場合にトレースをオンにします。

- *Address* フラグが設定されている場合に、 *Address* パラメーターで指定したアドレスの内容が変更されたとき。
- **at** *Address* パラメーターを指定している場合に、その at *Address* 命令を実行したとき。
- **in** *Procedure* フラグが組み込まれている場合に、その *Procedure* で指定したプロシージャーがアクティブなとき。
- **if** *Condition* フラグが組み込まれている場合に、 *Condition* パラメーターで指定した条件が真のとき。

フラグ

項目	説明
at *Address*	アドレスを指定します。このアドレスにある命令が実行されると、トレースが使用可能になります。
if *Condition*	条件を指定します。この条件が満たされると、トレースが使用可能になります。
in *Procedure*	プロシージャーを指定します。このプロシージャーがアクティブなときに、トレースが使用可能になります。

例

1. 実行された各命令をトレースするには、次のコマンドを入力します。

   ```
   tracei
   ```

2. アドレス `0x100020f0` の命令が、実行されるたびにトレースするには、次のコマンドを入力します。

   ```
   tracei at 0x100020f0
   ```

3. `main` プロシージャーがアクティブなときに、メモリー位置 `0x20004020` の内容が変更されるたびにトレースするには、次のコマンドを入力します。

   ```
   tracei 0x20004020 in main
   ```

4. アドレス `0x100020f0` の命令が、スレッド `$t4` により実行されるたびにトレースするには、次のコマンドを入力します。

   ```
   tracei at 0x100020f0 if ($running_thread == 4)
   ```

trace サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『**dbx** によるマシン・レベルのデバッグ』も参照してください。

tskip サブコマンド

tskip [*Number*]

tskip サブコマンドは、実行中のスレッドの実行を現在の停止位置から継続します。 *Number* パラメーターで指定されたスレッド・レベル・ブレークポイントの数は、実行中のスレッドについてはスキップされます。このサブコマンドは、システム・スコープ・スレッドでのみ開始できます。

他のすべてのスレッドはこの操作中に実行され、ユーザーが指定したすべてのブレークポイントおよび監視ポイントがキャッチされます。ブレークポイントまたは監視ポイントに達したスレッドが 1 つでもあると、実行が停止される場合があります。 **tskip** サブコマンドによって開始された実行が別のスレッドのイベントのために停止しても、前のスレッドについて指定された **tskip** カウントは依然としてアクティブで

あり、その **tskip** カウントによって指定されたスレッド・レベル・ブレークポイント数は、プロセスの継続時にそのスレッドについては無視されます。スレッドが終了すると、そのスレッドに関連した **tskip** カウントは削除されます。

残りのスレッドの **tskip** カウントを表示するには、**status** サブコマンドを使用してください。残りのスレッドの **tskip** カウントを削除するには、**delete** サブコマンドを使用してください。

例

実行中のスレッドについて、現行停止位置から数えて 2 番目のブレークポイントを検出するまで実行を継続するには、次のように入力します。

```
tskip 1
```

cont サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

tstep サブコマンド

tstep [*Number*]

tstep サブコマンドは、実行中のスレッドについて、現行ソース行から数えて指定数のソース行を実行します。 *Number* パラメーターは、**tstep** サブコマンドの実行回数を指定します。 *Number* パラメーターを指定しない場合は、**tstep** は 1 回だけ実行されます。このサブコマンドは、システム・スコープ・スレッドでのみ開始できます。

この操作時にすべてのスレッドが実行されます。 `$hold_next` が設定されている場合は、実行中のスレッド以外のすべてのスレッドが保留されます。

注**: tstep** サブコマンドの動作を制御するには、**set** サブコマンドの `$stepignore` 変数を使用します。`$stepignore` 変数を使用すると、**tstep** サブコマンドはデバッグ情報が使用できない大規模なルーチンを横切って次へ進むことができます。

例

1. 実行中のスレッドの実行を 1 ソース行だけ継続するには、次のように入力します。

   ```
   tstep
   ```
2. 実行中のスレッドの実行を 5 ソース行だけ継続するには、次のように入力します。

   ```
   tstep 5
   ```
3. **dbx** プログラムが次のサンプル・コードに示すような **printf** 関数に単一ステップで進まないようにするには、

   ```
   60 printf ("hello world /n");
   ```

   次のように入力します。

   ```
   set $stepignore="function"; step
   ```

cont サブコマンド、goto サブコマンド、**tnext** サブコマンド、set サブコマンド、および **tstepi** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

tstepi サブコマンド

tstepi [*Number*]

tstepi サブコマンドは、実行中のスレッドについて、現行命令から数えて指定数の命令を実行します。 *Number* パラメーターは、**tstepi** サブコマンドの実行回数を指定します。 *Number* パラメーターを指定しない場合は、**tstepi** は 1 回だけ実行されます。このサブコマンドは、システム・スコープ・スレッドでのみ開始できます。

この操作時にすべてのスレッドが実行されます。 `$hold_next` が設定されている場合は、実行中のスレッド以外のすべてのスレッドが保留されます。

例

1. 実行中のスレッドの実行を 1 マシン・インストラクションだけ継続するには、次のように入力します。

   ```
   tstepi
   ```

2. 実行中のスレッドの実行を 2 マシン・インストラクションだけ継続するには、次のように入力します。

   ```
   tstepi 5
   ```

gotoi サブコマンド、**tnexti** サブコマンド、および **tstep** サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

tstop サブコマンド

tstop { **in** *Procedure* | [*Variable*] **at** *SourceLine* [**if** *Condition*] } [**for $t***threadnumber*]

tstop サブコマンドは、スレッドのソース・レベルのブレークポイント停止を設定し、指定されたスレッドがブレークポイントに達するとアプリケーション・プログラムを一時停止します。指定されたスレッドは、イベントの作成と同時に存在していなければなりません。スレッドが 1 つも指定されていない場合は、現行スレッドが使用されます。次の条件のいずれかが起こると、指定されたスレッドは停止されます。

- **if** *Condition* フラグが使用され、*Condition* が True である。
- **in** *Procedure* フラグが使用され、*Procedure* が呼び出された。
- **at** *SourceLine* フラグが使用され、*SourceLine* 行番号に達した。*SourceLine* 変数は、整数として、または : (コロン) と整数が続くファイル名文字列として指定できます。

スレッド・レベルのブレークポイントは、システム・スコープ・スレッドでのみ設定できます。スレッド・レベルおよびプロセス・レベルのブレークポイントが同時に検出された場合、両方のブレークポイントが処理され、スレッド・レベルのブレークポイントが報告されます。スレッドが終了すると、そのスレッドに関連したイベントは削除されます。

フラグ

項目	説明
at *SourceLine*	行番号を指定します。
for $t *threadnumber*	スレッド番号を指定します。
if *Condition*	条件 (例えば、true) を指定します。
in *Procedure*	呼び出されるプロシージャーを指定します。

例

1. スレッド 2 の実行中に **func** プロシージャーの最初のステートメントで実行を停止するには、次のように入力します。

   ```
   tstop in func for $t2
   ```

2. 実行の行 12 で x 変数の値が変更されたときに現行スレッドの実行を停止するには、次のように入力します。

   ```
   tstop x at 12
   ```

ttrace サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

tstophwp サブコマンド

tstophwp *address size* [**for $t***threadnumber*]

tstophwp サブコマンドは、指定されたメモリー領域のスレッド・レベルのハードウェア監視ポイントを設定します。プログラムは、指定されたスレッドの実行中にその領域の内容が変化したときに停止します。指定されたスレッドは、イベントの作成と同時に存在していなければなりません。スレッドが 1 つも指定されていない場合は、現行スレッドが使用されます。スレッド・レベルの監視ポイント・イベントは、システム・スコープ・スレッドにのみ設定できます。スレッドが終了すると、そのスレッドに関連したイベントは削除されます。

注**:**

1. **tstophwp** サブコマンドが成功するかどうかは、ハードウェアに依存します。この機能は、POWER630 および POWER4 以降でのみ使用可能です。
2. 単一の監視ポイントのみ設定できるというハードウェアの制限により、**tstophwp** および **ttracehwp** を使用して別のハードウェア監視ポイント・イベントを作成しようとすると、アクティブなスレッド監視ポイント・イベントは矛盾するものとして動作することになります。これを避けるには、新しいイベントを作成する前に、前のイベントを削除する必要があります。また、アクティブなソフトウェア監視ポイント (**stop** および **trace** サブコマンドの呼び出しにより作成された) が存在すると、ハードウェア監視ポイントのパフォーマンス向上が打ち消されるため、これらのタイプのイベントも、ハードウェア監視ポイントを作成する前に削除して、矛盾を回避する必要があります。
3. プロセス・レベルの監視ポイントが存在する場合、スレッド・レベルの監視ポイントを持たないスレッドが、プロセス監視ポイントの位置を監視します。スレッドにスレッド・レベルの監視ポイントがある場合、そのスレッドはスレッド監視ポイントの位置を監視します。
4. スレッド・レベルのハードウェア監視ポイントとプロセス・レベルのハードウェア監視ポイントは、互いに矛盾せずに共存できます。
5. 同じアドレスについてプロセス・レベルの監視ポイントとスレッド・レベルの監視ポイントが存在する場合、プロセス・レベルの監視ポイント・イベントが報告されます。

フラグ

項目	説明
for $t *threadnumber*	スレッド番号を指定します。

例

スレッド 2 の実行中にアドレス 0x200004e8 で始まる 4 バイト・メモリー領域の内容が変化したときにプログラムを停止するには、次のように入力します。

```
tstophwp 0x200004e8 4 for $t2
```

ttracehwp サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

tstopi サブコマンド

tstopi { **in** *Procedure* | [*Address*] **at** *Address* [**if** *Condition*] } [**for $t***threadnumber*]

tstopi サブコマンドは、スレッドの命令レベルのブレークポイント停止を設定します。指定されたスレッドは、イベントの作成と同時に存在していなければなりません。スレッドが 1 つも指定されていない場合は、現行スレッドが使用されます。次の条件のいずれかが起こると、指定されたスレッドは停止されます。

- **if** *Condition* フラグが使用され、*Condition* が True である。
- **in** *Procedure* フラグが使用され、*Procedure* が呼び出された。
- **at** *Address* フラグが使用され、*Address* に達した。

スレッド・レベルのブレークポイントは、システム・スコープ・スレッドでのみ設定できます。スレッド・レベルのブレークポイントとプロセス・レベルのブレークポイントに同時に達した場合、両方のブレークポイントが処理され、スレッド・レベルのブレークポイントが報告されます。スレッドが終了すると、そのスレッドに関連したイベントは削除されます。

フラグ

項目	説明
at *Address*	マシン・インストラクションのアドレスを指定します。
for $t *threadnumber*	スレッド番号を指定します。
if *Condition*	条件を指定します。
in *Procedure*	呼び出されるプロシージャーを指定します。

例

1. スレッド 2 の実行中にアドレス 0x100020f0 で実行を停止するには、次のように入力します。

   ```
   tstopi at 0x100020f0 for $t2
   ```

2. 現行スレッドの実行中に **func** プロシージャーが入力されたときに実行を停止するには、次のように入力します。

   ```
   tstopi in func
   ```

ttracei サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

ttrace サブコマンド

ttrace { [*Variable*] **at** *SourceLine* | *Procedure* } [**if** *Condition*] [**for $t***threadnumber*]

ttrace サブコマンドは、指定されたスレッドの実行時に、指定されたプロシージャー、関数、ソース行、および変数に関するトレース情報を表示します。 *SourceLine* 変数は、整数として、または : (コロン) と整数が続くファイル名文字列として指定できます。**dbx** デバッグ・プログラムは、番号と各 **ttrace** サブコマンドを関連付けます。これらの番号を表示する場合は、**status** サブコマンドを使用します。トレースをオフにするには、**delete** サブコマンドを使用します。それぞれ、**enable** および **disable** サブコマンドを使用して、トレースを使用可能にすることおよび使用不可にすることができます。

スレッドが 1 つも指定されていない場合は、現行スレッドが使用されます。スレッド・レベルのトレースは、システム・スコープ・スレッドにのみ設定できます。指定されたスレッドは、イベントの作成と同時に存在していなければなりません。スレッドが終了すると、それに関連したイベントが削除されます。

フラグ

項目	説明
at *SourceLine*	トレースされる式を検索するソース行を指定します。
for $t *threadnumber*	スレッド番号を指定します。
if *Condition*	トレースを開始する条件を指定します。トレースは *Condition* が True の場合にのみ開始されます。
in *Procedure*	トレースされるプロシージャーまたは変数を検索するプロシージャーを指定します。

例

1. スレッド 2 の実行中に **printf** プロシージャーへの各呼び出しをトレースするには、次のように入力します。

   ```
   ttrace printf for $t2
   ```

2. 現行スレッドの実行中に `hello.c/` ファイルの行 22 の各実行をトレースするには、次のように入力します。

   ```
   ttrace "hello.c":22
   ```

ttracei サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

ttracei サブコマンド

ttracei [*Address*] **at** *Address* [**if** *Condition*] } [**for $t***threadnumber*]

次の条件のいずれかが起こると、**ttracei** サブコマンドは指定されたスレッドのトレースをオンにします。

- **if** *Condition* フラグが組み込まれ、*Condition* が True である。
- **at** *Address* フラグが指定され、*Address* の命令が実行された。

スレッドが 1 つも指定されていない場合は、現行スレッドが使用されます。スレッド・レベルのトレースは、システム・スコープ・スレッドにのみ設定できます。指定されたスレッドは、イベントの作成時に存在していなければなりません。スレッドが終了すると、それに関連したイベントが削除されます。

フラグ

項目	説明
at *Address*	アドレスを指定します。このアドレスにある命令が実行されると、トレースが使用可能になります。
for $t *threadnumber*	スレッド番号を指定します。
if *Condition*	条件を指定します。この条件が満たされると、トレースが使用可能になります。

例

1. スレッド 3 の実行中にアドレス 0x100020f0 の命令が実行されるたびにトレースするには、次のように入力します。

   ```
   tracei at 0x100020f0 for $t3
   ```

2. 現行スレッドによってアドレス 0x100020f0 の命令が実行されるたびにトレースするには、次のように入力します。

   ```
   tracei at 0x100020f0
   ```

ttrace サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

ttracehwp サブコマンド

ttracehwp *address size* [**for $t***threadnumber*]

ttracehwp サブコマンドは、指定されたメモリー領域に対してスレッド・レベルのハードウェア監視ポイントのトレースを設定します。 **dbx** デバッグ・プログラムは、指定されたスレッドの実行中にその領域の内容が変化したときにトレース情報を表示します。指定されたスレッドは、イベントの作成と同時に存在していなければなりません。スレッドが 1 つも指定されていない場合は、現行スレッドが使用されます。スレッド・レベルの監視ポイント・イベントは、システム・スコープ・スレッドにのみ設定できます。スレッドが終了すると、そのスレッドに関連したイベントは削除されます。

注**:**

1. **ttracehwp** サブコマンドが成功するかどうかは、ハードウェアに依存します。この機能は、POWER630 および POWER4 以降でのみ使用可能です。
2. 単一の監視ポイントのみ設定できるというハードウェアの制限により、**tstophwp** および **ttracehwp** を使用して別のハードウェア監視ポイント・イベントを作成しようとすると、アクティブなスレッド監視ポイント・イベントは矛盾するものとして動作することになります。これを避けるには、新しいイベントを作成する前に、前のイベントを削除する必要があります。また、アクティブなソフトウェア監視ポイント (**stop** および **trace** サブコマンドの呼び出しにより作成された) が存在すると、ハードウェア監視ポイントのパフォーマンス向上が打ち消されるため、これらのタイプのイベントも、ハードウェア監視ポイントを作成する前に削除して、矛盾を回避する必要があります。
3. プロセス・レベルの監視ポイントが存在する場合、スレッド・レベルの監視ポイントを持たないスレッドが、プロセス監視ポイントの位置を監視します。スレッドにスレッド・レベルの監視ポイントがある場合、そのスレッドはスレッド監視ポイントの位置を監視します。
4. スレッド・レベルのハードウェア監視ポイントとプロセス・レベルのハードウェア監視ポイントは、互いに矛盾せずに共存できます。
5. 同じアドレスについてプロセス・レベルの監視ポイントとスレッド・レベルの監視ポイントが存在する場合、プロセス・レベルの監視ポイント・イベントが報告されます。

フラグ

項目	説明
for $t *threadnumber*	スレッド番号を指定します。

例

スレッド 2 の実行中にアドレス 0x200004e8 で始まる 4 バイト・メモリー領域の内容が変化するたびにトレースするには、次のように入力します。

```
ttracehwp 0x200004e8 4 for $t2
```

tstophwp サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『複数のスレッドを含むプログラムのデバッグ』を参照してください。

unalias サブコマンド

unalias *Name*

unalias サブコマンドは、 *Name* パラメーターにより指定された別名を除去します。

例

別名 printx を除去するには、次のように入力します。

```
unalias printx
```

alias サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『サブコマンドの別名の作成』も参照してください。

unset サブコマンド

unset *Name*

unset サブコマンドは、 *Name* パラメーターにより指定された名前と関連付けられている **dbx** デバッグ・プログラムの変数を削除します。

例

浮動小数点レジスターの表示を禁止する変数を削除するには、次のように入力します。

```
unset $noflregs
```

set サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『特殊なデバッグ変数による印刷出力の変更』も参照してください。

up サブコマンド

up [*Count*]

up サブコマンドは、現行関数をスタックの上方へ *Count* 番号分移動させます。現行関数は、名前の解決に使用します。*Count* パラメーターのデフォルト値は 1 です。

例

1. 現行関数をスタックの上方へ 2 レベル移動するには、次のように入力します。

   ```
   up 2
   ```

2. スタック上の現行関数を表示するには、次のように入力します。

```
up 0
```

down サブコマンドを参照してください。また、 プログラミングの一般概念: プログラムの作成およびデバッグ の『現在のファイルまたはプロシージャーの変更』および『スタック・トレースの表示』も参照してください。

use サブコマンド

use [{ + | *Directory* | '['*RegularExpression* = *NewPath*']' } ...]

use サブコマンドは、**dbx** デバッグ・プログラムがソース・ファイルを検索するときの検索先となるディレクトリーのリスト、および適用されるパス・マッピングを設定します。引数を指定せずに **use** サブコマンドを指定すると、検索対象のディレクトリーと適用されるパス・マッピングの現行リストが表示されます。

@ (アットマーク) は、**dbx** プログラムに対して、オブジェクト・ファイル内に絶対パス名の情報があればそれを検索するように指示する特殊ディレクトリーです。 **@** という名前の相対ディレクトリーを検索する場合は、検索パスに **./@** を使用する必要があります。

use サブコマンドは、**+** (正符号) を使用して、検索対象となるディレクトリーのリストにディレクトリーまたはマッピングを追加します。**+** は、**use** サブコマンドへの入力として指定される場合は、ディレクトリーとマッピングの現行リストを表します。現行リストの最後にディレクトリーまたはマッピングを付加するには、新しいディレクトリーまたはマッピングの前に **+** を指定する必要があります。ディレクトリーを現行リストの先頭に追加するには、新しいディレクトリーまたはマッピングの後に **+** を指定する必要があります。**+** という名前のディレクトリーがある場合は、そのディレクトリーの絶対パス名 (例えば、**./+** または **/tmp/+** など) を指定します。

use サブコマンドは、**[** と **]** (大括弧) で囲まれている文字列を解釈します。この大括弧の中には、パス・マッピングとして **=** (等号) が入っています。これらのパス・マッピングは、特殊 **@** ディレクトリーと一緒に使用されます。これを使用すると、ソース・ファイルのディレクトリー構造全体がコンパイル後に再配置された場合に、ユーザーがソース・ファイルの場所を示すのがより簡単になります。

デバッグ中にソース・ファイルを見つけようとするときには、以下の規則が適用されます。

- リスト内のディレクトリーが、指定の順序で評価されます。
- リスト内のディレクトリーの評価時には、指定のファイルのためにディレクトリーが検索されます。そのディレクトリーにファイルが入っていて、読み取り可能であるときは、このファイルが使用されます。
- 特殊 **@** ディレクトリーの評価時に、1 つ以上のパス・マッピングが指定されていて、パス・マッピングの *RegularExpression* 部分がオブジェクト・ファイル内でファイルの絶対パス名情報の最初の *n* 文字に一致していて、パス・マッピングの *NewPath* 部分の置換によって読み取り可能ファイルが得られる場合は、このファイルが使用されます。
- 特殊 **@** ディレクトリーの評価時に、パス・マッピングが指定されていないか、または一致がない場合、絶対パス名情報に対応するディレクトリーが検索されます。そのディレクトリーにファイルが入っていて、読み取り可能であるときは、このファイルが使用されます。
- 複数のパス・マッピングから読み取り可能ファイルが得られる場合は、*RegularExpression* が絶対パス名情報と最も多い文字数 (1 ... n) で一致する (すなわち最も特定的な情報を持つ) パス・マッピングが適用され、その結果のファイルが使用されます。

- 複数のパス・マッピングから読み取り可能ファイルが得られ、しかも、パス・マッピングの特定の度合いが等しい場合は、リストの最初に最も近いパス・マッピングが適用され、その結果のファイルが使用されます。

注: 特殊 @ ディレクトリーがリストのメンバーに入っていない場合は、指定が可能であったどのパス・マッピングも無視されます。

例

1. 検索されるディレクトリーのリストを、現行ディレクトリー (.)、その親ディレクトリー (..)、および /tmp ディレクトリーに変更するには、次のように入力します。

   ```
   use . .. /tmp
   ```

2. 検索されるディレクトリーのリストを、現行ディレクトリー (.)、コンパイル時にソース・ファイル格納場所であったディレクトリー (@)、および ../source ディレクトリーに変更するには、次のように入力します。

   ```
   use . @ ../source
   ```

3. 検索されるディレクトリーのリストに、 /tmp2 ディレクトリーを追加するには、次のように入力します。

   ```
   use + /tmp2
   ```

4. 検索されるディレクトリーのリストの先頭に /tmp3 ディレクトリーを追加するには、次のように入力します。

   ```
   use /tmp3 +
   ```

5. 絶対パス名情報が /home/developer で始まるソース・ファイルが現在は /mnt の下に配置されていることを示すには、次のように入力します。

   ```
   use + [/home/developer=/mnt]
   ```

6. /home/developer で始まる絶対パス名情報をもつファイルを、最初に /latest から検索し、次にそのファイルがそこになければ、 /stable から検索するように **dbx** プログラムに指示するには、次のように入力します。

   ```
   use + [/home/developer=/latest] [/home/developer=/stable]
   ```

edit サブコマンド、 **list** サブコマンドも参照してください。

whatis サブコマンド

whatis *Name*

whatis サブコマンドは、*Name* パラメーターで指定した変数名、プロシージャー名、または関数名を指定する *Name* の宣言を表示します。パラメーターは、オプションとしてブロック名で修飾することができます。

注: **whatis** サブコマンドを使用できるのは、**dbx** デバッグ・プログラムの実行中に限られます。

例

1. x 変数の宣言を表示するには、次のように入力します。

   ```
   whatis x
   ```

2. main プロシージャーの宣言を表示するには、次のように入力します。

   ```
   whatis main
   ```

3. main 関数内の、 x 変数の宣言を表示するには、次のように入力します。

```
whatis main.x
```

4. 列挙型、構造、または共用体タグの宣言を表示するには、$$TagName を使用します。

```
(dbx) whatis $$status
enum $$status { run, create, delete, suspend };
```

where サブコマンド

where [all | *$tthreadumber* [(*startframe endframe*)] ...] [*startframe endframe*] [>*File*]

where サブコマンドは、フレーム番号 *startframe* から *endframe* までに関連したアクティブなプロシージャーおよび関数のリストを表示します。スタック・フレームの番号付けは、現在アクティブな関数のスタック・フレームから始まります (このフレームの番号は常に 0 となります)。 *n* 個のフレームがある場合、**main** 関数のフレームの番号は n-1 になります。>*File* フラグを使用すると、このサブコマンドの出力を指定したファイルにリダイレクトできます。

マルチスレッド環境では、オプション **all** を指定すると、使用可能なすべてのスレッドのスタックの詳細が表示されます。各スレッドのスタックの詳細を表示するには、where サブコマンドを使用してスレッド番号を指定します。各スレッドの開始フレームと終了フレームを指定しないと、グローバルの開始フレーム番号と終了フレーム番号でスタック・フレームが表示されます。オプションなしでコマンドを使用すると、現行スレッドのスタック・フレームが表示されます。

フラグ

項目	説明
>*File*	指定したファイルに出力をリダイレクトします。

frame サブコマンド、**up** サブコマンド、および **down** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『スタック・トレースの表示』も参照してください。

例

1. すべてのスレッドのスタックの詳細を表示するには、次のように入力します。

```
where all
```

2. スレッド $t1、$t2、および $t3 のスタックの詳細を表示するには、次のように入力します。

```
where $t1 $t2 $t3
```

3. スレッド $t2 のスタックの詳細をスタック・フレーム 2-3 で表示し、$t1 と $t3 のスタックの詳細を両方ともスタック・フレーム 1-4 で表示するには、次のように入力します。

```
where $t1 $t2(2 3) $t3 1 4
```

frame サブコマンド、**up** サブコマンド、および **down** サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『スタック・トレースの表示』も参照してください。

whereis サブコマンド

whereis *Identifier*

whereis サブコマンドは、指定した ID と名前が一致するすべての記号の完全修飾を表示します。記号が表示される順序には特に意味はありません。

例

x という名前のすべての記号の修飾名を表示するには、次のように入力します。

```
whereis x
```

which サブコマンドも参照してください。

which サブコマンド

which *Identifier*

which サブコマンドは、ID の完全修飾を表示します。完全修飾は、その ID が関連付けられている外部のブロックのリストで構成されています。

例

x 記号の完全修飾を表示するには、次のように入力します。

```
which x
```

whereis サブコマンドを参照してください。また、 プログラミングの一般概念*:* プログラムの作成およびデバッグ の『名前の有効範囲化』のセクションも参照してください。

ファイル

項目	説明
a.out	オブジェクト・ファイル。オブジェクト・コードが入っています。
core	メモリー・ダンプが入っています。
.dbxinit	初期コマンドが入っています。

トランザクション・メモリーを使用するアプリケーションのデバッグ

トランザクション・メモリー (TM) を使用するアプリケーションで、最も信頼性の高いデバッグ・エイドは、トランザクション例外およびサマリー・レジスター (*$texasr*)、トランザクション失敗ハンドラー・アドレス・レジスター (*$tfhar*)、およびトランザクション失敗命令アドレス・レジスター (*$tfiar*) の各変数です。

$texasr*、*$tfhar、および *$tfiar* の各変数は、**print** サブコマンドを使用して表示できます。これは、*$iar* 変数を表示するのに似ています。ただし、これらのレジスターの値は、**assign** サブコマンドを使用して操作することはできません。

$tfhar 変数および *$tfiar* 変数には、トランザクション失敗でデバッグされたテキスト・セグメントからの命令のアドレスが含まれています。*$iar* と同様に、*$tfhar* および *$tfiar* のレジスター変数の 2 つの最上位ビットは予約済みであり、これらのレジスター変数からアドレスを読み取ることなど考えられません。

注**:** $tfiar および $tfhar のレジスター変数を **list** サブコマンドと併用すると、**dbx** コマンドは、これらのレジスター変数の内容から 2 つの最上位ビットを除外したのち、アドレスを取り出します。

```
(dbx) list at $tfiar
```

または

```
(dbx) list at $tfhar
```

トランザクション失敗の原因は、*$texasr* 変数を使用して判別できます。 **tm_status** サブコマンドは、トランザクション失敗の原因および種類を解釈します。

dbx コマンドは、トランザクションのオカレンスをモニターし、以下の一連のメッセージを使用してトランザクション失敗の原因を表示します。 **run** サブコマンド、**rerun** サブコマンド、または **continue** サブコマンドを実行した後、以下のメッセージが表示されます。

- `Process {PID} may have failed a transaction - $texasr, $tfiar, $tfhar are valid and may be inspected`
- `Process {PID} may have performed a transaction - $texasr, $tfiar, $tfhar, are valid and may be inspected`

  このメッセージが表示されるのは、*$trace_good_transaction* 内部変数が設定されている場合のみです。
- `Process {PID} is in Transactional State – debugging efforts through dbx may result in repeated transaction failure or undefined behavior`
- `Process {PID} is in Suspended State – debugging efforts through dbx may result in repeated transaction failure or undefined behavior`

ここで、`PID` は、デバッグされているプロセスのプロセス ID です。

関連情報:

adb コマンド

a.out コマンド

ptrace サブルーチン

デバッグ・プログラム

---

dc コマンド

目的

任意精度の整数の算術演算を実行するための対話式卓上計算機を提供します。

構文

dc [*File*]

説明

dc コマンドは、任意精度の算術演算を実行する計算機です。**dc** コマンドは、ファイル終わり文字を読み取るまで、*File* パラメーターまたは標準入力から入力を取り込んで、標準出力に出力します。 **dc** コマンドは入力を受信すると、その値を評価して、評価を標準出力に書き込みます。このコマンドは 10 進整数について演算を実行しますが、入力基数、出力基数、および保持される小数桁の数を指定できます。**dc** コマンドは、スタックされた逆ポーランド表記法を用いる計算機として構成されています。

bc コマンドは、 **dc** コマンドのプリプロセッサーです。表記法は挿入表記で、構文はプログラムに対して関数および制御構造を使用する C 言語に類似しています。

サブコマンド

項目	説明
c	スタックをクリアします。**dc** コマンドは、スタックのすべての値をポップします。
d	スタックの一番上の値の複製を作成します。
f	スタックのすべての値を表示します。
i	スタックの一番上の値をポップして、さらに入力するために、その値をその数の基数として使用します。
I	入力基数をスタックの一番上にプッシュします。
k	スタックの一番上をポップして、その値を非負スケール・ファクターとして使用します。適切な桁数の数値が出力時に表示されて、乗算、除算、べき乗の間、保持されます。スケール・ファクター、入力基数、および出力基数は、すべてが一緒に変更される場合は妥当です。
l*x*	*x* 変数が示しているレジスター内の値をスタックにプッシュします。*x* 変数が示すレジスターは変更されません。レジスターはすべて、0 という値で始まります。
L*x*	*x* 変数をスタックとして扱い、その一番上の値をメイン・スタックにポップします。
o	スタックの一番上の値をポップし、以降の出力のために、その値を基数として使用します。
O	出力基数をスタックの一番上にプッシュします。
p	スタックの一番上の値を表示します。一番上の値は、変更されません。
P	スタックの一番上を文字列として解釈して、それを除去し、表示します。
q	プログラムを終了します。**dc** コマンドが文字列を実行中の場合は、再帰レベルを 2 レベルだけポップします。
Q	スタックおよび文字列実行レベルの一番上の値を、その値分だけポップします。
s*x*	スタックの一番上をポップして、それを *x* という名前のレジスターに格納します。この *x* 変数は文字でもかまいません。
S*x*	*x* 変数をスタックとして扱います。メイン・スタックの一番上をポップして、 *x* 変数が示すスタックにその値をプッシュします。
v	スタックの一番上のエレメントを、その平方根と置き換えます。オプションに小数部分があれば、それは平方根に反映されますが、そうでない場合は、スケール・ファクターは無視されます。
x	スタックの一番上のエレメントを文字列と見なし、それを **dc** コマンドの文字列として実行します。
X	スタックの一番上にある番号を、そのスケール・ファクターと置き換えます。
z	スタック内のエレメントの数をスタックにプッシュします。
Z	スタックの一番上の数字を、その数字の桁数と置き換えます。
Number	指定した値をスタックにプッシュします。 *Number* は、0 から 9 桁の連続した文字列です。負の数を指定するには、前に _ (下線) を付けます。数字には小数点を入れることができます。
+ - / * % ^	スタック上の最初の 2 つの値の加算 (+)、減算 (-)、乗算 (*)、除算 (/)、剰余計算 (%)、またはべき乗 (^) を実行します。 **dc** コマンドは、最初の 2 つの入力をスタックからポップして、それらの代わりに結果をスタックにプッシュします。 **dc** コマンドは、指数の小数部分を無視します。
[*String*]	大括弧で囲まれた *String* パラメーターを、スタックの一番上に取り込みます。
[= | > | <] *x*	スタックの最初の 2 つのエレメントをポップして、それらを比較します。宣言された関係に従って、*x* 変数が示すレジスターを評価します。
!	行の残りの部分をオペレーティング・システムのコマンドとして解釈します。
?	1 行の入力を取り込んで実行します。
;:	**bc** コマンドは、配列操作でこれらの文字を使用します。

例

1. **dc** コマンドを計算機として使用するには、次のコマンドを入力します。

```
You:    1 4 / p
System: 0
You:    1 k   [ Keep 1 decimal place ]s.
        1 4 / p
System: 0.2
You:    3 k   [ Keep 3 decimal places ]s.
        1 4 / p
System: 0.250
```

```
You:    16 63 5 / + p
System: 28.600
You:    16 63 5 + / p
System: 0.235
```

例に示すように、**dc** コマンドの中でコメントを使用できます。コメントは大括弧で囲み、直後に `s.` 文字を続けることができます。`[Comment]s.` の形式のコメントは、**dc** コマンドによって無視されます。大括弧で囲まれたコメントだけが、スタックの上部に格納されます。

キーボードから直接 **dc** コマンドを入力する場合は、 Ctrl-D を押して **bc** コマンド・セッションを終了し、シェル・コマンド・ラインに戻ります。

2. **dc** プログラム・ファイルをロードして実行するには、次のコマンドを入力します。

```
You:    dc prog.dc
        5 lf x p [ 5 factorial ]s.
System: 120
You:    10 lf x p [ 10 factorial ]s.
System: 3628800
```

この入力は、`prog.c` プログラム・ファイルに保存された **dc** プログラムを解釈して、ワークステーションのキーボードから読み取りを行います。`lf x` は、レジスター `f` に格納されている関数を評価します。このレジスターは、`prog.c` プログラム・ファイルで次のように定義することができます。

```
[ f: compute the factorial of n ]s.
[ (n = the top of the stack) ]s.
[ If 1>n do b; If 1<n do r ]s.
 [d 1 >b d 1 <r] sf
[ Return f(n) = 1 ]s.
 [d - 1 +] sb
[ Return f(n) = n * f(n-1) ]s.
 [d 1 - lf x *] sr
```

dc プログラム・ファイルは、任意のテキスト・エディターまたは **bc** コマンドの **-c** (コンパイル) フラグを付けることによって作成できます。キーボードから直接 **dc** コマンドを入力する場合は、 Ctrl-D を押して **bc** コマンド・セッションを終了し、シェル・コマンド・ラインに戻ります。

ファイル

項目	説明
/usr/bin/dc	**dc** コマンドが入っています。

関連情報:

bc コマンド

---

dcp コマンド

目的

複数のノードおよびハードウェア・デバイスでコマンドを並行して実行します。

構文

dcp [**-h**] [**-V**] [**-q**] [**-a**] [**--all-nodes** *context_list*] [**-A**] [**--all-devices** *context_list*] [**-n** *node_list*] [**-N** *nodegroups*] [**-d** *device_list*] [**-D** *devicegroups*] [**-C** *context*] [**-f** *fanout*] [**-l** *user_ID*] [**-o** *node_options*] [**-O** *device_options*] [**-p**] [**-P**] [**-Q**] [**-r** *node_remote_copy*] [**--device-rcp** *device_remote_copy*] [**-R**] [**-t** *timeout*] [**-X** *env_list*] [**-T**] [**-v**] *source_file... target_path*

説明

dcp コマンドは、リモート側のターゲット・ノード、ハードウェア・デバイス、あるいは両方との間で並行してファイルをコピーします。ターゲットは、複数のコンテキストから選択できます。コンテキストは、ノードおよびデバイスの定義を含むターゲット・データベース (NIM など) です。**dcp** コマンドは、指定されたノードまたはデバイスごとにリモート・コピー・コマンドを発行します。ファイルは、ターゲットからプルされると、コピーされた *source_file* 名にリモート・ノード名またはリモート・デバイス名が付加されて *target_path* に配置されます。**/usr/bin/rcp** コマンドは、構文およびセキュリティーのモデルです。**dcp** コマンドは、DSM 分散シェル・ユーティリティーです。**dsh** の構成および環境設定は、**dcp** の動作に影響を与えます。 詳細については、**dsh** コマンドを参照してください。

パラメーター

項目	説明
ターゲット・コンテキスト	ターゲット・コンテキストの指定は、**dcp** コマンドと **dsh** コマンドで同じです。**dcp** コマンドのコンテキストの指定について詳しくは、**dsh** マニュアル・ページの『ターゲット・コンテキスト』を参照してください。
ターゲットの指定	ターゲットの指定は、**dcp** コマンドと **dsh** コマンドで同じです。**dcp** コマンドのターゲットの指定について詳しくは、**dsh** マニュアル・ページを参照してください。
ターゲット・リスト	ターゲット・リスト構文は、**dcp** コマンドと **dsh** コマンドで同じです。
リモート・ユーザー	リモート・コピー・コマンドに対して *user_ID* を指定できます。リモート・ユーザーの指定は、**dcp** コマンドと **dsh** コマンドで同じです。

項目	説明
リモート・コピー・コマンド	**dcp** コマンドは、構成可能なリモート・コピー・コマンドを使用して、リモート・ターゲットでリモート・コマンドを実行します。AIX リモート・シェル **rcp** コマンド、**OpenSSH scp** コマンド、および **rsync** コマンドに対するサポートが明示的に提供されています。ノード・ターゲットについては、次の優先順位でパラメーターを使用して、リモート・コピー・コマンドが決定されます。 1. **-r** フラグ 2. **DCP_NODE_RCP** 環境変数 3. **/usr/bin/rcp** コマンド デバイス・ターゲットについては、次の優先順位でリモート・シェルが決定されます。 1. **--device-rcp** フラグ 2. **DCP_DEVICE_RCP** 環境変数 3. ターゲット・コンテキストによって定義されたデフォルト・デバイスのリモート・コピー・コマンド。 4. デバイス・ターゲット用に定義された **RemoteCopyCmd** 属性。 リモート・コピー・コマンドは、次の構文を使用して、コマンド・ライン・フラグまたは環境変数と一緒に指定されます。 [*context*:]*path*[,[*context*:]*path*]... ここで、*path* は、リモート・コピー・コマンドへのパスであり、*context:* はファイルのコピーに使用するリモート・コピー・コマンドのコンテキストを示します。コンテキストを指定せずにリモート・コピー・コマンド・パスを指定すると、そのパスは、リストで明示的なリモート・コピー・コマンド・パスが指定されていない他のすべてのコンテキストに適用されます。リモート・コピー・コマンド・オプションは、コマンド・ライン・フラグまたは環境変数を使用して構成できます。ノード・ターゲットについては、リモート・コピー・コマンド・オプションは、次の優先順位で決定されます。 1. **-o** フラグ 2. **DCP_NODE_OPTS** 環境変数 デバイス・ターゲットについては、次の優先順位でリモート・コピー・コマンド・オプションが決定されます。 1. **-O** フラグ 2. **DCP_DEVICE_OPTS** 環境変数 リモート・コピー・コマンド・オプションは、次の構文を使用して指定されます。 [*context:*]"*options*"[, [*context:*]"*options*"]... ここで、*options* は、リモート・コピー・コマンドのオプションであり、*context:* はファイルをコピーするのに使用されるリモート・シェル・オプションのコンテキストを示します。コンテキストを指定せずにオプションを指定すると、そのオプションは、リストで明示的なオプションが指定されていないその他すべてのコンテキストに適用されます。オプションは、**dcp** オプションと区別するために二重引用符 (**""**) で囲んで指定する必要があります。

項目	説明
コマンドの実行	**-f** フラグまたは **DSH_FANOUT** 環境変数を使用して指定できる並行リモート・コピー・コマンド・プロセス (ファンアウト) を指定します。ファンアウトは、並行実行できるリモート・シェル・コマンドの数によってのみ制限されます。管理サーバーで **DSH_FANOUT** 値を使用して試し、さらに高い値が適切であるかどうかを調べることができます。リモート・コピー・コマンド実行のタイムアウト値は、**-t** フラグまたは **DSH_TIMEOUT** 環境変数を使用して指定できます。いずれかのリモート・ターゲットがタイムアウト値の時間内に応答しない場合、**dcp** コマンドはエラー・メッセージを表示して終了します。**-T** フラグは、**dcp** コマンド実行に関する診断トレース情報を提供します。デフォルトの設定値とリモート・ターゲットに対して実行された実際のリモート・コピー・コマンドが表示されます。**dcp** コマンドは、**-Q** フラグを使用してサイレント実行することができます。その場合、ターゲットの標準出力または標準エラーは表示されません。

この変数のパラメーターを以下に示します。

source_file...

ターゲットとの間でコピーされるファイルの完全パスを指定します。複数のファイルを指定できます。**-R** フラグと同時に使用する場合は、単一のディレクトリーのみを指定できます。**-P** フラグと同時に使用する場合は、単一のファイルのみを指定できます。

target_path

1 つ以上の *source_file* ファイルのコピー先となるターゲット上の完全パスを指定します。 **-P** フラグが指定されている場合、*target_path* は、コピーされるファイルのローカル・ホスト上のロケーションです。リモート・ファイル・ディレクトリー構造が *target_path* の下に再作成され、*target_path* ディレクトリー内のコピーされた *source_file* 名にリモート・ターゲット名が付加されます。

キーワード

項目	説明
-a	デフォルトのコンテキストで定義されたすべてのノードをターゲット・リストに組み込みます。デフォルトのコンテキストは、**-C** フラグまたは **DSH_CONTEXT** 環境変数を使用して設定できます。
-A	デフォルトのコンテキストで定義されたすべてのデバイスをターゲット・リストに組み込みます。デフォルトのコンテキストは、**-C** フラグまたは **DSH_CONTEXT** 環境変数を使用して設定できます。このフラグは HMC では使用不可になっています。
--all-devices *context_list*	*context_list* にリストされているコンテキストで定義されたすべてのデバイスをターゲット・リストに組み込みます。デフォルトのコンテキストは、このリストに暗黙的に組み込まれていません。このフラグは HMC では使用不可になっています。
--all-nodes *context_list*	*context_list* にリストされているコンテキストで定義されたすべてのノードをターゲット・リストに組み込みます。デフォルトのコンテキストは、このリストに暗黙的に組み込まれていません。
-C	デバイス・ターゲットとの間でファイルをコピーするために使用されるリモート・コピー・コマンドの絶対パスを指定します。特定のコンテキストのリモート・コピー・コマンドは、パスの前に `context:` を含めることによって定義できます。
--context*context*	**dcp** コマンドがターゲット名を解決するときに使用するデフォルトのコンテキストを指定します。context 値は、`/opt/ibm/sysmgt/dsm/pm/Context` ディレクトリー内の有効なコンテキスト拡張モジュールに対応している必要があります。
--device-rcp *device_remote_copy*	監査サブシステムを始動します。*device_remote_copy* の構文は次のとおりです。 [*context:*]*path*[,[*context:*]*path*]... このフラグは HMC では使用不可になっています。このキーワードは構成ファイル内の命令を読み取り、デバイス・ターゲットのリモート・シェルを判別します。
-d | **--devices** *device_list*	ターゲット・リストに組み込むデバイス・ターゲットのリストを指定します。*device_list* の構文は次のとおりです。 [*context:*] [*user_ID@*] *device_name*[,¥ [*context:*][*user_ID@*]*device_name*]... このフラグは HMC では使用不可になっています。
-D | **--devicegroups** *devicegroups*	*devicegroups* リストで指定されているデバイス・グループで定義されたすべてのデバイスをターゲット・リストに組み込みます。 *devicegroups* の構文は次のとおりです。 [*context:*] [*user_ID@*]*devicegroup*[,¥ [*context:*] [*user_ID@*]*devicegroup*]... このフラグは HMC では使用不可になっています。
-f | **--fanout** *fanout*	並行実行中のリモート・シェル・プロセスの最大数に対するファンアウト値を指定します。ファンアウト値 1 を指定することによって、順次実行を指定できます。このフラグを省略した場合、デフォルトのファンアウト値 64 が使用されます。

項目	説明
-l | **--user** *user_ID*	リモート・コピー実行に使用するリモート・ユーザー名を指定します。
-h | **--help**	コマンドの使用方法に関する情報を表示します。
-n | **--nodes** *node_list*	ターゲット・リストに組み込むノード・ターゲットのリストを指定します。*node_list* の構文は次のとおりです。 [*context:*] [*user_ID@*]*node_name*[,¥ [*context:*] [*user_ID@*]*node_name*]...
-o | **--node-options** *node_options*	ノード・ターゲットに対するリモート・コピー・コマンドを受け渡すオプションを指定します。オプションは、**dcp** コマンド・フラグと区別するために二重引用符で囲んで指定する必要があります。特定のコンテキストのノードに対するオプションは、オプション・リストの前に `context:` を含めることによって定義できます。*device_options* の構文は次のとおりです。 [*context:*]"*options*"[, [*context:*]"*options*"]...
-N | **--nodegroups** *nodegroups*	*nodegroups* リストで指定されているノード・グループで定義されたすべてのノードをターゲット・リストに組み込みます。 *nodegroups* の構文は次のとおりです。 [*context:*] [*user_ID@*]*nodegroup*[,¥ [*context:*] [*user_ID@*]*nodegroup*]...
-O --device-options *device_options*	デバイス・ターゲットに対するリモート・コピー・コマンドを受け渡すオプションを指定します。オプションは、**dcp** コマンド・フラグと区別するために二重引用符で囲んで指定する必要があります。特定のコンテキストのデバイスに対するオプションは、オプション・リストの前に `context:` を含めることによって定義できます。*device_options* の構文は次のとおりです。 [*context:*]"*options*"[,[*context:*]"*options*"]... このフラグは HMC では使用不可になっています。
-p | --preserve	構成済みのリモート・コピー・コマンドによってインプリメントされたソース・ファイルの特性を保持します。
-P | --pull	ターゲットからファイルをプル (コピー) して、それらをローカル・ホスト上の *target_path* ディレクトリーに配置します。*target_path* はディレクトリーでなければなりません。リモート・マシンからプルされたファイルは、元のファイルと区別するために、ファイル名に `_target` が付加されます。**-P** フラグが **-R** フラグと同時に使用される場合、`_target` はディレクトリーに付加されます。**dcp -P | --pull** コマンドの呼び出しごとに 1 つのファイルのみを、指定されたターゲットからプルできます。
-Q	ターゲットの標準出力や標準エラーが表示されないよう、**dcp** コマンドをサイレント実行します。
-q | --show-config	すべての **dsh** ユーティリティー・コマンドに関連する現行の環境設定を表示します。このフラグには、現在インストール済みの有効なすべてのコンテキストに関するすべての環境変数および設定の値が含まれます。設定値のソース・コンテキストを示すために、それぞれの設定値の前に `context:` が付けられます。
-r | --node-rcp *node_remote_copy*	ノード・ターゲットとの間でファイルをコピーするために使用されるリモート・コピー・コマンドの絶対パスを指定します。特定のコンテキストのリモート・コピー・コマンドは、パスの前に `context:` を含めることによって定義できます。*node_remote_copy* の構文は次のとおりです。 [*context:*]*path*[,[*context:*]*path*]... パスに **rsync** が含まれる場合、**rsync** コマンドがリモート・コピーを実行することが想定されます。
-R | --recursive	ローカル・ディレクトリーからリモート・ターゲットにファイルを再帰的にコピーします。あるいは、**-P** フラグと同時に指定された場合は、リモート・ディレクトリーからローカル・ホストにファイルを再帰的にプル (コピー) します。*source_file* パラメーターを使用して単一のソース・ディレクトリーを指定できます。
-t | --timeout *timeout*	リモート・コピー・コマンドが各リモート・ターゲットで完了するのを待機する時間を秒単位で指定します。ターゲットがタイムアウト値の時間内に応答しない場合、**dcp** コマンドはエラー・メッセージを表示し、そのリモート・ターゲットに対するリモート・コピー・プロセスを終了します。指定されない場合、**dcp** コマンドは、リモート・コピー・プロセスが各ターゲットで完了するまで無制限に待機します。
-T | --trace	トレース・モードを活動化します。**dcp** コマンドの診断メッセージを標準出力に送信します。
-v | --verify	リモート・コマンドをターゲットで実行する前に、各ターゲットを検査します。ターゲットが応答しない場合、そのターゲットに対するリモート・コマンドの実行は取り消されます。
-X *env_list*	**dcp** コマンド環境変数を無視します。このオプションでは、無視してはならない環境変数名のコンマ区切りリストである引数が、受け入れられます。このオプションに対する引数がない場合、あるいは引数が空ストリングである場合、すべての **dcp** 環境変数が受け入れられません。

項目	説明
-V | --version	**dcp** コマンドの環境変数のバージョン情報を表示します。

DSH_CONTEXT
ターゲットを解決するときに使用するデフォルトのコンテキストを指定します。この変数は、**-C** フラグによって指定変更されます。

DSH_DEVICE_LIST
デバイス・ターゲットのリストを含むファイルを指定します。この変数は、**-d** フラグによって指定変更されます。この環境変数は HMC では無視されます。

DCP_DEVICE_OPTS
デバイス・ターゲットだけに、リモート・シェル・コマンドに使用されるオプションが指定されます。この変数は、**-0** フラグによって指定変更されます。この環境変数は HMC では無視されます。

DCP_DEVICE_RCP
デバイス・ターゲットとの間でファイルをコピーするために使用されるリモート・コピー・コマンドの絶対パスを指定します。この変数は、**--device-rcp** フラグによって指定変更されます。この環境変数は HMC では無視されます。

DSH_FANOUT
ファンアウト値を指定します。この変数は、**-f** フラグによって指定変更されます。

DCP_NODE_OPTS
ノード・ターゲットだけに、リモート・コピー・コマンドに使用されるオプションが指定されます。この変数は、**-o** フラグによって指定変更されます。

DCP_NODE_RCP
ノード・ターゲットとの間でファイルをコピーするために使用されるリモート・コマンドの絶対パスを指定します。この変数は、**-r** フラグによって指定変更されます。

DSH_NODE_LIST
ノード・ターゲットのリストを含むファイルを指定します。この変数は、**-n** フラグによって指定変更されます。 **WCOLL. DSH_NODEGROUP_PATH** 変数は、この変数によって置き換えられました。
DSH_NODE_LIST 変数は、**dsh** コンテキストのノード・グループ・ファイルを含むディレクトリーのコロンで区切られたリストも指定します。**dsh** コンテキストで **-a** フラグが指定されている場合、パスのすべてのノード・グループ・ファイルから固有のノード名のリストが収集されます。

DSH_TIMEOUT
各リモート・ターゲットからの出力を待機する時間を秒単位で指定します。この変数は、**-t** フラグによって指定変更されます。

RSYNC_RSH
この **rsync** 環境変数は、**rsync** コマンドのトランスポートとして使用されるリモート・シェルを指定します。リモート・コピー・コマンドの終了値がゼロ以外の場合、各リモート・コピー・コマンド実行の終了状況および終了値は、**dcp** コマンドからのメッセージに表示されます。リモート・コピー・コマンドからのゼロ以外の戻りコードは、リモート・コピー中にエラーが発生したことを示します。リモート・コピー・コマンドでエラーが発生した場合、そのターゲットに対するリモート・コピーの実行はバイパスされます。**dcp** コマンドがエラーなしに実行され、すべてのリモート・コピー・コマンドが終了コード 0 で完了した場合、**dcp** コマンドの終了コードは 0 です。**dcp** コマンドに内部エラーが発生するか、リモート・コピー・コマンドが正常に完了しない場合、**dcp** コマンドの終了値は 0 より大きくなります。終了値は、失敗したリモート・コピー・コマンド実行の一連のインスタンスごとに 1 ずつ増加します。

セキュリティー: **dcp** コマンドにセキュリティー構成要件はありません。すべてのリモート・コマンド・セキュリティー要件 (構成、認証、および許可) は、**dcp** コマンド用に構成された基礎となるリモート・コマンドによって課せられます。ローカル・ホストとリモート・ターゲットの間で認証および許可が構成されることが想定されます。対話式パスワード・プロンプトはサポートされません。パスワード・プロンプトが出された場合、リモート・ターゲットに対して実行はバイパスされ、エラーが表示されます。リモート環境およびリモート・シェル・コマンドに関連するセキュリティー構成は、ユーザー定義です。Kerberos バージョン 5 を使用するリモート・コマンドとして `/usr/bin/rcp` が構成される場合、最初に Kerberos **kinit** コマンドを実行してチケット許可チケットを取得する必要があります。また、Kerberos プリンシパルがターゲットのリモート・ユーザーのホーム・ディレクトリーの `.k5login` ファイル内にあることを確認する必要があります。

例

1. `/tmp/etc/hosts` ファイルをローカル・ホストから `node3`、`node4`、`node5` の `/etc` ディレクトリー、および NIM コンテキスト内の `device16` のユーザー `gregb` にコピーするには、次のコマンドを入力します。

```
dcp -n node3-node5 -d NIM:gregb@device16 /tmp/etc/hosts /etc:
```

2. /etc/hosts ファイルをクラスター内のすべての管理対象ノードからローカル・ホストの /tmp/hosts.dir ディレクトリーにコピーするには、次のコマンドを入力します。

   ```
   dcp -aP /etc/hosts /tmp/hosts.dir
   ```

   ターゲットの名前を指定するサフィックスが各ファイル名に付加されます。/tmp/hosts.dir ディレクトリーの内容は次のようになります。

hosts._node1	hosts._node4	hosts._node7
hosts._node2	hosts._node5	hosts._node8
hosts._node3	hosts._node6	

3. ファンアウトに 12 を指定して、/var/log/testlogdir ディレクトリーを、NIM コンテキストの NodeGroup1 および **dsh** コンテキストの DeviceGroup4 のすべてのターゲットからコピーし、各ディレクトリーをローカル・ホスト上に /var/log._target として保管するには、次のコマンドを入力します。

   ```
   dcp -C DSH -N NIM:NodeGroup1 -D DeviceGroup 4 -f 12 ¥ -RP /var/log/testlogdir /var/log
   ```

4. **rsync** コマンドを使用して、/localnode/smallfile および /tmp/bigfile を node1 上の /tmp にコピーするには、次のコマンドを入力します。

   ```
   RSYNC_RSH=/usr/bin/ssh; dcp -r /usr/bin/rsync -o "-z" ¥ -n node1 /localnode/smallfile /tmp/bigfile /tmp
   ```

   このコマンドは、**rsync** と、**rsync** に対する **RSYNC_RSH** 環境変数および **-z** フラグを使用します。

5. /etc/hosts ファイルをローカル・ホストからクラスター内のすべてのノードにコピーして、すべての **dcp** 環境変数を無視するには、次のコマンドを入力します。

   ```
   dcp -X -a /etc/hosts /etc/hosts
   ```

6. /etc/hosts ファイルを node1 および node2 からローカル・ホスト上の /tmp/hosts.dir ディレクトリーにコピーし、**DCP_NODE_OPTS** 以外のすべての **dcp** 環境変数を無視するには、次のコマンドを入力します。

   ```
   dcp -n node1,node2 -P -X 'DCP_NODE_OPTS' /etc/hosts /tmp/hosts.dir
   ```

関連資料:

210 ページの『dsh コマンド』

---

dd コマンド

目的

ファイルを変換してコピーします。

構文

dd [**bs=***BlockSize*][**cbs=***BlockSize*]

[**conv=** [ascii |block|ebcdic | ibm | unblock]

[lcase | ucase] [iblock]

[noerror] [swab] [sync]

[oblock] [notrunc]][**count=**

InputBlocks] [**files=***InputFiles*] [**fskip=**

SkipEOFs] [**ibs=***InputBlockSize*] [**if=**

InFile] [**obs=***OutputBlockSize*] [**of=**

OutFile] [**seek=***RecordNumber*] [**skip=**

SkipInputBlocks][**span=***yes*|*no*]

dd [*Option*=*Value*]

説明

dd コマンドは、 *InFile* パラメーターまたは標準入力を読み取って指定された変換を行い、変換したデータを *OutFile* パラメーターまたは標準出力にコピーします。入力および出力ブロック・サイズを指定して、物理的な未加工の入出力 (ロウ I/O) を使用できます。

注**:** *Block* という用語は、**dd** コマンドによって 1 つの操作で読み書きされるデータの量のことで、必ずしもディスク・ブロックと同じサイズであるとは限りません。

サイズを指定する場合は、バイト数で指定します。最後に **w**、**b**、または **k** が付いている数字は、それぞれ、2 倍、512 倍、または 1024 倍することを示します。**x** または ***** (アスタリスク) で区切った数字の対は積を表します。カウント・パラメーターは、バイト数ではなく 、コピーするブロックの数を要求します。

conv=ascii フラグおよび、**conv=ebcdic** フラグと関連付けられている、文字セットのマッピングは、相補演算です。この 2 つのフラグは、ASCII 文字と、ほとんどのワークステーションとキー・パンチに使われる EBCDIC 文字のサブセットとをマップします。

block、**unblock**、 **ascii**、**ebcdic**、または **ibm** のいずれかの変換を指定する場合は、 **cbs** パラメーター値を使用してください。**unblock** パラメーターまたは **ascii** パラメーターを指定すると、 **dd** コマンドにより固定長から可変長への変換が実行されます。それ以外の場合は、可変長から固定長への変換が実行されます。**cbs** パラメーターは固定長を決定します。

重要**:** **cbs** パラメーター値を最小入力ブロックより小さい値に指定すると、変換後のブロックが切り捨てられます。

処理の終了時に、**dd** コマンドは全体および部分的な入力ブロックと出力ブロックの数を報告します。

注**:**

1. 通常、必要なのは、出力ファイルへの書き込みアクセスだけです。しかし、出力ファイルが直接アクセス・デバイス上になく、**seek** フラグを使用する場合には、そのファイルへの読み取りアクセスも必要です。
2. **dd** コマンドは、 **conv=ascii** または **conv=unblock** フラグを設定した変換時にのみ改行文字を挿入し、**conv=ebcdic**、 **conv=ibm**、または **conv=block** フラグを設定した変換時にのみ埋め込みを行います。

3. ファイルをテープにコピーできる場合は、 **dd** コマンドではなく、できるだけ **backup**、 **tar**、または **cpio** コマンドを使用してください。これらのコマンドは、テープ・デバイスで使用するようになっています。テープ・デバイスの使用法の詳細については、**rmt** スペシャル・ファイルを参照してください。
4. **bs**、 **ibs**、および **obs** フラグで指定されるブロック・サイズの値は、必ず、使用されるメディアの物理ブロックのサイズの倍数でなければなりません。
5. **conv=sync** フラグを指定すると、 **dd** コマンドによって部分入力ブロックに NULL が埋め込まれます。したがって、いずれかの読み取りで (**ibs** フラグで指定された) データのフル・ブロックを受信しない場合、 **dd** コマンドはデータ・ストリームの途中に NULL を挿入します。パイプからの読み取りの場合、この挿入はよく発生します。
6. **bs** フラグのみを指定して、 **sync**、**noerror**、または **notrunc** 以外の変換を指定しない場合、各入力ブロックからのデータは、個別の出力ブロックとして書き込まれます。読み取りでフル・ブロックに満たないブロックが戻されて、 **sync** が指定されていない場合、結果として出力ブロックのサイズは入力ブロックと同じになります。**bs** フラグを指定しない場合、または **sync**、**noerror**、 **notrunc** 以外の変換を指定した場合は、入力の終わりに到達するまで、入力が処理され、フルサイズの出力ブロック内に収集されます。

デバイス全体のスパン

入力ファイルが出力デバイスの物理サイズより大きい場合は、**dd** を使用してデバイス全体をスパンすることができます。

注**:** 不適切なブロック・サイズはデータの不整合やオーバーラップの原因となるので、ブロック・サイズ *bs* の指定がデバイスの物理サイズの正確な倍数となるように注意する必要があります。

InFile と OutFile パラメーターのいずれかが stdin または stdout である場合は、**dd** によるデバイス全体のスパンは行われません。

出力デバイスがフルの場合には、書き込み時に、**dd** が次のデバイスにプロンプトを出すような方法でスパンが行われます。入力デバイスからの読み取り時に、デバイスの読み取りが終了していなくても、データが入力デバイスから完全に読み取られている場合は、**dd** が次のデバイスにプロンプトを出します。この場合、終了するには、'n' を押さなければならないことがあります。

フラグ

項目	説明
bs=*BlockSize*	**ibs** フラグおよび、 **obs** フラグを置き換えて、入力および出力ブロックのサイズを指定します。**bs** フラグで指定するブロック・サイズの値は必ず、使用されるメディアの物理ブロック・サイズの倍数でなければなりません。
cbs=*BlockSize*	**conv=block** のように、可変長から固定長への変換、および固定長から可変長への変換の変換ブロック・サイズを指定します。
count=*InputBlocks*	*InputBlocks* 変数で指定された入力ブロックの数だけをコピーします。

項目	説明
conv= *Conversion,...*	1 つ以上の変換オプションを指定します。複数の変換を指定する場合はコンマで区切ってください。指定できるオプションを次に示します。 **ascii** EBCDIC を ASCII に変換します。このオプションは、 **ebcdic** オプション、 **ibm** オプション、 **block** オプション、および **unblock** オプションとは、互換性がありません。 **block** 可変長レコードを固定長に変換します。長さは変換ブロック・サイズ (cbs) によって決定されます。このオプションは、**ascii** オプション、 **ebcdic** オプション、 **ibm** オプション、および **unblock** オプションとは、互換性がありません。 **ebcdic** ASCII を標準 EBCDIC に変換します。このオプションは、 **ascii** オプション、 **ibm** オプション、 **block** オプション、および **unblock** オプションとは、同時には指定できません。 **ibm** ASCII を EBCDIC の IBM® バージョンに変換します。このオプションは、 **ascii** オプション、 **ebcdic** オプション、 **block** オプション、および **unblock** オプションとは、同時には指定できません。 **iblock**、 **oblock** 直接アクセス・デバイスでの読み取りエラーまたは書き込みエラーから生じるデータの損失を最小限にとどめます。 **iblock** 変数を指定しているときに、ブロックの読み取りでエラーが発生した場合 (ブロック・サイズが 512 であるか、または、 **ibs=***InputBlockSize* 変数で指定したサイズである場合)、 **dd** コマンドは、同じデータ・ブロックを小さな単位で再読み取りしようとします。**dd** コマンドが入力デバイスのセクター・サイズを判別できる場合は、損傷を受けたブロックを一度に 1 セクターずつ読み取ります。判別できない場合は、一度に 512 バイトずつ読み取ります。入力ブロック・サイズ (**ibs**) は、この再試行のときのサイズの倍数でなければなりません。このオプションは、読み取りエラーに関連するデータの損失を 1 セクターにおさえます。 **oblock** 変換も出力に対して同じように機能します。 **lcase** 英文字をすべて小文字にします。 **noerror** エラーの発生時に処理を停止しません。 **notrunc** 出力ファイルを切り捨てません。出力への書き出しが明示的でないブロックは保存されます。 **ucase** 英文字をすべて大文字にします。 **swab** バイトの対をすべてスワップします。 **sync** すべての入力ブロックを、 **ibs** 値まで埋め込みます。 **unblock** 固定長ブロックを可変長に変換します。長さは変換ブロック・サイズ (cbs) によって決定されます。このオプションは、**ascii** オプション、 **ebcdic** オプション、 **ibm** オプション、および **block** オプションとは、同時には指定できません。
files=*InputFiles*	入力ファイルの *InputFiles* 変数の値によって指定されたファイルの数を、終了前にコピーします (入力が磁気テープ、または類似デバイスである場合にのみ有効です)。
fskip=*SkipEOFs*	コピーを開始する前に、 *SkipEOFs* 変数によって指定されたファイル終わり文字の数をスキップします。この *SkipEOFs* 変数は、マルチファイルの磁気テープ上での位置決めに役立ちます。
ibs=*InputBlockSize*	入力ブロック・サイズを指定します。デフォルトは 512 バイトまたは 1 ブロックです。 **ibs** フラグで指定するブロック・サイズの値は、必ず、使用中のメディアの物理ブロック・サイズの倍数でなければなりません。
if=*InFile*	入力ファイル名を指定します。標準入力がデフォルト値です。
obs=*OutputBlockSize*	出力ブロック・サイズを指定します。デフォルトは 512 バイトまたは 1 ブロックです。 **obs** フラグで指定するブロック・サイズの値は、必ず、使用中のメディアの物理ブロック・サイズの倍数でなければなりません。
of=*OutFile*	出力ファイル名を指定します。標準出力がデフォルト値です。

項目	説明
seek=*RecordNumber*	コピー前に、*RecordNumber* 変数によって指定されたレコードを出力ファイルの最初から探します。
skip=*SkipInputBlocks*	コピーを開始する前に、入力ブロックの指定された *SkipInputBlocks* 値をスキップします。
span=*yes* | *no*	yes が指定された場合は複数のデバイス間でのスパンを許可し、no が指定された場合はデフォルトで動作します。詳しくは、『デバイス全体のスパン』を参照してください。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	入力ファイルは正常にコピーされました。
>0	エラーが発生しました。

例

1. ASCII テキスト・ファイルを EBCDIC に変換するには、次のように入力します。

   ```
   dd  if=text.ascii  of=text.ebcdic  conv=ebcdic
   ```

   このコマンドを実行すると、 text.ascii ファイルが EBCDIC 表記に変換され、その EBCDIC バージョンが text.ebcdic ファイルに保管されます。

   注**: conv=ebcdic** パラメーターを指定すると、**dd** コマンドは ASCII の ^ (曲折アクセント記号) 文字を EBCDIC の未使用の文字 (16 進数 9A) に変換し、 ASCII の ~ (ティルド) を EBCDIC の ^ (NOT 記号) に変換します。
2. 可変長レコードである ASCII ファイル **/etc/passwd** を、 132 バイトの固定長 EBCDIC レコードのファイルに変換するには、次のように入力します。

   ```
   dd  if=/etc/passwd  cbs=132  conv=ebcdic  of=/tmp/passwd.ebcdic
   ```
3. 各レコードが 132 バイトの EBCDIC ファイルを小文字の可変長 ASCII 行に変換するには、次のように入力します。

   ```
   dd  if=/tmp/passwd.ebcdic  cbs=132  conv=ascii  of=/tmp/passwd.ascii
   ```
4. 可変長レコードの ASCII ファイル **/etc/passwd** を、IBM バージョンの EBCDIC による 132 バイトの固定長レコードのファイルに変換するには、次のように入力します。

   ```
   dd  if=/etc/passwd  cbs=132  conv=ibm  of=/tmp/passwd.ibm
   ```
5. 1KB ブロックのテープから、2KB のブロックを使用する別のテープにブロックをコピーするには、次のように入力します。

   ```
   dd  if=/dev/rmt0  ibs=1024  obs=2048  of=/dev/rmt1
   ```
6. **dd** コマンドをフィルターとして使用するには、次のように入力します。

   ```
   ls -l | dd  conv=ucase
   ```

   このコマンドにより、現行ディレクトリーの長いリストが大文字で表示されます。

   注**: dd** コマンドおよび **cpio** コマンドの 9348 磁気テープ装置モデル 12 に対するパフォーマンスは、デフォルトのブロック・サイズを変更すると向上します。ブロック・サイズを変更するには、**chdev** コマンドを次のように使用します。

```
chdev -l Device_name -a block_size=32k
```

7. 36 ブロック (512 バイト/ブロック) を使用して、 3.5 インチで 1.4MB のディスケットへ効率よく転送するには、次のように入力します。

   ```
   dd  if=Filename  of=/dev/rfd0  bs=36b  conv=sync
   ```

   このコマンドは、*Filename* パラメーターの値を一度に 1 シリンダーずつディスケット・デバイスに書き込みます。ディスクからの読み取りを行う場合と、ファイル・サイズがディスケット・ブロック・サイズの倍数でない場合は、 `conv=sync` が必要です。**dd** コマンドへの入力がファイルではなく、パイプである場合は、これを行わないでください。そうすると、ほとんどの入力で、最後のブロックの代わりに NULL が埋め込まれます。
8. ブロック・サイズが 720b ブロックに設定されている入力ファイルから、ブロックを 1.44MB サイズのディスケットにコピーするには、次のように入力します。

   ```
   dd if=testfile of=/dev/fd0 bs=720b conv=sync
   ```

   注**:** 入力ファイルが出力デバイスの物理サイズより大きい場合は、dd は、別のデバイスが必要であるというプロンプトを出します。
9. ブロック・サイズが 32k ブロックに設定されている入力ファイルから、ブロックをテープにコピーするには、次のように入力します。

   ```
   dd if=inputfile of=/dev/rmt0 bs=32k conv=sync
   ```
10. ブロック・サイズが 32k ブロックに設定されている現行ディレクトリーのファイルに、データのブロックをテープからコピーするには、次のように入力します。

    ```
    dd if=/dev/rmt0 of=outfile bs=32k conv=sync
    ```
11. ブロック・サイズが 720b ブロックに設定されている入力ファイルから、ブロックを 1.44MB サイズのディスケットにコピーするには、次のように入力します。

    ```
    dd if=testfile of=/dev/fd0 bs=720b conv=sync span=yes
    ```

    注**:** 入力ファイルが出力デバイスの物理サイズより大きい場合は、**dd** は別のデバイスが必要であるというプロンプトを出します。
12. ブロック・サイズが 32k ブロックに設定されている入力ファイルから、ブロックをテープにコピーするには、次のように入力します。

    ```
    dd if=inputfile of=/dev/rmt0 bs=32k conv=sync span=yes
    ```
13. ブロック・サイズが 32k に設定されているテープから、データのブロックを現行ディレクトリー内のファイルにコピーするには、次のように入力します。

    ```
    dd if=dev/rmt0 of=outfile bs=32k conv=sync span=yes
    ```

ファイル

項目	説明
/usr/bin/dd	**dd** コマンドが入っています。

関連情報:

cp コマンド

tar コマンド

システム・バックアップ

オペレーティング・システムのファイル

---

defif メソッド

目的

構成データベース内にネットワーク・インターフェースを定義します。

構文

defif [**-c** *Class* **-s** *Subclass*] **-t** *Type*

説明

defif メソッドは、指定されたインスタンスのネットワーク・インターフェースを定義します。現在、構成されているアダプターのインターフェースだけを定義します。指定されたインスタンスを定義するために、**defif** メソッドは以下のことを実行します。

1. カスタマイズされたインターフェース・インスタンスを、構成データベースに作成します。
2. インターフェース・インスタンスの論理名を取り出します。
3. 事前定義属性を検索します。
4. 定義されているインターフェース・インスタンスの依存関係を反映するように、カスタマイズ済み依存関係オブジェクト・クラスを更新します。
5. インターフェース・インスタンスの状況フラグを **defined** に設定します。

フラグ

項目	説明
-c *Class*	定義するインターフェース・クラスを指定します。有効な値は **if** です。
-s *Subclass*	定義するインターフェースのサブクラスを指定します。有効な値は次のとおりです。
	TR トークンリング
	EN イーサネット
	SL SLIP
	XT X.25
	LO ループバック

項目	説明	
-t *Type*	定義するインターフェースのタイプを指定します。有効な値は次のとおりです。	
	tr	トークンリング
	en	イーサネット
	sl	SLIP
	ie3	IEEE 802.3 イーサネット
	lo	ループバック
	xt	X.25

例

トークンリング・ネットワークのインターフェース・インスタンスを定義するには、次のフォーマットでメソッドを入力します。

```
defif -t tr
```

関連情報:

odm_run_method サブルーチン

TCP/IP ネットワーク・インターフェース

オブジェクト・データ・マネージャー

Writing a Device Method

---

definet メソッド

目的

システム構成データベース内に inet インスタンスを定義します。

構文

definet [**-c** *Class*]

説明

definet メソッドは、inet インスタンスのカスタマイズされた属性を指定して、ODM 構成データベース内にオブジェクトを作成します。このコマンドは次の操作を実行します。

1. カスタマイズされた inet インスタンスを作成する。
2. inet インスタンスの状況フラグを defined に設定する。

このメソッドは、高水準コマンド **mkdev** によって呼び出されるもので、コマンド・ラインから実行するためのものではありません。

注**: definet** メソッドは、プログラミング・ツールなので、コマンド・ラインからは実行しないでください。

フラグ

項目	説明
-c *Class*	定義する inet インスタンスを指定します。*Class* 変数の有効な値は **tcpip** だけです。

例

inet0 インスタンスを定義するには、次のメソッドを入力します。

```
definet
```

関連情報:

mkdev コマンド

odm_run_method サブルーチン

オブジェクト・データ・マネージャー

Writing a Device Method

---

defragfs コマンド

目的

ファイルシステムの連続フリー・スペースを増やします。

構文

| **defragfs** [**-q** | **-r** | **-s**] [**-f** [**-v**] [**-y**]] { *Device* | *FileSystem* }

説明

defragfs コマンドは、ファイルの割り当てをディスク上で分散させずに連続的になるように再編成して、ファイルシステムの連続的なフリー・スペースを増やします。 *Device* 変数を指定して、デフラグを行うファイルシステムを指定することができます。この変数は、論理ボリュームのパス名 (例えば **/dev/hd4** など) です。また、*FileSystem* 変数を使用して指定することもできます。この変数は **/etc/filesystems** ファイル内のマウント・ポイントです。

defragfs コマンドは、フラグメント化され圧縮されたファイルシステムを対象にしています。ただし、**defragfs** コマンドを使用することにより、フラグメント化されていないファイルシステム内で連続的なフリー・スペースを増やすこともできます。

このコマンドを正常に実行するには、ファイルシステムを読み取り/書き込みでマウントしなければなりません。 **-q** フラグ、**-r** フラグ、または **-s** フラグを使用すると、フラグメント化レポートが生成されます。これらのフラグを指定しても、ファイルシステムは変更されません。

defragfs コマンドは、スナップショット・ストレージ・オブジェクトにコピーしなければならないデータ量が原因で、スナップショットを使用する拡張ジャーナル・ファイルシステム (JFS2) ファイルシステムに対しては遅くなります。 **defragfs** コマンドは、スナップショットがあると警告メッセージを出します。**snapshot** コマンドは、スナップショットの削除に使用でき、その後 **defragfs** コマンドの完了後に新しいスナップショットの作成に再度使用することができます。

| JFS2 ファイルシステムでは、**defragfs** コマンドで **-f** フラグを指定して、データ・エクステントが隣接す
| るように再配置することにより、ファイルシステムをフラグメント解消できます。さらに、**-v** フラグを指

定すると、**defragfs** コマンドは、**defragfs** コマンド実行の前後でファイルシステムのフラグメントを表示します。 **-f**、**-v**、および **-y** の各フラグは、JFS2 ファイルシステムでのみ使用できます。 **-v** フラグは **-f** フラグとのみ併用できます。

-f フラグを指定した場合、**defragfs** コマンドの実行にはより多くの時間がかかります。 **defragfs** コマンドは保守の時間帯に実行することをお勧めします。

ファイルシステム・アクティビティーはいずれも、フラグメント解消プロセスのパフォーマンスを低下させるおそれがあります。

論理ボリュームの一部または全体がソリッド・ステート・ドライブ (SSD) 上にあるようなファイルシステムのパフォーマンスは、**defragfs** コマンドによって大幅に改善されない場合があります。

内部スナップショットがシステムに存在する場合、**defragfs** コマンドは実行できません。外部スナップショットがシステムに存在する場合、**defragfs** コマンドを **-f** フラグ指定で実行しない限り、**defragfs** コマンドは警告メッセージを出します。 **defragfs** コマンドを **-f** フラグ指定で実行する場合、**defragfs** コマンドを外部スナップショットを使用して実行することはできません。 **defragfs** コマンドは、スナップショット・ストレージ・オブジェクトにコピーしなければならないデータ量が原因で、スナップショットを使用する JFS2 ファイルシステムでは実行に時間がかかります。 **snapshot** コマンドを使用してスナップショットを削除し、**defragfs** コマンドの完了後に再度 **snapshot** コマンドを使用して新しいスナップショットを作成することができます。

defragfs コマンドは、他のファイルシステムとログ・ボリュームを共有しないファイルシステム上で実行すると、パフォーマンスが向上します。 **defragfs** コマンドを他のファイルシステムとログ・ボリュームを共有するファイルシステムで実行すると、**defragfs** コマンドは警告を表示し、確認を要求します。 **defragfs** コマンドを **-y** フラグ指定で実行すると、警告は抑止されます。 **-y** フラグは **-f** フラグとのみ併用できます。

フラグ

項目	説明
-f	ファイルシステム内の各ファイルのデータ・エクステントを再配置および結合します。このプロセスは、ファイルシステム・フリー・スペースの隣接性よりもファイル編成を優先します。
-q	ファイルシステムの現在の状態を報告します。
-r	ファイルシステムの現在の状態と、**-q**、**-r**、または **-s** フラグを指定せずに **defragfs** コマンドを実行した場合の状態を報告します。
-s	ファイルシステム内のフラグメント化を報告します。このオプションを使用すると、 **defragfs** は、パフォーマンスを低下させる可能性があるファイルシステム内のメタデータをパススルーします。
-v	フラグメント解消操作の開始時と終了時に、ファイルシステムのフラグメント化率 (%) を表示します。
-y	複数のファイルシステムが現在、同じログ・ボリュームを使用してマウントされている場合に、**defragfs** コマンドによって表示される警告メッセージを抑止します。警告メッセージが抑止されると、**defragfs** コマンド操作は中断なしに続行されます。

注**: -v** フラグと **-y** フラグは **-f** フラグとのみ併用できます。

出力

JFS ファイルシステムでは、**defragfs** コマンドによって報告されるメッセージの定義は以下のようになります。

Number of free fragments
ファイルシステム内の、空きフラグメントの数。

Number of allocated fragments
ファイルシステム内の、割り当てられたフラグメントの数。

Number of free spaces shorter than a block
ファイルシステム内の、1 ブロックより短いフリー・スペースの数。 1 フリー・スペースは、割り当てられていない連続したフラグメントの 1 セットを指します。

Number of free fragments in short free spaces
すべての短いフリー・スペース内の、フラグメントの合計数。1 つの短いフリー・スペースは、 1 ブロックより短いものを指します。

Number of fragments moved
移動したフラグメントの合計数。

Number of logical blocks moved
移動した論理ブロックの合計数。

Number of allocation attempts
フリー・フラグメントが再割り当てされた回数。

Number of exact matches
移動されるフラグメントがいくつかのフリー・スペースに正確に適合する回数。

Total number of fragments
ファイルシステム内の、フラグメントの合計数。

Number of fragments that may be migrated
デフラグ時に移動されるフラグメントの数。

File system is in percent fragmented
ファイルシステムがフラグメント化されている割合を、パーセンテージ (%) で表示します。

JFS2 ファイルシステムでは、**defragfs** コマンドによって報告されるメッセージの定義は以下のようになります。

Total allocation groups
ファイルシステム内の、割り振りグループの数。割り振りグループは、ファイルシステム上のスペースをチャンク (大きい塊) に分割します。割り振りグループでは、入出力パフォーマンスの良いメソッドとして知られている JFS2 リソース割り当てポリシーを使用できます。

Allocation groups defragmented
デフラグされた割り振りグループの数。

Allocation groups skipped - entirely free
完全にフリーであるためスキップされた割り振りグループの数。

Allocation groups skipped - too few free blocks
割り当てブロック内に再割り当てのためのフリー・ブロックがほとんどないため、スキップされた割り振りグループの数。

Allocation groups skipped - contains a large contiguous free space
デフラグの必要が無い、大きな連続したフリー・スペースを含むためにスキップされた割り振りグループの数。

Allocation groups are candidates for defragmenting
デフラグに適する割り振りグループの数。

Average number of free runs in candidate allocation groups
デフラグに適するとされる割り振りグループを対象とする、 1 割り振りグループごとのフリー実行 (free run) の平均回数。 1 フリー実行は、割り当てられていない、連続したブロックの 1 セットを指します。

Total number of blocks
ファイルシステム内の、ブロックの合計数。

Number of blocks that may be migrated
デフラグ時に移動されるブロックの数。

File system is in percent fragmented
ファイルシステムがフラグメント化されている割合を、パーセンテージ (%) で表示します。

Percentage of fragmentation in the file system: *percentage*
defragfs コマンド実行前後でのファイルシステムのフラグメント化率 (%)。次の例は、ファイルシステムのフラグメント化率 (%) を示しています。

```
# defragfs -fv /exampleFS
File fragmentation before defrag: 100.00%
File fragmentation after defrag: 0.00%
```

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. **/dev/lv00** 論理ボリュームに位置するファイルシステム **/data1** をデフラグするには、次のように入力します。

   ```
   defragfs /dev/lv00
   ```

2. マウント・ポイントを指定して、ファイルシステム **/data1** をデフラグするには、次のように入力します。

   ```
   defragfs /data1
   ```

3. ファイルシステム **/data1** について、現在の状態とデフラグ後の状態を示すレポートを生成するには、次のように入力します。

   ```
   defragfs  -r /data1
   ```

4. ファイルシステム **/data1** 内のフラグメント化についてのレポートを生成するには、次のように入力します。

   ```
   defragfs -s /data1
   ```

ファイル

項目	説明
/etc/filesystems	既知のファイルシステムをリストし、その特性を定義します。

関連情報:

crfs コマンド

lsfs コマンド

JFS のデータ圧縮

i ノードの可変数

defvsd コマンド

目的

仮想共用ディスクを持っているかのように、または使用しているかのようにノードを指定します。

構文

defvsd *logical_volume_name global_group_name vsd_name*

説明

このコマンドは、仮想共用ディスクとして使用するグローバルにアクセス可能なボリューム・グループ上に常駐する論理ボリュームを指定するために実行します。

System Management Interface Tool (SMIT) を使用して、**defvsd** コマンドを実行できます。 SMIT を使用するには、次のように入力します。

```
smit vsd_data
```

さらに、**Define a Virtual Shared Disk** オプションを選択します。

フラグ

-r コマンドが実行されているノードで指定された、ノード用の予想される発信シーケンス番号をリセットします。別ノードがリブート、キャストアウト、またはそのノードですべての仮想共用ディスクが再構成された場合に、このフラグを使用します。指定されたノードは、キャストインもされます。

注**:** このオプションは、 IBM サービス部門の直接の指図の下でのみ使用してください。このオプションは、通常の環境では使用しないでください。

-R コマンドが実行されているノード上の全ノードに対して、進行中および予想される発信シーケンス番号をリセットします。ノードのリブート後に、このフラグを使用します。仮想共用ディスク・ネットワーク内のすべてのノードは、キャストインされます。

注**:** このオプションは、 IBM サービス部門の直接の指図の下でのみ使用してください。このオプションは、通常の環境では使用しないでください。

-p 仮想共用ディスク並列処理のレベルを設定して、指定された数値にします。有効な範囲は、1 から 9 です。デフォルトは 9 です。値が大きければ、大規模要求に対する応答時間が短縮される可能性があります。 (仮想共用ディスクのパフォーマンス・チューニングの詳細は、「*RSCT for AIX 5L™: Managing Shared Disks*」を参照してください。)

この値は、仮想共有ディスク IP デバイス・ドライバーがカーネルで行う **uphysio** 呼び出し上の *buf_cnt* パラメーターです。コマンドが実行されているノード上の現行値を表示するには、**statvsd** を使用します。

-k ローカル・ノードで指定されたノード番号をキャストアウトします。ローカル・ノードは、キャストアウト・ノードからの要求を無視します。**-r** を使用して、ノードをキャストインして戻します。

注**:**

1. このフラグを使用する前に、後述の『制約事項』のセクションを参照してください。
2. このオプションは、 IBM サービス部門の直接の指図の下でのみ使用してください。このオプションは、通常の環境では使用しないでください。

-t 仮想共用ディスク・ドライバーによりキャッシュされた現行ルーティング・テーブルと mbuf ヘッダーをリストします。

-T すべてのキャッシュされた経路をクリアまたは解放します。

-v *vsd_name* **...**
指定された仮想共用ディスク上の読み取りおよび書き込み要求数の統計情報をリセットします。

-V すべての構成済み仮想共用ディスク上の読み取りおよび書き込み要求数の統計情報をリセットします。

-C **statvsd** コマンドで表示される仮想共用ディスク・デバイス・ドライバーのカウンターをリセットします。ただし例外は、クライアントとサーバーのノード間での、進行中および予想される発信要求シーケンス番号です。

-K ローカル・ノード上のすべてのノードをキャストアウトします。ローカル要求は引き続き受け入れられます。

注**:**

1. このフラグを使用する前に、後述の『制約事項』のセクションを参照してください。
2. このオプションは、 IBM サービス部門の直接の指図の下でのみ使用してください。このオプションは、通常の環境では使用しないでください。

-M 仮想共用ディスクの最大 IP メッセージ・サイズを設定します。これは、ある入出力要求に対して仮想共用ディスクがネットワーク上で送信するデータの最大サイズのブロックです。この制限は、ローカルの仮想共用ディスクの入出力ブロック・サイズにも影響を与えます。この値はバイト単位であり、ネットワークの最大伝送単位 (MTU) サイズより大きくしないでください。すべてのノードは同じ値を使用しなければなりません。推奨値は次のとおりです。

- 61440 (60KB) (スイッチの場合)
- 8192 (8KB) (ジャンボ・フレームのイーサネットの場合)
- 1024 (1KB) (1500 バイトの MTU イーサネットの場合)

パラメーター

logical_volume_name
仮想共用ディスクとして指定したい論理ボリュームの名前。この論理ボリュームは、指示されたグローバル・ボリューム・グループ上に常駐している必要があります。名前の長さは 15 文字以内でなければなりません。

global_group_name
vsdvg コマンドにより前回定義された、仮想共用ディスクとして指定したい、グローバルにアクセス可能なボリューム・グループ・グループの名前。名前の長さは 31 文字以内でなければなりません。

vsd_name
新規仮想共用ディスクに対して固有の名前を指定します。この名前は、RSCT ピア・ドメイン内で固有でなければなりません。また、今後の命名において競合の危険性を避けるために、クラスター全体に渡っても固有でなければなりません。推奨する命名規則は **vsdnn***gvg_name* です。名前の長さは 31 文字以内でなければなりません。

注**:** 他のデバイスで既に *vsd_name* が指定されている場合、**cfgvsd** コマンドは、仮想共用ディスクに対して失敗します。このエラーが起こるということは、その名前用に作成した特別の装置ファイルが、論理ボリュームのような他のデバイス・タイプを表している同じ名前のファイルをオーバーレイしたり、破棄したりしないことを裏付けています。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

制限

このコマンドはピア・ドメイン内のオンラインのノードから発行する必要があります。オンラインのピア・ドメインに移動するには **startrpdomain** コマンドを使用します。既存のピア・ドメイン内で、オンラインの特定ノードに移動するには **startrpnode** コマンドを使用します。RSCT ピア・ドメインの作成と管理の詳細については、「*RSCT* 管理ガイド」を参照してください。

例

1. 次の例は、グローバルにアクセス可能なボリューム・グループ (**vg1n1**) 上で、**lv1vg1n1** と呼ばれている論理ボリュームが、**vsd1vg1n1** という名前の仮想共用ディスクとして使用されることを示しています。

   ```
   defvsd lv1vg1n1 vg1n1 vsd1vg1n1
   ```

位置

/opt/rsct/vsd/bin/defvsd

関連情報:

vsdatalst コマンド

vsdvg コマンド

undefvsd コマンド

---

deleteX11input コマンド

目的

ODM (オブジェクト・データ管理) データベースから X11 入力拡張レコードを削除します。

構文

deleteX11input *DeviceName ...*

説明

deleteX11input コマンドは、 ODM データベースから X11 入力拡張レコードの削除を行うときに使用されます。指定されたそれぞれの *DeviceName* について、 ODM データベースはできるだけ数多くのオブジェクトのインスタンスを見つけ出します。このコマンドは、見つけ出されたそれぞれのデバイスを削除するかどうかを確認するようユーザーに照会します。部分的な名前を指定してもかまいません。

このコマンドは、root またはシステム・ユーザー・コマンドです。許可されていないユーザーがレコードを削除しようとすると、コマンドのアクションは失敗し、許可エラーが出されます。

パラメーター

項目	説明
DeviceName	X11 入力拡張デバイスの名前を指定します。

エラー・コード

項目	説明
No DeviceName is found in ODM Database	指定されたパターンと一致するオブジェクトが ODM データベース内にありません。
Usage: deleteX11input DeviceName	ユーザーがデバイス名を指定していません。

関連情報:

addX11input コマンド

listX11input コマンド

---

delta コマンド

目的

SCCS ファイルにデルタを作成します。

構文

delta [**-r** *SID*] [**-s**] [**-n**] [**-g** *List*] [**-p**] [**-m** *ModificationRequestList*] [**-y** [*Comment*]] *File ...*

説明

delta コマンドは、**get -e** コマンドによって取得されたファイル・バージョンへの変更を、指定したソース・コード制御システム (SCCS) ファイルにインストールします。

delta コマンドは、指定されたファイル *s* に対応する g ファイルを読み取って (SCCS が作成し、使用するファイルについては、 **get** コマンドのセクションを参照)、新しいデルタを作成します。g ファイルの行はすべて 512 字以下になります。

File 値にディレクトリーを指定すると、**delta** コマンドはディレクトリー内にあって、以前編集のためにチェックされたすべての SCCS ファイル (つまり、プレフィックス **s.** を持つすべてのファイル) に対して、要求されたアクションを実行します。 *File* 値に **-** (負符号) を指定すると、**delta** コマンドは標準入力を読み取って、個々の行を SCCS ファイルの名前として解釈します。 **delta** コマンドが標準入力を読み取る際には、**-y** フラグを指定しなければなりません。**v** ヘッダー・フラグが設定されている場合は、 **-m** フラグも指定しなければなりません。 **delta** コマンドは、ファイル終わり文字に達するまで、標準入力を読み取ります。

注**:** SOH ASCII 文字 (バイナリーの 001) で始まる行は、 **¥** (円記号) を使って、SOH を囲まない限り、SCCS ファイルに入れることはできません。SOH は SCCS にとって特別な意味を持っているので、エラーとなります。

get コマンドが大量のデータを生成しているときは、SCCS ファイルに対して **get** コマンドを使用した後、続けて同じファイルに **delta** コマンドを使用することは避けてください。その代わりに、**get** コマンドと **delta** コマンドを交互に使用してください。

delta コマンドは、SCCS ファイルの特定のバージョンについて行われた変更を保存します。 **delta** コマンドを使用するには、次のようにします。

1. **get -e** コマンドを使用して、編集可能バージョンのファイルを取り込みます。
2. そのファイルを編集します。
3. **delta** コマンドを使用して、 SCCS ファイルの新バージョンを作成します。

delta コマンドは、**-y** オプションが指定されていないと、コメントの入力を求めるプロンプトを表示します。コメントはその特定のデルタに適用され、SCCS ファイル・ヘッダーに記述されます。 **get** コマンドを使用してデルタを取り込んだときは、コメントは取得されず、取得したファイルのテキスト内に記述されません。コメントを使用して、デルタが作成された理由をトラックします。

コメントを見るには、エディターを使用して SCCS ファイルを調べ、**cat** コマンドで SCCS ファイルを表示画面に出力するか、あるいはファイルの選択した部分を **prs** コマンドを使用して標準出力に出力します。SCCS ファイルの内容を直接変更してはならないことに注意してください。デルタ・コメントを変更するには、**cdc** コマンドを使用します。

注**:** ファイルに拡張識別キーワードが含まれている場合は、そのファイルに **delta** コマンドを使用しないでください。読み取り専用ファイル・バージョンは、キーワードをテキスト値に置き換えます。読み取り専用ファイルに **delta** コマンドを使用すると、キーワードが失われます。この状況から回復するには、デルタを除去するか、あるいは再度ファイルを編集して、識別キーワードを変更してください。

ファイルの編集可能コピーが存在しない限り、SCCS は **delta** コマンドの使用を許可しません。

キーワードが失われないようにするには、**-f** フラグを指定して **admin** コマンドを使用し、**i** ヘッダー・フラグを指定します。このようにすれば、ファイル・バージョンにキーワードがないと、エラーとなります。

フラグ

項目	説明
-g *List*	**get** コマンドが g ファイルを作成するときに無視する SID (デルタ) のリストを指定します。このフラグを使用した後は、**get** コマンドは g ファイルを作成する際に指定されたデルタを無視します。
-m *ModificationRequestList*	SCCS ファイルに **v** ヘッダー・フラグ・セットが設定されている場合は、新しいデルタを作成する理由として、変更要求 (MR) 番号を入力しなければなりません。 **-m** フラグを指定しないで、**v** ヘッダー・フラグが設定されると、 **delta** コマンドは標準入力から MR を読み取ります。標準入力がワークステーションの場合は、 **delta** コマンドは MR の入力を求めるプロンプトを表示します。ファイルの終わりに達するまで、**delta** コマンドは入力を取り込み続けます。このコマンドは、コメントの前の MR は必ず読み取ります (**-y** フラグを参照)。リスト内の MR を区切るのに、ブランク、タブ文字、またはその両方を使用できます。 **v** ヘッダー・フラグに値がある場合、それは MR 番号の妥当性を検査するプログラムの名前と解釈されます。 **delta** コマンドが、MR 検証プログラムから非ゼロ終了値を戻す場合は、 **delta** コマンドは、MR 番号の一部が無効だったと想定して、実行を停止します。
-n	**delta** コマンド処理の完了時に通常は除去される g ファイルを保存します。
-p	デルタが適用される前後の SCCS ファイルの相違を (**diff** コマンドのフォーマットで) 標準出力に書き出します。フォーマットについての説明は、 **diff** コマンドのセクションを参照してください。

項目	説明
-r *SID*	SCCS ファイルに作成するデルタを指定します。このフラグを使用できるのは、同一人物によって同一の SCCS ファイルについて複数の未処理の **get -e** コマンドが実行された場合だけです。 *SID* 値は、**get** コマンド・ラインで指定された SID か、作成される SID (**get** コマンドによって報告される) のどちらかです。指定された SID の識別が一意的にできない場合、あるいは SID を指定されなければならないのに指定されていない場合には、エラーとなります。
-s	**delta** コマンドの正常終了時に、標準出力に通常書き出される情報を抑止します。
-y[*Comment*]	デルタを作成する理由を記述しているテキストを指定します。 null 文字列は有効な *Comment* 値と見なされます。コメント行に特殊文字やブランクが含まれている場合は、その行を単一引用符または二重引用符で囲まなければなりません。 **-y** フラグを指定しないと、**delta** コマンドはブランク行まで、またはファイルの終わりまで、標準入力からのコメントを読み取ります。 キーボード入力の場合は、**delta** コマンドはコメントの入力を求めるプロンプトを表示します。行の最後の文字が ¥ (円記号) の場合は、その行は無視されます。コメントの長さは、512 文字以内でなければなりません。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. SCCS ファイルに対して行った変更を記録するには、次のように入力します。

   ```
   delta s.prog.c
   ```

   これによって、SCCS ファイル `s.prog.c` にデルタが追加され、`prog.c` を編集して行った変更を記録します。次に、`delta` プログラムは、行った変更を要約したコメントを求めてきます。コメントを入力して、次にファイルの終わり文字を入力するか、またはリターン・キーを 2 回押して、コメントの入力が完了したことを示します。
2. SCCS ファイルについて行った変更を、簡潔な説明コメントを付けて記録するには、次のように入力します。

   ```
   delta -y "This delta contains the payroll function" s.prog.c
   ```

ファイル

項目	説明
/usr/bin/delta	**delta** コマンドが入っています。

関連資料:

152 ページの『diff コマンド』

関連情報:

cat コマンド

cdc コマンド

prs コマンド

ソース・コード制御システム (SCCS) の概要

---

deroff コマンド

目的

nroff、**troff**、**tbl** および **eqn** の各コマンド構成をファイルから除去します。

構文

deroff { **-ma -me -ms** [**-mm** [**-ml**]] } [**-i** | **-l**] [**-k**] [**-p**] [**-u**] [**-w**] [*File ...*]

説明

deroff コマンドは、英語テキストが入った指定のファイル (デフォルトでは標準入力) を読み取り、**troff** 要求、マクロ・コール、円記号の構成、**eqn** 構成 (**.EQ** 行と **.EN** 行の間および区切り文字と区切り文の間)、および **tbl** コマンド記述をすべて除去して、ファイルの残りの部分を書き出します。

deroff コマンドは通常、組み込まれているファイルのチェーンをたどります (**.so** および **.nx troff** コマンド要求)。ファイルが既に組み込まれている場合は、そのファイルを指定する **.so** 要求は無視されて、そのファイルを指定する **been .nx** 要求の実行が停止されます。

注**: deroff** コマンドは、完全な **troff** コマンド・インタープリターではないので、微妙な構成によって問題が起こる可能性があります。ほとんどのエラーでは、出力が少ないというより、多すぎる結果となります。

パラメーター

項目	説明
File	**deroff** コマンドが、**troff** コマンド、**eqn** コマンド、および **tbl** コマンドの処理の効力を除去するための英語のテキスト・ファイルを指定します。デフォルト・ファイルは標準入力です。

フラグ

項目	説明
-ma	テキスト内の **MA** (**man**) マクロを無視して、実行中のテキストだけが出力されるようにします。
-me	テキスト内の **ME** マクロを無視して、実行中のテキストだけが出力されるようにします。これはデフォルトです。
-ml	テキスト内の **MM** マクロ (**-mm** フラグ) を無視し、**MM** リスト構造を削除します。 -ml フラグを指定する場合は、**-mm** フラグも指定しなければなりません。 注**: -ml** フラグはネストしたリストでは使用しないでください。
-mm	**MM** マクロを無視します。
-ms	テキスト内の **MS** マクロを無視して、実行中のテキストだけが出力されるようにします。
-i	組み込まれているファイルの処理を抑止します。
-l	**/usr/lib** で始まる名前を持つ組み込まれているファイル (例えば、**/usr/lib/tmac** のマクロ・ファイル) の処理を抑止します。
-k	一緒に保持するよう指定されたブロックを保存します。デフォルトでは、テキストの取り込まれたブロックが除去されます。例えば、**.ne** 構成は除去されます。
-p	特殊なパラグラフを処理します。
-u	ASCII の下線および太字の制御シーケンスを除去します。このフラグは、**-w** フラグを自動的に設定します。
-w	出力をワード・リストにします。このとき 1 行あたり 1 ワードとし、他の文字はすべて削除します。このフラグが指定されなければ、出力はオリジナルに従います。 テキスト内では、ワードは文字で始まり 2 文字以上の文字が含まれる、文字、数字、アンパーサンド (&) およびアポストロフィ (') からなる任意の文字列です。しかし、マクロ・コールでは、ワードは 2 文字以上の文字で始まり、最低 3 文字以上を含む文字列です。区切り文字は文字、数字、句読点、アポストロフィ、アンパーサンド以外の任意の文字です。末尾のアポストロフィおよびアンパーサンドは、ワードから除去されます。

関連資料:

418 ページの『eqn コマンド』

関連情報:

nroff コマンド

tbl コマンド

troff コマンド

---

detachrset コマンド

目的

rset をプロセスから切り離します。

構文

detachrset [**-P**] *pid*

説明

detachrset コマンドは、rset をプロセスから切り離します。 rset をプロセスから切り離すと、プロセスがシステム内の任意のプロセッサーまたはメモリー領域 (あるいはその両方) を使用できるようになります。

フラグ

項目	説明
-P	区画 rset を指定されたプロセスから切り離します (*pid*)。

パラメーター

項目	説明
pid	プロセス ID

セキュリティー

ユーザーは root 権限を持っているかまたは CAP_NUMA_ATTACH 機能を使用できる必要があり、ターゲット・プロセスはコマンドの発行者と同じ有効な **userid** を持っている必要があります。プロセスから区画 rset を削除するには (**-P** オプション)、ユーザーは root 権限を持っている必要があります。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

プロセス 21414 から rset を切り離すには、次のように入力します。

```
detachrset 21414
```

ファイル

項目	説明
/usr/bin/detachrset	**detachrset** コマンドが入っています。

関連資料:

457 ページの『execrset コマンド』

関連情報:

attachrset コマンド

lsrset コマンド

mkrset コマンド

devinstall コマンド

目的

デバイス用のソフトウェア・サポートをインストールします。

構文

devinstall **-f** *File* **-d** *Device* [**-s**] [**-v**]

説明

devinstall コマンドは、デバイス用のソフトウェア・サポートをインストールします。 **-f** フラグで指定されたファイルにリストされているソフトウェア・パッケージをインストールします。

初期ソフトウェア・インストール後に追加するほとんどの新しいデバイスの場合、新しいデバイスに必要なソフトウェアは、**cfgmgr** コマンドの **-i** フラグを使用してインストールできます。

マシンの始動に必要なデバイスを新しいデバイスで置き換えることがあります。例えば、ルート・ボリューム・グループをサポートする SCSI アダプター・カード、またはコンソールをサポートするグラフィックス・アダプター・カードに換えることがあります。この場合、この新しいデバイスに必要なソフトウェア・サポートをインストールするまで、マシンは通常モードで始動しません。マシンを始動するには、システムの電源を切り、ハードウェアに添付されている指示に従って新しいハードウェアをインストールします。次に、マシンを保守モードで始動します。始動プロセス中に、新しいアダプターが検出され、**/tmp/device.pkgs** ファイルが作成されます。このファイルには、新しいハードウェアのサポートに必要なソフトウェア・パッケージの名前が書き込まれます。一度マシンを保守モードにすれば、**devinstall** コマンドを実行して新しいデバイスに必要なソフトウェアをインストールできます。

フラグ

項目	説明
-f *File*	インストールするパッケージのリストが入っているファイルを指定します。一般に、これは **cfgmgr** コマンドによって生成される **/tmp/device.pkgs** ファイルです。
-d *Device*	インストール・メディアがある場所を指定します。ここでは、テープやディスケットなどのハードウェア・デバイス、インストール・イメージが入っているディレクトリー、またはインストール・イメージそのものを指定することができます。インストール・メディアが IBM インストール・テープあるいは IBM 修正サービス・テープの場合は、テープ・デバイスを、no-rewind-on-close (クローズ時巻き戻しなし) と no-retension-on-open (オープン時保留なし) に指定しなければなりません。高密度テープの場合は **/dev/rmt0.1**、または低密度テープの場合は **/dev/rmt0.5** がこの例です。 IBM 提供のテープ以外の場合は、テープ供給者が指定するオプションを使用してください。
-s	**/var/adm/dev_pkg.fail** ファイルを上書きします。このファイルには、正常にインストールされなかったすべてのパッケージのリストが入っています。このリストを使用すると、異なるソースからのリカバリーまたはインストールが容易になります。
-v	詳細オプションを指定します。このオプションにより、**devinstall** コマンドは処理中に詳細情報を表示するようになります。

devinstall コマンドは、コマンド・ラインで指定したファイルに示されているデバイス・パッケージをインストールします。また、このコマンドは、**-I** "acXge **/var/adm/ras/devinst.log**" を指定すると、**geninstall** コマンドを実行します。ここで、a (apply)、c (commit)、X (extend fs)、g (auto_include)、e (log) はフラグを表し、 **/var/adm/ras/devinst.log** は、ログ・ファイルの絶対パス名です。 (これらのフラグの詳細については、**geninstall** コマンドのセクションを参照してください。) **devinstall** コマンドは、**geninstall** コマンドによって生成された要約ファイルを調べて、各パッケージのインストール試行の結果を検査し、この情報に基づいて 2 つのファイルを作成します。 **/var/adm/dev_pkg.fail** ファイルには、インストールできなかったパッケージ (ある場合) が記録されます。 **/var/adm/dev_pkg.success** ファイルには、正常にインストールされたすべてのパッケージが記録されます。

戻り値

戻り値 0 は、パッケージが 1 つもインストールされなかったことを示します。

戻り値 1 は、パッケージが少なくとも 1 つは正常にインストールされたことと、**bosboot** コマンドを実行されることを示します。

戻り値 2 は、**devinstall** コマンドが失敗したことを示しています。

/var/adm/dev_pkg.success ファイルには、正常にインストールされたパッケージが記録されます。
/var/adm/dev_pkg.fail ファイルには、インストールに失敗したパッケージが記録されます。

セキュリティー

特権制御: このコマンドを実行できるのは、root ユーザーだけです。

例

ソフトウェアをインストールして、マシンをデバイス・インストール・テープから始動させ、保守モードにした後で、新しいデバイスをサポートするには、次のように入力します。

```
devinstall -f /../tmp/device.pkgs -d /dev/rmt0.1
```

次に、**bosboot** コマンドを実行します。

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

ファイル

項目	説明
/dev/rmtn	ロウ・ストリーミング・テープ・インターフェースを指定します。

関連情報:

bosboot コマンド

cfgmgr コマンド

installp コマンド

---

devnm コマンド

目的

デバイスを命名します。

構文

devnm *Path* ...

説明

devnm コマンドは、*Path* パラメーターを読み取って、*Path* パラメーターがパスが常駐する、マウントされたファイルシステムに関連するスペシャル・ファイルを識別し、そのスペシャル・ファイルの名前を標準出力に書き出します。各 *Path* パラメーターは絶対パス名でなければなりません。

devnm コマンドは通常、**/etc/rc** コマンド・ファイルがルート・デバイスのマウント・テーブル・エントリーを作成するときに使用されます。

注**:** このコマンドは、ローカル・ファイルに対してのみ使用できます。

例

1. ファイルが入っているデバイスを確認するには、次のように入力します。

   ```
   devnm /diskette0/bob/textfile
   ```

   これによって、/diskette0/bob/textfile ファイルが常駐するスペシャル・デバイス・ファイルの名前が表示されます。ディスケットが /diskette0 デバイスとしてマウントされている場合は、**devnm** コマンドは次のメッセージを表示します。

```
fd0 /diskette0/bob/textfile
rfd0 /diskette0/bob/textfile
```

これは、/diskette0/bob/textfile ファイルが **/dev/fd0** ディスケット・ドライブに入っていることを意味します。

2. ファイルシステムが置かれているデバイスを確認するには、次のように入力します。

```
devnm /
```

これによって、ルート・ファイルシステム (/) が置かれているデバイスの名前が表示されます。画面には、次のリストが表示されます。

```
hd0 /
```

これは、ルート・ファイルシステム (/) が **/dev/hd0** デバイスに置かれていることを意味します。

ファイル

項目	説明
/dev	ディレクトリーを指定します。
/usr/sbin/devnm	**devnm** コマンドが入っています。

関連情報:

rc コマンド

---

devrsrv コマンド

目的

デバイス上で単一パス予約と永続予約の照会および破棄を行います。

構文

devrsrv -c *query* [**-e**] | *release* | **prin -s** *sa* | (**prout -s** *sa* **-r** *rkey* **-k** *sa_key* **-t** *prtype*) **-l** *devicename*

devrsrv -f -l *devicename*

devrsrv -d

説明

devrsrv コマンドは、デバイス上で単一パス予約と永続予約の照会および破棄を行います。このコマンドは、永続予約イン (**prin**) と永続予約アウト (**prout**) のサービス・アクションを実行します。

query サブコマンドは、デバイスの現在の予約状況を照会して表示します。 **release** サブコマンドは、単一パス予約を使用してデバイス上の予約を解除します。

prin サブコマンドは、すべての登録済みの予約キー、予約キー・ホルダー、および能力情報を表示します。**prout** サブコマンドは、デバイスへの特定の入出力パスを排他使用または共有使用するために、そのデバイスを予約するサービス・アクションを要求します。**prout** サブコマンドは、以下のサービス・アクションをサポートします。

項目	説明
RELEASE	デバイスに指定された永続予約を解除します。
CLEAR	すべての予約キーとすべての永続予約をクリアします。
PREEMPT	永続予約を優先使用するか、または登録を除去します。あるいはその両方を行います。
PREEMPT AND ABORT	永続予約を優先使用するか、または登録を除去します。あるいはその両方を行います。さらに、優先使用された、デバイスのすべての入出力パスに対して、すべてのタスクを停止します。
REGISTER AND IGNORE KEY	古いキー値の代わりに新しいキー値を登録します。

フラグ

項目	説明
-c	以下のサブコマンドを指定します。 照会　デバイス上の予約状況を照会して表示します。 **release**　SCSI-2 を使用して単一パス予約を持つデバイスを解放します。 **prin**　永続予約イン・サービス・アクションを指定します。 **prout**　永続予約アウト・サービス・アクションを指定します。
-d	**devrsrv** コマンドを使用して照会または操作されるすべてのディスクについて、ディスク名およびその他の識別情報をリストします。
-e	ディスクを排他モード (単一モードと診断モードの両方が含まれます) で開かないようにします。このフラグは、PR_exclusive と PR_shared の予約タイプにのみ適用できます。オブジェクト・データ・マネージャー (ODM) 予約ポリシーが single_path の場合、このフラグは無視されます。 **注:** このフラグを使用した場合、**devrsrv** コマンドが、ディスクの予約状況やディスクがローカル・ホスト上で既に開かれているかどうかを判別できないこともあります。
-f	他の入出力パスまたはホストが保有する予約を破棄します。単一パス予約の場合、**devrsrv** コマンドは SC_FORCED_OPEN アクションを発行して予約を破棄します。永続予約の場合、**devrsrv** コマンドは **prout** サブコマンドを CLEAR サービス・アクションと一緒に発行して、永続予約と登録をクリアします。
-k	サービス・アクションの予約キーを指定します。REGISTER、PREEMPT、および PREEMPT_ABORT の各サービス・アクションには、**-k** フラグが必須です。
-l	デバイス名を指定します。
-r	予約キーを指定します。REGISTER、PREEMPT、PREEMPT AND ABORT、および RELEASE の各サービス・アクションには、**-r** フラグが必須です。
-s	永続予約のサービス・アクションを指定します。**prin** サブコマンドの有効なサービス・アクションは、次のとおりです。 **0**　READ KEYS **1**　READ RESERVATION **2**　REPORT CAPABILITIES **prout** サブコマンドの有効なサービス・アクションは、次のとおりです。 **2**　RELEASE **3**　CLEAR **4**　PREEMPT **5**　PREEMPT AND ABORT **6**　REGISTER AND IGNORE EXISTING KEY

項目	説明
-t	永続予約のタイプを指定します。永続予約のタイプは、次のとおりです。 **1** 排他的書き込み **2** 排他的アクセス **3** 排他的書き込み登録者のみ **4** 排他的アクセス登録者のみ **5** すべての排他的書き込み登録者 **6** すべての排他的アクセス登録者

例

以下に、各種のシナリオに関する例を示します。

照会操作

1. hdisk0 デバイスがどのホストにも予約されていないときに、その予約状況を照会するには、次のコマンドを入力します。

   ```
   # devrsrv -c query -l hdisk0
   ```

   ```
   Device Reservation State Information
   ==================================================
   Device Name                     :  hdisk0
   Device Open On Current Host?    :  NO
   ODM Reservation Policy          :  SINGLE PATH RESERVE
   Device Reservation State        :  NO RESERVE
   ```

   この出力は、このデバイスが現在のホスト上でオープンされていないことと、「ODM Reservation Policy (オブジェクト・データ・マネージャー (ODM) 予約ポリシー)」が SINGLE PATH RESERVE であることを示しています。それは、予約ポリシーがこのデバイスに対して ODM の中に設定されていることを示します。「Device Reservation State (デバイス予約状態)」は、デバイス上に存在する予約を示します。SCSI コマンドのシーケンスを実行して、デバイス予約状態を検出することができます。
2. あるホストが hdisk1 デバイスを予約している場合、その予約状況を照会するには、次のコマンドを入力します。

   ```
   # devrsrv -c query -l hdisk1
   ```

   あるホストが単一パス予約を使用して、このデバイスを予約しています。

   ```
   Device Reservation State Information
   ==================================================
   Device Name                     :  hdisk1
   Device Open On Current Host?    :  NO
   ODM Reservation Policy          :  SINGLE PATH RESERVE
   Device Reservation State        :  SINGLE PATH RESERVE
   ```
3. hdisk2 デバイスが同じホスト上で予約されているときに、その予約状況を照会するには、次のコマンドを入力します。

   ```
   # devrsrv -c query -l hdisk2
   ```

   ```
   Device Reservation State Information
   ==================================================
   Device Name                     :  hdisk2
   Device Open On Current Host?    :  YES
   ODM Reservation Policy          :  SINGLE PATH RESERVE
   Device Reservation State        :  SINGLE PATH RESERVE
   Path Id of Reserved Path        :  0
   ```

4. ODM 予約ポリシーが PR SHARED の場合で、かつ、どのホストも hdisk2 デバイスを予約していない場合、そのデバイスの予約状況を照会するには、次のコマンドを入力します。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
==================================================
Device Name               :  hdisk0
Device Open               :  NO
ODM Reservation Policy    :  PR SHARED
ODM PR Key Value          :  7777
Device Reservation State  :  NO RESERVE
Registered PR Keys        :
555
777
PR Capabilities Byte[2]   :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]   :  0x0
PR Types Supported        :  NOT VALID
```

この照会出力のいくつかのフィールドの説明は、次のとおりです。

Registered PR Keys:
登録されたキーを表示します。このキーは、このデバイスを共有する全入出力パスから **prout** サブコマンドを REGISTER サービス・アクションと一緒に実行して登録されたものです。

PR Capabilities Bytes:
prin サブコマンドの REPORT CAPABILITIES サービス・アクションによって戻されたバイト 2 および 3 の内容を示します。例 4 の出力を解釈するには、SPC 規格を参照してください。

PR Types Supported:
デバイスがサポートしている永続予約タイプを表示します。このデバイスは、REPORT CAPABILITIES 出力の永続予約タイプ・マスク・フィールドで報告されたものです。

あるデバイス上で永続予約が保持されている場合、照会出力では、そのデバイスの予約に関する追加情報が次のように表示されます。

PR Reservation Type:
『フラグ』セクションに記載されている永続予約タイプ (PR Type) の値のいずれかを表示します。

PR Holder key Value:
現在の予約ホルダーの PR キー値を表示します。 PR Type が 5 または 6 の場合、永続予約キー値は 0 です。

永続予約イン **(prin)** 操作

1. すべての登録済みの予約キーを読み取るには、次のコマンドを入力します。

```
# devrsrv -c prin -s 0 -l hdisk0

Registered PR Keys              :
555
777
```

2. 現在の予約キー・ホルダーおよびタイプを読み取るには、次のコマンドを入力します。

```
# devrsrv -c prin -s 1 -l hdisk0

PR Generation Value     :  2
PR Type                 :  PR_EA_RO (EXCLUSIVE ACCESS, REGISTRANTS ONLY)
PR Holder Key Value     :  777
```

3. REPORT CAPABILITIES サービス・アクションを送信して、サポートされている PR 能力情報を戻すには、次のコマンドを入力します。

```
# devrsrv -c prin -s 2 -l hdisk0

PR Capabilities Byte[2]         :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]         :  0x0
PR Types Supported              :  NOT VALID
```

永続予約アウト **(prout)** 操作

RELEASE サービス・アクション

キー 1777 と PR 予約タイプ 4 で登録され、予約された IT-nexus から、永続予約を解除するには、次のコマンドを入力します。

```
# devrsrv -c prout -s 2 -r 1777 -t 4 -l hdisk0

Device Reservation State Information
================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  YES
ODM Reservation Policy          :  PR SHARED
ODM PR Key Value                :  7777
Device Reservation State        :  PR SHARED
Reservation will be cleared on the device. Do you want to continue y/n:y
```

この時点で照会を実行すると、「Device Reservation State」が「NO RESERVE」であると表示されます。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
================================================
Device Name              :  hdisk0
Device Open              :  NO
ODM Reservation Policy   :  PR SHARED
ODM PR Key Value         :  7777
Device Reservation State :  NO RESERVE
Registered PR Keys       :
555
1777
PR Capabilities Byte[2]  :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]  :  0x0
PR Types Supported       :  NOT VALID
```

CLEAR サービス・アクション

,

キー 555 で登録された入出力パスを使用して、CLEAR サービス・アクションを使用するデバイス・サーバーから永続予約を解除し、すべての登録を除去するには、次のコマンドを入力します。

```
# devrsrv -c prout -s 3 -r 555 -l hdisk0

Device Reservation State Information
================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  YES
ODM Reservation Policy          :  PR SHARED
ODM PR Key Value                :  5555
Device Reservation State        :  PR SHARED
Reservation will be cleared on the device. Do you want to continue y/n:y
```

この時点で照会を実行すると、デバイスから永続予約は解除されていて、登録は除去されています。

```
# devrsrv -c query -l hdisk0
```

```
Device Reservation State Information
==================================================
Device Name               :  hdisk0
Device Open               :  NO
ODM Reservation Policy    :  PR SHARED
ODM PR Key Value          :  5555
Device Reservation State  :  NO RESERVE
Registered PR Keys        :  No Keys Registered
PR Capabilities Byte[2]   :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]   :  0x0
PR Types Supported        :  NOT VALID
```

PREEMPT サービス・アクションと **PREEMPT_ABORT** サービス・アクション

予約ホルダー 444 で保持されている永続予約において、登録済みのキー 777 を持つ別の IT-nexus を優先するには、次のコマンドを入力します。

```
# devrsrv -c prout -s 4 -r 777 -k 444 -t 2 -l hdisk0
```

#devrsrv -c prout -s 4 -r 777 -k 444 -t 2 -l hdisk0 コマンドを実行する前は、照会出力は次のように表示されます。

```
# devrsrv -c query -l hdisk0
```

```
Device Reservation State Information
==================================================
Device Name               :  hdisk0
Device Open               :  NO
ODM Reservation Policy    :  PR SHARED
ODM PR Key Value          :  7777
Device Reservation State  :  PR EXCLUSIVE
PR Generation Value       :  5
PR Type                   :  PR_WE (WRITE EXCLUSIVE)
PR Holder Key Value       :  444
Registered PR Keys        :
777
444
PR Capabilities Byte[2]   :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]   :  0x0
PR Types Supported        :  NOT VALID
```

#devrsrv -c prout -s 4 -r 777 -k 444 -t 2 -l hdisk0 コマンドを実行した後のこの照会の出力では、この予約において、キー 777 を持つ IT-nexus が優先使用され、キー 444 の登録が抹消されたことが示されます。

```
# devrsrv -c query -l hdisk0
```

```
Device Reservation State Information
==================================================
Device Name              :  hdisk0
Device Open              :  NO
ODM Reservation Policy   :  PR SHARED
ODM PR Key Value         :  7777
Device Reservation State :  PR EXCLUSIVE
PR Generation Value      :  6
PR Type                  :  PR_EA (EXCLUSIVE ACCESS)
PR Holder Key Value      :  777
Registered PR Keys       :
777
PR Capabilities Byte[2]  :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]  :  0x0
PR Types Supported       :  NOT VALID
```

SINGLE PATH RESERVE ポリシーの **RELEASE** 操作

hdisk0 デバイス上の予約を解除するには、次のコマンドを入力します。

- シナリオ 1: 現在のホストは、その予約の所有者です。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
==================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  YES
ODM Reservation Policy          :  SINGLE PATH RESERVE
Device Reservation State        :  SINGLE PATH RESERVE
Path Id of Reserved Path        :  0
```

```
# devrsrv -c release -l hdisk0

Device Reservation State Information
==================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  YES
ODM Reservation Policy          :  SINGLE PATH RESERVE
Device Reservation State        :  SINGLE PATH RESERVE
Device is currently Open on this host by a process.Do you want to continue y/n:y
Command Successful
Reservation cleared on the device. Query operation may not work properly.
Close the application that holds the reservation and retry.
```

- シナリオ 2: 現在のホストは、その予約の所有者ではありません。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
==================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  NO
ODM Reservation Policy          :  SINGLE PATH RESERVE
Device Reservation State        :  SINGLE PATH RESERVE

Because the current host does not own the reservation on the device,
try the force option if you want to break the reservation.
```

```
# devrsrv -f -l hdisk0
```

このデバイスは、単一パス予約を使用して、すでに別のホストによって予約されています。

```
Device Reservation State Information
==================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  NO
ODM Reservation Policy          :  SINGLE PATH RESERVE
Device Reservation State        :  SINGLE PATH RESERVE
Reservation will be cleared on the device. Do you want to continue y/n:y
```

release コマンドが正常に実行されると、下記の照会オプションでは、「Device Reservation State (デバイス予約状態)」が NO RESERVE であると表示されることになります。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
==================================================
Device Name                     :  hdisk0
Device Open On Current Host?    :  NO
ODM Reservation Policy          :  SINGLE PATH RESERVE
Device Reservation State        :  NO RESERVE
```

強制モード

hdisk0 デバイスは、別の入出力パスから、キー 777 で予約されています。この予約を別のクライアントから解除するには、次のコマンドを入力します。

```
# devrsrv -f -l hdisk0

Device Reservation State Information
==================================================
Device Name                     :  hdisk16
Device Open On Current Host?    :  NO
ODM Reservation Policy          :  PR SHARED
ODM PR Key Value                :  5555
Device Reservation State        :  PR SHARED
Reservation will be cleared on the device. Do you want to continue y/n:y
Command Successful
```

devrsrv -f -l hdisk0 コマンドを実行する前は、照会を行うと次の出力が表示されます。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
==================================================
Device Name               :  hdisk0
Device Open               :  NO
ODM Reservation Policy    :  PR SHARED
ODM PR Key Value          :  5555
Device Reservation State  :  PR EXCLUSIVE
PR Generation Value       :  1
PR Type                   :  PR_WE (WRITE EXCLUSIVE)
PR Holder Key Value       :  777
Registered PR Keys        :
777
PR Capabilities Byte[2]   :  0xd  SIP_C   ATP_C   PTPL_C
PR Capabilities Byte[3]   :  0x0
PR Types Supported        :  NOT VALID
```

devrsrv -f -l hdisk0 コマンドを実行した後、下記のコマンド実行結果で表示されるのは、デバイスが予約されていないことです。

```
# devrsrv -c query -l hdisk0

Device Reservation State Information
==================================================
Device Name                     :  hdisk16
Device Open On Current Host?    :  NO
ODM Reservation Policy          :  PR SHARED
ODM PR Key Value                :  5555
Device Reservation State        :  NO RESERVE
Registered PR Keys              :  No Keys Registered
PR Capabilities Byte[2]         :  0x0
PR Capabilities Byte[3]         :  0x0
PR Types Supported              :  NOT VALID
```

関連情報:

マルチパス入出力 (Multiple Path I/O)

T10 技術委員会

---

df コマンド

目的

ファイルシステム上のスペースに関する情報を報告します。本書では、AIX **df** コマンドと System V バージョンの **df** について説明しています。

構文

df [[**-P**] | [**-I** | **-M** | **-i** | **-t** | **-v**]] [**-c**] [**-T** { `local` | `remote` | `vfstype`}] [**-F** {*output1 output2 output3* ...}] [**-k**] [**-m**] [**-g**] [**-s**] [*FileSystem* ... | *File...*]

説明

df コマンドは、ファイルシステム上の合計スペースと使用可能なスペースの情報を表示します。*FileSystem* パラメーターは、ファイルシステムが置かれているデバイスの名前、ファイルシステムがマウントされているディレクトリー、またはファイルシステムの相対パス名を指定します。*File* パラメーターには、ファイルまたはマウント・ポイントでないディレクトリーを指定します。*File* パラメーターを指定すると、 **df** コマンドはそのファイルまたはディレクトリーが置かれているファイルシステムの情報を表示します。
FileSystem または *File* パラメーターを指定しない場合、**df** コマンドは現在マウントされているすべてのファイルシステムの情報を表示します。ファイルシステム統計情報は、デフォルトではブロック単位 (512 バイト/ブロック) で表示されます。

df コマンドは、**statfs** システム呼び出しからファイルシステムのスペースに関する統計情報を入手しますが、**-s** フラグを指定すると、仮想ファイルシステム (VFS) 固有のファイルシステム・ヘルパーから統計情報を入手します。 **-s** フラグで引数を指定していないために、ヘルパーが統計情報を入手できなかった場合は、**statfs** システム呼び出しの統計情報が使用されます。 **df** コマンドの実行中にファイルシステムが変更されるなどのある種の例外的な状況では、**df** コマンドが表示する統計が正確でないことがあります。

注**:** ネットワーク・ファイルシステム (NFS) などの一部のリモート・ファイルシステムには、 **df** コマンドが必要とするすべての情報を提供しないものがあります。**df** コマンドは、サーバーから提供されなかった統計についてはブランクを出力します。

df コマンドは、NFSv4 ファイルシステムを完全にはサポートしません。ブロックおよびスペース情報の抽出には、**nfs4cl** コマンドを使用してください。

フラグ

項目	説明
-c	コロンで区切られたフォーマットで出力を表示します。
-F { *output1* *output2* *output3* ... }	出力パラメーターの見出しで指定された値のみを表示します。デフォルトでは、ファイルシステムの見出しとブロック割り振りの見出しが常にオンになっています。 次の値は、見出しの値として許容されます。 **%m** マウント済み **%u** 使用済み **%z** 使用済みパーセンテージ **%f** 空き **%l** 使用済み I ノード **%n** 空き I ノード **%p** 使用済み I ノードのパーセンテージ
-g	GB ブロックのユニットで統計情報を表示します。各ユニットのバイトでの値は非常に高いため、ファイルシステムの統計情報についての出力値は、浮動小数点を使用した数になります。
-i	ファイルシステム用に使用された inode の数、および使用中の inode のパーセンテージを表示します。指定されたファイルシステムがマウントされている場合は、デフォルトでこの情報が出力されます。
-I	ファイルシステムのブロックの合計数、使用されているスペース、フリー・スペース、使用されているスペースの割合のパーセント、マウント・ポイントなどの情報を表示します。
-k	ブロック単位 (1024 バイト/ブロック) で統計を表示します。

項目	説明
-m	MB ブロックのユニットで統計情報を表示します。各ユニットのバイトでの値は非常に高いため、ファイルシステムの統計情報についての出力値は、浮動小数点を使用した数になります。
-M	ファイルシステムのマウント・ポイント情報を 2 番目の列に表示します。
-P	ファイルシステムの情報を POSIX 移送可能フォーマットで表示します。 **-P** フラグを指定すると、ヘッダー行が次のように表示されます。 `Filesystem 512-blocks Used Available Capacity Mounted on¥n` **-P** フラグとともに **-k**、**-m**、または **-g** フラグを指定すると、 **-P** フラグとともにどのフラグが使用されているかによって、列見出しの 512-blocks がそれぞれのユニットに置き換えられます。 ファイルシステム統計情報は、次の順序で 1 行に表示されます。 *FileSystem, TotalSpace, UsedSpace, FreeSpace, UsedPercentage, MountPoint*
-s	コマンド・ライン引数でアンマウント済み JFS または拡張 JFS ファイルシステムの統計を表示します。**-s** フラグは、引数を指定しないと効果がありません。引数で指定されたファイルシステムが現在マウント済みであったり、引数がファイルの場合、**-s** フラグはその特定の引数には効果がありません。アンマウント済みファイルシステムの統計を収集するには、引数は JFS または拡張 JFS ファイルシステムのマウント・ポイントまたはデバイスであり、そのファイルシステムは **/etc/filesystems** にリストされている必要があります。また、ユーザーはそのデバイスに対して読み取りアクセス権を持っていなければなりません。
-t	割り当て済みの合計スペースを示す数値を出力に含めます。
-T { local | remote | vfstype }	ファイルシステムのタイプで出力にフィルターを掛けます。このフラグには次のいずれかのパラメーターを指定できます。 **local** ジャーナル・ファイルシステム (JFS) および拡張ジャーナル・ファイルシステム (JFS2) のファイルシステムのみを表示します。 **remote** ローカル以外のファイルシステムをすべて表示します。 **vfstype** 特定の仮想ファイルシステム (VFS) (例えば JFS、JFS2、ネットワーク・ファイル・システム・バージョン 4 (NFSv4) など) のファイルシステムのみを表示します。
-v	指定されたファイルシステムに関するすべての情報を表示します。

-m および **-g** フラグで指定されている出力パラメーターの値は、最も近い小数第 2 位に丸められます。**-k**、**-m**、および **-g** フラグで、すべてまたはいずれか 2 つのフラグが指定された場合は、最後に指定されたものが有効になります。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. マウントされているすべてのファイルシステムに関する情報を表示するには、次のように入力します。

   ```
   df
   ```

   システムで **/**、**/usr**、**/site**、**/usr/venus** といった各ファイルシステムがマウントされている場合は、**df** コマンドからの出力は次のようなものになります。

   ```
   Filesystem 512-blocks Free   %Used   Iused  %Iused  Mounted on
   /dev/hd0    19368     9976    48%     4714    5%     /
   /dev/hd1    24212     4808    80%     5031   19%     /usr
   /dev/hd2     9744     9352     4%     1900    4%     /site
   /dev/hd3     3868     3856     0%      986    0%     /usr/venus
   ```

2. /test ファイルシステムに関する情報を 1024 バイトのブロックで表示するには、次のように入力します。

   ```
   df -k /test
   ```

   ```
   Filesystem    1024 blocks      Free    %Used    Iused  %Iused  Mounted on
   /dev/lv11         16384     15824       4%       18      1%  /tmp/ravi1
   ```

   これにより、ファイルシステムの統計情報が 1024 バイトのディスク・ブロックで表示されます。

3. /test ファイルシステムに関する情報を MB ブロックで表示するには、次のように入力します。

   ```
   df -m /test
   ```

   ```
   Filesystem    MB blocks      Free    %Used    Iused  %Iused  Mounted on
   /dev/lv11         16.00     15.46       4%       18      1%  /tmp/ravi1
   ```

   これにより、ファイルシステムの統計情報が、最も近い小数第 2 位に丸められ、MB ディスク・ブロックで表示されます。

4. **/test** ファイルシステムに関する情報を GB ブロックで表示するには、次のように入力します。

   ```
   df -g /test
   ```

   ```
   Filesystem    GB blocks   Free     %Used    Iused  %Iused  Mounted on
   /dev/lv11          0.02   0.02        0%       18      1%  /tmp/ravi1
   ```

   これにより、ファイルシステムの統計情報が、最も近い小数第 2 位に丸められ、GB ディスク・ブロックで表示されます。

5. 現行ディレクトリーが置かれているファイルシステムの、使用可能スペースを表示するには、次のように入力します。

   ```
   cd/
   df .
   ```

   このコマンドからの出力は次のようなものになります。

   ```
   Device   512-blocks  free   %used   iused   %iused  Mounted on
   /dev/hd4    19368    9976    48%     4714    5%     /
   ```

6. 出力をコロンで区切られたフォーマットで表示するには、次のように入力します。

   ```
   df -c
   ```

   出力は以下の例のようになります。

   ```
   Filesystem:512-blocks:Free:%Used:Iused:%Iused:Mounted on
   /dev/hd4:491520:113168:77%:9930:42%:/
   /dev/hd2:5046272:27696:100%:43014:86%:/usr
   ```

7. ローカルにマウントされているすべてのファイルシステムに関する情報を表示するには、次のように入力します。

   ```
   df -T local
   ```

   出力は以下の例のようになります。

   ```
   Filesystem    512-blocks      Free  %Used    Iused %Iused Mounted on
   /dev/hd4          5898240   2104184   65%    16390     7%  /
   /dev/hd2          7602176   1698696   78%    56001    23%  /usr
   /dev/hd9var       3014656   2190976   28%    10987     5%  /var
   /dev/hd3          2883584   2137928   26%     1213     1%  /tmp
   /dev/hd1          655360    645240    2%     1727     3%   /home
   /dev/hd11admin    262144    261384    1%        5     1%   /admin
   /proc                  -         -    -         -     -    /proc
   /dev/hd10opt      786432    362672   54%     8926    18%   /opt
   /dev/livedump     524288    523552    1%        4     1%   /var/adm/ras/livedump
   /aha                   -         -    -       328     2%   /aha
   ```

8. すべての JFS2 ファイルシステムに関する情報を表示するには、次のように入力します。

   ```
   df -T jfs2
   ```

   出力は以下の例のようになります。

   ```
   Filesystem     512-blocks      Free %Used    Iused %Iused Mounted on
   /dev/hd4          5898240   2104184   65%    16390     7%  /
   /dev/hd2          7602176   1698696   78%    56001    23%  /usr
   /dev/hd9var       3014656   2190976   28%    10987     5%  /var
   /dev/hd3          2883584   2137928   26%     1213     1%  /tmp
   /dev/hd1           655360    645240    2%     1727     3%  /home
   /dev/hd11admin     262144    261384    1%        5     1%  /admin
   /dev/hd10opt       786432    362672   54%     8926    18%  /opt
   /dev/livedump      524288    523552    1%        4     1%  /var/adm/ras/livedump
   ```

9. すべての JFS2 ファイルシステムに関する、Free、Used、および Mounted on の情報を表示するには、次のように入力します。

   ```
   df -T jfs2 -F %f %u %m
   ```

   出力は以下の例のようになります。

   ```
   Filesystem    512-blocks      Free %Used    Mounted on
   /dev/hd4         5898240   2104184   65%    /
   /dev/hd2         7602176   1698696   78%    /usr
   /dev/hd9var      3014656   2190976   28%    /var
   /dev/hd3         2883584   2137928   26%    /tmp
   /dev/hd1          655360    645240    2%    /home
   /dev/hd11admin     262144    261384        /admin
   /dev/hd10opt      786432    362672        /opt
   /dev/livedump     524288    523552        /var/adm/ras/livedump
   ```

ファイル

項目	説明
/etc/filesystems	既知のファイルシステムをリストし、その特性を定義します。
/etc/vfs	仮想ファイルシステムのタイプに関する記述が入っています。

System V df コマンド

目的

フリーのディスク・ブロックおよびファイルの数を報告します。

構文

/usr/sysv/bin/df [**-a**] [**-l**] [[[**-e**] [**-g**] [**-n**]] | [[**-i**] [**-v**]] | **-t**]] [*FileSystem* ...] [*File* ...]

説明

df コマンドは、ファイルシステム上の合計スペースと使用可能なスペースの情報を表示します。ファイルシステムの統計情報は、 512 バイト・ブロックのユニットで表示されます。

フラグ

項目	説明
-a	デフォルトの操作を行い、マウント・ポイント、デバイス名、フリー・ブロックの数、および使用された inode (ファイル) の数を出力します。
-e	フリー・ファイルの数のみを出力します。
-g	すべての **statvfs** 構造を出力します。このオプションは、**-a**、 **-e**、**-i**、**-n**、 **-t**、および **-v** オプションを指定変更します。使用可能な数、合計数、およびフリー・ブロックの数が 512 バイトブロックで報告されます。
-i	inode の合計数、フリー inode の数、使用された inode の数、および使用中の inode のパーセンテージ (%) を表示します。
-l	ローカル・ファイルシステムについてのみ報告します。
-n	ファイルシステムのタイプを出力します。
-t	割り当てられたブロック数の合計を報告します。
-v	使用され、フリーであるブロックの数とともに、使用されたブロックのパーセンテージ (%) を報告します。

パラメーター

項目	説明
File	*File* パラメーターには、ファイルまたはマウント・ポイントでないディレクトリーを指定します。*File* パラメーターを指定すると、 **df** コマンドはそのファイルまたはディレクトリーが置かれているファイルシステムの情報を表示します。
FileSystem	*FileSystem* パラメーターは、ファイルシステムが置かれているデバイスの名前、ファイルシステムがマウントされているディレクトリー、またはファイルシステムの相対パス名を指定します。

注**:** *FileSystem* または *File* パラメーターを指定しない場合、**df** コマンドはすべての現在マウントされているファイルシステムについての情報を表示します。

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

例

1. マウントされているすべてのファイルシステムに関する情報を表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df
   ```

   出力は、次のようになります。

   ```
   /              (/dev/hd4       ):     19656 blocks    1504 files
   /usr           (/dev/hd2       ):   1139904 blocks   20254 files
   /var           (/dev/hd9var    ):     23096 blocks     512 files
   /tmp           (/dev/hd3       ):      2464 blocks     204 files
   /home          (/dev/hd1       ):     44208 blocks     146 files
   /proc          (/proc          ):         0 blocks       0 files
   /opt           (/dev/hd10opt   ):     13880 blocks     310 files
   ```

2. 現行ディレクトリーが置かれているファイルシステムについての情報を表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df .
   ```

3. すべてのマウントされたファイルシステム内の inode の合計数、フリーの inode の数、および使用可能な inode の数を表示するには、次のように表示します。

   ```
   /usr/sysv/bin/df -i
   ```

   出力は、次のようになります。

```
Mount Dir  Filesystem      iused    avail    itotal  %iused
/          /dev/hd4         1504     6688      8192     19%
/usr       /dev/hd2        20254   127202    147456     14%
/var       /dev/hd9var       512     3584      4096     13%
/tmp       /dev/hd3          204     5940      6144      4%
/home      /dev/hd1          146    14190     14336      2%
/proc      /proc               0        0         0       0
/opt       /dev/hd10opt      310     5834      6144      6%
```

4. **/tmp** ファイルシステム上のブロックの合計数、使用されたブロックの数、およびフリー・ブロックの数を表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df -v /tmp
   ```

5. ファイルシステムのタイプを表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df -n
   ```

6. すべてのローカル・ファイルシステム上の inode 情報を表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df -i -l
   ```

7. すべてのファイルシステム上の statvfs 構造情報を表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df -g
   ```

8. ファイルシステム上のフリー・ファイルの数を表示するには、次のように入力します。

   ```
   /usr/sysv/bin/df -e
   ```

ファイル

項目	説明
/usr/sysv/bin/df	System V **df** コマンドが入っています。
/etc/filesystems	ファイルシステム情報が入っています。

関連資料:

611 ページの『fsck コマンド』

関連情報:

filesystems ファイル

ファイルシステム

---

dfmounts コマンド

目的

マウントされたリソースの情報を表示します。

構文

dfmounts [**-F** *fstype*] [**-h**] [*server* ...]

説明

dfmounts コマンドは、ネットワーク・ファイルシステム (NFS) を介してクライアントによってリモート側でマウントされたローカル・システムを印刷します。また、リソースをマウントしたクライアントのリストも印刷します。 **dfmounts** コマンドは、ヘッダーと、フィールド内でその後に続く、空白文字で区切られたリソース情報のリストを印刷します。

各リソースについて、次のフィールドが表示されます。

RESOURCE
NFS については、ハイフン (-) でマークされています。

SERVER
リソースがマウントされていたマシンを示します。

PATHNAME
共用リソースのパスを示します。

CLIENTS
現在マウントされているリソースを持つシステムの、コンマ (,) で区切ったリスト。

フラグ

項目	説明
-F *fstype*	ファイルシステムのタイプ (fstype) を指定します。**nfs** タイプのファイルシステムのみがサポートされています。
-h	**dfmounts** の出力で、ヘッダー行を表示しません。

パラメーター

項目	説明
Server	ローカル・システムにリソースを使用可能にした、ネットワーク上のシステムを表します。*Server* は、マシンから利用可能にされたリソースを、各リソースを使用している現在のクライアントとともに出力します。このパラメーターを指定しない場合は、 **dfmounts** コマンドは、サーバーがローカル・システムであると想定した情報を出力します。**dfmounts** コマンドには、複数のサーバー名を指定できます。

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

セキュリティー

例

1. ファイルシステム・タイプ「**nfs**」について、システム「mercury」上のマウントされたリソースの情報を出力するには、次のように入力します。

   ```
   dfmounts -F nfs mercury
   ```

2. ファイルシステム・タイプ「**nfs**」について、システム上のマウントされたリソースの情報をヘッダーなしで印刷するには、次のように入力します。

   ```
   dfmounts -hF nfs
   ```

ファイル

項目	説明
/usr/bin/dfmounts	汎用の System V **dfmounts** コマンドが入っています。
/usr/lib/fs/nfs/dfmounts	nfs 用の System V **dfmounts** コマンドが入っています。
/etc/vfs	既知の仮想ファイルシステム実装についての記述が含まれています。

関連資料:

132 ページの『dfshares コマンド』

dfpd コマンド

目的

ロード・バランシングされているサーバーについての負荷統計情報をロード・マネージャーに提供します。

構文

/usr/sbin/dfpd [**-d**] [**-f** *ConfigurationFile*]

説明

DFP デーモン (**dfpd**) は、ロード・バランシングされているサーバー上で実行し、そのサーバーについての負荷統計情報をロード・マネージャーに提供します。これにより、ロード・マネージャーは、より将来の接続を使用可能なサーバーに送信することができ、このことがロード・バランシングに役立ちます。

dfpd デーモンは始動時に、*ConfigurationFile* パラメーターで指定されたファイルから自己の構成情報を読み取ります。パラメーターが指定されていない場合は、**dfpd** デーモンは自己の構成情報を **/etc/dfpd.conf** ファイルから読み取ります。

始動すると、**dfpd** デーモンは、構成ファイルで指定されたポートでロード・マネージャーからの接続を listen します。

DFP デーモンの構成ファイル

/etc/dfpd.conf ファイルは、編集することによって更新できます。 **/etc/dfpd.conf** ファイルのエントリーには、次の情報が含まれています。

MD5 キー・エントリーは秘密鍵 (64 文字までの) を指定します。この秘密鍵は、DFP クライアント、サーバー、およびロード・マネージャー間で同一でなければなりません。 MD5 キー・エントリーの例は、次のようになります。

```
md5key 1234567890abcdefabcdef12345678901234567890abcdefabcdef1234567890
```

ロード・マネージャー listener エントリーは、DFP サーバーがロード・マネージャーの接続を listen するポートを指定します。ロード・マネージャー・エントリーの例は、次のようになります。

```
ldlistener 9503
```

ポーリング・アイドル時間エントリーは、CPU アイドル時間の連続計算間の期間を指定します。ポーリング・アイドル時間エントリーの例は、次のようになります。

```
pollidletime 30
```

計算されたアイドル時間は、*mfactor* 値によって乗算されてから、ロード・マネージャーに報告されます。このことは、異なる能力のマシン間で重みを合理化するのに有用です。デフォルト値は、ホスト上の CPU の数です。 mfactor エントリーの例は、次のようになります。

```
mfactor 1
```

フラグ

項目	説明
-d	デバッグ・モードで実行し、デーモン・プロセスにはなりません。
-f *ConfigurationFile*	指定された *ConfigurationFile* をデーモンに使用させます。

dfsck コマンド

目的

異なるドライブ上にある 2 つのファイルシステムを同時に検査および修理します。

構文

dfsck [*FlagList1*] *FileSystem1* [*FlagList2*] *FileSystem2*

説明

dfsck コマンドを使用すると、2 つの異なるドライブ上にある 2 つのファイルを同時に検査できます。 2 組のファイルシステムにフラグとパラメーターを渡すには、*FlagList1* と *FlagList2* パラメーターを使用します。 *FlagList1* および *FlagList2* の有効なフラグのリストについては、それらのフラグのセクションを参照してください。フラグを引数の一部として指定する場合は、**-** (負符号) を使用して、ファイルシステム・グループのフラグを分離します。

dfsck コマンドを使用すると、 2 つの **fsck** コマンドと同時に対話できます。この場合の補助として、**dfsck** コマンドは各メッセージと一緒にファイルシステム名を表示します。 **dfsck** コマンドの質問に答えるときには、応答に 1 または 2 のプレフィックスを付けて、答が第 1 または第 2 のうち、どのファイルシステム・グループに関するものかを明示します。

重要**:** ルート・ファイルシステムの検査には、**dfsck** コマンドを使用しないでください。

フラグ

項目	説明
-d*BlockNumber*	指定されたディスク・ブロックへの参照を検索します。 **fsck** コマンドは、指定されたブロックを含むファイルを検出するたびに、関連する i ノード番号とすべてのパス名を表示します。
-f	高速検査を行います。通常は、正しい方法で電源を切らずにシステムを停止したために影響を受けるのは、そのときにマウントされているファイルシステムだけです。 **-f** フラグを指定すると、**fsck** コマンドは、正常にアンマウントされたファイルシステムの検査を行いません。 **fsck** コマンドは、ファイルシステム・スーパーブロック内の、 **s_fmod** フラグを検査することにより、正常にアンマウントされたファイルシステムを判別します。このフラグはファイルシステムをマウントするたびに設定され、正常にアンマウントすると消去されます。ファイルシステムが正常にアンマウントされれば、何も問題が発生するようなことはありません。ほとんどのファイルシステムは正常にアンマウントされるため、このようなファイルシステムは検査されず、検査に要する時間が短縮できます。
-i*i-NodeNumber*	指定された i ノードへの参照を検索します。**fsck** コマンドは、指定された i ノードのディレクトリー参照を検出するたびに、参照先への絶対パス名を表示します。
-n	**fsck** コマンドのすべての質問に対して `no` という応答が与えられるものと見なします。指定したファイルシステムは書き込み用にオープンされません。

項目	説明
-o *Options*	**fsck** コマンドにコンマで区切られたオプションを渡します。これらのオプションは、インプリメントされたファイルシステムに固有なオプションと見なされます。ただし、現在、次の項目はすべてのファイルシステムでサポートされています。 **mountable** 対象のファイルシステムがマウント可能な (現在アンマウントされている) 場合、値 0 を戻して **fsck** コマンドを正常終了させます。ファイルシステムがマウント可能でない場合、**fsck** コマンドは値 8 を戻して終了します。 **mytype** 該当するファイルシステムが、 **/etc/filesystems** ファイル内で指定されたものか、コマンド・ライン上の **-V** フラグで指定されたものと同じタイプである場合に、 **fsck** コマンドを正常終了 (0) させます。その他の場合は 8 の値が戻されます。例えば、 **/** (ルート・ファイルシステム) がジャーナル・ファイルシステムであれば、 `fsck -o mytype -V jfs /` は 0 値で終了します。
-p	小さな問題についてのメッセージは表示せず自動的に修正します。このフラグは、**-y** フラグとは異なり、すべてをシステム側にまかせるわけではありません。システムの通常の始動時に自動検査を行う場合に役に立ちます。システムが自動的に実行される場合は常に、システム始動手続きの一部として、このフラグを使用する必要があります。グループ単位の並列検査も可能です。
-t*File*	**fsck** コマンドでテーブルを格納するための十分なメモリーが獲得できない場合に、*File* パラメーターを検査対象のファイルシステム以外のファイルシステム上のスクラッチ・ファイルとして指定します。 **-t** フラグを指定せず、 **fsck** コマンドがスクラッチ・ファイルを必要としている場合には、fsck コマンドはスクラッチ・ファイル名を求めるプロンプトを表示します。ただし **-p** フラグを指定している場合は、 **fsck** コマンドは正常に実行されません。スクラッチ・ファイルがスペシャル・ファイルでなければ、**fsck** コマンド終了時に除去されます。
-V *VfsName*	**/etc/filesystems** ファイルの代わりに、*VFSName* 変数で指定された仮想ファイルシステムの記述を使って、記述を決定します。コマンド・ラインに **-V** *VfsName* フラグを指定しないと、**/etc/filesystems** ファイルが検査され、一致するスタンザの **vfs=***Attribute* が、正しいファイルシステム・タイプであると見なされます。
-y	**fsck** コマンドが発行するすべての質問に対して、 yes という応答が与えられるものと仮定します。このフラグによって、 **fsck** コマンドは必要と考えられるすべての処理を行います。このフラグは、損傷の激しいファイルシステムだけに使用してください。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 2 個の異なるドライブ上の 2 つのファイルを同時に調べるには、次のように入力します。

   ```
   dfsck  -p /dev/hd1 -  -p /dev/hd7
   ```

   /dev/hd1 と **/dev/hd7** デバイス上にあるファイルシステムが、異なる 2 個のドライブ上に置かれている場合、上記のコマンドは 2 つのファイルを同時に検査します。また、**/etc/filesystems** ファイル内のファイルシステム名を、指定することもできます。

ファイル

項目	説明
/usr/sbin/dfsck	**dfsck** コマンドが入っています。
/etc/filesystems	既知のファイルシステムをリストし、その特性を定義します。
/etc/vfs	仮想ファイルシステムのタイプに関する記述が入っています。
/etc/rc	システム始動時に実行されるコマンド (**fsck** コマンドなど) が入っています。

関連資料:

611 ページの『fsck コマンド』

関連情報:

istat コマンド

filsys.h ファイル

ファイルシステム

---

dfshares コマンド

目的

リモート・システムから使用可能なリソースをリストします。

構文

dfshares [**-F** *FileSystemType*] [**-h**] [*Server ...*]

説明

dfshares コマンドは、ネットワーク・ファイルシステム (NFS) を介してホストに使用可能にされたリソースについての情報を提供します。**dfshares** コマンドは、ヘッダー行と、その後にフィールドの区切り文字としてスペースを含んだ行のリストを出力します。

各リソースについて、次のフィールドが表示されます。

RESOURCE
server:path のフォームでエクスポートされたリソース名を表示します。

SERVER
リソースを提供しているマシンを表示します。

ACCESS
そのクライアント・システムに与えられたアクセス権限を表示します。しかし、**dfshares** は NFS リソースについてはこの情報を判別できないため、そのフィールドにハイフン (-) を転送します。

TRANSPORT
共用するリソースへのトランスポート・プロバイダーを表示します。しかし、**dfshares** は NFS リソースについてはこの情報を判別できないため、そのフィールドにハイフン (-) を転送します。

フラグ

項目	説明
-F *FileSystemType*	ファイルシステムのタイプを指定します。ファイルシステム **nfs** タイプのみサポートされます。
-h	**dfshares** の出力で、ヘッダー行を表示しません。

パラメーター

項目	説明
Server	ローカル・マシンにリソースを提供したネットワーク上のシステムを表します。このパラメーターを指定しない場合は、**dfshares** コマンドは、ローカル・システム自体についての情報を出力します。**dfshares** には、複数のサーバー名を指定できます。

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

例

1. **nfs** タイプのファイルシステムについてシステム mercury 上のリソース情報を出力するには、次のように入力します。

   ```
   dfshares -F nfs mercury
   ```

2. システム上のリソース情報をヘッダー無しで出力するには、次のように入力します。

   ```
   dfshares -hF nfs
   ```

ファイル

項目	説明
/usr/bin/dfshares	汎用の System V **dfshares** コマンドが入っています。
/usr/lib/fs/nfs/dfshares	タイプ **nfs** のファイルシステム用の System V **dfshares** コマンドが入っています。
/etc/vfs	既知の仮想ファイルシステムのインプリメンテーションについての記述が入っています。

関連資料:

127 ページの『dfmounts コマンド』

---

dhcpaction コマンド

目的

クライアントがリースを更新するたびに実行するスクリプトを提供します。

構文

/usr/sbin/dhcpaction *HostName DomainName IPAddress LeaseTime ClientID* { **A** | **PTR** | **BOTH** | **NONE** } { **NONIM** | **NIM** }

説明

dhcpaction コマンドは、DNS サーバーを更新するメソッドを提供します。それは、適切なイベントの順序で **nsupdate** コマンドを呼び出して、A レコード、PTR レコード、またはその両方を更新することによって行われます。**dhcpaction** コマンドは、DHCP クライアントおよびサーバー・デーモンにより呼び出されます。これは、updateDNS 文字列から呼び出されます。この設定は構成可能です。一部の環境、主に異機種混合環境では、A レコードまたは PTR レコードを更新できないクライアントもあるからです。デフォル

トのアクションでは、クライアントによって A レコードが更新され、サーバーによって PTR レコードから更新されます。ネットワーク管理者が実装したい任意のポリシーを許可するために、デーモン構成ファイル内にオプションが設定されている場合があります。

dhcpaction コマンドは、NIM と DHCP も同時に実行します。NIM パラメーターを指定された **dhcpaction** コマンドは、NIM オブジェクトの IP アドレスの変更があると、NIM オブジェクトの更新を実行しようとします。このアクションにより、オブジェクトの同期をとり続けることができます。そのためには、一部の保留操作を取り消さなければならないことがあります。オブジェクトはコメントされ、メッセージがマスター・マシンのコンソールに送信されます。オブジェクトの頻繁なリセットは避けてください。通常、アドレスは DHCP 環境で変更しないでください。NONIM オプションの設定は、クライアントだけで行ってください。

パラメーター

項目	説明
ClientID	DNS サーバーを更新するときに使用するクライアント ID を指定します。
DomainName	DNS サーバーを更新するときに使用するドメイン・ネームを指定します。
HostName	DNS サーバー内の更新するホスト名を指定します。
IPAddress	DNS サーバーでホスト名と関連付ける IP アドレスを指定します。
LeaseTime	DNS サーバーでホスト名と IP アドレスが関連付けられている所要時間を秒で指定します。

オプション

項目	説明
A | **PTR** | **BOTH** | **NONE**	DNS サーバー内の更新する必要があるレコード (ある場合) を指定します。
NONIM | **NIM**	NIM と DHCP の正しい相互作用を支援するために、スクリプトを実行する必要があるかどうかを指定します。これは、DHCP サーバー上では **NIM** にのみ設定する必要があります。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: すべてのユーザーが可能ですが、一部の NIM アクションについては root である必要があります。

ファイル

項目	説明
/usr/sbin/dhcpaction	**dhcpaction** コマンドが入っています。
/etc/dhcpcd.ini	DHCP のクライアント構成ファイルが入っています。

関連資料:

141 ページの『dhcpsd デーモン』

138 ページの『dhcprd デーモン』

関連情報:

bootp 構成ファイル

TCP/IP アドレスとパラメーターの割り当て - 動的ホスト構成プロトコル

---

dhcpcd デーモン

目的

動的ホスト構成プロトコル (DHCP) クライアントをインプリメントします。アドレスおよび構成情報を DHCP サーバーに提供します。

構文

システム・リソース・コントローラー (SRC) を使用して DHCP クライアントを実装する

startsrc -s dhcpcd [**-a** *Argument*] ...

SRC を使用せずに DHCP クライアントを実装する

dhcpcd [**-f** *ConfigurationFile*] [**-i** *IPAddress*] [**-l** *LeaseFile*] [**-n**] [**-o** *OptionsFile*] [**-r**] [**-t** *Seconds*] [**-T** *Minutes*]

説明

dhcpcd デーモンは、DHCP プロトコルを使用して、IP (インターネット・プロトコル) アドレスとその他のパラメーターを設定することによって、DHCP をインプリメントします。

dhcpcd デーモンは、通常、開始時に通常実行される `/etc/rc.tcpip` ファイルによって開始されます。デフォルトで、それはコメント化され、システムの開始時には実行されません。DHCP クライアントを使用可能にする System Management Interface Tool (SMIT) オプションがあります。

dhcpcd デーモンは、その構成ファイルを読み取り、構成ファイル内に指定されたインターフェース用の IP アドレスとその他の構成オプションの開始と取得を試みます。システムが始動しているとき、**dhcpcd** デーモンは、バックグラウンドで実行されます。このデーモンは、既に受信したアドレスを必要に応じて更新します。

また、**-i** フラグを使用すると、**dhcpcd** デーモンは、DHCP Inform モードで実行されます。このモードを使用して、クライアントは、IP アドレスを取得せずに DHCP サーバーから構成情報を取り出すことができます。それは静的アドレスには便利ですが、印刷サーバーなどの動的項目やその他のオプションには便利ではありません。IP アドレス・パラメーターを指定して **-i** フラグを使用する場合、**dhcpcd** デーモンは指定したアドレスについて 1 回実行されます。

refresh コマンドは、**dhcpcd** デーモンに構成ファイルを再読み取りさせる場合に使用できます。同じ応答を得るには、**SIGHUP** も使用できます。

dhcpcd デーモンのデフォルトの構成ファイルは、/etc/dhcpcd.ini です。これには、ログおよびネットワーク・インターフェース情報が入っています。

SMIT の **smit usedhcp** 高速パスを使用して、このコマンドを実行できます。

フラグ

項目	説明
-f *ConfigurationFile*	使用する構成ファイルを指定します。デフォルトは /etc/dhcpcd.ini ファイルです。
-i *IPAddress*	**dhcpcd** デーモンが DHCP Inform モードを使用するように指定します。IP アドレスは、どのインターフェースで構成情報を取得するかを DHCP に指示します。
-l *LeaseFile*	別の lease ファイルを指定します。この lease ファイルは、リース獲得時にクライアントによって生成されます。デフォルトでは、lease ファイルは /etc/dhcpc.db です。
-n	新規アドレスの受信時にインターフェースが再構成されないようにします。
-o *OptionsFile*	options ファイルを指定します。デフォルトでは、options ファイルは /etc/dhcpc.opt です。
-r	クライアント・デーモンを起動し、1 回実行してから停止させます。
-t *Seconds*	DHCP がそれ自体をバックグラウンドに配置するまでの待機秒数を指定します。これによって、システムは DHCP サーバーが見つからない場合でもブートを続行できます。
-T *Minutes*	時間を分単位で指定します。 DHCP クライアントは、このタイムアウト値の時間内にインターフェースのアドレスを構成できない場合 (例えば DHCP サーバーが使用可能でないため)、それ以降の試行を停止します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

ファイル

項目	説明
/usr/sbin/dhcpcd	**dhcpcd** デーモンが入っています。
/etc/dhcpcd.ini	デフォルトのクライアント構成ファイルが入っています。
/etc/services	インターネット・サービスで使用するソケットおよびプロトコルを定義します。
/etc/inetd.conf	**inetd** デーモンにより制御されるサービスを定義します。

関連情報:

startsrc コマンド

inetd コマンド

DHCP サーバー構成ファイル

TCP/IP デーモン

dhcpcd6 デーモン

目的

IPv6 のための動的ホスト構成プロトコル (DHCPv6) クライアントをインプリメントします。 DHCPv6 サーバーから IPv6 ノードの IPv6 アドレスと構成情報を入手します。

構文

システム・リソース・コントローラー (SRC) を使用して DHCPv6 クライアントを始動する

startsrc -s dhcpcd6 [**-a** *Argument*] ...

SRC を使用せずに DHCPv6 クライアントを始動する

dhcpcd6 [**-f** *ConfigurationFileName*] [**-u** *Client_duid_File*] [**-p** *ClientPort*] [**-t** *SolicitTimeout*]

説明

dhcpcd6 デーモンは、DHCPv6 プロトコルを使用して、IPv6 (インターネット・プロトコル・バージョン 6) アドレスとその他のパラメーターを設定することによって、DHCPv6 クライアントをインプリメントします。

dhcpcd6 デーモンは、通常、ブート時に実行する `/etc/rc.net` ファイルによって始動されます。デフォルトで、それはコメント化され、マシンの開始時には実行されません。システムが始動しているとき、**dhcpcd6** デーモンは、バックグラウンドで実行されます。

dhcpcd6 デーモンは構成ファイルを読み取り、構成ファイル内に指定されたインターフェースをとるため 1 つ以上の IPv6 アドレスおよびその他の情報オプションを取り上げ、取り込もうとします。サーバーから入手したアドレスは、サーバーが指示するとおりに更新されます。

DHCPv6 クライアントとしては DHCPv6 サーバーに IPv6 アドレスを割り当ててもらう必要がない場合は、クライアントは使用可能な DNS サーバーまたは NTP サーバーのリストなどの構成情報のみを入手することができます。その方法は、ノードが静的アドレスで構成されているときは、有益です。

refresh コマンドは、**dhcpcd6** デーモンに構成ファイルを再読み取りさせる場合に使用できます。同じ応答を得るには、**SIGHUP** も使用できます。

デフォルトの **dhcpcd6** 構成ファイルは `/etc/dhcpv6/dhcpc6.cnf` です。これには、ログおよびネットワーク・インターフェース情報が入っています。

フラグ

項目	説明
-f *ConfigurationFileName*	使用する構成ファイルを指定します。デフォルトは `/etc/dhcpv6/dhcpc6.cnf` です。
-p *ClientPort*	使用するクライアント・ポートを指定します。デフォルトは `546` です。
-t *SolicitTimeout*	クライアントが、終了しないでその前にサーバーから構成情報を請求するまでの時間を指定します。
-u *Client_duid_File*	使用するクライアント ID ファイルを指定します。デフォルトは `/etc/dhcpv6/dhcpc6.duid` です。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

例

1. `/usr/local` にある構成ファイル `dhcpcd6.cnf` で DHCPv6 クライアントを始動するには、次のコマンドを入力します。

   ```
   startsrc -s dhcpcd6 -a "-f /usr/local/dhcpcd6.cnf"
   ```

位置

`/usr/sbin/dhcpcd6`

ファイル

項目	説明
`/usr/sbin/dhcpcd6`	**dhcpcd6** クライアント・デーモンが入っています。
`/etc/dhcpv6/dhcpc6.cnf`	デフォルトの構成ファイルが入っています。
`/etc/dhcpv6/dhcpc6.db`	クライアント・リース・ファイルが入っています。このファイルは、クライアント・デーモンによって作成されるものであり、構成可能ではありません。
`/etc/dhcpv6/dhcpc6.duid`	クライアント ID ファイルが入っています。このファイルは、クライアント・デーモンによって作成されるものであり、構成可能ではありません。

関連情報:

startsrc コマンド

---

dhcprd デーモン

目的

ローカル・ネットワークから BOOTP および動的ホスト構成プロトコル (DHCP) パケットを転送します。

構文

システム・リソース・コントローラー (SRC) を使用して DHCP サーバーに情報を転送する

startsrc -s dhcprd [**-a** *Argument*] [**-a** *Argument*] ...

SRC を使用せずに DHCP サーバーに情報を転送する

dhcprd [**-f** *ConfigurationFile*]

説明

dhcprd デーモンは、ブロードキャスト・パケットを listen し、それらを受信して、適切なサーバーに転送します。それにより、ブロードキャストが他のネットワークに伝搬しないようにします。DHCP リレー・エージェントは、ローカル・ネットワークから一連のサーバーへの DHCP および BOOTP クライアント・ブロードキャスト・パケットの転送を処理します。BOOTP または DHCP クライアントから送信される最初のパケットは、このクライアント・システムからローカル・インターフェースを介して送られるブ

ロードキャスト・ソケットです。これらのパケットは、ネットワーク・ゲートウェイやルーターを介して渡すことはできません。 BOOTP または DHCP リレー・エージェントである **dhcprd** デーモンが、これらのパケットを該当するサーバーに送信します。

DHCP サーバーは `/etc/services` ファイルを読み取り、要求の受信に使用する必要があるポートを判別します。デフォルトのサービスは **dhcps** です。これは、**bootpd** デーモンが使用するポートと同じなので、実行できるデーモンは 1 つだけ (**dhcprd** または **bootpd** のいずれか) です。 **dhcprd** デーモンを選択する場合は、**/etc/inetd.conf** ファイルの `bootp` をアンコメントしてから、コマンド・ラインに `refresh -s inetd` と入力する必要があります。

注**: bootpd** デーモンが実行されている場合は、このプログラムを停止してから、デーモンを始動する必要があります。

フラグ

項目	説明
-f *ConfigurationFile*	使用する構成ファイルを指定します。デフォルトは `/etc/dhcprd.cnf` ファイルです。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

ファイル

項目	説明
`/usr/sbin/dhcprd`	**dhcprd** デーモンが入っています。
`/etc/dhcprd.cnf`	デフォルトの構成ファイルが入っています。
`/etc/services`	インターネット・サービスで使用するソケットおよびプロトコルを定義します。
`/etc/inetd.conf`	**inetd** デーモンにより制御されるサービスを定義します。

関連情報:

inetd コマンド

TCP/IP アドレスとパラメーターの割り当て - 動的ホスト構成プロトコル

システム・リソース・コントローラー

---

dhcpsconf コマンド

目的

グラフィカル・ユーザー・インターフェース (GUI) を介して DHCP (動的ホスト構成プロトコル) のサーバー構成を単純化します。

構文

dhcpsconf

説明

dhcpsconf コマンドは、ネットワーク管理者に構成ファイルの読み取り、保管、および変更ができるようにする X Window System GUI を開きます。さらに、実行中のサーバーからの統計の開始、停止、および取り出しができるようにします。

dhcpsconf コマンドはリストのセットを表示します。左側のリストは、使用可能なオプションとキーを示します。 **dhcpsconf** コマンドは `/etc/options` ファイルから、基本オプションとキーを読み取り、これらを総称リソース・タイプとして実行を開始します。ネットワーク管理者は、GUI を使い「リソース」メニュー・ボタンを選択することによって名前付きリソースのセットを定義できます。

「リソース定義」ダイアログ・ボックスで、ネットワーク管理者はネットワークに関するすべてのオプションと具体的細目を登録できます。このようにして、ネットワーク管理者は、ネットワーク、プリンター、ネームサーバー、DHCP サーバーなど、リソース・オブジェクトを定義できます。一度それを行うと、これらの新規リソースはメイン・パネルのキーおよびオプション表示に追加されます。これらのリソースを使用することにより、サーバー構成ファイルまたはサーバー構成ファイルのセットを作成できます。

GUI の最初の表示は空のマスター・ファイルです。マスター・ファイルには 1 つ以上のサーバーの定義が格納されていますが、実際には 1 つのサーバー読み取り可能なファイルです。マスター・ファイルは 1 つの DHCP サーバーで読み取り可能ですが、複数のサーバー情報を格納できます。このため、ネットワーク管理者はネットワークの単一サーバー・イメージを構成して、同一セットのデータを処理するためにサーバーのセットを作成し、すべてのデータを 1 つのファイルとして管理、表示することができます。

オプションおよびキーをサーバー・ウィンドウに追加するには、キーまたはオプションを選択して、編集ウィンドウでそのオプションまたはキーの実行先を選択し、そのキーまたはオプション・セクションに対応する追加ボタンを選択します。オプションは指定した位置の編集ウィンドウに追加されます。項目が名前付きリソースである場合は、そのまま追加されます。項目が標準デフォルトの 1 つである場合は、その項目の値の入力を求めるウィンドウが表示されます。

DHCP サーバーは、配信範囲内の項目を担当するネットワーク内のシステムをそれらのサーバーが指定するという点を除いて、他のキーとまったく同じように追加されます。キーには有効範囲から見た順序と構文から見た順序があります。コメントは実際にはキーではなく、場所に制約はありません。

サーバーは、その中にネットワーク、クラス、クライアント、またはオプションが指定されていることがあります。ネットワークは、サブネット、クラス、クライアント、またはオプションを持つ場合があります。サブネットはクラス、クライアント、またはオプションを持つ場合があります。クラスおよびクライアントはオプションだけを持つことがあります。

サーバーには、サーバーのみに適用される一連の構成パラメーターがあります。これらは、キー・リストで DHCP サーバー・キーにより指定するか、または「Server (サーバー)」メニュー・バーの下にあるデフォルトのサーバー・オプションを使用して指定します。デフォルトのサーバー・オプションはマスター・ファイルにのみ適用されます。マスター・ファイルに指定された DHCP サーバーは、デフォルトのオプションを受信しますが、変更することもできます。

「編集」ウィンドウにあるすべての項目は、編集、名前変更、表示、および削除されることがあります。これによって、項目を配置して、それが適切かどうか調べ、必要に応じて変更することができます。

構成ファイルが完了したら、単一のマスター・ファイルを保管し、一連のサーバー・ファイルを生成することができます。「**File menu (**ファイル・メニュー**)**」ボタンおよび「**Server menu (**サーバー・メニュー**)**」ボタンのどちらにも、「save (保管)」オプションがあります。「**File save (**ファイルの保管**)**」ボタンは、マスター・ファイルを保管するためのものです。「**Server save (**サーバーの保管**)**」ボタンは、特定のサーバーをファイルに保管するためのものです。

「**File menu (**ファイル・メニュー**)**」ボタンには、「quit (終了)」オプション、ファイルを検索する「open (開く)」オプション、およびそれまで作成したすべてのものを消去するための「new (新規)」オプションが含まれています。

「**Operations menu (**操作メニュー**)**」ボタンには、「status (状況)」ボタン、「start (開始)」ボタン、「stop (停止)」ボタン、「refresh (リフレッシュ)」ボタンがあります。これらのボタンにより、リモート・サーバーは状況を報告したり、新しい構成ファイルによりリフレッシュしたり、停止したりすることができます。また構成ファイルを送信して再始動することができます。

「**Help (**ヘルプ**)**」ボタンには、各ウィンドウ項目を説明する一連のヘルプ・ステートメントが含まれています。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: すべてのユーザー

ファイル

項目	説明
/usr/sbin/dhcpsconf	**dhcpsconf** コマンドが入っています。
/etc/dhcpcd.cnf	デフォルトのクライアント構成ファイルが入っています。

関連資料:

135 ページの『dhcpcd デーモン』

関連情報:

DHCP クライアント構成ファイル

TCP/IP アドレスとパラメーターの割り当て - 動的ホスト構成プロトコル

---

dhcpsd デーモン

目的

動的ホスト構成プロトコル (DHCP) サーバーをインプリメントします。アドレスおよび構成情報を DHCP クライアントに提供します。

構文

システム・リソース・コントローラー (SRC) を使用して DHCP クライアントに情報を提供する

startsrc -s dhcpsd [**-a** *Argument*] [**-a** *Argument*] ...

SRC を使用せずに DHCP クライアントに情報を提供する

dhcpsd [**-f** *ConfigurationFile*]

説明

DHCP サーバーは動的アドレスの割り当ておよび割り当ての保守を扱います。さらに、追加構成情報の配布も行います。 **dhcpsd** デーモンはバックグラウンドで実行され、サーバー情報のデータベースを保守します。そのデータベースには、ロギング・パラメーター、IP (インターネット・プロトコル) アドレス範囲、その他の構成情報、およびアクセシビリティー情報が入っています。初期データベースは構成ファイルで指定されます。構成ファイルには、DHCP クライアントの構成を開始するためのすべてのデータが入っています。

DHCP サーバーは、提供したアドレスとその持ち主のデータベースを保守します。これらのデータベースは、`/etc/dhcpsd.ar` および `/etc/dhcpsd.cr` ファイルで保持されます。始動時に、サーバーは構成ファイルを読み取って、使用可能アドレスの初期データベースをセットアップします。サーバーは、**refresh** コマンドまたは **SIGHUP** シグナルを受信して、構成ファイルを再読み取りします。

DHCP サーバーは `/etc/services` ファイルを読み取り、要求の受信に使用する必要があるポートを判別します。デフォルトのサービスは **dhcps** です。これは、**bootpd** デーモンが使用するポートと同じなので、実行できるデーモンは 1 つだけ (**dhcpsd** または **bootpd** のいずれか) です。 **dhcpsd** デーモンを選択する場合は、**/etc/inetd.conf** ファイルの `bootp` をコメント化してから、コマンド・ラインに `refresh -s inetd` と入力する必要があります。

注**: bootpd** デーモンが実行されている場合は、このプログラムを停止してから、デーモンを始動する必要があります。

フラグ

項目	説明
-f *ConfigurationFile*	使用する構成ファイルを指定します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

ファイル

項目	説明
/usr/sbin/dhcpsd	**dhcpsd** デーモンが入っています。
/etc/services	インターネット・サービスで使用するソケットおよびプロトコルを定義します。
/etc/inetd.conf	**inetd** デーモンにより制御されるサービスを定義します。

関連資料:

135 ページの『dhcpcd デーモン』

関連情報:

startsrc コマンド

システム・リソース・コントローラー

TCP/IP デーモン

---

dhcpsdv6 デーモン

目的

動的ホスト構成プロトコル (DHCPv6) サーバーをインプリメントします。アドレスおよび構成情報を DHCPv6 クライアントに提供します。

構文

システム・リソース・コントローラー (SRC) を使用して DHCPv6 クライアントに情報を提供する

startsrc -s dhcpsdv6 [-a *Argument*]

SRC を使用せずに DHCP クライアントに情報を提供する

dhcpsdv6 [-d] [-f *ConfigurationFile*] **[-a** *DadminPort*] **[-p** *ServerPort*]

説明

DHCPv6 サーバーは動的アドレスの割り当ておよび割り当ての保守を扱います。さらに、追加構成情報の配布も行います。 **dhcpsd** デーモンはバックグラウンドで実行され、サーバー情報のデータベースを保守します。そのデータベースには、ロギング・パラメーター、IP (インターネット・プロトコル) アドレス範囲、その他の構成情報、およびアクセシビリティー情報が入っています。初期データベースは構成ファイルで指定されます。構成ファイルには、DHCP クライアントの構成を開始するためのすべてのデータが入っています。

DHCPv6 サーバーは、提供したアドレスとその持ち主のデータベースを保守します。これらのデータベースは、/etc/dhcpv6/db_file.crbk および /etc/dhcpv6/db_file.cr ファイルで保持されます。始動時に、サーバーは構成ファイルを読み取って、使用可能アドレスの初期データベースをセットアップします。サーバーは、refresh コマンドまたは **SIGHUP** シグナルを受信して、構成ファイルを再読み取りします。

フラグ

項目	説明
-a	Dadmin ポートを指定します。デフォルトでは 942 です。
-d	デバッグ情報を表示します。
-f *ConfigurationFile*	使用する構成ファイルを指定します。デフォルトでは、構成ファイルは /etc/dhcpv6/dhcpsdv6.cnf です。
-p	着信要求を listen するためにサーバーが使用するポートを指定します。デフォルトでは 547 です。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

例

1. /usr/local に入っている構成ファイル dhcpsdv6.cnf を使用して DHCPv6 サーバーを始動するには、次のように入力します。

   ```
   startsrc -s dhcpsdv6 -a "-f /usr/local/dhcpsdv6.cnf"
   ```

位置

/usr/sbin/dhcpsdv6

ファイル

項目	説明
/usr/sbin/dhcpsdv6	**dhcpsdv6** デーモンが入っています。
/etc/dhcpv6/db_file.cr	クライアント・レコードが入っています。このファイルはサーバー・デーモンによって作成されるものであり、構成可能ではありません。
/etc/dhcpv6/db_file.crbk	クライアント・レコードが入っています。このファイルはサーバー・デーモンによって作成されるものであり、構成可能ではありません。
/etc/dhcpv6/dhcpsdv6.duid	サーバー ID ファイルが入っています。このファイルはサーバー・デーモンによって作成されるものであり、構成可能ではありません。

関連情報:

startsrc コマンド

---

diag コマンド

目的

ハードウェアの問題を判別します。

構文

diag [[**-a**] | [**-s** [**-c**]] [**-E** *days*] [**-e**] | [**-d** *Device* [**-c**] [**-v**] [**-e**] [**-A**]] | [**-B** [**-c**]] | [**-T** *taskname*] | [**-S** *testsuite*] | [**-c** **-d** *Device* **-L** **pending** | **complete**]

説明

diag コマンドは、幅広いタスクや保守援助プログラムを実行する開始点です。タスクおよび保守援助プログラムの大部分は、プラットフォーム固有のものです。以下に使用可能なタスクや保守援助プログラムを示します。

- 診断の実行
- 診断実行時オプションの表示または変更
- サービス・ヒントの表示
- 以前の診断結果の表示
- ハードウェア・エラー・レポートの表示
- ソフトウェア・プロダクト・データの表示
- 構成およびリソース・リストの表示
- ハードウェア重要プロダクト・データの表示
- リソース属性の表示
- ハードウェア重要プロダクト・データの変更
- メディアのフォーマット
- メディアの認証
- テスト・パターンの表示
- ローカル・エリア・ネットワーク・アナライザー
- リソース・リストへのリソースの追加
- リソース・リストからのリソースの削除
- SCSI バス・アナライザー
- マイクロコードのダウンロード
- ブート・リストの表示または変更
- 定期診断
- メディアのバックアップおよび復元
- ディスクの保守
- ダイヤルおよび LPFkeys の構成
- ドロワー構成の追加または削除
- カスタマイズ構成ディスケットの作成
- ディスク・ベースの診断の更新
- ISA アダプターの構成
- オペレーティング・システム・シェル・プロンプト (オンライン・サービス・モードのみ)
- マルチプロセッサー構成の表示または変更
  - 個々のプロセッサーを使用可能および使用不可にする
- BUMP 構成の表示または変更
  - 新規バイナリー・イメージによりフラッシュ EPROM を更新する
  - 診断モードを表示または変更する
  - リモート電話番号とモデム構成を表示または変更する
- 電子モード切り替えの表示または変更
- 補足メディアの処理 (スタンドアロン・モードのみ)

- 総称マイクロコードのダウンロード
- エラー・ログ分析の実行
- イーサネットに使用する保守援助プログラム
- 7135 RAIDiant アレイ保守援助プログラム
- SCSI デバイスの識別および取り外し
- SCSD テープ・ドライブ保守援助プログラム
- ESCON ビット・エラー率保守援助プログラム
- PCI RAID 物理ディスクの識別
- リング表示パワーオン・ポリシー (CHRP) の構成
- 監視ポリシー (CHRP) の構成
- リブート・ポリシーの構成 (CHRP)
- リモート保守ポリシー (CHRP) の構成
- ハードウェア管理ポリシー (CHRP) の保存または復元
- ファームウェア・デバイス・ノード情報 (CHRP) の表示
- スペア・セクターの可用性
- システムまたはサービス・プロセッサー・フラッシュ (CHRP) の更新
- システム環境センサ (CHRP) の表示
- チェック停止分析結果の表示
- アダプター内部ログの分析
- ログ修復処置
- SK-NET FDDI ファームウェアのフラッシュ
- マイクロコード・レベルの表示

System Management Interface Tool (SMIT) の **smit diag** 高速パスを使用して、このコマンドを実行できます。

フラグ

注**: diag** コマンドは、メニュー方式のプログラムなので、ほとんどのユーザーはフラグを使用する必要はありません。

項目	説明
-A	拡張モードを指定します。**-d** フラグを使用してデバイスも指定する必要があります。
-a	脱落リソースが除去されたかどうか、オフになったかどうかなどを質問することによって、ハードウェア構成の変更を処理します。脱落リソース (「M」で示される) および脱落リソース・パス (「P」で示される) は、診断リソース選択リストに組み込まれます。
-B	診断テストで、基本システムのテストを実行するように指示します。エラー・ログ分析をサポートする基本システムの領域では、エラー・ログ分析も行われます。
-c	マシンが無人で動作することを示します。(マシンからの質問は行われません。) 結果は標準出力に書き出されます。テストされるデバイスを指定するオプションのフラグも使用する必要があります (**d**、 **B**、または **s**)。
-d *Device*	診断テストを行うデバイスを指定します。
-E *Days*	エラー・ログ分析の実行の際、エラー・ログを検索するために使用する日数を指定します。このフラグは、他のフラグとともに動作します。

項目	説明
-e	エラー・ログ解析を実行します (選択されたデバイスでサポートされている場合)。テストは行われません。このフラグは、**-d** フラグとともに使用する必要があります。そうしないと、リソース選択メニューが表示されます。**-v** フラグとともに使用された場合は、 **-v** フラグが優先され、**-e** フラグは無視されます。
-S *testsuite*	以下のような、テスト対象のデバイスからなる特定のテスト・セットを示します。 1. 基本システム 2. 入出力デバイス 3. 非同期デバイス 4. グラフィック・デバイス 5. SCSI デバイス 6. 記憶デバイス 7. 共通デバイス 8. マルチメディア・デバイス
-L pending | complete	**-d** および **-c** オプションで指定されたリソースのログ修復処置。部品を交換したが、この部品がシステム内に残ることになるかどうかがまだ分からない場合は、**pending** パラメーターを使用します。部品を交換完了し、この部品がシステム内に残ることになると分かっている場合は、**complete** パラメーターを使用します。
-s	すべてのリソースに対して診断テストを実行します。
-T *taskname*	実行する特定の高速パス・タスクを指定します。以下のリストに、現行の高速パス・タスクを示します。 **format** メディア・フォーマット・タスク **certify** メディア認証タスク **download** マイクロコード・ダウンロード・タスク **disp_mcode** マイクロコード・レベル表示タスク **chkspares** スペア・セクター可用性タスク **identifyRemove** ホット・プラグ・タスク **注:** タスクは、プラットフォームおよびデバイスに従属しています。タスクによっては、システムで使用できないものもあります。
-v	システム検査モードで診断テストを行い、エラー・ログ分析は行いません。デフォルトは、デバイスをテストし、エラー・ログ分析を行う問題判別モードです。 **-e** フラグとともに使用された場合は、 **-v** フラグが優先され、**-e** フラグは無視されます。診断テストを実行するデバイスを指定するには、**-d** フラグとともに使用する必要があります。

セキュリティー

アクセス制御: root ユーザーだけがこのコマンドを実行できます。

特権制御: システム・グループ。

例

質問を行わないで、scdisk0 デバイスについて診断テストするには、次のように入力します。

```
diag -d scdisk0 -c
```

ファイル

項目	説明
/usr/sbin/diag	**diag** コマンドが入っています。

関連資料:

『diaggetrto コマンド』

150 ページの『diagsetrto コマンド』

---

diaggetrto コマンド

目的

診断ランタイム・オプションを表示します。

構文

diaggetrto [[-a] [-d] [-l] [-m] [-n] [-p] [-s]]

説明

diaggetrto コマンドは、 1 つ以上の診断ランタイム・オプションの値を表示します。 **diaggetrto** コマンドを使用して、以下のランタイム・オプションを表示することができます。

Display Diagnostic Mode Selection Menus
このオプションがオフの場合は、診断は問題判別モードでのみ実行されます。デフォルトはオンです。

Include Advanced Diagnostics
このオプションがオンの場合は、診断は、タスク選択メニューまたはコマンド・ラインから実行されるときに、拡張モードで実行されます。デフォルトはオフです。

Number of days used to search error log
このオプションは、エラー・ログ・エントリーが診断によって分析されなくなるまでの経過日数を制御するためのものです。デフォルトは 7 です。

Display Progress Indicators
このオプションがオンの場合は、進行標識をサポートする診断アプリケーションが進行標識を表示します。デフォルトはオンです。

Diagnostic Event Logging
このオプションがオンの場合は、診断アプリケーションがイベントをログします。デフォルトはオンです。

Diagnostic Event Log file size
このオプションは、診断イベント・ログの最大サイズを制御するためのものです。許容サイズの増分は、100 キロバイト単位で行われます。デフォルトは 100K です。

フラグ

項目	説明
-a	**Include Advanced Diagnostics** の値を表示します。
-d	**Diagnostic Event Logging** の値を表示します。
-l	**Diagnostic Event Log file size** の値を表示します。
-m	**Display Diagnostic Mode Selection Menus** の値を表示します。
-n	**Number of days used to search error log** の値を表示します。
-p	**Display Progress Indicators** の値を表示します。
-s	すべての診断ランタイム・オプションを表示します。

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

例

1. 診断イベント・ログ・サイズを表示するには、次のように入力します。

   ```
   /usr/lpp/diagnostics/bin/diaggetrto -l
   ```

2. 進行標識がオンになっているかどうかを検査し、かつ診断イベント・ロギングがオンになっているかどうかを検査するには、次のように入力します。

   ```
   /usr/lpp/diagnostics/bin/diaggetrto -p -d
   ```

3. エラー・ログを検索する日数を表示するには、次のように入力します。

   ```
   /usr/lpp/diagnostics/bin/diaggetrto -n
   ```

ファイル

項目	説明
/usr/lpp/diagnostics/bin/diaggetrto	**diagsetrto** コマンドが入っています。

関連資料:

150 ページの『diagsetrto コマンド』

144 ページの『diag コマンド』

---

diagrpt コマンド

目的

前の診断結果を表示します。

構文

diagrpt [[**-o**] | [**-s** *mmddyy*] | [**-a**] | [**-r**]]

説明

diagrpt コマンドは、前の診断セッションの結果を表示します。表示できる結果には、3 つのタイプがあります。

- **/etc/lpp/diagnostic/data** ディレクトリーに保管された診断結果ファイル。
- 診断イベント・ログ情報。
- CHRP システム上の NVRAM に保管された診断結果。

フラグ

項目	説明
-o	**/etc/lpp/diagnostics/data** ディレクトリーに保管された最後の診断結果ファイルを表示します。
-s *mmddyy*	指定された日以降に記録されたすべての診断結果ファイルを表示します。
-a	診断イベント・ログのロング・バージョンを表示します。
-r	診断イベント・ログのショート・バージョンを表示します。

例

1. 1999 年 1 月 31 日以降のすべての診断結果ファイルをリストするには、以下のように入力します。

   ```
   /usr/lpp/diagnostics/bin/diagrpt -s 013199
   ```

2. 診断イベント・ログのショート・バージョンを表示するには、以下のように入力します。

   ```
   /usr/lpp/diagnostics/bin/diagrpt -r
   ```

ファイル

項目	説明
/usr/lpp/diagnostics/bin/diagrpt	**diagrpt** コマンドが入っています。

関連資料:

144 ページの『diag コマンド』

---

diagsetrto コマンド

目的

診断ランタイム・オプションを設定します。

構文

diagsetrto [[**-a on** | **off**] [**-d on** | **off**] [**-l** *Size*] [**-m on** | **off**] [**-n** *Days*] [**-p on** | **off**]]

説明

diagsetrto コマンドは、任意の数の診断ランタイム・オプションの値を設定するために使用します。
diagsetrto コマンドを使用して、以下のランタイム・オプションを変更することができます。

Display Diagnostic Mode Selection Menus
このオプションがオフの場合は、診断は問題判別モードでのみ実行されます。デフォルトはオンです。

Include Advanced Diagnostics
このオプションがオンの場合は、診断は、タスク選択メニューまたはコマンド・ラインから実行されるときに、拡張モードで実行されます。デフォルトはオフです。

Number of Days Used to Search Error Log
このオプションは、エラー・ログ・エントリーが診断によって分析されなくなるまでの経過日数を制御するためのものです。デフォルトは 7 です。

Display Progress Indicators
このオプションがオンの場合は、進行標識をサポートする診断アプリケーションが進行標識を表示します。デフォルトはオンです。

Diagnostic Event Logging
このオプションがオンの場合は、診断アプリケーションがイベントをログします。デフォルトはオンです。

Diagnostic Event Log File Size
このオプションは、診断イベント・ログの最大サイズを制御するためのものです。許容サイズの増分は、100 キロバイト単位で行われます。デフォルトは 100K です。

フラグ

項目	説明
-a on | off	**Include Advanced Diagnostics** の値を設定します。
-d on | off	**Diagnostic Event Logging** の値を設定します。
-l *Size*	**Diagnostic Event Log file size** の値を設定します。
-m on | off	**Display Diagnostic Mode Selection Menus** の値を設定します。
-n *Days*	**Number of days used to search error log** の値を設定します。
-p on | off	**Display Progress Indicators** の値を設定します。

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

例

1. 診断イベント・ログ・サイズを 500K に設定するには、次のように入力します。
   ```
   /usr/lpp/diagnostics/bin/diagsetrto -l 500
   ```
2. 進行標識をオフにし、かつ診断イベント・ロギングをオフにするには、次のように入力します。
   ```
   /usr/lpp/diagnostics/bin/diagsetrto -p off -d off
   ```
3. エラー・ログを検索する日数を 50 に設定するには、次のように入力します。
   ```
   /usr/lpp/diagnostics/bin/diagsetrto -n 50
   ```

ファイル

項目	説明
/usr/lpp/diagnostics/bin/diagsetrto	**diagsetrto** コマンドが入っています。

関連資料:
148 ページの『diaggetrto コマンド』
144 ページの『diag コマンド』

---

diction コマンド

目的

不明瞭または冗長な文を強調表示します。

構文

diction [**-ml**] [**-mm**] [**-f** *PatternFile*] [**-n**] *File* ...

説明

diction コマンドは、英語のドキュメントで不明瞭または冗長表現のデータベースの語、句を含むすべての文を検索します。それぞれの句は [ ] (大括弧) で囲まれます。**diction** コマンドは、テキストを調べる前に **deroff** コマンドを実行するので、該当するフォーマット設定情報を含むヘッダー・ファイルを入力の一部として取り込みます。 **explain** コマンドは、 **diction** コマンドで検索される語句について対話式シソーラスを提供します。

標準フォーマット設定以外のマクロを使用すると、文の切れ目が不正確になることがあります。特に、**diction** コマンドは **-me** フラグを理解しません。

フラグ

項目	説明
-f *PatternFile*	不明な語法の例が入っているファイルを指定します。デフォルト・ファイルの他に、このファイルが使用されます。
-ml	**deroff** コマンドに、 **mm** マクロ・リストをスキップさせるようにします。ドキュメントに文のフラグメントのリストが数多く含まれている場合に使用できます。
-mm	デフォルトの **ms** マクロ・パッケージを指定変更します。
-n	**-f** フラグと一緒に使用している場合、デフォルトのファイルの使用を抑止します。*PatternFile* パラメーターで指定されるファイルのみが使用されます。

ファイル

項目	説明
/usr/lib/dict.d	デフォルトのパターンが入っています。

関連資料:

109 ページの『deroff コマンド』

461 ページの『explain コマンド』

関連情報:

troff コマンド

---

diff コマンド

目的

テキスト・ファイルを比較します。

構文

2 つのファイルの内容を比較する

diff [**-c**| **-C** *Lines* | **-D** [*String*] | **-e** | **-f** | **-n** | **-u** | **-U** *Lines*] [**-b**] [**-i**] [**-t**] [**-w**] *File1 File2*

diff [**-h**] [**-b**] *File1 File2*

ディレクトリーの内容をソートし、相違点のあるファイルを比較する

diff [**-c** | **-C** *Lines* | **-e** | **-f** | **-n** | **-u** | **-U** *Lines*] [**-b**] [**-i**] [**-l**] [**-r**] [**-s**] [**-S** *File*] [**-t**] [**-w**] *Directory1 Directory2*

diff [**-h**] [**-b**] *Directory1 Directory2*

説明

diff コマンドはテキスト・ファイルを比較します。このコマンドは、ファイルまたはディレクトリーの内容を比較することができます。

注**: diff** コマンドは、入力ファイルがテキスト・ファイルである場合のみ機能します。

Directory1 と *Directory2* の両パラメーターを指定すると、**diff** コマンドは両方のディレクトリー内の同じ名前を持つテキスト・ファイルを比較します。異なっているバイナリー・ファイル、共通サブディレクトリー、およびどちらか 1 つのディレクトリーのみにあるファイルが出力されます。

diff コマンドが通常のファイルに対して実行されている場合で、ディレクトリーの比較の際に異なるテキスト・ファイルを比較した場合、**diff** コマンドは両者が一致するには、ファイル内のどの行を変更しなければならないかを知らせます。*File1* パラメーターまたは *File2* パラメーターのどちらもディレクトリーでない場合は、どちらかに - (ハイフン) を指定して、標準入力を使用することができます。*File1* パラメーターがディレクトリーである場合、そのディレクトリー内では、*File2* パラメーターと同じファイル名のファイルが使用されます。

典型的な出力には、次のようなフォーマットの行が含まれます。

```
Lines Affected in File1   Action       Lines Affected in File2
Number1                   a            Number2[,Number3]
Number1[,Number2]         d            Number3
Number1[,Number2]         c            Number3[,Number4]
```

上記の行は、*File1* を *File2* に変換するための **ed** サブコマンドに似ています。アクション文字の前の番号は *File1* に属し、アクション文字の後の番号は *File2* に属します。したがって、**a** を **d** と入れ換えて右から左へ読むと、*File2* を *File1* に変換する方法も分かります。 **ed** コマンドのように、同一の組み合わせ (この場合は *Number1* = *Number2*) は、1 つの番号として略記されます。

diff コマンドは、上記の各行に続けて、先頭に **<:** (より小の記号とコロン) を付けて、最初のファイル内で影響を受ける行すべてを表示し、その次に 2 番目のファイル内で影響を受ける行を、先頭に **>** (より大) を付けて表示します。

終了値 0 は相違がなかったことを、1 は相違が見つかったことを意味し、2 はエラーを意味します。

注**: -c**、**-C**、**-D**、**-e**、**-f**、**-n**、**-u**、または **-U** のフラグのうち 2 つ以上を指定した場合は、コマンド・ラインの最後のフラグが優先されます。エラー・メッセージは表示されません。

フラグ

項目	説明
-b	行の終わりのホワイト・スペースはその長さに関係なく単一改行文字として処理され (改行文字の前にあるホワイト・スペースは無視されます)、改行文字を含まないホワイト・スペース文字の他の文字列は等しく比較されるようにします。
-C *Lines*	*Lines* 変数で指定された値と等しいコピー済みコンテキストの行数との **diff** コマンドによる比較を生成します。**-C** フラグは出力を若干変更します。出力は、関係するファイルの ID とファイルの作成日で始まります。各変更は、12 個の * (アスタリスク) からなる 1 つの行で分離されます。*File1* から除去された行には、- (負符号) のマークが付けられ、*File2* に追加された行には、+ (正符号) のマークが付けられます。一方のファイルからもう一方のファイルに変更された行は、両方のファイルで ! (感嘆符) が表示されます。相互の指定されたコピー済みコンテキスト行内にある変更については、グループ化されて出力されます。

項目	説明
-c	3 行のコピー済みコンテキストとの **diff** コマンドによる比較を生成します。 **-c** フラグは出力を若干変更します。出力は、関係するファイルの ID とファイルの作成日で始まります。各変更は、12 個の * (アスタリスク) からなる 1 つの行で分離されます。*File1* から除去された行には、- (負符号) のマークが付けられ、*File2* に追加された行には、+ (正符号) のマークが付けられます。一方のファイルからもう一方のファイルに変更された行は、両方のファイルで ! (感嘆符) が表示されます。相互の指定されたコピーのコンテキスト行内での変更内容は、グループ化されて出力されます。
-D [*String*]	**diff** コマンドに、標準出力上で *File1* と *File2* を組み合わせたバージョンを作成させます。 C プリプロセッサー制御が含まれ、これにより *String* を定義しないで結果をコンパイルすることは *File1* をコンパイルすることと等価になります。*String* を定義すると *File2* になります。
-e	**ed** エディターを使って *File1* を *File2* に変換するのに適したフォーマットで出力を行います。このフラグを使用すると、次のシェル・プログラムによって、ファイルの複数のバージョンの保守を簡易化します。必要なのは、発生源ファイル (**$1**) と、**diff** コマンドによって作成された一連のバージョン間 **ed** スクリプト (**$2**, **$3**, ...) だけです。最新バージョンは標準出力に表示されます。 `(shift; cat $*; echo '1,$p') | ed - $1` **-e** フラグがディレクトリーの比較に使用される場合は、追加のコマンドが出力に追加されます。したがって、結果は、2 つのディレクトリーに共通するテキスト・ファイルを *Directory1* での状態から *Directory2* での状態へ変換するためのシェル・スクリプトになります。 **注: -e** または **-f** フラグによって作成されたスクリプトを編集しても、 . (ピリオド) 1 つだけの行は作成できません。
-f	**ed** エディターの使用に適さないフォーマットで出力を行わせ、**-e** フラグ指定時の出力とは逆の順序で、*File1* を *File2* に変換するのに必要な変更を表示します。
-h	変更されたセクションが短く、正しく区切られている場合に、より速くなる可能性がある代替比較を実行します。**-h** フラグはどのような長さのファイルでも動作します。**-c**、**-C**、**-D**、**-e**、**-f**、および **-n** の各フラグは **-h** フラグと一緒には使用できません。**-b** フラグを除く他のフラグはすべて、**-h** フラグと一緒に使用した場合、無視されます。
-i	大文字と小文字の区別を無視します。例えば、小文字の **a** は、大文字の **A** と同じものとして扱われます。
-l	長い出力フォーマット。**diff** コマンドのテキスト・ファイルの比較結果はそれぞれ、**pr** コマンドにパイプ接続され、伝搬されます。テキスト・ファイルの相違がすべて報告されてから、他の相違が記憶され、要約されます。
-n	**-e** フラグの出力と類似した出力を生成しますが、順序は逆で、個々の挿入または削除コマンドで変更された行の数を伴います。これは、リビジョン・コントロール・システム (RCS) によって使用されるフォーマットです。
-r	よく出てくるサブディレクトリーに **diff** コマンドを再帰的に適用します。
-s	ファイルが同じである場合は報告します。同じでない場合は、報告しません。
-S [*File*]	ディレクトリーの比較を行うときに、*File* 変数によって指定されたファイルの前にあるファイル名を無視します。 **-S** フラグは *Directory1* パラメーターと *Directory2* パラメーターで指定されたディレクトリーだけに適用されます。 **-r** フラグを **-S** フラグと一緒に指定すると、**-S** フラグは、*Directory1* サブディレクトリーと *Directory2* サブディレクトリーで再帰的に機能しません。
-t	出力行のタブを拡張します。典型的な出力または **-c** フラグ出力は、各行の先頭に文字を追加します。これは、元のソース行のインデントに影響を与える場合があり、出力リストの解釈を難しくする場合があります。このフラグは元のソースのインデントを保持します。
-u	3 行の統合されたコンテキストとの **diff** コマンドによる比較を生成します。 この出力は **-c** フラグの出力と似ていますが、コンテキスト行は繰り返されません。その代わりに、コンテキスト行、削除された行、および追加された行が混ざり合って表示されます。
-U *Lines*	*Lines* 変数で指定された値と等しい統合されたコンテキストの行数との **diff** コマンドによる比較を生成します。この出力は **-C** フラグの出力と似ていますが、コンテキスト行は繰り返されません。その代わりに、コンテキスト行、削除された行、および追加された行が混ざり合って表示されます。
-w	スペースとタブ文字をすべて無視します。その他のブランク文字列もすべてブランク文字がないものと同様に扱います。例えば、`if ( a == b )` は、 `if(a==b)` と同じです。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	相違は見つかりませんでした。
1	相違が見つかりました。
>1	エラーが発生しました。

例

1. 2 つのファイルを比較するには、次のように入力します。

   ```
   diff chap1.back chap1
   ```

   これによって、chap1.bak ファイルと chap1 ファイルとの相違が表示されます。
2. ホワイト・スペースの量の違いを無視して 2 つのファイルを比較するには、次のように入力します。

   ```
   diff  -w prog.c.bak prog.c
   ```

   2 つの行の相違がワード間のスペースとタブの数だけである場合、**diff -w** コマンドはその 2 つの行を同じであると判断します。
3. **ed** コマンドが、あるファイルを別のファイルから再構成するのに使用できるコマンドの入ったファイルを作成するには、次のように入力します。

   ```
   diff  -e chap2 chap2.old >new.to.old.ed
   ```

   これによって new.to.old.ed という名前のファイルが作成されます。このファイルには、chap2 を、chap2.old でのテキストのバージョンに戻す、 **ed** サブコマンドが入っています。ほとんどの場合、new.to.old.ed は、 chap2.old よりもはるかに小さいファイルです。chap2.old を削除すると、ディスク・スペースを節約することができます。次のコマンドを入力すると、いつでも chap2.old を再構成することができます。

   ```
   (cat new.to.old.ed ; echo '1,$p') | ed - chap2 >chap2.old
   ```

   括弧内のコマンドは、**ed** エディターに送られる編集コマンドの終わりに、 1,$p を追加します。1,$p によって、**ed** コマンドは編集後に、標準出力にファイルを書き出すようになります。そして、この変更されたコマンド・シーケンスは **ed** コマンド (| ed) に送られ、エディターはそれを標準入力として読み取ります。**-** フラグは、 **ed** コマンドがファイル・サイズおよびその他の追加情報を表示しないようにします (chap2.old のテキストと混合されてしまうため)。

ファイル

項目	説明
/usr/bin/diff	**diff** コマンドが入っています。

関連情報:

bdiff コマンド

cmp コマンド

pr コマンド

入出力リダイレクト

diff3 コマンド

目的

3 つのファイルを比較します。

構文

diff3 [-e | -x | -E | -X | -3] *File1 File2 File3*

説明

diff3 コマンドは 3 つのファイルを比較して、異なるテキストの範囲を次のコードのフラグを付けて標準出力に書き出します。

項目	説明
====	3 つのファイルがすべて異なります。
====1	*File1* が異なります。
====2	*File2* が異なります。
====3	*File3* が異なります。

あるファイルのある範囲を、別のファイルに適合するように変換する場合に必要な変更の種類は、次の 2 つの出力方法のどちらかで示されます。

項目	説明
File:*Number1* **a**	テキストは *File* 内の行番号 *Number1* の後に追加されます。ここで、*File* は **1**、**2**、または **3** です。
File:*Number1*[,*Number2*]**c**	行 *Number1* から行 *Number2* までの範囲内のテキストが変更されます。*Number1* と *Number2* が同じ場合は、範囲は *Number1* と省略できます。

範囲の内容は、**c** の表示後に表示されます。 2 つのファイルの内容が同一の場合、 **diff3** コマンドは、それぞれの同一行の位置は表示しますが、付けられている番号が小さい方のファイルの内容は表示しません。

注**: -e** フラグによって生成された編集スクリプトは、. (ピリオド) から成る行を作成できません。

フラグ

項目	説明
-3	====3 のフラグが付いた変更だけを組み込むための編集スクリプトを作成します。
-E、 **-X**	これらのフラグはそれぞれ、**-e**、および **-x** フラグに似ていますが、重複している変更 (つまり、通常のリストで ==== のフラグが付く変更) の処理の仕方が異なります。両方のファイルの重複している行が、編集スクリプトによって挿入され、<<<<<< という行と >>>>>> という行で囲まれます。**-E** オプションは、リビジョン・コントロール・システム (RCS) マージによって使用され、マージされたファイル内で重複している変更を必ず保存して、ユーザーに知らせます。
-e	*File2* と *File3* 間のすべての変更 (つまり、通常 ==== および ==== 3 のフラグが付けられた変更) を *File1* に組み込むために、**ed** コマンドと一緒に使用する編集スクリプトを作成します。
-x	==== のフラグが付いた変更だけを組み込むための編集スクリプトを生成します。

例

3 つのファイル間の相違を表示するには、次のように入力します。

```
diff3 fruit.a fruit.b fruit.c
```

`fruit.a`、`fruit.b`、および `fruit.c` に、以下のデータが入っていると、

```
fruit.a                 fruit.b                 fruit.c
banana                  apple                   grape
grape                   banana                  grapefruit
kiwi                    grapefruit              kiwi
lemon                   kiwi                    lemon
mango                   orange                  mango
orange                  peach                   orange
peach                   pear                    peach
pare
```

diff3 コマンドからの出力は、これらのファイル間の相違を次のように示します (右側のコメントは出力されません)。

```
====            All three files are different.
1:1,2c          Lines 1 and 2 of the first file, fruit.a
  banana
  grape
2:1,3c          Lines 1 through 3 of fruit.b
  apple
  banana
  grapefruit
3:1,2c          Lines 1 and 2 of fruit.c
  grape
  grapefruit
====2           The second file, fruit.b, is different.
1:4,5c          Lines 4 and 5 the same in fruit.a and fruit.c.
2:4a            To make fruit.b look same, add after line 4.
3:4,5c
  lemon
  mango
====            The first file, fruit.a, is different.
1:8c
  pare
2:7c            fruit.b line 7 and fruit.c line 8 are the same
  pear
3:7a
```

ファイル

項目	説明
/usr/bin/diff3	**diff3** コマンドを示します。
/usr/lbin/diff3prog	**diff3** シェル・スクリプトによって呼び出されます。

関連資料:

299 ページの『ed または red コマンド』

関連情報:

Files コマンド

入出力リダイレクト

---

diffmk コマンド

目的

ファイル間の相違にマークを付けます。

構文

diffmk [{ **-ab***X* | **-ae***X* } [**-b**] [**-cb***X* | **-ce***X*] [**-db***X* | **-de***X*] *File1 File2* [*File3*]

説明

diffmk コマンドは、*File1* パラメーターで指定された英語のファイルを、 *File2* パラメーターで指定された英語のファイルと比較します。 diffmk コマンドは、その後 **nroff** コマンドと **troff** コマンドに対する **.mc** 要求 (変更マークの作成要求) が記述された別のファイルを作成します。*File1* パラメーターと *File2* パラメーターは、それぞれファイルの古いバージョンと新しいバージョンを指定します。 **diffmk** コマンドは、新しく作成されたファイルを、指定があれば *File3* パラメーターに書き込み、指定がなければ標準出力に書き出します。 *File3* ファイルには、*File2* ファイルの各行と挿入されたフォーマッター **.mc** 要求の行が入っています。*File3* ファイルがフォーマットされると、変更または挿入されたテキストの各行の右マージンに | (縦線) のマークが付けられます。マージンの * (アスタリスク) は、行が削除されたことを示します。

DIFFMARK 環境変数を定義すると、**diffmk** コマンドでファイルの比較に使用されるコマンド文字列が指定されます。 (通常、**diffmk** コマンドでは、**diff** コマンドを使用します。) 例えば、非常に大きなファイルをより適切に処理するために、**DIFFMARK** 変数を `diff -h` に設定することができます。

パラメーター

項目	説明
File1	*File2* パラメーターによって指定されたファイルと比較される英語ファイルを指定します。比較の結果から、*File3* パラメーターによって指定されたファイルが構成されます。*File1* は「古い」ファイルと見なされます。
File2	*File1* パラメーターによって指定されたファイルと比較される英語ファイルを指定します。比較の結果から、*File3* パラメーターによって指定されたファイルが構成されます。 *File2* は「新しい」ファイルと見なされます。
File3	*File2* ファイルの行を含み、**nroff** および **troff** コマンドの場合の挿入されたフォーマッター **.mc** 要求の行を格納するファイルを指定します。このファイルの内容は、*File1* パラメーターと *File2* パラメーターによって指定されたファイルの比較結果です。変更されたテキストは、フォーマットされると、その各行の右マージンに	(縦線) のマークが付けられます。* (アスタリスク) は行が削除されたことを示します。 *File3* が指定されない場合、比較の結果は標準出力に書き出されます。

フラグ

項目	説明
-ab*X*	追加された行が始まる位置に *X* というマークを付けます。
-ae*X*	追加された行が終る位置に *X* というマークを付けます。
-b	行上でタブまたはスペースが変更されただけの相違を無視します。
-cb*X*	変更された行が始まる位置に *X* というマークを付けます。
-ce*X*	変更された行が終る位置に *X* というマークを付けます。
-db*X*	削除された行が始まる位置に *X* というマークを付けます。
-de*X*	削除された行が終る位置に *X* というマークを付けます。

例

1. テキスト・ファイルの 2 つのバージョン間の相違にマークを付けるには、次のように入力します。

   ```
   diffmk chap1.old chap1 chap1.nroff
   ```

   これによって、**nroff** および **troff** 変更マーク要求を含む `chap1` のコピーが作成され、`chap1.old` に追加されたり、`chap1.old` で変更されたり、`chap1.old` から削除されたりしたテキストが識別されます。このコピーは、`chap1.nroff` ファイルに保管されます。

2. **nroff** 以外のメッセージと、 **troff** メッセージの相違点にマークを付けるには、次のように入力します。

   ```
   diffmk -ab'>>New:' -ae'<<End New' ¥
   chap1.old chap1 chap1.nroff
   ```

これによって、**diffmk** コマンドは `>>New:` を、新たに `chap1,` に追加された選択行の前に書き込み、追加された行の後の行に `<<End New` を書き込みます。変更と削除の場合は、**nroff/troff** コマンドが `|` (縦線) または `*` (アスタリスク) をマージンに付けます。

3. 別の **nroff/troff** コマンドのマーキング要求を使用して、ホワイト・スペースの変更を無視するには、次のように入力します。

   ```
   diffmk -b -cb'.mc %' chap1.old chap1 chap1.nroff
   ```

   これによって、変更に `%` (パーセント記号)、追加に `|` (縦線)、削除に `*` (アスタリスク) を付けるコマンドが組み込まれます。ワード間のスペースとタブの数が異なるだけの変更には、マークを付けません (`-b`)。

関連資料:

152 ページの『diff コマンド』

関連情報:

nroff コマンド

troff コマンド

---

dig コマンド

目的

DNS 検索ユーティリティー。

構文

dig [*@server*] [**-b** *address*] [**-c** *class*] [**-f** *filename*] [**-k** *filename*] [**-p** *port#*] [**-q** *name*] [**-t** *type*] [**-x** *addr*] [**-y** [hmac:] *name:key*] [**-4**] [**-6**] [*name*] [*type*] [*class*] [*queryopt...*]

dig [**-h**]

dig [*global-queryopt...*] [*query...*]

説明

dig (domain information groper) コマンドは、 DNS ネームサーバー問い合わせのための柔軟なツールです。これは DNS 検索を実行し、照会したネームサーバーから返された応答を表示します。多くの DNS 管理者は、**dig** コマンドを、その柔軟性、使いやすさ、および出力の明確さのため、DNS に関する問題のトラブルシューティングに使用します。**dig** は、通常コマンド・ラインの引数とともに使用されますが、ファイルからの検索リクエストを読み取るためのバッチ・モードもあります。以前のバージョンとは異なり、**dig** の BIND9 インプリメンテーションではコマンド・ラインからの複数の検索の発行を許可します。特定のネームサーバーへの照会が指定されない限り、**dig** コマンドは **/etc/resolv.conf** ファイル内にリストされている各サーバーへの照会を試みます。コマンド・ラインで引数またはオプションが指定されない場合は、**dig** コマンドは **"."** (root) の NS 照会を実行します。

${HOME}/.digrc ファイルを使用して **dig** コマンドにユーザーごとのデフォルトを設定できます。**dig** コマンドは、このファイルを読み取り、コマンド・ライン引数の前にオプションを適用します。

IN と **CH** のクラス名は、**IN** と **CH** の最上位のドメイン名とオーバーラップします。これらの最上位のドメインを検索する場合は、**-t** オプションと **-c** オプションを使用してタイプとクラスを指定するか、**-q** オプションを使用してドメイン名を指定するか、または **IN** と **CH** の名前を使用します。

フラグ

項目	説明
-b *address*	照会先アドレスの送信元 IP アドレスを設定します。これは、ホストのネットワーク・インターフェース、あるいは「0.0.0.0」または「::」のいずれかで有効なアドレスでなければなりません。「*#port*」を追加すると、オプションのポートを指定できます。
-c *class*	デフォルトの照会クラス (インターネットの IN) を指定変更します。***class*** パラメーター値は、Hesiod レコードの HS や CHAOSNET レコードの CH などのように、任意の有効なクラスです。
-f *filename*	処理する検索要求のリストを指定のファイル名から読み取ることにより、**dig** コマンドをバッチ・モードで操作します。このファイルには、多数の照会が、1 行につき 1 つずつ含まれています。ファイル内の各エントリーは、コマンド・ライン・インターフェースを使用した **dig** コマンドへの照会と同じ提示方法で構成されている必要があります。
-h	コマンド・ライン引数とオプションの要約を出力します。
-k *filename*	**-k** オプションを使用して TSIG キー・ファイルを指定し、**dig** コマンドによって送信された DNS 照会に署名します。
-p *port#*	非標準ポート番号を照会します。*port#* パラメーター値は、**dig** コマンドがその照会を標準の DNS ポート番号 53 の代わりに送信するポート番号です。このオプションは、非標準のポート番号での照会を listen するように構成されているネームサーバーをテストするために使用できます。
-q *name*	名前と他の引数を区別します。照会名を指定の *name* パラメーター値に設定します。
-t *type*	照会タイプを ***type*** パラメーター値に設定します。これには、BIND9 でサポートされている任意の有効な照会タイプを指定できます。逆検索を示す **-x** オプションが指定されない限り、デフォルトの照会タイプは **A** です。***type*** に AXFR を指定すると、ゾーン転送を要求できます。増分ゾーン転送 (IXFR) が必要な場合は、***type*** パラメーター値は **ixfr=N** に設定されます。増分ゾーン転送には、そのゾーンの SOA レコード内のシリアル番号が **N** であったため、そのゾーンに対して行われた変更が含まれます。
-x *addr*	逆検索 (アドレスを名前にマッピング) を単純化します。***addr*** パラメーター値は、小数点表記法による IPv4 アドレス、またはコロンで区切られた IPv6 アドレスです。このオプションを使用する場合、*name*、*class*、および *type* 引数を指定する必要はありません。**dig** コマンドは、自動的に `11.12.13.10.in-addr.arpa` のように名前を検索し、照会タイプとクラスをそれぞれ PTR と IN に設定します。
-y [hmac:] *name:key*	コマンド・ラインで TSIG キー自体を指定します。*hmac* は TSIG のタイプです。デフォルト値は HMAC-MD5 です。 ***name*** パラメーター値は TSIG キーの名前、***key*** パラメーター値は実際のキーです。キーは base-64 でエンコードされた文字列で、通常 **dnssec-keygen**(8) によって生成されます。**-y** オプションをマルチユーザー・システム上で使用する場合は、キーが ps(1) からの出力やシェルのヒストリー・ファイル内で見えるため、注意が必要です。**dig** コマンドで TSIG 認証を使用する場合、照会されるネームサーバーは使用されるキーとアルゴリズムを認識する必要があります。BIND では、これは該当するキーとサーバーのステートメントを **named.conf** ファイルに提供することによって行われます。
-4	**dig** コマンドで IPv4 照会のトランスポートのみを使用するようにします。
-6	**dig** コマンドで IPv6 照会のトランスポートのみを使用するようにします。

パラメーター

項目	説明
global-queryopt...	グローバル照会オプション (『複数の照会』を参照)。
query	照会オプション (『照会のオプション』を参照)。

照会のオプション

dig コマンドは、検索方法および結果の表示に影響する、数多くの照会オプションを提供します。これらのオプションの中には、照会ヘッダー内でフラグ・ビットを設定やリセットするもの、応答のどのセクションを印刷するかを決めるもの、タイムアウトや再試行の方針を決めるものなどがあります。各照会オプションは、前に正符号 (+) が付いたキーワードで識別されます。キーワードの中には、オプションを設定またはリセットするものがあります。これらには、そのキーワードの意味を否定するための文字列 `no` が前に付きます。また、キーワードの中には、タイムアウト間隔などの値をオプションに割り当てるものもあります。これらは、 **+keyword=value** の形式をとります。照会のオプションは、次のとおりです。

+[no]tcp

ネームサーバーを照会する際に、TCP を使用するか、または使用しないかを設定します。デフォルトの動作では、 AXFR または IXFR の照会が要求 (その場合は TCP 接続が使用される) されない限り、UDP が使用されます。

+[no]vc

ネームサーバーを照会する際に、TCP を使用するか、または使用しないかを設定します。**+[no]tcp** に対するこの代替構文は、下位互換性のために提供されています。**vc** は、バーチャル・サーキットを表します。

+[no]ignore

TCP での再試行を行わず、UDP 応答における切り捨てを無視します。デフォルトでは、 TCP での再試行が実行されます。

+domain=somename

/etc/resolv.conf ファイル内のドメイン指示で指定されているかのように検索リストが単一ドメイン *somename* を含むように設定し、**+search** オプションが指定されているかのように検索リストの処理を可能にします。

+[no]search

/etc/resolv.conf ファイル内の検索リストまたはドメイン指示 (存在する場合) で定義された検索リストを使用するか、または使用しないかを設定します。デフォルトでは、検索リストは使用されません。

+[no]defname

非推奨、**+[no]search** と同義語として扱われます。

+[no]aaonly

照会内に「aa」フラグを設定します。

+[no]adflag

照会内に AD (認証データ) ビットを設定するか、または設定しないかを指定します。現在のところ、AD ビットは応答においてのみ標準の意味を持ち、照会においては標準の意味を持ちませんが、完全性のために照会内にこのビットを設定する機能が提供されています。

+[no]cdflag

照会内に CD (checking disabled: 検査は使用不可) ビットを設定するか、設定しません。これは、応答の DNSSEC 検証を行わないようにサーバーに要求します。

+[no]cl

レコードを出力する際に、CLASS を表示するか、表示しません。

+[no]ttlid

レコードを出力する際に、TTL を表示するか、表示しません。

+[no]recursive

照会内で RD (recursion desired: 再帰要求) ビットの設定をトグルします。このビットはデフォルトで設定されており、通常 **dig** は再帰的照会を送信します。**+nssearch** または **+trace** 照会オプションが使用される場合は、再帰は自動的に使用不可になります。

+[no]nssearch

このオプションが設定されると、**dig** コマンドは、検索されている名前を含むゾーンに対して権限ネームサーバーの検索を試行し、そのゾーンについて各ネームサーバーが持つ SOA レコードを表示します。

+[no]trace
検索される名前についての root ネームサーバーからの代行パスのトレースをトグルします。トレースは、デフォルトで使用不可になっています。トレースが使用可能になると、**dig** コマンドは、検索されている名前を解決するために反復照会を行います。これは、その検索の解決に使用された各サーバーからの応答を示し、root サーバーからの委託に従います。

+[no]cmd
適用された **dig** のバージョンおよび照会オプションを示す、出力内の初期コメントの印刷をトグルします。デフォルトでは、このコメントは印刷されます。

+[no]short
簡単な応答を提供します。デフォルトでは、応答は詳細フォームで印刷されます。

+[no]identify
+short オプションが使用可能である場合に、応答を提供した IP アドレスとポート番号を表示するか、表示しません。簡略フォームでの応答が要求された場合は、デフォルトでは、応答を提供した送信元アドレスとサーバーのポート番号を表示しません。

+[no]comments
出力内のコメント行の表示を切り替えます。デフォルトでは、コメントを印刷します。

+[no]stats
統計情報 (照会が行われた際の応答のサイズなど) の印刷を切り替えします。デフォルトの動作では、照会の統計情報を印刷します。

+[no]qr
照会が送信されるたびに、照会を印刷するか、または印刷しないかを設定します。デフォルトでは、照会は印刷されません。

+[no]question
応答が返される際に、照会の質問セクションを印刷するか、または印刷しないかを設定します。デフォルトでは、質問セクションをコメントとして印刷します。

+[no]answer
応答内の応答セクションを表示するか、または表示しないかを設定します。デフォルトでは、表示します。

+[no]authority
応答内の権限セクションを表示するか、または表示しないかを設定します。デフォルトでは、表示します。

+[no]additional
応答内の追加セクションを表示するか、または表示しないかを設定します。デフォルトでは、表示します。

+[no]all
すべての表示フラグを設定するか、またはクリアするかを設定します。

+time=T
照会のタイムアウトを ***T*** 秒に設定します。デフォルトのタイムアウトは 5 秒です。***T*** パラメーター値を 1 未満に設定しようとしても、結果的には 1 秒の照会タイムアウトが適用されます。

+tries=A
サーバーへの UDP 照会を試行する回数を、デフォルトの 3 ではなく ***A*** パラメーター値に設定します。***A*** パラメーター値が 0 以下である場合は、再試行の回数は単純に 1 に切り上げられます。

+retry=T

サーバーへの UDP 照会を再試行する回数を、デフォルトの 2 ではなく ***T*** パラメーター値に設定します。**+tries** とは異なり、これには初期照会は含まれません。

+ndots=D

名前内に表示されるドットの数を ***D*** パラメーター値に設定します (絶対名と考えられるため)。デフォルト値は、**/etc/resolv.conf** ファイル内の **ndots** ステートメントを使用して定義された値か、あるいは **ndots** ステートメントが存在しない場合は 1 です。より少ないドット数を持つ名前は相対名として解釈され、検索内にリストされたドメイン内または **/etc/resolv.conf** ファイル内のドメイン指示内で検索されます。

+bufsize=B

EDNS0 を使用して公示される UDP メッセージのバッファー・サイズを ***B*** バイトに設定します。このバッファーの最大サイズおよび最小サイズは、それぞれ 65535 および 0 です。この範囲外の値は、それぞれ切り上げまたは切り下げが行われます。ゼロ以外の値では、EDNS 照会が送信されます。

+edns=#

照会での EDNS バージョンを指定します。有効な値は 0 から 255 です。EDNS バージョンを設定すると、EDNS 照会が送信されます。**+noedns** は、指定された EDNS バージョンをクリアします。

+[no]multiline

SOA レコードのようなレコードを、人間が理解できるコメントを備えた詳細な複数行のフォーマットで印刷します。デフォルトでは、**dig** 出力のマシンによる構文解析を容易にするために、各レコードを 1 行に印刷します。

+[no]fail

SERVFAIL を受信した場合に次のサーバーを試行しません。デフォルトでは、次のサーバーを試行しません。これは、通常のスタブ・リゾルバーの動作の逆です。

+[no]besteffort

誤った形式のメッセージのコンテンツを表示します。デフォルトでは、誤った形式の応答は表示しません。

+[no]dnssec

照会の追加セクションの OPT レコードで DNSSEC OK ビット (DO) を設定して、DNSSEC レコードの送信を要求します。

+[no]sigchase

DNSSEC 署名チェーンを追跡します。**dig** コマンドを -DDIG SIGCHASE でコンパイルする必要があります。

+trusted-key=####

トラステッド鍵を含むファイルを **+sigchase** で使用するように指定します。各 DNSKEY レコードは、それ自体の行になければなりません。指定されない場合、**dig** コマンドは **/etc/trusted-key.key** ファイルを検索し、次に現行ディレクトリーで **trusted-key.key** ファイルを検索します。**dig** コマンドを -DDIG SIGCHASE でコンパイルする必要があります。

+[no]topdown

DNSSEC 署名チェーンを追跡する場合に、トップダウン型の妥当性検査を実行します。 **dig** コマンドを -DDIG SIGCHASE でコンパイルする必要があります。

複数の照会

dig の BIND 9 インプリメンテーションでは、(**-f** バッチ・ファイル・オプションのサポートに加えて、) コマンド・ラインでの複数の照会の指定をサポートしています。これらの各照会で、フラグ、オプション、および照会オプションの独自のセットを指定できます。

この場合、各照会引数は、コマンド・ライン構文内の個々の照会を指します。それぞれの引数は、標準のオプションとフラグ、検索される名前、オプションの照会タイプ、クラス、およびその照会に適用しなければならない照会オプションで構成されます。

すべての照会に適用される必要がある照会オプションのグローバル・セットも指定することができます。これらのグローバル照会オプションは、コマンド・ライン上に提供される名前、クラス、タイプ、オプション、フラグ、および照会オプションの最初の組の前に置く必要があります。グローバル照会オプション (**+[no]cmd** オプションを除く) は、照会固有の照会オプション・セットによる指定変更が可能です。次に例を示します。

```
dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
```

この **dig** コマンド・ストリングは、3 つの検索 (**www.isc.org** の ANY 照会、127.0.0.1 の逆検索、および **isc.org** の NS レコードの照会) を行うためにコマンド・ラインから **dig** コマンドをどのように使用できるかを示しています。**+qr** のグローバル照会オプションが適用され、そのため **dig** コマンドは各検索について行った初期照会を示します。最後の照会には **+noqr** のローカル照会オプションがあり、これは **isc.org** の NS レコードを検索する際に **dig** コマンドが初期照会を印刷しないことを意味します。

IDN サポート

dig コマンドが国際化ドメイン名 (IDN) サポートを使用してビルドされている場合は、非 ASCII のドメイン名を受け入れて表示することができます。 **dig** コマンドは、ドメイン名の文字エンコードを適切に変換してから、DNS サーバーに要求を送信したり、サーバーからの応答を表示します。何らかの理由で IDN サポートをオフにする場合は、IDN DISABLE 環境変数を定義します。**dig** コマンドの実行時にこの変数を設定すると、次の IDN サポートは使用不可になります。

例

dig の典型的な呼び出しは次のようになります。

```
dig @server name type
```

それぞれの意味は次のとおりです。

server 照会するネームサーバーの名前または IP アドレス。これには、小数点表記法の IPv4 アドレス、またはコロン (:) で区切られた表記の IPv6 アドレスを指定できます。指定されたサーバーの引数がホスト名の場合、**dig** コマンドはそのネームサーバーを照会する前にその名前を解決します。サーバーの引数が 1 つも指定されない場合、**dig** コマンドは **/etc/resolv.conf** ファイルを参照し、そこにリストされたネームサーバーを照会します。応答するネームサーバーからの応答が表示されます。

name 検索されるリソース・レコードの名前。

type 必要な照会のタイプ (**ANY**、**A**、**MX**、**SIG** など) を示します。*type* 引数の値には、任意の有効な照会タイプを指定できます。*type* の引数が 1 つも指定されない場合、**dig** コマンドは ***A*** レコードについて検索を行います。

ファイル

項目	説明
/etc/resolv.conf	
${HOME}/.digrc	

関連資料:

794 ページの『host9 コマンド』

179 ページの『dnssec-keygen コマンド』

関連情報:

named9 コマンド

named-checkconf コマンド

---

digest コマンド

目的

ASCII フォーマットの **/etc/qconfig** ファイルを **/etc/qconfig.bin** ファイル (**qdaemon** コマンドで使用されるバイナリー・バージョンのキュー構成ファイル) に変換します。このコマンドは、コマンド・ラインに入力するのではなく、**qdaemon** コマンドによって呼び出されます。

構文

/usr/lib/lpd/digest *ASCIIFile BinaryFile*

説明

digest コマンドは ASCII 文字の入力ファイルを受け入れて、それをバイナリー・ファイルに変換します。このコマンドは、**qdaemon** コマンドによって使用され、**/etc/qconfig** ファイルをバイナリー・バージョンのファイルである **/etc/qconfig.bin** ファイルに変換します。

ファイル

項目	説明
/etc/qconfig	キュー構成ファイルが入っています。
/usr/sbin/qdaemon	キュー・デーモンが入っています。
/etc/qconfig.bin	**/etc/qconfig** ファイルの要約されたバイナリー・バージョンが入っています。

関連情報:

qdaemon コマンド

---

dircmp コマンド

目的

2 つのディレクトリーとその中にあるファイルの内容を比較します。

構文

dircmp [**-d**] [**-s**] [**-w** *num*] *Directory1 Directory2*

説明

dircmp コマンドは、*Directory1* と *Directory2* によって指定された 2 つのディレクトリーを比較して、その内容に関する情報を標準出力に書き出します。まず最初に、**dircmp** コマンドは各ディレクトリー内のファイル名を比較します。両方のディレクトリーに同一ファイル名があると、**dircmp** コマンドは両方のファイルの内容を比較します。

出力では、**dircmp** コマンドは各ディレクトリーごとに固有なファイルをリストします。そして、両方のディレクトリー内で、同じ名前ではあるが内容は異なっているファイルを出力します。フラグを指定しないと、両方のディレクトリー内の名前が同じであるだけでなく、内容も同じであるファイルも出力します。

diff -r コマンドは、 **dircmp** コマンドと同様の機能を提供します。

フラグ

項目	説明
-d	各共通ファイル名ごとに、異なるファイル内容を両方とも表示します。表示フォーマットは、**diff** コマンドと同じです。
-s	同一ファイルの名前を表示しません。
-w	出力の幅を、*num* で指定される文字数に変更します。
num	

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

注**:** ディレクトリー内容の相違は、エラーとは見なされません。

例

1. 2 つのディレクトリー内のファイル間の相違を要約するには、次のように入力します。

   ```
   dircmp proj.ver1 proj.ver2
   ```

   これによって、ディレクトリー `proj.ver1` と `proj.ver2` との間の相違の要約が表示されます。相違点をまとめたリストでは、1 つのディレクトリーにしかないファイルと 2 つのディレクトリーのどちらの中にもあるファイルは別々に示されています。ファイルが両方のディレクトリーにあると、 **dircmp** コマンドは、その 2 つのコピーが同一であるかないかに注目します。
2. ファイル間の相違を詳細に表示するには、次のように入力します。

   ```
   dircmp  -d  -s proj.ver1 proj.ver2
   ```

   -s フラグは、同一ファイルに関する情報を出しません。**-d** フラグは、両方のディレクトリー内の異なるファイルについて、**diff** リストを表示します。
3. ファイル間の相違の詳細を、90 文字に設定された出力行の幅で表示するには、次のように入力します。

   ```
   $dircmp -w 90 dir1 dir2
   ```

ファイル

項目	説明
/usr/bin/dircmp	**dircmp** コマンドが入っています。

関連情報

cmp コマンド、 **diff** コマンド。

ファイルシステム内のディレクトリーの構造と特性については、「オペレーティング・システムおよびデバイスの管理」の『ディレクトリー』を参照してください。

オペレーティング・システムによる入出力の処理については、「オペレーティング・システムおよびデバイスの管理」の『入出力リダイレクト』を参照してください。

関連資料:

152 ページの『diff コマンド』

関連情報:

cmp コマンド

Directories コマンド

入出力リダイレクト

---

dirname コマンド

目的

指定されたパスの最後の部分を除いた部分をすべて標準出力に出力します。

構文

dirname *Path*

説明

dirname コマンドは、指定されたパス名を読み取り、最後の / (スラッシュ) を除くすべてと、その後ろの文字を削除し、結果を標準出力に書き出します。最後の / の後ろに文字がなければ、**dirname** コマンドは最後の / の 1 つ手前の / を使用し、その後ろに続くすべての文字を無視します。 **dirname** コマンドでは、次の規則に従ってパス名を作成します。

1. *Path* パラメーターが // (2 つのスラッシュ) の場合、または *Path* パラメーターがスラッシュ文字のみで構成される場合は、その文字列を単一の / (スラッシュ) に変更します。ステップ 2 から 7 までをスキップします。
2. 指定されたパスから末尾の / を除去します。
3. *Path* パラメーター内に / 文字が残っていなければ、パスを単一の**.** (ピリオド) に変更します。ステップ 4 から 7 までスキップします。
4. パスから、スラッシュ以外の末尾の文字を除去します。
5. 残りのパスが // (2 つのスラッシュ) であれば、ステップ 6 に進みます。
6. パスから末尾の / 文字を除去します。
7. 残りのパスが空であれば、そのパスを単一の / に変更します。

例えば、次のように入力すると、

```
dirname //
```

結果は単一の / (スラッシュ) となります。次のように入力します。

```
dirname /a/b/
```

結果は /a となります。次のように入力します。

```
dirname a
```

結果は単一の . (ピリオド) となります。次のように入力します。

```
dirname a/b
```

結果はパス名 a となります。

一般に **dirname** コマンドと **basename** コマンドは、シェル・プロシージャー内のコマンド置換の中で、指定した入力ファイルの名前に何らかの変更を加えた出力ファイルの名前を指定する目的で使用されます。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

別のファイルと同じディレクトリーにあるファイルの名前を作成するには、次のように入力します。

```
AOUTFILE=`dirname $TEXTFILE`/a.out
```

これによって、シェル変数 AOUTFILE が、 TEXTFILE と同じディレクトリーにある、 **a.out** ファイルの名前に設定されます。 TEXTFILE が **/home/fran/prog.c** である場合は、 dirname $TEXTFILE の値は、**/home/fran** で、AOUTFILE は、 **/home/fran/a.out** になります。

ファイル

項目	説明
/usr/bin/dirname	**dirname** コマンドが入っています。

関連情報:

basename コマンド

sh コマンド

---

disable コマンド

disable コマンドには、AIX 印刷サブシステムの **disable**、および **System V** 印刷サブシステムの **disable** についての情報が組み込まれています。

目的

プリンター・キュー・デバイスを使用不可にします。

構文

disable [**-c**] [**-r***Reason*] *PrinterName ...*

説明

disable コマンドは、*PrinterName* パラメーターで指定されたプリンター・キュー・デバイスを使用不可 (オフライン) にします。

注**:** このコマンドを実行するには、root ユーザー権限を持っているか、または印刷キュー・グループに属している必要があります。

フラグ

項目	説明
-c	すべてのジョブ要求を取り消します。このフラグを使用することは、 **enq -K** コマンドを実行することと同じです。
-r*Reason*	プリンター・キュー・デバイスを使用不可にした理由を *Reason* 変数で指定します。このフラグは「ノーオペレーション」フラグなので、システムはこのフラグを無視します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 現在の印刷ジョブの完了を待たずにプリンター・キュー `lp0` をオフラインにするには、次のように入力します。

   ```
   disable  -c lp0
   ```

2. すべての印刷ジョブが完了してからプリンター・キュー `lp0` をオフラインにするには、次のように入力します。

   ```
   disable lp0
   ```

ファイル

項目	説明
/usr/sbin/qdaemon	キューイング・デーモン
/etc/qconfig	キュー構成ファイル
/etc/qconfig.bin	**/etc/qconfig** ファイルの要約バイナリー・バージョン
/var/spool/lpd/qdir/*	キュー要求
/var/spool/lpd/stat/*	デバイスの状況に関する情報
/var/spool/qdaemon/*	エンキューされたファイルの一時コピー

System V Print Subsystem の disable コマンド

目的

LP プリンターを使用不可にします。

構文

disable [*flags*] *printers*

説明

disable コマンドは、指定したプリンター を活動停止にして、**lp** が実行依頼した要求を印刷できないようにします。デフォルトでは、指定プリンターで現在印刷中の任意の要求は、同一のプリンター、または同一クラスのプリンターの別のメンバーで、そっくりそのまま再度印刷されます。プリンターがリモートである場合は、このコマンドを実行しても、ジョブのリモート・システムへの送信が停止されるだけです。プリンターを使用不可にするには、**disable** コマンドを、リモート・システム上で実行しなければなりません。(プリンターの状況を取得するには、**lpstat -p** を実行してください。)

プリンター名はシステム定義ワード なので、 ASCII 文字の大文字小文字しか使用できません。

`disable -?` と入力すると、システムはコマンド使用方法のメッセージを表示し、0 を返します。

フラグ

-c 指定されたプリンターのいずれかで現在印刷中の任意の要求を取り消します。このフラグは、**-W** フラグと一緒には使用できません。プリンターがリモートである場合、**-c** フラグは無視されます。

-r *reason*
プリンターを使用不可にする理由 を割り当てます。この理由 は、指定されたすべてのプリンターに適用されます。この理由 は、**lpstat -p** により報告されます。理由 にブランクが含まれている場合は、引用符で囲む必要があります。デフォルトの理由は、既存のプリンターでは *unknown reason* で、システムに追加されたばかりでまだ使用可能になっていないプリンターの場合は、 *new printer* です。

-W 現在印刷中の要求が終了するまで待機してから、指定されたプリンターを使用不可にします。

このフラグは、**-c** フラグと一緒には使用できません。プリンターがリモートである場合は、 **-W** フラグは無視されるだけです。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

ファイル

/var/spool/lp/*

関連資料:

382 ページの『enable コマンド』

関連情報:

cancel コマンド

lp コマンド

印刷キューの開始および停止

diskusg コマンド

目的

ユーザー ID ごとにディスク・アカウンティング・データを作成します。

構文

diskusg [**-X**] [**-U** *MaxUsers*] [**-i** *FileListName*] [**-p** *File*] [**-u** *File*] [**-v**] {
-s [*File ...*] | *FileSystem ...* }

説明

diskusg コマンドは、*File* または *FileSystem* パラメーターで指定されたファイル内のデータか、または標準入力から中間ディスク・アカウンティング情報を作成します。 **diskusg** コマンドは、1 ユーザーあたり 1 レコードを標準出力に書き出します。このコマンドは、**cron** デーモンの下で実行できる **dodisk** コマンドによって呼び出されます。出力のフォーマットは次のとおりです。

項目	説明
UID	ユーザーの数字のユーザー ID。
Login	ユーザーのログイン名。
Blocks	ユーザーに割り当てられた 512 バイトのディスク・ブロックの合計数。

このコマンドの出力は、**acctdisk** コマンドの入力になり、情報を合計アカウンティング・レコードに変換します。この合計アカウンティング・レコードは、他の合計アカウンティング・レコードとマージされて、日次レポートを作成します。

FileSystem パラメーターを指定すると、**diskusg** コマンドは指定されたファイルシステムの i ノードを読み取って、使用状況データを生成します。 *FileSystem* パラメーターはファイルシステム・デバイスのスペシャル・ファイル名でなければなりません。例えば、ルート・ファイルシステムの使用状況データを生成するには、 **/** (ルート) ディレクトリーの代わりに、 **/dev/hd4** デバイスを使用する必要があります。

File パラメーターを指定する場合は、入力が **diskusg** 出力フォーマットでなければなりません。

ディスク使用状況の詳細については、 **acctdusg** コマンドを参照してください。

注**:** このコマンドは、ローカル・デバイス以外に使用できません。

フラグ

項目	説明
-i *FileListName*	*FileListName* ファイルシステム内のデータを無視します。 *FileListName* 変数は、コンマで区切られているか、または引用符で囲まれているファイルシステム名のリストです。
-p *File*	*File* 変数で指定されたパスワード・ファイルを使用して、ログイン名を生成します。デフォルトは **/etc/passwd** ファイルです。
-s [*File*]	入力ファイルまたは標準入力のすべてのレコードを 1 つのレコードに結合します。入力データは、既に **diskusg** の出力フォーマットになっています。
-U *MaxUsers*	**diskusg** コマンドで処理できるユーザーの最大数を設定します。ユーザー数がデフォルトの 5000 よりも大きい場合にのみこのフラグが必要となります。
-u *File*	誰のユーザー ID にも課金されていない各ファイルのレコードを、指定された *File* 変数に書き込みます。それぞれのレコードは、スペシャル・ファイル名、i ノード番号、およびユーザー ID から構成されます。
-v	誰にも課金されていないすべてのファイルのリストを標準エラー出力に書き出します。

項目	説明
-X	最初の 8 文字に切り捨てることをせずに、各ユーザー名の使用可能なすべての文字を印刷して処理します。

セキュリティー

アクセス制御: このコマンドは、**adm** グループのメンバーのみに実行 (x) アクセス権限を与えます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

日次ディスク・アカウンティング情報を作成するには、次のような行を **/var/spool/cron/crontab/root** ファイルに追加します。

```
0 2 * * 4 /usr/sbin/acct/dodisk
```

このコマンドは、毎週木曜日 (4) の午前 2 時 (02) に、 **dodisk** コマンドを実行するように、 **cron** デーモンに指示します。**dodisk** コマンドは、**diskusg** と **acctdisk** の両方のコマンドを呼び出します。

注**:** この例を実行するには、root 権限を持っていなければなりません。

ファイル

項目	説明
/usr/sbin/acct/diskusg	**diskusg** コマンドが入っています。
/etc/passwd	ユーザーの基本属性が入っています。

関連情報:

acctdisk コマンド

acct サブルーチン

acct ファイル・フォーマット

システム・アカウンティング

---

dispgid コマンド

目的

すべての有効なグループ名のリストを表示します。

構文

dispgid

説明

dispgid コマンドは、システム上のすべてのグループ名のリストを (1 行につき 1 つの名前で) 表示するのに使用できます。このコマンド用のパラメーターはありません。データの検索のため、次のファイルが読み取り専用モードでアクセスされます。

- **/etc/passwd**
- **/etc/group**
- **/etc/security/user**
- **/etc/security/limits**
- **/etc/security/group**
- **/etc/security/environ**

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

例

1. マシン内のすべての有効なグループをリストするには、**dispgid** コマンドを次のように入力します。

   ```
   dispgid
   ```

   出力は、次のようになります。

   ```
   system
   staff
   bin
   sys
   adm
   uucp
   mail
   security
   cron
   printq
   audit
   ecs
   nobody
   usr
   perf
   ```

ファイル

項目	説明
/usr/sbin/dispgid	**dispgid** コマンドが入っています。
/etc/group	グループ情報が入っています。

関連資料:

『dispuid コマンド』

---

dispuid コマンド

目的

すべての有効なユーザー名のリストを表示します。

構文

dispuid

説明

このコマンドは、システム上のすべてのユーザー名のリストを (1 つの名前につき 1 行で) 表示するのに使用できます。このコマンド用のパラメーターはありません。ユーザー・データの検索のため、次のファイルが読み取り専用モードでアクセスされます。

- **/etc/passwd**
- **/etc/security/user**
- **/etc/security/user.roles**
- **/etc/security/limits**
- **/etc/security/environ**
- **/etc/group**
- **/etc/group**

終了状況

0 コマンドは正常に完了しました。

>0 エラーが発生しました。

例

1. ユーザーのマシン内のすべての有効なユーザーをリストするには、 dispuid コマンドを次のように入力します。

   ```
   dispuid
   ```

   出力は、次のようになります。

   ```
   root
   daemon
   bin
   sys
   adm
   uucp
   guest
   nobody
   lpd
   invscout
   imnadm
   user1
   ```

ファイル

項目	説明
/usr/sbin/dispuid	dispuid コマンドが入っています。
/etc/passwd	パスワード情報が入っています。

関連資料:

172 ページの『dispgid コマンド』

---

dist コマンド

目的

追加アドレスにメッセージを再配布します。

構文

dist [**+** Folder] [**-nodraftfolder** | **-draftfolder +***Folder*] [*Message* | **-draftmessage** *Message*] [**-annotate** [**-inplace** | **-noinplace**] | **-noannotate**] [**-form** *FormFile*] [**-editor** *Editor* | **-noedit**] [**-nowhatnowproc** | **-whatnowproc** *Program*]

説明

dist コマンドは、既存のメッセージを一連の新しいアドレスに再配布するためのインターフェースの働きをします。デフォルトでは、**dist** コマンドは、現行フォルダーから *UserMHDirectory***/draft** ファイルに現行メッセージをコピーして、エディターを始動します。現行フォルダーにデフォルト以外のメッセージを指定するには、*Message* パラメーターを使用します。

エディターは始動すると、各ヘッダー・フィールドの値の入力を求めるプロンプトを出します。 **dist** コマンドは、*UserMHDirectory***/distcomps** ファイルに定義されたヘッダー・フォーマットを使用します。 (このファイルがない場合、システムは **/etc/mh/distcomps** ファイルを使用します。) メッセージ本体は、現在再配布中のメッセージなので、メッセージ本体に書き込みを行わないでください。 *UserMHDirectory***/distcomps** ファイル以外のフォーマット・ファイルを定義するには、**-form** フラグを使用します。

デフォルト・エディターを変更するには、**-editor** フラグを使用するか、または **$HOME/.mh_profile** ファイルの、 `Editor:` エントリーを定義します。

エディターを終了するには、Ctrl-D キー・シーケンスを押します。エディターの終了時に、**dist** コマンドはメッセージ・ハンドラー (MH) の「`What Now?`」というプロンプトを始動します。使用可能な **whatnow** サブコマンドのリストを参照するには、 Enter キーを押してください。これらのサブコマンドを使用すれば、メッセージ・ヘッダーの編集を続行したり、メッセージ・ヘッダーを表示したり、メッセージの後処理を指示したり、**dist** コマンドの処理を終了することができます。

注**:** 送信時にメッセージが識別できるように、メッセージのヘッダーと本文の間に、ハイフンの行またはブランク行を 1 行入れておく必要があります。

再配布されるメッセージは、元のヘッダーと、新しいヘッダーに追加された本体で構成されています。 **dist** コマンドを使用して編集する **draft** ファイルは、ヘッダー・フィールドだけから構成されています。新しいドラフト・メッセージを持つ元のメッセージのコピーは自動的には格納されません。

元のメッセージに再配布情報に関する注釈を付けるには、**-annotate** フラグを使用します。このフラグは、元のメッセージに `Resent:` フィールドと、現在の日付および時刻を追加します。

フラグ

項目	説明
-annotate	再配布されるメッセージに、次のような行の注釈を付けます。 `Resent: date` `Resent: address` **-annotate** フラグはコマンドの実行後まで保存されないため、メッセージが **dist** コマンドから直接送信された時点で、注釈付けは完了します。 **-inplace** フラグを指定すると、注釈付きメッセージとのリンクを保存するために、注釈付けが所定の箇所で強制的に行われます。
-draftfolder *+Folder*	指定したフォルダーにドラフト・メッセージを入れます。**-draftfolder +***Folder* フラグの後に *Message* 変数が続いている場合は、**-draftmessage** フラグを使用した場合と同じです。 *+Folder* が指定されていないと、ドラフト・メッセージは *Current-Folder* に取り込まれます。

項目	説明
-draftmessage *Message*	ドラフト・メッセージを指定します。デフォルトでは、現行フォルダーに新しいドラフト・メッセージが作成されます。そのドラフト・メッセージが現行メッセージになります。
-editor *Editor*	配布するメッセージを作成するための初期エディターを指定します。
+*Folder*	再配布するメッセージが入っているフォルダーを指定します。フォルダーが指定されていないと、*Current-Folder* が代わりに用いられます。
-form *FormFile*	メッセージのフォーマットを決定します。**dist** コマンドは、指定されたフォーマット・ファイル内の各行を扱います。
-help	コマンド構文、使用可能なスイッチ (トグル)、およびバージョン情報を表示します。 注**:** メッセージ・ハンドラー (MH) の場合、このフラグ名は完全にスペルアウトしなければなりません。
-inplace	注釈付きメッセージとのリンクを保存するために、注釈付けが所定の箇所で強制的に行われるようにします。
Message	再配布するメッセージを指定します。メッセージの指定には、次の参照を使用します。 *Number* メッセージの番号。 **cur** または **.** (ピリオド) 現行メッセージ。これはデフォルトです。 **first** フォルダー内の最初のメッセージ。 **last** フォルダー内の最後のメッセージ。 **next** 現行メッセージの次のメッセージ。 **prev** 現行メッセージの直前のメッセージ。
-noannotate	注釈を抑止します。このフラグはデフォルトです。
-nodraftfolder	ドラフトを *UserMHDirectory*/**draft** ファイルに入れます。
-noedit	初期編集を抑制します。
-noinplace	適切な注釈付けを強制しません。このフラグはデフォルトです。
-nowhatnowproc	**dist** コマンドの対話式処理を抑止します。 **-nowhatnowproc** フラグはいかなる編集も発生しないようにします。
-whatnowproc *Program*	指定されたプログラムを始動すると、配布タスクが示されます。 *Program* 変数として **whatnow** コマンドを指定すると、**dist** コマンドは **whatnow** というファイル名を持つプログラムではなく、内部 **whatnow** プロシージャーを始動します。

プロファイル・エントリー

次のエントリーを、 *UserMHDirectory*/**.mh_profile** ファイルに入力します。

項目	説明
`Current-Folder:`	デフォルトの現行フォルダーを設定します。
`Draft-Folder:`	ドラフトを入れるデフォルトのフォルダーを設定します。
`Editor:`	デフォルトのエディターを設定します。
`fileproc:`	メッセージをリファイルするプログラムを指定します。
`Path:`	ユーザーの MH ディレクトリーを指定します。
`whatnowproc:`	「`What now?`」という質問のプロンプトを出すのに使用するプログラムを指定します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 現行メッセージを現行フォルダーから再配布するには、次のように入力します。

   ```
   dist
   ```

   ヘッダー・フィールドの値の入力を求めるプロンプトが表示されます。値を入力したら、Enter キーを押します。入力をスキップするには、値を入力しないで、Enter キーを押します。 Resent-to: フィールドは必須です。ヘッダーの完了後に、テキスト本体を変更しないでください。エディターを終了するには、Ctrl-D キー・シーケンスを押します。次のようなプロンプトが表示されます。

   ```
   What now?
   ```

   使用可能なオプションのリストを表示するには、Enter キーを押します。このメッセージを再配布したい場合は、send. と入力してください。メッセージは新しいリストのアドレスに再配布されます。

2. メッセージ・ドラフトが存在する場合に、新しいリストのアドレスにメッセージを再配布するには、次のように入力します。

   ```
   dist
   ```

   システムは次のようなメッセージで応答します。

   ```
   Draft "$HOME/Mail/draft" exists (43 bytes).
   Disposition? _
   ```

   このドラフトを再配布するには、次のように入力します。

   ```
   replace
   ```

   ヘッダー・フィールドの値の入力を求めるプロンプトが表示されます。値を入力したら、Enter キーを押します。入力をスキップするには、値を入力しないで、Enter キーを押します。 Resent-to: フィールドは必須です。ヘッダーの完了後に、テキスト本体を変更しないでください。エディターを終了するには、Ctrl-D キー・シーケンスを押します。次のようなプロンプトが表示されます。

   ```
   What now?
   ```

   使用可能なオプションのリストを表示するには、Enter キーを押します。ドラフトを再配布したい場合は、send. と入力してください。メッセージは新しいリストのアドレスに再配布されます。

3. schedules フォルダーから、メッセージ 15 を再配布するには、次のように入力します。

   ```
   dist  +schedules 15
   ```

   ヘッダー・フィールドの値の入力を求めるプロンプトが表示されます。値を入力したら、Enter キーを押します。入力をスキップするには、値を入力しないで、Enter キーを押します。 Resent-to: フィールドは必須です。ヘッダーの完了後に、テキスト本体を変更しないでください。エディターを終了するには、Ctrl-D キー・シーケンスを押します。次のようなプロンプトが表示されます。

   ```
   What now?
   ```

   使用可能なオプションのリストを表示するには、Enter キーを押します。メッセージを再配布するには、send と入力して Enter キーを押します。

ファイル

項目	説明
/etc/mh/distcomps	システムのデフォルト・メッセージ形式が入っています。
UserMHDirectory/**distcomps**	ユーザーのデフォルト・メッセージ形式が入っています。
UserMHDirectory/**draft**	現行ドラフト・ファイルが入っています。
/usr/bin/dist	**dist** コマンドの実行可能フォーマットが入っています。

関連資料:

598 ページの『forw コマンド』

関連情報:

whatnow コマンド

mh_alias コマンド

メール・アプリケーション

---

dmpuncompress コマンド

目的

ダンプ圧縮ファイルを復元します。

構文

/usr/bin/dmpuncompress [**-f**] [**-p**] [*File*]

説明

dmpuncompress コマンドは、ダンプ時に圧縮されたダンプ・ファイルをオリジナルどおりに復元します。

File パラメーターで指定された各圧縮ファイルは、解凍コピーによって除去および置換されます。この解凍ファイルは圧縮バージョンと同じ名前です。ただし、**.BZ** 拡張子は付いていません。ユーザーが root 権限を持っている場合、この解凍ファイルは、元のファイルと同じ所有者、グループ、モード、および変更時刻を保持します。ユーザーが root 権限を持っていない場合、このファイルは同じモードおよび変更時刻を保持しますが、新規所有者およびグループを獲得します。

フラグ

項目	説明
-f *File*	拡張を強制します。ファイルが既に存在している場合は、ファイルを上書きします。ユーザーに対して、システムは既存のファイルが上書きされることを示すプロンプトを出しません。ファイルのサイズは、実際には縮小しないことがあります。
-p *File*	オリジナルの **.BZ** ファイルと解凍ダンプ・ファイルを保存します。これにより、オリジナルのダンプ・ファイルの回復が成功している場合は圧縮ファイルの除去が無効になります。オリジナルのダンプ・ファイルの回復がエラーにより完全でなかった場合、このオプションは部分的なダンプの除去を使用不可にします。

終了状況

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. **dump.BZ** ファイルを解凍するには、次のように入力します。

   ```
   /usr/bin/dmpuncompress dump.BZ
   ```

   dump.BZ ファイルは解凍され、名前が **dump** に変更されます。
2. ファイルシステムに **dump.BZ** ファイルと新規に作成されたダンプ・ファイルを完了後に保持するには、次のように入力します。

   ```
   /usr/bin/dmpuncompress -p dump.BZ
   ```

位置

/usr/bin/dmpuncompress

関連情報:

savecore コマンド

snap コマンド

uncompress コマンド

システム・ダンプ機能

---

dnssec-keygen コマンド

目的

ドメイン・ネーム・システム・セキュリティー拡張 (DNSSEC) キー生成ツール。

構文

dnssec-keygen [**-a** *algorithm*] [**-b** *keysize*] [**-n** *nametype*] [**-c** *class*] [**-e**] [**-f** *flag*] [**-g** *generator*] [**-h**] [**-k**] [**-p** *protocol*] [**-r** *randomdev*] [**-s** *strength*] [**-t** *type*] [**-v** *level*] [*name*]

説明

dnssec-keygen コマンドは、DNSSEC (セキュア DNS) のキーを生成します。このコマンドは、TSIG (Transaction Signature: トランザクション署名) で使用するためのキーを生成することもできます。

フラグ

項目	説明
-a *algorithm*	暗号アルゴリズムを選択します。アルゴリズムは、次のいずれかの値になります。 • RSAMD5 • DSA • DH (Diffie-Hellman) • HMAC-MD5 これらの値の大/小文字は区別されます。 **注:** 1. DNSSEC については、RSASHA1 がアルゴリズムのインプリメントに必須であり、DSA が推奨されます。TSIG については HMAC-MD5 が必須です。 2. HMAC-MD5 と DH は **-k** フラグを自動的に設定します。
-b *keysize*	キー内のビット数を指定します。キー・サイズの選択は、使用されるアルゴリズムによります。RSAMD5 キーと RSASHA1 キーは、512 ビットから 4096 ビットの範囲内にある必要があります。DH キーは、 128 から 4096 ビットの範囲内にある必要があります。DSA キーは、 512 から 1024 ビットの範囲内にあり、64 の正確な倍数である必要があります。HMAC-MD5 キーは、1 から 512 ビットの範囲内にある必要があります。
-n *nametype*	キーのオーナー・タイプを指定します。*nametype* の値は、ZONE (DNSSEC ゾーン・キーの場合)、HOST または ENTITY (ホストに関連付けられたキーの場合)、USER (ユーザーに関連付けられたキーの場合)、または OTHER (DNSKEY) のいずれかである必要があります。これらの値の大/小文字は区別されません。
-c *class*	キーを含んでいるドメイン・ネーム・サーバー (DNS) レコードが、指定されたクラスを備えている必要があることを示します。これが指定されない場合、クラス IN が使用されます。
-e	RSAMD5 キーまたは RSASHA1 キーを生成する場合は、大きい指数を使用します。
-f *flag*	KEY または DNSKEY レコードの *flag* フィールドに指定のフラグを設定します。認識されるフラグは、KSK (鍵署名の鍵) DNSKEY のみです。
-g *generator*	DH キーを生成する場合は、この生成プログラムを使用します。許容値は 2 および 5 です。生成プログラムの指定がない場合は、可能であれば RFC 2539 からの既知のプライム記号が使用され、それ以外の場合はデフォルトは 2 となります。
-h	**dnssec-keygen** コマンドに対するオプションおよび引数の簡略な要約を印刷します。
-k	KEY レコードを生成します (DNSKEY レコードではありません)。
-p *protocol*	生成されるキーのプロトコル値を設定します。protocol は、 0 から 255 の範囲内の数です。デフォルトは 3 (DNSSEC) です。
-r *randomdev*	ランダムのソースを指定します。オペレーティング・システムが /dev/random ファイルまたはそれと同等のデバイスを提供しない場合、デフォルトのランダムのソースはキーボード入力になります。*randomdev* 引数は、デフォルトの代わりに使用されるランダム・データを含むキャラクター・デバイスまたはファイルの名前を指定します。特殊値 keyboard は、キーボード入力を使用する必要があることを示します。
-s *strength*	キーのストレングスの値を指定します。*strength* 引数は、0 から 15 の範囲内の数値で、現在のところ DNSSEC で定義された目的はありません。
-t *type*	キーの使用を示します。*type*は、AUTHCONF、NOAUTHCONF、NOAUTH、または NOCONF のいずれかでなければなりません。デフォルトは AUTHCONF です。AUTH はデータを認証する能力を示し、CONF はデータを暗号化する能力を示します。キー・タイプが NOAUTHCONF であるアルゴリズム (DH、HMAC-MD5、HMAC-SHA1、HMAC-SHA224、HMAC-SHA256、HMAC-SHA384、HMAC-SHA512) の場合、キーは生成されません。
-v *level*	デバッグのレベルを設定します。

パラメーター

項目	説明
name	コマンド・ラインで指定されたキーの名前。DNSSEC キーの場合、この名前はキーが生成されようとしているゾーンの名前に一致する必要があります。

生成されたキー

dnssec-keygen コマンドは、正常に完了すると、標準出力に Knnnn.+aaa+iiiii のフォームの文字列を印刷します。これは、生成されたキーの ID ストリングです。

- nnnn はキー名です。
- aaa はアルゴリズムの数値表現です。
- iiiii はキー ID (つまりフットプリント) です。

dnssec-keygen コマンドは、印刷される文字列に基づいた名前を使用して 2 つのファイルを作成します。Knnnn.+aaa+iiiii.key は公開鍵を含み、Knnnn.+aaa+iiiii.private は秘密鍵を含みます。

.key ファイルは、(直接、または $INCLUDE ステートメントで) ゾーン・ファイルに挿入できる DNSKEY レコードを含みます。.private ファイルは、アルゴリズム固有のフィールドを含みます。セキュリティー上の理由により、このファイルには一般の読み取り許可はありません。公開鍵と秘密鍵が同等のものであっても、.key と .private の両方のファイルが、HMAC-MD5 などの対称暗号化アルゴリズムに対して生成されます。

例

ドメイン example.com の 768 ビット DSA キーを生成するには、次のコマンドを入力します。

```
dnssec-keygen -a DSA -b 768 -n ZONE example.com
```

このコマンドにより、次のフォームの文字列が印刷されます。

```
Kexample.com.+003+26160
```

この例では、**dnssec-keygen** はファイル Kexample.com.+003+26160.key および Kexample.com.+003+26160.private を作成します。

関連資料:

159 ページの『dig コマンド』

関連情報:

named9 コマンド

nsupdate9 コマンド

rndc-confgen コマンド

---

dnssec-makekeyset コマンド

目的

ドメイン・ネーム・システム・セキュリティー拡張 (DNSSEC) ゾーン署名ツール。

構文

dnssec-makekeyset [**-a**] [**-s** *start-time*] [**-e** *end-time*] [**-h**] [**-p**] [**-r** *randomdev*] [**-t** *ttl*] [**-v** *level*] {*key...*}

説明

dnssec-makekeyset コマンドは、**dnssec-keygen** コマンドによって生成された 1 つ以上のキーから、1 つのキー・セットを生成します。これは、各キーの KEY レコードを含むファイルを作成し、各ゾーン・キーでキー・セットに自己署名を行います。出力ファイルは keyset-nnnn. のフォームです (*nnnn* はゾーン名)。

フラグ

項目	説明
-a	すべての生成された署名を検査します。
-s *start-time*	生成された SIG レコードが有効になる日付および時刻を指定します。これは、絶対時刻または相対時刻のいずれかを指定できます。絶対開始時刻は、 YYYYMMDDHHMMSS 表記の数字で示されます。 20000530144500 は、2000 年 05 月 30 日 14:45:00 UTC (協定世界時) を表します。相対開始時刻は +N (現在時刻から *N* 秒) で示されます。*start-time* が指定されない場合は、現在時刻が使用されます。
-e *end-time*	生成された SIG レコードが期限切れとなる日付および時刻を指定します。*start-time* 値と同様、絶対時刻は YYYYMMDDHHMMSS 表記で示されます。開始時刻 (start time) に対する相対時刻は、 +N (開始時刻から *N* 秒) で示されます。現在時刻に対する相対時刻は、now+N で示されます。 *end-time* が指定されなかった場合、開始時刻から 30 日の時間がデフォルトとして使用されます。
-h	**dnssec-makekeyset** コマンドに対するオプションおよび引数の簡略な要約を印刷します。
-p	ゾーンに署名する際に、疑似ランダム・データを使用します。これは、より速くなりますが、実ランダム・データを使用するよりも安全性が低下します。このオプションは、大きいゾーンに署名する場合や、エントロピー・ソースが限られている場合に便利です。
-r *randomdev*	ランダムのソースを指定します。オペレーティング・システムが /dev/random または同等のデバイスを提供しない場合、ランダム性のデフォルトのソースはキーボード入力です。 *randomdev* 値は、デフォルトの代わりに使用されるランダム・データを含むキャラクター・デバイスまたはファイルの名前を指定します。特殊値 keyboard は、キーボード入力を使用する必要があることを示します。
-t *ttl*	KEY および SIG レコードの TTL (time to live: 存続時間) を指定します。デフォルトは 3600 秒です。
-v *level*	デバッグのレベルを設定します。

パラメーター

項目	説明
key	キー・セット・ファイルに含まれるキーのリスト。これらのキーは、 **dnssec-keygen** コマンドで生成されたとおりに Knnnn.+aaa+iiiii のフォームで表されます。

例

次のコマンドは、dnssec-keygen マニュアル・ページで生成された **example.com** の DSA キーを含むキー・セットを生成します。

```
dnssec-makekeyset -t 86400 -s 20000701120000 -e +2592000 Kexample.com.+003+26160
```

この例では、**dnssec-makekeyset** コマンドはファイル keyset-example.com. を作成します。このファイルは、指定されたキーおよび自己生成した署名を含みます。.com ゾーンが DNSSEC 指向であり、 2 つのゾーンの管理者が互いに認証し、キーと署名を安全に交換するためのメカニズムを持っている場合は、example.com の DNS 管理者は、署名のために keyset-example.com. を .com の DNS 管理者へ送信することができます。

関連資料:

179 ページの『dnssec-keygen コマンド』

183 ページの『dnssec-signkey コマンド』

dnssec-signkey コマンド

目的

ドメイン・ネーム・システム・セキュリティー拡張 (DNSSEC) キー・セット署名ツール。

構文

dnssec-signkey [**-a**] [**-c** *class*] [**-s** *start-time*] [**-e** *end-time*] [**-h**] [**-p**] [**-r** *randomdev*] [**-v** *level*] *keyset key*

説明

dnssec-signkey コマンドは、キー・セットの署名を行います。一般に、キー・セットは子ゾーン用であり、**dnssec-makekeyset** コマンドによって生成されます。子ゾーンのキー・セットは、その親ゾーンのゾーン・キーで署名されます。出力ファイルは `signedkey-`*nnnn*`.` のフォームです (*nnnn* はゾーン名)。

フラグ

項目	説明
-a	すべての生成された署名を検査します。
-c *class*	キー・セットの DNS クラスを指定します。
-s *start-time*	生成された SIG レコードが有効になる日付および時刻を指定します。これは、絶対時刻または相対時刻のいずれかを指定できます。絶対開始時刻は、 `YYYYMMDDHHMMSS` 表記の数字で示されます。 20000530144500 は、2000 年 05 月 30 日 14:45:00 UTC (協定世界時) を表します。相対開始時刻は *+N* (現在時刻から *N* 秒) で示されます。*start-time* が指定されない場合は、現在時刻が使用されます。
-e *end-time*	生成された SIG レコードが期限切れとなる日付および時刻を指定します。start-time と同様、絶対時刻は `YYYYMMDDHHMMSS` 表記で示されます。開始時刻 (start time) に対する相対時刻は、 *+N* (開始時刻から *N* 秒) で示されます。現在時刻に対する相対時刻は、`now+N` で示されます。 *end-time* が指定されなかった場合、開始時刻から 30 日の時間がデフォルトとして使用されます。
-h	**dnssec-signkey** コマンドに対するオプションおよび引数の簡略な要約を印刷します。
-p	ゾーンに署名する際に、疑似ランダム・データを使用します。これは、より速くなりますが、実ランダム・データを使用するよりも安全性が低下します。このオプションは、大きいゾーンに署名する場合や、エントロピー・ソースが限られている場合に便利です。
-r *randomdev*	ランダムのソースを指定します。オペレーティング・システムが `/dev/random` または同等のデバイスを提供しない場合、ランダム性のデフォルトのソースはキーボード入力です。 *randomdev* は、デフォルトの代わりに使用されるランダム・データを含むキャラクター・デバイスまたはファイルの名前を指定します。特殊値 keyboard は、キーボード入力を使用する必要があることを示します。
-v *level*	デバッグのレベルを設定します。

パラメーター

項目	説明
keyset	子のキー・セットを含んでいるファイル。
key	子のキー・セットの署名に使用されるキー。

例

DNSSEC 指向の `.com` ゾーンの DNS 管理者は、次のコマンドを使用して、`example.com` (**dnssec-keygen** コマンドによって生成されたキーを使用して **dnssec-makekeyset** コマンドによって生成されたもの) のキー・セット・ファイルに署名します。

```
dnssec-signkey keyset-example.com. Kcom.+003+51944
```

この例で、**dnssec-signkey** は、 `example.com` キーおよび `.com` キーによる署名を含むファイル `signedkey-example.com.` を作成します。

関連資料:

179 ページの『dnssec-keygen コマンド』
181 ページの『dnssec-makekeyset コマンド』
『dnssec-signzone コマンド』

dnssec-signzone コマンド

目的

ドメイン・ネーム・システム・セキュリティー拡張 (DNSSEC) ゾーン署名ツール。

構文

dnssec-signzone [**-a**] [**-c** *class*] [**-d** *directory*] [**-e** *end-time*] [**-f** *output-file*] [**-g**] [**-h**] [**-k** *key*] [**-l** *domain*] [**-i** *interval*] [**-I** *input-format*] [**-j** *jitter*] [**-N** *soa-serial-format*] [**-o** *origin*] [**-O** *output-format*] [**-p**] [**-r** *randomdev*] [**-s** *start-time*] [**-t**] [**-v** *level*] [**-z**] *zonefile* [*key*...]

説明

dnssec-signzone コマンドは、ゾーンに署名します。このコマンドは、NSEC および RRSIG レコードを生成し、そのゾーンの署名付きバージョンを作成します。署名されたゾーンからの委任のセキュリティー状況 (つまり、子ゾーンがセキュアであるかどうか) は、それぞれの子ゾーンのキー・セット・ファイルが存在するかしないかで決まります。

フラグ

項目	説明
-a	すべての生成された署名を検査します。
-c *class*	ゾーンの DNS クラスを指定します。
-d *directory*	*directory* 引数に指定されたディレクトリーで、キー・セット・ファイルを検索します。
-k *key*	キー・フラグを無視する鍵署名の鍵として指定のキーを処理します。このオプションは、複数回指定できます。
-l *domain*	キー (DNSKEY) と DS セットに加えて DLV セットを生成します。このドメインがレコードの名前に追加されます。
-g	キー・セット・ファイルから子ゾーンの DS レコードを生成します。このフラグにより、既存の DS レコードが除去されます。
-s *start-time*	生成された RRSIG レコードが有効になる日付および時刻を指定します。これは、絶対時刻または相対時刻のいずれかを指定できます。絶対開始時刻は、 `YYYYMMDDHHMMSS` 表記の数字で示されます。 20000530144500 は、2000 年 05 月 30 日 14:45:00 UTC (協定世界時) を表します。相対開始時刻は *+N* (現在時刻から *N* 秒) で示されます。*start-time* 引数が指定されない場合、このコマンドは、現在時刻から 1 時間引いた時刻を使用します (クロック・スキューを許容するため)。
-e *end-time*	生成された RRSIG レコードが期限切れとなる日付および時刻を指定します。*start-time* 引数と同様、絶対時刻は `YYYYMMDDHHMMSS` 表記で示されます。開始時刻 (start time) に対する相対時刻は、 *+N* (開始時刻から *N* 秒) で示されます。現在時刻に対する相対時刻は、`now+N` で示されます。 *end-time* 引数を指定しない場合、コマンドは開始時刻から 30 日をデフォルトとして使用します。
-f *output-file*	署名されたゾーンを含む出力ファイルの名前を指定します。デフォルトでは、`.signed` が入力ファイルの名前に追加されます。
-h	**dnssec-signzone** コマンドのオプションと引数の簡略な要約を印刷します。
-i *interval*	入力として以前に署名されたゾーンが渡された場合、レコードは破棄される場合があります。インターバル・オプションは、現在時刻からのオフセットとしてサイクル間隔を指定します (秒単位)。RRSIG レコードがサイクル間隔後に期限切れになった場合、これは保持されます。そうでない場合は、すぐに期限切れになるものと見なされ、置き換えられます。デフォルトのサイクル間隔は、署名の終了と開始時刻間の差の 4 分の 1 です。*end-time* 引数も *start-time* 引数も指定されていない場合は、**dnssec-signzone** コマンドは、30 日間有効であり、サイクル間隔が 7.5 日の署名を生成します。したがって、既存の RRSIG レコードが 7.5 日未満で期限切れとなるものである場合、それらは置き換えられます。

項目	説明
-I *input-format*	入力ゾーン・ファイルのフォーマットを指定します。指定可能なフォーマットは、text (テキスト (デフォルト)) と raw (ロー) です。
-j *jitter*	修正された署名ライフタイムのあるゾーンに署名する場合は、署名時に発行されたすべての RRSIG レコードが同時に期限切れになります。例えば、ゾーンが増分して署名される場合は、前に署名されたゾーンが入力として署名者に渡され、期限切れのすべての署名がほぼ同時に再生成されなければなりません。*jitter* 引数は、署名の有効期限をランダム化して、増分署名の再生成を時間とともに拡張するために使用するジッター・ウィンドウを指定します。署名ライフタイムのジッターは、キャッシュの有効期限を延長することでバリデーターとサーバーに対しても有効となります。例えば、多くの RRSIG がすべてのキャッシュから同時に期限切れにならない場合、すべてのバリデーターがほぼ同時に再フェッチする必要がある場合よりも輻輳が少なくなります。
-n *ncpus*	使用するスレッドの数を指定します。デフォルトでは、このコマンドは、検出されたプロセッサーごとに 1 つのスレッドを開始します。
-N *soa-serial-format*	署名付きゾーンの SOA シリアル番号のフォーマットを指定します。*soa-serial-format* 引数は、次のいずれかの値になります。 **keep** SOA シリアル番号を変更しません。それはデフォルト値です。 **increment** RFC 1982 の演算を使用して SOA シリアル番号を増加させます。 **unixtime** エポックからの秒数に SOA シリアル番号を設定します。
-o *origin*	ゾーンの起点を指定します。指定されない場合、ゾーン・ファイルの名前が起点であると想定されます。
-O *output-format*	署名されたゾーンを含む出力ファイルのフォーマットを指定します。指定可能なフォーマットは、text (テキスト (デフォルト)) と raw (ロー) です。
-p	ゾーンに署名する際に、疑似ランダム・データを使用します。これは、より速くなりますが、実ランダム・データを使用するよりも安全性が低下します。このオプションは、大きいゾーンに署名する場合や、エントロピー・ソースが限られている場合に便利です。
-r *randomdev*	ランダムのソースを指定します。オペレーティング・システムが `/dev/random` ファイルまたはそれと同等のデバイスを提供しない場合、デフォルトのランダムのソースはキーボード入力になります。*randomdev* 引数は、デフォルトの代わりに使用されるランダム・データを含むキャラクター・デバイスまたはファイルの名前を指定します。特殊値 keyboard は、キーボード入力を使用する必要があることを示します。
-t	完了時に統計情報を印刷します。
-v *level*	デバッグのレベルを設定します。
-z	署名するものを判別する際にキーの KSK フラグを無視します。

パラメーター

項目	説明
zonefile	署名されるゾーンを含むファイル。
key	キー・セットの署名に使用されるキー。キーが指定されていない場合は、現行ディレクトリーに秘密鍵ファイルがあるすべてのゾーンのキーがデフォルトです。

例

次のコマンドは、`dnssec-keygen` コマンドで生成された DSA キーで、**example.com** ゾーンの署名を行います。ゾーンのキーは、ゾーン内にある必要があります。このゾーンまたはいずれかの子ゾーンに関連付けられているキー・セット・ファイルがある場合、それらのファイルは現行ディレクトリー `example.com` 内に存在する必要があります。次のコマンドを実行することができます。

```
dnssec-signzone -o example.com db.example.com Kexample.com.+003+26160
```

この例では、**dnssec-signzone** コマンドは `db.example.com.signed` ファイルを作成します。このファイルは、`named.conf` ファイル内のゾーン・ステートメントで参照されます。

関連情報:

named9 コマンド

named-checkconf コマンド
nsupdate9 コマンド
rndc-confgen コマンド

---

dodisk コマンド

目的

ディスク使用アカウンティングを開始します。

構文

/usr/sbin/acct/dodisk [**-X**] [**-o**] [*File ...*]

説明

dodisk コマンドは、 **diskusg** コマンドと、**acctdisk** コマンドを呼び出して、ディスク使用アカウンティングを開始します。**dodisk** コマンドに **-o** フラグを指定すると、**acctdusg** コマンドを使用してログイン・ディレクトリー別に、詳細ではあるが時間のかかるディスク・アカウンティングが開始されます。通常、**cron** デーモンが **dodisk** コマンドを実行します。

dodisk コマンドは、デフォルトでは、属性 **account=true** が入っている **/etc/filesystems** ファイル内にスタンザを持つ指定ファイルについてのみディスク・アカウンティングを行います。 *File* パラメーターでファイル名を指定すると、それらのファイルについてのみディスク・アカウンティングが行われます。

-o フラグを指定しない場合、*File* パラメーターにはマウント可能なファイルシステムのスペシャル・ファイル名を入れる必要があります。 **-o** フラグと *File* パラメーターの両方を指定した場合、それらのファイルはマウントされたファイルシステムのマウント・ポイントでなければなりません。

注**:** 分散環境ではアカウンティング・ファイルを複数のノード間で共用することはできません。各ノードには各種アカウンティング・ファイルの専用コピーがなければなりません。

フラグ

項目	説明
-o	**diskusg** コマンドではなく **acctdusg** コマンドを呼び出して、ログイン・ディレクトリー別のディスク・アカウンティングを開始します。
-X	各ユーザー名は、最初の 8 文字に切り捨てられずに、使用可能なすべての文字が処理されます。

セキュリティー

アクセス制御: このコマンドは、**adm** グループのメンバーのみに実行 (x) アクセス権限を与えます。

例

1. 自動ディスク使用アカウンティングを始動するには、**/var/spool/cron/crontabs/root** ファイルに次のように追加します。

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   ```

   この例は、**cron** デーモンにより読み取られ、処理される命令を示しています。 **dodisk** コマンドは、毎週木曜日 (4) の午前 2 時 (0 2) に実行されます。このコマンドは、通常 **cron** デーモンに与えられ

るアカウンティング命令の 1 つにすぎません。一般的な **cron** アカウンティング・エントリーの詳細については、「オペレーティング・システムおよびデバイスの管理」の『アカウンティング・システムの設定』を参照してください。

2. 8 文字より長いユーザー名のシステムでディスク使用アカウンティングを実行するには、**/var/spool/cron/crontabs/root** ファイルに次の行を追加します。

```
0 2 * * 4 /usr/sbin/acct/dodisk -X
```

ファイル

項目	説明
/usr/sbin/acct	アカウンティング・コマンドへのパス。
/etc/filesystems	ファイルシステムに関する情報が入っています。

関連資料:

171 ページの『diskusg コマンド』

関連情報:

acctdisk コマンド

cron コマンド

システム・アカウンティング

---

domainname コマンド

目的

現在の NIS ドメインの名前を表示または設定します。

構文

/usr/bin/domainname [*DomainName*]

説明

domainname コマンドは、現行 NIS ドメインの名前を表示または設定します。パラメーターを指定しないと、**domainname** コマンドにより現行 NIS ドメインの名前が表示されます。通常、ドメインには、同一管理下のホストのグループが含まれています。

domainname コマンドに引数を指定することによってドメインの名前を設定できるのは、root ユーザーだけです。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 新しいドメインを設定するには、次のように入力します。

```
domainname caesar
```

この例では、domainname コマンドは NIS ドメイン・ネームを caesar に設定しています。

2. ユーザーのマシンが属しているドメインの名前を見つけるには、次のように入力します。

```
domainname
```

関連情報:

ypinit コマンド

ypserv コマンド

Network Information Service (NIS) Overview for System Management

---

domlist コマンド

目的

ユーザーまたはプロセスのドメイン情報を表示します。

構文

domlist -p *pid*

説明

domlist コマンドは、呼び出し側に対して現在割り当てられているドメインに関するドメイン情報を提供します。フラグまたは引数が指定されない場合、**domlist** コマンドは、呼び出し側に割り当てられたドメインのリストと、ドメイン・データベースにある場合は各ドメインのテキスト記述を表示します。

また、**domlist** コマンドは、特権ユーザーがプロセスのドメイン情報をリストすることを許可します。**-p** フラグでプロセス ID を指定すると、特権ユーザーは、プロセスに関連付けられているドメインを表示できます。

フラグ

項目	説明
-p *PID*	指定されたプロセスのドメイン情報を表示します。

例

1. ユーザーに割り当てられたドメインのリストおよびそれらのテキスト記述を表示するには、次のコマンドを使用します。

```
domlist
```

2. プロセスに割り当てられたドメインのリストを表示するには、次のコマンドを使用します。

```
domlist -p <pid>
```

アクセスするファイル

項目	説明
ファイル	モード
/etc/security/domains	r

関連情報:

mkdom コマンド

lsdom コマンド

rmdom コマンド

setkst コマンド

---

dosdel コマンド

目的

DOS ファイルを削除します。

構文

dosdel [**-v**] [**-D** *Device*] *File ...*

説明

dosdel コマンドは、*File* パラメーターで指定された DOS ファイルを削除します。ディスクに関するフォーマット情報を得るには、 **-v** フラグを使用します。

1 つの例外を除き、 DOS ファイルの命名規則が使用されます。¥ (円記号) は、オペレーティング・システムにとって特別な意味を持つ場合があるので、 DOS パス名にサブディレクトリー名を指定するための区切り文字には、 / (スラッシュ) を使用してください。**dosdel** コマンドは、ディスクを検査する前に、ファイルまたはディレクトリー名の小文字を大文字に変換します。ファイル名はすべて絶対パス名 (相対パス名ではない) と見なされるので、最初の / (スラッシュ) は付ける必要はありません。

フラグ

項目	説明
-D *Device*	DOS デバイスの名前を **/dev/fd0** または **/dev/fd1** と指定します。デフォルト・デバイスは **/dev/fd0** です。
-v	ディスクのフォーマットに関する情報を、標準出力します。このフラグを使用して、デバイスが DOS ディスクであるかどうか確認してください。

例

デフォルト・デバイス上の DOS ファイルを削除するには、次のように入力します。

```
dosdel file.ext
```

ファイル

項目	説明
/usr/bin/dosdel	**dosdel** コマンドが入っています。

関連資料:

『dosdir コマンド』

192 ページの『dosformat コマンド』

194 ページの『dosread コマンド』

195 ページの『doswrite コマンド』

---

dosdir コマンド

目的

DOS ファイル用のディレクトリーをリストします。

構文

dosdir [-l [-e]] [-a] [-d] [-t] [-v] [-D *Device* **] [** *File ...* **|** *Directory ...* **]**

説明

dosdir コマンドは、指定された DOS ファイルまたはディレクトリーに関する情報を表示します。 **-d** フラグを指定しないでディレクトリーを指定すると、 **dosdir** コマンドは、そのディレクトリー内のファイルに関する情報を表示します。

1 つの例外を除き、 DOS ファイルの命名規則が使用されます。¥ (円記号) は、オペレーティング・システムにとって特別な意味を持つ場合があるので、 DOS パス名にサブディレクトリー名を指定するための区切り文字には、 / (スラッシュ) を使用してください。**dosdir** コマンドは、ディスクを検査する前に、ファイルまたはディレクトリー名の小文字を大文字に変換します。ファイル名はすべて絶対パス名 (相対パス名ではない) と見なされるので、最初の / (スラッシュ) は付ける必要はありません。

フラグ

項目	説明
-a	すべてのファイルに関する情報を出力します。これには、.(ドット) ファイルおよび .. (ドット - ドット) ファイルだけでなく、隠しファイルとシステム・ファイルが含まれます。
-d	ディレクトリーが指定されている場合でも、*File* 値をファイルとして処理します。*Directory* パラメーターにディレクトリーを指定すると、ディレクトリー内のファイルの情報ではなく、ディレクトリー自体の情報がリスト表示されます。
-D *Device*	DOS デバイスの名前を **/dev/fd0** または **/dev/fd1** と指定します。デフォルト・デバイスは **/dev/fd0** です。
-e	**-l** フラグを使用して、割り当てられたクラスターのリストをファイルに書き込みます。

項目	説明
-l	ファイルの作成日、サイズ (バイト単位)、属性を含むクラスターのリストを作成します。サブディレクトリーのサイズは 0 バイトと指定されます。属性は次のような意味を持っています。 **A (**アーカイブ**)** 最後に変更されて以来、ファイルはバックアップされていません。 **D (**ディレクトリー**)** ファイルはサブディレクトリーで、通常の DOS ディレクトリー検索に含まれません。 **H (**隠しファイル**)** ファイルは通常の DOS ディレクトリー検索に含まれません。 **R (**読み取り専用**)** ファイルは変更できません。 **S (**システム**)** ファイルはシステム・ファイルで、通常の DOS ディレクトリー検索に含まれません。
-t	指定されたディレクトリーで始まるディレクトリー・ツリー全体をリスト表示します。
-v	ディスクのフォーマットに関する情報を、標準出力します。このフラグを使用して、デバイスが DOS ディスクであるかどうか確認してください。

例

/dev/fd0 上の DOS ファイルのディレクトリーを読み取るには、次のように入力します。

```
dosdir
```

このコマンドは、ファイル名およびディスク・スペース情報を次のように出力します。

```
PG3-25.TXT
PG4-25.TXT
PG5-25.TXT
PG6-25.TXT
Free space: 312320 bytes
```

/dev/fd1 上の DOS ファイルのディレクトリーを読み取るには、次のように入力します。

```
dosdir -D/dev/fd1
```

このコマンドは、ファイル名およびディスク・スペース情報を次のように出力します。

```
PG7-25.TXT
PG8-25.TXT
PG9-25.TXT
PG10-25.TXT
Free space: 312320 bytes
```

ファイル

項目	説明
/usr/bin/dosdir	**dosdir** コマンドが入っています。

関連資料:

189 ページの『dosdel コマンド』

192 ページの『dosformat コマンド』

194 ページの『dosread コマンド』

195 ページの『doswrite コマンド』

dosformat コマンド

目的

DOS ディスケットをフォーマットします。

構文

dosformat [**-V** *Label*] [**-D** *Device* | **-4**]

説明

dosformat コマンドは、ディスケットを DOS フォーマットでフォーマットします。

デフォルト・デバイスおよび DOS ディスケット・ドライブのフォーマットは、3.5 インチ、1.44MB のディスケットの場合、**/dev/fd0** です。通常、密度は 1.44MB か 2.88MB のいずれかで、ドライブがサポートする密度によって異なります。他の DOS ディスケット・ドライブのフォーマットは、**-D** フラグまたは **-4** フラグを使用することによって実行されます。

ボリューム・ラベルを指定するには、**-V** フラグを使用します。

注**:** このコマンドの目的は、このオペレーティング・システムと DOS システム間のファイルの転送を容易にすることです。DOS システム始動ファイルを入れるためのディスケットをこのコマンドでフォーマットすることはお勧めできません。

フラグ

項目	説明
-V	*Label* パラメーターを DOS ボリューム・ラベルとしてディスケットに書き込みます。

項目	説明
-D *Device*	ディスケット・ドライブの種類とサイズを指定します。 *Device* パラメーターには、次の値を指定できます。

3.5 インチ、1.44MB (デフォルト)

/dev/fd0
1.44MB (デフォルト)

/dev/fd0h
1.44MB

/dev/fd0l
720KB

/dev/fd0.18
1.44MB

/dev/fd0.9
720KB

3.5 インチ、2.88 メガのドライブの場合、次のように入力します。

/dev/fd0
2.88MB (デフォルト)

/dev/fd0h
2.88MB

/dev/fdol
720KB

/dev/fd0.36
2.88MB

/dev/fd0.18
1.44MB

/dev/fd0.9
720KB

5.25 インチ、1.2 メガのドライブの場合、次のように入力します。

/dev/fd0
1.2MB (デフォルト)

/dev/fd0.15
1.2MB

/dev/fd0.9
360KB

項目	説明
-4	ディスケット・サイズを低密度に指定します。

例

1. 「homework」というボリューム・ラベルを持つ 3.5 インチ、 1.44MB のディスケットをフォーマットするには、次のように入力します。

   ```
   dosformat -V homework
   ```

2. 5.25 インチ、360KB のディスケットをフォーマットするには、次のように入力します。

   ```
   dosformat -D /dev/fd1.9
   ```

   または

```
dosformat -D /dev/fd1 -4
```

ファイル

項目	説明
/usr/bin/dosformat	**dosformat** コマンドが入っています。

関連資料:

189 ページの『dosdel コマンド』
190 ページの『dosdir コマンド』
『dosread コマンド』
195 ページの『doswrite コマンド』

---

dosread コマンド

目的

DOS ファイルをコピーします。

構文

dosread [**-a**] [**-v**] [**-D** *Device*] *File1* [*File2*]

説明

dosread コマンドは、 *File1* 変数が指定する DOS ファイルを、標準出力に、あるいは *File2* 変数が指定するファイルに、コピーします。*File2* 変数に対してパス名が指定されていないと、DOS ファイルはルート・ディレクトリーにコピーされます。

特に指定がなければ、**dosread** コマンドは、*File1* のディレクトリー・エントリーに指定されたバイト数をコピーします。つまり、規則ではディレクトリーのレコード・サイズが 0 になっているので、特にディレクトリーのコピーは機能しません。

唯一の例外である ¥ (円記号) を除き、DOS ファイルの命名規則を使用できます。¥ (円記号) は DOS にとって特別な意味を持つ場合があるので、DOS パス名にサブディレクトリー名を指定する区切り文字には、/ (スラッシュ) を使用してください。**dosdir** コマンドは、ディスクを検査する前に、ファイルまたはディレクトリー名の小文字を大文字に変換します。ファイル名はすべて絶対パス名 (相対パス名ではない) と見なされるので、最初の / (スラッシュ) は付ける必要はありません。

注**:**

1. **dosread** コマンドは、* (アスタリスク) および ? (疑問符) のワイルドカード文字を特殊な意味を持つものとは解釈しません。ファイル名の拡張子を指定しないと、ブランクの拡張子を指定した場合と同様にファイル名の照合が行われます。
2. このコマンドの名前をカスタマイズすることはできません。このコマンドは必ず **dosread** という名前でなければなりません。
3. **dosread** コマンドは、 DOS ディスケットが入っているデフォルト・ドライブからファイルを読み取ります。次に、**dosread** コマンドは、そのファイルを現行ディレクトリーに、このオペレーティング・システムが認識するファイルとしてコピーします。 DOS ディスケットにサブディレクトリーが入っていると、 **dosread** コマンドは、このオペレーティング・システム内に、それに対応する新しいサブディ

レクトリーを作成しません。ユーザーがサブディレクトリーを作成して、その新規サブディレクトリーにコピーする個々の DOS ファイルを指定する必要があります。

フラグ

項目	説明
-a	各 CR-LF (復帰、改行) キー・シーケンスを改行文字に置き換え、Ctrl-Z (ASCII SUB) キー・シーケンスを行の終わりの文字として解釈します。
-D *Device*	DOS デバイスの名前を **/dev/fd0** または **/dev/fd1** と指定します。デバイス 変数のデフォルト値は **/dev/fd0** です。このデバイスは、DOS ディスク・フォーマットでなければなりません。
-v	ディスクのフォーマットに関するファイル情報を、標準出力します。このフラグを使用して、デバイスが DOS ディスクであるかどうか確認してください。

例

1. DOS からテキスト・ファイルをコピーするには、次のように入力します。

   ```
   dosread  -a chap1.doc chap1
   ```

   このコマンド・シーケンスは、デフォルト・デバイス **/dev/fd0** 上の DOS テキスト・ファイル、¥CHAP1.DOC を、現行ディレクトリー内の chap1 にコピーします。
2. DOS ディスケットからバイナリー・ファイルをコピーするには、次のように入力します。

   ```
   dosread -D/dev/fd1 /survey/test.dta /home/fran/testdata
   ```

   このコマンド・シーケンスは、 **/dev/fd1** 上の DOS データ・ファイル、 ¥SURVEY¥TEST.DTA を、/home/fran/testdata にコピーします。
3. ディスケット上の DOS ファイルをすべてコピーするには、次のように入力します。

   ```
   dosdir | awk '!/There are/ {print $1}'|xargs -t -i dosread {} {}
   ```

   このコマンド・シーケンスは、 DOS ディスクが入っているデフォルト・ドライブからファイルを読み取り、そのファイルを現行ディレクトリーにコピーします。

ファイル

項目	説明
/usr/bin/dosread	**dosread** コマンドが入っています。
/dev/fd0	ディスケット・ドライブのデバイス名が入っています。

関連資料:

189 ページの『dosdel コマンド』

『doswrite コマンド』

関連情報:

awk コマンド

ファイルのタイプ

doswrite コマンド

目的

ファイルを DOS ファイルにコピーします。

構文

doswrite [**-a**] [**-v**] [**-D***Device*] *File1 File2*

説明

doswrite コマンドは、 *File1* パラメーターで指定したファイルを、 *File2* パラメーターで指定した DOS ファイルにコピーします。**doswrite** コマンドはいくつかのファイルを 1 つの DOS ディスケットにコピーします。**doswrite** コマンドではいくつかのファイルを複数の DOS ディスケットにコピーすることはできません。

doswrite コマンドは、標準の DOS 命名規則を使用して *File2* パラメーターで指定したファイルを DOS デバイスに書き出します。 DOS の ¥ (円記号) は DOS オペレーティング・システムにとって特別な意味を持つ場合があるので、*File2* パラメーターにサブディレクトリー名を指定する際は ¥ (円記号) を使用しないでください。代わりに / (スラッシュ) 文字を使用してください。

doswrite コマンドは、*File1* パラメーターで指定した小文字を大文字に変換してから、DOS デバイスを検査します。すべてのファイル名が絶対パス名 (相対パス名ではない) と見なされるので、最初の / (スラッシュ) は付ける必要はありません。

File2 パラメーターで指定したファイルに / (スラッシュ) が含まれている場合、その間にある各コンポーネントはディレクトリーとして存在しなければならず、最後のコンポーネント (指定されたファイル) は存在していてはなりません。同じ名前のファイルが存在する場合は、すべて上書きされます。

注**:**

1. このコマンドでは、ワイルドカード文字 * と ? (アスタリスクと疑問符) は特別な方法では処理されません (ただし、シェルによって処理されます)。ファイル名の拡張子を指定しないと、ブランクの拡張子を指定した場合と同様にファイル名の照合が行われます。
2. このコマンドは、 **doswrite** と指定されなければなりません。
3. 1 つの DOS ディレクトリーには、最高で 244 個のファイルが入ります。

フラグ

項目	説明
-a	NL (改行文字) を CR-LF (復帰、改行) シーケンスに置き換えます。ファイルの末尾に Ctrl-Z が追加されます。
-D *Device*	DOS デバイスの名前を **/dev/fd0** または **/dev/fd1** と指定します。デフォルト・デバイスは **/dev/fd0** です。このデバイスは、DOS ディスク・フォーマットでなければなりません。
-v	ディスクのフォーマットに関する情報を、標準出力します。このフラグを使用して、デバイスが DOS ディスクであるかどうか確認してください。

例

1. テキスト・ファイルを DOS ディスケットにコピーするには、次のように入力します。

   ```
   doswrite  -a chap1 chap1.doc
   ```

   これによって、現行ディレクトリー内のファイル `chap1` が、デフォルト・デバイス **/dev/fd0** 上の DOS テキスト・ファイル、 `¥CHAP1.DOC` にコピーされます。
2. バイナリー・ファイルを DOS ディスケットにコピーするには、次のように入力します。

   ```
   doswrite -D/dev/fd1 /home/fran/testdata /survey/test.dta
   ```

これによって、データ・ファイル /home/fran/testdata が、 **/dev/fd1** 上の DOS ファイル、¥SURVEY¥TEST.DTA にコピーされます。

3. 現行ディレクトリーのすべてのファイルを、デフォルト・ドライブの DOS ディスケットにコピーするには、次のように入力します。

```
for i in *
do
doswrite $i $i
done
```

ファイル

項目	説明
/usr/bin/doswrite	**doswrite** コマンドが入っています。
/dev/fd0	ディスケット・ドライブのデバイス名が入っています。

関連資料:

190 ページの『dosdir コマンド』

192 ページの『dosformat コマンド』

194 ページの『dosread コマンド』

---

dp コマンド

目的

日付を解析して再設定します。

構文

dp [**-form** *File* | **-format** *String*] [**-width** *Number*] *Date*

説明

dp コマンドは、日付を解析して再設定します。 **dp** コマンドをユーザーが始動することはできません。 **dp** コマンドは他のプログラムから、その絶対パス名 (通常は、**/usr/lib/mh/dp**) で呼び出されます。

dp コマンドは、日付として指定された個々のメールのヘッダー文字列を解析し、その文字列を再設定しようとします。 **dp** コマンドのデフォルトの出力フォーマットは ARPA RFC 822 標準です。解析できない個々の文字列に対して、**dp** コマンドはエラー・メッセージを表示します。

パラメーター

項目	説明
Date	解析される日付を指定します。

フラグ

項目	説明
-form *File*	*Date* パラメーターで指定されたデータを、*File* 変数によって記述された代替表示方法に再設定します。
-format *String*	*Date* パラメーターで指定されたデータを、*String* 変数によって記述された代替表示方法に再設定します。デフォルトの式設定文字列は次のとおりです。 `%<(nodate{text})error:%{text}%|%(putstr(pretty{text}))%>`
-help	コマンド構文、使用可能なスイッチ (トグル)、およびバージョン情報を表示します。 注: メッセージ・ハンドラー (MH) の場合、このフラグ名は完全にスペルアウトしなければなりません。
-width *Number*	日付とエラー・メッセージを表示するのに **dp** コマンドが使用する桁の最大数を設定します。デフォルトはディスプレイの幅です。

ファイル

項目	説明
$HOME/.mh_profile	MH ユーザー・プロファイルが入っています。
/etc/mh/mtstailor	MH コマンド定義が入っています。

関連情報:

ap コマンド

メール・アプリケーション

---

dpid2 デーモン

目的

dpid2 Distributed Protocol Interface - SNMP マルチプレクサー・プロトコル (DPI-SMUX) コンバーター・デーモンをバックグラウンド・プロセスとして開始します。

構文

dpid2 [**-d** [*Level*]]

説明

dpid2 コマンドは、 **dpid2** DPI-SMUX コンバーター・デーモンを開始します。このコマンドは、root 権限を持つユーザーまたはシステム・グループのメンバーのみが発行できます。

dpid2 DPI-SMUX コンバーター・デーモンは、RFC 1592 で定義された標準の Simple Network Management Protocol (SNMP) DPI バージョン 2.0、および RFC 1227 で定義された SNMP SMUX プロトコルおよび管理情報ベース (MIB) に準拠しています。

dpid2 デーモンは、SMUX コンバーターに対して DPI 2.0 として動作します。これは、`/usr/sbin/hostmibd` などの DPI サブエージェントに AIX SNMP バージョン 1 エージェントとの通信を許可するために使用されます。コンバーターは、 DPI2 メッセージを SMUX プロトコル・メッセージに変更し、その逆もまた同じです。**dpid2** デーモン自体は、 SMUX のピアとしてインプリメントされます。これは、**snmpd** エージェントの一部である SMUX サーバーの TCP ポート 199 に接続します。 **dpid2** デーモンは、DPI2 サブエージェント (例えば `/usr/sbin/hostmibd`) に対しては、DPI2 エージェントのように動作します。これは、DPI2 サブエージェントからの接続要求を、任意の TCP ポート上で listen します。これは、MIB 変数 dpiPortForTCP (1.3.6.1.4.1.2.2.1.1.1) を介して **snmpd** エージェントで **dpid2** デーモンによって登録されるポート番号です。DPI2 サブエージェントは、 **snmpd** エージェントへ dpiPortForTCP.0 (1.3.6.1.4.1.2.2.1.1.1.0) インスタンスの get-request 照会を送信することにより、 **snmpd**

からのこのポート番号を確認します。DPI2 サブエージェントは、DPI2 エージェントが listen している TCP ポート番号を認識した後、そのポートへの接続を試みます。

dpid2 デーモンは通常、 /etc/rc.tcpip シェル・スクリプトが呼び出された際、システムの起動中に実行されます。

dpid2 デーモンは、 System Resource Controller (SRC) を使用して制御される必要があります。dpid2 のコマンド・ラインでの入力は推奨されません。

dpid2 デーモンを操作するには、次の SRC コマンドを使用します。

startsrc
サブシステム、サブシステム・グループ、またはサブサーバーを始動します。

stopsrc
サブシステム、サブシステムのグループ、またはサブサーバーを停止します。

refresh
サブシステムまたはサブシステム・グループが該当する構成ファイルを再読み取りします。

lssrc サブシステム、サブシステムのグループ、あるいはサブサーバーの状況を取得します。

注**: snmpdv3** エージェント自体は DPI2 エージェントとして動作し、dpiPortForTCP.0 TCP ポート上で listen します。したがって、**snmpdv3** エージェントを使用している場合、**dpid2** デーモンは必要ありません。そのため、 **dpid2** デーモンはシステム起動時に実行されず、 **/etc/rc.tcpip** 内の dpid2 行はコメント化されます。

フラグ

項目	説明
-d *Level*	トレースまたはデバッグのレベルを指定します。 **8** DPI レベル 1 **16** DPI レベル 2 **32** 内部レベル 1 **64** 内部レベル 2 **128** 内部レベル 3 複数トレース・レベルの番号を付加します。 **注: -d** フラグが指定されているが、レベル番号が指定されていない場合、デフォルト・レベルは 56 です。**-d** フラグが指定されない場合は、デフォルトのレベルは 0 です。

例

1. **dpid2** デーモンを開始するには、次のようなコマンドを入力します。

   ```
   startsrc -s dpid2 -a "-f /tmp/dpid2.log"
   ```

   このコマンドは、 **dpid2** デーモンを開始し、デバッグ・レベル 0 で情報を /tmp/dpid2.log ファイルへログを記録します。
2. 通常の方法で **dpid2** デーモンを停止するには、次のコマンドを入力します。

   ```
   stopsrc -s dpid2
   ```

   このコマンドは、 **dpid2** デーモンを停止させます。**-s** フラグは、後に続くサブシステムを停止させることを指定します。

3. **dpid2** デーモンから簡略状況を入手するには、次のコマンドを入力します。

   ```
   lssrc -s dpid2
   ```

   このコマンドは、デーモン名、デーモンのプロセス ID、デーモンの状態 (アクティブまたは非アクティブ) を戻します。

ファイル

項目	説明
/etc/snmpd.conf	**snmpd** バージョン 1 エージェント構成ファイル内の SMUX ピア・エントリーを指定します。
/etc/snmpd.peers	SMUX のピアの構成を指定します。
/etc/mib.defs	SNMP エージェントおよびマネージャーが認識して処理する必要のある MIB 変数が定義されています。

関連資料:

799 ページの『hostmibd デーモン』

関連情報:

snmpdv1 コマンド

---

dping コマンド

目的

ノードまたはデバイスを並行して ping します。

構文

dping [**-h**] [**-v**] [**-a**] [**-s**] [**-S**] [**-r**] [**-i** *interface...]* [**-w** " **selectstr**"] [**-H** *host_list*] [**-f** *filename*] [**-N** *nodegroup...*] [**-d** *devicename...*] [**-D** *devicegroup*] [[**-n**] *node_list*]

説明

dping コマンドは、指定されたサーバーを ping します。このコマンドは、ノードの状況を取得したり、Rational® Method Composer (RMC) およびそのハートビートに問題があると疑われる場合に使用できます。 **dping** コマンドは、ノードまたはデバイスを並行して ping するのに使用されます。代替ノード・インターフェース (例えば **eth1** または **mryi0**) すべてを順番に ping するために使用します。ノードを指示することによって、その他のノードを ping するために使用します。

キーワード

項目	説明
-a	すべてのノードを ping することを指定します。このフラグは、**-n**、**-N**、**-f**、**-d**、**-H**、または **-w** フラグと組み合わせることはできません。
-c	複数のノードからの同一の出力を省略して、出力を 1 回表示します。
-d *devicename...*	ping する 1 つ以上のデバイスをコンマで区切って指定します。アスタリスク文字 (*) は、すべてのデバイスを示します。
-D *devicegroup...*	ping する 1 つ以上のデバイス・グループをコンマで区切って指定します。
-f *filename*	ノードのリストを含むファイルを指定します。指定されたファイル名がダッシュ文字 (**-**) である場合、標準入力からリストが読み取られます。ファイルには複数の行を含めることができ、各行にコンマまたはスペースで区切られた 1 つ以上のノードのホスト名またはノード範囲をリストすることができます。
-h	コマンドの使用方法に関する情報を表示します。

項目	説明
-H *host_list*	ping するホスト名をコンマまたはスペースで区切って指定します。これらのホスト名は、定義済みの NIM ノードであってはなりません。スペースで区切られたホスト名は、二重引用符で囲んで指定する必要があります。**-H** フラグは、**-n**、**-N**、**-f**、**-d**、**-w**、または **-a** フラグと一緒に指定することはできません。
-i *interface..*	指定されたノードごとに、ping する 1 つ以上のネットワーク・インターフェースをコンマで区切って指定します。このフラグは、*nodename-interface.domain* がノード上のそのネットワーク・アダプターの IP アドレスに解決されることを想定します。**dping** を実行する前に、ホスト名の解決をセットアップしておく必要があります。いずれかのホスト名が空ストリングである場合 (例えば、**"eth1,eth2"**)、このフラグは 1 次ホスト名も ping します。
-n *node_list*	ping する 1 つ以上のノードのホスト名またはノード範囲をコンマまたはスペースで区切って指定します。このフラグは、**-N** および **-f** と同時に使用できます。ホスト名の値が指定される最後の引数である場合、**-n** を使用せずに指定できます。ノード範囲について詳しくは、noderange ファイルを参照してください。
-N *nodegroup..*	コマンドを実行する 1 つ以上のノード・グループをコンマで区切って指定します。
-r	再帰的に ping します。このフラグは、正常に ping されたノードに対して **dsh** コマンドを実行します。それらのノードから、**dping** コマンドで指定された残りのすべてのノードが ping されます。
-s	並行ではなく順次にノードを ping します。このフラグは、**-S** フラグと同時には使用できません。
-S	ping の結果の要約のみを表示します。このフラグは、**-s** フラグと同時には使用できません。
-v	詳細モードを指定します。
-w *selectstr*	選択文字列の "where" 節と一致するノードを表示します。文字列全体を二重引用符で囲んで指定すると、属性値の文字列を単一引用符で囲んで指定することができます。アスタリスク文字 (*) は、"where" 節が指定されなかった場合と同様にすべてのノードを示します。**-w** フラグは、**-n**、**-N**、**-f**、**-d**、**-H**、または **-a** フラグと同時には指定できません。

セキュリティー

このコマンドを実行するには、クラスター管理サーバーへの root アクセス権限が必要です。

例

1. すべてのノードを ping するには、次のように入力します。

   ```
   dping -a
   ```

   出力は以下のようになります。

   node1.localdomain: ping (alive)

   node2.localdomain: noping (unreachable)

   node3.localdomain: ping (alive)

2. **group1** ノード・グループ および **eth1** ノード・インターフェースを ping するには、次のように入力します。

   ```
   dping -N group1 -i eth1
   ```

   出力は以下のようになります。

   node1-eth1.localdomain: ping (alive)

   node2-eth1.localdomain: noping (unreachable)

3. ホスト名 **node1-eth2.clusters.com** を ping するには、次のように入力します。

   ```
   dping -i eth2 node1.clusters.com
   ```

   出力は以下のようになります。終了状況 0 (コマンドは正常に完了しました。)**1** (コマンドは失敗しました。)**10** (ノードまたはデバイスが指定されませんでした。)

drmgr コマンド

目的

drmgr コマンドは、動的ロジカル・パーティショニング (DLPAR) スクリプトをインストールおよび構成するために使用します。

構文

drmgr { **-i** *script_name* [**-w** *minutes*] [**-f**] | **-u** *script_name* } [**-D** *hostname*]

drmgr [**-b**]

drmgr [**-R** *script_install_root_directory*]

drmgr [**-S** *syslog_ID*]

drmgr [**-l**]

説明

DLPAR スクリプトは、オペレーティング・システムに関するリソースの追加や削除を行うことで、アプリケーションおよびミドルウェアによるリソースの消費 (例えば、特定のプロセッサーや大量のピン済みメモリーなど) を調整するために、システム管理者およびベンダーによって提供されます。DLPAR スクリプトは、 DLPAR 操作の前と後の両方で実行されます。 DLPAR スクリプトにより、アプリケーションは正確に静止して再始動できます。

注**:** 指定されたアクション・フラグを組み合わせることはできません。つまり、ユーザーは、**-R** と **-S** フラグ、 **-l** と **-R** フラグ、などを組み合わせることはできません。

フラグ

フラグ	説明
-b	**drmgr** コマンドによって管理される、スクリプトの情報ファイルを再作成します。一般に、このオプションは、別のシステムからスクリプトを復元するときにのみ使用する必要があります。
-D *hostname*	スクリプトを開始できるシステムのホスト名を指定します。
-f	既存のスクリプトの差し替えを強制実行します。
-i *script_name*	*script_name* スクリプトをインストールします。*script_name* は、完全パスを備えている必要があります。このパスが指定されない場合は、現行ディレクトリーが想定されます。名前に何らかの矛盾がある場合、**drmgr** コマンドは警告を出し、スクリプトをインストールしません。 **-f** フラグの指定により、既存のいかなるスクリプトも上書きすることができます。
-l	現在インストールされている DLPAR スクリプトに関する詳細を表示します。
-R *base_script_directory*	ベース・スクリプト・インストール・ディレクトリーを変更します。
-S *syslog_ID*	指定された syslog ID ストリングを使用して、syslog メッセージをログに記録します。この ID ストリングは、**drmgr** によって syslog にログとして記録されるすべてのエントリーに付加されます。
-u *script_name*	DLPAR スクリプトをアンインストールします。そのスクリプトが **-D** オプションを使用してインストールされたものである場合は、そのスクリプトのアンインストールに同じパラメーターを使用する必要があります。ディレクトリーが指定されなかった場合、**drmgr** コマンドは DLPAR スクリプトをインストール・ディレクトリー「全体」から除去します。
-w *minutes*	そのスクリプトのベンダーによって指定された時間制限値を指定変更します。スクリプトは、指定された制限時間を超えると停止されます。

終了状況

0 要求された操作を正常に完了しました。

>0 コマンドが失敗しました。失敗の原因として、以下のことが考えられます。

- ファイルまたはディレクトリーが存在しない。
- パラメーターの長さがシステムしきい値 (PATH_MAX) を超えている。
- 指定された引数が多すぎる。
- このコマンドを実行するための root 権限がない。

関連情報:

動的ロジカル・パーティショニング

---

drslot コマンド

目的

動的に再構成可能なスロット (例えばホット・プラグ・スロット) を管理します。

構文

ホット・プラグ・スロットを識別するには

、次のように入力します。

drslot -i { -s *Slot* | **-l** *DeviceName* } **-c** *ConnectorType*

デバイスの構成のためにホット・プラグ・スロットを作成するには

、次のように入力します。

drslot -a -s *slot* **-c** *ConnectorType* [**-I**]

デバイスの除去のためにホット・プラグ・スロットを作成するには

、次のように入力します。

drslot -r { **-s** *slot* | **-l** *DeviceName*} **-c** *ConnectorType* [**-I**]

デバイスの除去および置換のためにホット・プラグ・スロットを作成するには

、次のように入力します。

drslot -R { **-s** *slot* | **-l** *DeviceName* } **-c** *ConnectorType* [**-I**]

説明

drslot コマンドは、動的に再構成可能なスロット、つまりホット・プラグ・スロットを管理します。ホット・プラグ・スロットとは、システムの電源をオフにしたり、オペレーティング・システムをリブートしたりしなくても構成できるエンティティーを接続するための、プラグイン・ポイントです。追加 (**-a**) 操作の場合は、**-s** フラグを使用して、スロットに固有 ID を割り当てる方法で、そのスロットを直接指定しなければなりません。識別 (**-i**)、除去 (**-r**)、および置換 (**-R**) 操作の場合は、スロットを、 **-s** フラグを使用して直接指定するか、あるいは間接的に指定する必要があります。スロットに接続されているデバイスに論理名を割り当てる **-l** フラグを使用すると、スロットを間接的に指定することができます。 **drslot** コマンドは、指定されたデバイスが接続されているスロットを判別し、そのスロットを管理します。

注**:**

1. 除去および置換操作は失敗に終わります。ただし、識別されたスロットに接続されたデバイスが構成解除されている場合を除きます。デバイスを正常に構成解除する方法に関する詳細は、「オペレーティング・システムおよびデバイスの管理」の『ホット・プラグ・コネクターの管理』を参照してください。
2. 追加または置換操作の後には、新規デバイスをアクティブかつ作動可能にして、オペレーティング・システムで使用できるようにするために、 **cfgmgr** コマンドを実行しなければなりません。

フラグ

注**: -a** フラグ、 **-i** フラグ、**-r** フラグ、 **-R** フラグは同時に使用できません。

項目	説明
-a	ホット・プラグ・スロットが、そこに接続されるデバイスを構成できるようにします。まずスロットが識別され、スロットの確認を求めるプロンプトが出されます。次に、デバイスがそのスロットに接続されていることを確認するプロンプトが出されます。デバイスが接続されていることが確認されれば、スロットの準備ができ、デバイスの構成が可能になります。
-c *ConnectorType*	操作中の *Slot* の *ConnectorType* を指定します。例えば、ホット・プラグ PCI スロットの *ConnectorType* は、`pci` です。このフラグは、**-a** フラグ、**-i** フラグ、**-r** フラグ、および **-R** フラグと同時に指定しなければなりません。
-i	ホット・プラグ・スロットを識別します。スロットの識別はハードウェアに依存します。例えば、スロットがそれに関連した LED を備えている場合は、 **drslot -i** コマンドを送出すると、 LED が点滅することがあります。
-I	**-a** (追加)、**-r** (除去)、および **-R** (置換) フラグを使用する場合は、識別ステップをスキップするように指定します。このフラグは、既に適切なスロットを識別済みであることが確認できる場合にのみ、使用するようにしてください。
-l *DeviceName*	*DeviceName* を指定します。これは、管理対象のスロットに接続されているデバイスの論理デバイス名です。 **-s** フラグが使用されない場合は、このフラグを、**-i** (識別)、**-r** (除去)、または **-R** (置換) フラグとして使用しなければなりません。
-r	**rmdev** コマンドまたは SMIT の同等のコマンドを使用してあらかじめ構成解除してあるデバイスを除去するためのホット・プラグ・スロットを準備します。まずスロットが識別され、スロットの確認を求めるプロンプトが出されます。そのスロットに、ビジュアル・インジケーターが関連付けられている場合は、それをオフにします。最後に、スロットからデバイスを除去する準備が行われ、デバイスがスロットから除去されたことを確認するプロンプトが出されます。
-R	あらかじめ構成解除してあるデバイスを除去し、同一のデバイスに置き換えるために、ホット・プラグ・スロットを準備します。デバイスは、**rmdev** コマンドまたは SMIT の同等のコマンドを使用して構成解除する必要があります。 **drslot** がスロットを識別し、ユーザーにはそのスロットの確認を求めるプロンプトが出されます。次に、デバイスを置換するためのスロットを準備します。準備ができると、デバイスが置換されたことを確認するプロンプトが出されます。デバイスがホット・プラグ・スロットで置換されたことが確認されれば、スロットの準備ができ、デバイスの構成が可能になります。
-s *Slot*	**drslot** が働く *Slot* を指定します。このフラグは、追加 (**-a**) 操作の場合は必須です。 **-l** フラグが使用されていない場合は、このフラグを識別 (**-i**)、除去 (**-r**)、または置換 (**-R**) 操作に使用しなければなりません。 *Slot* の形式は、プラットフォームあるいは *ConnectorType* に依存します。

例

1. 特定の PCI ホット・プラグ・スロットを識別するには、次のように入力します。

   ```
   drslot  -i  -c pci  -s U0.1-P1-I3
   ```

   この例では、このスロットに LED が関連付けられています。システムが次のようなメッセージを表示することがあります。

   ```
   The visual indicator of the specified PCI slot has been set to the identify state.Press
   Enter to continue or enter x to exit.
   ```

   `U0.1-P1-I3` によって指定されたスロットの LED は、 Enter キーを押すまで点滅します。

2. ホット・プラグ・スロットに、そのスロットを確認しないでホット接続可能なイーサネット・アダプターを追加するには、次のように入力します。

   ```
   drslot  -a  -I  -c pci  -s U0.1-P1-I3
   ```

   スロットを識別する確認のプロンプトは出されません。確認プロンプトは新しいアダプターをスロットに入れるべきときに表示され、次のようなメッセージが表示されます。

   ```
   The visual indicator for the specified PCI slot has been set to the action state.Insert the
   PCI card into the identified slot, connect any devices to be configured, and press Enter to
   continue.Enter x to exit.
   ```

   アダプターを接続後、Enter を押せば、スロットの準備は完了です。
3. 特定の PCI スロットを、そのスロットの SCSI カードを取り替える前に識別するには、次のように入力します。

   ```
   drslot  -R  -c pci  -s U0.2-P1-I3
   ```

   システムは次のようなメッセージを表示します。

   ```
   The visual indicator of the specified PCI slot has been set to the identify state.Press
   Enter to continue or enter x to exit.
   ```

   PCI スロットの LED が明滅するので、そのスロットが識別されます。 Enter キー以外の任意のキーを押すと、コマンドを終了します。 Enter キーを押すと、このスロットの操作が継続されます。継続すると、PCI スロットの LED がアクション状態に変わり、システムは、次のようなメッセージを表示します。

   ```
   The visual indicator for the specified PCI slot has been set to the action state.Replace the
   PCI card in the identified slot, reconnect any devices to be configured, and press Enter to
   continue. Enter x to exit.Exiting now leaves the PCI slot in the removed state.
   ```

ファイル

/usr/sbin/drslot

関連情報:

lsslot コマンド

rmdev コマンド

cfgmgr コマンド

PCI ホット・プラグ管理

dscrctl コマンド

目的

オペレーティング・システムのデフォルトのプリフェッチ特性を設定します。

構文

コンピューターのハードウェア・ストリームの特性を照会するには、次のように入力します。

dscrctl -q

コンピューター上のオペレーティング・システムのデフォルトのプリフェッチの深さを一時的 (現行セッションのみ) または永続的 (再始動操作ごと) に設定するには、次のように入力します。

dscrctl [-n] [-b] -s *dscr_value*

始動時にオペレーティング・システムのデフォルトのプリフェッチの深さに対する永続設定を取り消すには、次のように入力します。

dscrctl -c

説明

dscrctl -q サブコマンドは、ハードウェア・ストリームの数、およびプラットフォームとオペレーティング・システムのデフォルトのプリフェッチの深さを表示します。このサブコマンドはすべてのユーザーが実行できます。

dscrctl -s サブコマンドは、オペレーティング・システムのデフォルトのプリフェッチの深さを設定します。このサブコマンドを実行するには、root 権限を持っている必要があります。このデフォルト値を変更するには、現行セッションに対しては **-n** フラグ、始動時には **-b** フラグ、あるいは現行セッションと始動時の両方に対しては **-n -b** フラグを **dscrctl** コマンドと一緒に使用します。

dscrctl -c オプションにより、始動時にオペレーティング・システムのデフォルトのプリフェッチの深さ設定が取り消されます。このオプションは **dscrctl** コマンドを **/etc/inittab** ファイルから除去することにより、次回の再始動後に有効になります。

フラグ

-q プラットフォームでサポートされるハードウェア・ストリームの数、およびファームウェアとオペレーティング・システムのデフォルトのプリフェッチの深さを表示します。

-c **dscrctl** コマンドを **/etc/inittab** ファイルから除去することにより、始動時にデフォルトのプリフェッチの深さの永続設定を取り消します。

-n オペレーティング・システムのデフォルトのプリフェッチの深さに対する実行時の値を変更します。このフラグは **-s** フラグと一緒に使用します。この変更はブート操作後は持続されません。

-b **dscrctl** コマンドを **/etc/inittab** ファイルに追加することにより、ブート操作後も変更を持続させます。このフラグは **-s** フラグと一緒に使用します。

-s *dscr_value*
オペレーティング・システムのデフォルトのプリフェッチの深さに対する値を新規に定義します。この値は、0x で始まっている場合を除き (この場合は 16 進数として扱われます)、10 進数として扱われます。

例

1. 現行セッションに対してオペレーティング・システムのデフォルトのプリフェッチの深さに対する値を 13 に設定するには、次のように入力します。

   ```
   # dscrctl -n -s 13
   ```

2. ハードウェア・ストリーム・メカニズムの現在の設定を表示するには、次のように入力します。

```
# dscrctl -q
```

次の出力が表示されます。

```
Current DSCR settings:
Data Streams Version = V2.06
number_of_streams = 16
platform_default_pd = 0x5 (DPFD_DEEP)
os_default_pd = 0xd (DSCR_SSE | DPFD_DEEP)
```

関連情報:

/etc/inittab ファイル

---

dscreen コマンド

目的

ダイナミック・スクリーン・ユーティリティーを始動します。

構文

dscreen [**-i** *InfoFile*] [**-t** *TermType*]

説明

dscreen コマンドは、ダイナミック・スクリーン・ユーティリティーを始動します。これにより単一の物理端末を、一度に複数の仮想セッションまたはスクリーンに接続します。

フラグが指定されないと、**dscreen** コマンドは **TERM** 環境変数で指定された端末の記述を、**DSINFO** 環境変数で指定されたファイルから読み取ります。 **DSINFO** 環境変数が指定されていない場合には、端末の記述を **/etc/dsinfo** ファイルから読み取ります。端末の記述には、通常次の構成情報が入っています。

- ダイナミック・スクリーン・ユーティリティーで使用するキーとその機能。
- 端末が使用できるスクリーン・メモリーのページ数。
- ダイナミック・スクリーン・ユーティリティーにアクセスし、それを使用するために、送信または受信する必要のあるコード・シーケンス。

フラグ

項目	説明
-i *InfoFile*	ダイナミック・スクリーン・ユーティリティーで使用する代替キー・マッピングを含むファイルを指定します。このオプションは、最初に定義したダイナミック・スクリーン・キーがユーザーのアプリケーションの 1 つと矛盾する場合に役に立ちます。 このフラグを指定しないと、**DSINFO** 環境変数が設定されていれば、端末構成情報はそこで指定されたファイルから読み取られます。そうでなければ、その情報は **/etc/dsinfo** ファイルから読み取られます。
-t *TermType*	キー・マッピングを含むファイルから端末記述が読み取られるよう指定します。このオプションは、希望する端末タイプが **TERM** 環境変数での設定と一致しないときに役に立ちます。

例

1. キー・マッピングのデフォルトを使用して、ダイナミック・スクリーン・ユーティリティーを始動するには、次のように入力します。

```
dscreen
```

これにより、デフォルトの **/etc/dsinfo** ファイルで指定されているとおりに、**DSINFO** 環境変数と **TERM** 環境変数が設定されます。

2. ダイナミック・スクリーン・ユーティリティーを始動して、代替キー・マッピングを含むファイルを指定し、そのファイルから読み取られる端末記述を識別するには、次のように入力します。

   ```
   dscreen -i myfile -t myterm
   ```

   これにより、ユーザーが作成した `myinfo` という名前の、 **dsinfo** タイプのファイルからの情報を使用して、通常は必要のないキー・マッピングを処理します。また、`myinfo` ファイルには `myterm` と名付けられた端末定義も入っています。

3. ダイナミック・スクリーン・ユーティリティーを始動し、代替端末設定を指定するには、次のように入力します。

   ```
   dscreen -t wy60-wp
   ```

   この端末定義 (**/etc/dsinfo** ファイルに定義されている) によって、**dscreen** は使用中のワード・プロセッサー機能を持つアプリケーションの制御キーのコマンド・シーケンスと矛盾しないようなキー・アクションを割り当てるように設定します。

ファイル

項目	説明
/etc/dsinfo	ダイナミック・スクリーン・ユーティリティー用の端末記述が入っています。

関連情報:

ダイナミック・スクリーン・ユーティリティー

---

dshbak コマンド

このコマンドは、IBM Distributed Shell Management (DSM) ソフトウェアの一部です。このコマンドは、*/opt/ibm/sysmgt/dsm/bin/dshbak* ロケーションにあります。

目的

dsh コマンドからのフォーマットされた出力を表示します。

構文

dshbak [-c | -x]

説明

dshbak コマンドは、dsh コマンドからの出力をフォーマットします。**dshbak** コマンドの構文は次のとおりです。

host_name: リモート・コマンドからの出力行

dshbak コマンドは、行をフォーマットして、それらを次のように標準出力に書き込みます。*host_name3* および *host_name4* からの出力は同一で、**-c** フラグが指定されているものとします。

```
 HOSTS -------------------------------------------------------
          host_name1
          -------------------------------------------------------------
          .
          .
```

```
lines from dsh with host_names stripped off
.
.
HOSTS -------------------------------------------------------
host_name2
-------------------------------------------------------------
.
.
lines from dsh with host_names stripped off
.
.
HOSTS -------------------------------------------------------
host_name3             host_name4
-------------------------------------------------------------
.
.
lines from dsh with host_names stripped off
.
.
```

複数ノードからの出力が省略形式で表示される場合、ホスト名はアルファベット順で表示されます。出力が省略されない場合、出力はホスト名別にアルファベット順でソートされます。**dshbak** コマンドは、フィルターに掛けられた出力の 1000 行ごとに「.」を書き込みます。

-x フラグが指定される場合、**dshbak** コマンドがノードごとに表示する追加のヘッダー行は除外されます。**dshbak** コマンドは、次のように、ノード名を使用して出力をソートし、内容を表示します。

```
host_name1: lines from dsh started
            .
            .
            lines from dsh continued
            .
            .
            lines from dsh ended
            host_name2: lines from dsh started
            .
            .
            lines from dsh continued
            .
            .
            lines from dsh ended
```

フラグ

項目	説明
-c	複数のノードからの同一の出力を省略して、出力を 1 回で表示します。
-x	**dshbak** がノードごとに表示する追加のヘッダー行を除外します。このフラグは短縮出力を提供し、**dshbak** コマンドは出力をノード名別にソートして、内容を表示します。このフラグを **-c** と同時に使用してはなりません。

セキュリティー

注**:** Kerberos バージョン 5 のリモート・コマンドを実行する前に、**kinit** コマンドを実行してチケット許可チケットを取得する必要があります。その他のセキュリティーに関する考慮事項は、リモート・シェル・コマンドと同様です。

例

1. 複数のノードで発行されたコマンドの結果を『説明』で使用されたフォーマットで表示するには、次のコマンドを入力します。

   ```
   dsh -n node1,node2,node3 cat /etc/passwd | dshbak
   ```

2. 複数のノードで発行されたコマンドの結果を同一の出力で表示するには、次のコマンドを入力します。
   ```
   dsh -w host1,host2,host3 pwd | dshbak -c
   ```
3. 複数のノードで発行されたコマンドの結果を短縮出力で表示するには、次のコマンドを入力します。

   注: 出力は、ホスト名別にアルファベット順でソートされます。
   ```
   dsh -w host1,host2,host3 date | dshbak -x
   ```

標準エラー

dshbak フィルターが使用された場合、標準エラーのエラー・メッセージがすべての標準出力メッセージの前に表示されます。 **-c** フラグを指定する場合も指定しない場合も、同じ動作になります。

---

dsh コマンド

目的

複数のノードおよびハードウェア・デバイスでコマンドを並行して実行します。

構文

dsh -h dsh -V dsh -q dsh [-a] [--all-nodes *context_list*]

[**-A**] [**--all-devices** *context_list*] [**-n**

node_list] [**-N** *nodegroups*] [**-d** *device_list*]

[**-D** *devicegroups*] [**-C** *context*]

[**-c**] [**-e**] [**-E** *environment_file*

] [**-f** *fanout*] [**-F** *output_path*]

[**-i**] [**-l** *user_ID]* [**-L**]

[**--log** *log_file*] [**-m**] [**-o**

node_options] [**-O** *device_options*] [**-Q**]

[**-r** *node_remote_shell*] [**--device-rsh**

device_remote_shell] [**-s**] [**-S**

csh | **ksh**] [**-t** *timeout*]

[**-T**] [**-v**] [**-X**

env_list] [**-z**] [**--report**

report_path] [**--report-name** *report_name*] [*command_list*]

説明

dsh コマンドは、ノード、ハードウェア・デバイス、あるいは両方のリモート・ターゲットでコマンドを並行して実行します。ターゲットは、複数のコンテキストから選択できます。コンテキストは、ノードおよ

びデバイスの定義を含むターゲット・データベース (NIM データベースなど) です。 **dsh** コマンドは、指定されたターゲットごとにリモート・シェル・コマンドを発行します。すべてのノードからのコマンドの結果を簡単に管理できるように、フォーマットされた方法ですべてのターゲットからの出力を返します。**/usr/bin/rsh** は構文およびセキュリティーのモデルであり、dsh コマンドは DSM 分散シェル・ユーティリティーです。

パラメーター

項目	説明
ターゲット・コンテキスト	**dsh** コマンドのターゲット・コンテキストは、ターゲットまたはターゲット・グループが定義されているデータベースです。**-C** コンテキスト・フラグまたは **DSH_CONTEXT** 環境変数を使用してデフォルトのコンテキストを構成できます。いずれのパラメーターも指定されない場合、dsh コマンドが NIM 管理サーバーから実行されるときのデフォルトのコンテキストは **NIM** で、それ以外の場合のデフォルトのコンテキストは **DSH** です (**DSH** コンテキスト を参照)。コンテキスト拡張ファイルを **/opt/ibm/sysmgt/dsm/pm/Context** ディレクトリーにインストールすることにより、コンテキストは **DSH** ユーティリティー・コマンドで使用されます。ターゲットまたはターゲット・グループのコンテキストは、ターゲット名をコンテキスト名で修飾することによって明示的に指定でき、非修飾ターゲット名にデフォルトのコンテキストを指定することによって暗黙的に定義できます (『ターゲット・リスト』を参照)。
DSH コンテキスト	**DSH** コンテキストは、すべての **DSH** ユーティリティー・コマンド用の組み込みコンテキストです。これは、ローカル・ファイル・システムに含まれるユーザー定義ノード・グループ・データベースを許可します。**DSH_NODEGROUP_PATH** 環境変数は、ノード・グループ・データベースへのパスを指定します。このディレクトリー内の各ファイルは、ノード・グループを表し、グループ・メンバーであるノードごとに 1 つのホスト名または TCP/IP アドレスを含みます。ブランク行および # 記号で始まるコメント行は無視されます。**DSH** コンテキストのすべてのノードが要求された場合、**DSH_NODEGROUP_PATH** ディレクトリー内のすべてのグループから完全なノード・リストが作成され、**/var/ibm/sysmgt/dsm/dsh/$DSH_NODEGROUP_PATH/AllNodes** のキャッシュに入れられます。このファイルは、グループ・ファイルが変更されたり、**DSH_NODEGROUP_PATH** ディレクトリーに追加されるたびに再作成されます。デバイス・ターゲットは、**DSH** コンテキストではサポートされません。
ターゲットの指定	ターゲットは、リモート・コマンドが発行されるノードまたはハードウェア・デバイスです。ノード・ターゲットは、**-a**、**--all-nodes***context_list*、**-n** *node_list*、および **-N** *nodegroups* フラグ、または **DSH_NODE_LIST** 環境変数を使用して指定されます。 **-N** フラグと **DSH_NODE_LIST** 環境変数の両方が使用されている場合、グループおよびリストは、重複がある場合はすべてそれを除去してマージされます。**注: WCOLL** は **DSH_NODE_LIST** 環境変数によって置き換えられました。デバイス・ターゲットは、**-A**、**--all-devices** *context_list*、**-d** *device_list*、および **-D** *devicegroups* フラグ、または **DSH_DEVICE_LIST** 環境変数を使用して指定されます。 ターゲットの一部としてローカル・ホストが含まれる場合、*command_list* は、構成済みのリモート・シェルを介してではなくローカル・ホストで直接的に実行されます。ただし、ローカル・ホストでの実行のために *user_ID* が指定されている場合は除きます (「Remote user」を参照)。**DSH_NODE_LIST** および **DSH_DEVICE_LIST** 環境変数は、ターゲット・ノードおよびターゲット・デバイスをリストするファイルを指定します。ファイル・フォーマットは 1 行ごとに 1 ターゲットです。ブランク行および # 記号で始まるコメント行は無視されます。ノードとデバイスの両方のターゲットを同時に指定できますが、デバイスとノードの両方に同じターゲット名を使用することはできません。類似の名前が使用された場合、プログラムは重複するターゲットをスキップして、他のターゲットで実行を続行します。ノードおよびデバイスのターゲットは、ノード範囲を使用して指定することもできます。詳しくは、**noderange** ファイルを参照してください。複数回にわたって同じターゲットが指定された場合、リモート・コマンドは指定されたターゲットで 1 回実行されます。

項目	説明
ターゲット・リスト	ターゲットおよびターゲット・グループは次のフォーマットを使用して指定されます。 [*context*:] [*user_ID*@] *target* [, [*context*:] [*user_ID*@] *target*]... ここで、context はターゲットの明示的なコンテキスト指定であり、*user_ID* はリモート側からコマンドをターゲットで実行する場合に使用するオプションのユーザー名であり、*target* はターゲットのコンテキストで許可されたターゲットの名前または TCP/IP アドレスです。**noderange** 式の場合、**noderange** 式の計算の結果であるリスト内の各ターゲットに対して *user_ID* が使用されます。ターゲット・リストがダッシュ **(-)** のみを使用して指定される場合、ターゲットを対話式に指定することができます。ノードの場合のプロンプトは **dsh node>** で、デバイスの場合のプロンプトは **dsh device>** です。一度に 1 行で次の構文を使用してターゲット・リストを指定します。 [*context*:] [*user_ID*@] *target*ここで、context はターゲットの明示的なコンテキスト指定であり、*user_ID* はリモート側からコマンドをターゲットで実行する場合に使用するオプションのユーザー名であり、target はターゲットのコンテキストで許可されたターゲットの名前または TCP/IP アドレスです。**noderange** 式の場合、**noderange** 式の計算の結果であるリスト内の各ターゲットに対して *user_ID* が使用されます。ターゲット・リストがダッシュ (-) のみを使用して指定される場合、ターゲットを対話式に指定することができます。ノードの場合のプロンプトは **dsh node>** で、デバイスの場合のプロンプトは **dsh device>** です。一度に 1 行で次の構文を使用してターゲット・リストを指定します。 [*context*:][*user_ID*@]*target* 完了したら、Ctrl-d を押して続行します。
コマンドの指定	リモート・ターゲットで実行するコマンドは、*command_list* dsh パラメーターによって指定されます。このパラメーターは、対話モードのコマンド・ラインでコマンドを入力するか、標準入力から *command_list* を提供するか、あるいは **-e** フラグを使用してローカル・スクリプトを実行することで指定できます。*command_list* dsh パラメーターの構文は "**command[; command]...**" です。ここで、command は、リモート・ターゲットで実行するコマンドです。リスト内のすべてのコマンドがリモート側で実行され、特殊文字がリモート・ターゲットで正しく解釈されるようにするために、引用符が必要です。**-e** フラグを使用することにより、ローカル・ホスト上のスクリプト・ファイルが各リモート・ターゲットで実行されます。**-e** が指定される場合、*command_list* はスクリプト名およびスクリプトに対する引数です。次に例を示します。 `dsh -e[flags] script_filename [arguments]...` *script_filename* ファイルは、各リモート・ターゲット上の **/tmp** ディレクトリーのランダム・ファイル名にコピーされ、その後でターゲット上で実行されます。*command_list* パラメーターが指定されない場合、**dsh>** は対話式コマンド・ライン・モードになり、dsh> プロンプトを出します。 **dsh>** プロンプトで **[!] "command"** 構文を使用してコマンドを入力します。ここで、command は、リモート・ターゲットで実行するコマンドです。コマンドの前に感嘆符 (!) を指定すると、コマンドはローカル・ホストでのみ実行され、リモート・ターゲットでは実行されません。dsh コマンドは解決されたターゲットでコマンドを実行し、その結果が表示されます。その後、dsh> プロンプトに戻ります。コマンド・ライン・モードを終了するには、dsh> プロンプトで exit と入力します。dsh コマンドは、標準入力から読み取られるコマンドも含めて、いずれの対話式コマンドとも連動しません。
リモート・ユーザー	リモート・ターゲット用に使用する *user_ID* は、ターゲットの構文の一部として指定するか (『ターゲット・リスト』を参照)、**-l** (小文字の L) フラグを使用して指定することができます。両方の方法が使用された場合、*user_ID* は次のように決定されます。 1. *user_ID@target* として指定されるターゲットの場合、ターゲットでのリモート実行のために *user_ID* が使用され、*-l* フラグは無視されます。 2. *user_ID@target* により指定されていないターゲットの場合、ターゲットでのリモート実行に使用される *user_ID* は、*-l* フラグを使用して指定された **user_ID** となります。 **-l** が指定されない場合、コマンドを実行している現行ユーザーが指定されます。ターゲット・リストに含まれるローカル・ホスト用に *user_ID* が指定されている場合、リモート・シェル・コマンドは、セキュア・ログイン ID を確認するためにローカル・ホストで *command_list* を実行します。

項目	説明
リモート・シェル・コマンド	リモート・ターゲットで実行するコマンドは、*command_list* dsh パラメーターによって指定されます。このパラメーターは、対話モードのコマンド・ラインでコマンドを入力するか、標準入力から *command_list* を提供するか、あるいは **-e** フラグを使用してローカル・スクリプトを実行することで指定できます。 *command_list* dsh パラメーターの構文は **"command[; command]..."** です。ここで、command は、リモート・ターゲットで実行するコマンドです。リスト内のすべてのコマンドがリモート側で実行され、特殊文字がリモート・ターゲットで正しく解釈されるようにするために、引用符が必要です。**-e** フラグを使用することにより、ローカル・ホスト上のスクリプト・ファイルを各リモート・ターゲットで実行できます。**-e** が指定される場合、**command_list** はスクリプト名およびスクリプトに対する引数です。例: dsh *-e[flags] script_filename [arguments]... script_filename* ファイルは、各リモート・ターゲット上の **/tmp** ディレクトリーのランダム・ファイル名にコピーされ、その後でターゲット上で実行されます。**command_list** パラメーターが指定されない場合、dsh は対話式コマンド・ライン・モードになり、dsh> プロンプトを表示します。 dsh> プロンプトで **[!] "command"** 構文を使用してコマンドを入力します。ここで、command は、リモート・ターゲットで実行するコマンドです。コマンドの前に感嘆符 (!) を指定すると、コマンドはローカル・ホストでのみ実行され、リモート・ターゲットでは実行されません。dsh コマンドは解決された各ターゲットに対して各コマンドを実行し、結果が表示されて、dsh> プロンプトに戻ります。コマンド・ライン・モードを終了するには、dsh> プロンプトで exit と入力します。dsh コマンドは、標準入力から読み取られるコマンドも含めて、いずれの対話式コマンドとも連動しません。
リモート・ユーザー	リモート・ターゲット用に使用する **user_ID** は、ターゲットの構文の一部として指定するか (『ターゲット・リスト』を参照)、**-l** (小文字の L) フラグを使用して指定することができます。両方の方法が使用された場合、*user_ID* は次のように決定されます。 1. *user_ID@target* として指定されるターゲットの場合、ターゲットでのリモート実行のために *user_ID* が使用され、*-l* フラグは無視されます。 2. *user_ID@target* により指定されていないターゲットの場合、ターゲットでのリモート実行に使用される *user_ID* は、*-l* フラグを使用して指定された *user_ID* となります。**-l** が指定されない場合は、コマンドを実行している現行ユーザーになります。ターゲット・リストに含まれるローカル・ホスト用に **user_ID** が指定される場合、リモート・シェル・コマンドは、セキュア・ログインを確認するためにローカル・ホストで *command_list* を実行します。
REMOTE SHELL ENVIRONMENT	リモート・ターゲットで使用されるシェル環境のデフォルトは、リモート・コマンド実行に使用される **user_ID** 用に定義されたシェルです。リモート・コマンド実行に使用されるコマンド構文は、**-S** フラグを使用して指定できます。 **-S** が指定されない場合、構文はデフォルトで ksh 構文になります。コマンドがリモート・ターゲットで実行される際に使用されるパスは、現行ユーザーのシェルで定義されている **DSH_PATH** 環境変数によって決定されます。**DSH_PATH** が設定されていない場合、使用されるパスはリモート・シェルのデフォルトのパスです。例えば、リモート・ターゲットのローカル・パスを設定するには、**DSH_PATH=$PATH** を使用します。 **-E** フラグは、ローカル環境定義ファイルを各リモート・ターゲットにエクスポートします。このファイルで指定されている環境変数は、*command_list* の実行前にリモート・シェル環境で定義されます。

項目	説明
コマンドの実行	並行リモート・シェル・コマンド・プロセスの最大数 (ファンアウト) は、**-f** フラグまたは **DSH_FANOUT** 環境変数を使用して指定できます。ファンアウトは、並行して実行できるリモート・シェル・コマンドの数によって制限されます。管理サーバーで **DSH_FANOUT** 値を使用して試し、さらに高い値が適切であるかどうかを調べることができます。リモート・コマンド実行のタイムアウト値は、**-t** フラグまたは **DSH_TIMEOUT** 環境変数を使用して指定できます。いずれかのリモート・ターゲットがタイムアウト値の時間内に標準出力または標準エラーのいずれかに出力を行わない場合、dsh コマンドはエラー・メッセージを表示して終了します。**-s** フラグを使用してストリーム・モードが指定される場合、出力は、各ターゲットから使用可能になった時点で返されます。このプロセスでは、出力を戻す前に *command_list* がすべてのターゲットで完了するのを待機しません。これにより、パフォーマンスが改善されますが、出力はソートされません。**-z** フラグは、*command_list* の中で最後にリモート・ノードで発行されたコマンドからの終了コードを表示するために使用されます。 **注:** OpenSSH は、最後に発行されたリモート・コマンドの終了状況をその終了状況として返します。これは、dsh の動作に影響を与えます。また、**-c** フラグを使用する必要があります。リモート・ノードで発行されたコマンドがバックグラウンドで実行された場合、コマンドは終了状況を表示しません。 **-m** フラグは、状況メッセージを標準出力に表示することによって、dsh コマンドの実行をモニターします。それぞれの状況メッセージの前には dsh> が付けられます。**-T** フラグは、dsh コマンド実行に関する診断トレース情報を提供します。デフォルトの設定値とリモート・ターゲットで実行された実際のリモート・シェル・コマンドが表示されます。リモート・ターゲットに対して、エラー検出またはリカバリー・メカニズムは提供されません。標準エラーおよび標準出力への dsh コマンド出力を分析して、適切な一連のアクションを判別することができます。対話モードでは、コマンドをリモート・ターゲットで実行できない場合 (例えば、リモート・シェル・コマンドの結果がゼロ以外の戻りコードである場合)、**-c** フラグが指定されていないと、dsh コマンドのこの呼び出しでは、後続のコマンドはこのノードに送信されません。
コマンドの出力	**dsh** コマンドは、各リモート・シェル・プロセスからの出力を表示するために待機して、その後で新しいリモート・シェル・プロセスを開始します。このデフォルトの動作は、**-s** フラグによって指定変更されます。dsh コマンド出力は、リモート・コマンドからの標準エラーおよび標準出力で構成されます。dsh 標準出力は、リモート・シェル・コマンドからの標準出力です。 dsh 標準エラーは、リモート・シェル・コマンドからの標準エラーです。各行の前には、出力を生成したノードのホスト名が付けられています。ホスト名の後に、: 文字およびコマンド出力行が続きます。ノード別にグループ化された同一の出力を表示するためのフィルターは別々に提供されています。詳しくは、**dshbak** コマンドを参照してください。各ターゲットの出力は、**-F** *output_path* フラグを使用してファイルにコピーできます。各ターゲットの標準出力は *output_path* ディレクトリー内の *target.output* ファイルに書き込まれ、各ターゲットの標準エラーは **output_path** ディレクトリー内の *target.error* ファイルに書き込まれます。**-F** フラグは、コンソール上の出力を抑止しません。**-Q** フラグを使用してコマンドをサイレント実行することができます。その場合、各ターゲットの標準出力または標準エラーの出力は表示されません。**-F** フラグが指定される場合、出力は引き続き出力ファイルに書き込まれます。
レポート	dsh コマンドからの出力をローカル・ホスト上のレポートに保存できます。**--report** *report_path* フラグを指定すると、指定された *report_path* ディレクトリーにレポートを生成できます。レポートは、**DSH_REPORT** 環境変数を *report_path* で定義することによって活動化されます。**--report** フラグは、**DSH_REPORT** 環境変数を指定変更します。**--report-name** フラグは、レポートが活動化されている場合にレポート名を定義します。レポート名は、レポート・ファイルを含む *report_path* のサブディレクトリーでもあります。複数のレポートに同じ名前を付けられるように、サブディレクトリー名に数値索引が付加されます。**--report-name** フラグが使用されない場合、名前はデフォルトで Unspecified になります。XML 結果ファイルのほか、HTML および XML の要約レポート・ファイルが作成されます。シグナル: シグナル 2 (INT)、シグナル 3 (QUIT)、およびシグナル 15 (TERM) が、リモート・ターゲットで実行されているコマンドに伝搬されます。シグナル 19 (CONT)、シグナル 17 (STOP)、およびシグナル 18 (TSTP) のデフォルトは dsh です。通常、dsh コマンドはこれらのシグナルに応答しますが、これらのシグナルはリモート側で実行されているコマンドには影響を与えません。その他のシグナルは dsh によって決定され、デフォルトで dsh コマンドに対して影響を与えます。リモート側で実行されているコマンドへの伝搬により、すべての現行の子プロセスは終了します (SIGTERM)。パラメーター *command_list* リモート・ターゲットで実行するコマンドのリストを指定します。*command_list* パラメーターの構文は次のとおりです。" **command[; command...**"

キーワード

項目	説明
-a	デフォルトのコンテキストで定義されたすべてのノードをターゲット・リストに組み込みます。デフォルトのコンテキストは、**-C** フラグまたは **DSH_CONTEXT** 環境変数を使用して設定できます。
-A	デフォルトのコンテキストで定義されたすべてのデバイスをターゲット・リストに組み込みます。デフォルトのコンテキストは、**-C** フラグまたは **DSH_CONTEXT** 環境変数を使用して設定できます。このフラグは HMC では使用不可になっています。
--all-nodes*context_list*	*context_list* にリストされているコンテキストで定義されたすべてのノードをターゲット・リストに組み込みます。デフォルトのコンテキストは、このリストに暗黙的に組み込まれていません。このフラグは HMC では使用不可になっています。**--all-nodes**。
--all-devices *context_list*	*context_list* にリストされているコンテキストで定義されたすべてのデバイスをターゲット・リストに組み込みます。デフォルトのコンテキストは、このリストに暗黙的に組み込まれていません。このフラグは HMC では使用不可になっています。
-C | **--continue**	対話モードでのみ、ホストに対するリモート・シェル・コマンドの終了値がゼロ以外である場合でも、ノードをターゲット・リスト内で保持します。
-C| **--context***context*	ターゲット名を解決するときに使用するデフォルトのコンテキストです。context 値は、**/opt/ibm/sysmgt/dsm/pm/Context** ディレクトリー内の有効なコンテキスト拡張モジュールに対応している必要があります。例えば、**/opt/ibm/sysmgt/dsm/pm/Context/DSH.pm** ファイルは DSH コンテキスト用のモジュールです。
-d | **--devices** *device_list*	ターゲット・リストに組み込むデバイス・ターゲットのリストを指定します。*device_list* の構文は次のとおりです。 [*context:*][*user_ID@]device_name*[,[*context:*]¥ [*user_ID@*]*device_name*]... このフラグは HMC では使用不可になっています。
--device-rsh*device_remote_shell*	デバイス・ターゲットでのリモート・コマンド実行に使用されるリモート・シェル・コマンドの絶対パスを指定します。特定のコンテキストのリモート・シェルは、パスの前に context: を含めることによって定義できます。*device_remote_shell* の構文は次のとおりです。 [*context:*]*path*[,[*context:*]*path*]... このフラグは HMC では使用不可になっています。 -
-D | **--devicegroups** *devicegroups*	*devicegroups* リストで指定されているデバイス・グループで定義されたすべてのデバイスをターゲット・リストに組み込みます。 *devicegroups* の構文は次のとおりです。 [*context:*] [*user_ID@*]*devicegroup*[,[*context:*]¥ [*user_ID@*]*devicegroup*]... このフラグは HMC では使用不可になっています。
-e | **--execute**	*command_list* がリモート・ターゲットで実行されるローカル・スクリプト・ファイル名および引数を指定することを示します。スクリプト・ファイルは、リモート・ターゲットにコピーされてから、指定された引数を使用してリモート側で実行されます。*DSH_NODE_RCP* および *DSH_DEVICE_RCP* 環境変数は、それぞれ、スクリプト・ファイルをノードおよびデバイスのターゲットにコピーするために使用するリモート・コピー・コマンドを指定します。
-E | **--environment** *environment_file*	*environment_file* に *command_list* の実行前にターゲットにエクスポートする環境変数定義が含まれることを示します。*DSH_NODE_RCP* および *DSH_DEVICE_RCP* 環境変数は、それぞれ、ファイルをノードおよびデバイスのターゲットにエクスポートするために使用するリモート・コピー・コマンドを指定します。
-f | **--fanout** *fanout_value*	並行実行中のリモート・シェル・プロセスの最大数に対するファンアウト値を指定します。ファンアウト値 1 を指定することによって、順次実行を指定できます。**-f** が指定されない場合、デフォルトのファンアウト値の **64** が使用されます。
-F | **--output** *output_path*	標準出力を *output_path*/*target_name.output* にコピーして、標準エラーを *output_path*/*target_name.error* にコピーします。出力は、引き続き、標準出力および標準エラーに送信されます。標準出力および標準エラーを抑止するには、**-Q** フラグを使用します。
-i | **--notify**	ターゲットが応答していないことを示し、ターゲットに対するリモート実行を続行するようにプロンプトを出します。**-v** フラグは、**-i** フラグと同時に指定してください。
-l (小文字の **L)** |**--user** *user_ID*	リモート・コマンド実行に使用するリモート・ユーザー名を指定します。
-h | **--help**	コマンドの使用方法に関する情報を表示します。

項目	説明
-n | **--nodes** *node_list*	ターゲット・リストに組み込むノード・ターゲットのリストを指定します。*node_list* の構文は次のとおりです。 [*context:*] [*user_ID@*]*node_name*[, [*context*:]¥ [*user_ID@*]*node_name*]...
-L | **--no-locale**	ローカル・ホストのロケール定義をリモート・ターゲットにエクスポートしないことを指定します。ローカル・ホストのロケール定義は、デフォルトで各リモート・ターゲットにエクスポートされます。dsh コマンドが実行されるたびに、出力がファイルに付加されます。
--log *log_file*	指定された *log_file* へのロギングを使用可能に設定します。
-m | **--monitor**	各ターゲットでの実行中に状況メッセージを表示することによって、リモート・シェル実行をモニターします。
-N | **--nodegroups** *nodegroups*	*nodegroups* リストで指定されているノード・グループで定義されたすべてのノードをターゲット・リストに組み込みます。 *nodegroups* の構文は次のとおりです。 [*context:*] [*user_ID@*] *nodegroup* [, [*context*:]¥ [*user_ID@*]*nodegroup*]...
-o--node-options *node_options*	ノード・ターゲットに対するリモート・シェル・コマンドに受け渡すオプションを指定します。オプションは、dsh オプションと区別するために二重引用符 ("") で囲んで指定する必要があります。特定のコンテキストのノードに対するオプションは、オプション・リストの前に context: を含めることによって定義できます。*node_options* の構文は次のとおりです。 [*context*:]" *options* "[,[*context*:]"*options*"]...
-O | **--device**-options *device_options*	デバイス・ターゲットに対するリモート・シェル・コマンドに受け渡すオプションを指定します。オプションは、dsh オプションと区別するために二重引用符で囲んで指定する必要があります。特定のコンテキストのデバイスに対するオプションは、オプション・リストの前に context: を含めることによって定義できます。*device_options* の構文は次のとおりです。 [*context*:]"*options*"[,[*context*:]"*options*"]...このフラグは HMC では使用不可になっています。
-Q |**--silent**	サイレント・モードを指定します。ターゲットの出力は、標準出力または標準エラーに書き込まれません。モニター・メッセージは標準出力に書き込まれます。
-q | **--show-config**	すべての DSH ユーティリティー・コマンドに関連する現行の環境設定を表示します。これには、現在インストール済みの有効なすべてのコンテキストに関するすべての環境変数および設定の値が含まれます。設定値のソース・コンテキストを示すために、それぞれの設定値の前に context: が付けられます。
-r | **--node-rsh** *node_remote_copy*	ノード・ターゲットとの間でファイルをコピーするために使用するリモート・シェル・コマンドの絶対パスを指定します。特定のコンテキストのリモート・シェル・コマンドは、パスの前に context: を含めることによって定義できます。*node_remote_copy* の構文は次のとおりです。 [*context:*]*path*[,[*context*:]*path*]...パスに **rsync** が含まれる場合、**rsync** コマンドがリモート・コピーを実行することが想定されます。
--report *report_path*	レポート生成を使用可能に設定して、レポートの保存先となるディレクトリーへのパスを指定します。**--report-name** *report_name* レポートの生成時に使用する名前を指定します。指定しないと、名前はデフォルトの Unspecified になります。このフラグは、**--report** フラグを指定する場合にのみ使用できます。
-s| **--stream**	出力を各ターゲットから使用可能になった時点で返すことを指定します。出力を返す前に *command_list* がターゲットで完了するのを待機しません。
-S | **--syntax csh** | **ksh**	リモート・ターゲットで使用するシェル構文を指定します。指定しない場合、ksh 構文が使用されます。
-t |***--timeout****timeout*	現在実行中のリモート・ターゲットからの出力を待機する時間を秒単位で指定します。指定されたタイムアウトまでにターゲットから出力を得られない場合、dsh コマンドはエラー・メッセージを表示して、応答に失敗したリモート・ターゲットに対する実行を終了します。タイムアウトが指定されない場合、dsh は、無制限に待機して、すべてのリモート・ターゲットからの出力の処理を続行します。 **-i** フラグと同時に指定された場合、ユーザーに対して、出力を待機する追加のタイムアウト間隔を指定するよう求めるプロンプトが出されます。
-T | **--trace**	トレース・モードを使用可能に設定します。dsh コマンドは、各ターゲットに対する実行中に診断メッセージを標準出力に表示します。

項目	説明
-v | **--verify**	リモート・コマンドをターゲットで実行する前に、各ターゲットを検査します。ターゲットが応答しない場合、このターゲットに対するリモート・コマンドの実行は取り消されます。**-i** フラグと同時に指定される場合、ユーザーに対して、検査要求を再試行するよう求めるプロンプトが出されます。
-X *env_list*	dsh 環境変数を無視します。このオプションでは、無視してはならない環境変数名のコンマ区切りリストを引数として指定できます。このオプションに対する引数がない場合、あるいは引数が空ストリングである場合、すべての dsh 環境変数が無視されます。このフラグを最後のフラグとして指定することはできません。
-V | **--version**	dsh コマンドのバージョン情報を表示します。

項目	説明
-z | **--exit-status**	各ターゲットに対して最後にリモート側で実行された非同期コマンドの終了状況を表示します。リモート・ノードで発行されたコマンドがバックグラウンドで実行された場合、終了状況は表示されません。終了状況リモート・シェルの終了値がゼロ以外の場合、各リモート・シェル実行の終了値は、dsh コマンドからのメッセージに表示されます。リモート・シェルからのゼロ以外の戻りコードは、リモート・シェルでエラーが発生したことを示します。この戻りコードは、リモート側で発行されたコマンドの終了コードには関連していません。リモート・シェルでエラーが発生した場合、そのターゲットに対するリモート・コマンドの実行はバイパスされます。dsh コマンドがエラーなしに実行され、すべてのリモート・シェル・コマンドが終了コード 0 で完了した場合、dsh コマンドの終了コードは 0 です。 dsh 内部エラーが発生するか、リモート・シェル・コマンドが正常に完了しない場合、dsh コマンドの終了値は 0 より大きくなります。終了値は、失敗したリモート・コマンド実行の一連のインスタンスごとに 1 ずつ増加します。リモート側で発行されたコマンドがバックグラウンドで実行される場合、リモート側で発行されたコマンドの終了コードは 0 です。環境変数 **DSH_CONTEXT** ターゲットを解決するときに使用するデフォルトのコンテキストを指定します。この変数は、**-C** フラグによって指定変更されます。**DSH_DEVICE_LIST** デバイス・ターゲットのリストを含むファイルを指定します。 この変数は、**-d** フラグによって指定変更されます。この環境変数は HMC では無視されます。**DSH_DEVICE_OPTS** デバイス・ターゲットに対するリモート・シェル・コマンドでのみ使用するオプションを指定します。この変数は、**-O** フラグによって指定変更されます。この環境変数は HMC では無視されます。**DSH_DEVICE_RCP** ローカル・スクリプトおよびローカル環境構成ファイルをデバイス・ターゲットにコピーするために使用されるリモート・コピー・コマンドの絶対パスを指定します。 この環境変数は HMC では無視されます。**DSH_DEVICE_RSH** デバイス・ターゲットでのリモート・コマンド実行に使用するリモート・シェルの絶対パスを指定します。この変数は、**--device-rsh** フラグによって指定変更されます。この環境変数は HMC では無視されます。 **DSH_ENVIRONMENT** リモート・コマンドの実行前にターゲットにエクスポートする環境変数定義が含まれるファイルを指定します。この変数は、**-E** フラグによって指定変更されます。 **DSH_FANOUT** ファンアウト値を指定します。 この変数は、**-f** フラグによって指定変更されます。**DSH_LOG** ロギングに使用するファイルの絶対パスを指定します。この変数は、**--log** フラグによって指定変更されます。**DSH_NODE_LIST** ノード・ターゲットのリストを含むファイルを指定します。WCOLL.Hel DSH_NODE_OPTS は、**DSH_NODE_LIST** 変数によって置き換えられました。ノード・ターゲットだけに、リモート・シェル・コマンドに使用されるオプションが指定されます。この変数は、**-o** フラグによって指定変更されます。**DSH_NODE_RCP** ローカル・スクリプトおよびローカル環境構成ファイルをノード・ターゲットにコピーするために使用するリモート・コピー・コマンドの絶対パスを指定します。**DSH_NODE_RSH** ノード・ターゲットでのリモート・コマンド実行に使用するリモート・シェルの絶対パスを指定します。この変数は、**-r** フラグによって指定変更されます。**DSH_NODEGROUP_PATH** DSH コンテキストのノード・グループ・ファイルを含むディレクトリーのコロンで区切られたリストを指定します。 DSH コンテキストで **-a** フラグが指定されている場合、パスのすべてのノード・グループ・ファイルから固有のノード名のリストが収集されます。**DSH_OUTPUT**。

項目	説明
	標準出力および標準エラーのコピー用の基本ファイル名を指定します。出力は、引き続き、標準出力および標準エラーに送信されます。この変数は、**-F** フラグによって指定変更されます。 **DSH_PATH** ターゲットで使用するコマンド・パスを設定します。 **DSH_PATH** が設定されていない場合、リモート側の *user_ID* のプロファイルで定義されているデフォルト・パスが使用されます。dsh コマンドを HMC に対して実行するために、**DSH_PATH** を使用することはできません。**DSH_REPORT**。 レポートの保存先となるディレクトリーの絶対パスが設定されている場合に、レポートを使用可能に設定します。この変数は、**--report** フラグによって指定変更されます。**DSH_SYNTAX** リモート・ターゲットで使用するシェル構文 (ksh または csh) を指定します。指定しない場合、ksh 構文が使用されます。この変数は、**-S** フラグによって指定変更されます。**DSH_TIMEOUT** 各リモート・ターゲットからの出力を待機する時間を秒単位で指定します。この変数は、**-t** フラグによって指定変更されます。セキュリティー dsh コマンドにセキュリティー構成要件はありません。すべてのリモート・コマンド・セキュリティー要件 (構成、認証、および許可) は、dsh 用に構成された基礎となるリモート・コマンドによって課せられます。 このコマンドでは、ローカル・ホストとリモート・ターゲットの間で認証および許可が構成されていることが想定されます。対話式パスワード・プロンプトはサポートされません。パスワード・プロンプトが出された場合、あるいはリモート・ターゲットに対する許可または認証が失敗した場合、そのリモート・ターゲットに対してエラーが表示され、実行はバイパスされます。リモート環境およびリモート・シェル・コマンドに関連するセキュリティー構成は、ユーザー定義です。リモート・コマンドが **/usr/bin/rsh** として構成されていて、このコマンドが Kerberos バージョン 5 を使用するように構成されている場合、最初に Kerberos **kinit** コマンドを実行してチケット許可チケットを取得し、Kerberos プリンシパルがターゲット上のリモート・ユーザーのホーム・ディレクトリーの **k5login** ファイル内にあることを確認する必要があります。

例

1. ノード・ターゲット **node1** および **node2** で ps コマンドを実行するには、次のように入力します。

   ```
   dsh -n node1,node2 "ps"
   ```

2. **myhosts** ファイルにリストされている各ノード・ターゲットで **ps** コマンドを実行するには、次のように入力します。

   ```
   DSH_NODE_LIST=./myhosts; dsh ps
   ```

3. **NodeGroup1** で定義されているノード・ターゲットで実行するためにコマンドを対話モードで入力するには、次のように入力します。

   ```
   dsh -N NodeGroup1
   ```

4. すべての NIM 管理対象ノードおよび DSH コンテキスト・ノード・グループ **NodeGroup2** のユーザーの数を表示するには、次のように入力します。

   ```
   dsh --all-nodes NIM -N DSH:NodeGroup2 "who | wc -l"
   ```

5. ノード・ターゲットおよびデバイス・ターゲットのリストを対話式に入力してから、日付コマンドを対話モードで実行するには、次のように入力します。

   ```
   dsh -n - -d -
   ```

   さらに次のように入力すると、下記のような出力が表示されます。

   ```
   dsh node> node1
   dsh node> gregb@node2
   dsh node>
   dsh device> CSM:kathyc@device1
   dsh device>
   dsh> date  node1: Wed Apr 13 17:15:59 EDT 2005
   gregb@node2: Wed Apr 13 17:15:59 EDT 2005
   kathyc@device1: Wed Apr 13 17:15:59 EDT 2005
   dsh> exit #
   ```

6. クラスター内のすべてのノードで **ls** コマンドを実行して、すべての dsh 環境変数を無視するには、次のように入力します。

   ```
   dsh -X -a ls
   ```

7. **node1** で **ps** コマンドを実行して、**DSH_NODE_OPTS** を除くすべての dsh 環境変数を無視するには、次のように入力します。

   ```
   dsh -n node1 -X 'DSH_NODE_OPTS' ps
   ```

---

dslpaccept コマンド

目的

ディレクトリー・サービス System V 印刷システムの印刷キュー要求を受け付けます。

構文

dslpaccept *PrintQueueName*

説明

dslpaccept および **dslpreject** コマンドは、待ち状態となっている印刷要求を受け付けるかまたは拒否するように印刷キューを設定するのに使用します。**accept** や **reject** コマンドとは異なり、ディレクトリーを使用可能なコマンドは、ディレクトリー・サービスが使用可能であれば、リモート印刷システムを制御できます。これは、これらのコマンドがディレクトリー・サーバー上の印刷キュー・オブジェクトに直接書き込むためです。

このコマンドのユーザーは、そのユーザーが管理者となるディレクトリー・コンテキスト内で、ディレクトリー・サービスが使用可能であり、また、そのディレクトリー上の書き込み、変更、検索、および読み取りの権限が設定されている必要があります。

パラメーター

項目	説明
PrintQueueName	*PrintQueueName* パラメーターは、印刷キュー・オブジェクトの相対識別名 (RDN) です。リスト内でコンマ (,) で区切ることにより、複数の印刷キュー名を指定することができます。

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 指定された印刷キューは認識できないことを示します。

3 このユーザーは変更の権限を持っていないことを示します。

4 無効な RDN が提供されたことを示します。

5 この値は既に設定されていることを示します。

6 このコマンドはディレクトリー・サービスに連絡できないことを示します。

7 その他のエラーを示します。

例

1. 要求を受け付けるように印刷キュー hpcolor を設定するには、次のように入力します。

```
dslpaccept hpcolor
```

関連資料:

221 ページの『dslpadmin コマンド』
226 ページの『dslpenable コマンド』
230 ページの『dslpsearch コマンド』

関連情報:

lpstat コマンド

---

dslpaccess コマンド

目的

System V 印刷サブシステムの印刷キューに対する、ディレクトリーを使用できないユーザーおよびシステムからのアクセスを許可または拒否します。

構文

dslpaccess -q *QueueName* **-a** *AllowList* | **-d** *DenyList*

説明

dslpaccess コマンドは、ディレクトリーを使用可能な印刷キューに対するユーザーおよびシステムのアクセスを許可または拒否します。これは、**lpadmin** コマンドの **-u** オプションをモデルにしています。

エントリーのコンマ (,) で区切ったリストから構成されるリストを許可および拒否します。これらの各リストは、次のようにして、ログイン ID、またはシステム名とログイン ID を指定することができます。

```
[[LoginID]|[System!LoginID]],[[LoginID]|[System!Login-ID]],...
```

LoginID、 または *System*、またはその両方は、すべての該当するエントリーを許可または拒否するワイルドカード **all** に設定することができます。**all** の使用には注意が必要です。**all** エントリーが 1 つのリストに追加されると、 *LoginID* または *System* の該当する値について、すべての **all** 以外のエントリーが他のリストから削除されます。*System* のデフォルトは、ローカル・ホストです。

このコマンドのユーザーは、そのユーザーが管理者となるディレクトリー・コンテキスト内で、ディレクトリー・サービスが使用可能であり、また、そのディレクトリー上の書き込み、変更、検索、および読み取りの権限が設定されている必要があります。

フラグ

項目	説明
-a *AllowList*	許可リストに追加するユーザーのリストを指定します。これらが拒否リスト上に存在する場合は、拒否リストから削除されます。このオプションは、 **-d** オプションとともに使用することはできません。
-d *DenyList*	拒否リストに追加するユーザーのリストを指定します。これらが許可リスト上に存在する場合は、許可リストから削除されます。このオプションは、 **-a** オプションとともに使用することはできません。
-q *QueueName*	キュー名パラメーターは、印刷キューの相対識別名 (RDN) です。ディレクトリー・コンテキスト内に印刷キュー名が存在しない場合は、コマンドは失敗します。

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 指定された印刷キューは認識できないことを示します。

3 ユーザーが該当するアクセス制御権を持っていないことを示します。

4 無効な RDN が提供されたことを示します。

5 この値は既に設定されていることを示します。

6 その他のエラーを示します。

例

1. 次の例では、ホスト systemX 上の印刷キュー printq1 への fredb アクセスをユーザーに付与します。

   ```
   dslpaccess -q printq1 -a systemX!fredb
   ```

2. 次の例では、すべてのホスト上でのユーザー tomt の印刷キュー printq1 へのアクセスを拒否します。

   ```
   dslpaccess -q printq1 -d all!tomt
   ```

関連資料:

228 ページの『dslpprotocol コマンド』

229 ページの『dslpreject コマンド』

230 ページの『dslpsearch コマンド』

関連情報:

lpstat コマンド

---

dslpadmin コマンド

目的

System V 印刷サブシステムに対してディレクトリーを使用可能な印刷サービスを構成します。

構文

dslpadmin [[**-q** *PrintQueueName* [**-D** *QueueDescription*] [**-n** *LocalQueueName*] [**-o banner** | **nobanner**] [**-A mail** | **none**] [**-F** *FaultRecovery*] [[**-P** *PhysicalPrinterName*] [**-s** *NetworkEntityName*]]] [**-P** *PhysicalPrinterName* [**-T** *PrinterType*] [**-l** *Location*] [**-L** *PDLList*]] [**-q** *PrintQueueName* **-P** *PhysicalPrinterName* [**-I** *ContentType*] [[**-i** *InterfaceScript*] | [**-m** [**Standard** | **PS**]]] [**-o** *PrintOptions*]] [**-q** *PrintQueueName* [**-I** *ContentType*]]] [**-q** *PrintQueueName* **-s** *NetworkEntityName* [**-a** *PrintSystemDNSName* | *PrinterSystemAddress*] [**-t BSD** | **HPNP**]]

dslpadmin [**-q** *PrintQueueName* [**-u** *PhysicalPrinterName*] [**-U** *ObjectRDN*]]

dslpadmin [**-x** *PrintQueueName*] [**-X** *PhysicalPrinterName*] [**-r** *NetworkEntityName*]

dslpadmin [**-h**]

説明

dslpadmin コマンドは、ディレクトリーを使用可能な印刷サービスを構成するために、以下の機能を実行する際に使用されます。

- システムに印刷キューおよび物理プリンターを追加する。
- 印刷キューおよび物理プリンターを変更する。
- システムから印刷キューおよび物理プリンターを除去する。
- ネットワーク・プリンターのためのネットワーク・エンティティー・オブジェクトを追加および削除する。

dslpadmin コマンドは、 **lpadmin** (これはディレクトリー指向ではない) で提供される機能のディレクトリー指向バージョンを提供し、従来型の「フラット・ファイル」構成システムを継続して使用します。両方のシステムが使用されている場合は、プリンター・サブシステムは、ディレクトリー内で最初に見つかった情報を採用することに注意してください。2 つの構成システム間で命名の矛盾が起きないように保証するのは、管理者の責任です。

ディレクトリーを使用可能なコマンドは、 DN (Distinguished Name: 識別名) ではなく RDN (Relative Distinguished Name: 相対識別名) を使用します。例えば、DN が「cn=test,ou=printq,ou=print,cn=aixdata」の場合にディレクトリーを使用可能なキューを作成するには、*PrintQueueName* として RDN「test」のみを使用してください。

管理者がその印刷キューをホスティングするシステム上にいない場合に印刷キューを構成する際は、 **-i** の *InterfaceScript* パラメーターおよび **-T** の *PrinterType* パラメーターはチェックされません。これは、リモート・システムにはチェックのためにアクセスすることができないからです。そのため、指定された *InterfaceScript* および *PrinterType* が確実にリモートのホスティング・システム上に存在するようにするのは、管理者の責任です。

1 つのコマンド・ラインには、**-q**、**-P**、**-s** フラグの任意の組み合わせ、または **-x**、**-X**、**-r** フラグの任意の組み合わせを含めることができますが、いずれか 1 つの組み合わせのみが可能です。複数のディレクトリー・オブジェクトが同時に作成または変更される場合は、 3 つのオブジェクト・タイプ (プリンター、印刷キュー、ネットワーク・エンティティー) 間に適切なリンクがセットアップされます。

フラグ

項目	説明
-a *PrinterSystemDNSName* | *PrinterSystemAddress*	DNS 名またはネットワーク・アドレスをシステムに関連付けます。指定された引数が IPv4 または IPv6 アドレスとして解釈できる場合は、これはアドレスであり、解釈できない場合は DNS 名が想定されます。-a フラグは、 **-s** で指定されたネットワーク・エンティティー・オブジェクトを変更させるか、あるいはまだ存在していない場合は作成させます。管理者は、新しい印刷システム・オブジェクトを追加する代わりに既存の UNIX システム・オブジェクトを変更してしまうことが無いように、ネットワーク・エンティティー・オブジェクトに固有の名前が与えられるようにする必要があります。このフラグは **-s** フラグを必要とします。
-A [mail | none]	印刷要求が失敗した場合に、メールによるメッセージを生成するよう印刷システムに指示します。このメールは、その物理プリンターのオーナー、またはそのプリンターにオーナーがいないか、ユーザーがメール・アドレスを持っていない場合には、その印刷キューをホスティングするシステムの root ユーザーに送付されます。デフォルトは **none** です。このフラグは **-q** フラグを必要とします。
-D *QueueDescription*	**-q** フラグで指定された印刷キュー・オブジェクトについて、記述コメントを定義します。この記述は、ユーザーが **lpstat** コマンドを使用して印刷キューの完全な説明を要求した際に、常に表示されます。ホワイト・スペースを含む文字列は、二重引用符で囲む必要があります。このフラグは **-q** フラグを必要とします。

項目	説明
-F *FaultRecovery*	印刷キューの障害リカバリー・モードを定義します。このフラグは、印刷キュー上のプリンターが印刷要求を印刷している時に障害に遭った場合に使用されるリカバリーを指定します。*FaultRecovery* の値は、次のいずれかを指定できます。 **continue** 印刷が停止したページのトップから、印刷を継続します。これには、自動的に継続する前に障害がクリアされるのを待機するためのフィルターが必要です。 **beginning** 要求の印刷を、始めから再開します。 **wait** *PhysicalPrinterName* での印刷を使用不可にし、管理者またはユーザーが印刷を再び使用可能にするのを待機します。 この待機中に、停止した印刷要求をサブミットした管理者またはユーザーは、印刷を再開する場所を指定する変更要求を発行することができます。印刷が使用可能となる前に変更要求が発行されなければ、フィルターで可能とされる場合は停止したページのトップから印刷が再開され、そうでない場合は要求は始めから印刷されます。 *FaultRecovery* のデフォルト値は、 **beginning** です。このフラグは **-q** フラグを必要とします。
-h	簡潔なヘルプ画面を表示します。
-i *InterfaceScript*	指定された印刷キューを介してアクセスされた際の、プリンターの *InterfaceScript* のパス名。このフラグは、 **-P** フラグが指定されていない場合は無効です。通常、インターフェース・スクリプトはユーザーによって提供されます。このフラグは、**-m** も指定されている場合は使用できません。このフラグは、**-q** フラグと **-P** フラグの両方を必要とします。
-I *ContentType*[, *ContentType*, ...]	印刷キューのコンテンツ・タイプを指定します。印刷キューが、リスト内のコンテンツ・タイプでの印刷要求を処理できるようにします。リスト内に複数の *ContentType* を含める場合は、 *ContentType* パラメーターはコンマ (,) で区切る必要があります。フォーマットの詳細な説明については、 **lpadmin** のマニュアル・ページを参照してください。これは、 **-P** フラグおよび **-q** フラグを必要とします。
-l *Location*	プリンターのロケーションを定義します。これは、プリンターが物理的にどこに位置しているかを示す文字列で、例えば "Building X, Room 6" などとなります。これは、**dslpsearch** コマンドによって検索できます。この値は、1 回設定されると、上書きのみが可能で、削除はできません。このフラグは **-P** フラグを必要とします。
-L *PDL*[, *PDL*, ...]	そのプリンターがサポートする Page Description Language (PDL) のリストを指定します。これは、そのプリンターがサポートする PDL を公示するのに使用され、**dslpsearch** コマンドを使用して検索できます。**AUTOSW**、**PCL**、 **PCLXL**、**POSTSCRIPT**、**TEXT**、**ESCP**、 **PJL**、**SIMPLE**、および **OTHER** PDL がサポートされています。既存の物理プリンター・オブジェクトの変更に **-L** フラグが使用される場合は、このリストが既存のリストを置き換えます。このフラグは **-P** フラグを必要とします。
-m [**standard** | **PS**]	指定された印刷キューを介してアクセスされた際の、プリンターのモデル・インターフェース・プログラム。これは、その印刷キューに使用されるモデル・インターフェース・スクリプトを選択します。物理プリンター・オブジェクトが作成され、 **-m** フラグおよび **-i** フラグのどちらも指定されていない場合は、デフォルトは **standard** です。このフラグは、 **-i** も指定されている場合は使用できません。このフラグは、**-q** フラグと **-P** フラグの両方を必要とします。
-n *LocalQueueName*	印刷キューのローカル名を定義します。ディレクトリーを使用できないホスト上にそのキューがある場合は、通常、この名前がキューの RDN と異なっているだけです。これは、受信側システム上の印刷キューを識別するために、着信リモート・ネットワーク接続によって使用されます。デフォルト値は、印刷キューの RDN です。このフラグは **-q** フラグを必要とします。
-o [**banner** | **nobanner**]	この印刷キューによってバナー・ページが常に生成されるかどうかを定義します。デフォルト値である **banner** はすべての印刷要求に対してバナー・ページが印刷されるよう強制し、 **nobanner** は、ユーザーにより、バナー・ページが印刷されないように指定して印刷ジョブをサブミットすることを可能にします。このフラグは **-q** フラグを必要とします。

項目	説明
-o *PrintOption=Value*[, ...]	印刷オプションの値を指定します。**-o** フラグで使用可能な印刷オプションの詳しい説明については、 **lpadmin** の文書を参照してください。このフラグは、 **-q** フラグと **-P** フラグの両方を必要とします。
-P *PhysicalPrinterName*	物理プリンター・オブジェクトを作成または変更します。*PhysicalPrinterName* 引数は、プリンター・オブジェクトの RDN を指定します。そのオブジェクトがまだ存在していない場合は、 **dslpadmin** はそれを作成します。
-q *PrintQueueName*	**dlspadmin** が印刷キュー・オブジェクトを作成または変更します。*PrintQueueName* 引数は、印刷キュー・オブジェクトの RDN を指定します。新しい印刷キューを追加する時は、追加される印刷キューの *NetworkEntityName* および *PhysicalPrinterName* をコマンドに知らせるため、 **-s** フラグおよび **-P** フラグを指定する必要があります。その印刷キュー・オブジェクトが存在しない場合は、 **dslpadmin** はそれを作成します。 1 つのコマンド・ラインには、**-q**、**-P**、**-s** フラグの任意の組み合わせ、または **-x**、**-X**、**-r** フラグの任意の組み合わせを含めることができますが、いずれか 1 つの組み合わせのみが可能です。複数のディレクトリー・オブジェクトが同時に作成または変更される場合は、 3 つのオブジェクト・タイプ (プリンター、印刷キュー、ネットワーク・エンティティー) 間に適切なリンクがセットアップされます。
-r *NetworkEntityName*	ネットワーク・エンティティー・システム・オブジェクトを削除します。非プリンター・システム・オブジェクトを削除しないように、注意が必要です。確実に正しいオブジェクトが削除されるようにするのは、管理者の責任です。
-s *NetworkEntityName*	その印刷キューにホスティングするネットワーク・エンティティー・システム・オブジェクトを指定します。**-a** も指定された場合は、そのオブジェクトは作成または変更されます。*NetworkEntityName* 引数は、現行ディレクトリー・コンテキスト内のオブジェクトの RDN を指定します。ネットワーク・エンティティー・オブジェクトは、リモート・クライアントが印刷キューにアクセスするのに使用する必要があるネットワーク・アドレスを定義します。
-t [BSD | HPNP]	この「ネットワーク・プリンター」の印刷キューに使用される印刷プロトコルを定義します。再試行およびタイムアウトの値は、それらのネットワーク・プリンター用のデフォルト値に設定されます。これらの値を変更するには、 **dslpprotocol** コマンドを使用する必要があります。このフラグは、 BSD または HPNP プロトコルをサポートするネットワーク・プリンターに対してのみ使用する必要があることに注意してください。このフラグは **-q** フラグを必要とします。
-T *PrinterType*[, *PrinterType*, ...]	プリンター・タイプのリスト。これは、例えば "hplaserjet" などのように、そのプリンターを 1 つ以上のプリンター・タイプのものであると識別します。詳細については、 **lpadmin** のマニュアル・ページを参照してください。このフラグは **-P** フラグを必要とします。
-u *PhysicalPrinterName*	指定された物理プリンターを、そのオブジェクトは削除せずに、印刷キュー (**-q** フラグで指定される) からリンク解除します。このフラグは **-q** フラグを必要とします。
-U *ObjectRDN*	物理プリンターまたは印刷キュー・オブジェクト (*ObjectRDN* で指定される) のいずれかを、そのオブジェクトは削除せずに、印刷キュー (**-q** フラグで指定される) からリンク解除します。このフラグは **-q** フラグを必要とします。
-x *PrintQueueName*	印刷キュー・オブジェクトを削除します。
-X *PhysicalPrinterName*	物理プリンター・オブジェクトを削除します。

終了状況

0 成功を示します。

255 (または **-1)**
構成におけるエラーを示します。エラーまたは障害について説明するエラー・メッセージが表示されます。

例

次の各例は、ディレクトリーを使用可能な UNIX システムにユーザーがログオンした際の dslpadmin コマンドの使用法について示しています。

1. 次の例では、印刷キューの RDN が「denlj5n」、物理プリンターの RDN が「denplj5n」で、BSD リモート印刷プロトコルを使用する HP LaserJet ネットワーク・プリンターを追加します。これは、"HP JetDirect (PostScript)" の記述、プリンター・タイプ "PS-b"、およびモデル・インターフェース・スクリプト "PS" を印刷キューに指定しています。このプリンターは、ネットワーク・アドレス "p_hplj.ibm.com" を持ちます。

   ```
   dslpadmin -q denlj5n -P denplj5n -T PS-b -D "HP JetDirect (Postscript)" ¥
   -I PS -m PS -A mail -o nobanner  -s denslj5n -a p_hplj.ibm.com -t BSD
   ```

   この印刷システムは、この印刷キューに対するコンテンツ・タイプ PS の印刷要求を許可し、バナー・ページを使用不可にすることを許可します。

2. 次の例では、印刷キューの RDN が「dehpnp」、物理プリンターの RDN が「dephpnp」で、HPNP リモート印刷プロトコルを使用する、HP LaserJet PostScript ネットワーク・プリンターを追加します。これは、"HPNP (PCL)" の記述、プリンター・タイプ "hplaserjet"、およびモデル・インターフェース・スクリプト "standard" を印刷キューに指定しています。このプリンターは、ネットワーク・アドレス "p_hplj.ibm.com" を持ちます。

   ```
   dslpadmin -q dehpnp -P dephpnp -T hplaserjet -D "HPNP (PCL)" -I pcl ¥
   -m standard -A mail -s deshpnp -a p_hplj.ibm.com -t HPNP
   ```

   この印刷システムは、この印刷キューに対するコンテンツ・タイプ PCL の印刷要求を許可し、バナー・ページ無しが要求された場合は要求を拒否します。プリンターの障害が発生した場合は、この印刷システムはプリンターのオーナーにメールを送付します。

3. 次の例では、HP LaserJet PostScript プリンターを削除します。

   ```
   dslpadmin -x delj5n  -X deplj5n
   ```

4. 次の例では、HPNP プリンターを削除します。

   ```
   dslpadmin -x dehpnp  -X dephpnp -r deshpnp
   ```

関連資料:

219 ページの『dslpaccept コマンド』

220 ページの『dslpaccess コマンド』

関連情報:

lpstat コマンド

---

dslpdisable コマンド

目的

System V 印刷サブシステムの印刷キュー要求を使用不可にします。

構文

dslpdisable [**-r** *Reason*] *PrintQueueName*

説明

dslpenable および **dslpdisable** コマンドは、待ち状態にされている印刷処理要求からの印刷キューを使用可能または使用不可にするのに使用します。**enable** や **disable** コマンドとは異なり、ディレクトリーを使用可能なコマンドは、ディレクトリー・サービスが使用可能であれば、リモート印刷システムを制御できます。これは、これらのコマンドがディレクトリー・サーバー上の印刷キュー・オブジェクトに直接書き込むためです。

フラグ

項目	説明
-r *Reason*	この印刷キューを使用不可にする理由を割り当てます。ホワイト・スペースを含む文字列は、二重引用符で囲む必要があります。 *Reason* は、lpstat コマンドによって表示される文字列です。何も指定されていない場合、デフォルトの理由は設定されていません。

パラメーター

項目	説明
PrintQueueName	*PrintQueueName* パラメーターは、印刷キューの RDN です。これには、印刷キューのリストを指定できます。ディレクトリー・コンテキスト内に印刷キュー名が存在しない場合は、コマンドは失敗します。

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 指定された印刷キューは認識できないことを示します。

3 このユーザーは変更の権限を持っていないことを示します。

4 無効な RDN が提供されたことを示します。

5 この値は既に設定されていることを示します。

6 このコマンドはディレクトリー・サービスに連絡できないことを示します。

7 その他のエラーを示します。

例

理由 "routine maintenance" を指定して、印刷キュー printer1 を使用不可にするには、次のように入力します。

```
dslpdisable -r "routine maintenance" printer1
```

関連資料:

228 ページの『dslpprotocol コマンド』

229 ページの『dslpreject コマンド』

230 ページの『dslpsearch コマンド』

関連情報:

lpstat コマンド

---

dslpenable コマンド

目的

System V 印刷サブシステムの印刷キュー要求を使用可能にします。

構文

dslpenable *PrintQueueName*

説明

dslpenable および **dslpdisable** コマンドは、待ち状態にされている印刷処理要求からの印刷キューを使用可能または使用不可にするのに使用します。**enable** や **disable** コマンドとは異なり、ディレクトリーを使用可能なコマンドは、ディレクトリー・サービスが使用可能であれば、リモート印刷システムを制御できます。これは、これらのコマンドがディレクトリー・サーバー上の印刷キュー・オブジェクトに直接書き込むためです。

パラメーター

項目	説明
PrintQueueName	*PrintQueueName* パラメーターは、印刷キューの RDN です。これには、印刷キューのリストを指定できます。ディレクトリー・コンテキスト内に印刷キュー名が存在しない場合は、コマンドは失敗します。

サブコマンド

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 指定された印刷キューは認識できないことを示します。

3 このユーザーは変更の権限を持っていないことを示します。

4 無効な RDN が提供されたことを示します。

5 この値は既に設定されていることを示します。

6 このコマンドはディレクトリー・サービスに連絡できないことを示します。

7 その他のエラーを示します。

例

1. 印刷キュー hpcolor を使用可能にするには、次のように入力します。

```
dslpenable hpcolor
```

関連資料:

221 ページの『dslpadmin コマンド』

228 ページの『dslpprotocol コマンド』

229 ページの『dslpreject コマンド』

関連情報:

lpstat コマンド

dslpprotocol コマンド

目的

System V 印刷サブシステムのための印刷キューのリモート印刷プロトコルを構成します。

構文

dslpprotocol -t *RemoteProtocol* **[-T** *TimeOut* **] [-R** *Retry* **] [-r]** *PrintQueueName*

dslpprotocol -l [-S] *PrintQueueName*

説明

dslpprotocol コマンドは、リモート印刷クライアントが印刷要求を印刷キューへ送る際に使用できる「リモート印刷プロトコル」を構成するのに使用します。

ディレクトリーを使用可能な場合にリモート印刷キューへ印刷するには、クライアントはまずそれが使用できるリモート印刷プロトコルを取得する必要があります。これは、そのディレクトリー内の印刷キュー・オブジェクトから取得されます。これは、 BSD または HPNP のいずれか 1 つ、あるいはその両方が可能です。1 つの印刷キューに対して複数のプロトコルが構成された場合、UNIX 印刷システムは読み込む最初の値を使用するので、通常 1 つのキューには 1 つのプロトコルが構成されることになります。

PrintQueueName パラメーターは、印刷キューの相対識別名 (RDN) です。*PrintQueueName* に割り当てられた値が存在しない場合は、コマンドは失敗します。

このコマンドのユーザーは、そのユーザーが管理者となるディレクトリー・コンテキスト内で、ディレクトリー・サービスが使用可能であり、また、そのディレクトリー上の書き込み、変更、検索、および読み取りの権限が設定されている必要があります。

フラグ

項目	説明
-l	その印刷キューに関連したリモート印刷プロトコル・パラメーターの説明を印刷します。
-t *RemoteProtocol*	印刷要求をこの印刷キューへ送る際に使用できるリモート印刷プロトコルを指定します。サポートされるプロトコル・タイプの値は、 **bsd** および **hpnp** です。デフォルトの値は **bsd** です。
-T *TimeOut*	指定されたプロトコルについてのネットワーク接続のタイムアウト値 (つまり、ネットワーク接続が、切断される前に、アイドル状態で活動状態を保っていなければならない時間) を設定します。タイムアウトを使用不可にするために、値 n を指定することもできます。値 0 は、接続がアイドルになるとすぐに、接続をドロップさせます。デフォルト値は 10 分であり、実質的な上限はありません。**-T** オプションの完全な定義については、 **lpsystem** のマニュアル・ページを参照してください。
-r	このオプションは、印刷キュー・オブジェクトから指定されたプロトコルを除去するのに使用します。このオプションは、 **-t** オプションも指定されることを必要とします。
-R *Retry*	指定されたプロトコルについてのネットワーク接続の再試行時間 (つまり、失敗後に、ネットワーク接続の再確立を試行するまで待機する時間 (単位: 分)) を設定します。デフォルト値は 2 分です。値 0 は、即時に接続の再試行を行います。この値は、 **-T** オプションを使用して指定されたタイムアウト値より短い必要があることに注意してください。すべての作業が使用不可である場合にドロップされた接続が再試行されるのを防ぐために、値 n を指定することもできます。この値には、実質的な上限はありません。「ネットワーク・プリンター」については、再試行の時間は 0 に設定する必要があります。**-R** オプションの完全な定義については、 **lpsystem** のマニュアル・ページを参照してください。
-S	印刷キューのプロトコルのセットアップを簡易フォーマットで表示するために、 **-l** オプションとともに使用されます。

パラメーター

項目	説明
PrintQueueName	*PrintQueueName* パラメーターは、印刷キューの相対識別名 (RDN) です。*PrintQueueName* に割り当てられた値が存在しない場合は、コマンドは失敗します。

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 指定された印刷キューは認識できないことを示します。

3 このユーザーは変更の権限を持っていないことを示します。

4 無効な RDN が提供されたことを示します。

5 この値は既に設定されていることを示します。

6 その他のエラーを示します。

例

1. BSD リモート印刷プロトコルを許可するように印刷キュー printq1 を設定するには、次のように入力します。

   ```
   dslpprotocol -t BSD printq1
   ```

2. BSD プロトコルを印刷キュー hpcolor から除去するには、次のように入力します。

   ```
   dslpprotocol -r -t BSD hpcolor
   ```

関連資料:

219 ページの『dslpaccept コマンド』
220 ページの『dslpaccess コマンド』
221 ページの『dslpadmin コマンド』
226 ページの『dslpenable コマンド』

---

dslpreject コマンド

目的

ディレクトリー・サービス System V 印刷システムに対する印刷キュー要求を拒否します。

構文

dslpreject [**-r** *Reason*] *PrintQueueName*

説明

dslpaccept および **dslpreject** コマンドは、待ち状態となっている印刷要求を受け付けるかまたは拒否するように印刷キューを設定するのに使用します。**accept** や **reject** コマンドとは異なり、ディレクトリーを使用可能なコマンドは、ディレクトリー・サービスが使用可能であれば、リモート印刷システムを制御できます。これは、これらのコマンドがディレクトリー・サーバー上の印刷キュー・オブジェクトに直接書き込むためです。既に待ち状態になっている印刷要求は、 **dslpreject** コマンドの影響を受けません。

このコマンドのユーザーは、そのユーザーが管理者となるディレクトリー・コンテキスト内で、ディレクトリー・サービスが使用可能であり、また、そのディレクトリー上の書き込み、変更、検索、および読み取りの権限が設定されている必要があります。

フラグ

項目	説明
-r *Reason*	拒否する理由を割り当てます。ホワイト・スペースを含む文字列は、二重引用符で囲む必要があります。*Reason* は、 **lpstat** コマンドによって表示される文字列です。何も指定されていない場合、デフォルトの理由は設定されていません。

パラメーター

項目	説明
PrintQueueName	*PrintQueueName* パラメーターは、印刷キュー・オブジェクトの RDN です。リスト内でコンマ (,) で区切ることにより、複数の印刷キュー名を指定することができます。

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 指定された印刷キューは認識できないことを示します。

3 このユーザーは変更の権限を持っていないことを示します。

4 無効な RDN が提供されたことを示します。

5 この値は既に設定されていることを示します。

6 このコマンドはディレクトリー・サービスに連絡できないことを示します。

7 その他のエラーを示します。

例

1. 印刷キューが要求を拒否するよう設定し、トナーが無い (no toner) という理由を指定するには、次のように入力します。

```
dslpreject -r "no toner" printer1
```

関連資料:

225 ページの『dslpdisable コマンド』

228 ページの『dslpprotocol コマンド』

『dslpsearch コマンド』

関連情報:

lpstat コマンド

---

dslpsearch コマンド

目的

System V 印刷サブシステム上の印刷システム・オブジェクトについて、ディレクトリーを検索します。

構文

dslpsearch [**-q** [**-p**]] | [**-P**] [**-o** *SearchOptions*]

説明

dslpsearch コマンドにより、ユーザーおよび管理者は印刷システム・オブジェクトについてディレクトリーの検索が可能になります。例えば、ユーザーはカラーの PostScript ファイルを印刷できるプリンターを検索することができます。このコマンドの主な使用は、検索文字列に一致する印刷キューの検索になります。

dslpsearch コマンドは、検索文字列に一致するオブジェクトの DN (Distinguished Name: 識別名) を返します。ただし、他のディレクトリーを使用可能なコマンドを使用するためには、RDN (Relative Distinguished Name: 相対識別名) が必要です。例えば、DN `"cn=testqueue,ou=printq,ou=print,cn=aixdata"` が **dslpsearch** コマンドによって返されると、印刷キューの参照には RDN `" testqueue"` のみが使用されます。

フラグ

項目	説明
-q	検索オプションに一致する印刷キューを検索します。検索は物理プリンター・オブジェクトについて行われますが、表示されるのはこれらのプリンターにサービスを行う印刷キューです。これはデフォルトの検索タイプです。**-q** オプションは、 **-P** とともに指定することはできません。
-p	このオプションは、 **-q** オプションとともに使用され、その印刷キューにサービスを行っている物理プリンターのリストも表示させます。
-P	検索文字列に一致する物理プリンターを検索します。**-P** オプションは、 **-q** とともに指定することはできません。
-o *SearchOptions*	複数の検索オプションを、コンマで区切った 1 つのリストで構成することができます。各オプションは、次のものから構成できます。 • 1 つ以上の、次に示す Page Description Languages (PDL): **AUTOSW**、 **PCL**、**PCLXL**、**POSTSCRIPT**、 **TEXT**、**ESCP**、**PJL**、**SIMPLE**、**OTHER** • 次のプリンター機能のいずれか: **COLOR**、**DUPLEX**、 **TRAYS**、**FINISH** • `location=xxxxxxxx` または `location='aaaa bbbbb'` で指定される、 1 つ以上の物理プリンターのロケーション • location= で定義される文字列値は、文字列の両端にワイルドカードを配置して検索されます。そのため、`location=Room1` は、例えば `"Building X, Room1, Bay6"` などのように、ロケーションに `"Room1"` を持つすべてのプリンターを検出します。文字列値には、例えば `location="Building X*Bay6"` などのように、内部にワイルドカード (*) を組み込むこともできます。複数のロケーション値は、検索で OR (論理和演算) されます。 • 次に、検索文字列を含む有効なコマンド・ラインを示します。 `dslpsearch -q -o PCL,ESCP,location=room2,COLOR` `dslpsearch -q -p -o "PS, location='Building 1, Room1', DUPLEX"`

終了状況

0 成功を示します。

1 無効なオプションを示します。

2 ディレクトリー・ツリー上での検索が失敗したことを示します。

3 無効なディレクトリー・コンテキストを示します。

4 このコマンドはディレクトリー・サービスに連絡できないことを示します。

例

1. 次のコマンド・ラインは、検索オプションに一致する印刷キューを検索します。
   ```
   dslpsearch -q -o search-options
   ```
2. 次の例は、検索オプションに一致する物理プリンターを検索します。
   ```
   dslpsearch -P -o search-options
   ```

関連資料:

219 ページの『dslpaccept コマンド』
220 ページの『dslpaccess コマンド』
221 ページの『dslpadmin コマンド』

関連情報:

lpstat コマンド

---

dspcat コマンド

目的

メッセージ・カタログの全部または一部を表示します。

構文

カタログ内のメッセージを表示する

dspcat *CatalogName* [*SetNumber* [*MessageNumber*]]

gencat コマンド用に出力をフォーマットする

dspcat -g *CatalogName* [*SetNumber*]

説明

dspcat コマンドは、特定のメッセージ、セット内のすべてのメッセージ、またはカタログ内のすべてのメッセージを表示します。 **dspcat** コマンドは、メッセージを標準出力に出力します。

注**:** C または POSIX のロケール環境で **LC__FASTMSG** 属性が `False` に設定されている場合、**dspcat** コマンドは **NLSPATH** 環境変数の下でカタログ・ファイルを検索します。

LC__FASTMSG 属性は、C および POSIX のロケールに対してデフォルト・メッセージが使用されること、および **LC__FASTMSG** 属性が `True` に設定されていると **NLSPATH** 環境変数が無視されることを指定します。

`/etc/environment` パスでは、**LC__FASTMSG** 属性のデフォルト値は `True` です。

CatalogName パラメーターはメッセージ・カタログを指定します。 *SetNumber* パラメーターは、*CatalogName* パラメーターで指定したカタログ内のセットを指定します。 *MessageNumber* パラメーターは、 *SetNumber* パラメーターで指定したセット内の特定のメッセージを指定します。3 つのパラメーターをすべて指定すると、 **dspcat** コマンドは特定のメッセージを表示します。 *MessageNumber* パラメーター

を指定しないと、 **dspcat** コマンドはセット内のすべてのメッセージを表示します。存在しない *SetNumber* または *MessageNumber* パラメーター値を指定すると、 **dspcat** コマンドはエラー・メッセージを表示し、0 以外の戻り値を戻します。*CatalogName* パラメーターだけを指定すると、 **dspcat** コマンドはカタログ内のすべてのメッセージを表示します。*MessageNumber* パラメーターを設定する場合は、必ず *SetNumber* パラメーターを指定する必要があります。

CatalogName パラメーターの値に / (スラッシュ) を使わなければ、**dspcat** コマンドは、**NLSPATH** 環境変数と **LC_MESSAGES** カテゴリーを使って、指定されたメッセージ・カタログを検索します。

フラグ

項目	説明
-g	**gencat** コマンドへの入力として使われる出力をフォーマットします。 **-g** フラグを使用すると、*MessageNumber* パラメーターは無効になります。

例

`test.cat` ファイルのセット番号 `1` のメッセージ番号 `2` を表示するには、次のように入力します。

```
dspcat test.cat 1 2
```

ファイル

項目	説明
/usr/bin/dspcat	**dspcat** コマンドが入っています。

関連資料:

675 ページの『gencat コマンド』

関連情報:

runcat コマンド

catopen コマンド

メッセージ機能

dspmsg コマンド

目的

選択されたメッセージをメッセージ・カタログから表示します。

構文

dspmsg [**-s** *SetNumber*] *CatalogName MessageNumber* [**'***DefaultMessage***'** [*Arguments*]]

説明

dspmsg コマンドは、**gencat** コマンドで作成されたメッセージ・カタログから特定のメッセージのテキストを表示します。または、メッセージが取り出せない場合は、コマンドにパラメーターとして与えられたデフォルト・メッセージを表示します。 **dspmsg** コマンドは、メッセージを標準出力に出力します。このコマンドは、**echo** コマンドの代わりにシェル・スクリプトで使用するためのものです。

注**: dspmsg** コマンドは、**LC_FASTMSG** が C または POSIX ロケール環境で偽に設定されている場合は、**NLSPATH** の下でカタログ・ファイルを検索します。

LC__FASTMSG は、デフォルト・メッセージが C および POSIX ロケールに対して使用されること、および **LC__FASTMSG** が真に設定されている場合は、**NLSPATH** が無視されることを指定します。

LC__FASTMSG のデフォルト値は、**/etc/environment** において真です。

NLSPATH 環境変数および **LC_MESSAGES** カテゴリーは、*CatalogName* パラメーターの値に / (スラッシュ) が使われていない場合に、指定されたメッセージ・カタログを見つける目的で使用されます。*CatalogName* パラメーターで指定されたカタログが見つからない場合、または *MessageNumber* パラメーター (およびオプションの *SetNumber* 値) で指定されたメッセージが見つからない場合には、指定された *DefaultMessage* 値が表示されます。 *DefaultMessage* 値が指定されていなければ、システムが生成したエラー・メッセージが表示されます。

dspmsg コマンドでは、文字列引数に **%s**、%n$s、**%ld**、または %n$ld **printf** サブルーチン変換指定が含まれる場合は、10 個までの文字列引数をメッセージに置き換えることができます。変換指定用の引数で存在しないものがあると、**dspmsg** エラー・メッセージが出力されます。標準の **printf** サブルーチン制御文字エスケープ (例えば、**¥n**) が認識されます。

カタログ内では、**printf** サブルーチン・フォーマット文字列を使用するようにしてください。このフォーマットでは、メッセージ内のフォーマット文字列が、デフォルト・メッセージとは異なる順序になっていても、引数が正しく挿入されます。メッセージの挿入に **%n$s** 表記法を使う場合は、デフォルト・メッセージを単一引用符で囲まなければなりません。

フラグ

項目	説明
-s *SetNumber*	オプションのセット番号を指定します。*SetNumber* 変数のデフォルト値は 1 です。

例

`test.cat` のセット番号 1、メッセージ番号 2 を表示するには、次のように入力します。

```
dspmsg  -s  1  test.cat  2  'message  %s  not  found'  2
```

メッセージが見つからないと、「`message 2 not found`」が表示されます。

ファイル

項目	説明
/usr/bin/dspmsg	**dspmsg** コマンドが入っています。

関連資料:

232 ページの『dspcat コマンド』

関連情報:

mkcatdefs コマンド

catclose コマンド

メッセージ機能

---

dtaction コマンド

目的

指定された引数のある CDE アクションを起動します。

構文

dtaction [**-contextDir** *context_dir*]¥

[**-execHost** *host_name*] [**-termOpts** *terminal_arguments*]

[**-user** *user_name*] *action_name*

[*action_arg*] ...

説明

dtaction コマンドを使用すると、CDE 開発環境に接続されていないアプリケーションまたはシェル・スクリプトがアクション要求を起動できるようになります。

action_name と呼ばれるアクションが、コマンド・ラインで提供される *action_arg* と一緒に起動されます。

単一の *action_name* が必要です。ユーザーは任意の数の *action_args* を提供できます。

action_name および *action_args* の解釈は、アクション・データベース内のアクションの定義によって決まります。

アクションは、システム・アクション・データベース・ファイルの 1 つまたはユーザーの私用アクション・データベース・ファイルの 1 つに定義できます。

action_args は、ファイルの絶対パス名または相対パス名です。 **dtaction** コマンドは、指定されたアクションにファイルのこのリストを渡します。

以下の条件が検出されると、エラー・ダイアログが通知されます。

- デスクトップ環境が初期化できない
- ユーザーまたはパスワードが無効
- 要求されたユーザーに ID を変更できない
- アクション名が指定されていない

フラグ

項目	説明
contextDir *context_dir*	コマンド・アクションの現在の作業ディレクトリーが *action_name* の定義内で定義されていない場合は、デフォルトのディレクトリー・コンテキストを指定します。
execHost *host_name*	コマンド・アクションの代替実行ホスト *host_name* を指定します。アクションがコマンド・アクションでない場合は、**dtaction** コマンドはこのオプションを無視します。アクションの EXEC_HOST 仕様で指定されたホストではなく、*host_name* に対してアクションが試行されます。指定されたアクションがいずれかの適格ホストで起動できない場合は、エラーが通知されます。
termOpts *terminal_arguments*	タイプ NO_STDIO 以外のコマンド・アクション用に提供される端末エミュレーターを対象とする引数を指定します。 *terminal_arguments* 文字列の中にホワイト・スペース文字がある場合、その文字列を引用符で囲んで、シェルから保護する必要があります。これらの引数は変更されずに端末エミュレーターに渡されるので、ユーザーは妥当な文字列を使用する必要があります。特に、*terminal_arguments* では、コマンドを端末エミュレーター・ウィンドウで実行することを指定する引数 (つまり、**-e** フラグを指定した **dtterm1** を使用するもの) は使用できません。

項目	説明
user *user_name*	ユーザー名を指定します。現在 **dtaction** がその指定されたユーザーとして実行されていない場合は、指定されたユーザーのパスワードまたは root ユーザーのパスワードを収集するためにプロンプト・ダイアログが表示されます。有効なパスワードが入力された後、**dtaction** コマンドは、要求されたユーザーとして実行されるように変更されてから、要求されたアクションを開始します。

パラメーター

項目	説明
action_name	起動するアクションの名前を指定します。
action_arg	ファイルの絶対ファイル名または相対ファイル名を指定します。

環境変数

項目	説明
DTDATABASESEARCHPATH	アクション・サービスにアクション・データベースの場所を知らせる、コンマで区切られたディレクトリーのリスト (オプションとして、host: プレフィックス)。

終了状況

次の終了値が戻されます。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

dtaction コマンドは、PAM によって使用可能にされるサービス名 **dtaction** のアプリケーションです。

user *user_name* オプションで指定されたユーザー名がログイン・ユーザー名と異なる場合、

dtaction コマンドは指定されたアクションを起動する前にユーザー認証を行います。このコマンドは PAM 認証と従来型の認証のいずれにも対応しています。

システム全体の認証に PAM を使用するには、root ユーザー・アクセス権を確立し、**/etc/security/login.cfg** ファイルの **usw** スタンザの *auth_type* 属性の値を PAM_AUTH に変更します。

PAM が使用可能であるときに使用される認証メカニズムは、**/etc/pam.conf** 内のログイン・サービスの構成によって異なります。

dtaction コマンドは、**auth** モジュール・タイプに関する **/etc/pam.conf** エントリーを必要とします。

dtaction サービス用の **/etc/pam.conf** では、次の構成が推奨されます。

```
dtaction        auth                required        /usr/lib/security/pam_aix
```

例

1. アクションを起動するには、次のように入力します。

   ```
   dtaction Xterm
   ```

これにより、X Windows 端末エミュレーター (Xterm) が起動します。

2. リモート・ホストでアクションを起動するには、次のように入力します。

```
dtaction -execHost hostname Xterm
```

これで、指定されたリモート・ホスト上で Xterm が実行されます。

3. 別のユーザーとしてアクションを起動するには、次のように入力します。

```
dtaction -user username Xterm
```

これで、指定されたユーザーとして Xterm が実行されます。

位置

/usr/dt/bin/dtaction

標準エラー

dtaction コマンドは、診断エラー・メッセージを標準エラーに書き込みます。標準エラーは **$HOME/.dt/errorlog** にリダイレクトされます。

ファイル

項目	説明
/etc/pam.conf	PAM 認証メカニズムを決定します。
/etc/security/login.cfg	システム全体の PAM 認証を決定します。

関連資料:

239 ページの『dtlogin コマンド』

265 ページの『dtsession コマンド』

---

dtappintegrate コマンド

目的

共通デスクトップ環境アプリケーションの統合ツールです。

構文

dtappintegrate -s *ApplicationRoot* [**-t** *TargetPath*] [**-l** *Language*] [**-u**]

説明

dtappintegrate コマンドは、アプリケーションの CDE 構成ファイルをアプリケーション固有の位置からシステムの位置へリンクさせて、影響を受ける言語用のシステムのブラウザー・ヘルプ・ボリュームを更新します。 **dtappintegrate** コマンドは、アプリケーションのインストール・プロセス中に使用されます。アプリケーションのインストール・スクリプトでは、最後に **dtappintegrate** コマンドが起動されます。

CDE ポリシーに従ったアプリケーション・ルート (**$APP_ROOT** を指す) の下には、次の 4 つの主要サブディレクトリーがあります。

項目	説明
$APP_ROOT/dt/appconfig/types/*Language*	ファイル・タイプ、フロント・パネル、アクション・ファイル用。
$APP_ROOT/dt/appconfig/appmanager/*Language*	アプリケーション・グループ・ファイル用。
$APP_ROOT/dt/appconfig/icons/*Language*	CDE マネージャーが使用するアイコン用。
$APP_ROOT/dt/appconfig/help/*Language*	アプリケーションのヘルプ用。例えば、デフォルト言語のアプリケーション **SpreadSheet** は、そのデスクトップ・アイコンを、**/opt/SpreadSheet/dt/appconfig/icons/C/*.bm** と **/opt/SpreadSheet/dt/appconfig/icons/C/*.pm** の下にロードします。ここで、**/opt/SpreadSheet** は **$APP_ROOT** の値です。

注**: $APP_ROOT** は、このマニュアルの構文の事例で使うルート・ディレクトリーですが、実行時環境では使用されません。上記のすべての CDE 構成ファイルとサブディレクトリーは、共通の最上位ディレクトリーの下に置かれ、必ずデフォルト言語サブディレクトリー **C** を含みます。

最も単純な場合、コマンドの入力は、**/opt/thisapp** などアプリケーション・ルートです。このコマンドの出力は、アプリケーション・サーバー上の対応するサブディレクトリーとファイルです。これらのサブディレクトリーとファイルには、上記のアプリケーションの CDE 構成ファイルへの相対シンボリック・リンクが入っており、次のシステム位置の下に置かれます。

項目	説明
/etc/dt/appconfig	最上位レベルのアプリケーション構成サブディレクトリーで、次のサブディレクトリーから構成されます。 **/etc/dt/appconfig/types/***Language***/** ***.dt** リンクおよび ***.fp** リンクが入っています。 **/etc/dt/appconfig/appmanager/***Language***/** アプリケーション・グループ・サブディレクトリーへのリンクと、アプリケーション・マネージャーの下で実行されるアクション・スクリプト・ファイルが入っています。 **/etc/dt/appconfig/help/***Language***/** アプリケーションのルートの下にインストールされたヘルプ・ファイルへのシンボリック・リンクが入っています。 **/etc/dt/appconfig/icons/***Language***/** アプリケーション用の CDE アイコンへのシンボリック・リンクが入っています。

フラグ

項目	説明
-s *ApplicationRoot*	*ApplicationRoot* にあるアプリケーションを統合します。このフラグは必須です。
-t *TargetPath*	アプリケーション CDE 構成ファイルを、アプリケーション固有の位置から、システム位置ではなく *TargetPath* にリンクします。このフラグはオプションです。 **-t** フラグを指定すると、ファイルが指定されたサブディレクトリーの下にリンクされます。例えば、**-t /etc/dt/private** と指定すると、アプリケーション・ヘルプ・ファイルが **/etc/dt/private/help/***Language* の下にシンボリック・リンクが作成されます。一般にこのフラグは、アプリケーションのインストール後スクリプトではなく、別のアプリケーションを作成したいシステム管理者が使用します。デフォルト (**-t** を指定しない状態) では、アプリケーション・サブディレクトリーのルートは、アプリケーション・ホストに対してグローバルです。ホストにインストールされているすべてのアプリケーションは、その構成ファイルを他のアプリケーションの構成ファイルとマージできるように、同じ場所にコピーされます。

項目	説明
-l *Language*	統合する言語を指定します。基本的に、このフラグはアプリケーションの CDE 構成ファイルの検索対象となるディレクトリーを示します。このパラメーターを指定しなければ、すべての言語が統合されます。このパラメーターはオプションです。
-u	アプリケーションの統合を取り消します。このフラグはオプションです。

dtlogin コマンド

目的

CDE ログイン・サービスを実行します。

構文

dtlogin [**-config** *configuration_file*] [**-daemon**] [**-debug** *debug_level*] [**-error** *error_log_file*] [**-nodaemon**] [**-resources** *resource_file*] [**-server** *server_entry*] [**-session** *session_program*] [**-udpPort** *port_number*]

説明

dtlogin コマンドがサポートする主要なタスクは次のとおりです。

- 明示的管理対象のローカルおよびリモート・ディスプレイ、および XDMCP 管理対象のリモート・ディスプレイの **dtgreet** ログイン画面を起動する。
- GUI ログイン画面から従来型端末 (キャラクター) ログインにアクセスする。
- システム依存ユーザーの認証およびログインを行う。
- 選択されたセッションを起動する。

dtlogin コマンドは、キャラクター端末で **init**、**getty**、および **login** が提供するサービスと同様のサービスを提供します。これらのサービスには、ログインおよびパスワードのプロンプト、ユーザーの認証、セッションの実行が含まれます。セッション は、特定プロセスのライフタイムによって定義されます。従来のキャラクター・ベースの端末の世界では、セッションはユーザーのログイン・シェル・プロセスです。DT のコンテキストでは、セッションは DT セッション・マネージャーです。 DT セッション・マネージャーを使用しない場合、終了オプションのあるウィンドウ・マネージャーまたはシェルを実行する端末エミュレーターのいずれかが、代表的な代替プログラムとなります。端末エミュレーターのライフタイムは実行されるシェル・プロセスのライフタイムです。 この場合、X セッションはキャラクター・ベース端末セッションのエミュレーションに縮小されます。セッションが終了すると、**dtlogin** は X サーバーをリセットし、(オプションで) プロセス全体を再始動します。

dtlogin コマンドは、X Display Manager Control Protocol Version 1.0 (XDMCP) を使用してリモート・ディスプレイの管理をサポートします。 **dtlogin** は、XDMCP から間接照会を受け取ると、チューザー・プロセスを実行して、ディスプレイに代わって XDMCP BroadcastQuery (または指定されたホストへの XDMCP Query) を実行することができ、XDMCP ディスプレイ管理を提供する可能なホストのメニューを提供することができます。この機能は、ホスト・メニューを提供しない X 端末で役に立ちます。

dtlogin はユーザーに表示される最初のインターフェースを提供するものであるため、簡単に使用でき、特定のサイトのニーズに従って容易にカスタマイズできるように設計されています。

ログイン・ウィンドウ

ログイン・ウィンドウを使用すると、ユーザーは、ユーザー ID とパスワードを入力し、起動セッションを選択し、起動ロケールを選択することができます。また、ユーザーは、X サーバーをリセットしたり、X サーバーを一時的に中断してキャラクター・ログイン・プロンプトにアクセスしたりすることができます。

ログイン・ウィンドウの内容は次のとおりです。

ログイン・フィールド
ユーザーが ID を入力できる入力フィールドを提供します。

パスワード・フィールド
ユーザーがパスワードを入力できる入力フィールドを提供します (エコーなし)。

「**OK (**了解**)**」ボタン
ユーザーを認証し、セッションを起動します。

「**Clear (**消去**)**」ボタン
ログイン・フィールドおよびパスワード・フィールドを消去します。

オプション
ユーザーはロケール名とログイン・セッション・タイプを選択できます。また、ユーザーは、X サーバーを再始動したり、キャラクター・ログイン・プロンプト (ローカル・ディスプレイ用) に切り替えたりすることができます。「Options (オプション)」メニューの内容は次のとおりです。

「**Languages (**言語**)**」
「Languages (言語)」メニューを表示します。ログイン画面の「Options (オプション)」メニューから言語を選択すると、ログイン画面が即時にローカライズされ、次のセッションのために **LANG** 変数が設定されます。ログイン画面のローカリゼーションと **LANG** の設定は、セッションが終了するとデフォルト値に戻ります。このメニューの内容は、システムにインストールされたロケールによって異なる場合があります。ロケールは **languageList** リソースを使用して指定変更することができます。 C のデフォルト・ロケールは、language リソースを使用して指定変更できます。指定されたシステム・ロケールまたは **languageList** ロケールは、「Languages (言語)」メニューのメニュー項目として表示されます。 **languageName** リソースを使用すると、特定のロケール名について、表示される代替テキストを指定できます。

「**No-windows (**ウィンドウなし**)**」
キャラクター・ログイン・プロンプトを表示します (ローカル・ディスプレイのみ)。

「**Reload Login (**再ロード・ログイン**)**」
X サーバーを再始動し、ログイン画面に戻ります。

「**Resources (**リソース**)**」
使用されるリソースをリストします。

「**Sessions (**セッション**)**」
「Sessions (セッション)」メニューを表示します。ユーザーは、ログインしたときに開始するセッション・タイプを選択できます。以下のメニュー項目があります。

「**DT Session (DT** セッション**)**」
通常のデスクトップ・セッション (Xsession) を開始します。

「**Fail-safe Session (**復旧セッション**)**」
復旧セッション (Xfailsafe) を開始します。

「**Help (**ヘルプ**)**」
ヘルプ・メッセージを表示します。

サーバーの制御

dtlogin コマンドは、POSIX シグナルを使用してローカル・サーバーを制御します。 SIGHUP シグナルにより、サーバーがリセットされ、すべてのクライアント接続がクローズされ、その他のクリーンアップ処理が実行されると予想されます。また、SIGTERM シグナルにより、サーバーが終了すると予想されます。しかし、これらのシグナルが予想通りのアクションを実行しない場合は、**resetSignal** および **termSignal** リソースで代替シグナルを指定することができます。

XDMCP を使用しないリモート・サーバーを制御するためには、**dtlogin** はディスプレイ上のウィンドウ階層を検索し、KillClient X プロトコル要求を使用して、次のセッション用に端末のクリーンアップを試みます。しかし、ウィンドウが作成されたクライアントのみが検出されるため、すべてのクライアントが実際に kill されるとは限りません。 XDMCP は、より確実なメカニズムを提供します。**dtlogin** が初期接続をクローズすると、セッションが終了し、端末は他のすべての接続のクローズが必要になります。

dtlogin の制御

dtlogin コマンドは、SIGHUP と SIGTERM の 2 つのシグナルに応答します。 **dtlogin** は、SIGHUP に送信されると、構成ファイルおよび servers リソースによって指定されたファイルを再読み取りして、追加または除去されたエントリーがあるかどうか判別します。新しいエントリーが追加された場合、**dtlogin** は関連したディスプレイ上でセッションを開始します。除去されたエントリーは即時に使用不可にされます。つまり、進行中のセッションは通知なしに終了し、新しいセッションは開始されません。 SIGTERM に送信された場合は、**dtlogin** は進行中のすべてのセッションを終了し、コマンドも終了します。これはシステムのシャットダウンに使用できます。

国際化対応

すべてのラベルおよびメッセージはローカライズ可能です。 **dtlogin.cat** メッセージ・カタログには、デフォルト・ラベルおよびデフォルト・メッセージのローカライズされた表記が含まれています。 **dtlogin** コマンドは、**LANG** 環境変数で指示された適切なメッセージ・カタログを読み取って、ローカライズされた文字列を表示します。認証画面のオプションの 1 つを使用すると、ユーザーはデフォルトの言語を後続セッション用に指定変更することができます。認証画面が選択された言語用にローカライズされている場合、画面はその言語で再表示されます。画面がローカライズされていない場合は、デフォルトの言語で表示されます。いずれの場合も、**LANG** 環境変数は結果のセッション用に適切に設定されます。

ディスプレイのデフォルトの言語を変更するには、dtlogin 構成ファイル内でリソース言語が使用可能です。認証画面に表示されるデフォルトの言語セットを指定変更するには、**dtlogin** 構成ファイル内で **languageList** リソースが使用可能です。ロケール名から「Language (言語)」メニューで表示されるテキストへのマッピングを提供するために、**languageName** リソースが使用可能です。

認証と監査

dtlogin コマンドは、PAM によって使用可能にされるサービス名 **dtlogin** のログイン・サービスです。 **dtlogin** クライアントは、従来のローカル UNIX ログインおよび監査のほかに、PAM 認証をサポートします。 Kerberos や B1 などの追加の認証または監査機能が、個別ベンダーによって追加されることがあります。

システム全体の認証に PAM を使用するには、root ユーザー・アクセス権を確立し、**/etc/security/login.cfg** ファイルの **usw** スタンザの *auth_type* 属性の値を `PAM_AUTH` に変更します。

PAM が使用可能であるときに使用される認証メカニズムは、**/etc/pam.conf** 内のログイン・サービスの構成によって異なります。 **dtlogin** コマンドは、**auth**、**account**、**password**、および **session** モジュール・タイプに関する **/etc/pam.conf** エントリーを必要とします。 **dtlogin** サービス用の **/etc/pam.conf** では、次の構成が推奨されます。

```
dtlogin        auth              required       /usr/lib/security/pam_aix
dtlogin        account           required       /usr/lib/security/pam_aix
dtlogin        password          required       /usr/lib/security/pam_aix
dtlogin        session           required       /usr/lib/security/pam_aix
```

X サーバーのセキュリティー

X サーバーでは、ユーザー・ベースのアクセス制御とホスト・ベースのアクセス制御の両方が提供されます。デフォルトでは、**dtlogin** は X サーバーに対してユーザー・ベースのアクセス制御 (MIT-MAGIC-COOKIE-1) を使用します。このレベルのセキュリティーでは、ユーザー単位でのアクセス制御が可能です。このアクセス制御方式では、クライアントが渡した許可データがサーバー上の許可データに一致した場合に、クライアントはアクセスを許可されます。ユーザーがログインすると、この許可データはデフォルトで **$HOME/.Xauthority** ファイルに保管されて、保護されます。

しかし、非セキュア・ネットワークのある環境では、ホスト・ベースのアクセス制御メカニズムの使用が望ましい場合があります。ユーザー・ベースのアクセス制御の場合、どのようなホストでも秘密鍵を発見しさえすれば接続が可能になるという弱点があります。また、ユーザー・ベースのアクセス制御には、R2 または R3 クライアントがサーバーに接続できないという欠点もあります。

authorize リソースは、**dtlogin** でユーザー・ベースのアクセス制御とホスト・ベースのアクセス制御のいずれを使用するかを制御します。詳しくは、xhost コマンドおよび xauth コマンドを参照してください。

「Resources (リソース)」

dtlogin コマンドは **dtlogin** 構成ファイルの内容によって制御されます。このファイルのデフォルトは **/usr/dt/config/Xconfig** です。リソースの一部は **dtlogin** の一般的な動作を制御します。その他のリソースは特定のディスプレイ用として指定できます。

一般リソース

以下の **dtlogin** 一般リソースは、ディスプレイ固有ではなく、該当する場合はすべてのディスプレイに適用されます。

項目 | 説明

accessFile

クラス: AccessFile

クラス型:
string

デフォルト:
Null

説明: 無許可 XDMCP サービスを防止し、XDMCP IndirectQuery 要求の転送を可能にするために、このファイルには、このマシンへの直接アクセスを許可されているホスト名のデータベース、または照会の転送先のホストのリストを持っているホスト名のデータベースが入っています。フォーマットの記述については、Xaccess ファイルのセクションを参照してください。このリソースを設定しない場合、すべてのホストが XDMCP サービスを許可されます。

項目	説明
authDir	クラス: AuthDir クラス型: string デフォルト: **/var/dt** 説明: XDMCP を使用するディスプレイ用の許可ファイルを一時的に保管するために **dtlogin** が使用するディレクトリー名。
autoRescan	クラス: AutoRescan クラス型: boolean デフォルト: true 説明: セッションが終了してファイルが変更された後、**dtlogin** が構成ファイルおよびサーバー・ファイルを再スキャンするかどうかを制御します。メイン・プロセスに SIGHUP シグナルを送信して、これらのファイルの再読み取りを **dtlogin** に強制することができます。
daemonMode	クラス: DaemonMode クラス型: boolean デフォルト: false 説明: **dtlogin** コマンドは関連付けられていないデーモン・プロセスにそれ自身をメークすることができます。これは、親プロセスを fork して終了してから、ファイル・ディスクリプターをクローズし、制御端末を解放するという手順で行われます。これは **dtlogin** をデバッグしようとする場合に不便です。このリソースを False に設定すると、**daemonMode** が使用不可になります。
debugLevel	クラス: DebugLevel クラス型: int デフォルト: 0 説明: この整数リソースに非ゼロ値を指定すると、デバッグ情報の表示が可能になります。また、この指定により、デーモン・モードが使用不可になり、通常は使用されないビット・バケットへ情報がリダイレクトされます。
errorLogFile	クラス: ErrorLogFile クラス型: string デフォルト: NULL 説明: 通常、エラー出力はシステム・コンソールに送信されます。エラー出力をリダイレクトするには、このリソースを任意のファイル名に設定します。 **Xsetup**、**Xstartup**、および **Xreset** により **stderr** に送信されたすべての出力が、そのファイルに入れられます。

項目	説明
errorLogSize	クラス: errorLogSize クラス型: int デフォルト: 50 説明: このリソースはエラー・ログ・ファイルの最大サイズを K バイト単位で指定します。制限に達すると、**dtlogin** は、ファイル・サイズが最大値の 75% に縮小されるまで、ファイル内のエントリーを古いものから削除します。ファイルの切り捨てが行われた後、エラー・ログ・ファイルに (例えば cat または tail を使用して) アクセスするユーザーは、ファイルをいったん閉じてから、ファイルに記録された後続の情報を見るために再オープンしてアクセスする必要があります。
exportList	クラス: ExportList クラス型: string デフォルト: NULL 説明: スペースまたはタブで区切られた変数名のセットが入っています。名前付きの各変数は、**dtlogin** 環境から取得され、サーバーおよびセッションの環境にロードされます。詳しくは、『環境』のセクションを参照してください。
fontPathHead	クラス: FontPathHead クラス型: string デフォルト: NULL 説明: デフォルトの X サーバー・フォント・パスの前に付加される値。
fontPathTail	クラス: fontPathTail クラス型: string デフォルト: NULL 説明: デフォルトの X サーバー・フォント・パスの後に付加される値。
keyFile	クラス: KeyFile クラス型: string デフォルト: **/usr/dt/config/Xkeys** 説明: XDM-AUTHENTICATION-1 スタイルの XDMCP 認証では、**dtlogin** と端末間で秘密鍵が共用される必要があります。このリソースは、それらの値を入れるファイルを指定します。ファイルの各エントリーは、ディスプレイ名と共有鍵で構成されます。デフォルトでは、**dtlogin** には XDM-AUTHENTICATION-1 のサポートが含まれていません。これは一般に配布できない DES が必要なためです。

項目	説明
lockPidFile	クラス: LockPidFile クラス型: boolean デフォルト: true 説明: **dtlogin** がファイル・ロックを使用して **dtlogin** の複数インスタンスの並行実行を防止するかどうかを制御します。
networkDevice	クラス: NetworkDevice クラス型: string デフォルト: **/dev/dtremote** 説明: リモート接続の場合、**/etc/utmp** の `line` の値もデバイスとして **/dev** ディレクトリー内に存在することが、**finger** などのコマンドが正しく動作するために必要です。このリソースは、リモート・ディスプレイの接続時に **dtlogin** が作成する **/dev** ファイルのパス名を指定します。大部分のプラットフォームでは、このファイルは **/dev/null** へのシンボリック・リンクとして作成されます。指定値は `/dev/` で始める必要があります。それ以外の値は廃棄され、ファイルは作成されません。
pidFile	クラス: PidFile クラス型: string デフォルト: NULL 説明: メイン **dtlogin** プロセスのプロセス ID の ASCII 表記を入れるために、指定されたファイル名が作成されます。このファイルは、**dtlogin** にシグナルを送信するときに使用できます。 **dtlogin** クライアントも、ファイル・ロックを使用して、同じマシン上で複数の dtlogin が実行されるのを防止しようとします。詳しくは、**lockPidFile** リソースを参照してください。
removeDomainname	クラス: RemoveDomainname クラス型: boolean デフォルト: true 説明: XDMCP クライアントのディスプレイ名のコンピューティングでは、通常、**dtlogin** は端末用に絶対パスによるホスト名を作成します。これは紛らわしいことがあるため、ホスト名のドメイン名部分がこの変数の設定時のローカル・ホストのドメイン名と同じ場合は、**dtlogin** はドメイン名部分を除去します。
requestPort	クラス: RequestPort クラス型: int デフォルト: 177 説明: **dtlogin** が着信 XDMCP 要求を listen するときに使用する UDP ポート番号を示します。システムのデバッグの必要がない限り、このリソースのデフォルト値はそのままにしてください。

項目	説明
servers	クラス: Servers クラス型: string デフォルト: :0 Local local /system_dependent_path/X :0 説明: サーバー・エントリーをすべて (1 行に 1 つずつ) 入れたファイル名を指定するか (値がスラッシュで始まる場合)、または単一のサーバー・エントリーを指定します。各エントリーは、常時管理する必要があって XDMCP を使用しないディスプレイを示します。各エントリーの一般構文は次のとおりです。 *DisplayName DisplayClass DisplayType*[@ite] [Command [options]] それぞれの意味は次のとおりです。 *DisplayName* **-display** オプションで任意の X プログラムに渡すことができる値。この文字列は、特定のディスプレイを指定するためにディスプレイ固有のリソースで使用されます。したがって、名前の一致にとくに注意する必要があります。例えば、ユーザーの別のリソースが `Dtlogin._0.session` と指定されている場合、`:0 local /usr/bin/X11/X :0` ではなく `localhost:0 local /usr/bin/X11/X :0` を使用します。このフィールドのアスタリスク (*) は **dtlogin** によって `hostname:0` に展開されます。 *DisplayClass* ディスプレイ・クラス部分も、リソースのクラス部分としてディスプレイ固有のリソースで使用されます。類似したディスプレイの大きな集合 (例えば、X 端末のグループ) があって、それらのディスプレイのいくつかのグループ用にリソースを設定したい場合に、この方法が役に立ちます。 XDMCP を使用する場合、ディスプレイにはディスプレイ・クラスを指定する必要があります。ご使用のデバイスの場合に妥当な標準ディスプレイ・クラス文字列については、ご使用の X 端末の資料を参照してください。 *DisplayType* `local` を指定した場合は、このエントリーについて X サーバーを開始する必要があることを示します。 `remote` の値を指定した場合は、既存の X サーバーを接続する必要があることを示します。 **@ite** ローカル・ビットマップで、ユーザーはログイン画面を使用して「**Command Line Login (**コマンド・ライン・ログイン**)**」オプションを選択することができます。このオプションは、X サーバーを一時的に中断し、従来型のキャラクター `login:` プロンプトを表示します。ここでユーザーはログインして、X 関連以外のタスクを実行することができます。ユーザーがタスクを完了してログアウトすると、X サーバーが再起動され、ログイン画面が再表示されます。「**Command Line Login (**コマンド・ライン・ログイン**)**」モードをサポートするためには、ディスプレイに関連付けられた内部端末エミュレーター (ITE) デバイスが必要です。デフォルトでは、**dtlogin** は ITE デバイス「コンソール」(**/dev/console**) をディスプレイ `:0` に関連付けます。ユーザーの構成がこのデフォルトに一致しない場合は、ITE が関連付けられているディスプレイには `@device` を指定し、servers ファイルにリストされているそれ以外のすべてのディスプレイには `@none` を指定します。 **Command [options]** X サーバーを開始する文字列。 **dtlogin** クライアントは常に指定された *DisplayName* を使用して X サーバーに接続します。したがって、ユーザーは、X サーバーにオプションとして明示的な接続番号 (前の例の `:0`) を指定する必要があります。

項目	説明
sysParmsFile	クラス: SysParmsFile クラス型: string デフォルト: **/system_dependent_path** 説明: シェル・コマンドが入っているファイルを指定します。シェル・コマンドの 1 つはシステムのタイムゾーン環境変数 (**TZ**) を設定するものです。シェル構文 `TZ=` を使用してタイムゾーンが設定されている場合、**dtlogin** はこの情報を使用してユーザー・セッション用のタイムゾーンを設定できます。
timeZone	クラス: TimeZone クラス型: string デフォルト: NULL 説明: **dtlogin** 用のローカル・タイムゾーンを指定します。これは **dtlogin** の環境に **TZ** 変数の値としてロードされ、以降のすべてのセッションに継承されます。システムによっては、タイムゾーン設定の入った構成ファイル (例えば、 **/etc/src.sh**) を維持している場合があります。 **sysParmsFile** リソースも参照してください。
wakeupInterval	クラス: WakeupInterval クラス型: int デフォルト: 10 説明: ユーザーがログイン画面から「**Command Line Login (**コマンド・ライン・ログイン**)**」モードを選択した場合、**dtlogin** は X サーバーを終了し、従来型のキャラクター・ベースのログイン・プロンプト `login:` が可視になります。ユーザーが **wakeupInterval** 秒の 2 倍の時間内にログインしなかった場合は、X サーバーが再起動されます。ユーザーがログインした後、**dtlogin** は **wakeupInterval** 秒ごとに、ユーザーがログアウトしたかどうかを検査します。ユーザーがログアウトした場合は、X サーバーが再起動され、ログイン画面が再表示されます。

ディスプレイ・リソース

dtlogin コマンドのディスプレイ・リソースは、すべてのディスプレイまたは特定の 1 つのディスプレイについて指定することができます。特定のディスプレイを指定するには、`Dtlogin` と最終リソース名セグメントの間のリソース名にディスプレイ名を挿入します。例えば、`Dtlogin.expo_0.startup` は、`expo:0` ディスプレイ上の起動シェル・ファイルを定義するリソースの名前です。リソース・マネージャーは、リソースの名前をその値からコロンで分離し、リソース名の各部分をドットで区切ります。そのため、**dtlogin** は、リソース名を生成するときにドット (**.**) およびコロン (**:**) の代わりに下線 (**_**) を使用します。

また、ディスプレイ名の代わりにクラス名を挿入することにより、クラス用にリソースを指定することもできます。 XDMCP の管理対象でないディスプレイの場合は、servers リソースで参照されるファイル内にクラス所属を指定することができます。 XDMCP を使用するディスプレイでは、XDMCP パケットの一部としてクラス所属が提供されます。

以下の **dtlogin** 一般リソースは、ディスプレイ固有ではなく、該当する場合はすべてのディスプレイに適用されます。

項目	説明
authorize	クラス: Authorize クラス型: boolean デフォルト: false 説明: authorize は、**dtlogin** がサーバー接続の許可を生成して使用するかどうかを制御するブール・リソースです。 **authName** リソースも参照してください。
authName	クラス: AuthName クラス型: string デフォルト: **MIT-MAGIC-COOKIE-1** 説明: **authorize** リソースを使用する場合、**authName** は使用される許可のタイプを指定します。現在、**dtlogin** は MIT-MAGIC-COOKIE-1 許可のみをサポートします。 XDM-AUTHORIZATION-1 のサポートも可能ですが、DES は一般に配布できません。 動的にサポートされる許可のタイプは XDMCP 接続で決まるため、**authName** はこの場合無視されます。 **authorize** リソースも参照してください。
authFile	クラス: AuthFile クラス型: string デフォルト: NULL 説明: **-auth** サーバー・コマンド・ライン・オプションを使用して、**dtlogin** からサーバーへ許可データを伝達します。このリソースが消去されるとサーバーの許可メカニズムが使用不可になるため、書き込み保護ディレクトリーに保持して、消去を防止します。 NULL の場合は、**dtlogin** はファイル名を生成します。
chooser	クラス: Chooser クラス型: デフォルト: 説明: 特殊なホスト名 CHOOSER にリダイレクトされる間接照会用ホスト・メニューを提供するプログラム実行を指定します。デフォルトは **/usr/dt/bin/dtchooser** です。 Xaccess ファイルのセクションを参照してください。
cpp	クラス: Cpp クラス型: string デフォルト: system dep. 説明: **xrdb** が使用する C プリプロセッサーのパスを指定します。

項目	説明
environment	クラス: Environment クラス型: string デフォルト: system dep. 説明: スペースまたはタブで区切られた *name=value* のペアのセットが入っています。各は、サーバーおよびセッションの環境にロードされます。詳しくは、『環境』のセクションを参照してください。
failsafeClient	クラス: FailsafeClient クラス型: string デフォルト: **/system_dep./xterm** 説明: デフォルトのセッションが実行に失敗すると、**dtlogin** はこのプログラムにフォールバックします。このプログラムは引数なしで実行されますが、セッションが使用するはずだったものと同じ環境変数を使用して実行されます。
grabServer	クラス: GrabServer クラス型: boolean デフォルト: true 説明: セキュリティーを向上させるために、**dtlogin** は、名前とパスワードの読み取り中にサーバーおよびキーボードをグラブします。 **grabServer** リソースは、名前とパスワードの読み取り中にサーバーを保留するかどうかを指定します。 false の場合、サーバーはキーボードのグラブが成功した後でグラブ解除されます。False でない場合は、サーバーはセッション開始の直前までグラブされます。
grabTimeout	クラス: GrabTimeout クラス型: int デフォルト: 3 秒 説明: **dtlogin** がグラブの成功を待機する最大時間を指定します。別のクライアントそのサーバーをグラブしている場合、またはおそらくネットワーク待ち時間が非常に長い場合にも、グラブが失敗することがあります。 **grabTimeout** リソースのデフォルトは 3 秒です。ユーザーが画面上のよく似たウィンドウに惑わされることがあるので、このリソースは慎重に使用する必要があります。グラブが失敗すると、**dtlogin** はサーバー (可能な場合) およびセッションを kill してから再起動します。一部の X 端末は、サーバーのグラブ中にログイン画面を表示できません。 **grabServer** を False に設定すれば画面の表示が可能になりますが、ログイン画面の内容をコピーされてユーザーのログイン名が盗まれるという可能性が生じます。キーボードはグラブされたままであり、パスワードはエコーされないので、パスワードが盗まれることはありません。

項目	説明
language	クラス: Language クラス型: string デフォルト: system dep. 説明: **LANG** 環境変数のデフォルトの設定値を指定します。 **dtlogin** 画面がその言語用にローカライズされている場合は、その言語で適切に表示されます。それ以外の場合は、C 言語で表示されます。ユーザーはログイン画面のオプションを使用して、この設定値を一時的に指定変更できます。後続のセッションが終了すると、**LANG** 変数は設定値に戻ります。
languageList	クラス: LanguageList クラス型: string デフォルト: NULL 説明: ログイン画面の「Language (言語)」メニューに表示される言語のデフォルト・セットを、ユーザーが指定変更できるようにします。特定のディスプレイで実際に使用される言語のセットが、システムにインストールされているセットより小さい場合、これが役に立ちます。リソース値は、該当の **LANG** 環境変数に有効な値のリストです。言語値は、1 つ以上のスペースまたはタブで分離する必要があります。
languageName	クラス: LanguageName クラス型: string デフォルト: NULL 説明: ログイン画面の「Language (言語)」メニューに表示されるデフォルトのロケール名を、ユーザーが代替テキストに指定変更できるようにします。この指定変更により、例えば、`En_US` という項目の代わりに `English (United States)` という項目がユーザーに表示されることになります。このリソースは、次のように、**Dtlogin** **local_name*.**languageName**: *text* という形式で指定します。 `Dtlogin*En_US.languageName: English (United States)` `Dtlogin*Fr_CA.languageName: French (Canadian)`
openDelay	クラス: OpenDelay クラス型: int デフォルト: 5 秒 説明: 応答の鈍いサーバーをオープンする継続した試行の間の所要時間 (秒) を指定します。
openRepeat	クラス: OpenRepeat クラス型: int デフォルト: 5 秒 説明: 応答の鈍いサーバーをオープンする継続した試行の回数を指定します。

項目	説明
openTimeout	クラス: OpenTimeout クラス型: int デフォルト: 30 秒 説明: 応答の鈍いサーバーのオープンを実際に試行しているときの待ち時間を指定します。この時間は、**connect** システム・コールで経過する最大時間と同じです。
pingInterval	クラス: PingInterval クラス型: int デフォルト: 5 分 説明: リモート・ディスプレイの消失を発見するために、**dtlogin** は、X 接続を使用し、XSync 要求を送信して、不定期にリモート・ディスプレイを ping します。 **pingInterval** リソースは継続した ping の試行の間の時間 (分) を指定します。
pingTimeout	クラス: PingTimeout クラス型: int デフォルト: 5 分 説明: 端末が要求に応答するのを待つ最大待ち時間 (分) を指定します。端末が応答しない場合、セッションは終了します。 **dtlogin** クライアントはローカル・ディスプレイを ping しません。サーバーが (例えば、リモート・ファイルシステム・サービスを) 待機中で ping に応答しない場合、その結果としてローカル・セッションが終了することがあってはなりません。
reset	クラス: Reset クラス型: string デフォルト: NULL 説明: セッション終了後に (root として) 実行されるプログラムを指定します。このリソースを設定しない場合、実行されるプログラムはありません。標準的な名前は **Xreset** です。『Xreset ファイル』を参照してください。
resetForAuth	クラス: ResetForAuth クラス型: boolean デフォルト: false 説明: サンプル・サーバーでの許可のオリジナル・インプリメンテーションの際、許可ファイルは、初期接続の検査時ではなく、サーバー・リセット時に再読み取りされていました。**dtlogin** はディスプレイに接続する直前に許可情報を生成するので、古いサーバーは現在の許可情報を取得しません。このリソースを使用すると、**dtlogin** はファイルのセットアップ後にサーバーに SIGHUP を送信するので、追加のサーバー・リセットが起こり、その時間内に新しい許可情報が読み取られます。

項目	説明
resetSignal	クラス: Signal クラス型: int デフォルト: 1 SIGHUP 説明: **dtlogin** がサーバーのリセットのために送信するシグナルを指定します。
resources	クラス: Resource クラス型: string デフォルト: NULL 説明: リソース・データベースとして **xrdb** によりディスプレイの画面 0 のルート・ウィンドウにロードされるファイルの名前を指定します。このリソース・データベースは認証プロシージャーの開始直前にロードされるので、ログイン・ウィンドウの外観を制御することができます。認証画面に関するセクションで、このファイルに入れるのに適した各種リソースについて説明していますので、参照してください。このリソースにはデフォルト値はありませんが、標準的な名前は **Xresources** です。
session	クラス: Session クラス型: string デフォルト: **/usr/dt/bin/Xsession** 説明: 認証ユーザー用に実行されるセッションを指定します。デフォルトでは、**/usr/dt/bin/Xsession** ファイルが実行されます。標準的な名前は **Xsession** です。Xsession ファイルを参照してください。
setup	クラス: Setup クラス型: string デフォルト: NULL 説明: 認証画面の表示前に (root として) 実行されるプログラムを指定します。デフォルトでは、実行されるプログラムはありません。標準的な名前は **Xsetup** です。 Xsetup ファイルを参照してください。
startAttempts	クラス: StartAttempts クラス型: int デフォルト: 4 説明: 応答の鈍いサーバーのオープンを試行する際の dtlogin の動作を制御するために、**openDelay**、**openRepeat**、**openTimeout**、および **startAttempts** の 4 つの数字リソースがあります。このリソースは、サーバーのオープンを中止する前にプロセス全体が実行される回数を指定します。 **openRepeat** 試行が行われた後、または特定の回の試行中に **openTimeout** 秒が経過した場合、**dtlogin** はサーバーを終了してから再起動して、再接続を試行します。このプロセスが **startAttempts** 回繰り返された時点で、ディスプレイは非活動と宣言され、使用不可にされます。

項目	説明
startup	クラス: Startup クラス型: string デフォルト: NULL 説明: 認証プロセスの成功後に (root として) 実行されるプログラムを指定します。デフォルトでは、実行されるプログラムはありません。ここで使用されるファイルの標準的な名前は **Xstartup** です。 Xstartup ファイルのセクションを参照してください。
systemPath	クラス: SystemPath クラス型: string デフォルト: system_dep._path 説明: **dtlogin** クライアントは、起動スクリプトおよびリセット・スクリプト用の **PATH** 環境変数をこのリソースの値に設定します。このエントリーには「.」がないことに注意してください。ルートをこのようにしておくと、多くのシステム侵入の試みを防ぐよい方法になります。
systemShell	クラス: SystemShell クラス型: string デフォルト: /bin/sh 説明: **dtlogin** クライアントは、起動スクリプトおよびリセット・スクリプト用の **SHELL** 環境変数をこのリソースの値に設定します。
terminateServer	クラス: TerminateServer クラス型: boolean デフォルト: false 説明: セッションが終了したとき X サーバーを (リセットする代わりに) 終了するかどうかを指定します。このオプションを使用すると、サーバーが時間と共に無限に肥大化する場合に、サーバーの連続実行の時間を制限することができます。
termSignal	クラス: Signal クラス型: int デフォルト: 15 (SIGTERM) 説明: **dtlogin** がサーバーを終了するために送信するシグナルを指定します。

項目	説明
userAuthDir	クラス: UserAuthDir クラス型: string デフォルト: /var/dt 説明: **dtlogin** は、通常のユーザー許可ファイル (**$HOME/.Xauthority**) への書き込みができない場合に、このディレクトリー内に固有ファイル名を作成し、作成されたファイルで環境変数 **XAUTHORITY** を指します。
userPath	クラス: UserPath クラス型: string デフォルト: system_dep._path 説明: **dtlogin** クライアントは、セッション用の **PATH** 環境変数をこの値に設定します。これはコロンで区切ったディレクトリーのリストでなければなりません。
xdmMode	クラス: XdmMode クラス型: boolean デフォルト: false 説明: true の場合、**$HOME/.xsession** ファイルは、ユーザーが認証されると **dtsession** からではなく **Xsession** から実行されます。
xrdb	クラス: Xrdb クラス型: string デフォルト: **/system_dep./xrdb** 説明: リソースのロードに使用するプログラムを指定します。認証画面はキーボードから *name-password* のペアを読み取ります。これは Motif ツールキットのクライアントですから、色、フォント、およびいくつかのレイアウト・オプションをこのリソースで制御できます。この画面用の一般リソースは、**resources** リソースで指定されたファイル (デフォルトでは **Xresources**) に書き込む必要があります。テキストやフォントなど、言語固有の値は、**Dtlogin app-defaults** ファイルに指定します。

ロゴ・リソース

認証画面のデフォルト・ロゴは、ユーザーの選択したビットマップまたはピックスマップで置き換えることができます。リソースを指定する際は、文字列 `Dtlogin*logo*` の前書きを付ける必要があります。

項目	説明
bitmapFile	クラス: BitmapFile クラス型: string デフォルト: NULL 説明: ロゴに使用するビットマップ・ファイルまたはピックスマップ・ファイルの絶対パス名を指定します。
background	クラス: Background クラス型: pixel デフォルト: #a8a8a8 説明: ロゴの背景色を指定します。
topShadowPixmap	クラス: topShadowPixmap クラス型: string デフォルト: 25_foreground 説明: ロゴのボーダー・シャドーに使用するピックスマップを指定します。

以下のリソースは、ログイン画面で使用するグリーティング文字列を記述します。リソースを指定する際は、文字列 `Dtlogin*greeting*` の前書きを付ける必要があります。

項目	説明
foreground	クラス: Foreground クラス型: pixel デフォルト: 黒 説明: ウェルカム・メッセージの前景色を指定します。
background	クラス: Background クラス型: pixel デフォルト: 動的 説明: ウェルカム・メッセージの背景色を指定します。デフォルトは、カラー・システムの場合はライト・グレー、モノクローム・システムの場合は白です。
fontlist	クラス: FontList クラス型: FontList デフォルト: -*-*schoolbook-medium-i-normal--18-* 説明: ウェルカム・メッセージに使用するフォントを指定します。

項目	説明
labelString	クラス: LabelString クラス型: string デフォルト: Welcome to %LocalHost% 説明: ウェルカム・メッセージに使用する文字列を指定します。テキストに改行文字 (0 を含めると、複数の行を指定することができます。テキストにトークン %LocalHost% を含めた場合、ログイン・サービスを提供するホストの名前で置き換えられます。テキストにトークン %DisplayName% を含めた場合、ディスプレイ名で置き換えられます。
perLabelString	クラス: LabelString クラス型: string デフォルト: Welcome %s 説明: 個別設定ウェルカム・メッセージに使用する文字列を指定します。このメッセージは、ユーザー名が入力された後で表示されます。 % は、入力されたユーザー名で置き換えられます。
alignment	クラス: Alignment クラス型: string デフォルト: ALIGNMENT_CENTER 説明: ウェルカム・メッセージの位置合わせに使用する文字列を指定します。有効な値は、ALIGNMENT_BEGINNING、ALIGNMENT_CENTER、および ALIGNMENT_END です。

マット・リソース

以下のリソースは、ログイン画面で使用するマット・レイアウトを記述します。リソースを指定する際は、文字列 Dtlogin*matte. の前書きを付ける必要があります。

項目	説明
width	クラス: Width クラス型: int デフォルト: 高解像度ディスプレイの場合は 806、中解像度ディスプレイの場合は 755、低解像度ディスプレイの場合は 585 説明: **login_matte** に使用する幅を指定します。
height	クラス: Height クラス型: int デフォルト: 高解像度ディスプレイの場合は 412、中解像度ディスプレイの場合は 385、低解像度ディスプレイの場合は 300 説明: **login_matte** に使用する高さを指定します。

ラベル・リソース

以下のリソースは、ログイン画面で使用するフォント・レイアウトを記述します。リソースを指定する際は、`string Dtlogin*.` という前書きを付ける必要があります。

項目	説明
labelFont	クラス: LabelFont クラス型: string デフォルト: 高解像度ディスプレイの場合は -*-swiss 742-medium-r-normal-*-140-*-p-110-*、低解像度ディスプレイの場合は -*-swiss 742-bold-r-normal-*-140-*-p-100-* 説明: ボタンおよびラベルに使用する **labelFont** を指定します。
textFont	クラス: TextFont クラス型: string デフォルト: 高解像度ディスプレイの場合は -*-prestige-medium-r-normal-*-128-72-*、低解像度ディスプレイの場合は -*-helvetica-bold-r-normal-*-100-* 説明: ボタンおよびラベルに使用する **textFont** を指定します。

フラグ

フラグは、**-config** を除き、すべて構成ファイル内でリソースとしても指定できる値を指定します。典型的な場合は、カスタマイズはコマンド・ライン・オプションではなく構成ファイルを使用して行われます。これらのフラグは、デバッグおよび 1 回限りのテストで最も役に立ちます。

項目	説明
-config *configuration_file*	残りの構成パラメーターを指定するリソース・ファイルを指定します。このファイルは、**dtlogin** のデフォルト **Xconfig** ファイルに置き換わります。詳しくは、『Xconfig ファイル』のセクションを参照してください。
-daemon	`daemonMode` リソースの値として **true** を指定します。この値により、**dtlogin** は、初めて起動されたときに (他のデーモンのホストと同様に) すべてのファイル・ディスクリプターをクローズし、制御端末を関連付け解除して、バックグラウンドに退きます。
-debug *debug_level*	*debug_level* リソースの数値を指定します。非ゼロ値を指定すると、**dtlogin** はデバッグ・ステートメントを端末に表示します。また、この指定により、**daemonMode** リソースが使用不可にされ、**dtlogin** は同期実行を強制されます。
-error *error_log_file*	*error_log_file* リソースの値を指定します。詳しくは、『Xerrors』ファイルのセクションを参照してください。
-nodaemon	リソースの値として `false` を指定します。
-resources *resource_file*	*resource_file* リソースの値を指定します。詳しくは、『Xresources ファイル』のセクションを参照してください。
-server *server_entry*	*server_entry* リソースの値を指定します。詳しくは、『Xservers ファイル』のセクションを参照してください。

項目	説明
-udpPort *port_number*	**requestPort** リソースの値を指定します。この指定により、**dtlogin** が XDMCP 要求をモニターするポート番号が設定されます。 XDMCP は既知の登録済み udp ポート 177 を使用するため、デバッグの場合を除いてこのリソースの変更は避けてください。
-session *session_program*	*session_program* リソースの値を指定します。詳しくは、『Xconfig ファイル』のセクションを参照してください。

環境変数

dtlogin コマンドは、以下のデフォルト環境でユーザーのセッションを起動します。

項目	説明
DISPLAY	関連したディスプレイ名に設定されます。
EDITOR	**/usr/dt/bin/dtpad** に設定されます。
HOME	ユーザーのホーム・ディレクトリーに設定されます。
KBD_LANG	適用可能な言語の **LANG** の値に設定されます。
LANG	現行の NLS 言語があれば、その言語に設定されます。
LC_ALL	現行の NLS 言語があれば、その言語に設定されます。
LC_MESSAGES	現行の NLS 言語があれば、その言語に設定されます。
LOGNAME	ユーザー名に設定されます。
MAIL	**/usr/mail/$USER** に設定されます (システム依存)。
PATH	**userPath** リソースの値に設定されます。
USER	ユーザー名に設定されます。
SHELL	ユーザーのデフォルト・シェルに設定されます (**/etc/passwd** から)。
TERM	**dtterm** に設定されます。
TZ	*timeZone* リソースの値またはシステム・デフォルトに設定されます。
XAUTHORITY	権限ファイルに設定されます。

環境リストへの追加

結果の環境変数の望ましいスコープに応じて、さらに次の 4 つのメソッドがあります。これらのメソッドは上記のリストを変更するか、上記のリストに追加されるものです。

- **exportList** リソースを使用すると、親プロセスから **dtlogin** プロセスに提供される変数をエクスポートすることができます。このメソッドで指定される変数は、ディスプレイの X サーバー・プロセスとユーザーのセッションの両方で使用でき、デフォルトの設定をすべて指定変更します。このリソースは、少なくとも 1 つのスペースまたはタブで区切られた *name=value* の文字列を受け入れます。
- **environment** リソースは **dtlogin** 構成ファイルで使用できます。これを使用すると、環境変数をグローバルにまたはディスプレイ単位で設定することができます。このメソッドで指定される変数は、ディスプレイの X サーバー・プロセスとユーザーのセッションの両方で使用でき、デフォルトの設定をすべて指定変更します。このリソースは、少なくとも 1 つのスペースまたはタブで区切られた *name=value* の文字列を受け入れます。この文字列を構文解析するシェルはないため、指定する値は定数でなければなりません。次に例を示します。

  ```
  Dtlogin*environment:MAIL_HOST=blanco MAIL_SERVER=pablo
  ```

  注**: LANG** 環境変数と **TZ** 環境変数には、構成ファイル内に専用のリソースがあります。これらの環境変数を環境で設定してはなりません。
- シェルに夜処理が必要な環境変数または他の環境変数の値に依存する環境変数は、起動スクリプト **Xsession** で指定できます。これらの変数はディスプレイ上のすべてのユーザーの環境にロードされます

が、X サーバー・プロセスにはロードされません。これらの変数は同じ変数の前の設定をすべて指定変更します。 **Xsession** スクリプトは、環境変数を設定するために **ksh** 構文を受け入れます。次に例を示します。

```
MAIL=/usr/mail/$USER
```

- パーソナル環境変数は、**$HOME/.dtprofile** スクリプト・ファイル内にユーザー単位で設定できます。**dtlogin** コマンドは、このファイル内のコマンドのために、**sh**、**ksh**、または **csh** のいずれかの構文を受け入れます。このファイル内のコマンドは環境変数を設定するものだけで、端末入出力を実行するものではありません。ただし、**tset** または **stty** は例外です。 **.dtprofile** の最初の行が `#!/bin/sh`、`#!/bin/ksh`、または `#!/bin/csh` の場合は、**dtlogin** は該当するシェルを使用して **.dtprofile** を構文解析します。それ以外の場合は、ユーザーのデフォルト・シェル (**$SHELL**) が使用されます。

終了状況

次の終了値が戻されます。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. CDE ログイン・サービスをデーモンとして開始するには、次のように入力します。

   ```
   /usr/dt/bin/dtlogin -daemon
   ```

2. CDE ログイン・サービスをデバッグ・モードで開始するには、次のように入力します。

   ```
   /usr/dt/bin/dtlogin -debug 1
   ```

位置

/usr/dt/bin/dtlogin

標準エラー

dtlogin コマンドが戻すエラー・メッセージは次のとおりです。

- `Login incorrect; please try again.`
- `Unable to change to home directory.`
- `Sorry. Maximum number of users already logged in.`
- `Login error, invalid user ID.`
- `Login error, invalid group ID.`
- `Login error, invalid audit ID.`
- `Login error, invalid audit flag.`
- `Logins are currently disabled.`
- `Your current password has expired.`

ファイル

dtlogin コマンドは、広い範囲の環境で動作するように設計されており、特定のシステムに適するように変更できる 1 組の構成ファイルを提供します。デフォルトの **dtlogin** 構成ファイルは **/usr/dt/config** に入っています。ただし、**Xsession** は例外で、**/usr/dt/bin** に保管されています。構成ファイルは次のとおりです。

項目	説明
/usr/dt/config/Xconfig	他の **dtlogin** 構成ファイルおよび **dtlogin** の動作を指定します。
/usr/dt/config/Xaccess	XDMCP サービスを要求しているディスプレイからのアクセスを制御します。
/usr/dt/config/Xservers	**dtlogin** が明示的に管理するディスプレイのリストが入っています。
/usr/dt/config/Xresources	ログイン画面の外観を指定するリソース定義が入っています。
/usr/dt/config/Xsetup	ログイン画面を表示する前に root として実行されるスクリプト。
/usr/dt/config/Xstartup	ユーザーが正常に認証された後で root として実行されるスクリプト。
/usr/dt/bin/Xsession	ユーザーのセッションを開始する認証ユーザーとして実行されるスクリプト。
/usr/dt/config/Xfailsafe	復旧セッションを開始する認証ユーザーとして実行されるスクリプト。
/usr/dt/config/Xreset	ユーザーのセッションが終了した後で root として実行されるスクリプト。

Xconfig ファイル

Xconfig ファイルは、**dtlogin** 用の一般リソースを含んでおり、**dtlogin** 構成ファイル・ツリーの先頭に位置します。 **Xconfig** は、他の **dtlogin** 構成ファイルおよびログ・ファイルの位置を指定し、さらに **dtlogin** の動作を指定します。他の **dtlogin** 構成ファイルおよびログ・ファイルの位置は、リソース定義によって指定されます。デフォルトは次のとおりです。

Dtlogin.errorLogFile
    /var/dt/Xerrors

Dtlogin.pidFile
    /var/dt/Xpid

Dtlogin.accessFile
    Xaccess

Dtlogin.servers
    Xservers

Dtlogin*resources
    %L/Xresources

Dtlogin*setup
    Xsetup

Dtlogin*startup
    Xstartup

Dtlogin*reset
    Xreset

Dtlogin*failsafeClient
    Xfailsafe

Dtlogin*session
    /usr/dt/bin/Xsession

accessFile、**servers**、**resources**、**setup**、**startup**、**reset**、**failsafeClient**、または **session** に指定されたパスが相対パスの場合、**dtlogin** はまずディレクトリー **/etc/dt/config** 内のファイルを検索し、次に **/usr/dt/config** を検索します。

注**:** 一部のリソースは * でコンポーネントを分離して指定されます。これらのリソースは、* をディスプレイ名で置き換えて、個別のディスプレイごとに固有にすることができます。詳しくは、『ディスプレイ・リソース』を参照してください。

デフォルトの **Xconfig** ファイルは **/usr/dt/config/Xconfig** です。システム管理者は、**/usr/dt/config/Xconfig** を **/etc/dt/config/Xconfig** にコピーし、**/etc/dt/config/Xconfig** を変更して、**Xconfig** をカスタマイズできます。デフォルトの **Xconfig** ファイルには、ベンダー固有のリソース定義および例のほかに、先行する構成ファイルおよびログ・ファイルのエントリーが入っています。

Xaccess ファイル

accessFile リソースによって指定されるデータベース・ファイルは、XDMCP サービスを要求しているディスプレイからのアクセスを制御するために **dtlogin** が使用する情報を提供します。このファイルには、直接照会およびブロードキャスト照会への応答を制御するエントリー、間接照会への応答を制御するエントリー、およびマクロ定義の、3 種類のエントリーが含まれています。

直接エントリーのフォーマットは、ホスト名またはパターンのいずれかです。パターンがホスト名と相違する点は、1 つ以上のメタキャラクターを含んでいることです (* は 0 個以上の文字の任意のシーケンスに一致し、? は任意の単一文字に一致します)。メタキャラクターはディスプレイ・デバイスのホスト名と比較されます。エントリーがホスト名の場合、比較はすべてネットワーク・アドレスを使用して行われるので、正しいネットワーク・アドレスに変換される名前はすべて使用できます。パターンの場合は、正規のホスト名だけが比較に使用されるので、別名を突き合わせないようにする必要があります。ホスト名またはパターンの前に感嘆符 (!) 文字を付けると、そのエントリーに一致するホストは除外されます。

間接エントリーにもホスト名またはパターンが入っていますが、その後に間接照会の送信先のホスト名またはマクロのリストが続きます。間接エントリーでは、ログイン画面を表示できるホストのメニューを提供するために、**dtlogin** が **dtchooser** を実行することを指定することもできます。

マクロ定義には、マクロ名と、ホスト名およびこのマクロの展開先の他のマクロのリストが入っています。マクロをホスト名から区別するために、マクロ名は % 文字で始まります。マクロはネスト可能です。

特定のディスプレイ・ホストへのアクセスを検査する場合、各エントリーが順にスキャンされ、最初に一致したエントリーが応答を決定します。間接エントリーのスキャン中は直接エントリーおよびブロードキャスト・エントリーは無視されます。また、その逆もいえます。ブランク行は無視されます。# はコメント区切り文字として処理され、その行の残りは無視されます。¥newline があると改行が無視され、複数行に及ぶ間接ホスト・リストが可能になります。

次に示すものは、Xaccess ファイルの例です。

```
#
# Xaccess - XDMCP access control file
#

#
# Direct/Broadcast query entries
#
!xtra.lcs.mit.edu # disallow direct/broadcast service for xtra
bambi.ogi.edu     # allow access from this particular display
*.lcs.mit.edu     # allow access from any display in LCS

#
```

```
# Indirect query entries
#

#define %HOSTS macro
%HOSTS                expo.lcs.mit.edu xenon.lcs.mit.edu ¥
                      excess.lcs.mit.edu kanga.lcs.mit.edu

#force extract to contact xenon
extract.lcs.mit.edu xenon.lcs.mit.edu

#disallow indirect access by xtra
!xtra.lcs.mit.edu   dummy

#all others get to choose among %HOSTS
*.lcs.mit.edu       %HOSTS
```

XDMCP アクセスが認可されている場合、X 端末用の許可情報が入っているディレクトリーを **authDir** で指定して、そのディレクトリー内に一時ファイルを作成することができます。セッションが開始されると、この一時ファイルは削除されます。

ブロードキャスト照会または間接照会用のホスト・メニューを提供しない X 端末の場合は、**chooser** プログラムを代わりに使用することができます。 **Xaccess** ファイルで、間接ホスト・リストの最初のエントリーとして CHOOSER を指定します。 **chooser** プログラムは、リスト内の残りのホスト名のそれぞれに照会要求を送信し、応答のあったすべてのホストのメニューを表示します。リストに BROADCAST という語を入れることができます。その場合、**chooser** はブロードキャストを送信します。応答のあったすべてのホストのメニューは、この場合も表示されます。オペレーティング・システムによっては、UDP パケットをブロードキャストできないものもあります。その場合、この機能は役に立ちません。

chooser プログラムを使用する **Xaccess** ファイルの一例を次に示します。

```
#offer a menu of these hosts to extract
extract.lcs.mit.edu CHOOSER %HOSTS

#offer a menu of all hosts to xtra
xtra.lcs.mit.edu    CHOOSER BROADCAST
```

chooser 用に使用するプログラムは **chooser** リソースで指定します。このプログラム用のリソースは、resources によって名前を指定されたファイルに書き込むことができます。デフォルトの **Xaccess** ファイルは **/usr/dt/config/Xaccess** です。システム管理者は、**/usr/dt/config/Xaccess** を **/etc/dt/config/Xaccess** にコピーし、次に **/etc/dt/config/Xaccess** を変更して、**Xaccess** をカスタマイズすることができます。デフォルトの **Xaccess** ファイルにはエントリーがありません。

Xservers ファイル

Xservers ファイルには、管理するディスプレイのリストが入っています。デフォルトの **Xservers** ファイルは **/usr/dt/config/Xservers** です。システム管理者は、**/usr/dt/config/Xservers** を **/etc/dt/config/Xservers** にコピーし、次に **/etc/dt/config/Xservers** を変更して、**Xservers** をカスタマイズすることができます。デフォルトの **Xservers** ファイルには、1 つのローカル・ディスプレイのエントリーが 1 つ入っています。

Xresources ファイル

Xservers ファイルには、ログイン画面の外観を指定するリソース定義が入っています。デフォルトの **Xresources** ファイルは **/usr/dt/config/Xresources** です。システム管理者は、 **/usr/dt/config/Xresources** を **/etc/dt/config/Xresources** にコピーし、次に **/etc/dt/config/Xresources** を変更して、**Xresources** をカスタマイズすることができます。

Xsetup ファイル

Xsetup ファイルは通常はシェル・スクリプトです。 root ユーザーだけがこのスクリプトを実行できます。実行の際は特にセキュリティーに注意する必要があります。このスクリプトはログイン画面が表示される前に実行されます。どんな種類の引数もこのスクリプトには渡されません。 **dtlogin** コマンドは、ログイン画面を表示する前に、このスクリプトの終了を待ちます。

デフォルトの **Xsetup** ファイルは **/usr/dt/config/Xsetup** です。システム管理者は、 **/usr/dt/config/Xsetup** を **/etc/dt/config/Xsetup** にコピーし、次に **/etc/dt/config/Xsetup** を変更して、**Xsetup** をカスタマイズすることができます。デフォルトの **Xsetup** ファイルには、ベンダー固有のコードが入っていますが、通常は、ログイン画面を表示する前に X サーバーをセットアップするコード (キーボード・マップのセットアップなど) が入っています。

Xstartup ファイル

Xstartup ファイルは通常はシェル・スクリプトです。 root ユーザーだけがこのスクリプトを実行できます。実行の際は特にセキュリティーに注意する必要があります。このスクリプトには、日替わりメッセージを表示したり、ユーザーに代わってその他のシステム・レベルの機能を実行したりするコマンドが書き込まれます。このスクリプトを使用するために、以下の環境変数が設定されます。

DISPLAY
: 関連したディスプレイ名に設定されます。

HOME
: ユーザーのホーム・ディレクトリーに設定されます。

PATH
: **systemPath** リソースの値に設定されます。

USER ユーザー名に設定されます。

SHELL
: **systemShell** リソースの値に設定されます。

どんな種類の引数もこのスクリプトには渡されません。 **dtlogin** コマンドは、ユーザー・セッションを開始する前に、このスクリプトの終了を待ちます。このスクリプトの終了値がゼロ以外の値の場合、**dtlogin** はセッションを即時に中断し、別の認証サイクルを開始します。

デフォルトの **Xstartup** ファイルは **/usr/dt/config/Xstartup** です。システム管理者は、 **/usr/dt/config/Xstartup** を **/etc/dt/config/Xstartup** にコピーし、次に **/etc/dt/config/Xstartup** を変更して、**Xstartup** をカスタマイズすることができます。デフォルトの **Xstartup** ファイルには、コンソールでセッションが実行されるユーザーに **/dev/console** の所有権を変更するコードが入っています。

Xsession ファイル

Xsession スクリプトは、ユーザーのセッションを初期化し、デスクトップ・セッション・マネージャーを呼び出します。このスクリプトは許可ユーザーのアクセス権によって実行され、いくつかの環境変数が事前設定されています。事前設定変数については、『環境変数』のリストを参照してください。

デフォルトの **Xsession** ファイルは **/usr/dt/bin/Xsession** です。システム管理者は、 **/usr/dt/bin/Xsession** を **/etc/dt/config/Xsession** にコピーし、次に **/etc/dt/config/Xsession** を変更して、**Xsession** をカスタマイズすることができます。その場合、**Xconfig** で定義されたセッション・リソースも、カスタマイズされた **Xsession** ファイルを参照するように変更する必要があります。 **Xconfig** ファイルを更新する方法については、『Xconfig ファイル』を参照してください。デフォルトの **Xsession** ファイルにはセッション初期化コードが入っています。ベンダー固有のコードも入っていますが、このファイルの一般機能は次のとおりです。

- ユーザーの **$HOME/.dtprofile** のソースとなる。
- すべての **/etc/dt/config/Xsession.d/*** スクリプトのソースとなる。
- すべての **/usr/dt/config/Xsession.d/*** スクリプトのソースとなる。
- デスクトップ・ウェルカム・クライアント **dthello** をバックグラウンドで起動する。
- アプリケーション検索パス・スクリプト **dtsearchpath** のソースとなる。
- ヘルプ・セットアップ・クライアント **dthelpgen** をバックグラウンドで起動する。
- アプリケーション・マネージャー・ディレクトリー・セットアップ・クライアント **dtappgather** をバックグラウンドで起動する。
- デスクトップ・セッション・マネージャー **dtsession** を実行する。

システム管理者は、**Xsession** ファイルのカスタマイズは行わないでください。

Xreset ファイル

Xreset スクリプトは、**Xstartup** と対称的に、ユーザー・セッションが終了した後で実行されます。 **Xreset** スクリプトは root ユーザーによって実行されるので、**Xstartup** のコマンドの効果を元に戻すコマンド (ファイル・サーバーからディレクトリーをアンマウントするコマンドなど) を入れておく必要があります。 **Xstartup** に渡された環境変数の収集は、**Xreset** にも渡されます。

デフォルトの **Xreset** ファイルは **/usr/dt/config/Xreset** です。システム管理者は、 **/usr/dt/config/Xreset** を **/etc/dt/config/Xreset** にコピーし、次に **/etc/dt/config/Xreset** を変更して、**Xreset** をカスタマイズすることができます。デフォルトの **Xreset** ファイルには、**/dev/console** の所有権を root に再変更するコードが入っています。

Xerrors ファイル

Xerrors スクリプトには、**dtlogin** からのエラー・メッセージと、**Xsetup**、**Xstartup**、または **Xreset** から **stderr** への出力が入ります。システム管理者は、このファイルの内容を **dtlogin** のトラブルシューティングに使用することができます。 **errorLogSize** リソースは、**Xerrors** ファイルのサイズを制限して、無制限に肥大化することを防ぐことができます。ファイルが要求されたサイズより大きくなって、**dtlogin** により切り捨てが行われた場合、ファイルにアクセスするユーザー (例えば、**cat** または **tail** を使用するユーザー) は、(ファイルの切り捨てが行われた後) ファイルをクローズしてからアクセスのために再オープンして、ファイルに記録された後続の情報をみる必要があります。

システム管理者は、**Xconfig** ファイルに **errorLogFile** リソースを設定して、**Xerrors** のパス名を変更することができます。

Xpid ファイル

Xpid スクリプトには、マスター **dtlogin** プロセスのプロセス ID が入っています。この ID は **dtlogin** にシグナルを送信するときに使用できます。システム管理者は、**Xconfig** ファイルに **pidFile** リソースを設定して、**Xpid** のパス名を変更することができます。

関連資料:

234 ページの『dtaction コマンド』

265 ページの『dtsession コマンド』

dtscript コマンド

目的

X Window System 環境で使用される簡単なダイアログを作成します。

構文

dtscript [**-xrm** *options*] [**-dir** *Path*] [**-file** *FileName*] [**-workspace** *WorkspaceName*]

注**: -xrm** オプション を使用する場合は、他のフラグより先に指定する必要があります。

説明

デスクトップ・スクリプトでは、ユーザーがパレットからダイアログにドラッグ・アンド・ドロップする Motif ウィジェットのサブセットがサポートされます。ダイアログ内では、どのウィジェットでも移動またはサイズ変更することができます。また、用意された特殊エディターを使用して、ウィジェットの属性を編集することもできます。

ウィジェットに必要な動作を与えるため、コールバックを入力できます。ダイアログが完了すると、デスクトップ・スクリプトはそのダイアログの dtksh コードを生成します。

フラグ

項目	説明
-dir *Path*	File Select ダイアログに表示されているデスクトップ・スクリプトの現行ディレクトリーを *Path* に設定します。
-file *FileName*	呼び出された既存のダイアログ *FileName* をロードします。*FileName* 引数は、絶対パス名、現行ディレクトリーを基準としたパス名、または **-dir** 値に関連したパス名のいずれかで指定できます。
-workspace *WorkspaceName*	デスクトップ・スクリプトを、それに対応する CDE ワークスペースにロードします。
-xrm *options*	すべての指定 (*options*) を入力できます。このフラグを使わなければ、このような指定はリソース・ファイルに入力しなければなりません。

例

ウィンドウからデスクトップ・スクリプトを起動するには、次のように入力します。

```
dtscript
```

ファイル

項目	説明
/usr/dt/bin/dtscript	**dtscript** コマンドが入っています。

dtsession コマンド

目的

CDE セッションを管理します。

構文

dtsession [*options*] ...

説明

dtsession コマンドは、ログインからログアウトまでのユーザー・セッション中に、ICCCM 1.1 準拠のセッション管理機能を提供します。このコマンドによってウィンドウ・マネージャーが開始され、ユーザーは、セッションの保管、セッションの復元、セッションのロック、スクリーンセーバーの開始、およびデスクトップ互換クライアントに対する色の割り当てを行うことができます。

注**:** デスクトップ・ログイン・マネージャー **dtlogin** は、**Xsession** スクリプトを介して **dtsession** クライアントを自動的に起動します。 **dtsession** クライアントは、既存の X サーバー上の **Xsession** スクリプトを使用して開始することもできます。 **dtsession** セッション・マネージャーはウィンドウ・マネージャーを自動的に開始します。

dtsession コマンドがサポートするタスクは次のとおりです。

- セッションを初期化する
- ウィンドウ・マネージャーを開始する
- ホーム・セッションまたは現行セッションを復元する
- コマンドが出されたときまたはタイムアウトが発生したときにセッション・ロックを提供する
- コマンドが出されたときまたはタイムアウトが発生したときにセッション・スクリーンセーバーを提供する
- 他のデスクトップ・クライアントに対して色割り当てサーバーとして機能する
- ホーム・セッションまたは現行セッションを保管する
- ログアウト時に確認ダイアログを表示する
- ログアウト時にセッション選択ダイアログを表示する
- セッションを終了する

「Sessions (セッション)」

セッションとは、ユーザーのデスクトップに存在するアプリケーション、設定値、およびリソースの集合です。セッション管理は、特殊なセッション・マネージャー (**dtsession** など) がユーザー・セッションを保管および復元できるようにする、規則とプロトコルのセットです。システムにログインしたユーザーは、ログオフしたときに使用していたものと同じ実行中のアプリケーション、設定値、およびリソースを提供されます。 ユーザーが初めてデスクトップにログインしたときは、デフォルトの初期セッションがロードされます。それ以後は、**dtsession** は現行セッションおよびホーム・セッションの概念をサポートします。

以下のセッションが定義されています。

初期セッション
: ユーザーが初めてデスクトップにログインすると、**dtsession** はシステム・デフォルト値を使用してユーザーの初期セッションを生成します。詳しくは、『セッション・リソース管理』および『セッション・アプリケーション管理』を参照してください。

現行セッション
: 実行中のユーザー・セッションは、保管されたホーム・セッションまたは保管された現行セッションからログイン時に復元された場合も、システム・デフォルトの初期セッションの場合も、常に現行セッションと見なされます。ユーザーがセッションを終了すると、ユーザーのスタイル・マネージャーの起動設定に基づいて現行セッションが自動的に保管されます。ユーザーが次回にデスクトップにログインすると、前に保管された現行セッションが再起動されます。デスクトップは前回にユーザーがログアウトしたときと同じ状態に復元されます。

ホーム・セッション
: もう 1 つのオプションでは、ユーザーがログアウトしたときの状態にかかわりなく、デスクトップはユーザーがログインするたびに同じ状態に復元されます。ユーザーは、現行セッションの状態を保管してから、ログインするたびにデスクトップでこの状態のセッションが開始されるように、スタイル・マネージャーの起動を設定することができます。

ディスプレイ固有のセッション
: 特定のディスプレイ用に特定のセッションを実行するために、ユーザーはディスプレイ固有のセッションを作成することができます。そのためには、ユーザーは **$HOME/.dt/sessions** ディレクトリーを **$HOME/.dt/***display* にコピーすることができます。ここで、display は、実際の非修飾ホスト名です (例えば、`pablo:0` は有効ですが、`pablo.gato.com:0` または `local:0` は無効です) 。ユーザーがディスプレイ `pablo:0` でログインすると、そのデザイン・パターン固有のセッションが優先されます。

ICCCM セッション管理プロトコル

アプリケーションがログアウト時に保管され、ログイン時に再始動されるためには、そのアプリケーションは単純なセッション管理プロトコルに参加している必要があります。 **dtsession** コマンドは ICCCM 1.1 セッション管理プロトコルをサポートします。

その状態の保管が必要なアプリケーションは、WM_SAVE_YOURSELF プロトコルに参加することができます。そのためには、アプリケーションは、その最上位ウィンドウの 1 つのみに WM_SAVE_YOURSELF プロパティーを設定する必要があります。セッションが保管されると、**dtsession** はアプリケーションの最上位ウィンドウに WM_SAVE_YOURSELF クライアント・メッセージを送信します。アプリケーションはその状態の静止保管に進みます。状態の保管中は、アプリケーションはユーザーとどのような対話もできません。アプリケーションはその状態をファイルに保管すると考えられますから、セッション・マネージャーは **DtSessionSavePath** という便利な関数を提供しています。この関数は、アプリケーションがその状態を保管できるファイルの絶対パス名を戻します。アプリケーションがその状態を保管している間、**dtsession** はアプリケーションからの完了通知を待ちます。保管が完了したことを **dtsession** に伝えるために、アプリケーションは最上位ウィンドウの WM_COMMAND プロパティーを更新する必要があります。

アプリケーションの最上位ウィンドウの WM_COMMAND プロパティーは 2 つの目的に奉仕します。第 1 に、このプロパティーの変化により、アプリケーションがその状態の保管を完了し、**dtsession** が次のアプリケーションに進むことが可能になったことが、**dtsession** に示されます。第 2 に、WM_COMMAND プロパティーの値には、セッションの開始時にアプリケーションを再起動するために **dtsession** が使用するコマンド・ラインが含まれていると予想されます。アプリケーションは、絶対パス名を指定して開始された場合、WM_COMMAND 値を設定するときに絶対パス名を使用する必要があります。状態の保管は必要でないが再始動が必要なアプリケーションは、アプリケーションの開始時に WM_COMMAND 値を 1 回だけ設定できます。

セッションの復元

セッションの起動時に、**dtsession** はどのセッションを復元するかを決定します。次のリストはその優先順位を示しています。

1. ディスプレイ固有の現行セッションまたはホーム・セッション
2. 現行セッションまたはホーム・セッション
3. 初期セッション

セッション・リソース管理

セッション・マネージャーは、X サーバーの RESOURCE_MANAGER プロパティーを使用し、そのプロパティーに基づいてすべてのアプリケーションにデスクトップ・リソースを使用可能にします。セッション・マネージャーは、RESOURCE_MANAGER を次のようにロードします。

1. システム・デフォルト・リソースをロードする。
2. システム管理者固有のリソースをマージする。
3. ユーザー指定のリソースをマージする。

デスクトップのデフォルト・リソースは、**/usr/dt/config/$LANG/sys.resources** ファイルにあります。これらのリソースは、RESOURCE_MANAGER プロパティーを通して各ユーザー・セッションに使用可能にされます。このファイルは、以降のデスクトップ・インストール時に無条件に上書きされるため、編集しないでください。

システム管理者は、**/etc/dt/config/$LANG/sys.resources** ファイルを作成することにより、システム・デフォルト・リソースを指定変更するか、さらに多くのリソースを指定することができます。このファイルはセッションの起動時にデスクトップ・デフォルト・リソースにマージされるため、このファイルには新しいリソース仕様または更新されたリソース仕様のみを入れるようにしてください。それはデスクトップ・デフォルト・リソース・ファイルのコピーを作成するよりも好ましいといえます。このファイルで指定されたリソースは、RESOURCE_MANAGER プロパティーを通して各ユーザー・セッションに使用可能にされます。このファイルで指定されたリソースは、デスクトップ・デフォルト・リソース・ファイルで指定されたものより優先されます。

ユーザーは、**$HOME/.Xdefaults** ファイルを編集して、デスクトップ・デフォルト・リソースとシステム管理者リソースを指定変更することができます。このファイルで指定されたリソースは、RESOURCE_MANAGER プロパティーを通してそのユーザー・セッションのみに使用可能にされ、デスクトップ・デフォルト・リソース・ファイルやシステム管理者リソース・ファイルで指定されたリソースより優先されます。

注**:** X ツールキット組み込み機能は、アプリケーション・リソースを RESOURCE_MANAGER または **$HOME/.Xdefaults** のいずれかからロードするが、両方からはロードしないことを指定します。通常、これはユーザーの **$HOME/.Xdefaults** ファイルが無視されることを意味します。しかし、セッション・マネージャーは、セッション起動時に RESOURCE_MANAGER にマージして、**$HOME/.Xdefaults** を調整します。ユーザーがその **$HOME/.Xdefaults** ファイルを変更すると、それらの変更は、ユーザーが **ReloadResources** アクションを起動するまで、新しいアプリケーションには不可視です。

ReloadResources アクションは、システム指定のリソース、システム管理者指定のリソース、およびユーザー指定のリソースと一緒に、RESOURCE_MANAGER を再ロードするように、セッション・マネージャーに指示します。それによって、システム管理者指定またはユーザー指定のリソース・ファイルに対して行われた変更が、新しいアプリケーションに使用可能になります。

セッション・アプリケーション管理

セッション起動時に、セッション・マネージャーはセッションの一部として保管されたアプリケーションをすべて再起動します。ユーザーの初期セッションの一環として復元される、システムのデフォルトのアプリケーション・セットは、**/usr/dt/config/$LANG/sys.session** ファイルにあります。このファイルは、以降のデスクトップ・インストール時に無条件に上書きされるため、編集しないでください。

システム管理者は、**/etc/dt/config/$LANG/sys.session** ファイルを作成して、ユーザーの初期セッションの一部として復元されたアプリケーションのセットを置き換えることができます。このファイルは、リソー

ス・ファイルと異なり、デスクトップ・デフォルト・ファイルを完全に置き換えるので、システム・デフォルト・ファイルのコピーを作成して必要なすべて変更を行うことができます。

ウィンドウ・マネージャー

dtsession コマンドはウィンドウ・マネージャーを開始します。デフォルトでは、**/usr/dt/bin/dtwm** が開始されます。 **wmStartupCommand** リソースを使用すれば、代替ウィンドウ・マネージャーを指定することができます。詳しくは、ワークスペース・マネージャーの仕様を参照してください。

スタイル・マネージャー

スタイル・マネージャーは、ユーザーが現行セッションについてデスクトップおよび X サーバーのさまざまな設定を変更するためのインターフェースを提供します。詳しくは、スタイル・マネージャの仕様を参照してください。

カラー・サーバー

dtsession コマンドは、デスクトップ用のカラー・サーバーとして機能し、その構成に使用できる次のリソースのセットを提供しています。

foregroundColor
前景色にピクセルを割り当てるかどうかを制御します。

dynamicColor
読み取り専用の色を割り当てるかどうかを指定します。

shadowPixmaps
トップ・シャドーまたはボトム・シャドーに色を割り当てるかどうかを指定します。

colorUse
色の割り当てを制限します。

writeXrdbColors
*background リソースと *foreground リソースをリソース・データベースに書き込むかどうかを指定します。

詳しくは、カラー・サーバー・リソースのセクションを参照してください。

セッション・ロック

dtsession コマンドは、セッションのロックを提供します。現行セッションは、フロント・パネルのロック・アイコンを押して直接ロックできます。 X サーバーがサポートしている場合は、指定された非アクティブ期間後に現行セッションをロックすることができます。セッションをアンロックするには、ユーザーは、自分のログイン・パスワード、root ユーザーのログイン・パスワード、または **keys** リソースで指定されたいずれかのユーザーのログイン・パスワードを入力する必要があります。 keys リソースについて詳しくは、『画面ロックと画面保管のリソース』を参照してください。

dtsession コマンドは、サービス名 **dtsession** の PAM 対応セッション・マネージャーです。このプログラムは、セッションのアンロックのために従来型の UNIX 認証のほかに PAM 認証をサポートします。DCE で必要とされるような追加の再認証機能を、個別ベンダーが追加することもできます。

認証に PAM を使用するようにシステム全体の構成を設定するには、root ユーザー・アクセス権を確立し、**/etc/security/login.cfg** ファイルの **usw** スタンザの *auth_type* 属性の値を PAM_AUTH に変更します。

PAM が使用可能であるときに使用される認証メカニズムは、**/etc/pam.conf** 内のログイン・サービスの構成によって異なります。**dtsession** コマンドは、**auth** モジュール・タイプに関する **/etc/pam.conf** エントリーを必要とします。 **dtsession** サービス用の **/etc/pam.conf** では、次の構成が推奨されます。

```
dtsession       auth            required        /usr/lib/security/pam_aix
```

スクリーンセーバー

dtsession コマンドは、セッション・ロックの一部として外部スクリーンセーバーをフロント・パネルから起動するためのサポートを提供します。また、X サーバーによるサポートがある場合は、指定された非アクティブ期間後のスクリーンセーバーの起動をサポートします。スクリーンセーバーをデスクトップに統合する方法については、スクリーンセーバーの仕様を参照してください。

X サーバーのスクリーンセーバー拡張機能

dtsession コマンドが指定された非アクティブ期間後にセッション・ロックまたはスクリーンセーバー起動を提供できるかどうかは、X サーバーのスクリーンセーバー拡張機能が使用可能かどうかで決まります。**dtsession** コマンドは、X Consortium Sample X11 Screen Saver Extension 1.0 と HP X Screen Saver Extension をサポートします。 **dtsession** コマンドがこれらの拡張機能の両方またはいずれかを認識できるかどうかは、ベンダーに依存します。

セッション・マネージャーの始動

dtsession コマンドは、**Xsession** スクリプトから開始する必要があります。**Xsession** はログイン・マネージャーの仕様で説明されています。 **Xsession** の開始は、デフォルトのログイン・シーケンスの一環として **dtlogin** から行うことをお勧めします。ただし、一部のシステムでは、**xinit**、**x11start**、**startx** などのプロキシー・プログラムから **Xsession** を開始することもできます。

カラー・サーバー・リソース

項目	説明

項目	説明
colorUse	クラス: ColorUse クラス型: string デフォルト: DEFAULT 説明: ユーザー・インターフェースに使用する色の数を指定します。カラー・サーバーは、次のように、画面の表示面の数に基づいてモニターのタイプを判別します。 **1**、**2**、または **3** 面 **(B_W)** 単色システムを指定します。カラー・パレットはユーザー・インターフェースに 2 つのカラー・セルを使用します。この構成では、BlackWhite (黒白) と WhiteBlack (白黒) の 2 つのカラー・パレットのみが使用可能です。これらのパレットを動的に変更することはできません。パレットを変更するには、カラー・パレットを使用しているすべてのアプリケーションを再起動する必要があります。このリソース値により、**ShadowPixmaps** は True に、**ForegroundColor** は (選択されたパレットに応じて) 黒または白のいずれかに強制されます。 **4** または **5** 面 **(LOW_COLOR)** ロー・カラー・システムを指定します。カラー・パレットには 2 つのカラー・セットがあり、黒および白 (カラー・セル 0 および 1) を含む最大 12 のカラー・セルがユーザー・インターフェースに使用されます。カラー・セルの数は、リソース **ShadowPixmaps** および **ForegroundColor** を使用して減らすことができます。 **6** 面 **(MEDIUM_COLOR)** ミディアム・カラー・システムを指定します。カラー・パレットには 4 つのカラー・セットがあり、黒および白 (カラー・セル 0 および 1) を含む最大 22 のカラー・セルがユーザー・インターフェースに使用されます。カラー・セルの数は、リソース **ShadowPixmaps** および **ForegroundColor** を使用して減らすことができます。 **7+** 面 **(HIGH_COLOR)** ハイ・カラー (多彩) システムを指定します。カラー・パレットには 8 つのカラー・セットがあり、黒および白 (カラー・セル 0 および 1) を含む最大 42 のカラー・セルがユーザー・インターフェースに使用されます。カラー・セルの数は、リソース **ShadowPixmaps** および **ForegroundColor** を使用して減らすことができます。
dynamicColor	クラス: DynamicColor クラス型: boolean デフォルト: true 説明: このリソースは、True または False の値をとります。 **dynamicColor** リソースは、使用するカラー・セルの数を減らすために使用されます。パレットを選択した後で変更しないと思われる場合は、**dynamicColor** を False に設定できます。False に設定した場合は、デスクトップ・スタイル・マネージャーを使用して色を動的に変更することはできません。選択されたパレットは次のセッションで有効になります。次回にセッションが開始されると、カラー・サーバーはすべてのクライアントが共有できる読み取り専用カラー・セルを使用するので、使用されるカラー・セルの数が減ります。

項目	説明
foregroundColor	クラス: ForegroundColor クラス型: string デフォルト: DYNAMIC 説明: このリソースは、White、Black、または Dynamic の値をとります。 **foregroundColor** リソースでは、すべてのテキスト (前景) が、ピクセル 0 または 1 (Black または White) のいずれかを使用することになるか、あるいは、ColorSet ごとに背景色に応じて変化する前景専用のカラー・セルを持つ (Dynamic) ことになります。 White または Black に設定すると、ColorSet ごとに使用されるカラー・セルの数が 1 つずつ減ります。
shadowPixmaps	クラス: ShadowPixmaps クラス型: string デフォルト: DEFAULT 説明: カラー・システムの場合、このリソースは True または False の値をとります。 True の場合、**topShadowColor** と **bottomShadowColor** は背景と同じピクセルを使用します。また、単一カラーの代わりに **topShadowPixmap** と **bottomShadowPixmap** が指定され、3-D の外観が得られます。その場合、カラー・セルの数が ColorSet ごとに 2 つずつ減ります。このリソースのデフォルトは、色面が 4 以下 (カラー・セルが 16 以下) のシステムでは True、色面が 4 を超えるシステムでは False です。
writeXrdbColors	クラス: WriteXrdbColors クラス型: boolean デフォルト: true

画面ロックと画面保管のリソース

項目	説明
keys	クラス: Keys クラス型: unsigned char デフォルト: NULL 説明: ユーザーによってロックされているときにはいつでも画面をアンロックできるキー・ホルダーをリストします。これはコンマで区切られたユーザー ID のリストです。例えば、あるセッション中に、ユーザー `kim` が次のリソースをアクティブにしている場合、ユーザー `fred` および `keith` は、`kim` が画面をロックしたときでも、それをアンロックできます。 `Dtsession*keys: fred,keith`

項目	説明
passwordTimeout	クラス: passwordTimeout クラス型: unsigned int デフォルト: 10 説明: パスワード・ダイアログが画面から除去されるまでの時間 (秒) を指定します。画面がロックされると、ポインターはロック・カーソルを表示し、ユーザー・パスワードを要求するダイアログが表示されます。 *passwordTimeout* 秒以内にポインターまたはキーボードからのアクティビティーが検出されない場合は、ダイアログは画面から除去されます。ポインターまたはキーボード・イベントが検出されるとすぐにダイアログが再表示されます。 *passwordTimeout* が 0 の場合は、パスワード・ダイアログは画面がロックされている間ずっと表示されたままになります。デフォルト値は 10 秒です。

その他のリソース

項目	説明
queryServerSettings	クラス: QueryServerSettings クラス型: boolean デフォルト: false 説明: **dtsession** コマンドがログアウト時にサーバーにそのすべての設定値を照会するか、あるいはデスクトップのスタイル・マネージャーを使用して設定された設定値のみを保管するかを指定します。サーバーに照会することにより、すべての設定値が確実に保管されます。しかし、全照会を行うとパフォーマンスの低下が生じます。デフォルト値は False です。つまり、サーバーは照会されません。
saveFontPath	クラス: SaveFontPath クラス型: boolean デフォルト: false
wmStartupCommand	クラス: WmStartupCommand クラス型: executable path デフォルト: NULL 説明: ログイン時に代替ウィンドウ・マネージャーの起動が可能になります。このリソースが NULL の場合、**dtsession** は **/usr/dt/bin/dtwm** を起動します。代替起動のコマンドは次のようになります。 `Dtsession*wmStartupCommand: /usr/bin/X11/mwm` このコマンドにはシェルに対するコマンドを含めてはなりません。また、引用符で囲んではなりません。**/usr/dt/bin/dtwm** 以外のウィンドウ・マネージャーを使用した場合、クライアントは復元されますが、正しい位置に復元されないことがあります。デフォルトでは、このリソースの値は NULL です。

終了状況

次の終了値が戻されます。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. 前のセッションを復元せずにコマンド・ラインからセッション・マネージャーを起動するには、次のように入力します。

   ```
   dtsession -norestore
   ```

位置

/usr/dt/bin/dtsession

ファイル

項目	説明
/usr/dt/config/$LANG/sys.session	ユーザーの初期セッション用のアプリケーションのデスクトップ・デフォルト・セット。
/etc/dt/config/$LANG/sys.session	ユーザーの初期セッション用のシステム管理者指定のアプリケーション・セット。
/usr/dt/config/$LANG/sys.resources	デスクトップ・デフォルト・リソース。
/etc/dt/config/$LANG/sys.resources	システム管理者指定のリソース。
$HOME/.Xdefaults	ユーザー指定のリソース。 **注: dtsession** コマンドはセッション情報を **$HOME/.dt/display** または **$HOME/.dt/sessions** に保管します。これらのディレクトリーの内容をユーザーが直接編集してはなりません。
/usr/dt/app-defaults/$LANG/Dtsession	デフォルトの **dtsession** リソース。

関連資料:

234 ページの『dtaction コマンド』

239 ページの『dtlogin コマンド』

---

dtterm コマンド

目的

既存のアプリケーションの実行時サポートを提供します。

構文

dtterm [Flags...]

説明

dtterm クライアントは、ANSI X3.64-1979 および ISO 6429:1992(E) に準拠した文字端末用に書かれた既存のアプリケーションの実行時サポートを提供します。

フラグ

注**: dtterm** 端末エミュレーターは、標準 X ツールキットのすべてのコマンド・ライン・フラグならびに追加のフラグを受け入れます。これらすべてを、以下にリストします (フラグが **-** ではなく **+** で始まる場合は、そのフラグはデフォルト値に復元されます)。

項目	説明
-132	DECCOLM エスケープ・シーケンスが認識されるようにし、**dtterm** ウィンドウは正しくサイズ変更を行います。通常、80 列モードと 132 列モード間を切り換える DECCOLM エスケープ・シーケンスは、無視されます。関連するリソース: c132
+132	DECCOLM エスケープ・シーケンスが無視されるようにします。これはデフォルトの動作です。関連するリソース: c132
-aw	自動ラップアラウンドを許可することを示します。これによって、カーソルが行の右端の位置にあってテキストが出力されたとき、自動的にカーソルを次の行の先頭に折り返しできるようにします。これはデフォルトの動作です。関連するリソース: autoWrap
+aw	自動ラップアラウンドを許可しないことを示します。関連するリソース: autoWrap
-background *background_color*	端末ウィンドウのバックグラウンドならびに、スクロール・バーおよび X11 ポインター・カーソル用に使用されるデフォルトのバックグラウンドを指定します。 CDE のもとでは、このフラグのデフォルトは、原色セット選択ピクセルかバックグラウンド・ピクセルです。-bs を参照してください。 CDE でない場合は、このフラグのデフォルトは、黒色という最終的な頼みの綱をもつ *background/*Background です。 *background_color* は、使用すべき背景色を説明します。関連するリソース: background
-bd *border_color*	すべてのウィンドウ用のボーダー・カラーを指定します。 **dtwm** および **mwm** のようなウィンドウ・マネージャーが使用される場合は、シェル・ウィジェットのボーダーは見えない可能性があります。デフォルト・カラーは黒です。 *border_color* は、使用すべきボーダー・カラー (縁取りの色) を説明します。関連するリソース: borderColor
-bg *background_color*	-background と同じ。 *background_color* は、使用すべき背景色を説明します。関連するリソース: background
-bordercolor *border_color*	上記の -bd に同じ。 *border_color* は、使用すべきボーダー・カラー (縁取りの色) を説明します。関連するリソース: borderColor
-borderwidth *border_width*	シェル・ウィジェットのウィンドウのボーダー幅を指定します。この値は、**dtwm** および **mwm** のようなウィンドウ・マネージャーで指定変更することができます。デフォルトは 0 です。 *border_width* は、ウィンドウ境界の幅をピクセルで指定します。関連するリソース: borderWidth
-bs	端末ウィンドウは、端末ウィンドウの背景色に Motif 選択のカラーを背景色の代りに使用することを指定します。これはデフォルトの動作です。関連するリソース: backgroundIsSelect
+bs	端末ウィンドウは、端末ウィンドウの背景色の代りに Motif 選択のカラーを使用すべきでないことを指定します。関連するリソース: backgroundIsSelect
-bw *border_width*	-borderwidth と同じ。関連するリソース: borderWidth
-C	**/dev/console** 宛ての出力を、端末ウィンドウに変更することを指定します。これは、通常は ITE 上に表示される出力が X サーバーの表示を上書きしないようにする方法として設けられたものです。これは、出力を任意のシステムの **/dev/console** から任意の X サーバーへ送信するための一般的な機構として設けられたものではありません。

注**:** このフラグが機能するには、これの所有権と、**/dev/console** に対する読み取り/書き込みアクセス権を持っている必要があります。

項目	説明
-display *display_name*	**dtterm** が使用する X11 ディスプレイ・サーバーを指定します。デフォルトは、$DISPLAY 環境変数の値です。 *display_name* は、接続先の X11 サーバーを指定します。
-e *program_argument...*	**dtterm** の始動時にサブプロセスとして起動される実行可能プログラムを指定します。このフラグは、コマンド・ラインで最後のフラグでなければなりません。 *program_argument* は、実行すべきプログラムおよびコマンド・ラインの引数を指定します。
-fb *fontset*	太字の端末テキストを表示する際に使用する XFontSet を指定します。これは、Motif XmFontList として指定する必要があります。文字またはモノスペースのフォントだけがサポートされます。プロポーショナル・フォントを使用する際の動作は未定義です。デフォルトの太字フォントは、userFont の XLFD 名に基づいて生成されます。そのフォントが利用不能の場合は、太字テキストは (1 ピクセル・オフセットを使用して) userFont を重ね打ちして生成されます。 *fontset* は、使用すべき太字端末 XFontSet を指定します。関連するリソース: userFont
-fg *foreground_color*	端末ウィンドウの前景色ならびに、スクロール・バーおよび X11 ポインター・カーソルに使用されるデフォルトの前景色を指定します。 CDE のもとでは、このリソースのデフォルトは、原色セットのフォアグラウンド・ピクセルです。 CDE でない場合、このリソースのデフォルトは、白色という最終的な頼みの綱をもつ *foreground または *Foreground です。 *foreground_color* は、使用すべき前景色を指定します。関連するリソース: foreground
-fn *fontset*	端末テキストを表示する際に使用する XFontSet を指定します。これは、Motif XmFontList として指定する必要があります。文字またはモノスペースのフォントだけがサポートされます。プロポーショナル・フォントを使用する際の動作は未定義です。このフォントは、端末テキスト以外 (メニュー・バー、ポップアップ・メニュー、ダイアログ、など) を表示するためには使用されません。デフォルトは、XmText ウィジェットと同じ方法で、親掲示板 (XmBulletinBoard を参照) の XmNtextFontList 値を使用することです。*fontset* は、使用すべき端末 XFontSet を指定します。関連するリソース: userFont
-font *fontset*	-fn と同じ。*fontset* は、使用すべき端末 XFontSet を指定します。関連するリソース: userFont
-foreground *foreground*	-fg と同じ。*foreground* は、使用すべき前景色を指定します。関連するリソース: foreground
-geometry *geometry_string*	端末ウィンドウの推奨サイズおよび推奨位置を指定します。デフォルトのサイズは、80 文字 x 24 行です。デフォルトの位置はありません。 *geometry_string* は、使用すべき端末形状を指定します。関連するリソース: geometry
-help	**dtterm** の使用状況を要約したメッセージを表示します。
-iconic	端末エミュレーターは、最初はアイコン化して画面に表示する必要があることを指定します。関連するリソース: iconic
+iconic	端末エミュレーターは、最初は通常のウィンドウとして画面に表示する必要があることを指定します。これはデフォルトの動作です。関連するリソース: iconic
-j	ジャンプ・スクロールを使用する必要があることを指定します。ジャンプ・スクロールを使用すると、スクリーンは同時に複数行スクロールすることができます。このことにより、端末にテキストが複数行送信される場合には、スクリーンをより高速に更新することができます。ジャンプ・スクロール可能な行の最大数は、端末ウィンドウの行数に制限されます。すべての行は表示されます。これはデフォルトの動作です。関連するリソース: jumpScroll
+j	ジャンプ・スクロールを使用すべきでないことを指定します。ジャンプ・スクロールについての説明は、-j を参照してください。関連するリソース: jumpScroll
-kshMode	**ksh** モードを使用可能にすることを指定します。 **ksh** モードのもとでは、拡張修飾ビットをセットしてキーを押すと、非拡張のキー・ストロークによって生成された文字が後に続くエスケープ文字が生成されます。このフラグは、emacs および、**ksh** または **ied** の emacs コマンド・ライン・エディター・モードで使用するために設けられています。これは、拡張単一バイトの文字および複数バイトのアジアの文字を生成するための、メタ・キーの通常の使用 ¥ と矛盾します。関連するリソース: kshMode
+kshMode	**ksh** モードを使用可能にすべきでないことを指定します。これはデフォルトの動作です。関連するリソース: kshMode
-l	出力ロギングを使用可能にします。ロギングを使用可能にすると、サブプロセスから受信したすべての出力は、ファイルあるいはコマンド・パイプラインの、いずれかに記録されます (-If フラグによる指定に従う)。データはサブプロセスから直接記録されるため、これには端末伝送制御手順にしたがって送信されたすべてのエスケープ文字および復帰/改行のペアが含まれます。出力は、エスケープ・シーケンスによって使用可能あるいは使用不可にされます。関連するリソース: logging
+l	出力ロギングを使用不可にします。出力ロギングについての説明は、 -l を参照してください。このフラグはデフォルトです。関連するリソース: logging

項目	説明
-lf *file_name*	-l フラグで記述された出力ログの保管先ファイルの名前を指定します。 *file_name* が、パイプ・シンボル (|) で始まる場合は、残りの文字列はパイプのエンドポイントとして使用されるコマンドであると見なされます。デフォルトのファイル名は DttermLog*XXXXX* (ここで、*XXXXX* は **dtterm** のプロセス ID です) で、**dtterm** の始動元のディレクトリーで作成されます。最後の 5 つの文字が *XXXXX* である場合は、それらはプロセス ID で置換されます。 *file_name* は、使用すべきログ・ファイル名を指定します。関連するリソース: logFile
-ls	始動されるシェルはログイン・シェルである必要があることを示します (すなわち、 argv[0] の先頭文字はダッシュで、このシェルはシステムのプロファイルおよびユーザーの **$HOME/.profile** (**ksh** および **sh** の場合)、またはシステムの **csh.login** およびユーザーの **$HOME.login** (**csh** の場合) を読み取る必要があることを示します)。関連するリソース: loginShell
+ls	通常の (ログイン以外の) シェルを始動すべきことを指定します。これはデフォルトの動作です。関連するリソース: loginShell
-map	**dtterm** がマップ式以外である (アイコン化されている) 場合は、サブプロセスの出力に際して自分自身をマップ (アイコン化解除) する必要があることを示します。サブプロセスの出力時に、**dtterm** が自分自身をマップしない初期期間は、 mapOnOutputDelay リソースを通じて指定することができます。関連するリソース: mapOnOutput
+map	特別なマッピング動作が必要でないことを指定します。これはデフォルトの動作です。関連するリソース: mapOnOutput
-mb	ユーザーが右マージンの近くに入力した場合に、**dtterm** はマージン・ベルを鳴らす必要があることを指示します。使用される実際の距離は、-nb フラグによって指定されます。関連するリソース: marginBell
+mb	ユーザーが右マージンの近くに入力した場合に、マージン・ベルを鳴らす必要がないことを示します。これはデフォルトです。関連するリソース: marginBell
-ms *pointer_color*	端末ウィンドウ (X11) のポインター・カーソルに使用する前景色を指定します。デフォルトは端末ウィンドウの前景色を使用することです。 foreground を参照してください。 *pointer_color* は、使用するポインターの前景色を指定します。関連するリソース: pointerColor

項目	説明
-name *prog_name*	**dtterm** ウィンドウの X11 名を指定します。 *prog_name* は使用する名前です。
-nb *number*	マージン・ベルが鳴るようにする場合、右マージンから何桁目かを指定します。デフォルトは 10 文字です。関連するリソース: nMarginBell
-r	前景色と背景色が逆になって **dtterm** ウィンドウが表示されるようにします。これは、-rv フラグおよび、 -reverse フラグと同じです。
+r	通常の前景色と背景色で **dtterm** ウィンドウが表示されるようにします。これはデフォルトで、+rv フラグと同じです。
-reverse	前景色と背景色が逆になって **dtterm** ウィンドウが表示されるようにします。これは、-r フラグおよび、 -rv フラグと同じです。
-rv	前景色と背景色が逆になって **dtterm** ウィンドウが表示されるようにします。これは、オプション	グローバル・オプションを選択し、次に「windowBackground」オプション・メニューを「Inverse」に変更することと同じです。このフラグを指定して始動した **dtterm** ウィンドウでは、「Window Background」オプション・メニューが「Inverse」に設定されています。「Global Option」を参照してください。
+rv	通常の前景色と背景色で **dtterm** ウィンドウが表示されるようにします。これはデフォルトです。
-rw	反転ラップアラウンドが使用可能となるように指定します。関連するリソース: reverseWrap
+rw	反転ラップアラウンドが使用できないように指示します。これはデフォルトです。関連するリソース: reverseWrap
-Sccn	端末エミュレーターは、事前にオープンされた pty または STREAMS デバイスで実行すべきことを指定します。このフラグは、pty または STREAMS デバイスのスレーブ名が tty??の形式である場合に使用するよう設けられています (すなわち、tty 後の丁度 2 文字)。このフラグは、**dtterm** が別のアプリケーションからある方針に基づいて起動される場合に使用するためのものです。 *cc* は、pty または STREAMS デバイスのスレーブ名の最後の 2 文字を指定します。この場合、スレーブ名は tty??です。この値は無視されますが、長さは丁度 2 文字でなければなりません。 *n* は、pty または STREAMS デバイスの既にオープンされたマスター・サイドに対応するファイル・ディスクリプターの番号を指定します。

項目	説明
-Sc.n	このフラグは、上記の **-Sccn** と同じですが、さらに大きな pty 名前空間を持ったシステム用に設けられています。 *c* は、pty スレーブ名の最後のコンポーネントを指定します。この値は無視され、空である可能性があります。 *n* は、既にオープンされた pty のマスター・サイドに対応するファイル・ディスクリプターの番号を指定します。
-sb	スクロール・バーを表示する必要があることを指示します。これはデフォルトです。関連するリソース: scrollBar
+sb	スクロール・バーを表示する必要がないことを示します。関連するリソース: scrollBar
-sf	標準の VT220 エスケープ・シーケンスに代わって、ファンクション・キー用に Sun Function Key エスケープ・コードを生成することを示します。関連するリソース: sunFunctionKeys
+sf	Sun Function Key エスケープ・コードに代わって、ファンクション・キー用に標準のエスケープ・シーケンスを生成することを示します。これはデフォルトの動作です。関連するリソース: sunFunctionKeys
-sl *screens[s|l]*	端末バッファー内で、ウィンドウの長さを超過した行数を指定します。フラグ値は、オプションの接尾辞が後に続く数で構成されます。接尾辞が組み込まれていないか、または接尾辞が **l** (ell) の場合は、端末バッファーの全長はスクリーンに端末ウィンドウの長さをプラスしたものです。接尾辞が **s** (ess) の場合は、端末バッファーの全長は (スクリーンに 1 (いち) をプラスしたもの) に端末ウィンドウの長さを掛けたものです。 **dtterm** は、ウィンドウのサイズがより大きくサイズ変更されると、バッファー対ウィンドウの比率を同じに保とうとします。デフォルトは **4s** です。 *screens* は、保存すべきスクリーンあるいは行の数を指定します。関連するリソース: saveLines
-ti *term_id*	端末 ID の照会への正しい応答を選択するために使用する名前を指定します。有効な値は、vt100、vt101、vt102、および vt220 です。デフォルト値は、vt220 です。 *term_id* は、使用すべき端末 ID を指定します。
-title *title_string*	ウィンドウ・タイトルを指定します。 -e フラグを使用すると、デフォルトはプログラムのパスの最後のコンポーネントとなります。 -e フラグを使用しないと、デフォルトは、 **dtterm** を実行するのに使用される名前の最後のコンポーネント (すなわち argv[0]) となります。 *title_string* は、使用する表題を指定します。関連するリソース: title
-tm *term_modes*	端末設定のためのキーワードが入った文字列、およびこれらのキーワードが結合される文字を指定します。使用可能なキーワードには、intr、quit、erase、kill、eof、eol、swtch、start、stop、brk、susp、dsusp、rprnt、flush、weras、および lnext が含まれます。特定のアーキテクチャーに該当しないキーワードは、正しく構文解析され無視されます。制御文字は、^ の後に文字が続く形式 (例えば、^c または ^u) で指定され、 ^? は削除を示すために使用されます。これは、端末処理が開始されるたびに **stty** を行わないでデフォルトの端末設定を指定変更するのに便利です。デフォルトは null です。 *term_modes* は、端末モードの文字列を指定します。関連するリソース: ttyModes
-tn *term_name*	$TERM 環境変数に設定する名前を指定します。デフォルトは **vt220** です。*term_name* は使用すべき端末名を指定します。関連するリソース: termName
-usage	使用に関するメッセージをスクリーンに出力します。
-vb	ビジュアル・ベルが音声ベルに優先することを示します。 Control-G を受信するたびに端末ベルを鳴らす代わりに、ウィンドウが点滅します。関連するリソース: visualBell
+vb	音声ベルがビジュアル・ベルに優先することを示します。これはデフォルトの動作です。関連するリソース: visualBell
-w *border_width*	-borderwidth と同じ。 *border_width* は、ウィンドウ境界の幅をピクセルで指定します。
-xrm *resource_string*	X11 リソース・マネージャー形式のリソースをコマンド・ラインに指定できるようにします。 *resource_string* は、X11 リソース文字列を指定します。

「Resources (リソース)」

項目	説明
allowSendEvents	端末エミュレーターが、(別のアプリケーションにより生成および送信された) 合成イベントを許可することを指定します。このリソースを使用可能にすると、セキュリティー・リスクが出てきます。デフォルトは False です。
appCursorDefault	True の場合は、カーソル・キーは最初はアプリケーション・モードです。 False の場合は、それらは最初はカーソル・モードです。デフォルトは False です。
appKeypadDefault	True の場合は、キーパッドのキーは最初アプリケーション・モードです。 False の場合は、それらは最初数字モードです。デフォルトは False です。
autoWrap	自動ラップアラウンドを最初、使用可能にするか否かを指定します。デフォルトは True です。
background	端末ウィンドウの背景色ならびにスクロール・バー用に使用されるデフォルトの背景色を指定します。 CDE のもとでは、このリソースのデフォルトは、原色セットの選択ピクセルか、あるいは原色セットのバックグラウンド・ピクセルになります。 backgroundIsSelect を参照してください。デフォルトは、原色セットのバックグラウンド・ピクセルです。 CDE でない場合は、このリソースはデフォルトとして黒になります。
backgroundIsSelect	True の場合、このリソースは、端末ウィンドウが背景色の代りに Motif 選択カラーを端末ウィンドウの背景色に使用することを指定します。デフォルトは False です。
blinkRate	カーソルが明滅している間オン/オフの時間をミリ秒で指定します。 250 の値は、1 秒に 2 回カーソルを明滅させます。0 の値は、明滅をオフにします。デフォルト値は 250 です。
borderColor	ウィンドウのボーダー・カラーを定義します。 **dtwm** および **mwm** のようなウィンドウ・マネージャーが使用される場合は、ウィンドウ境界は見えない可能性があります。デフォルトは「黒」です。
borderWidth	シェル・ウィジェットのウィンドウのボーダー幅を指定します。この値は、**dtwm** および **mwm** のようなウィンドウ・マネージャーで指定変更することができます。デフォルトは 0 です。
c132	80 から 132 の間のカラムをもつウィンドウに切り替わる DECCOLM エスケープ・シーケンスを認めるべきか否かを指定します。デフォルトは False です。
charCursorStyle	テキスト・カーソルの形状を指定します。 char_cursor_box という値は、基本フォントの境界ボックスの幅と高さをもつカーソルを指定します。 char_cursor_bar の値は、基本フォントの境界ボックスの幅と 2 ピクセルの高さをもち、その頂上がベースラインの上に描かれたカーソルを指定します。デフォルトは、char_cursor_box です。
consoleMode	**/dev/console** 宛ての出力を、端末ウィンドウに変更することを指定します。これは、通常は ITE 上に表示される出力が X サーバーの表示を上書きしないようにする方法として設けられたものです。これは、出力を任意のシステムの **/dev/console** から任意の X サーバーへ送信するための一般的な機構として設けられたものではありません。このフラグが機能するには、これの所有権と、**/dev/console** に対する読み取り/書き込みアクセス権を持っている必要があることに注意してください。デフォルトは False です。
foreground	端末ウィンドウの前景色ならびに、スクロール・バー用に使用されるデフォルトの前景色およびポインター・カーソル用に使用されるカラーを指定します。 CDE のもとでは、このリソースのデフォルトは、原色セットのフォアグラウンドです。そうでない場合、デフォルトは「白」です。
geometry	端末ウィンドウの推奨サイズおよび推奨位置を指定します。デフォルトのサイズは、80 文字 x 24 行です。デフォルトの位置はありません。
iconGeometry	端末エミュレーター・アイコンの推奨位置を指定します。ウィンドウ・マネージャーはこの値を無視することがあります。デフォルトはありません。
iconic	True の場合は、アイコン化した画面上に最初端末エミュレーターを配置することを指定します。ウィンドウ・マネージャー (**dtwm** および **mwm** を含む) は、この値を無視する場合があります。デフォルトは False です。
iconicName	アイコンに名前を指定します。-e フラグを使用すると、デフォルトはプログラムのパスの最後のコンポーネントとなります。 -e フラグを使用しないと、デフォルトは、 **dtterm** を実行するのに使用される名前のベース名 (すなわち argv[0]) となります。
jumpScroll	ジャンプ・スクロールを使用する必要があることを指定します。ジャンプ・スクロールを使用すると、スクリーンは同時に複数行スクロールすることができます。このことにより、端末にテキストが複数行送信される場合には、スクリーンをより高速に更新することができます。ジャンプ・スクロール可能な最大行数は、ディスプレイの行数に限定されます。すべての行が必ず表示されます。デフォルトは True です。

項目	説明
kshMode	**ksh** モードを使用可能にすることを指定します。 **ksh** モードのもとでは、拡張修飾ビットをセットしてキーを押すと、非拡張のキー・ストロークによって生成された文字が後に続くエスケープ文字が生成されます。このフラグは、emacs 行コマンドおよび、**ksh** または **ied** の emacs コマンド・ライン・エディター・モードと一緒に使用するために設けられています。これは、拡張単一バイトの文字および複数バイトのアジア文字を生成するためのメタ・キーの通常の使用と対立します。デフォルトは False です。
logFile	後述の出力ログの書き込み先ファイルの名前を指定します。ファイル名が、パイプ・シンボル (|) で始まる場合は、残りの文字列はパイプのエンドポイントとして使用されるコマンドであることが前提となります。デフォルトのファイル名は DttermLog*XXXXX* (ここで、*XXXXX* は固有の文字列) で、サブプロセスの始動元のディレクトリーに作成されます。最後の 5 文字が *XXXXX* である場合は、それらは固有の文字列で置換されます。
ロギング	出力ロギングを使用可能にします。ロギングを使用可能にすると、サブプロセスから受信したすべての出力は、 (logFile フラグによって指定されたとおりに) ファイル、あるいはコマンド・パイプラインのいずれかに記録されます。データはサブプロセスから直接記録されるため、これには端末伝送制御手順にしたがって送信されたすべてのエスケープ文字および復帰/改行のペアが含まれます。出力は、エスケープ・シーケンスによって使用可能あるいは使用不可にされます。デフォルトは False です。
logInhibit	デバイスおよびファイルのロギングを使用禁止にすることを指定します。デフォルトは False です。
loginShell	始動されるシェルがログイン・シェルであることを指定します (すなわち、 argv[0] の先頭文字はダッシュで、このシェルに対して、システムのプロファイルおよびユーザーの **$HOME/.profile** (**ksh** および **sh** の場合)、またはシステムの **csh.login** およびユーザーの **$HOME/.login** (**csh** の場合) を読み取る必要があることを指示します)。デフォルトは False です。

項目	説明
mapOnOutput	端末エミュレーターがマップされていない (アイコン化されている) 場合、サブプロセスの出力に際して自分自身をマップする (アイコン化解除する) ことを指示します。サブプロセスの出力に際して、端末エミュレーターが自分自身をマップしない初期期間を、 mapOnOutputDelay リソースにより指定してもかまいません。デフォルトは False です。
mapOnOutputDelay	始動後に、 **dtterm** が mapOnOutput リソースを認めない秒数を指定します。これにより、ウィンドウを自動マッピングすることなく初期出力 (例えば、シェル・プロンプト) を端末に送信することができます。デフォルトは 0 (遅延なし) です。
marginBell	ユーザーが右マージンの近くに入力した場合に、ベルを鳴らすか否かを指定します。デフォルトは False です。
menuBar	プルダウン・メニューを表示する必要があることを指定します。デフォルトは True です。
menuPopup	ポップアップ・メニューを使用可能にすることを指定します。デフォルトは True です。
nMarginBell	マージン・ベルを鳴らす場合、右マージンから何文字目でマージン・ベルを鳴らすかを指定します。デフォルトは 10 です。
pointerBlank	ポインター・カーソルをブランキング・モードにする必要があることを指定します。このモードでは、ポインターを移動するとカーソルはオンになり、選択された秒数が経過した後か、あるいはキーボード入力が行われた後のいずれかでブランクになります。この時間差は、pointerBlankDelay リソースにより設定されます。デフォルトは False です。
pointerBlankDelay	ポインターを移動した後、ポインター・カーソルをブランキングするまでに待機する秒数を定義します。値 0 は、キーボード入力でのみポインターのブランキングを起動します。デフォルトは 2 秒です。
pointerColor	端末ウィンドウのポインター (X11) カーソルに使用する前景色を指定します。デフォルトは端末ウィンドウの前景色を使用することです。 foreground を参照してください。
pointerColorBackground	端末ウィンドウのポインター (X11) カーソルに使用する背景色を指定します。デフォルトは端末ウィンドウの背景色を使用することです。 background を参照してください。
pointerShape	ポインター・カーソルとして使用する X カーソル・フォント文字を指定します。これは、先行する XC_ を除去して、組み込みファイルからの文字列として指定する必要があります。デフォルトは **xterm** です。
reverseVideo	反転表示を使用するか否かを指定します。デフォルトは False です。
reverseWrap	反転ラップアラウンドを使用可能にするか否かを指定します。デフォルトは False です。

項目	説明
saveLines	端末バッファー内でウィンドウの長さを超過した行数を指定します。この値は、数値にオプションの接尾辞が続いたもので構成されます。接尾辞が組み込まれていないか、または接尾辞が **l** (ell) の場合は、端末バッファーの全長はスクリーンに端末ウィンドウの長さをプラスしたものです。接尾辞が **s** (ess) の場合は、端末バッファーの全長は (スクリーンに 1 (いち) をプラスしたもの) に端末ウィンドウの長さを掛けたものです。 **dtterm** は、ウィンドウのサイズがより大きくサイズ変更されると、バッファー対ウィンドウの比率を同じに保とうとします。デフォルトは **4s** です。
scrollBar	スクロール・バーを可視にすべきか否かを指定します。デフォルトは True です。
sunFunctionKeys	標準の VT220 エスケープ・シーケンスの代りに、ファンクション・キー用に Sun Function Key エスケープ・コードを生成するか否かを指定します。デフォルトは False です。
termId	端末 ID の照会への正しい応答を選択するために使用する名前を指定します。有効な値は、vt100、vt101、vt102、および vt220 です。デフォルト値は、vt220 です。
termName	$TERM 環境変数の名前を定義します。デフォルト値は、vt220 です。
title	ウィンドウ・タイトルを指定します。 -e フラグを使用すると、デフォルトはプログラムのパスの最後のコンポーネントとなります。 -e フラグを使用しないと、デフォルトは、 **dtterm** を実行するのに使用される名前の最後のコンポーネント (すなわち argv[0]) となります。
ttyModes	端末設定のためのキーワードが入った文字列、およびこれらのキーワードが結合される文字を指定します。使用可能なキーワードには、intr、 quit、 erase、 kill、 eof、 eol、 swtch、 start、 stop、 brk、 susp、 dsusp、 rprnt、 flush、 weras、および Inext が含まれます。特定のアーキテクチャーに該当しないキーワードは、正しく構文解析され無視されます。制御文字は、^ の後に文字が続く形式 (例えば、^c または ^u) で指定され、 ^? は削除を示すために使用されます。これは、端末処理が開始されるたびに **stty** を行わないでデフォルトの端末設定を指定変更するのにとても便利です。デフォルトは null です。
userBoldFont	太字の端末テキストを表示する際に使用する XFontSet を指定します。これは、Motif XmFontList として指定する必要があります。文字またはモノスペースのフォントだけがサポートされます。プロポーショナル・フォントを使用する際の動作は未定義です。デフォルトの太字フォントは、userFont の XLFD 名に基づいて生成されます。そのフォントが利用不能の場合は、太字テキストは (1 ピクセル・オフセットを使用して) userFont を重ね打ちして生成されます。
userFont	端末テキストを表示する際に使用する XFontSet を指定します。これは、Motif XmFontList として指定する必要があります。文字またはモノスペースのフォントだけがサポートされます。プロポーショナル・フォントを使用する際の動作は未定義です。このフォントは、非端末テキスト (メニュー・バー、ポップアップ・メニュー、ダイアログ、など) を表示するためには使用されません。 デフォルトは、XmText ウィジェットと同じ方法で、親掲示板 (XmBulletinBoard(3X) を参照) の XmNtextFontList 値を使用することです。
visualBell	ビジュアル・ベルが音声ベルに優先することを指定します。 CTRL-G を受信するたびに端末ベルを鳴らす代わりに、ウィンドウが点滅します。デフォルトは False です。

ポインターの使用法

注**: dtterm** を使用することにより、いろいろなテキストの領域を選択することができます。選択は、クライアント間通信規則マニュアル (Inter-Client Communication Conventions Manual (ICCCM)) に指定されたモデルに基づいて行われます。 **dtterm** は、1 次セレクションしかサポートしません。1 次転送を使用して選択されたテキストをコピーしたりペーストすることができます。入力は、キーボード入力として扱われ、カーソルで挿入されます。選択/挿入の操作およびこれらのデフォルト指定について、以下に説明します。

項目	説明
選択	左側ボタンを使用して、コピーするテキストを選択します。コピーするテキストの先頭にポインターを移動し、次に左側ボタンを押したままカーソルをそのテキストの終わりまで移動してから、ボタンを離します。現在選択されているテキストは、マウスを移動させずに左方ボタンを一度クリックすることにより選択解除することができます。
挿入	真ん中のボタンは、1 次セレクションからテキストをペーストし、それをキーボード入力として扱います。

アクション

項目	説明
bell *([Percentage])*	このアクションにより、指定したパーセントだけ基本ボリュームより高いまたは低いボリュームでキーボード・ベルが鳴ります。
break ( )	このアクションにより、ブレーク・シグナルが子プロセスに送信されます。
cancel ( )	このアクションにより、**CAN** (キャンセル) 文字が子プロセスに送信されます。
do ( )	このアクションにより、**Do** キーに関連したエスケープ・シーケンスを子プロセスに送信されます。
edit-key *(string)*	このアクションにより、対応する編集キーに関連したエスケープ・シーケンスが子プロセスに送信されます。これらのキーの解釈はアプリケーション固有のものです。文字列に有効な値は、find、insert、next、prior、 remove、および select です。
extend-start ( )	現在選択されているテキストの拡張機能を始動させます。 **注:** 現在の選択を拡張します。選択されるテキストの量は、マウスをクリックする回数により異なります。
function-key-execute *(num [,type])*	このアクションにより、対応するファンクション・キー *num* に関連したエスケープ・シーケンスが子プロセスに送信されます。 *num* に有効な値は、1 から 35 までです。タイプが機能するように設定されている (あるいは、まったく設定されていない) と、ファンクション・キー *num* に関連したエスケープ・シーケンスは子プロセスに送信されます。 *type* が **UDK** に設定されていると、ユーザー定義のキー *num* に関連した文字列が子プロセスに送信されます。
grab-focus ( )	このアクションにより、マウス・クリックの回数に従って、以下のいずれかが実行されます。 1 回のクリックで、選択済みテキストが選択解除され、ポインター位置に選択アンカーが設定されます。 2 回のクリックで、ワードが選択され、3 回のクリックでテキストの行が選択され、そして 4 回のクリックですべてのテキストが選択されます。
hard-reset ( )	このアクションにより、端末エミュレーターでハード・リセットが行われます。
help ( )	このアクションにより、DEC VT220 Help キーに関連したエスケープ・シーケンスが子プロセスに送信されます。このキーの解釈はアプリケーション固有のものです。
keymap *(name)*	このアクションにより、リソース名が接尾辞 Keymap (大文字小文字の区別は重要) をもつ名前である新規の変換テーブルが動的に定義されます。名前「None」はオリジナルの変換テーブルを復元します。
keypad-key-execute *(string)*	このアクションにより、対応するキー・パッド・キーに関連したエスケープ・シーケンスが子プロセスに送信されます。これらのキーの解釈はアプリケーション固有のものです。 *string* の有効な値には、以下のものが含まれます。f1-f4、space、tab、 enter、equal、multiply、add、separator、subtract、decimal、 divide、および 0 から 9 まで。
move-cursor *(direction)*	このアクションにより、対応するカーソルの動作に関連したエスケープ・シーケンスが子プロセスに送信されます。これらのキーの解釈はアプリケーション固有のものです。 *direction* の有効な値には、up、 down、backward、および forward が含まれます。
redraw-display ( )	このアクションにより、テキスト・ウィンドウの内容が再ドローされます。
scroll *(count [,units])*	このアクションにより、カウントがゼロより小さい場合はディスプレイ・メモリーがスクロールダウンされ、 *count* がゼロより大きい場合はスクロールアップされます。スクロールされる行の数は *count* および *units* に基づいて行われます。 *units* の有効な値は、page、halfpage、または line です。*units* のデフォルトは、line です。
select-adjust ( )	このアクションにより、選択が拡張されます。選択されるテキストの量は、マウスをクリックする回数により異なります。 1 クリック = CHAR (文字) 2 クリック = ワード 3 クリック = 行 4 クリック = バッファー
select-all ( )	このアクションによりすべてのテキストが選択されます。

項目	説明
select-page ( )	このアクションにより、スクリーン上のすべてのテキストが選択されます。
self-insert ( )	このアクションにより、押されたキーに関連する文字が子プロセスに送信されます。
soft-reset ( )	このアクションにより端末のソフト・リセットが行われます。
stop *(state)*	このアクションにより、子プロセスからデータを読み取るプロセスが切り替えられたり、始動、あるいは中止されます。*state* の有効な値は、toggle、on、および off です。
string *(string)*	このアクションにより、指定されたテキスト *string* が、タイプ入力されたものとして挿入されます。 *string* にホワイト・スペースまたは英数字以外の文字が入っている場合は、引用符で囲む必要があります。 *string* は、文字 0x で始まる場合は、16 進法の文字定数として解釈されます。
tab ( )	このアクションにより、タブが子プロセスに送信されます。
visual-bell ( )	このアクションにより、ウィンドウが素早く点滅されます。
Virtual Bindings	仮想キーの割り当ては取引先固有のものです。仮想割り当ては、**dtterm** ウィジェットが入力フォーカスを持っている場合は適用されません。仮想ボタンと仮想キーの割り当てについては、VirtualBindings を参照してください。

ファイル

項目	説明
/usr/bin/diff	**diff** コマンドが入っています。

関連情報:

Files コマンド

入出力リダイレクト

---

du コマンド

目的

ディスクの使用状況を要約します。

構文

du [**-a** | **-s**] [**-k**] [**-m**] [**-g**][**-l**] [**-r**] [**-x**] [**-H** | **-L**][*File ...*]

説明

du コマンドは、ファイルに使用されているブロック数を表示します。指定される *File* パラメーターが実際のディレクトリーであれば、そのディレクトリー内のすべてのファイルについて報告されます。 *File* パラメーターに何も指定されない場合は、 **du** コマンドは現行ディレクトリー内のファイルを使用します。

File パラメーターがディレクトリーである場合は、報告されるブロック数は、そのディレクトリー内のファイルに割り当てられたブロックとそのディレクトリーそのものに割り当てられたブロックの合計です。

du コマンドのオブジェクトが JFS2 スナップショット内に存在するファイルまたはディレクトリーである場合、**du** コマンドはスナップショットの作成時にそのポイント・イン・タイムのオブジェクトに関する情報を示します。この情報には、スナップショット自体が削除された場合に回復するスペースの量は含まれません。

-a フラグを指定すると、個々のファイルのブロック数が報告されます。**-a** フラグを使用するかどうかにかかわらず、 *File* パラメーターで指定した個々のファイルは常にリストされます。

-s フラグを指定すると、すべての指定されたファイル、または 1 つのディレクトリー内のすべてのファイルについて、合計ブロックが報告されます。

ブロック数には、各ファイルの間接ブロックが含まれます。ブロック数は、システムが使用するクラスター・サイズに関係なく、512 バイト単位で計算されます。 **-k** フラグを指定すると、ブロック・カウントを 1024 バイト単位で計算します。

注**:**

1. 複数のリンクを持つファイルは、1 つのエントリーについてのみカウントされ書き込まれます。
2. ブロック・カウントはファイル・サイズにのみ基づいているため、割り当て解除されたブロックは、報告されるブロック・カウントには含まれません。
3. **du** がファイル属性を取得できないか、あるいはディレクトリーを読み取れない場合は、エラーを報告し、コマンドの終了状況が影響を受けます。

フラグ

項目	説明
-a	指定されたファイルごとに、ファイルのディスク使用率が表示されます。指定されたディレクトリーごとに、ディレクトリー内 (すべてのサブディレクトリーを含む) の各個別ファイルのディスク使用率が表示されます。このフラグと、**-s** フラグを対比してください。
-g	デフォルトの 512 バイト単位ではなく、GB 単位でブロック・カウントを計算します。バイトでの各ユニットの値は非常に高いため、ディスク使用状況の出力値は、浮動小数点を使用した数になります。
-H	シンボリック・リンクがコマンド・ラインに指定されている場合は、**du** コマンドは、リンクによって参照されるファイルまたはファイル階層のサイズを計算します。
-k	デフォルトの 512 バイト単位ではなく 1024 バイト単位でブロック数の計算を行います。
-l	複数のリンクを持っているファイルの場合リンク間でブロックを均等に割り当てます。デフォルトでは、2 つ以上のリンクを持つファイルは 1 回だけカウントされます。
-L	シンボリック・リンクが、コマンド・ラインに指定されているかまたはファイル階層のトラバース中に検出された場合は、 **du** コマンドは、リンクによって参照されるファイルまたはファイル階層のサイズを計算します。
-m	デフォルトの 512 バイト単位ではなく、MB 単位でブロック・カウントを計算します。バイトでの各ユニットの値は非常に高いため、ディスク使用状況の出力値は、浮動小数点を使用した数になります。
-r	アクセスできないファイル名とディレクトリー名を報告します。これはデフォルトです。
-s	指定されたファイルごとに、ファイルのディスク使用率が表示されます。指定されたディレクトリーごとに、ディレクトリー内 (すべてのサブディレクトリーを含む) のすべてのファイルの合計ディスク使用量が表示されます。このフラグを、**-a** フラグと比較してください。
-x	ファイル・サイズを評価するときに、*File* パラメーターで指定したファイルまたはディレクトリーと同じデバイスにあるファイルのみを評価します。例えば、複数のデバイス上にファイルが入っているディレクトリーを指定することができます。この場合、**-x** フラグを指定すると、ディレクトリーと同じデバイス上にあるすべてのファイルのブロック・サイズが表示されます。

注**:**

1. **-k**、**-m**、および **-g** フラグで、すべてまたはいずれか 2 つのフラグが指定された場合は、最後に指定されたものが有効になります。**-m** および **-g** フラグが指定されているディスク使用状況の出力は、最も近い小数第 2 位に丸められます。
2. 相互に排他的なオプションである **-H** と **-L** の両方を同時に指定した場合、このコマンドはエラーを報告しません。最後に指定されたオプションにより、ユーティリティーの動作が決定します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. ディレクトリー・ツリーおよびその個々のサブツリーのディスク使用状況を要約するには、次のように入力します。

   ```
   du /home/fran
   ```

   これにより、 /home/fran ディレクトリーおよびその個々のサブディレクトリー内のディスク・ブロックの数が表示されます。
2. ディレクトリー・ツリーおよびその個々のサブツリーのディスク使用状況を 1024 バイト・ブロックで要約するには、次のように入力します。

   ```
   du -k /home/fran
   ```

   これにより、 **/home/fran** ディレクトリーおよびその個々のサブディレクトリー内の 1024 バイトのディスク・ブロックの数が表示されます。
3. ディレクトリー・ツリーおよびその個々のサブツリーのディスク使用状況を MB ブロックで要約するには、次のように入力します。

   ```
   du -m /home/fran
   ```

   これにより、 **/home/fran** ディレクトリーおよびその個々のサブディレクトリー内の MB ディスク・ブロックの数が、最も近い小数第 2 位に丸められて表示されます。
4. ディレクトリー・ツリーおよびその個々のサブツリーのディスク使用状況を GB ブロックで要約するには、次のように入力します。

   ```
   du -g /home/fran
   ```

   これにより、 **/home/fran** ディレクトリーおよびその個々のサブディレクトリー内の GB ディスク・ブロックの数が、最も近い小数第 2 位に丸められて表示されます。
5. 各ファイルのディスク使用状況を表示するには、次のように入力します。

   ```
   du  -a /home/fran
   ```

   これによって、/home/fran ディレクトリーの各ファイルおよびサブディレクトリーに含まれるディスク・ブロックの数が表示されます。ディレクトリーの横の数字は、そのディレクトリー・ツリーのディスク使用状況です。正規ファイルの横の数字は、そのファイルだけのディスク使用状況です。
6. 1 つのディレクトリー・ツリーの合計ディスク使用状況のみを表示するには、次のように入力します。

   ```
   du  -s /home/fran
   ```

   -s フラグは **du** コマンドに対して、/home/fran ディレクトリーと、その中にあるファイルの合計ディスク使用量だけを表示するように指示します。デフォルトでは、**du** コマンドがファイルまたはディレクトリーを読み取れない場合、このコマンドはエラー・メッセージを表示します。
7. /home/fran ディレクトリーのトラバース中に検出された通常ファイルに加えて、すべてのシンボリック・リンクによって参照されるファイルおよびファイル階層のディスク使用状況を表示するには、次のように入力します。

   ```
   du -L /home/fran
   ```
8. シンボリック・リンク mylink によって参照されるファイルまたはファイル階層のディスク使用状況を報告するには、次のように入力します。

```
du -H mylink
```

ファイル

項目	説明
/usr/bin/du	**du** コマンドが入っています。

関連資料:

121 ページの『df コマンド』

関連情報:

Directories コマンド

Files コマンド

---

dump コマンド

目的

オブジェクト・ファイルの選択された部分のダンプをとります。

構文

dump { **-a -c -d -g -h -l -n -o -p-r -s -t -u -v -H -R -T**} [**-z***Name* [*,Number*] [**+z***Number*]] [**-t***Index* [**+t***Index*]] [**-X** {**32**|**64**|**32_64**|**d64**|**any**}] *File ...*

注**: -z** *Name* フラグと、*Number* パラメーターの間にスペースを入れないでください。

説明

dump コマンドは、指定された *File* パラメーターの選択された部分のダンプをとります。 **dump** コマンドの処理の対象は、オブジェクト・ファイル、アーカイブ・オブジェクト・ファイル、および実行可能ファイルです。

フラグ

項目	説明
-a	指定された各アーカイブの各メンバーのアーカイブ・ヘッダーのダンプをとります。
-c	文字列テーブルのダンプをとります。
-d	各セクションの生データのダンプをとります。
-g	アーカイブ記号テーブルのグローバル記号のダンプをとります。
-h	セクションのヘッダーのダンプをとります。
-l	行番号情報のダンプをとります。
-n	すべてのローダー・セクション情報のダンプをとります。
-o	個々の任意のヘッダーのダンプをとります。
-p	ヘッダーを出力しないようにします。
-r	再配置情報のダンプをとります。
-s	生データを個々に選択してダンプします。
-t	記号テーブル・エントリーのダンプをとります。
-t*Index*	*Index* パラメーターで指定されたインデックス記号テーブル・エントリーだけのダンプをとります。 **+t** フラグと一緒に **-t** フラグを使用して、記号テーブル・エントリーの範囲を指定してください。
+t*Index*	*Index* パラメーターで終了するまでの範囲の記号エントリーのダンプをとります。この範囲は、最初の記号テーブル・エントリーまたは **-t** フラグで指定されたエントリーから始まります。
-u	*File* パラメーターの名前にアンダーラインを付けます。

項目	説明
-v	数値表記ではなく、記号表記で情報のダンプを書き出します。 **-v** フラグは、**-o** フラグと **-s** フラグ以外のすべてのフラグと併用できます。
-z*Name*[,*Number*]	*Name* パラメーターの行番号エントリー、または指定された番号で始まる一連の行番号エントリーのダンプをとります。
+z*Number*	*Number* パラメーターまでのすべての行番号のダンプをとります。
-H	ローダー・セクションのヘッダーのダンプをとります。 **-H** フラグは、実行可能ファイルのみに適用されます。
-R	ローダー・セクションの再配置エントリーのダンプをとります。 **-R** フラグは、実行可能ファイルにのみ適用されます。
-T	ローダー・セクションの記号テーブル・エントリーのダンプをとります。 **-T** フラグは実行可能テーブルにのみ適用されます。
-X *mode*	**dump** で検査するオブジェクト・ファイルのタイプを指定します。 *mode* は、次のいずれかでなければなりません。 **32** 32 ビットのオブジェクト・ファイルのみを処理します。 **64** 64 ビットのオブジェクト・ファイルのみを処理します。 **32_64** 32 ビットおよび 64 ビットの両方のオブジェクト・ファイルを処理します。 **d64** 不連続の 64 ビット XCOFF ファイル (マジック番号 = U803XTOCMAGIC) を検査します。 **any** サポートされるオブジェクト・ファイルのすべてを処理します。 デフォルトでは、32 ビットのオブジェクト・ファイルが処理されます (64 ビットのオブジェクトは無視されます)。また、*mode* は、**OBJECT_MODE** 環境変数で設定することもできます。例えば、**OBJECT_MODE=64** のとき、**dump** はすべての 64 ビットのオブジェクトを処理して、32 ビットのオブジェクトを無視します。 **-X** フラグは **OBJECT_MODE** 変数を指定変更します。

例

1. a.out ファイルの文字列テーブルのダンプをとるには、次のように入力します。

   ```
   dump -c a.out
   ```

2. XCOFF データ・セクションの内容のダンプを標準出力に書き出すには、次のように入力します。

   ```
   dump -d a.out
   ```

3. オブジェクト・ファイルのヘッダーのダンプをとるには、次のように入力します。

   ```
   dump -o a.out
   ```

4. a.out ファイルの行番号情報のダンプをとるには、次のように入力します。

   ```
   dump -l a.out
   ```

5. a.out ファイルの再配置情報のダンプをとるには、次のように入力します。

   ```
   dump -r a.out
   ```

6. a.out オブジェクト・ファイルの、テキスト・セクションの内容のダンプをとるには、次のように入力します。

   ```
   dump -s a.out
   ```

7. a.out オブジェクト・ファイルの、記号テーブル情報のダンプをとるには、次のように入力します。

   ```
   dump -t a.out
   ```

8. 記号テーブル・エントリーの、 20 から 31 までを、ヘッダー情報を付けずに出力するには、次のように入力します。

   ```
   dump -p -t20 +t30 a.out
   ```

9. lib.a 内の 64 ビットのオブジェクトのみからオブジェクト・ファイルのヘッダーのダンプをとるには、次のようにします。

```
dump -X64 -o lib.a
```

関連情報:

ar コマンド

size コマンド

a.out コマンド

ar コマンド

---

dumpcheck コマンド

目的

ダンプ・デバイスとコピー・ディレクトリーがシステム・ダンプを受信できることを確認します。ダンプを入れるのにリソースが不足していると思われる場合は、デフォルトでは、エラーがログに記録されます。

構文

/usr/lib/ras/dumpcheck [[**-l**] [**-p**] [**-t** *TimeParameters*] [**-P**]] | [**-r**]

説明

/usr/lib/ras/dumpcheck コマンドは、システム・ダンプが使用するディスク・リソースの確認のために使用されます。最大のダンプ・デバイスでも小さすぎてダンプが受信できない場合、あるいは、ダンプがページング・スペースに入る際にコピー・ディレクトリーに十分なスペースがない場合は、コマンドはログにエラーを記録します。

dumpcheck は通常は、現地時間の毎日午後 3 時に cron によって実行されます。これは、 **-r** フラグを使用して、 root の **crontab** からそれを除去するか、 **-t** *TimeParameters* を使用して、dumpcheck が実行される時刻を変更することによって、変えることができます。 SMIT から構成することもできます。 **dumpcheck** は、保守援助プログラムをインストールすると、root の **crontab** に自動的に追加されます。

最大の効果を上げるためには、**dumpcheck** を、システムの負荷が最も大きいときに実行しなければなりません。その場合には、システム・ダンプが最大サイズになっているのが普通です。また、**dumpcheck** がダンプ・サイズを監視していても、ダンプがダンプ・デバイスやコピー・ディレクトリーに適合しないという事態が起こる可能性はあります。ちょうどダンプ時刻にシステム負荷がピークになった場合に、こういう状態になることがあります。

dumpcheck 関数は、保守援助プログラムのファイルセットの一部として、自動的にインストールされます。

フラグ

項目	説明
-l	エラー・ログに警告を記録します。パラメーターが指定されていない場合は、これがデフォルトです。
-p	警告が出されたらそれを標準出力に表示します。
-P	変更が永続的に行われることを示します。つまり、この変更は、これ以降に **dumpcheck** 機能が実行されるたびに適用されます。 **-t** および **-r** フラグが指定されていれば、 **-P** フラグは不要です。 **-P** フラグが指定されている場合、**dumpcheck** は、確認せずに **crontab** エントリーを単純に変更します。
-r	この関数の **crontab** エントリーを除去して、その構成解除を有効にします。このコマンドは通常、 **cron** によって実行されます。 **-r** フラグは単独で指定しなければなりません。他のフラグと同時に使用することはできません。
-t *TimeParameters*	**dumpcheck** を実行する時刻を変更します。 *TimeParameters* フラグは、一重引用符または二重引用符で囲まなければなりません。このフラグは、**crontab** 時刻パラメーター (つまり、**crontab** ファイル内の行の最初の 5 つのパラメーター) を指定します。時刻パラメーターの形式については、 **crontab** コマンドを参照してください。 **-t** フラグは、**-r** フラグと同時には使用できません。 **-t** フラグが指定されている場合、**dumpcheck** は、確認せずに **crontab** エントリーを変更します。

セキュリティー

このコマンドを実行できるのは root ユーザーに限られます。

例

1. ダンプ・リソースを調べ、その結果をログに記録する代わりに標準出力に表示するには、次のように入力します。

   ```
   /usr/lib/ras/dumpcheck -p
   ```

   この変更を永続的に行うには、つまり、 **crontab** エントリーで変更を行うには、次のように入力します。

   ```
   /usr/lib/ras/dumpcheck -p -P
   ```

2. 月曜から金曜までの午前 9 時と午後 3 時に **dumpcheck** を実行させるには、次のように入力します。

   ```
   /usr/lib/ras/dumpcheck -t "0 9,15 * * 1-5"
   ```

   デフォルトに戻るには、次のように入力します。

   ```
   /usr/lib/ras/dumpcheck -t "0 15 * * *"
   ```

   SMIT を使用して、 **dumpcheck** の実行回数を構成することもできます。

3. このフィーチャーの実行を停止するには、次のように入力します。

   ```
   /usr/lib/ras/dumpcheck -r
   ```

   このタスクにも SMIT を使用することができます。

関連情報:

sysdumpdev コマンド

システム・ダンプ機能

---

dumpctrl コマンド

目的

システム・ダンプとライブ・ダンプを管理します。

構文

dumpctrl -k

dumpctrl -R [**l** | **s**] [**-P**]

dumpctrl -s [**-c** | **-C** *comp-path-list*] [**-l** | **-L** *comp-alias-list*] [**-t** | **-T** *type_subtype*] [**-r**] [**-u**]

dumpctrl -q*c* [**-c** *comp-path-list*] [**-l** *comp-alias-list*] [**-t** *type_subtype*] [**-r**] [**-u**] [**-p** | **-P**]

dumpctrl -q*l* [**-p** | **-P**]

dumpctrl -q*s* [**-p** | **-P**]

dumpctrl [**-P**] [*global_attribute*]

dumpctrl [**-c** *comp-path-list*] [**l** *comp-alias-list*] [**-t** *type_subtype*] [**-r**] [**-u**] [**-n** | **-p** | **-P** | **-x**] [*per-component_attribute*]

説明

ダンプ・コンポーネントには、次の 2 つのタイプがあります。

component
RAS インフラストラクチャーで指定されたコンポーネント (**ras_register()** カーネル・サービスで作成されたもの) を参照します。

legacy component
dmp_add() または **dmp_ctl()** カーネル・サービスで指定されたダンプ・コンポーネントを参照します。

dumpctrl コマンドは、どのコンポーネントがライブ・ダンプまたはシステム・ダンプに登録されるかについての情報を入手し、ダンプ特性を照会して変更するために使用します。

コンポーネントは、絶対パス名、デバイス論理の別名、タイプまたはサブタイプで指定されます。複数のコンポーネントまたはコンポーネント・リストを指定するために、複数のフラグを使用できます。

フラグ

少なくとも 1 つのフラグを指定する必要があります。

項目	説明
-c *comp-path-list*	コンポーネントをパス名で指定します。ワイルドカードを使用できます。**-c all** コマンドを使用すると、すべてのコンポーネントを指定できます。
-k	カーネルのダンプ・リストをリフレッシュします。このフラグは、デフォルトで 5 分ごとに実行されます。この時間を変更するには、root ユーザーの **crontab** コマンドを編集し、**/usr/sbin/dumpctrl -k** のエントリーを変更します。詳しくは、**crontab** コマンドを参照してください。ダンプを手動で追加したり除去した場合は、その後に **dumpctrl -k** コマンドを実行する必要があります。 システムは、前回ファイルシステムに書き込めなかったダンプをヒープ内に保持している場合、それらのダンプの書き込みを試行し、この時点でそれらのストレージ・スペースを再利用します。
-l *comp-alias-list*	コンポーネントを別名で指定します。ワイルドカードを使用できます。
-r	指定されたコンポーネントのサブコンポーネントをダンプします。

項目	説明
-q *cmd*	ライブ・ダンプまたはシステム・ダンプの属性を照会します。 • **-qc** フラグは、コンポーネント固有のライブ・ダンプ属性とシステム・ダンプ属性を表示します。**-qc** フラグを **-p** または **-P** フラグと一緒に使用すると、コンポーネントごとの永続的属性を照会できます。**-qc** フラグは、**-c**、**-l**、**-t** のいずれのフラグも指定されていない場合、すべてのコンポーネントの属性を表示します。すなわち、**-c all** がデフォルトです。 このコマンドの出力例を次に示します。 `dumpctrl -qc -r -l vmm -l proc` `----------------------------------------------------------------------` `  Component name  | Have | Live Dump | System Dump` `  |Alias |  /level  |  /level` `----------------------------------------------------------------------` `vmm  | no  |  on/3  |  on/3` `  .pft  | no  |  on/3  |  on/3` `...` `proc  | no  |  on/4  |  on/3` `  ...` • **-ql** フラグはグローバル・ライブ・ダンプ設定をリストします。**-ql** フラグを **-p** または **-P** フラグと一緒に使用すると、永続的なグローバル・ライブ・ダンプ設定を照会できます。 • **-qs** *type* フラグは、グローバル・システム・ダンプ属性を表示します。**-qs** を **-p** または **-P** フラグと一緒に使用すると、グローバル・システム・ダンプ属性を照会できます。
-r	コンポーネントの階層で指定されたコンポーネントの下にコンポーネントを組み込みます。
-Rx	ダンプ設定をそのデフォルトに復元します。*x* には、ライブ・ダンプ設定では l、システム・ダンプ設定では s を指定できます。これは、グローバルなダンプ設定のみをリセットします。コンポーネントを個別に指定することはできません。これらのすべての設定を再始動後も有効にするには、**-P** フラグと新しいブート・イメージが必要です。
-t *type_subtype*	コンポーネントを *type_subtype* 名で指定します。
-s	ダンプ・リポジトリーですべてのライブ・ダンプのパス名とタイトルをリストします。コンポーネントが **-c**、**-l**、または **-t** フラグで指定されている場合は、指定されたコンポーネントのダンプのみが、表示されるダンプのリストに含まれます。コンポーネントが **-C**、**-L**、または **-T** フラグで指定されている場合は、指定された障害のあるコンポーネントのダンプのみが、表示されるダンプのリストに含まれます。
-C *comp-path-list*	コンポーネントをパス名で指定します。ワイルドカードを使用できます。予約名 all を使用してすべてのコンポーネントを指定することもできます。**-C** フラグは、**-s** フラグとともに使用する場合のみ有効です。
-L *comp-alias-list*	コンポーネントを別名で指定します。ワイルドカードを使用できます。**-L** フラグは、**-s** フラグとともに使用する場合のみ有効です。
-T *type_subtype*	コンポーネントを *type_subtype* 名で指定します。 **-T** フラグは、**-s** とともに使用する場合のみ有効です。
-u	コンポーネントの階層で指定されたコンポーネントの上にコンポーネントを組み込みます。

Persistence flags

項目	説明
-p	変更内容は、新規に作成されるコンポーネントにのみ適用されます。これは **dumpctrl** コマンドの実行後に作成される RAS インフラストラクチャーのコンポーネントです。
-P	指定された変更を永続的なものにします。変更内容は再始動後も有効のままです。新しいブート・イメージが必要な場合は、それについて通知するメッセージが生成されます。**-P** フラグが適用されるのは、コンポーネント属性、ライブ・ダンプのグローバルな使用可能化または使用不可化、グローバル・ライブ・ダンプのレベル、レガシー・コンポーネントの使用可能化または使用不可化、およびシステム・ダンプ・デバイスの指定です。
-n	変更内容は既存のコンポーネントに適用されます。**-p** または **-P** のいずれも指定しない場合は、**-n** フラグがデフォルトです。変更内容を現行のコンポーネントと新規作成のコンポーネントの両方に適用するには、**-n** フラグと **-p** フラグを使用します。
-x	この永続性の指定を削除します。**-x** フラグは、永続的な (**-P**) パーシスタンスの指定を削除します。この指定は、当初 **-P** フラグで指定したのと同じ方法で指定する必要があります。

再帰的に下るカスタマイズ (**-r** フラグにより指定) は、他のすべてのカスタマイズよりも優先されます。この場合、再帰的に下るカスタマイズではない他のカスタマイズに対してそれらが指定される順序は関係ありません。

設定されたカスタマイズが不明だが、デフォルトのシステム設定を復元したい場合は、以下のいずれかのアクションを実行できます。

- /var/adm/ras/raspertune ファイルで、カスタマイズに関連する行を削除し、**bosboot** コマンドを実行して AIX を再始動します。
- /var/adm/ras/raspertune ファイルで、指定されている該当のフラグとパラメーターを見つけます。 **-x** フラグを使用して、そのカスタマイズを削除します。 **bosboot** コマンドを実行し、AIX を再始動します。

さまざまなダンプ属性が永続性とどのように相互作用するかについては、『属性』のライブ・ダンプ属性とシステム・ダンプ属性テーブルを参照してください。

属性

ダンプ属性は attribute=value の形式を使用できます。次に例を示します。

```
dumpctrl dir=/usr/dumps freespc=20
```

この例では、ダンプ・ディレクトリーが /usr/dumps、フリー・スペースのしきい値が 20% に設定されています。

ショートカットがいくつかあります。例えば、**ldmpenable=yes** と同じ ldmpon 属性などです。

コンポーネントが指定されると、未知の属性が **RASCD_DMP_PASS_THROUGH** を使用してこれらのコンポーネントのコールバックに渡されます。

次の表では、ライブ・ダンプ属性を示します。

表 *1.* ライブ・ダンプ属性とデフォルト

属性	指定	デフォルト値
ldmpenable	ライブ・ダンプを使用可能にするかを指定します。 使用可能な値は yes と no です。 **ldmpenable=yes** の代わりに ldmpon 属性、**ldmpenable=no** の代わりに ldmpoff 属性を使用することができます。	yes 詳しくは、以下の注 (1 (294 ページ)) を参照してください。
dir	ライブ・ダンプのディレクトリー名を指定します。	/var/adm/ras/livedump
freespc	ライブ・ダンプのフリー・スペースのしきい値を 0 から 99 の 10 進数を使用して指定します。	25 (25% の意味)
ldmplevel	ライブ・ダンプのレベルを 0 から 9 の 10 進数を使用して指定します。 **ldmplevel=1, 3, 7** の代わりに、**ldmpminimal**、**ldmpnormal**、または ldmpdetail 属性を指定することができます。	3 (normal) 詳しくは、以下の注 (1 (294 ページ)) を参照してください。
heapsz	ライブ・ダンプのヒープ・サイズを 10 進数 (メガバイト) を使用して指定します。	0 詳しくは、以下の注 (2 (294 ページ)) を参照してください。

表 1. ライブ・ダンプ属性とデフォルト (続き)

属性	指定	デフォルト値
duptype	重複ダンプの抑止タイプを指定します。使用可能な値は次のとおりです。 • all • pre • post • none	all
maxfreeze	推奨されるシステム・フリーズの最大インターバルを 10 進数 (ミリ秒) を使用して指定します。	100 ms

次の表では、システム・ダンプ属性を示します。

表 2. システム・ダンプ属性とデフォルト

属性	指定	デフォルト値
sdmpenable	システム・ダンプを使用可能にするかを指定します。 使用可能な値は yes と no です。 **sdmpenable=yes** または **sdmpenable=no** の代わりに、sdmpon または sdmpoff を指定することができます。	yes 詳しくは、以下の注 (3 (294 ページ)) を参照してください。
legacyenable	ダンプのレガシー・コンポーネントを使用可能にするかを指定します。 使用可能な値は yes と no です。 **legacyenable=yes** または **legacyenable=no** の代わりに、legacyon または legacyoff を指定することができます。	yes
sdmplevel	システム・ダンプのレベルを 0 から 9 の 10 進数を使用して指定します。 **sdmplevel=1, 3, 7** の代わりに、**sdmpminimal**、**sdmpnormal**、または sdmpdetail 属性を指定することができます。	3 (normal) 詳しくは、以下の注 (4 (294 ページ)) を参照してください。
copydir	コピー・ディレクトリーのパス名を指定します。	/var/adm/ras
forcecopy	**forcecopy** 属性を使用可能にするかを指定します。 使用可能な値は yes と no です。 ダンプをブート時にページング・スペースからコピーしなければならないときに、コピー・ディレクトリーにスペースがない場合は、**forcecopy** 値が yes になっていると、そのダンプを取り外し可能メディアにコピーするようにプロンプトが出されます。この値が no になっていると、ダンプはコピーされずに、システムは通常どおりにブートします。ただし、ダンプは失われる可能性があります。	yes
keyseq	キー・シーケンスで常にダンプを行うかを指定します。 使用可能な値は yes と no です。	no
primary	1 次ダンプ・デバイスのパス名を指定します。	/dev/hd6 または /dev/lg_dumplv
secondary	2 次ダンプ・デバイスのパス名を指定します。	/dev/sysdumpnull

注:

1. **ldmpenable** 属性と **ldmplevel** 属性は、コンポーネントとともに指定するか、またはコンポーネントなしで指定することができます。コンポーネントなしで指定すると、属性は対応するグローバル属性に適用されます。
2. **heapsz** 属性 (ヒープ・サイズ) を 0 に設定することができます。ダンプの初期化時に、システムは実メモリーの量に基づいてライブ・ダンプのヒープ・サイズを計算します。これは最小で 64 MB および実メモリーのサイズの 1/64 です。
3. **sdmpenable** 属性が指定されている場合は、コンポーネントを個別に指定する必要があります。コンポーネントが指定されていない場合は、**sdmpenable** 属性を指定することはできません。システム・ダンプを使用不可にすることはできないためです。
4. **sdmplevel** 属性は、コンポーネントとともに指定するか、またはコンポーネントなしで指定することができます。コンポーネントなしで指定すると、システム・デフォルト・レベルに適用されます。**sdmplevel** がグローバル **sdmplevel** 値より大きなコンポーネントは、システム・ダンプには含まれません。

次の表では、ライブ・ダンプ属性とその永続性を示します。

表 *3.* ライブ・ダンプ属性と永続性

属性	説明	永続性
ldmpenable	ライブ・ダンプが使用可能	永続性フラグによって制御され、**-P** フラグには新しいブート・イメージが必要です。
dir	ライブ・ダンプのディレクトリー	即時およびシステムの再始動後に有効になります。
freespc	ライブ・ダンプのフリー・スペースのしきい値	即時およびシステムの再始動後に有効になります。
ldmplevel	ライブ・ダンプのレベル	永続性フラグによって制御され、**-P** フラグには新しいブート・イメージが必要です。
heapsz	ライブ・ダンプのヒープ・サイズ	即時およびシステムの再始動後に有効になります。
duptype	重複ダンプの抑止タイプ	即時およびシステムの再始動後に有効になります。
maxfreeze	推奨されるシステム・フリーズの最大インターバル	即時およびシステムの再始動後に有効になります。

注**:** 永続性が属性に影響を与えるのは、これらが RAS インフラストラクチャーのコンポーネントに適用される場合のみです。また、永続性は、グローバルなライブ・ダンプのレベルとグローバルな使用可能状況または使用不可状況を制御します。

次の表では、システム・ダンプ属性とその永続性を示します。

表 *4.* システム・ダンプ属性と永続性

属性	説明	永続性
sdmpenable	システム・ダンプが使用可能	永続性フラグによって制御され、**-P** フラグには新しいブート・イメージが必要です。
legacyenable	ダンプのレガシー・コンポーネント	即時、および **-P** フラグによるシステムの再始動後に有効になります。**-P** フラグに新しいブート・イメージは必要ありません。
sdmplevel	システム・ダンプのレベル	永続性フラグによって制御され、**-P** フラグには新しいブート・イメージが必要です。
copydir	コピー・ディレクトリー	即時およびシステムの再始動後に有効になります。
forcecopy	コピーできない場合はブート時メニューを提示する	即時およびシステムの再始動後に有効になります。
keyseq	キー・シーケンスで常にダンプを行う	即時およびシステムの再始動後に有効になります。

表 4. システム・ダンプ属性と永続性 (続き)

属性	説明	永続性
primary	1 次ダンプ・デバイス	即時、および **-P** フラグによるシステムの再始動後に有効になります。**-P** フラグに新しいブート・イメージは必要ありません。
secondary	2 次ダンプ・デバイス	即時、および **-P** フラグによるシステムの再始動後に有効になります。**-P** フラグに新しいブート・イメージは必要ありません。

注: 永続性が属性に影響を与えるのは、これらがコンポーネントに適用される場合です。

copydir、**forcecopy**、**keyseq**、**primary**、**secondary** の各属性は、**sysdumpdev** コマンドの **-d**、**-D**、**-k**、**-K**、**-p**、**-s** の各フラグで指定された、対応する属性のように機能します。詳しくは、「*Commands Reference, Volume 5*」の **sysdumpdev** コマンドを参照してください。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
ゼロ以外の値	エラーが発生しました。このコマンドは以下の条件の場合に失敗します。 • 1 つ以上のパラメーターが無効である。 • 1 つ以上の属性が無効である。 • コンポーネントを指定できない。 • 少なくとも 1 つのコンポーネントを指定する必要がある。 • 永続性の指定が見つからない。(それは **-x** フラグで発生する場合があります。)

セキュリティー

このコマンドは、root ユーザーのみが実行できます。

---

dumpfs コマンド

目的

ファイルシステム情報のダンプをとります。

構文

dumpfs { *FileSystem* | *Device* }

説明

dumpfs コマンドは、指定されたファイルシステムまたはスペシャル・デバイスのスーパーブロック、i ノード・マップ、およびディスク・マップ情報を出力します。生成されたリストは、ファイルシステム情報を見つける際に使用できます。 **dumpfs** コマンドは、主としてデバッグの目的で使用されます。

dumpfs コマンドは、 JFS2 スナップショットに対しても実行できます。**dumpfs** コマンドは、指定されたスナップショットのスーパーブロック、スナップショット・マップ、およびブロック・マップの xtree コピーを出力します。

注**: dumpfs** コマンドは、UDF、NFS、または JFS ディスケットに対しては機能しません。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

/dev/hd4 の情報を出力するには、次のように入力します。

```
dumpfs /dev/hd4
```

関連資料:

611 ページの『fsck コマンド』

関連情報:

mkfs コマンド

e

以下の AIX コマンドは文字 e で始まります。

echo コマンド

目的

標準出力に文字列を書き出します。

構文

echo [*String* ...]

説明

echo コマンドは、文字列を標準出力に書き出します。*String* はスペースで区切り、最後に指定する *String* パラメーターの後に改行文字を付けます。 *String* パラメーターを指定しないと、ブランク行 (改行文字) が表示されます。

通常、ハイフンで始まるフラグと文字列は、— (二重ハイフン) を使用することで区別できます。**echo** コマンドではフラグはサポートされていないので、— (二重ハイフン) はリテラルとして処理されます。

echo コマンドは次のエスケープ規則を認識します。

項目	説明
¥a	警告文字を表示します。
¥b	バックスペース文字を表示します。
¥c	出力の中の最後の引数の後に続く改行文字を抑止します。 **¥c** シーケンスの後に続く文字はすべて無視されます。
¥f	用紙送り文字を表示します。
¥n	改行文字を表示します。
¥r	復帰文字を表示します。
¥t	タブ文字を表示します。
¥v	垂直タブ文字を表示します。
¥¥	円記号を表示します。
¥0*Number*	ASCII 値が 0 か、1 桁、2 桁、3 桁の 8 進数である 8 ビットの文字を表示します。

注**: bsh**、**ksh**、および **csh** の各コマンドには、それぞれ組み込みの **echo** サブコマンドが含まれています。**echo** コマンドと **bsh** および **ksh** の **echo** サブコマンドは、同様に機能します。 **csh echo** サブコマンドは、**echo** コマンドと同様に機能しません。

¥ (円記号) はシェル内の引用文字です。これは、¥ をエスケープ文字と一緒に使用するか、 "¥" や '¥' のように引用符で囲まない限り、円記号は、コマンドの展開時にシェルによって取り除かれることを意味しています。

シェルの展開後に、**echo** コマンドは入力内のエスケープ・シーケンスに基づいて出力を書き出します。次の「円記号の削除」の表では、最初にシェルにより、続いて、**echo** コマンドにより、コマンド内の円記号がどのように削除されるかを示しています。

円記号の削除

入力されるコマンド	シェルによる展開後	**echo** コマンドによる処理後
echo hi¥¥¥¥there	echo hi¥¥there	hi¥there
echo 'hi¥¥¥¥there'	echo 'hi¥¥¥¥there'	hi¥¥there
echo "hi¥¥¥¥there'	echo "hi¥¥there"	hi¥there

終了状況

このコマンドは次の終了値を戻します。

項目　説明
0　正常終了。
>0　エラーが発生しました。

例

1. メッセージを標準出力に書き出すには、次のように入力します。

   ```
   echo Please insert diskette . . .
   ```

2. 特殊文字を含んでいるメッセージを表示するには、次のように入力します。

   ```
   echo "¥n¥n¥nI'm at lunch.¥nI'll be back at 1:00."
   ```

   これによって、3 行スキップされて、次のメッセージが表示されます。

   ```
   I'm at lunch.
   I'll be back at 1:00.
   ```

   注: メッセージにエスケープ・シーケンスが含まれている場合は、そのメッセージを引用符で囲まなければなりません。引用符で囲まないと、シェルは ¥ (円記号) をメタキャラクターと見なして別の処理を行います。

3. パターン・マッチング文字と一緒に、 **echo** コマンドを使用するには、次のように入力します。

   ```
   echo The back-up files are: *.bak
   ```

   このように入力すると、「The back-up files are:」というメッセージが表示され、その後に現行ディレクトリー内の .bak で終わるファイル名が表示されます。

4. 1 行のテキストをファイルに追加するには、次のように入力します。

   ```
   echo Remember to set the shell search path to $PATH. >>notes
   ```

   このように入力すると、**PATH** シェル変数の値をシェルが置換した後、ファイル注釈の終わりにメッセージが追加されます。

5. メッセージを標準エラー出力に書き込むには、次のように入力します。

   ```
   echo Error: file already exists. >&2
   ```

   このコマンドは、エラー・メッセージを標準エラーにリダイレクトします。 >&2 を省略すると、メッセージは標準出力に書き出されます。

ファイル

項目	説明
/usr/bin/echo	**echo** コマンドが入っています。

関連情報:

bsh コマンド

ksh コマンド

printf コマンド

入出力リダイレクト

---

ed または red コマンド

目的

テキスト・ファイルの行エディターです。

構文

ed [**-p** *String*] [**-s** | **-**] [*File*]

red [**-p***String*] [**-s** | **-**] [*File*]

説明

ed コマンドは行編集プログラム ed エディターを始動します。このプログラムは一度にファイルを 1 つだけ処理するもので、ファイルを一時編集バッファーにコピーして、そのコピーに対して変更を行います。ed エディターは、edit エディター、 ex エディター、 vi エディターも含まれているエディター・ファミリーに属しています。ed エディターは、ユーザーが指定する変更をバッファー内で行います。 ed エディターは、 write (**w**) サブコマンドを使用しない限り、ファイルそのものの変更は行いません。

ed コマンドで ed エディターを始動する場合、編集したいファイルの名前を指定できます。これは **e** サブコマンドでも可能です。**ed** コマンドが新しいファイルをバッファーに読み取ると、そのファイルの内容がバッファーの前の内容に入れ代わります。

red コマンドは、**ed** コマンドの制限付きバージョンで、制限付きシェル (**rsh**) と一緒に使用されます。**red** コマンドでは、現行ディレクトリー、または **/tmp** ディレクトリー内にあるファイルのみ編集できます。 **!** サブコマンドは使用できません。

ed エディター・サブコマンドは、0 個、1 個、または 2 個のアドレスから構成され、その後ろに 1 文字のサブコマンド、続いて、そのサブコマンドのオプションのパラメーターが置かれます。アドレスはバッファー内の 1 行または複数行を指定します。どのサブコマンドもデフォルト・アドレスを持っているので、ほとんどの場合アドレスを指定する必要はありません。

バッファー内の別の行をアドレス指定しない限り、ed エディターでは現在行以外は編集できません。データの移動やコピーは行単位でのみ可能です。 ed エディターは大きなファイルを編集する場合や、シェル・プログラム内で編集を行う場合に便利です。

ed エディターは次の 2 つのモードのいずれかで作動します。

項目	説明
コマンド・モード	コマンド・モードでは、ed エディターはサブコマンドを認識して実行します。 ed エディターの開始時はコマンド・モードになっています。 **.** (ピリオド) を入力してから Enter を押して、コマンド・モードになっているかどうか確認してください。
テキスト入力モード	テキスト入力モードでは、ed エディターによってテキストをファイル・バッファー内に入力できますが、サブコマンドは認識されません。 **a**サブコマンド、**c** サブコマンド、または **i** サブコマンドを使用して、テキスト入力モードに入ります。テキスト入力モードを終了して、コマンド・モードに戻るには、行の始めに **.** (ピリオド) だけを入力します。テキスト入力モードのままでバッファーに **.** (ピリオド) を入れるには、 1 文字に続けて **.** (ピリオド) を入力します。次に、テキスト入力モードを終了し、 **s** サブコマンドでその文字を除去します。

次に、**ed** エディターの上限を列挙します。

- 各ファイル名には、64 文字まで使用できます。
- 各グローバル・サブコマンド・リストには、 256 文字まで使用できます。
- バッファー・サイズは、128,000 文字です。

注**:** バッファーには元のファイルと編集情報が入ります。

最大行数は、使用できるメモリーの量によって異なります。最大ファイル・サイズは、使用可能な物理データ・ストレージ (ディスクまたはテープ・ドライブ) の量、またはユーザー・メモリーで許可される最大行数によって異なります。

フラグ

項目	説明
-p *String*	エディターのプロンプトを *String* パラメーターに設定します。 *String* のデフォルトは null (プロンプトなし) です。
-s	エディターが、**e** サブコマンド、 **r** サブコマンド、 **w** サブコマンドで表示する文字数を抑止します。また、このフラグは、**e** サブコマンドと、 **q** サブコマンドに関する診断メッセージを抑止し、 **!** サブコマンドの後の ! (感嘆符) プロンプトを抑止します。
-	**-s** フラグと同じ機能をもっています。

パターン・マッチング

ed エディターは、パターン文字列を作成するために正規表現 (RE) として使用できる特殊パターン・マッチング文字の限定フォーマットをサポートしています。これらのパターンをアドレスで使用して行を指定したり、一部のサブコマンドで使用して行の一部を指定することができます。

正規表現

次の正規表現は、単一の文字または照合エレメントと次のように突き合わせます。

項目	説明
Character	*Character* そのものと突き合わせます。通常の文字 (特殊パターン・マッチング記号以外の) を指定することができます。
.	改行文字を除く任意の 1 文字と一致します。

項目	説明
[*String*]	文字列内の任意の 1 文字と突き合わせます。ある種のパターン・マッチング文字は、大括弧で囲むと次のような特別な意味になります。 **^** *String* パラメーターの最初の文字が、 ^ (脱字記号) であれば、 *String* パラメーター内の文字と改行文字を除く任意の文字と突き合わせます。この条件は、^ が文字列 [^*String*] 内で先頭の文字である場合にのみ適用されます。 **-** 現在の照合シーケンスに従って連続する ASCII 文字の範囲を示します。例えば、[a-f] は [abcdef]、[aAbBcCdDeEfF]、[abcdef] と同じで、アクセント記号の付いた a および e 文字を含む場合もあります。照合シーケンスは、文字の等価クラスを定義できます。 負符号が文字列の最初にある場合 ([-*String*])、最初の脱字記号の直後にある場合 ([^-*String*])、または文字列の最後にある場合 [*String*-]、負符号の意味は失われます。 **]** 右大括弧 (]) が文字列の最初にある場合 ([]*String*])、または最初の脱字記号の直後にある場合 ([^]*String*]) は、右大括弧は文字列のターミネーターではなく、文字列の一部として機能します。

パターンの形成

次の規則は、正規表現 (RE) からパターンを形成する方法を示しています。

- 単一の通常文字から構成されている RE は、文字列内のそれと同じ文字と一致することになります。
- RE でそのうしろに * (アスタリスク) が付いている場合には、 * (アスタリスク) の前にある文字が 0 回以上発生する場合と一致することになります。例えば、以下のパターンは、

  ```
  ab*cd
  ```

  次のどの文字列にも一致しますが、

  ```
  acd
  abcd
  abbcd
  abbbcd
  ```

  次の文字列とは一致しません。

  ```
  abd
  ```

  どのような選択肢があっても、左端を基準にして最も長いマッチング文字列が選択されます。例えば、次の文字列の場合、

  ```
  122333444
  ```

  パターン .* は 122333444 と一致し、パターン .*3 は 122333 と一致し、パターン .* 2 は 122 と一致することになります。
- 後に次のものが続く RE は、次のように一致します。

項目	説明
¥{*m*¥}	その RE と一致する *m* 個の文字と完全に 一致します。
¥{*m*,¥}	その RE と一致する *m* 個以上 の文字と一致します。
¥{*m*,*n*¥}	その RE と一致する *m* 個以上 *n* 個以下の任意の数の 文字と一致します。

  数字 *m* と *n* は、0 以上 255 以下の整数でなければなりません。どのような選択肢が存在しても、このパターンは、できる限り多くの該当する文字と一致します。
- 同じ文字シーケンスを含んでいる文字列と一致するパターンに、 RE を組み合わせることができます。例えば、AB¥*CD は文字列 AB*CD と一致し、[A-Za-z]*[0-9]* は、数字の任意の組み合わせ (ない場合も含む) が後に続く、英字の任意の組み合わせ (ない場合も含む) を含むすべての文字列と一致します。

- 文字シーケンス ¥(*Pattern*¥) がマークするサブパターンと一致する文字列は、記号で囲まれていない場合にそのシーケンスが一致する文字列と同じです。
- 文字シーケンス ¥*Number* は、パターン内でサブパターンが既に一致した同じ文字列 (上記の規則を参照) に一致します。*Number* パラメーターのパターンは数字を表します。パターン ¥*Number* は、*Number* パラメーターで指定したサブパターンのオカレンス (左から右へ数える) に一致する文字列と一致します。

  例えば、以下のパターンは、

  ```
  ¥(A¥)¥(B¥)C¥2¥1
  ```

  文字列 ABCBA と一致します。サブパターンはネストできます。

一致するパターンの制限

パターンは、行の最初のセグメント、最後のセグメント、または行全体と一致するように制限することができます。 null パターン // (2 個のスラッシュ) は、直前のパターンを繰り返します。

行の最初のセグメントとの一致

^*Pattern* パラメーターは、行の最初の文字位置で始まる文字列だけに一致します。

行の最後のセグメントとの一致

Pattern$ パラメーターは、行の最後の文字 (改行文字以外) で終わる文字列だけに一致します。

行全体の一致

^*Pattern*$ パラメーターは、行全体が一致する制限付きでパターンを指定します。

行のアドレッシング

ed エディターには、3 種類のアドレスが使われます。行番号アドレス、現在行からの相対アドレス、およびパターン・アドレスです。現在行 (通常は、サブコマンドの影響を受ける最後の行) は、バッファー内の参照点です。

行のアドレッシングでは、次の操作を行うことができます。
- 新しい現在行の指定
- アドレス指定した行の表示
- コマンドを所定の行で実行

アドレスを受け入れないサブコマンドは、アドレスの存在をエラーと見なします。アドレスを受け入れるサブコマンドは、指定されたアドレスまたはデフォルト・アドレスの、どちらかを使用できます。受け入れ範囲を超えるアドレスが指定された場合、コマンドは最後 (一番右) のアドレスを使用します。

大抵の場合、コンマ (,) はアドレスを区切ります (2,8 のように)。セミコロン (;) もアドレスを区切ることができます。アドレス間にセミコロンを入れると、ed エディターは現在行を最初のアドレスに設定し、2 番目のアドレスを算出します (例えば、検索開始行を設定する場合)。アドレスの組で、最初のアドレスは、2 番目のアドレスよりも小さい数値でなければなりません。

行番号と記号アドレスを使用すると、次のタスクを実行することができます。
- 現在行のアドレッシング

- 行番号によるアドレッシング
- 先頭行の前の行のアドレッシング
- 最終行のアドレッシング
- アドレッシングした行の上の行のアドレッシング
- アドレッシングした行の下の行のアドレッシング
- 先頭行から最終行までのアドレッシング
- 現在行から最終行までのアドレッシング
- 行のグループのアドレッシング
- 指定したパターンが入っている行の次行のアドレッシング
- 指定したパターンが入っている行の前行のアドレッシング
- マークした行のアドレッシング

現在行のアドレッシング

. (ピリオド) は、現在行をアドレス指定します。 . (ピリオド) は、ほとんどの ed エディター・サブコマンドのデフォルトであり、指定する必要はありません。

行番号によるアドレッシング

バッファーの指定された行をアドレス指定するには、次のように入力します。

```
Number
```

この場合、*Number* パラメーターは行番号を表します。次に例を示します。

```
2253
```

これにより、行番号 2253 が現在行としてアドレス指定されます。

先頭行の前の行のアドレッシング

バッファーの先頭行の前の行をアドレス指定するには、次のように入力します。

```
0
```

最終行のアドレッシング

バッファーの最終行をアドレス指定するには、次のように入力します。

```
$
```

アドレッシングした行の上の行のアドレッシング

現在行よりも指定した行数だけ上にある行をアドレス指定するには、次のように入力します。

```
-Number
```

この場合、*Number* パラメーターは、アドレス指定したい行が現在行よりも何行上にあるかを示します。次に例を示します。

```
-5
```

現在行の 5 行上にある行が、現在行としてアドレス指定されます。

また、- のみを指定して現在行の真上にある行をアドレス指定することもできます。- には累積効果があります。例えば、アドレス - (2 つのハイフン) を指定すると、現在行の 2 行上の行がアドレス指定されます。

アドレッシングした行の下の行のアドレッシング

現在行から指定した行数だけ下にある行をアドレス指定するには、次のように入力します。

```
+Number
```

この場合、*Number* パラメーターは、アドレス指定したい行が現在行から何行下にあるかを示します。+ (正符号) はオプションです。次に例を示します。

```
+11
```

現在行から 11 行下にある行が、現在行としてアドレス指定されます。

また、+ のみを指定して、現在行の真下にある行をアドレス指定することもできます。+ には累積効果があります。例えば、アドレス + + (2 つの正符号) を指定すると、現在行から 2 行下にある行がアドレス指定されます。

先頭行から最終行までのアドレッシング

先頭行から最終行までをアドレス指定するには、次のように入力します。

```
,
```

, (コンマ) は、アドレスの組 1,$ (先頭行から最終行まで) を表します。先頭行が現在行になります。

現在行から最終行までのアドレッシング

現在行から最終行までをアドレス指定するには、次のように入力します。

```
;
```

; (セミコロン) は、アドレスの組 .,$ (現在行から最終行まで) を表します。

行のグループのアドレッシング

行のグループをアドレス指定するには、次のように入力します。

```
FirstAddress,LastAddress
```

この場合、*FirstAddress* パラメーターは、アドレス指定したいグループ内の先頭行の行番号 (または記号アドレス) です。*LastAddress* パラメーターは、グループ内の最終行の行番号 (または記号アドレス) です。グループ内の先頭行が現在行になります。次に例を示します。

```
3421,4456
```

これにより、行 3421 から 4456 までがアドレス指定されます。行 3421 が現在行になります。

指定したパターンが入っている行の次行のアドレッシング

一致する文字列が入っている行の次行をアドレス指定するには、次のように入力します。

```
/Pattern/
```

この場合、*Pattern* パラメーターは、文字列または正規表現です。検索は、現在行の次の行から始まり、パターンに一致する行が見つかると停止します。必要であれば、検索処理はバッファーの終わりまで移動し、バッファーの先頭に折り返し、一致する行が見つかるまで継続されるか、または現在行に戻ります。次に例を示します。

```
/Austin, Texas/
```

これにより、Austin, Texas が入っている行の次の行が現在行としてアドレス指定されます。

指定したパターンが入っている行の前行のアドレッシング

パターンに一致する値が入っている前行をアドレス指定するには、次のように入力します。

```
?Pattern?
```

この場合、*Pattern* パラメーターは、文字列または正規表現です。?*Pattern*? 構造を指定すると、 /*Pattern*/ と同様に、バッファー全体を検索することができますが、検索は逆方向に行われます。次に例を示します。

```
?Austin, Texas?
```

これにより、Austin, Texas が入っている行の前の行が、現在行としてアドレス指定されます。

マークした行のアドレッシング

k サブコマンドでマークした行をアドレス指定するには、次のように入力します。

```
'x
```

この場合、*x* パラメーターは *a* から *z* までの小文字です。次に例を示します。

```
'c
```

これにより、**k** サブコマンドで c とマークした行がアドレス指定されます。

サブコマンド

ed エディター・サブコマンドを使用すると、次の操作を行うことができます。

- ファイルの編集
- ファイルの操作
- その他の機能の実行
  - プロンプト文字列を変更する
  - システム・コマンドを入力する
  - ed エディターを終了する
  - ヘルプを要求する

大抵の場合、1 行に入力できる ed エディター・サブコマンドは 1 つだけです。ただし、 **l** (list) サブコマンドと、 **p** (print) サブコマンドは、 **e** (edit)、 **E** (Edit)、 **f** (file)、 **q** (quit)、 **Q** (Quit)、 **r** (read)、 **w** (write)、 **!** (オペレーティング・システム・コマンド) を除く、いずれのサブコマンドにも追加することができます。

e、**f**、**r**、および **w** サブコマンドでは、ファイル名をパラメーターとして指定できます。 ed エディターは、サブコマンドに使用された最後のファイル名をデフォルト・ファイル名として格納します。ファイル名を指定せずに **e**、**E**、 **f**、**r**、**w** のいずれかのサブコマンドを次に使用すると、デフォルトのファイル名が使用されます。

ed エディターは、2 つのメッセージ、つまり、? (疑問符) または ?File、のいずれかでエラー状態に応答します。ed エディターは、割り込み信号 (Ctrl-C キー・シーケンス) を受信すると、?を表示してコマンド・モードに戻ります。 ed エディターは、ファイルを読み取ると、ASCII null 文字および最後の改行文字の後のすべての文字を破棄します。

ファイルの編集

ed エディター・サブコマンドを使用すると、以下のタスクを実行できます。

- テキストの追加
- テキストの変更
- テキストのコピー
- テキストの削除
- テキストの表示
- 行の結合と分割
- 一括変更
- テキストのマーク付け
- テキストの移動
- テキストの保存
- テキストの検索
- テキストの置換
- テキスト変更の取り消し

注**:** 次の ed サブコマンドの説明では、デフォルトのアドレスは括弧で囲んで示しています。この括弧は入力しないでください。 (ピリオド) のアドレスは、現在行を示します。他の空の行の 1 桁目にある . (ピリオド) は、コマンド・モードに戻るための信号です。

テキストの追加

項目	説明
(.)**a** [**l**] [**n**] [**p**] *Text*.	**a** (append) サブコマンドは、アドレス指定された行の後 のバッファーにテキストを追加します。 **a** サブコマンドは、テキストが最後に挿入された行を現在行に設定します。行が挿入されなかった場合は、アドレス指定された行が現在行になります。アドレス 0 を指定すると、バッファーの先頭にテキストが追加されます。 追加したテキストを表示する場合は、**l** (list)、**n** (number)、または **p** (print) オプション・サブコマンドを入力します。 テキストを入力するときは、各行の終わりに Enter キーを押してください。各行の終わりに Enter を押さないと、行に文字を入力した後に、ed エディターによってカーソルが自動的に次の行に移動されます。 ed エディターは、画面に占める行数には関係なく、Enter を押す前に入力されたものをすべて 1 行として処理します。 テキストをすべて入力した後に改行の先頭で . (ピリオド) を入力します。

項目	説明
(.)**i** **[l]** **[n]** **[p]***Text*.	**i** (insert) サブコマンドは、アドレス指定された行の前 にテキストを挿入し、最後にテキストが挿入された行を現在行に設定します。挿入がない場合、**i** サブコマンドはアドレス指定された行を現在行に設定します。このサブコマンドには 0 アドレスは使用できません。 挿入したテキストを表示したい場合は、 **l** (list)、**n** (number)、 **p** (print) オプション・サブコマンドのいずれかを入力します。 テキストを入力するときは、各行の終わりに Enter キーを押してください。各行の終わりに Enter を押さないと、行に文字を入力した後に、ed エディターによってカーソルが自動的に次の行に移動されます。 ed エディターは、画面に占める行数には関係なく、Enter を押す前に入力されたものをすべて 1 行として処理します。 テキストをすべて入力した後に改行の先頭で . (ピリオド) を入力します。 **注: i** サブコマンドと、 **a** サブコマンドとの違いは、テキストの挿入位置のみです。

ed エディター・サブコマンドを使用すると、さまざまな場所にテキストを挿入できます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行の後にテキストを追加する
- 現在行の前にテキストを追加する
- アドレス指定した行の後にテキストを追加する
- アドレス指定した行の前にテキストを追加する
- 検索パターンが入っている行の後にテキストを追加する
- 検索パターンが入っている行の前にテキストを追加する
- 検索パターンが入っていない行の後にテキストを追加する
- 検索パターンが入っていない行の前にテキストを追加する

現在行の後にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   a[l][n][p]
   ```

   この場合、**l**、 **n**、および **p** は、追加されたテキストを表示するオプション・サブコマンドです。
2. テキストを入力して Enter を押します。
3. **.** (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

現在行の前にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   i[l][n][p]
   ```

   この場合、**l**、 **n**、および **p** は、追加されたテキストを表示するオプション・サブコマンドです。
2. テキストを入力して Enter を押します。
3. **.** (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

アドレス指定した行の後にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   Addressa[l][n][p]
   ```

   この場合、*Address* パラメーターは、挿入したテキストの前の行の行番号です。 **l**、**n**、および **p** オプション・サブコマンドは、追加されたテキストを表示します。

2. テキストを入力して Enter を押します。
3. . (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

アドレス指定した行の前にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   Addressi[l][n][p]
   ```

   この場合、*Address* パラメーターは、挿入したテキストの後の行の行番号です。 **l**、**n**、および **p** オプション・サブコマンドは、追加されたテキストを表示します。
2. テキストを入力して Enter を押します。
3. . (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

検索パターンが入っている行の後にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   [Address]g/Pattern/a[l][n][p]
   ```

   この場合、*Address* は、*Pattern* パラメーターで指定したパターンが検索される行の範囲を指定するオプション・パラメーターです。 *Pattern* パラメーターは、文字列または正規表現です。*Address* パラメーターを省略すると、 ed エディターはパターンが入っている行を探してファイル全体を検索します。 **l**、**n**、および **p** オプション・サブコマンドは、追加されたテキストを表示します。
2. 円記号を入力します。

   ```
   ¥
   ```
3. テキストを入力します。追加したテキスト内で新しい行を始めるには、次のように円記号を入力し、

   ```
   ¥
   ```

   次に Enter を押します。コマンド内で指定したパターンが入っている各行の後に、入力したテキストが追加されます。
4. Enter を押すと、コマンド・モードに戻ります。

検索パターンが入っている行の前にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   [Address]g/Pattern/i[l][n][p]
   ```

   この場合、*Address* は、*Pattern* パラメーターで指定したパターンが検索される行の範囲を指定するオプション・パラメーターです。 *Pattern* パラメーターは、文字列または正規表現です。*Address* パラメーターを省略すると、 ed エディターはパターンが入っている行を探してファイル全体を検索します。 **l**、**n**、および **p** オプション・サブコマンドは、追加されたテキストを表示します。
2. 円記号を入力します。

   ```
   ¥
   ```
3. テキストを入力します。追加したテキスト内で新しい行を始めるには、次のように円記号を入力し、

   ```
   ¥
   ```

   次に Enter を押します。コマンド内で指定したパターンが入っている各行の前に、入力したテキストが追加されます。
4. Enter を押すと、コマンド・モードに戻ります。

検索パターンが入っていない行の後にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   [Address]g/Pattern/a[l][n][p]
   ```

   この場合、*Address* は、 *Pattern* パラメーターで指定したパターンが入っていない行を探すため検索される行の、範囲を指定するオプション・パラメーターです。 *Pattern* パラメーターは、文字列または正規表現です。*Address* を省略すると、ed エディターはパターンが入っていない行を探すためファイル全体を検索します。**l**、**n**、および **p** オプション・サブコマンドは、追加されたテキストを表示します。
2. 円記号を入力します。

   ```
   ¥
   ```
3. テキストを入力します。追加したテキスト内で新しい行を始めるには、次のように円記号を入力し、

   ```
   ¥
   ```

   次に Enter を押します。コマンド内で指定したパターンが入っていない各行の後に、入力したテキストが追加されます。
4. Enter を押すと、コマンド・モードに戻ります。

検索パターンが入っていない行の前にテキストを追加する

1. 次のサブコマンドを入力します。

   ```
   [Address]g/Pattern/i[l][n][p]
   ```

   この場合、*Address* は、 *Pattern* パラメーターで指定したパターンが入っていない行を探すため検索される行の、範囲を指定するオプション・パラメーターです。 *Pattern* パラメーターは、文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはパターンが入っていない行を探してファイル全体を検索します。 **l**、**n**、および **p** オプション・サブコマンドは、追加されたテキストを表示します。
2. 円記号を入力します。

   ```
   ¥
   ```
3. テキストを入力します。追加したテキスト内で新しい行を始めるには、次のように円記号を入力し、

   ```
   ¥
   ```

   次に Enter を押します。コマンド内で指定したパターンが入っていない各行の前に、入力したテキストが追加されます。
4. Enter を押すと、コマンド・モードに戻ります。

テキストの変更

項目	説明
(.,.)**c** **[l]** **[n]** **[p]***Text*.	**c** (change) サブコマンドは、置換対象としてアドレス指定された行を削除して、新しく入力された行と置き換えます。**c** サブコマンドは、最後に入力された新しい行を現在行に設定します。あるいは入力されていない場合は、削除されていない最初の行が現在行になります。 挿入したテキストを表示したい場合は、 **l** (list)、**n** (number)、 **p** (print) オプション・サブコマンドのいずれかを入力します。 新しいテキストを入力し、各行の終わりで Enter キーを押します。新しいテキストをすべて入力したら、1 行ごとに . (ピリオド) を入力してください。

ed エディターを使用すると、さまざまな方法でテキストを変更することができます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行のテキストを変更する
- 1 行または行範囲のテキストを変更する
- 指定したパターンが入っている行のテキストを変更する
- 指定したパターンが入っていない行のテキストを変更する

現在行のテキストを変更する

1. 次のサブコマンドを入力します。

   ```
   c[l][n][p]
   ```

   この場合、**l**、 **n**、および **p** は、変更されたテキストを表示するオプション・サブコマンドです。
2. テキストを入力して Enter を押します。
3. **.** (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

1 行または行範囲のテキストを変更する

1. 次のサブコマンドを入力します。

   ```
   Addressc[l][n][p]
   ```

   この場合、*Address* パラメーターは、変更したい行または行範囲のアドレスです。 **l**、 **n**、および **p** オプション・サブコマンドは、変更されたテキストを表示します。
2. テキストを入力して Enter を押します。
3. **.** (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

指定したパターンが入っている行のテキストを変更する

1. 次のサブコマンドを入力します。

   ```
   Addressg/Pattern/c[l][n][p]
   ```

   この場合、*Address* パラメーターは、 *Pattern* パラメーターで指定したパターンを検索したい、行範囲のアドレスです。 **l**、 **n**、および **p** オプション・サブコマンドは、変更されたテキストを表示します。
2. 円記号を入力します。

   ```
   ¥
   ```
3. 新しいテキストを入力します。新しいテキスト内で新しい行を始めるには、次のように円記号を入力し、

   ```
   ¥
   ```

   次に Enter を押します。
4. コマンド・モードに戻るには、Enter キーをもう一度押し、. (ピリオド) を入力してから、さらにもう一度 Enter を押します。

指定したパターンが入っていない行のテキストを変更する

1. 次のサブコマンドを入力します。

   ```
   Addressv/Pattern/c[l][n][p]
   ```

この場合、*Address* パラメーターは、 *Pattern* パラメーターで指定したパターンを検索したい、行範囲のアドレスです。 **l**、 **n**、および **p** オプション・サブコマンドは、変更されたテキストを表示します。

2. 円記号を入力します。

   ¥

3. 新しいテキストを入力します。新しいテキスト内で新しい行を始めるには、次のように円記号を入力し、

   ¥

   次に Enter を押します。

4. コマンド・モードに戻るには、Enter キーをもう一度押し、. (ピリオド) を入力してから、さらにもう一度 Enter を押します。

テキストのコピー

項目	説明
(.,.)(.,.) **t***Address* **[p] [l] [n]**	**t** (transfer) サブコマンドは、アドレス指定した行のコピーを、*Address* パラメーターで指定した行の後に挿入します。 **t** サブコマンドはアドレス 0 を受け入れてバッファーの先頭に行を挿入します。 **t** サブコマンドは、コピーされた最後の行を現在行に設定します。 転送されたテキストを表示したい場合は、**l** (list)、**n** (number)、または **p** (print) のオプション・サブコマンドを入力します。

1 行または行範囲をコピーすると、指定した行は元の位置に残り、コピーが新しい位置に挿入されます。アドレスまたはパターンを指定して、コピーする行を選択できます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行をコピーする
- アドレスで指定した行をコピーする
- 指定したパターンが入っている行をコピーする
- 指定したパターンが入っていない行をコピーする

現在行をコピーする

1. 次のサブコマンドを入力します。

   ```
   tAddress[l][n][p]
   ```

   この場合、*Address* パラメーターは、現在行のコピーを後ろに挿入したい行の行番号または記号アドレスです。**l**、 **n**、および **p** オプション・サブコマンドは、コピーされた行を表示します。

2. テキストを入力して Enter を押します。
3. **.** (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

アドレスで指定した行をコピーする

1. 次のサブコマンドを入力します。

   ```
   LineNumbertDestinationAddress[l][n][p]
   ```

   この場合、*LineNumber* パラメーターは、コピーしたい行のアドレスです。また、*DestinationAddress* パラメーターは、コピーを後ろに挿入したい行です。 **l**、 **n**、および **p** オプション・サブコマンドは、コピーされた行を表示します。

2. テキストを入力して Enter を押します。
3. . (ピリオド) を入力してからもう一度 Enter を押すと、コマンド・モードに戻ります。

指定したパターンが入っている行をコピーする

次のサブコマンドを入力します。

```
[Address]g/Pattern/t[DestinationAddress][l][n][p]
```

この場合、*Address* は指定したパターンを含む行を探すために検索する行の範囲を指定するオプション・パラメーターで、*Pattern* パラメーターは検索対象のテキスト、*DestinationAddress* はコピーしたテキストの前の行を表すオプション・パラメーターです。 **l**、 **n**、および **p** オプション・サブコマンドは、コピーされた行を表示します。

Address パラメーターを省略すると、 ed エディターはパターンが入っている行を探してファイル全体を検索します。*DestinationAddress* パラメーターを省略すると、現在行の後にコピーされたテキストが挿入されます。

指定したパターンが入っていない行をコピーする

次のサブコマンドを入力します。

```
[Address]v/Pattern/t[DestinationAddress][l][n][p]
```

この場合、*Address* は指定したパターンを含まない行を探すために検索する行の範囲を指定するオプション・パラメーターで、*Pattern* パラメーターはテキスト、*DestinationAddress* はコピーしたテキストの前の行を表すオプション・パラメーターです。 **l**、 **n**、および **p** オプション・サブコマンドは、コピーされた行を表示します。

Address パラメーターを省略すると、ed エディターはパターンが入っていない行を探してファイル全体を検索します。 *DestinationAddress* パラメーターを省略すると、現在行の後にコピーされたテキストが挿入されます。

テキストの削除

項目	説明
(.,.)(.,.) **d** [**l**] [**n**] [**p**]	**d** (delete) サブコマンドは、アドレス指定した行をバッファーから除去します。最後に削除された行の後の行が現在行になります。削除された行がもともとバッファーの終わりにあった場合は、新しい最終行が現在行になります。 削除した結果を表示したい場合は、**l** (list)、**n** (number)、**p** (print) のオプション・サブコマンドのいずれかを入力します。

ed エディターを使用すると、いくつかの方法でテキストを削除することができます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行を削除する
- 行または行範囲を削除する
- 指定したパターンが入っている行または行範囲を削除する
- 指定したパターンが入っていない行または行範囲を削除する
- 現在行からテキストを削除する
- 選択した行内のテキストを削除する
- アドレス指定した行からテキストを削除する

- 指定したパターンが入っている行からテキストを削除する
- 指定したパターンが入っている行から別のパターンを削除する
- 指定したパターンが入っていない行から別のパターンを削除する

現在行を削除する

次のサブコマンドを入力します。

```
d[l][n][p]
```

この場合、**l**、 **n**、および **p** は、削除された行を表示するオプション・サブコマンドです。

行または行範囲を削除する

次のサブコマンドを入力します。

```
Addressd[l][n][p]
```

この場合、*Address* パラメーターは、削除する行の行番号または記号アドレスです。また、 **l**、 **n**、および **p** は、削除された行 (1 行または複数行) を表示するオプション・サブコマンドです。

指定したパターンが入っている行または行範囲を削除する

次のサブコマンドを入力します。

```
[Address]g/Pattern/d[l][n][p]
```

この場合、*Address* は、検索したい行の行番号または記号アドレスを指定するオプション・パラメーターです。 *Pattern* パラメーターは、検索したいテキストを表す文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターは指定したパターンが入っている行を探してファイル全体を検索します。 **l**、 **n**、および **p** オプション・サブコマンドは、削除された行を表示します。

指定したパターンが入っていない行または行範囲を削除する

次のサブコマンドを入力します。

```
[Address]v/Pattern/d[l][n][p]
```

この場合、*Address* は、検索したい行の行番号または記号アドレスを指定するオプション・パラメーターです。 *Pattern* パラメーターは、検索したいテキストを表す文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターは指定されたパターンが入っていない行を探してファイル全体を検索します。 **l**、 **n**、および **p** オプション・サブコマンドは、削除された行を表示します。

現在行からテキストを削除する

1. 次のサブコマンドを入力します。

   ```
   s/Pattern
   ```

   この場合、*Pattern* パラメーターは、削除するテキストを表す文字列または正規表現です。
2. パターンの最初のインスタンス を行から削除するには、次のように入力します。

   ```
   //
   ```

   または

   パターンのすべてのインスタンス を行から削除するには、次のように入力します。

   ```
   //g
   ```

3. 削除した結果を表示したい場合は、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l

   n

   p
   ```
4. Enter を押します。

選択した行内のテキストを削除する

1. 選択する行範囲のアドレスを入力します (すべての行を選択する場合は、このステップを省きます)。
2. ステップ 4 の *Pattern* パラメーターで、指定する行を選択するには、次のように入力します。

   ```
   g
   ```

   または

   ステップ 4 の *Pattern* パラメーターで指定しない 行を選択するには、次のように入力します。

   ```
   v
   ```
3. 検索したいテキストを入力するには、次のサブコマンドを実行します。

   ```
   /Pattern/s
   ```

   この場合、*Pattern* パラメーターは検索したいテキストです。
4. 次のいずれかのコマンドを入力して、希望する削除を行います。

   選択した各行内で、*Pattern* パラメーターの最初のインスタンスを削除するには、次のように入力します。

   ```
   ///
   ```

   選択した各行内で、*Pattern* パラメーターのすべてのインスタンスを削除するには、次のように入力します。

   ```
   ///g
   ```

   選択された各行内で、*Pattern* パラメーターのオカレンスを指定した数だけ最初の方から削除するには、次のように入力します (この場合、*Number* パラメーターは整数です)。

   ```
   ///Number
   ```

   Pattern パラメーターで選択した各行内で、*OtherPattern* パラメーターで指定した最初の文字列を削除するには、次のように入力します (この場合、*OtherPattern* パラメーターは、検索したいパターンです)。

   ```
   /OtherPattern//
   ```

   Pattern パラメーターで指定した各行内で、*OtherPattern* パラメーターのすべてのインスタンスを削除するには、次のように入力します。

   ```
   /OtherPattern//g
   ```

   Pattern パラメーターで指定した各行内で、*OtherPattern* のオカレンスを指定した数だけ最初の方から削除するには、次のように入力します (この場合、*Number* パラメーターは整数です)。

   ```
   /OtherPattern//Number
   ```
5. 削除した結果を表示したい場合は、次のいずれかのオプション・サブコマンドを入力します。

l

n

p

6. Enter を押します。

例えば、ある行範囲 からパターンのすべてのインスタンスを削除するには、次のように入力します。

```
38,$g/tmp/s/gn
```

上記の例では、行 38 から最終行まで (38,$) のすべての行で tmp 文字列が検索され、この行範囲内の各行で tmp 文字列のすべてのインスタンス (/g) が削除されます。次に、テキストが削除された行とその行番号 (n) が表示されます。

パターンが入っているすべての行 から、そのパターンのすべてのインスタンスを削除するには、次のように入力します。

```
g/rem/s///gl
```

上記の例では、 rem 文字列が入っているすべての行 (g) を探して、ファイル全体が検索されます (アドレス・パラメーターを省略したため)。見つかった各行から rem 文字列のすべてのインスタンス (///g) が削除され、テキストが削除された行が、各行の印刷されない文字を含めて、表示されます (l)。

アドレス指定した行からテキストを削除する

1. 次のサブコマンドを入力します。

   ```
   Addresss/Pattern
   ```

   注**:** *Address* パラメーターの後に **s** サブコマンドを指定します。この場合、*Address* パラメーターは、パターンを削除する行の行番号、行番号の範囲、または記号アドレスです。*Pattern* パラメーターは、削除するテキストを表す文字列または正規表現です。
2. 各行からパターンの最初のインスタンス を削除するには、次のように入力します。

   ```
   //
   ```

   または

   各行からパターンのすべてのインスタンス を削除するには、次のように入力します。

   ```
   //g
   ```
3. 削除した結果を表示したい場合は、次のいずれかのオプション・サブコマンドを入力します。

   l

   n

   p
4. Enter を押します。

指定したパターンが入っている行からテキストを削除する

1. 次のサブコマンドを入力します。

   ```
   [Address]g/Pattern/s
   ```

この場合、*Address* は、指定したパターンが入っている行の行番号、行番号の範囲、または記号アドレスを指定するオプション・パラメーターです。 *Pattern* パラメーターは、検索して削除するテキストを表す文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはパターンを探してファイル内のすべての行を検索します。

2. パターンが入っている各行から、パターンの最初のインスタンス を削除するには、次のように入力します。

   ```
   ///
   ```

   または

   パターンが入っている各行から、パターンのすべてのインスタンス を削除するには、次のように入力します。

   ```
   ///g
   ```

3. 削除した結果を表示したい場合は、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```

4. Enter を押します。

指定したパターンが入っている行から別のパターンを削除する

1. 次のサブコマンドを入力します。

   ```
   [Address]g/SearchPattern/s
   ```

   この場合、*Address* は、指定したパターンが入っている行の行番号、行範囲、または記号アドレスを指定するオプション・パラメーターです。 *SearchPattern* パラメーターは、変更したい行に入っているテキストを表す文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターは指定したパターンを探してファイル内のすべての行を検索します。

2. 削除するテキストを指定するには、次のように入力します。

   ```
   /DeletePattern/
   ```

3. 各行からパターンの最初のインスタンス を削除するには、次のように入力します。

   ```
   /
   ```

   または

   各行からパターンのすべてのインスタンス を削除するには、次のように入力します。

   ```
   /g
   ```

   注**:** サブコマンドの文字列全体は、次のようになります。

   ```
   [Address]g/SearchPattern/s/DeletePattern//[g]
   ```

4. 削除した結果を表示したい場合は、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l
   ```

n

p

5. Enter を押します。

例えば、指定したパターンが入っている行から、別のパターンの最初のインスタンスを削除するには、次のように入力します。

```
1,.g/rem/s/tmp//l
```

上記の例では、先頭行から現在行 (1,.) まで、 rem 文字列が入ったすべての行 (g) を検索しています。検出した各行から tmp 文字列の最初のインスタンス (/) が削除され、テキストが削除された行が、各行の印刷されない文字を含めて、表示されます (l)。

指定したパターンが入っていない行から別のパターンを削除する

1. 次のサブコマンドを入力します。

   ```
   [Address]v/SearchPattern/s
   ```

   この場合、*Address* は、指定したパターンが入っている行の行番号、行範囲、または記号アドレスを指定するオプション・パラメーターです。 *SearchPattern* パラメーターは、変更したい行に入っていないテキストを表す文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターは指定したパターンを探してファイル内のすべての行を検索します。
2. 削除するテキストを指定するには、次のように入力します。

   ```
   /DeletePattern/
   ```
3. パターンの最初のインスタンス を削除するには、次のように入力します。

   ```
   /
   ```

   または

   各行からパターンのすべてのインスタンス を削除するには、次のように入力します。

   ```
   /g
   ```

   注**:** サブコマンドの文字列全体は、次のようになります。

   ```
   [Address]v/SearchPattern/s/DeletePattern//[g]
   ```
4. 削除した結果を表示したい場合は、次のいずれかのオプション・サブコマンドを入力します。

   l

   n

   p
5. Enter を押します。

例えば、指定したパターンが入っていない行から、別のパターンの最初のインスタンスを削除するには、次のように入力します。

```
1,.v/rem/s/tmp//l
```

上記の例では、先頭行から現在行 (1,.) まで、 rem 文字列が入っていないすべての行 (v) を検索しています。検出した各行から tmp 文字列の最初のインスタンス (/) が削除され、テキストが削除された行が、各行の印刷されない文字を含めて、表示されます (l)。

テキストの表示

項目	説明
(.,.)**l**	**l** (list) サブコマンドは、アドレス指定した行を、視覚的に明確なフォーマットで標準出力に書き出し、文字 ¥¥¥、¥¥a、 ¥¥b、¥¥f、 ¥¥r、¥¥t、および ¥¥v を、それに対応するエスケープ・シーケンスに書き出します。**l** サブコマンドは、印刷可能でない文字を 3 桁の 8 進数として書き出し、文字の各バイトの先頭に ¥ (円記号) を付けます (最上位のバイトを先頭にします)。 **l** サブコマンドは長い行を折り返します。¥ (円記号) / (スラッシュ) 改行文字シーケンスを書き込むと、折り返し点を指定することができます。折り返しは 72 桁目で発生します。 $ (ドル記号) は、各行の終わりを示します。 **l** サブコマンドは、**e**、**E**、**f**、**q**、**Q**、**r**、**w**、または **!** サブコマンドを除き、どの ed エディター・サブコマンドにも追加できます。現在行番号は、最後に書き込まれた行のアドレスに設定されます。
(.,.)**n**	**n** (number) サブコマンドは、アドレス指定した行を表示しますが、各行の前に行番号とタブ文字 (ブランク・スペースとして表示) が付きます。 **n** は、最後に表示された行を現在行に設定します。 **n** サブコマンドは、 **e**、 **f**、 **r**、 **w** 以外の、任意の ed エディター・サブコマンドに付加できます。例えば **dn** サブコマンドは、現在行を削除し、新しい現在行とその行番号を表示します。
(.,.)**p**	**p** (print) サブコマンドは、アドレス指定した行を表示し、最後に表示した行を現在行に設定します。 **p** サブコマンドは、 **e**、 **f**、 **r**、 **w** 以外の、任意の ed エディター・サブコマンドに付加できます。例えば **dp** サブコマンドは、現在行を削除し、新しい現在行を表示します。
(.)**=**	アドレスを指定しない場合、**=** (等号) サブコマンドは現在行の行番号を表示します。アドレス $ を前に付けると、**=** サブコマンドはバッファー内の最終行の番号を表示します。 **=** サブコマンドは、現在行を変更しないため、 **g** サブコマンド、または **v** サブコマンドには付加できません。

指定したパターンが入っている行、または入っていない行を検索するときには、検索したい行番号の範囲を指定することができます。 ed エディター・ファイル内では、さまざまな方法で 1 行または行範囲を選択して表示できます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- アドレス指定した行または行範囲を表示する
- アドレス指定した行または行範囲を非印刷文字を含めて表示する
- アドレス指定した行または行範囲を、行番号を含めて表示する
- 検索パターンが入っている行を表示する
- 検索パターンが入っている行を、非印刷文字を含めて表示する
- 検索パターンが入っている行を、行番号を含めて表示する
- 検索パターンが入っていない行を表示する
- 検索パターンが入っていない行を、非印刷文字を含めて表示する
- 検索パターンが入っていない行を、行番号を含めて表示する

アドレス指定した行または行範囲を表示する

次のサブコマンドを入力します。

```
Addressp
```

この場合、*Address* パラメーターは、表示したい行の行番号または記号アドレスです。

アドレス指定した行または行範囲が画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

アドレス指定した行または行範囲を非印刷文字を含めて表示する

次のサブコマンドを入力します。

```
Addressl
```

この場合、*Address* パラメーターは、表示したい行の行番号または記号アドレスです。

アドレス指定した行または行範囲が非印刷文字も含めて画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

アドレス指定した行または行範囲を、行番号を含めて表示する

次のサブコマンドを入力します。

```
Addressn
```

この場合、*Address* パラメーターは、表示したい行の行番号または記号アドレスです。

アドレス指定した行または行範囲が画面に表示されます。各行の行番号は、行の隣に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

検索パターンが入っている行を表示する

次のサブコマンドを入力します。

```
Addressg/Pattern/p
```

この場合、*Address* パラメーターは行の範囲で、 *Pattern* パラメーターは検索したい文字列または正規表現です。

指定したパターンが入っている行または行範囲が、画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

検索パターンが入っている行を、非印刷文字を含めて表示する

次のサブコマンドを入力します。

```
[Address]g/Pattern/l
```

この場合、*Address* は、行の範囲を指定するオプション・パラメーターで、 *Pattern* パラメーターは検索したい文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはファイル全体を検索します。

指定したパターンが入っている行または行範囲が、画面に表示されます。非印刷文字も画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

検索パターンが入っている行を、行番号を含めて表示する

次のサブコマンドを入力します。

```
[Address]g/Pattern/n
```

この場合、*Address* は、行の範囲を指定するオプション・パラメーターで、 *Pattern* パラメーターは検索したい文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはファイル全体を検索します。

指定したパターンが入っている行または行範囲が、画面に表示されます。各行の行番号は、行の隣に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

検索パターンが入っていない行を表示する

次のサブコマンドを入力します。

```
[Address]v/Pattern/p
```

この場合、*Address* は、行の範囲を指定するオプション・パラメーターで、 *Pattern* パラメーターは検索したい文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはファイル全体を検索します。

指定したパターンが入っていない行または行範囲が、画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

検索パターンが入っていない行を、非印刷文字を含めて表示する

次のサブコマンドを入力します。

```
[Address]v/Pattern/l
```

この場合、*Address* は、行の範囲を指定するオプション・パラメーターで、 *Pattern* パラメーターは検索したい文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはファイル全体を検索します。

指定したパターンが入っていない行または行範囲が、非印刷文字を含めて画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

検索パターンが入っていない行を、行番号を含めて表示する

次のサブコマンドを入力します。

```
[Address]v/Pattern/n
```

この場合、*Address* は、行の範囲を指定するオプション・パラメーターで、 *Pattern* パラメーターは検索したい文字列または正規表現です。*Address* パラメーターを省略すると、ed エディターはファイル全体を検索します。

指定したパターンが入っていない行または行範囲が、行番号と一緒に画面に表示されます。行範囲が長すぎて画面に表示しきれない場合、ed エディターは、アドレス指定した先頭行から画面に入るだけの行数を表示します。

行の結合と分割

項目	説明
(.,.+1)**j** [**l**] [**n**] [**p**]	**j** (join) サブコマンドは、連続する行の間にある改行文字を除去して各行を結合します。アドレスを 1 つしか指定しないと、**j** サブコマンドは何も行いません。 結合された行を表示したい場合は、**l** (list)、 **n** (number)、または **p** (print) サブコマンドを入力します。これらのサブコマンドはオプションです。

ed エディターを使用すると、いくつかの方法で行を結合または分割できます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行と次行を結合する
- アドレス指定した行を結合する
- 現在行を分割する

- アドレス指定した行を分割する

現在行と次行を結合する

次のサブコマンドを入力します。

```
j[l][n][p]
```

この場合、**l**、 **n**、および **p** は、結合された行を表示するオプション・サブコマンドです。

アドレス指定した行を結合する

次のサブコマンドを入力します。

```
Addressj[l][n][p]
```

この場合、*Address* パラメーターは、1 つの行を形成する連続した行の集合で、 **l**、 **n**、および **p** は、結合された行を表示するオプション・サブコマンドです。

現在行を分割する

1. 指定したパターンの後の現在行を分割するには、次のサブコマンドを入力します。

   ```
   s/Pattern/Pattern¥
   ```

   この場合、*Pattern* パラメーターは、分割したい行の前にある文字列です。

   注**:** *Pattern* パラメーターで入力する 2 つの文字列は完全に同じでなければなりません。
2. Enter を押します。
3. 次のように円記号を入力します。

   /
4. 分割された行を表示するには、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l

   n

   p
   ```
5. Enter を押します。

アドレス指定した行を分割する

1. 指定したパターンの後のアドレス指定した行を分割するには、次のサブコマンドを入力します。

   ```
   Addresss/Pattern/Pattern¥
   ```

   この場合、*Address* パラメーターは分割したい行のアドレスで、*Pattern* パラメーターは分割したい行の前にある文字列です。

   注**:** *Pattern* パラメーターで入力する 2 つの文字列は完全に同じでなければなりません。
2. Enter を押します。
3. 次のように円記号を入力します。

   /
4. 分割された行を表示するには、次のいずれかのオプション・サブコマンドを入力します。

l

n

p

5. Enter を押します。

一括変更

項目	説明
(1,$) **g/***Pattern***/***SubcommandList* **[l] [n] [p]**	**g** (global) サブコマンドは、まず、*Pattern* パラメーターと一致するすべての行にマークを付けます。パターンは固定文字列または正規表現のいずれかです。次にこのサブコマンドはマークされた各行を現在行に設定して、*SubcommandList* パラメーターを実行します。単一のサブコマンドまたは一連のサブコマンドの最初のサブコマンドは、**g** サブコマンドと同じ行に入力します。それに続くサブコマンドはそれぞれ別の行に入力します。最終行を除いて、これらの行の末尾には ¥ (円記号) を付けてください。 *SubcommandList* パラメーターには、 **a**、 **i**、および **c** サブコマンドと、その入力を組み込むことができます。通常、*SubcommandList* パラメーターの最後のコマンドは入力モードを終了させる **.** (ピリオド) であれば、**.** (ピリオド) は省略できます。*SubcommandList* パラメーターがない場合は現在行が表示されます。 *SubcommandList* パラメーターには、**g**、 **G**、 **v**、または **V** サブコマンドを組み込むことはできません。 変更した結果を表示したい場合は、**l** (list)、**n** (number)、または **p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。 **注: g** サブコマンドは、 **v** サブコマンドと同様、パターンに一致するものを含まないすべての行に対して、 *SubcommandList* パラメーターを実行します。
(1,$) **G/***Pattern***/ [l] [n] [p]**	対話型 **G** (Global) サブコマンドは、*Pattern* パラメーターに一致するすべての行にマークを付け、マークが付けられた最初の行を表示し、現在行をその行に設定して、サブコマンドを待ちます。パターンは固定文字列または正規表現のいずれかです。 **G** サブコマンドは、**a**、 **i**、 **c**、 **g**、 **G**、**v**、および **V** サブコマンドを受け入れません。サブコマンドが終了すると、**G** サブコマンドはマークが付けられた次の行を表示して、同様に処理を続けます。**G** サブコマンドは、改行文字を null のサブコマンドとして解釈します。 :& (コロン・アンパーサンド) があると、 **G** サブコマンドは前のサブコマンドを再実行します。 **G** サブコマンドを停止するには、Ctrl+C を押します。
(1,$) **v/***Pattern***/***SubcommandList* **[l] [n] [p]**	変更した結果を表示したい場合は、**l** (list)、**n** (number)、または **p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。 **v** サブコマンドは、*Pattern* パラメーターに一致するものを含んでいないすべての行に対して、*SubcommandList* パラメーター内のサブコマンドを実行します。パターンは固定文字列または正規表現のいずれかです。 変更した結果を表示したい場合は、**l** (list)、**n** (number)、または **p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。 **v** サブコマンドは、**a**、 **i**、 **c**、 **g**、 **G**、および **V** サブコマンドを受け入れません。 **注: v** サブコマンドは、 **g** サブコマンド (パターンに一致するすべての行に対して、*SubcommandList* パラメーターを実行する) を補足します。

項目	説明
(1,$) **V**/*Pattern*/ **[l] [n] [p]**	**V** サブコマンドは、*Pattern* パラメーターに一致しないすべての行にマークを付け、マークを付けた最初の行を表示し、現在行をその行に設定して、サブコマンドを待ちます。パターンは固定文字列または正規表現のいずれかです。 変更した結果を表示したい場合は、**l** (list)、**n** (number)、または **p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。 **V** サブコマンドは、**a**、 **i**、 **c**、 **g**、 **G**、および **v** サブコマンドを受け入れません。 **注: V** サブコマンドは、 **G** サブコマンド (パターンに一致する行に、マークを付けます) の機能を補足するものです。

テキストのマーク付け

項目	説明
(.)**k***x* **[l] [n] [p]**	**k** (mark) サブコマンドは、アドレス指定した行に *x* パラメーターで指定した名前のマークを付けます。この名前は ASCII 文字の小文字でなければなりません。アドレス '*x* (マーク文字の前に引用符が 1 つ) は、この行にアドレス指定します。 **k** サブコマンドは、現在行を変更しません。 マークを付けたテキストを表示したい場合は、**l** (list)、 **n** (number)、**p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。

現在行にマークを付ける

次のサブコマンドを入力します。

```
kLetter[l][n][p]
```

この場合、*Letter* パラメーターは、マークを表す *a* から *z* までの文字で、 **l**、 **n**、および **p** は、マークの付いたテキストを表示するオプション・サブコマンドです。

アドレス指定した行にマークを付ける

次のサブコマンドを入力します。

```
AddresskLetter[l][n][p]
```

この場合、*Address* パラメーターは、マークを付けたい行の行番号または記号アドレスです。*Letter* パラメーターは、マークとして使用する *a* から *z* までの文字です。 **l**、 **n**、および **p** オプション・サブコマンドは、マークの付いたテキストを表示します。

テキストの移動

項目	説明
(.,.)(.,.) **m***A* **[l] [n] [p]**	**m** (move) サブコマンドは、1 つ以上のアドレス指定した行の位置を変更します。移動する最初の行は、*A* パラメーターでアドレス指定した行の後に置かれます。 A パラメーターが 0 であれば、アドレス指定した行または行範囲はファイルの先頭に移動します。移動する行のアドレスを *A* パラメーターに指定することはできません。 **m** サブコマンドは、最後に移動した行を現在行に設定します。 削除した結果を表示したい場合は、**l** (list)、 **n** (number)、または **p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。

1 行または一連の行を移動すると、指定した行が元の位置から削除され、新しい位置に配置されます。どの行を移動するかは、アドレスまたはパターンで選択することができます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行を移動する
- アドレスで指定した行を移動する
- 指定したパターンが入っている行を移動する
- 指定したパターンが入っていない行を移動する

現在行を移動する

次のサブコマンドを入力します。

```
mAddress[l][n][p]
```

この場合、*Address* パラメーターは、現在行の前の行の行番号または記号アドレスです。 **l**、 **n**、 **p** は、移動した行を表示するオプション・サブコマンドです。

アドレスで指定した行を移動する

次のサブコマンドを入力します。

```
LineNumbermDestinationAddress[l][n][p]
```

この場合、*LineNumber* パラメーターは移動したい行のアドレスで、 *DestinationAddress* パラメーターは移動先となる行の前の行です。 **l**、 **n**、および **p** オプション・サブコマンドは、移動した行を表示します。

指定したパターンが入っている行を移動する

次のサブコマンドを入力します。

```
[Address]g/Pattern/m[DestinationAddress][l][n][p]
```

この場合、*Address* は指定したパターンを含む行を探すために検索する行の範囲を指定するオプション・パラメーターで、 *Pattern* パラメーターは検索対象のテキスト、*DestinationAddress* は移動先となる行の前の行を表すオプション・パラメーターです。 **l**、 **n**、および **p** オプション・サブコマンドは、移動した行を表示します。

Address パラメーターを省略すると、 ed エディターはパターンが入っている行を探してファイル全体を検索します。*DestinationAddress* パラメーターを省略すると、移動されたテキストは現在行の後ろに挿入されます。

指定したパターンが入っていない行を移動する

次のサブコマンドを入力します。

```
[Address]v/Pattern/m[DestinationAddress][l][n][p]
```

この場合、*Address* は指定したパターンを含まない行を探すために検索する行の範囲を指定するオプション・パラメーターで、*Pattern* パラメーターはテキスト、*DestinationAddress* は移動したテキストの前の行を表すオプション・パラメーターです。 **l**、 **n**、および **p** オプション・サブコマンドは、移動した行を表示します。

Address パラメーターを省略すると、ed エディターはパターンが入っていない行を探してファイル全体を検索します。 *DestinationAddress* パラメーターを省略すると、移動されたテキストは現在行の後ろに挿入されます。

テキストの保存

項目	説明
(1,$)**w** *File*	**w** (write) サブコマンドは、アドレス指定した行をバッファーから *File* パラメーターで指定したファイルに移動します。ファイルが存在しない場合、**umask** の設定で別のファイル作成モードが指定されていなければ、 **w** サブコマンドは許可コード 666 (全ユーザーに対する読み取り/書き込み許可) を指定してファイルを作成します。 **w** サブコマンドは、デフォルトのファイル名を変更しません (*File* パラメーターが ed エディターを始動した後に使用した最初のファイル名でない場合)。ファイル名を入力しないと、**w** サブコマンドはデフォルトのファイル名を使用します。 **w** サブコマンドは、現在行を変更しません。 ed エディターは、バッファーからファイルを正常に書き込むと、書き込んだ文字数を表示します。ファイル名の代わりに **!***Command* サブコマンドを指定すると、 **w** サブコマンドは、 *Command* パラメーターで指定された、オペレーティング・システム・コマンドの出力結果を読み取ります。 **w** サブコマンドは、デフォルトのファイル名として指定したオペレーティング・システム・コマンドの名前を保存しません。 **注:** 0 は **w** サブコマンドにとって正しいアドレスではないので、**ed** コマンドで空のファイルを作成することはできません。

ファイルへの変更は、いくつかの方法で保存することができます。上記のフォーマットを使用して、次の操作を実行できます。

- ファイルを現行ファイルに保存する
- ファイルの一部を現行ファイルに保存する
- ファイルを別のファイルに保存する
- ファイルの一部を別のファイルに保存する

ファイルを現行ファイルに保存する

次のサブコマンドを入力します。

```
w
```

現行ファイルが現在の名前で保存され、 ed エディターは書き込まれた文字数を表示します。

ファイルの一部を現行ファイルに保存する

次のサブコマンドを入力します。

```
Addressw
```

この場合、*Address* パラメーターは、書き込まれる行または行範囲を指定します。 ed エディターは書き込まれた文字数を表示します。

ファイルを別のファイルに保存する

次のサブコマンドを入力します。

```
w File
```

この場合、*File* パラメーターは、書き込み先のファイルの名前です。

現行ファイルは、*File* パラメーターで指定したファイルに保存されます。 ed エディターは書き込まれた文字数を表示します。

ファイルの一部を別のファイルに保存する

次のサブコマンドを入力します。

```
Addressw File
```

この場合、*Address* パラメーターは、書き込まれる行または行範囲を指定します。*File* パラメーターは、書き込み先となるファイルを指定します。

指定された行が *File* パラメーターで指定されたファイルに保存されます。 ed エディターは書き込まれた文字数を表示します。

テキストの検索

現在行から正方向または逆方向に、テキストのパターンを検索することができます。パターンには、リテラル文字と特殊文字 ^ (脱字記号)、$ (ドル記号)、. (ピリオド)、[(左大括弧)、] (右大括弧)、* (アスタリスク)、¥ (円記号)、% (パーセント記号)、& (アンパーサンド) キーからなる文字列または正規表現を使用できます。

ed エディターを使用すると、次のテキスト検索を実行できます。

- 正方向に検索する
- 逆方向に検索する
- 同じ方向に再度検索をする
- 反対方向に再度検索をする

正方向に検索する

次のサブコマンドを入力します。

```
/Pattern
```

この場合、*Pattern* パラメーターは、検索対象となるテキストを指定する文字列または正規表現です。

カーソルは、パターンで指定したテキストの先頭文字に移動します。

逆方向に検索する

次のサブコマンドを入力します。

```
?Pattern
```

この場合、*Pattern* パラメーターは、検索対象となるテキストを指定する文字列または正規表現です。

カーソルは、パターンで指定したテキストの先頭文字に移動します。

同じ方向に再度検索をする

次のサブコマンドを入力します。

```
/
```

カーソルは、最後の検索コマンド内のパターンで指定したテキストの最も近いインスタンスの先頭文字に移動します。

反対方向に再度検索をする

次のサブコマンドを入力します。

```
?
```

カーソルは、最後の検索コマンド内のパターンで指定したテキストの最も近いインスタンスの先頭文字に移動します。

テキストの置換

項目	説明
(.,.)**s/***Pattern*/*Replacement*/ **[l] [n] [p]** (.,.)**s/***Pattern*/*Replacement*/**ng [l] [n] [p]**	**s** (substitute) サブコマンドは、アドレス指定した各行を検索して *Pattern* パラメーターと一致する文字列を探し、その文字列を *Replacement* パラメーターに指定された値で置き換えます。パターンは固定文字列または正規表現のいずれかです。一括 (global) サブコマンド (**g**) を指定しない場合、 **s** サブコマンドは、アドレス指定した各行で最初に一致した文字列だけを置換します。**g** サブコマンドを指定すると、**s** サブコマンドはアドレス指定した各行で一致する文字列のすべてのオカレンスを置換します。パターンに一致するものが見つからなければ、**s** サブコマンドは、エラー・メッセージ ? (疑問符) を戻します。 置換後のテキストを表示したい場合は、**l** (list)、**n** (number)、**p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。 **注:** スペースまたは改行文字を除き、どの文字でも *Pattern* パラメーターと *Replacement* パラメーターを区切ることができます。 **s** サブコマンドは、最後に変更した行を現在行に設定します。 *Number* パラメーター (整数) を指定すると、アドレス指定された各行において、その行上の文字列のうち最初に一致する番号が置換されます。 *Replacement* パラメーターに **&** (アンパーサンド) を使うと、 *Pattern* パラメーターと同じ値を持ちます。例えば、サブコマンド **s/are/&n't/** はサブコマンド **s/are/aren't/** と同じように機能し、現在行で **are** を **aren't** に置換します。¥& (円記号、アンパーサンド) では、*Replacement* パラメーター内で **&** 文字を使用した場合の特殊な意味は無効になります。 サブパターンは、文字列 ¥ ((円記号、左小括弧) と ¥) (円記号、右小括弧) で囲まれたパターンの部分で、パターンは囲み文字がない場合と同じように機能します。 *Replacement* パラメーターでは、¥*Number* はサブパターンに一致する文字列を指します。例えば、**s/¥(t¥)¥(h¥)¥(e¥)/t¥1¥2ose** サブコマンドは、現在行にパターン **the** に一致するものがあれば、**the** を **those** に置換します。サブパターンがネストされているか連続しているかに関係なく、¥*Number* は、 *Number* パラメーターで指定したオカレンスを、区切り文字 ¥) (円記号、右小括弧) の左から数えて表します。 % (パーセント記号) 文字を、*Replacement* パラメーターとして単独で使用すると、**s** サブコマンドは直前の *Replacement* パラメーターを再使用します。ただし、% が *Replacement* パラメーターの一部である場合、または ¥ (円記号) が前に付いている場合は、特別な意味を持ちません。 改行文字を行に代入することによって、行を分割することができます。 *Replacement* パラメーターで次のようにします。¥-Enter キー・シーケンスを押すと、改行文字 (非表示) が入力され、それ以降の文字列と共にカーソルが次の行に移動します。 **g** サブコマンドや、 **v** サブコマンドのリストの一部を、改行文字で置換することはできません。

ed エディターを使用すると、いくつかの方法でテキストを置換することができます。上記のフォーマットを使用して、次の編集タスクを実行できます。

- 現在行内のテキストを置換する
- アドレス指定した行または行範囲内のテキストを置換する
- 指定したパターンが入っている行内で、そのパターンを置換する
- あるパターンが入っている行内で、別のパターンを置換する
- あるパターンが入っていない行内で、別のパターンを置換する

現在行内でテキストを置換する

1. 次のサブコマンドを入力します。

   ```
   s/OldString/NewString
   ```

   この場合、*OldString* パラメーターは既存のテキストで、*NewString* パラメーターはそれと置換したいテキストです。
2. 次のいずれかを入力します。

   現在行内で、*OldString* パラメーターの最初のインスタンスを *NewString* パラメーターに置換するには、次のように入力します。

   ```
   /
   ```

   現在行内で、*OldPattern* パラメーターのすべてのインスタンスを *NewString* パラメーターに置換するには、次のように入力します。

   ```
   /g
   ```
3. 変更後のテキストを表示するには、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l

   n

   p
   ```
4. Enter を押します。

アドレス指定した行または行範囲内のテキストを置換する

1. 次のサブコマンドを入力します。

   ```
   Addresss/OldPattern/NewString
   ```

   この場合、*Address* パラメーターは、テキストを置換したい行または行範囲のアドレスです。*OldPattern* パラメーターは既存のテキストです。*NewString* パラメーターは、置換したいテキストです。
2. 次のいずれかを入力します。

   各行で、*OldPattern* パラメーターの最初のインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /NewString/
   ```

   各行で、*OldPattern* パラメーターのすべてのインスタンスを、*NewString* パラメーターに置換するには、以下のように入力します。

   ```
   /NewString/g
   ```

   各アドレス行で *NumberOldPattern* パラメーターの最初のインスタンスを *NewString* パラメーターに置換するには、次のように入力します。

   ```
   /NewString/Number
   ```
3. 変更後のテキストを表示するには、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l
   ```

```
n

p
```

4. Enter を押します。

指定したパターンが入っている行内で、そのパターンを置換する

1. 次のサブコマンドを入力します。

   ```
   Addressg/Pattern/s//NewString
   ```

   この場合、*Address* パラメーターは、*Pattern* パラメーターで指定したパターンを検索したい行範囲のアドレスです。*NewString* パラメーターは、*Pattern* パラメーターの置換に使用したいテキストです。
2. 次のいずれかを入力します。

   各行で、*Pattern* パラメーターの最初のインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /
   ```

   各行内で、*Pattern* パラメーターのすべてのインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /g
   ```
3. 変更後のテキストを表示するには、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l

   n

   p
   ```
4. Enter を押します。

あるパターンが入っている行内で、別のパターンを置換する

1. 次のサブコマンドを入力します。

   ```
   Addressg/Pattern/s/OldString/NewString
   ```

   この場合、*Address* パラメーターは、*Pattern* パラメーターで指定したパターンを検索したい行範囲のアドレスです。*OldString* パラメーターは、置換したいテキストです。*NewString* パラメーターは、*OldString* の代わりに用いるテキストです。
2. 次のいずれかを入力します。

   Pattern パラメーターが入っている各行内で、*OldString* パラメーターの最初のインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /
   ```

   Pattern パラメーターを含む各行内で、*OldString* パラメーターのすべてのインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /g
   ```
3. 変更後のテキストを表示するには、次のいずれかのオプション・サブコマンドを入力します。

```
l

n

p
```

4. Enter を押します。

あるパターンが入っていない行内で別のパターンを置換する

1. 次のサブコマンドを入力します。

   ```
   Addressv/Pattern/s/OldString/NewString
   ```

   この場合、*Address* パラメーターは、*Pattern* パラメーターで指定したパターンを検索したい行範囲のアドレスです。*OldString* パラメーターは、置換したいテキストです。*NewString* パラメーターは、*OldString* の代わりに用いるテキストです。
2. 次のいずれかを入力します。

   Pattern パラメーターが入っていない各行内で、*OldString* パラメーターの最初のインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /
   ```

   Pattern パラメーターが入っていない各行内で、*OldString* パラメーターのすべてのインスタンスを、*NewString* パラメーターに置換するには、次のように入力します。

   ```
   /g
   ```
3. 変更後のテキストを表示するには、次のいずれかのオプション・サブコマンドを入力します。

   ```
   l

   n

   p
   ```
4. Enter を押します。

テキスト変更の取り消し

項目	説明
u [**l**] [**n**] [**p**]	**u** (undo) サブコマンドは、バッファーの状態を、ed エディター・サブコマンドによる前回の変更以前の状態に戻します。 **u** サブコマンドでは、 **e**、 **f**、および **w** サブコマンドの取り消しはできません。 変更した結果を表示したい場合は、**l** (list)、**n** (number)、または **p** (print) サブコマンドのいずれかを入力します。これらのサブコマンドはオプションです。

テキストの変更を取り消す

次のサブコマンドを入力します。

```
u[l][n][p]
```

この場合、**l**、 **n**、および **p** は、変更を表示するオプション・サブコマンドです。最後に保存した後でテキストに対して実行した追加、変更、移動、コピー、削除の各編集機能がすべて変更前の状態に戻ります。

ファイルの操作

ed エディター・サブコマンドを使用すると、ファイルを操作して次のタスクを実行できます。

- 現行ファイルに別のファイルを追加する
- デフォルトのファイル名を変更する
- 別のファイルの編集

現行ファイルに別のファイルを追加する

項目	説明
($)**r** *File*	**r** (read) サブコマンドは、バッファー内のアドレス指定した行の後にファイルを読み込みます。 **r** サブコマンドは、バッファー内の前の内容は削除しません。 *File* パラメーターを指定しないと、デフォルトのファイルがあれば、 **r** サブコマンドはそれをバッファーに読み込みます。 **r** サブコマンドは、デフォルトのファイル名を変更しません。 アドレス 0 を指定すると、 **r** サブコマンドはファイルをバッファーの先頭へ読み込みます。ファイルの読み取りが成功すると、 **r** サブコマンドはバッファーに読み込んだ文字数を表示して、最後に読み込んだ行を現在行に設定します。 **r** サブコマンド内の *File* パラメーターを、 ! (感嘆符) に置き換えると、その行の残りは、出力が読み込まれるオペレーティング・システムのシェル・コマンドと解釈されます。 **r** サブコマンドは、オペレーティング・システムのコマンド名をデフォルトのファイル名として保管しません。

現在行の後にファイルを挿入する

次のサブコマンドを入力します。

```
r File
```

この場合、*File* パラメーターは挿入されるファイルの名前です。

ed エディターは、*File* パラメーターで指定したファイルを、現行ファイル内の現在行の後に読み込み、現行ファイルに読み込んだ文字数を表示します。

アドレスで指定した行の後にファイルを挿入する

次のサブコマンドを入力します。

```
Addressr File
```

この場合、*Address* パラメーターは、挿入先となる行の前の行を指定します。*File* パラメーターは、挿入されるファイルの名前です。

ed エディターは、*File* パラメーターで指定したファイルを、現行ファイル内の指定した行の後に読み込んで、現行ファイルに読み込んだ文字数を表示します。

デフォルトのファイル名を変更する

項目	説明
f [*File*]	**f** (file name) サブコマンドは、デフォルトのファイル名 (格納された、最後に使用したファイルの名前) を、*File* パラメーターで指定した名前に変更します。*File* パラメーターを指定しないと、 **f** サブコマンドは、デフォルト・ファイル名を表示します。 (**e** サブコマンドは、デフォルトのファイル名を保管します。)

ファイル名を表示する

次のサブコマンドを入力します。

```
f
```

ed エディターは、編集バッファー内のファイルの名前を表示します。

ファイルに名前を付ける

次のサブコマンドを入力します。

```
f File
```

この場合、*File* パラメーターは、編集バッファー内のファイルの新しい名前です。

編集バッファー内のファイルの名前が変更されます。

別のファイルの編集

項目	説明
e *File*	**e** (edit) サブコマンドは、最初にバッファーの内容をすべて削除し、現在行をバッファーの最終行に設定して、バッファーに読み込まれた文字数を表示します。 (**w** サブコマンドにより) 前回の内容を保存した後に、バッファーに変更を行った場合、ed エディターはそのバッファーを消去する前に、 ? (疑問符) を表示します。 **e** サブコマンドは、*File* パラメーターを、その後の **e**、**r**、または **w** サブコマンドで必要に応じて使用される、デフォルトのファイル名として保管します。(デフォルトのファイル名を変更するには、 **f** サブコマンドを使用してください。) *File* パラメーターを ! (感嘆符) に置き換えると、 **e** サブコマンドは、その行の残りをオペレーティング・システムのシェル・コマンドと解釈して、コマンド出力を読み込みます。**e** サブコマンドは、シェル・コマンドの名前をデフォルトのファイル名としては保管しません。
E *File*	**E** (Edit) サブコマンドは、**e** サブコマンドと同様に機能します。ただし、**E** サブコマンドは、最後に **w** サブコマンドを実行した後で加えられたバッファーへの変更を、検査しません。ファイルを再編集する前に行った変更はすべて失われます。

e または **E** サブコマンドを使用すると、次のタスクを実行できます。

- 現行ファイルを保存せずに再編集する
- 現行ファイルを保存してから再編集する
- 現行ファイルを保存した後で別のファイルを編集する
- 現行ファイルを保存せずに別のファイルを編集する

現行ファイルを保存せずに再編集する

次のサブコマンドを入力します。

```
E
```

ed エディターは、ファイル内の文字数を表示します。ファイルを再編集する前に行った変更はすべて失われます。

現行ファイルを保存してから再編集する

次のサブコマンドを入力します。

```
e
```

ed エディターは、ファイル内の文字数を表示します。

現行ファイルを保存した後で別のファイルを編集する

次のサブコマンドを入力します。

```
e File
```

この場合、*File* パラメーターは、編集したい新規ファイルまたは既存のファイルの名前です。

既存のファイルの場合、ed エディターはファイル内の文字数を表示します。新しいファイルの場合、ed エディターは ? (疑問符) とそのファイルの名前を表示します。

現行ファイルを保存せずに別のファイルを編集する

次のサブコマンドを入力します。

```
E File
```

この場合、*File* パラメーターは、編集したい新規ファイルまたは既存のファイルの名前です。

既存のファイルの場合、ed エディターはファイル内の文字数を表示します。新しいファイルの場合、ed エディターは ? (疑問符) とそのファイルの名前を表示します。

ed エディター・サブコマンドのその他の機能

ed エディター・サブコマンドを使用すると、次のタスクを実行できます。

- プロンプト文字列を変更する
- システム・コマンドを入力する
- ed エディターを終了する
- ヘルプを要求する

プロンプト文字列を変更する

項目	説明
P	**P** (Prompt) サブコマンドは * (アスタリスク) で表される ed エディターのプロンプト文字列をオンまたはオフにします。初期状態では、**P** サブコマンドはオフになっています。

プロンプト文字列の表示を開始または停止する

次のサブコマンドを入力します。

```
P
```

ed エディターのプロンプトである * (アスタリスク) が、以前の設定に応じて表示されるか、または表示されなくなります。

システム・コマンドを入力する

項目	説明
! *Command*	! サブコマンドを使用すると、ed エディターを終了せずにオペレーティング・システム・コマンドを実行できます。 ed エディターのサブコマンド・ラインで ! サブコマンドの後にあるものは、すべてオペレーティング・システム・コマンドと解釈されます。そのコマンド文字列のテキスト内で、ed エディターは、エスケープされていない % (パーセント記号) がある場合、それを現行ファイル名と置き換えます。 ! ed エディター・サブコマンドの後に、 ! (感嘆符) を入力すると、直前のオペレーティング・システム・コマンドを繰り返すことができます。オペレーティング・システムのコマンド・インタープリター (**sh** コマンド) が、コマンド文字列を展開した場合、ed エディターは展開された行をエコーします。 ! サブコマンドは、現在行を変更しません。

! サブコマンドを使用すると、次の操作を実行できます。

- 1 つのオペレーティング・システム・コマンドを実行する
- 1 つのオペレーティング・システム・コマンドを再度実行する
- 複数のオペレーティング・システム・コマンドを実行する

1 つのオペレーティング・システム・コマンドを実行する

次のサブコマンドを入力します。

```
!Command
```

この場合、*Command* パラメーターは、通常はプロンプトから入力されるオペレーティング・システム・コマンドを指定します。

コマンドが実行され、その出力が表示されます。コマンドが完了すると、エディターは ! (感嘆符) を表示します。

1 つのオペレーティング・システム・コマンドを再度実行する

次のサブコマンドを入力します。

```
!
```

前に実行されたオペレーティング・システム・コマンドが実行され、その出力が表示されます。コマンドが完了すると、エディターは ! (感嘆符) を表示します。

複数のオペレーティング・システム・コマンドを実行する

1. 次のサブコマンドを入力して、オペレーティング・システムのプロンプトを表示させます。

   ```
   !sh
   ```
2. オペレーティング・システム・コマンドを入力します。
3. Enter を押してコマンドを実行し、その出力を表示します。
4. ステップ 2 と 3 を繰り返して、オペレーティング・システム・コマンドの実行をさらに続けます。
5. Ctrl+D を押して、コマンド・モードに戻ります。エディターは ! (感嘆符) を表示します。

ed エディターを終了する

項目	説明
q	**q** (quit) サブコマンドは、最後の変更内容が入力された後でバッファーがファイルに保存されたかどうかを検査してから、ed エディターを終了します。バッファーがファイルに保存されていなければ、**q** サブコマンドは **?**(疑問符) メッセージを表示します。ed エディターを終了するには、**q** サブコマンドをもう一度入力します。現行ファイルに書き込まれた変更内容は失われます。
Q	**Q** (Quit) サブコマンドは、バッファーをファイルに保存した後で変更が行われたかどうか検査せずに、ed エディターを終了します。最後に保存した後でバッファーに対して行われた変更はすべて失われます。

編集結果を検査してから終了する

1. 次のサブコマンドを入力します。

   ```
   q
   ```

2. ed エディターが **?** を表示したら、次のいずれかのサブコマンドを入力します。

   終了する前に変更結果を保存するには、次のように入力し、

   ```
   w
   ```

   次に Enter を押します。

   変更結果を保存せずに終了するには、次のように入力します。

   ```
   q
   ```

3. Enter を押します。

終了して編集結果を破棄する

1. 次のサブコマンドを入力します。

   ```
   Q
   ```

2. Enter を押します。最後に保存した後でバッファーに対して行われた変更はすべて失われます。

ヘルプを要求する

項目	説明
h	**h** (help) サブコマンドは、最後に表示された **?** (疑問符) 診断またはエラー・メッセージに関する簡単なヘルプ・メッセージを表示します。
H	**H** (Help) サブコマンドを使用すると、ed エディターは、その後のすべての ? (疑問符) 診断メッセージに対するヘルプ・メッセージが表示します。また、**H** サブコマンドは、前の ? (疑問符) があれば、その説明も表示します。 **H** サブコマンドは、このモードのオンとオフを交互に切り替えます。初期状態ではオフになっています。

ヘルプ・メッセージの表示を開始または停止する

次のサブコマンドを入力します。

```
H
```

ed エディターからの ? の応答に対して、前の設定値に応じて、ヘルプ・メッセージが表示されたり、表示されなかったりします。

前回のヘルプ・メッセージを表示する

次のサブコマンドを入力します。

```
h
```

ed エディターからの最後の ? の応答に対して、ヘルプ・メッセージが表示されます。

ed エディターでの文字クラスのサポート

標準の *Patterns* 式では、範囲式は、現行ロケールの照合シーケンス内で 2 つの文字の間に入るすべての文字セットに対するマッチングを行います。範囲式の構文は次のとおりです。

[*character-character***]**

照合シーケンスの第 1 の文字は、第 2 の文字と等しいかそれ以下でなければなりません。例えば、[a-c] は En_US ロケール内の文字、a、b、c のいずれとも一致します。

範囲式は、通常、文字クラスのマッチングを行うのに使用されます。例えば、[0-9] はすべての数字を意味し、[a-z A-Z] はすべての文字を意味します。範囲が現行ロケールの照合シーケンスに従って解釈される場合、このフォーマットを使うと予期しない結果を招くことがあります。

上記のフォーマットの代わりに、[ ] (大括弧) で囲んだ文字クラスの式を使用して、文字のマッチングを行ってください。システムは、このタイプの式を現行ロケールの定義に従って解釈します。しかし、範囲式に文字クラス式は使用できません。

文字クラス式の構文は次のとおりです。

[:*CharacterClass***:]**

つまり、左大括弧、コロン、文字クラス名、もう 1 つのコロン、右大括弧の順になります。

すべてのロケールでサポートされる文字クラスは、次のとおりです。

項目	説明
[:upper:]	大文字
[:lower:]	小文字
[:alpha:]	大文字と小文字
[:digit:]	数字
[:alnum:]	英数字
[:xdigit:]	16 進数字
[:punct:]	句読文字 (制御文字でも英数字でもない)
[:space:]	スペース、タブ、復帰、改行、垂直タブ、用紙送りの各文字
[:print:]	スペースを含む印刷可能文字
[:graph:]	スペースを含まない印刷可能文字
[:cntrl:]	制御文字
[:blank:]	スペースとタブ文字

大括弧は文字クラス定義の一部です。大文字の ASCII 文字または ASCII 数字に一致させるには、次の正規表現を使います。

```
[[:upper:] [:digit:]]
```

`[A-Z0-9]` という式は使いません。

ロケールによっては上記のリスト以外の文字クラスをサポートすることがあります。

改行文字は `[:space:]` 文字クラスの一部ですが、この文字クラスでは一致しません。改行文字は、特殊検索文字 $ (ドル記号) と ^ (脱字記号) でのみ一致させることができます。

終了状況

ed コマンドおよび **red** コマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

関連資料:

『edit コマンド』

関連情報:

rsh コマンド

sed コマンド

view コマンド

---

edit コマンド

目的

初心者ユーザー用の簡単なライン・エディターを提供します。

構文

edit [**-r**] [*File* ...]

説明

edit コマンドは、ex エディターの簡易バージョンである、初心者用に設計されたライン・エディターを始動します。 edit エディターは、ed エディター、ex エディター、vi エディターなどのエディター・ファミリーに属しています。edit エディターについての知識は、さらに進んだ機能を持つ他のエディターを学習するときに役立ちます。ファイルの内容を編集するには、次のように入力します。

```
edit  File
```

File パラメーターに既存のファイル名を指定すると、**edit** コマンドはそのファイルをバッファーにコピーして、その中の行数と文字数を表示します。次に **:** (コロン) プロンプトを表示して、標準入力からサブコマンドを読み込む用意ができたことを示します。

File パラメーターで指定したファイルがまだ存在しなければ、**edit** コマンドはそのことを表示して新しいファイルを作成します。 *File* パラメーターには複数のファイル名を指定することができます。その場合、**edit** コマンドは最初のファイルをバッファーにコピーして、後で使用するために残りのファイル名を引数リストに格納します。 **w** サブコマンドを使って変更を書き込むまで、edit エディターは、編集されるファイルに変更を加えません。

edit エディターは次の 2 つのモードのいずれかで動作します。

項目	説明
コマンド・モード	edit エディター・サブコマンドを認識して実行します。edit エディターの開始時はコマンド・モードになっています。開始時以外の時点でコマンド・モードに入るには、行の始めに **.** (ピリオド) だけを入力します。
テキスト入力モード	edit エディター・バッファーにテキストを入力することができます。テキスト入力モードに入るには、 **append (a)** サブコマンド、 **change (c)** サブコマンド、または **insert (i)** サブコマンドのいずれかを使用します。テキスト入力モードを終了するには、行の始めに **.** (ピリオド) だけを入力します。

フラグ

項目	説明
-r	エディターまたはシステムに誤動作があった場合、編集中のファイルを回復させます。

ファイル内の行のアドレッシング

edit エディターは、次の 3 種類のアドレスを使用します。

- 行番号アドレス
- 相対位置アドレス
- パターン・アドレス

行番号アドレス

行番号アドレスは、ファイル内の行を行番号またはシンボル名で指定します。この方法で行または行範囲のアドレスを指定するのが最も簡単です。

先頭行をそのシンボル名でアドレス指定するには、次のように入力します。

```
.
```

最終行をそのシンボル名でアドレス指定するには、次のように入力します。

```
$
```

また、行番号アドレスまたは記号アドレスをコンマまたはセミコロンで区切って、行の範囲を指定することもできます。第 2 のアドレスは、範囲内で最初にアドレス指定した行より後ろにある行でなければなりません。

次に例を示します。

```
1,5
```

これにより、1 行目から 5 行目までがアドレス指定されます。

```
.,$
```

上記の場合は、先頭行から最終行までがアドレス指定されます。

相対位置アドレス

edit エディターは、行を現在行に対して相対的にアドレス指定することができます。 *-Number* パラメーターまたは *+Number* パラメーターで始まるアドレスはそれぞれ、指定した数だけ現在行の前または後ろにある行をアドレス指定します。

次に例を示します。

```
+8
```

これにより、現在行から 8 行後の行がアドレス指定されます。

-Number アドレスまたは *+Number* アドレスとシンボル名を併用して、先頭行または最終行に対して相対的に行をアドレス指定することもできます。

次に例を示します。

```
.+3
```

これにより、先頭行から 3 行後の行がアドレス指定されます。

```
$-10
```

これにより、最終行から 10 行前の行がアドレス指定されます。

パターン・アドレス

バッファー内で特定のパターンを検索して、行のアドレスを指定することができます。 edit エディターは、正方向または逆方向に検索し、*Pattern* パラメーターに一致するものがある最初の行で停止します。必要であれば、検索処理は一致するものが見つかるまで、バッファーの終わりまたは先頭で折り返すか、あるいは現在行に戻ります。

正方向に検索するには、次のように入力します。

```
/Pattern/
```

逆方向に検索するには、次のように入力します。

```
?Pattern?
```

Pattern パラメーターをコンマまたはセミコロンで区切って行の範囲を指定することもできます。第 2 のアドレスは、範囲内で最初にアドレス指定した行より後ろにある行でなければなりません。

次に例を示します。

```
Pattern,Pattern
```

次の文字は、*Pattern* パラメーターの一部として使用すると特殊な意味を持ちます。

項目	説明
^	*Pattern* パラメーターの最初の文字として使用すると、行の先頭に一致します。
$	*Pattern* パラメーターの最後の文字として使用すると、行の終わりに一致します。

edit エディター・サブコマンドの使用法

edit エディター・サブコマンドは、現在行に影響し、現在行は **.** (ピリオド) で表されます。edit エディターの始動時には、現在行はバッファー内の最終行です。バッファーを編集すると、現在行はサブコマンドの影響を受けた最後の行に変わります。ファイルの別の部分で作業を行うには、現在行の検索方法と、ファイル内の別の行をアドレス指定する方法を、理解している必要があります。

edit エディター・サブコマンドを使用すると、次のタスクを実行できます。

- テキストの追加
- 現行ファイルの名前変更
- テキストの変更
- テキストの削除
- 現行ファイルの名前と状況の表示
- テキストの表示と現在行の検索
- 別のファイルの編集
- edit エディターの終了
- 一括変更
- テキストの移動またはコピー

- システム・クラッシュ後のファイルの保存
- テキストの保存
- テキストの置換
- 変更の取り消し

テキストの追加

次のサブコマンドで、*Address* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
[*Address*]**append** (**a**) *Text* **.**	*Address* パラメーターを指定しない場合、入力したテキストは現在行の後に追加されます。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、**a** サブコマンドは指定した行の後にテキストを追加します。アドレス 0 を指定すると、**a** サブコマンドはバッファーの先頭にテキストを置きます。 テキストを入力するときは、各行の終わりで Enter キーを押してください。テキストすべてを入力し終わったら、行の始めに **.** (ピリオド) だけを入力して、テキスト入力モードを終了し、コマンド・モードに戻ります。 **1,$p** サブコマンドを使用すると、バッファーの内容全体を表示できます。 **注: a** サブコマンドと、**i** サブコマンドとの違いは、テキストの挿入位置です。
[*Address*]**insert** (**i**)*Text***.**	*Address* パラメーターを指定しない場合、現在行の前にテキストが挿入されます。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、**i** サブコマンドは指定した行の前にテキストを挿入します。アドレス 0 を指定することはできません。 テキストを入力するときは、各行の終わりに Enter キーを押してください。テキストをすべて入力し終わったら、行の始めに **.** (ピリオド) だけを入力して、テキスト入力モードを終了し、コマンド・モードに戻ります。 **1,$p** サブコマンドを使用すると、バッファーの内容全体を表示できます。 **注: i** サブコマンドと、**a** サブコマンドとの違いは、テキストの挿入位置です。

現行ファイルの名前変更

項目	説明
file *File*	現行ファイルの名前を、*File* パラメーターで指定した名前に変更します。 edit エディターは、このファイルが編集されたものとは見なしません。

テキストの変更

次のサブコマンドで、*Address* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
[*Address1*,*Address2*]**change** (**c**). *Text*	*Address* パラメーターを指定しないと、現在行が入力したテキストに置換されます。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、**c** サブコマンドはアドレス指定した行または行範囲を置換します。複数のアドレスをコンマで区切って、行の範囲を指定できます。 テキストを入力するときは、各行の終わりに Enter キーを押してください。テキストをすべて入力し終わったら、行の始めに **.** (ピリオド) だけを入力して、テキスト入力モードを終了し、コマンド・モードに戻ります。 **1,$p** サブコマンドを使用すると、バッファーの内容全体を表示できます。最後に入力した行が現在行になります。

テキストの削除

次のサブコマンドで、*Address* パラメーターと *Buffer* パラメーターはオプションです。アドレスおよびバッファーを指定するときは、大括弧を入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
[*Address1*,*Address2*]**delete** [*Buffer*] (**d**)	*Address* パラメーターを指定しないと、現在行が削除されます。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、**d** サブコマンドはアドレス指定した行または行範囲を削除します。複数のアドレスをコンマで区切って、行の範囲を指定できます。最後に削除された行の次の行が現在行になります。 a から z までの小文字を入力してバッファーを指定すると、edit エディターはアドレス指定した行をそのバッファーに保存します。大文字を指定すると、ed エディターはそのバッファーに行を追加します。 **pu** サブコマンドを使用すると、削除した行をバッファーに戻すことができます。

現行ファイルの名前と状況の表示

次のサブコマンドには、完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
file (**f**)	現行ファイル名を次の関連情報と一緒に表示します。 • **w** サブコマンドの最後の実行時以降に、ファイルが修正されたかどうか • 現在行の番号 • バッファー内の行数 • 現在行の位置を示すバッファー内のパーセント

テキストの表示および現在行の検索

次のサブコマンドで、*Address* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) のいずれかを使用できます。

項目	説明
[*Address1,Address2*]**number** (**nu**)	アドレス指定した行または行範囲の前に、そのバッファーの行番号が表示されます。 *Address* パラメーターを指定しないと、**nu** サブコマンドは現在行とその番号を表示します。 アドレスを指定すると、**nu** サブコマンドはアドレス指定した行または行範囲を表示します。複数のアドレスをコンマで区切って、行の範囲を指定できます。最後に表示された行が現在行になります。
[*Address1,Address2*] **print** (**p**)	アドレス指定した行または行範囲が表示されます。 *Address* パラメーターを指定しないと、**p** サブコマンドは現在行を表示します。 アドレスを指定すると、**p** サブコマンドはアドレス指定した行または行範囲を表示します。複数のアドレスをコンマで区切って、行の範囲を指定できます。最後に表示された行が現在行になります。
[*Address*]**=**	アドレス指定した行の行番号が表示されます。*Address* パラメーターを指定しないと、**=** サブコマンドは現在行の行番号を表示します。
[*Address*]**z**	アドレス指定した行で始まるテキストの画面が表示されます。 *Address* パラメーターを指定しないと、**z** サブコマンドは現在行で始まるテキスト画面を表示します。
[*Address*]**z-**	アドレス指定した行が最下部に入っているテキストの画面が表示されます。 *Address* パラメーターを指定しないと、**z-** サブコマンドは現在行が最下部に入っているテキスト画面を表示します。
[*Address*]**z.**	アドレス指定した行が中央に入っているテキスト画面が表示されます。 *Address* パラメーターを指定しないと、**z.** サブコマンドは現在行が中央に入っているテキスト画面を表示します。

別のファイルの編集

次のサブコマンドには、完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
edit *File* (**e**)	*File* パラメーターで指定した新しいファイル上で編集セッションを開始します。エディターは、まず、**write (w)** サブコマンドの最後の実行時以降に、バッファーが編集されたかどうかを調べます。 最後の **w** サブコマンドの後でファイルが編集されていると、edit エディターは警告メッセージを表示して **e** サブコマンドを取り消します。それ以外の場合、edit エディターはエディター・バッファーの内容を削除し、指定したファイルを現行ファイルにして、新しいファイル名を表示します。 このファイルを編集できることを確認した後で、edit エディターはファイルをバッファーに読み込みます。 edit エディターは、エラーなしでファイルを読み込むと、読み込んだ行数と文字数を表示します。最後に読み込まれた行が新しい現在行になります。
next (**n**)	コマンド・ライン引数リスト内で指定した次のファイルを、編集のためにバッファーにコピーします。

edit エディターの終了

次のサブコマンドには、完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
quit (**q**)	**write (w)** サブコマンドを使用した後、編集セッションを終了します。バッファーを変更し、変更結果を書き込んでいないと、警告メッセージが表示されて、編集セッションは終了されません。
quit!(**q!**)	編集セッションを終了します。最後の **w** サブコマンドの後にバッファーに対して行った変更内容は破棄されます。

一括変更

次のサブコマンドで、*Address* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
[*Address1*,*Address2*]**global/***Pattern*/*SubcommandList* (**g**)	アドレス指定した行のうち、*Pattern* パラメーターに一致する各行にマークを付けます。 edit エディターは、マークされた各行について、*SubcommandList* パラメーターで指定した一連のサブコマンドのリストを実行します。 *Address* パラメーターを指定しないと、**g** サブコマンドは現在行に対して機能します。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、**g** サブコマンドはアドレス指定した行または行範囲に対して処理を行います。複数のアドレスをコンマで区切って、行の範囲を指定できます。 単一のサブコマンド、またはサブコマンド・リスト内の最初のサブコマンドを、**g** サブコマンドと同じ行に入力します。残りのサブコマンドそれぞれは別の行に入力する必要があります。この場合、各行 (最終行を除く) には **¥** (円記号) を付けます。デフォルトのサブコマンドは、 **print (p)** サブコマンドです。 サブコマンド・リストには、 **append (a)** サブコマンド、 **insert (i)** サブコマンド、**change (c)** サブコマンドと、それに関連する入力を指定できます。この場合、最後のピリオドがコマンド・リストの最終行にある場合は、省略してもかまいません。 **注: undo (u)** サブコマンドと、**g** サブコマンドは、サブコマンド・リストには指定できません。

テキストの移動またはコピー

次のサブコマンドで、*Address1* パラメーターと *Address2* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。*Address3* パラメーターは必ず指定する必要があります。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) のいずれかを使用できます。

項目	説明
[*Address1*,*Address2*]**move** *Address3* (**m**)	アドレスまたはアドレスの範囲を指定しないと、*Address3* パラメーターで指定した行の後に、場合によっては、現在行が移動されます。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、**m** サブコマンドはアドレス指定した行または行範囲を移動します。複数のアドレスをコンマで区切って、アドレスの範囲を指定できます。移動された行の先頭行が現在行になります。
[*Address1*,*Address2*]**yank** [*Buffer*] (**ya**)	この場合、*Buffer* は、 a から z までの単一の英文字で指定するオプション・パラメーターです。 **pu** サブコマンドを使用すると、これらの行を別のファイルに入れることができます。

項目	説明
[*Address*]**put** [*Buffer*] (**pu**)	指定した *Buffer* パラメーターの内容を取り出し、現在行の後に配置します (アドレスを指定していない場合)。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。 アドレスを指定すると、 **pu** サブコマンドは、指定したバッファーの内容を取り出し、アドレス指定した行の後に配置します。*Buffer* パラメーターを指定しないと、**pu** サブコマンドは最後に削除またはコピーされたテキストを復元します。 **pu** サブコマンドを、**delete (d)** サブコマンドと併用すると、ファイル内で行を移動できます。また、**yank (ya)** サブコマンドと併用すると、ファイル間で行を複製できます。 **pu** および **ya** サブコマンドは、マクロ内では使用できません。

システム誤動作後のファイルの保存

項目	説明
preserve	システムが誤動作した場合と同方式で、エディター・バッファーの現在の内容を保存します。 **write (w)** サブコマンドを実行した結果、エラーが発生したとき、作業を保存する方法が分からない場合に、このサブコマンドを使用します。ファイルを回復させるには、**recover** サブコマンドを使用します。
recover *File*	*File* パラメーターで指定したファイルを、システムの保存領域から回復させます。このサブコマンドは、システムがクラッシュした後、または **preserve** サブコマンドを実行した後に使用します。

テキストの保存

次のサブコマンドで、*Address* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
[*Address1,Address2*]**write** [*File*] (**w**)	アドレスを指定しないと、バッファーの内容全体を *File* パラメーターで指定したファイルに書き込みます。 アドレスを指定すると、**w** サブコマンドはアドレス指定した行または行範囲を、指定したファイルに書き込みます。複数のアドレスをコンマで区切って、行の範囲を指定できます。 edit エディターは、書き込んだ行数と文字数を表示します。 ファイルを指定しないと、edit エディターは現行ファイル名を使用します。 *File* パラメーターが存在しなければ、エディターは新しいファイル・パラメーターを作成します。

テキストの置換

次のサブコマンドで、*Address* パラメーターはオプションです。アドレスを指定するときは、大括弧は入力しないでください。完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) のいずれかを使用できます。

項目	説明
[*Address1,Address2*]**substitute/***Pattern*/*Replacement*/(**s**)[*Address1,Address2*]**substitute/***Pattern*/*Replacement*/**g**	アドレス指定した各行で、指定した *Pattern* パラメーターの最初のインスタンスを置換します。**global (g)** サブコマンドを、 **s** サブコマンドの終わりに追加すると、 *Pattern* パラメーターのインスタンスすべてを置換できます。 アドレスを指定しない場合、**s** サブコマンドは現在行に対して処理を行います。バッファー内での位置が正しくないときは、場合によっては、 現在行の検索をするか、 アドレスを指定する必要があります。アドレスを指定すると、**s** サブコマンドはアドレス指定した行または行範囲に対して処理を行います。複数のアドレスをコンマで区切って、行の範囲を指定できます。

変更の取り消し

次のサブコマンドには、完全なサブコマンドの形またはその省略形 (小括弧で囲まれている) を使用できます。

項目	説明
undo (u)	最後のバッファー編集サブコマンドによってバッファー内で行った変更を取り消して、変更前の状態に戻します。**write (w)** サブコマンド、または **edit (e)** サブコマンドの、取り消しはできません。 **注: global** サブコマンドは、**u** サブコマンドでは、1 つのサブコマンドと見なされます。

関連資料:

299 ページの『ed または red コマンド』

454 ページの『ex コマンド』

関連情報:

vi または vedit

---

edquota コマンド

目的

ユーザーおよびグループのクォータを編集します。

構文

ユーザー・クォータを編集する

edquota [**-u**] [**-p** *Proto-UserName*] *UserName* ...

グループ・クォータを編集する

edquota [**-g** [**-p** *Proto-GroupName*] *GroupName* ...]

ユーザーまたはグループの猶予期間を編集する

edquota -t [**-u** | **-g**]

説明

edquota コマンドは、JFS ファイルシステムのクォータを作成して編集します。 JFS2 ファイルシステムのクォータの管理方法については、「*Commands Reference, Volume 3*」の **j2edlimit** コマンドを参照してください。

edquota コマンドは、各ユーザーとグループの現在のディスク・クォータを含む一時ファイルを作成します。edquota コマンドは、**/etc/filesystems** ファイルから、設定された割り当てを使ってファイルシステムのリストを決定します。また、**edquota** コマンドは、割り当ての追加および変更ができるように、一時ファイルに対して vi エディター (または **EDITOR** 環境変数で指定したエディター) を起動します。

注**: EDITOR** 環境変数にエディターを指定する場合、そのエディターの絶対パス名を指定する必要があります。

割り当てはファイルシステムごとに別々に管理されます。ユーザーまたはグループの割り当てを作成または編集する場合、その割り当ては特定のファイルシステムに適用されます。割り当ての設定は、割り当てを使用する各ファイルシステムに対して行われなければなりません。

デフォルトの場合、または **-u** フラグと一緒に使用する場合は、**edquota** コマンドはコマンド・ラインの *UserName* パラメーターで指定した 1 人または複数のユーザーの割り当てを編集します。 **-g** フラグと一緒に使用する場合、**edquota** コマンドは *GroupName* パラメーターで指定した 1 つ以上のグループの割り当てを編集します。 **-p** フラグは、プロトタイプ・ユーザー (*UserName*) またはプロトタイプ・グループ (*Proto-GroupName*) を示します。-p フラグを指定すると、このコマンドで設定したクォータが指定したユーザーまたはグループ用に複製されます。

猶予期間 (デフォルトでは 1 週間) の間は、各ユーザーは設定されているソフト・リミットを超過することを許されます。猶予期間が過ぎると、ソフト・リミットは強化されてハード・リミットとなります。この猶予期間は、日単位、時間単位、分単位、秒単位で指定できます。値 0 はデフォルトの猶予期間が与えられたことを表し、値 1 秒は猶予期間が与えられないことを表します。 **-t** フラグは、猶予期間を変更します。

一時ファイルに表示されるフィールドは、次のとおりです。

項目	説明
Blocks in use	このユーザーまたはグループが使用している 1KB 単位のファイルシステム・ブロックの数。
Inodes in use	このユーザーまたはグループが現在使用しているファイルの数。
Block soft limit	通常の操作中にユーザーまたはグループが使用を許可される 1KB 単位のブロックの数。
Block hard limit	通常の操作中にユーザーまたはグループが使用を許可される 1KB 単位のブロックの総数。割り当て猶予期間中の一時保存も含まれます。
Inode soft limit	通常の操作中にユーザーまたはグループが作成を許可されるファイルの数。
Inode hard limit	ユーザーまたはグループが作成を許可されるファイルの総数。割り当て猶予期間中に作成される一時ファイルも含まれます。

注**:** ハード・リミットの値が 1 の場合は、割り当てが許可されないことを意味します。ソフト・リミットの値が 1 で、ハード・リミットの値が 0 の場合は割り当てが一時的にのみ許可されたことを表します。

エディターがある場合は、**edquota** コマンドは一時ファイルを読み込み、変更に合わせてバイナリーの割り当てファイルを変更します。

ハード・リミットおよびソフト・リミットは、全体で 1 KB 単位のブロックになるように指定する必要があります。

フラグ

項目	説明
-g	指定した 1 つ以上のグループの割り当てを編集します。
-p	**-u** フラグを一緒に指定すると、プロトタイプ・ユーザーに対して設定されたクォータが、指定したユーザー用に複製されます。 **-p** フラグを **-g** フラグと一緒に指定すると、プロトタイプ・グループに対して設定されたクォータが指定された各グループ用に複製されます。
-t	ソフト・リミットがハード・リミットになる前に、割り当ての制限を超えることのできる猶予期間を変更します。猶予期間のデフォルト値は 1 週間です。 **-u** フラグを使って起動すると、**/etc/filesystems** ファイルで指定したユーザー割り当てを持つすべてのファイルシステムに猶予期間が設定されます。 **-g** フラグを使って起動すると、**/etc/filesystems** ファイルで指定したグループ割り当てを持つすべてのファイルシステムに猶予期間が設定されます。 **注: edquota** コマンドを使用して猶予期間を変更した後、**quotaon** コマンドの後に **quotaoff** コマンドが実行されて **quota.user** および **quota.group** ファイルがリフレッシュされるまで、新しい猶予期間値は適用されません。既に旧猶予期間が経過しているユーザーが、新しい猶予期間を使用するためには、ファイルシステムの使用率を、ソフト・リミット未満のレベルに下げる必要があります。将来的に、このようなユーザーがそのソフト・リミットを超えた時に、新しい猶予期間が有効になります。
-u	1 人または複数のユーザーの割り当てを編集します。

注**:** ユーザーまたはグループの名前に指定されている値が数字だけで構成されている場合、その値はユーザーまたはグループの ID として扱われます。したがって、割り当ては、名前ではなく、ID に対して編集されます。

セキュリティー

項目	説明
アクセス制御:	このコマンドは、root ユーザーだけが実行できます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

ユーザー `davec` 用に設定された割り当てをプロトタイプとして使用して、ユーザー `sharl` 用割り当てを作成するには、次のように入力します。

```
edquota -u -p davec sharl
```

ファイル

項目	説明
quota.user	ユーザー・クォータを指定します。
quota.group	グループ・クォータを指定します。
/etc/filesystems	ファイルシステムの名前と位置が入っています。

関連情報:

quota コマンド

quotacheck コマンド

quotaon および quotaoff

ディスク・クォータ・システムの概要

---

efsenable コマンド

目的

システムで暗号化ファイルシステム (EFS) 機能を活動化します。

構文

efsenable -a [**-v**] [**-k** <*algo*>] [**-f** <*cipher*>] [**-m** <*mode*>] [**-u** <yes|no>] [**-e** <*algo*>] [**-d** *Basedn*]

efsenable -q

説明

efsenable コマンドは、システムで EFS 機能を活動化します。このコマンドは、EFS 管理の鍵ストア、ユーザーの鍵ストア、およびセキュリティー・グループの鍵ストアを作成します。鍵ストアとは、EFS セキュリティー情報を含む鍵リポジトリーです。EFS 管理の鍵ストアへのアクセス・キーは、新規作成のアク

ティブなユーザーの鍵ストアとセキュリティー・グループの鍵ストアに保管されます。**efsenable** コマンドは、**/var/efs** ディレクトリーを作成します。**/etc/security/user** ファイルと **/etc/security/group** ファイルは、新しい EFS 属性で更新されます。また、**efsenable** コマンドは **Config_Rules** ODM データベースを更新します。

注**:** このコマンドを正常に機能させるには、Crypto Library (CLiC) パッケージの **clic.rte** がシステムにインストールされていなければなりません。この EFS コマンドでは、Role Based Access Control (RBAC) がシステムで使用可能になっている必要があります。これはデフォルトの設定です。

注**:** Crypto Library (CLiC) のファイルセット **clic.rte.lib** は、**efsenable** の AIX リリース 6.1 TL3 およびそれ以降では最低限 4.6 でなければなりません。

フラグ

項目	説明
-a	システムで EFS 機能を活動化します。
-d *Basedn*	LDAP サーバーに基本識別名 (DN) ou=UsrKeystore、ou=GrpKeystore、ou=EfsCookies、および ou=AdmKeystore を設定して、鍵ストアのローカル・ディレクトリー構造とともに鍵ストア・エントリーを容易に作成します。このフラグと一緒に引数として渡される *Basedn* は、鍵ストアの基本識別名の *Basedn* として使用されます。
-v	詳細モード。
-k *algo*	キーのデフォルト・アルゴリズム。*algo* フラグは、次のいずれかの値になります。 • RSA_1024 (デフォルト) • RSA_2048 • RSA_4096
-f *cipher*	ファイルのデフォルト暗号。*cipher* フラグは、次のいずれかの値になります。 • AES_128_CBC (デフォルト) • AES_192_CBC • AES_256_CBC • AES_128_ECB • AES_192_ECB • AES_256_ECB
-m *mode*	鍵ストアのデフォルト・モード。*mode* フラグは、次のいずれかの値になります。 • admin (デフォルト) • guard
-u [yes|no]	ユーザーがモードを変更できるかどうかを指定します。デフォルト値は「yes」です。
-e *algo*	EFS 管理キーのアルゴリズム。可能な *algo* 値は、**-k** フラグのものと同じです。
-q	使用可能なアルゴリズムのリストを表示します。

終了状況

項目	説明
0	コマンドは正常に実行されました。
1	コマンドの実行中にエラーが発生しました。
2	コマンド・ラインで構文エラーが発生しました。

セキュリティー

項目	説明
アクセス制御:	このコマンドは、root ユーザー、または **aix.security.efs** 権限を持ち、セキュリティー・グループのメンバーであるユーザーのみが実行できます。

例

1. 使用可能なアルゴリズムを表示するには、次のように入力します。

   ```
   efsenable -q
   ```

2. デフォルト・パラメーターを指定して EFS を活動化するには、次のように入力します。

   ```
   efsenable –a
   ```

3. キーに対してデフォルト以外のアルゴリズム、およびファイルに対して暗号を指定して EFS を活動化するには、次のように入力します。

   ```
   efsenable –a –k RSA_4096 –f AES_256_CBC –e RSA_4096
   ```

4. LDAP サーバーにローカル・ディレクトリー構造とともに作成された基本 DN を使用して EFS を活動化するには、次のコマンドを入力します。

   ```
   efsenable –a –d cn=aixdata
   ```

ファイル

項目	説明
/etc/security/user	EFS 属性の更新が入っています。
/etc/security/group	EFS 属性の更新が入っています。
/var/efs/users/	ユーザーの鍵ストアのディレクトリーが入っています。
/var/efs/groups/	グループの鍵ストアのディレクトリーが入っています。
/var/efs/efs_admin/	EFS 管理の鍵ストアのディレクトリーが入っています。
/var/efs/efsenabled	EFS がシステムで使用可能であることを示します。

関連情報:

基本オペレーティング・システムの保護

---

efskeymgr コマンド

目的

暗号化ファイルシステム (EFS) の鍵 (鍵ストア) のためのユーザーとグループのリポジトリーを管理します。

構文

efskeymgr -?

efskeymgr -q

efskeymgr -V

efskeymgr [-L *load_module*]**-C** <*group*>

efskeymgr -P < *Open-SSH Public Key file* >

注**:** 公開鍵ファイルは **~/.ssh/ directory** ディレクトリーにあります。

efskeymgr [-L *load_module*] [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-v**

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] **-m**

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-o** *<cmd>*

efskeymgr [-L *load_module*] [**-d**] [**-c** *<cmd>*]

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-n**

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-r** *<mode>*

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-s** *<ks2>*

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-S** *<ks2>*

efskeymgr[-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-R** *<algo>*

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-D** *<fp>*

efskeymgr [-L *load_module*] [**-d**] [**-k** *<ks>*] [**-g**] [**-p** *<pw>*] **-e** *<file>*

説明

efskeymgr コマンドは、EFS に必要なすべての鍵管理操作専用です。EFS が **efsenable** コマンドを使用してシステムで使用可能になっていると、鍵ストア (公開鍵と秘密鍵のリポジトリー) が **/var/efs** ディレクトリーに作成されます。

ユーザー鍵ストアの初期パスワードは、ユーザーのログイン・パスワードです。グループ鍵ストアと admin 鍵ストアは、パスワードではなく、アクセス・キーで保護されます。アクセス・キーは、このグループに属するすべてのユーザー鍵ストア内に保管されます。

鍵ストアを開くと (ログイン時または **efskeymgr** コマンドで明示的に)、この鍵ストアに入っている秘密鍵がカーネルにプッシュされ、処理と関連付けられます。アクセス・キーが鍵ストア内で見つかった場合は、対応する鍵ストアも開き、鍵が自動的にそのカーネルにプッシュされます。

鍵ストアは、admin モードと guard モードの 2 つの管理モードをサポートします。

admin モード
: このモードに鍵ストアを設定すると、**aix.security.efs** RBAC 権限と admin 鍵ストアへのアクセス・キーを持つ EFS 管理者は、鍵ストアを開いて、パスワードのリセット、鍵の再生成、アクセス・キーの追加や削除などの管理を行うことができます。

guard モード
: このモードに鍵ストアを設定すると、EFS 管理者は鍵ストアにアクセスできません。このモードでは、鍵ストアへのパスワードを失うと、秘密鍵をリカバリーできません。

鍵ストアのパスワードがログイン・パスワードと同じ場合は、鍵ストアがログイン時に自動的に開き、その鍵がセッションで使用可能になります。**passwd** コマンドが使用され、既存のパスワードが指定された場合、鍵ストアのパスワードはログイン・パスワードと同期します。鍵ストアのパスワードが何らかの時点でログイン・パスワードと同期していない場合は、**efskeymgr** コマンドを使用して鍵ストアのパスワードを変更できます。パスワードが同期化されないと、ログイン時に鍵が自動的にセッションと関連付けられません。

次のコマンドは、*cmd* コマンドの実行に対してのみ、EFS 証明書を認可したり、削除したりします。*cmd* コマンドが戻ると、前のプロセスの証明書が復元されます。

efskeymgr –o <cmd> and efskeymgr –c <cmd>

秘密鍵が鍵ストアで再生成される場合、新規の秘密鍵が作成され、古い鍵には「deprecated」(非推奨) のマークが付きます。

注**:** 新規の鍵はカーネルにはプッシュされません。新規の鍵をファイル操作に使用できるようにするには、**efskeymgr** コマンドを使用するか、セッションを閉じてから開いて、鍵ストアを再度開く必要があります。
非推奨の鍵は、ファイルの暗号化解除に使用できますが、ファイルの暗号化には使用できません。非推奨の鍵は鍵ストアから除去できます。ただし、この場合、その古い鍵で暗号化されたファイルはすべてアクセス不能となります。

注**:** この EFS コマンドでは、Role Based Access Control (RBAC) がシステムで使用可能になっている必要があります。これはデフォルトの設定です。

遅延オペレーション

鍵ストアは、コマンドまたはアクションで直接変更できない場合があります。この場合は、特殊ファイルが鍵ストアのディレクトリーに作成され、鍵ストアが次回開く際に解析されます。この特殊ファイルは、Cookie と呼ばれます。admin モードの鍵ストアの場合、Cookie は鍵ストアが開く際に自動的に解析されます (ログイン時または **efskeymgr** コマンドの実行時)。guard モードの鍵ストアの場合、Cookie は自動的に解析されません。ユーザーはその承認をその鍵ストアの変更ごとに与える必要があります。セッションを開く際、1 つ以上のオペレーションが EFS 鍵ストアで保留中の場合はメッセージが表示されます。

- 秘密鍵を再生成する必要がある。
- **group**/*group1* 鍵ストアへのアクセスを認可される。

efskeymgr -v コマンドを実行して、保留中のオペレーションを処理する必要があります。

以下のアクションが可能です。

- Private key regeneration (秘密鍵の再生成)。この結果、新規の秘密鍵が生成され、古い秘密鍵に「deprecated」(非推奨) のマークが付きます。
- New access key (新規のアクセス・キー)。この Cookie を受け入れると、新規の鍵ストアへのアクセスを取得します (例えば、ユーザーが追加されたグループの鍵ストアなど)。
- Remove access key (アクセス・キーの削除)。この Cookie を受け入れると (アクセス・キーがグループから削除される場合など)、鍵ストアへのアクセスを失います。

注**:** **efskeymgr** コマンドを **-v** フラグなど鍵ストアを開くフラグで実行すると、各 Cookie で行いたい操作についてのプロンプトが出されます。選択項目は次のとおりです。

- Accept the cookie (Cookie の受け入れ): 鍵ストアが Cookie に応じて変更された後に、その Cookie が破棄されます。
- Postpone the cookie (Cookie の延期): 鍵ストアは変更されず、その Cookie は削除されません。次回アクションについてのプロンプトが出されます。
- Delete the cookie (Cookie の削除): 鍵ストアは変更されず、その Cookie が削除されます。**efskeymgr** コマンドを使用して、アクションを再実行する必要があります。

フラグ

項目	説明
一般フラグ:	
-d	詳細モード。
-g	鍵ストアが開く際に保留中のオペレーションを処理しません。
-k *ks*	オペレーションは、アクティブ・ユーザーの鍵ストアの代わりに *ks* 鍵ストアがターゲットになります。*ks* 値は次のようになります。 **user/<login>** ユーザー <login> 鍵ストア。 **group/<grpname>** グループ <grpname> 鍵ストア。 **admin/** EFS 管理鍵ストア。
-L *load_module*	鍵ストアの操作に使用するロード可能モジュールを指定します。
-p *pw*	鍵ストアを開くために使用するパスワード。これは **ps** コマンドなどを使用して他のユーザーが表示できてしまうため、このフラグの使用はお勧めしません。
-P *filename*	**~/.ssh/authorized_keys** ディレクトリーにある OpenSSH ファイルに格納されているすべてのキーの公開鍵 Cookie をプッシュします。
コマンドのフラグ (鍵ストア・ファイルへのアクセスなし):	
-?	コマンド・ヘルプと出口を表示します。
-q	鍵の再生成に対してサポートされるアルゴリズムのリストを表示します。
-V	カーネル内でアクティブなプロセスの証明書に関連付けられた鍵を表示します。
コマンド用フラグ (鍵ストアへの読み取り専用アクセス):	
-c <*cmd*>	カーネルからすべての鍵を削除し、*cmd* コマンドを実行します。*cmd* コマンドが終了すると、鍵は復元されます。
-m	鍵ストアで保留中のすべてのオペレーションをリストします。
-o <*cmd*>	鍵ストアを開いて鍵をプッシュし、*cmd* コマンドを実行します。*cmd* コマンドが終了すると、鍵は廃棄されます。
-v	鍵ストア・ファイルの内容を表示します。
コマンド用フラグ (鍵ストアへの読み取り/書き込みアクセス):	
-C <*group*>	*group* グループの鍵ストアを作成します。
-D <*fp*>	非推奨の秘密鍵を鍵ストアから削除します。*fp* 値はキーの指紋です。
-e <*file*>	鍵ストアをファイルにエクスポートします。ファイルは PKCS#12 形式でエンコードされ、鍵ストアからの公開鍵と秘密鍵を含みます。このファイルは、openssh などで使用できます。
-n	ユーザー鍵ストアの場合は、鍵ストア用の新規パスワードについてのプロンプトを出します。グループ鍵ストアの場合は、新規アクセス・キーを生成し、グループ・メンバーに送信します。admin 鍵ストアの場合は、新規アクセス・キーを生成します。その後で、鍵は **efskeymgr** コマンドを使用して EFS 管理者に送信されなければなりません。
-R <*algo*>	鍵ストアの秘密鍵を再生成します。*algo* パラメーターの有効な値については、**-q** フラグを参照してください。
-r <*mode*>	鍵ストアの管理モードを変更します。*mode* 値は次のようになります。 **admin** EFS 管理者は鍵ストアを管理できます。保留中のオペレーションは自動的に適用されます。 **guard** EFS 管理者は鍵ストアを管理できません。保留中のオペレーションについてのプロンプトがユーザーに対して出されます。
-S <*ks2*>	*ks2* アクセス・キーを鍵ストアから削除します。鍵ストアを次回開く際、*ks2* 秘密鍵は自動的にプッシュされません。
-s <*ks2*>	鍵ストアのアクセス・キーを *ks2* 鍵ストアに送信します。 *ks2* キーを次回開く際、鍵ストアの秘密鍵は自動的にロードされます。

終了状況

項目	説明
0	コマンドが正常に実行されました。
1	コマンドの実行中にエラーが発生しました。
2	コマンド・ラインで構文エラーが発生しました。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 自分の鍵ストアの内容を表示するには、次のように入力します。

   ```
   efskeymgr –v
   ```

2. アクティブ・シェルに関連付けられた鍵を表示するには、次のように入力します。

   ```
   efskeymgr -V
   ```

3. 自分の鍵ストアから秘密鍵を再生成するには、次のように入力します。

   ```
   efskeymgr –R RSA_1024
   ```

4. 非推奨の鍵を削除するには、次のように入力します。

   ```
   efskeymgr –D dbb62547:d6925088:45357fd3:54cddbba:27b255a9
   ```

5. グループ「students」のアクセス・キーをユーザー「joe」へ送信するには、次のように入力します。

   ```
   efskeymgr –k group/students –s user/joe
   ```

6. **~/.ssh/authorized_keys** ファイルにインストール済みの公開鍵が格納されているターゲットの鍵ストアに入っている、Open-SSH クライアント・ユーザーの Open-SSH 公開鍵 Cookie をプッシュするには、次のように入力します。

   ```
   efskeymgr  -P  ~/.ssh/authorized_keys
   ```

7. グループ鍵ストアを LDAP に直接作成するには、次のように入力します (構成されている場合)。

   ```
   efskeymgr -L LDAP -C staff
   ```

ファイル

項目	説明
/var/efs	すべての鍵ストアが入っています。
/etc/security/user	ユーザー鍵ストアの作成と管理のための EFS 属性が入っています。
/etc/security/group	グループ鍵ストアの作成のための EFS 属性が入っています。

関連資料:

347 ページの『efsenable コマンド』

356 ページの『efsmgr コマンド』

関連情報:

基本オペレーティング・システムの保護

efskstoldif コマンド

目的

ローカルで定義されている特定の EFS ユーザー鍵ストアまたはグループ鍵ストアを ldif フォーマットで **stdout** に出力します。

構文

efskstoldif -d *baseDN* [**-u** | **-g**] {ALL | *Name* [*Name*] ...}

説明

efskstoldif コマンドは、ローカルで定義されている EFS ユーザー鍵ストアまたはグループ鍵ストアのファイルからデータを読み取り、その結果を ldif フォーマットで **stdout** に出力します。ファイルにリダイレクトする場合は、**-b** フラグを指定した **ldapadd** コマンドまたは **ldif2db** コマンドを使用して結果を LDAP サーバーに追加できます。

efskstoldif コマンドは、**/etc/security/ldap/sectoldif.cfg** ファイルを読み取って、データのエクスポート先であるユーザー、グループ、および Cookie のサブツリーの名前を決定します。**efskstoldif** コマンドは、ファイルに定義されている USERKEYSTORE、GROUPKEYSTORE、EFSCOOKIES、および ADMINKEYSTORE のタイプにのみデータをエクスポートします。このファイルで指定される名前は、**–d** フラグで指定される基本識別名 (DN) の下にサブツリーを作成するために使用されます。詳しくは、「AIX Version 6.1 TL 4」の /etc/security/ldap/sectoldif.cfg ファイルを参照してください。

LDIF 出力生成では、ユーザーまたはグループの **efs_keystore_access** 属性または **efs_adminks_access** 属性は参照されません。値が「file」または「ldap」のいずれでも、LDIF フォーマットが生成されます。ユーザー鍵ストアまたはグループ鍵ストアのいずれでも、ldif フォーマットが生成されます。これらの鍵ストアの Cookie がある場合は、それらに対しても ldif 生成が行われます。

注**:** Cookie がファイルにある場合でも、それらに対して LDIF 生成が行われます。システム管理者は、必要に応じて LDAP とファイル上の鍵ストア・エントリーに整合性があるかどうかを確認する必要があります。

フラグ

項目	説明
-d *baseDN*	EFS 鍵ストアのデータを配置するための基本識別名 (DN) を指定します。
-g ALL*Names ...*	次の引数で指定されるグループに対して出力を生成するようにコマンドに指示します。
	ALL すべてのグループに対して出力を生成するように指定します。
	名前 単一のグループ名またはブランクで区切ったグループ名のリストを指定します。
-u ALL*Names ...*	次の引数で指定されるユーザーに対して出力を生成するようにコマンドに指示します。
	ALL すべてのユーザーに対して出力を生成するように指定します。
	名前 単一のユーザー名またはブランクで区切ったユーザー名のリストを指定します。

終了状況

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドは、root ユーザーのみに実行 (x) アクセス権限を与えます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権操作を実行できます。特権命令を実行できるのは特権ユーザーのみです。権限および特権について詳しくは、「セキュリティー」の『特権コマンド・データベース』を参照してください。このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

ファイル

項目	説明
/etc/security/user	ユーザー鍵ストアの作成と管理のための EFS 属性が入っています。
/etc/security/group	ユーザー鍵ストアの作成と管理のための EFS 属性が入っています。
/var/efs	すべての鍵ストアが入っています。

例

1. すべてのユーザー鍵ストアおよびグループ鍵ストアの内容を cn=aixdata の基本 DN を使用して ldif フォーマットにエクスポートするには、次のコマンドを入力します。

   ```
   efskstoldif –d cn=aixdata
   ```

2. すべてのユーザー鍵ストアの内容を cn=aixdata の基本 DN を使用して ldif フォーマットにエクスポートするには、次のコマンドを入力します。

   ```
   efskstoldif –d cn=aixdata –u ALL
   ```

3. すべてのグループ鍵ストアの内容を cn=aixdata の基本 DN を使用して ldif フォーマットにエクスポートするには、次のコマンドを入力します。

   ```
   efskstoldif –d cn=aixdata –g ALL
   ```

4. 選択したユーザー鍵ストアのみの内容を cn=aixdata の基本 DN を使用して ldif フォーマットにエクスポートするには、次のコマンドを入力します。

   ```
   efskstoldif –d cn=aixdata –u davis smith
   ```

5. 選択したグループ鍵ストアのみの内容を cn=aixdata の基本 DN を使用して ldif フォーマットにエクスポートするには、次のコマンドを入力します。

   ```
   efskstoldif –d cn=aixdata –g finance managers
   ```

関連情報:

mksecldap コマンド

sectoldif コマンド

/etc/security/ldap/sectoldif.cfg コマンド

基本オペレーティング・システムの保護

efsmgr コマンド

目的

暗号化ファイルシステム (EFS) に対するファイルの暗号化と暗号化解除を管理します。

構文

efsmgr -?

efsmgr -q [**-v**]

efsmgr -C <*cipher*> [**-v**]

efsmgr [**-c** <*file*>] **-e** <*file*> [**-v**]

efsmgr [**-c** <*cipher*>] [**-s**] **-E** <*dir*> [**-v**]

efsmgr [**-c** <*cipher*>] **-t** <*file*> [**-v**]

efsmgr [**-c** <*cipher*>] [**-s**] **-T** <*dir*> [**-v**]

efsmgr -d <*file*> [**-v**]

efsmgr [**-s**] **-D** <*dir*> [**-v**]

efsmgr -l <*file*> [**-v**]

efsmgr [**-s**] **-L** <dir> [**-v**]

efsmgr -a <*file*> [**-u** <*user*> | **-g** <*group*>] [**-v**]

efsmgr -r <*file*> [**-u** <*user*> | **-g** <*group*>] [**-v**]

説明

efsmgr コマンドは、EFS でのファイルの暗号化の管理専用です。暗号化ファイルは、EFS が使用可能な JFS2 ファイルシステムでのみ作成できます。システムでの EFS の使用可能化に関する詳細については、『**mkfs**』、『**chfs**』、『**crfs**』、および『**efsenable**』のコマンドを参照してください。

暗号化ファイルを作成する方法は 2 通りあります。1 つは次のコマンドを使用して明示的に作成する方法、もう 1 つはファイルが作成されるファイルシステムまたはディレクトリーに継承が設定されたときに暗黙的に作成する方法です。

```
efsmgr -e <file>
```

継承がディレクトリーに設定された場合、このディレクトリー内で作成された新規ファイルはすべてデフォルトで暗号化されます。ファイルの暗号化に使用される暗号は、継承された暗号です。新規ディレクトリーもまた同じ暗号を継承します。サブディレクトリーで継承が使用不可の場合、このサブディレクトリー内で作成された新規ファイルは暗号化されません。

継承がファイルシステムに設定された場合、このファイルシステム内で作成された新規ファイルはすべて継承された暗号を使用して暗号化されます。継承が異なる暗号を使用してディレクトリーとファイルシステムの両方で設定された場合、このディレクトリー内で作成された新規ファイルは、ディレクトリーから継承された暗号を使用して暗号化されます。

ディレクトリーまたはファイルシステムで継承を設定したり除去しても、既存のファイルには影響がありません。**efsmgr** コマンドは、ファイルの暗号化または暗号化解除には暗黙的に使用する必要があります。

暗号ファイルを作成する場合は、事前にファイル所有者の秘密鍵をこの処理にロードしなければなりません。暗号化ファイルへのアクセス権限は、鍵ストアを持つユーザーまたはグループに付与されます。鍵ストアとは、EFS セキュリティー情報を含む鍵リポジトリーです。ユーザーとグループのリポジトリーの管理に関する詳細については、**efskeymgr** コマンドを参照してください。

暗号ファイルが開くとき、任意アクセス制御 (DAC) とアクセス制御リスト (ACL) のファイル・アクセス権がチェックされます。アクセス権が付与されている場合、この処理用にカーネルにロードされた鍵から、ファイルの保護鍵のいずれかと一致する秘密鍵が検索されます。一致する鍵が見つかった場合は、ファイルの内容を読み取ることができます。見つからなかった場合は、アクセスが拒否されます。

注**:** この EFS コマンドでは、Role Based Access Control (RBAC) がシステムで使用可能になっている必要があります。これはデフォルトの設定です。

フラグ

項目	説明
-c *<cipher>*	この暗号は、継承された暗号またはデフォルトの暗号の代わりに使用します。有効な *cipher* 値については、**-q** コマンドを参照してください。
-g *<group>*	このグループは、EFS アクセス・リストに追加されるか、または EFS アクセス・リストから除去されます。*group* 値は、gid またはグループ名です。
-s	この操作のターゲットは、ディレクトリーではなくファイルシステムです。この場合、*dir* パラメーターは、EFS サポートのあるファイルシステムのマウント・ポイントでなければなりません。
-u *<user>*	このユーザーは、EFS アクセス・リストに追加されるか、または EFS アクセス・リストから除去されます。*user* 値は、uid またはログイン名です。
-v	詳細モード。
-?	コマンド・ヘルプと出口を表示します。
-a *<file>*	**-u** フラグと **-g** フラグで指定したユーザーとグループのリストに対して、指定したファイルへのアクセスを追加します。
-C *<cipher>*	ユーザーのデフォルト暗号を *cipher* 値に変更します。
-D *<dir>*	ディレクトリーで継承を除去します。ファイルシステム全体にコマンドを適用するには、**-s** フラグを追加する必要があります。
-d <file>	指定したファイルの暗号化を解除します。
-E *<dir>*	*dir* ディレクトリーに継承を設定します。ファイルシステム全体にコマンドを適用するには、**-s** フラグを追加する必要があります。
-e <file>	指定したファイルを暗号化します。
-L *<dir>*	指定したディレクトリーで継承された暗号を表示します。
-l *<file>*	指定したファイルの暗号化情報をリストします (暗号、ファイルの暗号化を解除できる鍵)。
-q	サポートされる暗号のリストを表示します。
-r *<file>*	**-u** フラグと **-g** フラグで指定したユーザーとグループのリストに対して、指定したファイルへのアクセスを取り消します。
-T *<dir>*	指定したディレクトリーで継承された暗号を変更します。ファイルシステム全体にコマンドを適用するには、**-s** フラグを追加する必要があります。
-t *<file>*	指定したファイルの暗号鍵をリフレッシュします。これはファイルの暗号の変更にも使用できます。

終了状況

項目	説明
0	コマンドは正常に実行されました。
1	コマンドの実行中にエラーが発生しました。
2	コマンド・ラインで構文エラーが発生しました。

例

1. **database.txt** ファイルを強力な暗号を使用して暗号化するには、次のように入力します。

   ```
   efsmgr –e database.txt –c AES_256_CBC
   ```

2. ファイルをオープンできる鍵のリストを表示するには、次のように入力します。

   ```
   efsmgr –l database.txt
   ```

3. ユーザー joe とグループ maintainers へのアクセスをファイルに追加するには、次のように入力します。

   ```
   efsmgr –a database.txt –u joe –g maintainers
   ```

4. ホーム・ディレクトリーのファイルシステムで継承を設定するには、次のように入力します。

   ```
   efsmgr –c AES_128_CBC –s –E /home
   ```

ファイル

項目	説明
/etc/security/user	ユーザー用のデフォルトの暗号属性が入っています。

関連資料:

347 ページの『efsenable コマンド』

349 ページの『efskeymgr コマンド』

関連情報:

mkfs コマンド

基本オペレーティング・システムの保護

---

egrep コマンド

目的

ファイルの内容をパターンで検索します。

構文

egrep [**-h**] [**-i**] [**-p[** *Separator*]] [**-s**] [**-u**] [**-v**] [**-w**] [**-x**] [**-y**] [[**-b**] [**-n**] | [**-c** | **-l** | **-q**]] { { **-e***Pattern* | **-f***StringFile* } ... | *Pattern* } [*File ...*]

説明

egrep コマンドは入力ファイル (デフォルトは標準入力) を検索して、*Pattern* パラメーターで指定したパターンに一致する行を探します。これらのパターンは、 **ed** コマンド内で使用する場合と同じように、完全な正規表現です (ただし、 ¥ (円記号) と ¥¥ (2 つの円記号) を除く)。また、**egrep** コマンドには次の規則が適用されます。

- + (正符号) が後に続く正規表現は、 1 回またはそれ以上の正規表現のオカレンスに一致します。
- ? (疑問符) が後に続く正規表現は、正規表現のゼロ回または 1 回のオカレンスと一致します。

- | (縦線) または改行文字で区切られた複数の正規表現は、いずれかの正規表現と一致する文字列に一致します。
- 正規表現は、( ) (小括弧) で囲んでグループ化することができます。

改行文字は、正規表現で一致させることはできません。

演算子の優先順位は、[、]、*、?、+、連結、|、改行文字の順です。

> 注**: egrep** コマンドは、**-E** フラグを指定した **grep** コマンドと同じです。ただし、エラー・メッセージと使用方法メッセージが異なり、**-s** フラグの機能も異なります。

複数の *File* パラメーターを指定した場合、 **egrep** コマンドは一致する行を含むファイルを表示します。シェルに対して特別な意味を持つ文字 ($、*、[、|、^、(、)、¥) は、 *Pattern* パラメーター内では引用符で囲まなければなりません。 *Pattern* パラメーターが単純な文字列でない場合、通常はパターン全体を単一引用符で囲まなければなりません。 `[a-z]` のような式では、負符号は該当する照合シーケンスから見た範囲を示します。照合シーケンスは文字範囲内で使用する等価クラスを定義できます。これは、高速の決定論的なアルゴリズムを使用しますが、このアルゴリズムは指数のスペースを必要とすることがあります。

注**:**

1. 各行は 2048 バイト以内に制限されています。
2. 現在、パラグラフ (**-p** フラグの指定) の長さは、5000 文字に限定されています。
3. 予測できない結果を生ずるので、**grep** コマンドをスペシャル・ファイル上で実行するのは避けてください。
4. 入力行には、NULL 文字を使用しないでください。
5. 入力ファイルの終わりには、改行文字を付けてください。
6. 複数のフラグを同時に指定できますが、フラグの中には他のフラグを指定変更してしまうものがあります。例えば、**-l** と **-n** を同時に指定すると、ファイル名だけが標準出力に書き出されます。

フラグ

項目	説明
-b	各行の前に、その行が見つかったブロック番号を付けます。このフラグを使用すると、ディスク・ブロック番号をコンテキストで見つけるときに便利です。 **-b** フラグは、標準入力またはパイプからの入力と一緒には使用できません。
-c	一致した行の数のみを表示します。
-e *Pattern*	パターン を指定します。これは、単純パターン と同じ働きをしますが、パターン が - (ハイフン) で始まる場合に便利です。
-f *StringFile*	文字列を含むファイルを指定します。
-h	複数ファイルの処理中に、ファイル名を抑止します。
-i	比較を行うときに、大文字と小文字の区別を無視します。
-l	一致している行のファイルの名前を表示します (1 回)。各ファイル名は、改行文字で区切られます。標準入力を検索した場合は、パス名 `"(StandardInput)"` が戻されます。
-n	各行の前に、ファイル内におけるその行の相対行番号を付けます。
-p[*Separator*]	一致する行を含むパラグラフ全体を表示します。パラグラフは、*Separator* パラメーターで指定された、パラグラフ・セパレーターで区切られます。パラグラフ・セパレーターは検索パターンと同じ書式のパターンです。パラグラフ・セパレーターを含んでいる行は、セパレーターとしてのみ使用され、出力には含まれません。デフォルトのパラグラフ・セパレーターはブランク行となります。
-q	行の一致に関係なく、すべての出力が標準出力に書き出されないようにします。入力行を選択すると、終了して状況 0 を戻します。
-s	エラー・メッセージだけを表示します。これは、状況を検査するのに便利です。
-u	出力をバッファーしません。
-v	指定されたパターンと一致する行を除くすべての行を表示します。

項目	説明
-w	ワードの検索を行います。
-x	指定したパターンに正確に一致し、余分の文字がない行を表示します。
-y	比較を行うときに、大文字と小文字の区別を無視します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	一致が見つかりました。
1	一致が見つかりませんでした。
>1	構文エラーが見つかったか、(一致が見つかったとしても) ファイルにアクセスできませんでした。

例

パターン・マッチング文字 +、?、|、(、) を含む拡張パターンを使用するには、次のように入力します。

```
egrep "¥(([A-z]+|[0-9]+)¥)" my.txt
```

これは、小括弧内の文字や小括弧内の数字を含む行を表示しますが、小括弧で囲まれた文字と数字の組み合わせは表示しません。 (y) や (783902) には一致しますが、 (alpha19c) には一致しません。

> 注**: egrep** コマンドを使用する際、 ¥ ((左括弧が後に続く円記号)、または ¥) (右括弧が後に続く円記号) はテキスト内の括弧に一致しますが、 ((左括弧) と) (右括弧) は、パターンの部分をグループ化する特別な文字です。**grep** コマンドを使用するときは、逆になります。

ファイル

項目	説明
/usr/bin/egrep	**egrep** コマンドへのハード・リンクが入っています。
/bin/egrep	**egrep** コマンドへのシンボリック・リンクを指定します。

関連資料:

739 ページの『grep コマンド』

関連情報:

awk コマンド

シェル・コマンド

ナショナル・ランゲージ・サポートの概要

---

eimadmin コマンド

目的

エンタープライズ ID マッピング (EIM) ドメインを管理します。

構文

eimadmin -a | **-p** | **-l** | **-m** | **-e -D** | **-R** | **-I** | **-A** | **-C [-s** *switch*] **[-v** *verboseLevel*] **[-c** *accessType*] **[-f** *accessUserType*] **[-g** *registryParent*] **[-i** *identifier*] **[-j** *otherIdentifier*] **[-k** *URI*] **[-n** *description*] **[-o** *information*] **[-q** *accessUser*] **[-r** *registryName*] **[-t** *associationType*] **[-u** *registryUser*] **[-x** *registryAlias*] **[-y** *registryType*] **[-z** *registryAliasType*] **[-d** *domainDN*] **[-h** *ldapHost*] **[-b** *bindDN*] **[-w** *bindPassword*] **[-K**

keyFile **[-P** *keyFilePassword*] **[-N** *certificateLabel*]] **[-S** *connectType*]

説明

eimadmin コマンドは、AIX System Services Shell ツールです。管理者はこれを使用して、EIM ドメインを定義することと、レジストリー、ID、および ID とレジストリー・ユーザー間の関連を用いてドメインの事前準備をすることができます。管理者はまた、**eimadmin** を使用してユーザー (および他の管理者) が EIM ドメインへアクセスできるようにしたり、EIM エンティティーをリストしたり除去したりします。

管理者は、**eimadmin** コマンドを次の 2 つの方法で使用します。

- **eimadmin** コマンドのコマンド・ライン・オプションを用いて情報を組み込む。
- **eimadmin** コマンドが参照する入力ファイルに情報を組み込む。

ファイルは手動でも、データベースからレコードをエクスポートすることによっても作成できます。管理者は、コマンド・ライン・オプションを組み合わせて指定することによって、ユーティリティーの処理を指示します。

eimadmin コマンドは、以下のアクションを実行します。

- オブジェクトの追加 (**-a**)
- オブジェクトのパージ (**-d**)
- オブジェクトのリスト (**-l**)
- オブジェクトに関連した属性の変更 (**-m**)
- 属性の消去 (**-e**)

上記のアクションは、以下のオブジェクトに関して実行されます。

- ドメイン (**-D**)
- レジストリー (**-R**)
- ID (**-I**)
- 関連 (**-A**)
- アクセス権限 (**-C**)

注**:**

1. 各 **eimadmin** コマンドは、1 つのアクションと 1 つのオブジェクト・タイプを含んでいなければなりません。オブジェクトとそれに関して実行するアクションによっては、EIM が追加のパラメーターを必要とすることがあります。
2. 一部のオプションは複数値属性用です。その場合は属性を複数回指定できます。その他のオプションは、単一値属性用なので、1 回だけしか属性を指定できません。 (単一値属性用のオプションを繰り返して指定した場合、**eimadmin** が処理するのは、コマンドの中にある最初の値だけです。) この規定を除いて、パラメーターを指定する順序はどのような順序でもかまいません。
3. **eimadmin** コマンドのパラメーターは、いくつかの方法で指定できます。
   - アクションとオブジェクトを連結して、埋め込まれたハイフンを省略する。`-aD`
   - 両方のハイフンを組み込み、2 つのオプションはスペースで区切る。`-a -D`

   したがって、次の例は 有効ではありません。なぜなら、両方のハイフンが組み込まれているのに、**-D** の前にスペースがないからです。`-a-D`

フラグ

eimadmin コマンドは、以下のアクション・フラグを受け取ります。

項目	説明
-a	オブジェクトを追加します。(オブジェクト定義とその属性を作成します。)
-e	属性を消去します。(単一値属性をクリアするか、複数値属性を除去します。)
-l	オブジェクトをリストします。(オブジェクト定義とその属性を検索します。)
-m	属性を変更します。(単一値属性を変更するか複数値属性を追加することによって、既存オブジェクトを変更します。)
-p	オブジェクトをパージします。(オブジェクト定義とその属性を除去します。)

eimadmin コマンドは、以下のオブジェクト・フラグを受け取ります。

項目	説明
-A	関連。これは EIM ドメイン内の ID とユーザー ID 間の関係です。
-C	アクセス権限。これは EIM 定義の LDAP アクセス制御グループです。
-D	ドメイン。これは ID、ユーザー・レジストリー、および ID とユーザー ID 間の関連で、LDAP ディレクトリーに保管されます。
-I	ID。これは個人または EIM ドメインに関与しているエンティティーの名前です。
-R	レジストリー。これはユーザー・レジストリーの名前です。関連はユーザー・レジストリー内の ID とユーザー ID 間で定義されます。

eimadmin は、以下のプロセス制御フラグを受け取ります。

項目	説明
-s *switch*	*switch* は、**eimadmin** コマンド機能が作動する方法に影響を及ぼす値を指定します。以下の値を指定できます。 **RMDEPS** ドメインまたはシステム・レジストリーを除去するときに、それに従属しているものも除去します。これにより、そのドメインで定義されているすべての ID とレジストリーが最初に除去されるので、ドメインの除去が容易になります。また、これにより、そのレジストリーで定義されているすべてのアプリケーション・レジストリーが最初に除去されるので、システム・レジストリーの除去が容易になります。 重要: **eimadmin** コマンドは、従属しているものを除去する前に、それらが存在していることを知らせないので、このスイッチは注意して使用してください。
-v *verboseLevel*	*verboseLevel* パラメーターは、**eimadmin** コマンドが表示するトレースの詳細レベルを制御する 1 から 10 の整数です。 (これは **eimadmin** ユーティリティーで問題を診断するためのものです。) デフォルト値は 0 で、トレース情報を取らないことを示しています。整数値 1 から 10 を指定して、トレース情報の量を少ないものから多いものにすることができます。ユーティリティーはこの値を検査して、そのレベル以下で定義されているトレース情報を表示します。以下のレベルは特定の情報を表示させます。 • 3 を指定すると EIM API 呼び出しパラメーターと戻り値を表示します。 • 6 を指定するとオプション値と入力ファイル・ラベルを表示します。 • 9 を指定するとユーティリティー・ルーチンのエントリー・ステートメントと終了ステートメントを表示します。

eimadmin コマンドは、以下のテーブルにリストされている必須属性フラグおよびオプション属性フラグを受け取ります。フラグ・オプションは、示されていない限り、単一値です。オプションを複数回指定すると、ユーティリティーは最初のものだけを処理します。

注:

1. これらの属性は、コマンド・オプションとしてまたは入力ファイルのフィールドとして指定できます。コマンド・オプションとして指定する場合は、ブランクを含む値は引用符 (") または (') で囲む必要があります。引用符は単一ワード値ではオプションです。引用符を付けないで複数ワード値を指定すると、結局、コマンド・ライン・オプションが切り捨てられます。最初のワードの後の値はすべて切り捨てられます。
2. 次の特殊文字は *registryName*、*registryParent*、または *identifier* では使用できません。

   , **=** **+** **<** **>** **#** **;** **¥** ***

項目	説明
-c *accessType*	EIM ドメイン内でユーザーがもつアクセス権限のスコープを指定します。 *accessType* は、以下の値の 1 つでなければなりません。 **ADMIN** 管理アクセスを指定します。 **REGISTRY** レジストリー・アクセスを指定します。**REGISTRY** を指定した場合は、レジストリー値 (**-r**) も指定する必要があります。レジストリー値は特定のレジストリー名であっても、すべてのレジストリーへのアクセスを示すアスタリスク (*) であってもかまいません。 **IDENTIFIER** ID アクセスを指定します。 **MAPPING** マッピング操作アクセスを指定します。
-f *accessUserType*	アクセス・ユーザー名のタイプを指定します。*accessUserType* は、以下のタイプの 1 つでなければなりません。 **DN** *accessUser* は識別名です。 **KERBEROS** *accessUser* は Kerberos ID です。
-g *registryParent*	システム・レジストリーの名前を指定します。アプリケーション・レジストリーはシステム・レジストリーのサブセットです。アプリケーション・レジストリーを追加する場合は、**-r** オプションおよび **-g** オプションを使用しなければなりません。 **-r** 値は、定義するアプリケーション・レジストリーです。**-g** オプションは、既に存在するシステム・レジストリーです。
-i *identifier*	固有 ID 名を指定します。例: `John Day`
-j *otherIdentifier*	非固有 ID 名を指定します。例: `John` **注:** このオプションを複数回指定して、複数の非固有 ID を割り当てることができます。
-k *URI*	レジストリーの Universal Resource Identifier (URI) を指定します (存在する場合)。
-n *description*	ドメイン、レジストリー、ID、または関連に関連付ける任意のテキスト (ユーザー提供) を指定します。 **注:** 関連付け先に対してだけユーザー説明を定義することができます。
-o *information*	ID または関連に関連付ける追加情報を指定します。 **注:** 関連付け先に対してだけユーザー情報を定義することができます。このオプションを複数回指定して、複数の情報断片を割り当てることができます。
-q *accessUser*	指定された *accessUserType* に応じて、EIM アクセスで用いるユーザーの識別名 (DN) または Kerberos ID を指定します。
-r *registryName*	レジストリーの名前を指定します。新規レジストリーを追加するとき、**-g** オプションも一緒に指定しない限り、 **eimadmin** は、そのレジストリーをシステム・レジストリーとして処理します。 **-g** オプションを指定すると、**eimadmin** は、そのレジストリーをアプリケーション・レジストリーとして処理します。

項目	説明
-t *associationType*	ID とレジストリー間の関係を指定します。*associationType* は、以下の 1 つでなければなりません。 **ADMIN** 管理目的用の ID にユーザー ID を関連付けることを指定します。 **SOURCE** そのユーザー ID が検索操作の (からの) ソースであることを示します。 **TARGET** そのユーザー ID が検索操作の (に対する) ターゲットであることを示します。 **注:** このオプションを複数回指定して、複数の関係を定義することができます。
-u *registryUser*	レジストリー内のユーザー定義のユーザー ID を指定します。
-x *registryAlias*	レジストリーの別の名前を指定します。複数の別名を割り当てるには、このオプションを複数回指定する必要があります。
-y *registryType*	レジストリーのタイプを指定します。**eimadmin** が認識する定義済みタイプには、以下のものが含まれます。 • RACF® • OS/400® • KERBEROS (大/小文字を無視する場合) • KERBEROSX (大/小文字を正確に識別する場合) • AIX • NDS • LDAP • PD (Policy Director) • WIN2K 以下の 2 つの正規化方式のうちの 1 つを用いて、固有の OID を連結することによって独自のタイプを作成することもできます。 • caseIgnore • caseExact
-z *registryAliasType*	レジストリー別名のタイプを指定します。独自の値を作ることも、以下の推奨値のうちの 1 つを使用することもできます。 • DNSHostName • KerberosRealm • IssuerDN • RootDN • TCPIPAddress • LdapDnsHostName **注:** コマンド・ライン・オプションのセットまたは単一入力データ・レコードに対して、**eimadmin** コマンドは *registryAliasType* の最初の指定だけを認識します。しかし、**eimadmin** コマンドは、複数のレジストリー別名を認識して、それらのすべてを単一の *registryAliasType* に関連付けます。

eimadmin コマンドは、以下の接続タイプ・フラグを受け取ります。

項目	説明
-b *bindDN*	LDAP との簡易バインドに使用する識別名を指定します。
-d *domainDN*	EIM ドメインの完全識別名 (DN) を指定します。 *domainDN* は、'ibm-eimDomainName=' で始まり以下のエレメントから構成されます。 *domainName* 作成する EIM ドメインの名前です。例: **MyDomain** 親の識別名 ディレクトリー情報ツリー階層内のあるエントリーのすぐ上のエントリーの識別名 (o=ibm,c=us など) です。次に例を示します。 `ibm-eimDomainName=MyDomain,o=ibm,c=us`
-h *ldapHost*	EIM データを制御している LDAP サーバーの URL とポートを指定します。フォーマットは次のとおりです。 `ldap://some.ldap.host:389` `ldaps://secure.ldap.host:636`
-K *keyFile*	SSL キー・データベース・ファイルの名前 (絶対パス名を含む) を指定します。ファイルが見つからない場合は、認証証明書が入っている RACF キー・リングの名前であると想定されます。この値は、セキュア LDAP ホスト (接頭部が ldaps://) との SSL 通信で必要になります。次に例を示します。 `/u/eimuser/ldap.kdb`
-N *certificateLabel*	キー・データベース・ファイルまたは RACF キー・リングから使用する証明書を指定します。このオプションを指定しないと、ファイルまたはリング内でデフォルトとマークされた証明書が使用されます。
-P *keyFilePassword*	キー・データベース・ファイル内の暗号化された情報にアクセスするために必要なパスワードを指定します。代わりに、file:// を用いて stash ファイルをプレフィックス変換することによって、このオプションで SSL パスワード stash ファイルを指定することもできます。次に例を示します。 `secret or file:///u/eimuser/ldapclient.sth` **注:** コマンド・ラインで、**-K** オプションにキー・データベース・ファイルの名前を指定して、**-P** オプションは指定しないと、**eimadmin** コマンドは、キー・ファイルのパスワードを要求するプロンプトを出します。
-S *connectType*	LDAP サーバーに対する認証方式を指定します。*connectType* は、以下の値の 1 つでなければなりません。 • **SIMPLE** (バインド DN とパスワード) • **CRAM-MD5** (バインド DN と保護パスワード) • **EXTERNAL** (デジタル証明書) • **GSSAPI** (Kerberos) 指定しないと、*connectType* のデフォルトは **SIMPLE** になります。接続タイプ **GSSAPI** では、デフォルトの Kerberos 信任状が使用されます。この信任状は、**eimadmin** を実行する前に、kinit などのサービスを使用して確立されている必要があります。 KINIT および関連情報については、「AIX Authentication Service Administration」を参照してください。
-w *bindPassword*	バインド DN と関連したパスワードを指定します。

ユーティリティーが必要とする接続情報には、EIM ドメイン (**-d**) とその制御サーバー (**-h**)、サーバーに対する認証 (バインド) に用いる ID (**-b**、**-w**、または **-K**、**-P**、**-N**)、および認証方式 (**-S**) が含まれます。

ドメイン (**-D**) 以外のオブジェクト・タイプについては、ドメイン、サーバー、およびバインド ID の指定はオプションです。これらが指定されないと、情報は RACF プロファイルから取得されます。

注**:** 接続情報のいずれかを指定する場合は、またその接続タイプで必要な値のフルセットを指定する必要があります。 1 つ以上の値 (すべてではない) を省略すると、エラーになります。次のテーブルは、**eimadmin** コマンドで指定するときの、接続タイプおよびホスト・タイプごとに必要な値とオプションの

値を示しています。

接続タイプ/ホスト・タイプ	必要な値	オプションの値
SIMPLE または CRAM-MD5/セキュア (ldaps://)	-d, -h, -b, -w, -K, -P	-N
SIMPLE または CRAM-MD5/非セキュア (ldap://)	-d, -h, -b, -w	
EXTERNAL/セキュア (ldaps://)	-d, -h, -K, -P, -S	-N
EXTERNAL/非セキュア (ldap://)	サポートされない	サポートされない
GSSAPI/セキュア (ldaps://)	-d, -h, -K, -P, -S	-N
GSSAPI/非セキュア (ldap://)	-d, -h, -S	

注:

1. 上記のテーブルには次の 2 つの例外があります。
   - 入力ファイルを用いて値を指定する場合は、ドメイン・オプション (**-d**) はドメイン機能に対しては必要ありません。
   - -K で RACF キー・リングを指定すると、SSL キー・データベース・ファイル・パスワードまたは stash ファイル (-P) は必要ありません。
2. **eimadmin** コマンドは、**-w** が必要であるのにコマンド・ラインでそれが指定されなかった場合は、簡易 BIND パスワードを要求するプロンプトを出して、**-P** が必要であるのにコマンド・ラインでそれが指定されなかった場合は、SSL キー・データベース・ファイル・パスワードを要求するプロンプトを出します。

以下のテーブルに、オブジェクト・タイプおよびアクション・ペアごとに必要なフラグとオプションのフラグが要約されています。ほとんどのオプションの値は、コマンド・ラインで指定する代わりに入力ファイルでも指定することができます。

オブジェクト・タイプ (アクション)	フラグ	コメント
D (**a**)	• 必要: **d**、**h** • オプション: **n**	ドメインを追加します。
D (**p**)	• 必要: **d**、**h** • オプション: **s**	ドメインを除去します。ドメインが空でない場合は、-s RMDEPS を組み込みます。
D (**l**)	• 必要: **d**、**h** • オプション: なし	ドメインをリストします。すべてのドメインをリストするには、-d* を指定します。
D (**m**)	• 必要: **d**、**h** • オプション: **n**	ドメイン属性を変更または追加します。
D (**e**)	• 必要: **d**、**h** • オプション: **n**	ドメイン属性を除去または消去します。
R (**a**)	• 必要: **r**、**y** • オプション: **g**、**k**、**n**、**x**、**z**	レジストリーを追加します。**-g** も一緒に指定されないと (この場合、**-r** 値は新規アプリケーション・レジストリーを示しています)、**-r** で指定された値は新規システム・レジストリーと想定されます。
R (**p**)	• 必要: **r** • オプション: **s**	レジストリーを除去します。
R (**l**)	• 必要: **r** • オプション: **y**	レジストリーをリストします。指定された **-r** 値検索フィルター (これにはワイルドカード * が含まれている可能性もあります) と一致するドメイン内のすべてのレジストリー・エントリーを戻します。

オブジェクト・タイプ (アクション)	フラグ	コメント
R (**m**)	• 必要: **r** • オプション: **k**、**n**、**x**、**z**	レジストリー属性 (レジストリー別名を含む) を変更または追加します。
R (**e**)	• 必要: **r** • オプション: **k**、**n**、**x**、**z**	レジストリー属性 (レジストリー別名を含む) を除去または消去します。
I (**a**)	• 必要: **i** • オプション: **j**、**n**、**o**	ID を追加します。
I (**p**)	• 必要: **i** • オプション: なし	ID を除去します。
I (**l**)	• 必要: **i** • オプション: なし	固有 ID 名別に ID をリストします。指定された **-i** 値検索フィルター (これにはワイルドカード * が含まれている可能性もあります) と一致するドメイン内のすべての ID エントリーを戻します。
I (**l**)	• 必要: **j** • オプション: なし	非固有 ID 名別に ID をリストします。指定された **-j** 値検索フィルター (これにはワイルドカード * が含まれている可能性もあります) と一致する非固有 ID をもったドメイン内のすべての ID エントリーを戻します。
I (**m**)	• 必要: **i** • オプション: **j**、**n**、**o**	ID 属性を変更または追加します。
I (**e**)	• 必要: **i** • オプション: **j**、**n**、**o**	ID 属性を除去または消去します。
A (**a**)	• 必要: **i**、**r**、**u**、**t** • オプション: **n**、**o**	関連を追加します。**-t** オプションを繰り返して、複数の関連タイプを追加することができます。 **-n** および **-o** フラグは TARGET 関連にだけ関連しています。
A (**p**)	• 必要: **i**、**r**、**u**、**t** • オプション: なし	関連を除去します。**-t** オプションを繰り返して、複数の関連タイプを除去することができます。
A (**l**)	• 必要: **i** • オプション: **t**	関連をリストします。指定された **-i** 固有 ID のドメイン内のすべての関連を戻します。特定の関連タイプに戻されるエントリーを制限するには、**-t** 値を指定します。
A (**m**)	• 必要: **r**、**u** • オプション: **n**、**o**	関連属性を変更または追加します。**-n** および **-o** フラグは TARGET 関連にだけ関連しています。
A (**e**)	• 必要: **r**、**u** • オプション: **n**、**o**	関連属性を除去または消去します。**-n** および **-o** フラグは TARGET 関連にだけ関連しています。
C (**a**)	• 必要: **c**、**q**、**f** • オプション: **r**	アクセスを追加します。アクセス・タイプ REGISTRY に対して、特定の **-r** レジストリー値、またはドメイン内のすべてのレジストリーへのアクセスを示すワイルドカード * を提供します。
C (**p**)	• 必要: **c**、**q**、**f** • オプション: **r**	アクセスを除去します。アクセス・タイプ REGISTRY に対して、特定の **-r** レジストリー値、またはドメイン内のすべてのレジストリーへのアクセスを示すワイルドカード * を提供します。
C (**l**)	• 必要: **c** • オプション: **r**	タイプ別にアクセスをリストします。アクセス・タイプ REGISTRY に対して、特定の **-r** レジストリー値、またはドメイン内のすべてのレジストリーへのアクセスを示すワイルドカード * を提供します。
C (**l**)	• 必要: **q**、**f** • オプション: なし	ユーザー別にアクセスをリストします。

終了状況

eimadmin コマンドは完了すると、以下の終了コードのうちの 1 つを戻します。

項目	説明
0	成功。
4	1 つ以上のエラーが起こったが、すべてのレコードが処理された (入力ファイルを指定した場合)。
8	入力ファイル (指定された場合) の最後に到達する前に、処理を停止させる重大エラーが発生した。

例

1. 単一ドメインをリストするには、次のように入力します。

   ```
   eimadmin -lD -h ldap://my.server -b "cn=EIM admin,o=MyCompany,c=US" -d "ibm-eimDomainName=My Employees,o=My Company,c=US"
   ```

   これは次の出力に似たものを戻します。

   ```
   domain name: My Employees
   domain DN: ibm-eimDomainName=My Employees,o=My Company,c=US
   description: employees in my company
   ```

2. 単一レジストリーをリストするには、次のように入力します。

   ```
   eimadmin -lR -r MyRegistry
   ```

   これは次の出力に似たものを戻します。

   ```
   registry: MyRegistry
   registry kind: APPLICATION
   registry parent: MySystemRegistry
   registry type: RACF
   description: my racf registry
   URI: ldap://some.big.host:389/profileType=User,cn=RACFA,o=My Company,c=US
   registry alias: TCPGROUP
   registry alias type: DNSHostName
   ```

3. ID をリストするには、次のように入力します。

   ```
   eimadmin -lI -i "J.C.Smith"
   ```

   これは次の出力に似たものを戻します。

   ```
   unique identifier: J.C.Smith
   other identifier: J.C.Smith
   other identifier: Joseph
   other identifier: Joe
   description: 004321
   information: D01
   information: 1990-04-11
   ```

4. 宛先関連をリストするには、次のように入力します。

   ```
   eimadmin -lA -i "J.C.Smith" -t target
   ```

   これは次の出力に似たものを戻します。

   ```
   unique identifier: J.C.Smith
   registry: MyRegistry
   registry type: RACF
   association: target
   registry user: SMITH
   description: TSO
   information: 1989-08-01
   information: ADMIN1
   ```

5. アクセスをリストするには、次のように入力します。

   ```
   eimadmin -lC -c admin
   ```

これは次の出力に似たものを戻します。

```
access user: cn=JoeUser,o=My Company,c=us
access user: cn=admin1,o=My Company,c=us
access user: cn=admin2,o=My Company,c=us
```

位置

/usr/bin/eimadmin

セキュリティー

LDAP 管理者には、**eimadmin** コマンドを使用して、それが提供するすべての機能にアクセスする権限があります。 EIM 管理者は、以下の条件が真である限りは、コマンドを使用することができます。

- EIM ドメインを収容している LDAP サーバーに定義された BIND 識別名およびパスワードをもっている。
- BIND 識別名が次の EIM 権限のうちの 1 つをもっている。
  - EIM 管理者
  - EIM レジストリー管理者
  - EIM レジストリー X 管理者
  - EIM ID 管理者

標準エラー

eimadmin コマンドは、パスワードを要求するため、またはエラーを示すためにメッセージを表示します。入力ファイルを使用していない場合は、正常終了のメッセージを受信することは期待しないでください。入力ファイル内のレコードを処理するとき、**eimadmin** は、50 レコードごとの進行中メッセージに加えて、処理の開始時と停止時に情報メッセージを表示します。

注**:** **eimadmin** コマンドは、リスト (-l) 要求に対して 1 つ以上のデータ行を戻します。ただし、一致する EIM エントリーがないことが分かったか、またはバインド ID がそのデータをアクセスすることを許可されていない場合を除きます。

関連情報:

eimadmin.conf コマンド

---

elogevent コマンド

目的

イベント応答リソース・マネージャー (ERRM) によって生成されたイベント情報を、指定されたログ・ファイルに記録します。

構文

elogevent [**-h**] *log_file*

説明

elogevent は、イベントが発生すると、イベント応答リソース・マネージャー (ERRM) から生成される環境変数で、ERRM によってポストされるイベント情報を収集します。このスクリプトは、イベント応答リ

ソースによって実行されるアクションとして使用できます。また、その他のユーザー定義アクションを作成するためのテンプレートとして使用することもできます。このスクリプトは常に、メッセージを英語で戻します。

ERRM 環境変数に関するイベント情報が戻されます。イベント情報には以下が含まれます。

Local Time
イベントまたはリアーム・イベントが監視される時刻。ERRM から提供される実際の環境変数は、ERRM_TIME です。この値は、現地時間に変換され、表示される前に読み取り可能なフォームに変換されます。

このスクリプトでは、**alog** コマンドを使用して、指定された *log_file* へイベント情報を書き込んだり、そこから読み取ったりします。

フラグ

-h スクリプトの使用方法に関するステートメントを標準出力に書き込みます。

パラメーター

log_file
イベント情報を記録するファイルの名前を指定します。*log_file* パラメーターには、絶対パスを指定する必要があります。

log_file は、循環ログとして処理され、サイズは 64KB の固定です。*log_file* がいっぱいになると、新しいエントリーによって既存の最も古いエントリーが上書きされます。

log_file が既に存在する場合、イベント情報はそのファイルに追加されます。*log_file* が存在しない場合は、ファイルが作成されて、イベント情報はそこに書き込まれます。

終了状況

0 スクリプトが正常に実行されました。

1 必須の *log_file* が指定されていません。

2 *log_file* のパスが無効です。

制限

- このスクリプトは、ERRM が実行されているノード上で実行する必要があります。
- このスクリプトを実行するユーザーは、イベント情報が記録される *log_file* に対して書き込み許可を持っている必要があります。

標準出力

-h フラグを指定すると、スクリプトの使用方法に関するステートメントが標準出力に書き込まれます。

例

1. 情報をログに記録するには、ログ・ファイルを `/tmp/event.log` として指定します。 ERRM が次のコマンドを実行します。

   ```
   /opt/rsct/bin/elogevent/tmp/event.log
   ```

   コマンドの実行時に、**/tmp/event.log** ファイルが存在している必要はありません。
2. **/tmp/event.log** ファイルの内容を確認するには、次のコマンドを実行します。

   ```
   alog -f /tmp/event.log -o
   ```

次の出力例には、**/var** ファイルシステム (ファイルシステム・リソース) の警告イベントが示されています。

```
===============================================================================
Event reported at Mon Mar 27 16:38:03 2007

Condition Name:                         /var space used
Severity:                               Warning
Event Type:                             Event
Expression:                             PercentTotUsed>90

Resource Name:                          /var
Resource Class Name:                    IBM.FileSystem
Data Type:                              CT_UINT32
Data Value:                             91
```

位置

/opt/rsct/bin/elogevent

---

emgr コマンド

目的

暫定修正マネージャーを始動して、システムの暫定修正のインストール、除去、リスト、および検査を行います。

構文

emgr -l [**-L** *Label* | **-n** *interimfixNumber* | **-u** *VUID*] [**-v**{**1**|**2**|**3**}] [**-X**] [**-a** *path*]

emgr -e *interimfixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-b** | **-k** | **-I**] [**-p**] [**-q**] [**-m**] [**-o**] [**-X**] [**-a** *path*]

emgr -i *interimfixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-C**] [**-p**] [**-q**] [**-X**] [**-a** *path*]

emgr -C -i *interimfixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-p**] [**-q**] [**-X**] [**-a** *path*]

emgr -C -L *Label* [**-p**] [**-q**] [**-X**]

emgr -r -L *Label* | **-n** *interimfixNumber* | **-u** *VUID* | **-f** *ListFile* [**-w** *Directory*] [**-a** *path*] [**-b** | **-k** | **-I**] [**-p**] [**-q**] [**-X**]

emgr -c [**-L** *Label* | **-n** *interimfixNumber* | **-u** *VUID* | **-f** *ListFile*] [**-w** *Directory*] [**-a** *path*] [**-v**{**1**|**2**|**3**}] [**-X**]

emgr -M | **-U** [**-L** *Label* | **-n** *interimfixNumber* | **-u** *VUID* | **-f** *ListFile*] [**-w** *Directory*] [**-a** *path*] [**-X**]

emgr -R *interimfixLabel* [**-w** *Directory*] [**-a** *path*] [**-X**]

emgr -P [*Package*] [**-a** *path*] [**-X**]

emgr -d -e *interimfixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-v**{**1**|**2**|**3**}]

説明

emgr (暫定修正マネージャー) コマンドを使用して、システム暫定修正をインストールおよび管理することができます。暫定修正マネージャーは、 **epkg** コマンドを使用して作成されたパッケージをインストールし、暫定修正情報を含むデータベースを保守します。**emgr** コマンドは以下の操作を実行します。

- 暫定修正パッケージのインストール
- 暫定修正の除去
- 暫定修正のリスト作成
- 暫定修正の検査
- 暫定修正のマウント
- 暫定修正のアンマウント
- パッケージのロックの表示
- インストールされた暫定修正の強制除去

注**:**

- 暫定修正マネージャーでロックされたファイルセットの更新 (**installp**、**install_all_updates**、または **smit update_all** コマンドを使用) を試行すると、ロックされたファイルセットを示す通知が表示されます。場合によっては、ファイルセットがインストールされなかった理由を示す通知がありません。**lslpp** コマンドを使用すると、IFIXLOCKED 状態にあるすべてのロックされたファイルセットが表示されます。
- アクティブなプロセスで使用されている暫定修正またはサービス更新で更新されたライブラリーまたは実行可能プログラムは、再始動しないとそのプロセスには反映されません。例えば、ksh を変更する更新では、既に実行されている ksh プロセスにはその変更が反映されません。同様に、**libc.a** ライブラリーへの更新は、既に実行されているプロセスには反映されません。また、ライブラリーの更新後にそのライブラリーと同一ライブラリーの **dlopen** 操作を使用するプロセスでは、再始動しないと不整合が発生する可能性があります。

Ifix の参照

暫定修正を参照する方法は、次のとおりです。

ラベルによる参照
: 特定のシステム上にインストールされている各暫定修正は、固有の暫定修正ラベルを保持しています。これは、すべての異なるデータベース・オブジェクトをバインドする固有なキーです。ラベルにより暫定修正を参照するには、パラメーターとしてそのラベルを **-L** フラグに渡します。例えば、ラベル **ABC123** で暫定修正に関する検査操作を実行するには、次のように入力します。

  ```
  emgr -cL ABC123
  ```

Ifix ID による参照
: 特定のシステム上にインストールされている各暫定修正は、暫定修正 ID を保持しています。暫定修正 ID は、暫定修正が暫定修正データベース内にリストされている順序番号に過ぎません。暫定修正のリスト作成に基づいて暫定修正に関する操作を実行する場合は、このオプションを使用する方が便利です。**emgr** コマンドは、特定の操作を実行する前に暫定修正 ID を暫定修正ラベルに変換します。ID により暫定修正を参照するには、パラメーターとして ID を **-n** フラグに渡します。

  注**:** 暫定修正が除去および追加されるたびに、Ifix ID が変更される可能性があります。特定の暫定修正またはすべての暫定修正をリストするには、**-l** フラグを使用して、現行の暫定修正 ID 番号を

常に確認してください。
例えば、ID が 1 の最初の暫定修正に関する検査操作を実行するには、次のように入力します。

```
emgr -cn1
```

VUID による参照

暫定修正パッケージはどのエンティティーによっても正式に追跡されないので、複数の暫定修正パッケージに対して同じ暫定修正ラベルを使用することができます。ただし、 **emgr** コマンドは、同じ暫定修正ラベルをもつ複数の暫定修正の同時インストールを受け入れません。VUID (仮想的固有 ID (Virtually Unique ID)) を使用して、同じ暫定修正ラベルをもつ複数のパッケージを区別することができます。 **emgr** コマンドは、特定の操作を実行する前に VUID を暫定修正ラベルに変換します。例えば、 VUID が **000775364C00020316020703** のインストール済み暫定修正をリストするには、次のように入力します。

```
emgr -l -u 000775364C00020316020703
```

注**:** VUID は、暫定修正のインストールおよび除去のプレビュー・フェーズで表示されます。VUID は、**-v** フラグを使用して詳細レベル 3 でリストする場合も表示されます。

Ifix ロギング

下記の操作は、**emgr** コマンド・ログ・ファイル **/var/adm/ras/emgr.log** に記録されます。

- インストール
- 除去
- 検査
- マウント
- アンマウント
- 強制除去

installp による暫定修正の自動除去

暫定修正は **epkg** コマンドで圧縮することができ、そこには APAR 参照番号を含む APAR 参照ファイルを入れることができます。APAR 参照番号を使用することにより、修正が出荷されたすべてのテクノロジー・レベルについて、**installp** は暫定修正から APAR にマップし直すことができます。適用されるテクノロジー・レベル、Service Pack、あるいは PTF にこの暫定修正が含まれていると **installp** が判断した場合には、**installp** は更新を適用するまえにこの暫定修正を自動的に除去します。

暫定修正が自動除去の対象となった場合、**emgr** コマンドはこの暫定修正のインストール時に以下のメッセージを表示します。

```
ATTENTION: Interim fix is enabled for automatic removal by installp.
```

並行更新

emgr コマンドは、並行更新と呼ばれる新しい形の暫定修正のインストールをサポートします。この形式の暫定修正には、AIX カーネルに対する変更、またはそのカーネル拡張のいずれかが含まれています。これはシステム・メモリーに直接適用することができ、システムをリブートする必要がありません。このようにシステム・メモリーに直接パッチを適用することで、ディスク上のシステムの現行カーネルを含むファイルを変更せずに、カーネルの変更を安全に評価してテストすることができます。システム・メモリーに適用された並行更新は、システムをリブートすると持続しません。ただし、**-C** フラグを使用してディスクに対する並行更新による変更をコミットする場合を除きます。並行更新は、同じモジュールの別のパッチに直接適用することができます。既存のパッチを除去する必要はありません。ただし、ロードするモジュールのバー

ジョンは 1 つのみでなければなりません。また、REBOOT_REQUIRED 状態では暫定修正の並行更新操作 (メモリー内またはディスク上) は、システムがリブートするまで実行できません。

emgr コマンドは、NIM シン・サーバー (ディスクレス・クライアントまたはデータレス・クライアント) でのメモリー内並行更新の適用をサポートしています。各シン・サーバーはオペレーティング・システム・ファイルを他のクライアントと共有します (**/usr** ディレクトリーは読み取り専用である)。このため、ディスクへの並行更新をコミットする **emgr** オプション (**-C** フラグ) は、シン・サーバーでは無効です。

注**:** シン・サーバーの共有オペレーティング・システム・ファイルのパッチがディスクに適用される必要がある場合、暫定修正はシン・サーバー用の NIM マスター上の SPOT リソースに適用されることがあります。暫定修正の SPOT へのインストールの詳細については、「Installation Guide」の『Installing an Interim Fix into a SPOT resource』セクション、または、ご使用の NIM マスター上の **/usr/lpp/bos.sysmgt/nim/README** ファイル (NIM IFIX/EMGR セクション) を参照してください。

シン・サーバーでは **/usr** ファイルシステムが読み取り専用であるため、**emgr** データベースは **/var/emgrdata** ディレクトリーに配置されることになります。

ある種の **emgr** 操作 (bosboot およびファイルシステム拡張など) は、シン・サーバー環境ではサポートできません。その結果、**-C**、**-e**、**-I**、**-k**、および **–X** の各 **emgr** フラグは、シン・サーバー環境ではサポートされません。また、シン・サーバーの場合は bosboot 操作をサポートできないため、 **–b** フラグ (リブートを必要とする暫定修正に対応した bosboot プロセスをスキップ) は、利用可能なときはいつも使用されることになります。

フラグ

項目	説明
-a *path*	インストールのための代替ディレクトリー・パスを指定します。 **注: -a** フラグは、**emgr** コマンドの **-e** フラグと **-a** フラグが暫定修正のインストールで使用された場合に限り、暫定修正の除去で機能します。**-e** フラグと **-a** フラグを使用して暫定修正がインストールされなかった場合、**emgr** コマンドは代替ディレクトリー・パスから暫定修正を完全に除去しません。 次善策として、次のコマンドを使用して、代替ディレクトリーにインストールされた暫定修正を除去してください。 `chroot /alt_inst /usr/sbin/emgr -r -L efix_label`
-b	**emgr** コマンドが、リブートを必要とする暫定修正に対して、通常の AIX **bosboot** プロセスをスキップするようにします。
-c	検査操作を指定します。**emgr** コマンドが、指定された暫定修正 (複数の場合もある) に関する検査操作を実行するようにします。
-C	ディスクに対する並行更新を含む暫定修正をコミットします。このオプションは、**-i** オプションと一緒に使用する必要があります。あるいは、暫定修正が **-i** オプションと一緒に適用された後に使用できます。これにより、並行更新がシステムのリブート後にも持続します。 並行更新がコミットされると、モジュールに他のパッチがあるかどうかにかかわらず、除去することでモジュールがパッチを適用していない元の状態に復元されます。モジュールの既存のパッチはコミットの実行時にすべて除去されます。
-d	目次およびトポロジーを表示します。このオプションを **-v** フラグと併用すると、詳細出力の表示に役立ちます。
-e *interimfixPackage*	暫定修正パッケージ・ファイルのパスを指定し、暫定修正パッケージをインストールします。暫定修正パッケージ・ファイルは、**epkg** コマンドを使用して作成し、かつ 16 ビットの圧縮拡張子 **.Z** で終わる必要があります。
-f *ListFile*	以下のいずれかを含むファイルを指定します。 • インストール操作に関するパッケージのロケーションのリスト (1 行に 1 つずつ) • 除去、マウント、アンマウント、および検査の各操作に関する暫定修正ラベルのリスト (1 行に 1 つずつ) **emgr** コマンドは、ブランク行または最初の非空白文字が **#** 文字の行をすべて無視します。

項目	説明
-i *interimfixPackage*	並行更新を含む暫定修正のパッケージ・ファイルのパスを指定し、その並行更新をシステム・メモリーに適用します。この更新は、**-C** フラグを使用しない限り、システムのリブート後は持続しません。 **-i** フラグを使用して、並行更新を同じモジュールの別の更新に適用することもできます。このような並行更新を「フォローオン」(後続の並行更新) と呼びます。
-I	**bosboot** コマンドの **-I** フラグを使用して AIX **bosboot** の低レベル・デバッガーを実行します。
-k	**bosboot** コマンドの **-D** フラグを使用して AIX **bosboot** 中に低レベル・デバッガーをロードします。
-l	**emgr** コマンドが、指定された暫定修正 (複数の場合もある) に関するリスト操作を実行するようにします。
-L *Label*	暫定修正ラベルによるこの操作のための暫定修正を選択します。
-m	**emgr** コマンドがマウント・インストールを実行するようにします。暫定修正がマウント・インストールされると、暫定修正ファイルはターゲット・ファイルの上にマウントされます。
-M	**emgr** コマンドが、**-m** フラグを使用してマウント・インストールされた暫定修正 (複数の場合もある) をマウントするようにします。**-M** フラグは、**-m** フラグを使用してインストールされ、 **-U** フラグまたは他の方法 (例えば、システムのリブート) でアンマウントされた暫定修正をマウントするためのものです。
-n *interimfixID*	暫定修正 ID を指定してこの操作のための暫定修正を選択します。
-o	暫定修正のインストールにより既存のパッケージを上書きできることを指定します。
-p	**emgr** コマンドがインストールまたは除去のプレビューを実行するようにします。プレビューはすべての検査操作で実行されますが、変更は行われません。
-P [*Package*]	パッケージ表示操作を指定します。この操作では、暫定修正マネージャーによってロックされているすべてのパッケージ、パッケージのインストーラー、およびロック・ラベル (複数の場合もある) が表示されます。
-q	エラーおよび重大な警告以外のすべての出力を抑止します。
-r	**emgr** コマンドが、指定された暫定修正 (複数の場合もある) に関する除去操作を実行するようにします。 アクティブなパッチを除去すると、モジュールに既存のパッチがある場合はそのパッチが復元されます。既存のパッチがない場合は、モジュールがパッチを適用していない元の状態に復元されます。
-R *Label*	**emgr** コマンドが強制除去操作を実行するようにします。このオプションは、暫定修正ファイルの除去、除去スクリプトの実行、またはブート処理を実際に行わずに、暫定修正ラベルに関連付けられている暫定修正データおよびパッケージ・ロックを除去します。このオプションを使用できるのは、1 回に 1 つの暫定修正に限られます。暫定修正ラベルは、ターゲット暫定修正の識別に必要です。 重要: • 暫定修正除去のこの方法は、緊急プロシージャーとみなしてください。この方法ではターゲット・システム上に不整合が生じる可能性があるので、暫定修正除去の他のすべての方法が失敗した場合にのみ、この強制除去方法を使用してください。 • インストールされた暫定修正を除去するには、標準的な除去プロセス (**-r** flag) を使用する必要があります。緊急時の手順では、**-R** フラグを使用してラベルを強制除去することができます。 **-R** フラグでは、ラベルを除去するために **-F** フラグが必要です。 **-R** フラグで **-F** フラグを指定する (例: `emgr -FR ifix_label`) と、強制除去オプションによって暫定修正ファイル、保存されたデータ、除去実行スクリプトのいずれも削除されることはありません。このオプションは、標準の除去プロセスが実行できない場合にのみ使用してください。
-u *VUID*	VUID を指定してこの操作のための暫定修正を選択します。
-U	**emgr** コマンドが、**-m** フラグを使用してマウント・インストールされた暫定修正 (複数の場合もある) をアンマウントするようにします。
-v{**1**|**2**|**3**}	リスト操作の詳細レベルまたは検査操作の検査レベルを指定します。有効なレベルは 1、2、および 3 です。
-w *Directory*	**emgr** コマンドがデフォルトの **/tmp** ディレクトリーの代わりに指定の作業ディレクトリーを使用するようにします。
-X	要求された **emgr** 操作を実行するのに十分なスペースがないファイルシステムの拡張を試みます。このオプションは、暫定修正パッケージおよび **emgr** コマンドで提供される使用可能なスペースおよびサイズ見積もりに基づいてファイルシステムを拡張します。 **注:** 1. **-X** フラグが使用される場合でも、インストール時に使用可能なディスク・スペースが使い果たされる可能性があります。インストール時に他のファイルが同じファイルシステム内で作成または拡張されている場合は、この可能性が高くなります。 2. リモート・ファイルシステムは、 **emgr** コマンドを使用して拡張することはできません。

終了状況

0 すべての **emgr** コマンド操作が正常に完了しました。

>0 エラーが発生しました。

セキュリティー

aix.system.install 権限を持つシステム管理者またはユーザーは、マルチレベル・セキュア (MLS) システムで **emgr** コマンドを実行できます。Ifix データ、保存ファイル、および一時ファイルは、root ユーザーのみがアクセス可能です。

emgr コマンドは、システムから、サポートされる MD5 生成コマンドを探します。見つかると、**emgr** コマンドは MD5 チェックサムをユーザーに表示します。ユーザーは、この MD5 合計を、保護されたソースとクロスチェックすることができます。 MD5 生成コマンドが見つからない場合は、**emgr** コマンドはその後の処置をとりません。

ユーザーは、EMGR_MD5_CMD シェル変数をエクスポートすることによって MD5 コマンドへのパスを強制的に設定できます。この変数には、MD5 生成コマンドへの絶対パスが入っている必要があります。

注**:** この機能は、暫定修正管理のオリジナル・リリースではサポートされません。**bos.rte.install** を最新レベルに更新することにより、暫定修正管理を最新レベルに更新するようにお勧めします。

例

1. **games.020303.epkg.Z** と呼ばれる暫定修正パッケージのインストールをプレビューするには、次のように入力します。

   ```
   emgr -p -e games.020303.epkg.Z
   ```

2. **games.020303.epkg.Z** と呼ばれる暫定修正パッケージをインストールし、追加のスペースが必要な場合にファイルシステムを自動的に拡張するには、次のように入力します。

   ```
   emgr -X -e games.020303.epkg.Z
   ```

3. システム上のすべての暫定修正をリストするには、次のように入力します。

   ```
   emgr -l
   ```

4. レベル 3 で暫定修正ラベル **games** のリストを作成するには、次のように入力します。

   ```
   emgr -lv3 -L games
   ```

5. ラベル **games** をもつ暫定修正を除去するには、次のように入力します。

   ```
   emgr -r -L games
   ```

6. ファイル **/tmp/myfixes** 内の暫定修正ラベルの除去をプレビューするには、次のように入力します。

   ```
   emgr -rp -f /tmp/myfixes
   ```

7. 検査レベル 2 ですべての暫定修正を検査するには、次のように入力します。

   ```
   emgr -cv2
   ```

8. 検査レベル 1 (デフォルトの検査レベル) で暫定修正 ID 番号 3 を検査するには、次のように入力します。

   ```
   emgr -c -n3
   ```

9. VUID が **000775364C00020316020703** の暫定修正を検査レベル 3 で検査するには、次のように入力します。

   ```
   emgr -u 000775364C00020316020703 -c -v3
   ```

10. すべてのロック・パッケージおよびその暫定修正ラベルをリストするには、次のように入力します。

```
emgr -P
```

11. **installp** パッケージ **bos.rte.lvm** をロックしているすべての暫定修正ラベルをリストするには、次のように入力します。

    ```
    emgr -P bos.rte.lvm
    ```

12. **games.020303.epkg.Z** と呼ばれる暫定修正パッケージをマウント・インストールし、 AIX **bosboot** を抑止するには、次のように入力します。

    ```
    emgr -e games.020303.epkg.Z -mb
    ```

13. **-m** オプションを使用してシステム上にマウント・インストールされたすべての暫定修正ファイルをマウントするには、次のように入力します。

    ```
    emgr -M
    ```

14. 暫定修正ラベル **games** に関連付けられているすべての暫定修正ファイルをアンマウントするには、次のように入力します。

    ```
    emgr -U -L games
    ```

15. システム・メモリーに対する並行更新を含む **kernelmod.031007.epkg.Z** と呼ばれる暫定修正のパッケージを適用するには、次のように入力します。

    ```
    emgr -i kernelmod.031007.epkg.Z
    ```

16. 暫定修正のラベル **kernelmod** に関連付けられた、ディスクに対する並行更新をコミットするには、次のように入力します。

    ```
    emgr -C -L kernelmod
    ```

17. システム・メモリーに対する並行更新を含む **kernelmod2.031007.epkg.Z** と呼ばれる暫定修正のパッケージを適用し、さらにディスクに対する並行更新をコミットするには、次のように入力します。

    ```
    emgr -i kernelmod2.031007.epkg.Z -C
    ```

18. 暫定修正パッケージ **test.102403.epkg.Z** についてレベル 3 の詳細出力を表示するには、次のように入力します。

    ```
    emgr -v3 -d test.102403.epkg.Z
    ```

ファイル

項目	説明
/usr/sbin/emgr	**emgr** コマンドが入っています。
/usr/emgrdata/DBS/ifix.db	暫定修正ヘッダー・データベースが入っています。
/usr/emgrdata/DBS/files.db	暫定修正ファイル・データベースが入っています。
/usr/emgrdata/DBS/pkglck.db	パッケージ・ロック・データベースが入っています。
/usr/emgrdata/DBS/prereq.db	前提条件データベースが入っています。
/usr/emgrdata/DBS/e2eprereq.db	暫定修正前提条件データベースが入っています。
/usr/emgrdata/DBS/aparref.db	APAR 参照ファイル・データベースが入っています。

関連資料:

409 ページの『epkg コマンド』

関連情報:

bosboot コマンド

オプションのソフトウェア・プロダクトおよび保守更新のインストール

emstat コマンド

目的

エミュレーション例外割り込みの統計情報を表示します。

構文

emstat [**-a** | **-v**] [*Interval*] [*Count*]

説明

emstat コマンドは、エミュレーション例外割り込みの統計情報を表示します。既存のアプリケーションまたはライブラリー (旧式のプロセッサー・アーキテクチャーから削除された命令を含む) を新しいプロセッサー上で実行すると、エミュレーション例外が発生することがあります。この種の命令は、無効な命令プログラム例外の原因となる場合があるからです。 オペレーティング・システムはこのような例外割り込みをキャッチし、古い命令をエミュレートして、プログラムの機能を保守しますが、そのために、プログラムのパフォーマンスが低下することがあります。

マシンを最後にリブートしてからのエミュレーション例外割り込みのカウントと、現在のインターバルでのカウントが表示されます。ユーザーは、オプションで、位置合わせ例外割り込みの統計情報を表示したり、個々のプロセッサー・エミュレーション統計情報を表示したりできます。

デフォルト出力では、毎秒ごとの統計情報が表示されます。サンプリングの間隔と反復の回数を、 指定することもできます。

パラメーター

項目	説明
Interval	サンプルからサンプルまでの間隔。
Count	反復の回数。

フラグ

項目	説明
-a	位置合わせ例外割り込みの統計情報を表示します。このフラグは、**-v** フラグと一緒には使用できません。
-v	個々のプロセッサーの統計情報を表示します。このフラグは、**-a** フラグと一緒には使用できません。

例

1. エミュレーションの統計情報を毎秒ごとに表示するには、次のように入力します。

   ```
   emstat
   ```

   これにより次の出力が生成されます。

   ```
     Emulation  Emulation
     SinceBoot      Delta
       8845591          0
       8845591          0
       8845591          0
       8845591          0
       8845591          0
       8845591          0
     ...
   ```

2. エミュレーションと位置合わせ例外割り込みの統計情報を 2 秒ごとに、合計 5 回表示するには、次のように入力します。

```
emstat -a 2 5
```

このコマンドにより次の出力が生成されます。

```
  Alignment  Alignment  Emulation  Emulation
  SinceBoot      Delta  SinceBoot      Delta
   21260604          0   70091846          0
   23423104    2162500   72193861    2102015
   25609796    2186692   74292759    2098898
   27772897    2163101   76392234    2099475
   29958509    2185612   78490284    2098050
```

3. エミュレーションの統計情報を、個々のプロセッサーにつき 5 秒ごとに表示するには、次のように入力します。

```
emstat -v 5
```

これにより次の出力が生成されます。

```
  Emulation  Emulation  Emulation  Emulation
  SinceBoot      Delta    Delta00    Delta01
   88406295          0          0          0
   93697825    5291530          0    5291530
   98930330    5232505    5232505          0
  102595591    3665261     232697    3432564
  102595591          0          0          0
```

関連情報:

alstat コマンド

---

emsvcsctrl コマンド

目的

イベント管理サブシステムを始動します。

構文

emsvcsctrl [-a | -s | -k | -d | -c | -t | -o | -h]

説明

emsvcsctrl は、イベント管理サブシステムを始動する制御スクリプトです。イベント管理は RSCT の分散サブシステムで、 IBM RS/6000® サーバーに関する一連のハイ・アベイラビリティー・サービスを提供します。システム・リソースの状態の情報と、クライアント・プログラムにとって重要なリソースの状態に関する情報とを突き合わせて、イベントを作成します。クライアント・プログラムでは、イベントを使用して、システム障害を検出し、障害から回復できるので、システムの可用性が高まります。emsvcsctrl 制御スクリプトでは、イベント管理サブシステムの操作を制御します。このサブシステムは、システム・リソース・コントローラー (SRC) の制御下にあり、emsvcs と呼ばれるサブシステム・グループに属します。各サブシステムには、デーモンが関連付けられています。また、emsvcsctrl スクリプトは、AIX リソース・モニター・サブシステムの操作も制御します。このサブシステムは、SRC の制御下にあり、emsvcs サブシステム・グループに属します。各サブシステムには、デーモンが関連付けられています。

イベント管理と AIX リソース・モニター・サブシステムのインスタンスは、HACMP/ES クラスター内の各ノード上で実行されます。操作の観点からすると、イベント管理サブシステム・グループは次のように編成されています。

サブシステム
: イベント管理

サブシステム・グループ
: emsvcs

SRC サブシステム
: emsvcs サブシステムは、haemd デーモンに関連付けられています。

emaixos
: emaixos は、harmad デーモンに関連付けられています。

デーモン
: haemd デーモンは、イベント管理サービスを提供します。harmad デーモンは、AIX オペレーティング・システム・リソースのリソース・モニターです。

emsvcsctrl スクリプトは通常、コマンド・ラインから実行されるとは限りません。通常は、システムのインストール時に、HACMP/ES 始動スクリプト・コマンドによって呼び出されます。

emsvcsctrl スクリプトは、イベント管理サブシステムの操作に対する各種制御を提供します。

- サブシステムの追加、始動、停止、および削除
- サブシステムのクリーンアップ
- トレースのオンおよびオフ

サブシステムの追加**:** -a フラグを指定すると、制御スクリプトが mkssys コマンドを使用して、イベント管理と AIX リソース・モニター・サブシステムを SRC に追加します。制御スクリプトは次のように動作します。

1. emsvcs および emaixos サブシステムが停止していることを確認します。
2. SRC から emsvcs および emaixos サブシステムを (まだ存在している場合に) 除去します。
3. emsvcs サブシステムを SRC へ追加します。
4. emaixos サブシステムを SRC へ追加します。
5. mkgroup コマンドを使用して haemrm グループを追加します (ただし、まだ存在していない場合のみ)。発生したエラーはすべて /var/ha/log/em.mkgroup という名前のログ・ファイルに書き込まれます。
6. /var/ha/lck/haem および /var/ha/soc/haem ディレクトリーを作成します (ただし、まだ存在していない場合のみ)。発生したエラーはすべて /var/ha/log/em.mkdir という名前のログ・ファイルに書き込まれます。
7. イベント管理構成データベース (EMCDB) をインストール場所である /opt/rsct/install/config/em.HACMP.cdb から、その実行場所である /etc/ha/cfg/em.HACMP.cdb へコピーします。コピーによってエラーが発生すると、そのエラーはログ・ファイル /var/ha/log/em.cp へ書き込まれます。

サブシステムの始動**:** -s フラグを指定すると、制御スクリプトが startsrc コマンドを使用して、イベント管理サブシステム emsvcs、および AIX リソース・モニター・サブシステム emaixos を始動します。

サブシステムの停止**:** -k フラグを指定すると、制御スクリプトが stopsrc コマンドを使用して、イベント管理サブシステム emsvcs、および AIX リソース・モニター・サブシステム emaixos を停止します。

サブシステムの削除**:** -d フラグを指定すると、制御スクリプトが rmssys コマンドを使用して、イベント管理サブシステムと AIX リソース・モニター・サブシステムを SRC から除去します。制御スクリプトは次のように動作します。

1. emsvcs および emaixos サブシステムが停止していることを確認します。
2. rmssys コマンドを使用して、SRC から emsvcs および emaixos サブシステムを除去します。

サブシステムのクリーンアップ**:** -c フラグを指定すると、制御スクリプトがすべてのシステム区画のイベント管理サブシステムを停止し、SRC から除去します。制御スクリプトは次のように動作します。

1. stopsrc -g emsvcs コマンドを使用して、サブシステム・グループ内のサブシステムのすべてのインスタンスを停止します。
2. rmssys コマンドを使用して、サブシステム・グループ内のサブシステムのすべてのインスタンスを SRC から除去します。
3. イベント管理構成データベース (EMCDB) を実行時位置である /etc/ha/cfg/em.HACMP.cdb から除去します。

トレース機能をオンにする**:** -t フラグを指定すると、制御スクリプトが haemtrcon コマンドを使用して、haemd デーモンのトレース機能をオンにします。また、traceson コマンドを使用して、harmad デーモンのトレースも使用可能にします。

トレース機能をオフにする**:** -o フラグを指定すると、制御スクリプトが haemtrcoff コマンドを使用して、haemd デーモンのトレース機能をオフにします。また、tracesoff コマンドを使用して、harmad デーモンのトレースも使用不可にします。

ロギング**:** この実行中、イベント管理デーモンは通常、AIX エラー・ログにエントリーを書き込むことによって、その操作とエラーに関する情報を提供します。それが不可能な場合は、エラーはログ・ファイル /var/ha/log/em.default.cluster_name に書き込まれます。

フラグ

-a サブシステムを追加します。

-s サブシステムを始動します。

-k サブシステムを停止します。

-d サブシステムを削除します。

-c サブシステムをクリーンアップします。

-t サブシステムのトレース機能をオンにします。

-o サブシステムのトレース機能をオフにします。

-h 使用方法に関する情報を表示します。

セキュリティー

有効なユーザー ID である **root** を使用して実行する必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 エラーが発生したことを示します。

制限

このコマンドは、HACMP™ 環境でのみ有効です。

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

1. イベント管理サブシステムを SRC へ追加するには、次のように入力します。

   ```
   emsvcsctrl -a
   ```

2. イベント管理サブシステムを始動するには、次のように入力します。

   ```
   emsvcsctrl -s
   ```

3. イベント管理サブシステムを停止するには、次のように入力します。

   ```
   emsvcsctrl -k
   ```

4. イベント管理サブシステムを SRC から削除するには、次のように入力します。

   ```
   emsvcsctrl -d
   ```

5. イベント管理サブシステムをクリーンアップするには、次のように入力します。

   ```
   emsvcsctrl -c
   ```

6. イベント管理デーモンのトレース機能をオンにするには、次のように入力します。

   ```
   emsvcsctrl -t
   ```

7. イベント管理デーモンのトレース機能をオフにするには、次のように入力します。

   ```
   emsvcsctrl -o
   ```

位置

/opt/rsct/bin/emsvcsctrl
emsvcsctrl スクリプトが含まれています。

ファイル

/var/ha/log/em.default.cluster_name
クラスター cluster_name 上の haemd デーモンのデフォルト・ログが含まれています。

/var/ha/log/em.cp
イベント管理構成データベースのコピー中に発生したエラーのログが入っています。

/var/ha/log/em.trace.cluster_name
クラスター名が cluster_name のクラスターに関する、haemd デーモンのトレース・ログが入っています。

/var/ha/log/em.mkgroup
haemrm グループの作成中に発生したエラーのログが入っています。

/var/ha/log/em.mkdir
/var/ha/lck/haem および /var/ha/soc/haem ディレクトリーの作成中に発生したエラーのログが入っています。

---

enable コマンド

enable コマンドには、**AIX** 印刷サブシステムの **enable**、および **System V** 印刷サブシステムの **enable** についての情報が組み込まれています。

AIX 印刷サブシステムの enable コマンド

目的

プリンター・キュー・デバイスを使用可能にします。

構文

enable *PrinterName ...*

説明

enable コマンドは、*PrinterName* パラメーターで指定したプリンター・キュー・デバイスをオンライン状態にするか、またはプリンター・キュー・デバイスをそのシステムで使用可能にします。

注:

1. このコマンドを実行するには、root ユーザー権限を持っているか、または印刷キュー・グループに属している必要があります。
2. `enable -?` と入力すると、システムは次のエラー・メッセージを表示します。

   ```
   enq: (FATAL ERROR): 0781-048: Bad queue or device name: -?
   ```

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

プリンター・キュー・デバイス `lp0:lpd0` を使用可能にするには、次のように入力します。

```
enable lp0:lpd0
```

ファイル

項目	説明
/etc/qconfig	キュー構成ファイルが入っています。
/etc/qconfig.bin	**/etc/qconfig** ファイルの要約されたバイナリー・バージョンが入っています。
/usr/sbin/qdaemon	キュー・デーモンが入っています。
/var/spool/lpd/qdir/*	キュー要求が入っています。
/var/spool/lpd/stat/*	デバイスの状況に関する情報が入っています。
/var/spool/qdaemon/*	キューにあるファイルの一時コピーが入っています。

System V 印刷サブシステムの enable コマンド

目的

LP プリンターを使用可能にします。

構文

enable ***printers***

説明

enable コマンドは、指定されたプリンター を起動し、 **lp** コマンドが実行依頼した要求が印刷できるようにします。プリンターがリモートである場合は、このコマンドを実行しても、要求のリモート・システムへの転送ができるようになるだけです。リモート・システム上で、**enable** コマンドを再度実行する必要があります。 (プリンターの状況を取得するには、**lpstat -p** を実行してください。)

プリント・デバイスの属性が変更された場合、その変更は **enable** によって認識されます。したがって、デバイスの定義または割り当てを変更するには、そのデバイス上のプリンターを使用不可にして、デバイスを変更し、次に、**enable** を実行する必要があります。 **enable** を実行すると、新しいデバイス属性が有効になります。

プリンター名はシステム定義ワード なので、 ASCII 文字の大文字小文字しか使用できません。

`enable -?` と入力すると、システムはコマンド使用方法のメッセージを表示し、0 を返します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

ファイル

/var/spool/lp/*

関連資料:

168 ページの『disable コマンド』

関連情報:

cancel コマンド

lp コマンド

印刷キューの開始および停止

---

enotifyevent コマンド、notifyevent コマンド

目的

イベント応答リソース・マネージャー (ERRM) によって生成されたイベント情報を、指定されたユーザー ID にメールで送信します。

構文

enotifyevent [**-h**] [*user-ID*]

notifyevent [**-h**] [*user-ID*]

説明

enotifyevent スクリプトは常にメッセージを英語で戻します。 **notifyevent** スクリプトのメッセージが戻される言語は、ロケール設定により異なります。

これらのスクリプトは、イベント応答リソース・マネージャー (ERRM) が環境変数内にポストしたイベント情報をキャプチャーします。この環境変数は、イベントの発生時に ERRM が生成します。これらのスクリプトは、イベント応答リソースが実行するアクションとして使用することができます。また、他のユーザー定義アクションを作成するためのテンプレートとしても使用できます。

ERRM 環境変数に関するイベント情報が戻されます。また、イベント情報には以下も含まれます。

Local Time
イベントまたはリアーム・イベントが監視される時刻。ERRM から提供される実際の環境変数は、ERRM_TIME です。この値は、現地時間に変換され、表示される前に読み取り可能なフォームに変換されます。

AIX では、これらのスクリプトは、**mail** コマンドを使用して、指定されたユーザー ID にイベント情報を送信します。ユーザー ID を指定すると、それが有効であると想定され、検証されずに使用されます。ユーザー ID を指定しないと、コマンドを実行しているユーザーがデフォルトとして使用されます。

user-ID はオプションで、イベント情報をメール送信する宛先のユーザー ID を指定できます。*user-ID* を指定しないと、コマンドを実行しているユーザーがデフォルトとして使用されます。

フラグ

-h スクリプトの使用方法に関するステートメントを標準出力に書き込みます。

パラメーター

log_file
イベント情報を記録するファイルの名前を指定します。*log_file* パラメーターには、絶対パスを指定する必要があります。

AIX では、*log_file* は循環ログとして処理され、サイズは 64KB の固定です。*log_file* がいっぱいになると、新しいエントリーによって既存の最も古いエントリーが上書きされます。

その他のプラットフォームでは、*log_file* のサイズは限定されていないので、上書きされることはありません。ファイル・サイズは、管理者が定期的にエントリーを除去しないかぎり、無限に増え続けます。

log_file が既に存在する場合、イベント情報はそのファイルに追加されます。*log_file* が存在しない場合は、ファイルが作成されて、イベント情報はそこに書き込まれます。

終了状況

0 コマンドが正常に実行されました。

制限

1. これらのスクリプトは、ERRM が実行されているノード上で実行する必要があります。
2. **mail** コマンドは、ファイルを読み取るために使用されます。

標準出力

-h フラグを指定すると、スクリプトの使用方法に関するステートメントが標準出力に書き込まれます。

例

1. **mail** コマンドを使用すると、イベント情報の内容を読むことができます。次の例には、**/var** ファイルシステム (ファイルシステム・リソース) の警告イベントをフォーマットおよびログ記録する方法が示されています。

```
=============================================================================
Event reported at Sun Mar 26 16:38:03 2002

Condition Name:      /var space used
Severity:     Warning
Event Type:       Event
Expression:       PercentTotUsed>90

Resource Name:     /var
Resource Class Name:    IBM.FileSystem
Data Type:     CT_UINT32
Data Value:     91
```

位置

/opt/rsct/bin/enotifyevent
enotifyevent スクリプトが入っています。

/opt/rsct/bin/notifyevent
notifyevent スクリプトが入っています。

---

enq コマンド

目的

ファイルをキューに入れます。

構文

ファイルに名前を付ける

enq [**-**] [**-B** *CharacterPair*] [**-c**] [**-C**] [**-G**] [**-j**] [**-m** *Text*] [**-M** *File*] [**-n**] [**-N** *Number*] [**-o** *Option*] [**-P** *Queue*] [**-r**] [**-R** *Number*] [**-t** "*User*"] [**-T** *Title*] [**-Y**] [**-Z** *Name*] *File*

印刷ジョブの優先順位を変更する

enq -a *Number* **-#** *JobNumber*

状況を表示する

enq [**-q** | **-A**] [**-L**] [**-W**] [**-e**] [**-#** *JobNumber*] [**-u** *Name*] [**-w** *Seconds*] [**-s**]

キューおよびキュー・デーモン状況を変更する

enq [**-d**] [**-D**] [**-G**] [**-K**] [**-L**] [**-q** | **-A**] [**-U**]

オプションを取り消す

enq [**-X**] [**-x***Number*] [**-P***Printer*]

印刷ジョブを保留、解放、または別のキューに移動する

enq {**-h** | **-p** |**-Q** *NewQueue* } { **-#** *JobNumber* [**-P** *Queue*] | **-u** *User* | **-P** *Queue* }

印刷ジョブをキューに入れて保留する

enq -H *File ...*

説明

enq コマンドは、共用リソース (通常はプリンター・デバイス) に対する要求をキューに入れる汎用ユーティリティーです。 **enq** コマンドを使用すると、要求をキューに入れたり、取り消したり、要求の優先順序を変更したり、キューやデバイスの状況を表示できます。

enq コマンドは、すべてのフラグが共に機能するわけではないため、5 つの異なった構文図を持っています。これらのフラグの一部はファイル処理用で、*FileName* をオプションとして受け取ります。その他のフラグは、印刷ジョブの優先順位の変更、状況の表示、キューの状況またはキューのデーモンの変更、印刷ジョブの取り消しなどに使用されます。

ファイルを特定のキューに入れる場合、 **-P** フラグ (**-P** *Queue*) を使用します。そのキューに対して複数のデバイスが使用されている場合、キューの名前の後にそのデバイス (*:device*) を指定して、特定のデバイスを要求することもできます。デバイスを指定しない場合、ジョブは最初に使用可能となったデバイスに送信されます。ファイルを指定しない場合、**enq** コマンドは標準入力をファイルにコピーして、それを印刷キューに入れます。

enq コマンド要求は、その要求に関連するオペレーター・メッセージを持つことができます。この機能は、分散環境や多くのユーザーを持つシステム上で役立ちます。このメッセージは、そのジョブの印刷を始める前に、特別な用紙や違った色の用紙をプリンターに装着するなどの要求情報をプリンターのオペレーターに伝えます。このメッセージは、 **-m** フラグと、 **-M** フラグで指定します。**qdaemon** コマンドは、**enq** コマンドの要求を処理します。**qdaemon** コマンドが要求とその関連メッセージを開始する準備ができると、システムは **qdaemon** プロセスが実行されるコンピューターのコンソールにメッセージを表示します。メッセージのテキストにはプロンプトが付いていて、継続するように要求する信号を送る方法、または要求を取り消す方法を、プリンターのオペレーターに伝えます。

enq -A コマンドで生成された表示には、リモート・キューに対する 2 つのエントリーが含まれています。第 1 のエントリーには、クライアントのローカル・キューとローカル・デバイス名、その状況情報が含まれます。第 2 のエントリーは第 1 のエントリーのすぐ後にあり、クライアントのローカル・キュー名 (再度) の後にリモート・キュー名が表示されます。リモート・キューに対して実行依頼されたジョブは最初にローカル側に表示され、ジョブがリモート・マシン上で処理されるときにリモート・デバイスに移動します。

状況コマンドがリモート・ホストと交信する際に、コマンドがリモート・マシンからの応答を待つ間、画面が頻繁に停止するように見えます。2 つのマシン間の接続を確立できない場合、コマンドは最終的にタイムアウトになります。

注**:**

1. ファイルをキューに入れるには、そのファイルへの読み取りアクセスを持っていなければなりません。ファイルをキューから除去するには (**-r** フラグを参照)、そのファイルが入っているディレクトリーへの書き込みアクセスも必要です。
2. **enq** コマンドが実行された後で印刷される前に、引き続きファイルを変更する場合は、 **-c** フラグを指定してください。
3. ファイルをプリンター上のキューに入れるときには、フラグはどんな順序で指定してもかまいません。
4. **-d** および **-G** フラグは、即座に機能します。コマンド・ライン上でこれらの 2 つのフラグの前にある構文エラーが報告されます。コマンド・ライン上でこれらの 2 つのフラグの後にある構文エラーは無視されます。

フラグ

ファイル処理オプション

enq コマンドにファイル名のリストを提供すると、提供したファイルがすべてキューに入れられ、デフォルト・デバイスまたは指定したデバイス上でファイルの処理が行われます。

項目	説明
-	**enq** コマンドをフィルターとして機能させます。ファイルを指定しないと、**enq** コマンドは自動的に標準入力を読み込みます。ただし、ファイルを指定する場合、ダッシュ **(-)** を使用して、**enq** コマンドに標準入力を読み取らせるよう強制することができます。ダッシュ **(-)** は、実際はフラグではなく、特別なタイプのファイル名です。したがって、そのコマンド・ライン上で他のフラグをすべて指定した後に指定する必要があります。
-B *CharacterPair*	次のような *CharacterPair* の値に従って、バースト・ページの印刷を制御します (**n** は「never (なし)」、**a** は「always (必須)」、**g** は「group (グループ)」を示します。先頭の文字はヘッダー、2 番目の文字はトレーラーを表します)。 **HT** 説明 **nn** ヘッダーなし、トレーラーなし。 **na** ヘッダーなし、各ファイルにトレーラーあり。 **ng** ヘッダーなし、ジョブの最後にトレーラーあり。 **an** 各ファイルにヘッダーあり、トレーラーなし。 **aa** ジョブの各ファイルにヘッダーとトレーラーあり。 **ag** 各ファイルにヘッダー、ジョブの後にトレーラーあり。 **gn** ジョブの最初にヘッダー、トレーラーなし。 **ga** ジョブの最初にヘッダー、各ファイルの後にトレーラーあり。 **gg** ジョブの最初にヘッダー、ジョブの最後にトレーラーあり。 **/etc/qconfig** ファイルの中のヘッダーとトレーラーのスタンザは、バースト・ページのデフォルトの処理を定義します。 **注:** リモート印刷環境のデフォルトでは、ヘッダー・ページが印刷され、トレーラー・ページは印刷されません。
-c	ファイルをコピーします。ディスク・スペースを節約するために、**enq** コマンドはファイル名を記憶しますが、実際にはファイル自体をコピーしません。現行コピーが印刷されるのを待つ間に、ファイルの変更を継続したい場合には、**-c** フラグを使用します。
-C	エラー・メッセージとジョブの完了通知に、**write** コマンドでなく、**mail** コマンドを使用するよう指定します。(このフラグを使うとプリンターからのフィードバックがより良く行われるので、PostScript アプリケーションを書くときに便利です)。エラー・メッセージとジョブ完了メッセージ (どちらも **piobe** コマンドで生成) は、プリンターから読み取ったデータと一緒にメールで返送されます。 **-C** フラグはローカル印刷ジョブにのみ適用されます。リモート・プリンターに送信されたジョブが完了したときに通知が必要な場合は、**-n** フラグを使用してメール・メッセージを受信します。 **注: qdaemon** とプリンター・バックエンドからはどうしてもリダイレクトできないメッセージがいくつかあります。それらのメッセージはシステム・エラーであり、**/dev/console** ファイルに直接送られます。
-j	メッセージ `Job number is: nnn` を、標準出力に表示するよう指定します。この場合、 `nnn` は割り当てられたジョブ番号です。そのメッセージは、ジョブがローカル印刷キューに渡された場合に表示されます。
-m *Text*	**enq** コマンド要求と一緒にオペレーター・メッセージを渡します。指定したテキストにはメッセージが入ります。
-M *File*	**enq** コマンド要求と一緒にオペレーター・メッセージを渡します。指定したファイルにはメッセージのテキストが入ります。

項目	説明
-n	ジョブの終了を通知します。 **-t** フラグを同時に指定すると、 **enq** コマンドによって、要求の宛先も通知されます (**-t** フラグのセクションを参照)。
-N *Number*	*Number* が示す部数だけファイルを印刷します。通常、ファイルは 1 回だけ印刷されます。
-o *Option*	バックエンド固有のフラグがバックエンドに渡されるよう指定します。したがって、各キューには、**enq** コマンド・ラインに組み込まれる可能性があってもこのセクションで説明されていないフラグが存在します。これらのフラグのリストについては、**piobe** コマンドのセクションを参照してください。
-P *Queue*	ジョブが送信されるキューを指定します。キュー上の特定のデバイスは、`-P Queue:Device` と入力して指定できます。
-r	ファイルの印刷が正常に行われた後に、そのファイルを除去します。
-R *Number*	現行ジョブの優先順位を *Number* に設定します。このフラグは、ジョブの実行を要求するときに使用されます。ジョブの実行を要求した後に優先順位を変更する場合は、 **-a** フラグを使用してください。数が大きいほど、優先順位も高くなります。デフォルトの優先順位は 15 です。大部分のユーザーの場合、最大優先順位は 20 で、root ユーザー権限を持つユーザーの場合は 30 です。
-t "*User*"	送達のために出力に *User* というラベルを付けます。通常、送達のために、出力には **enq** コマンド要求を発行したユーザーのラベルが付けられます。*User* の値は、正規のユーザー ID の要件を満たす単一のワードであることが必要です。
-T *Title*	**-q** フラグを指定していると、ヘッダー・ページにタイトルを付け、そのタイトルを表示します。通常、ジョブのタイトルはファイル名となります。 **enq** コマンドが標準入力から読み込みを行う場合は、ジョブ・タイトルは **STDIN.#** で、**#** は **enq** コマンドのプロセス ID です。
-Y	**enq** コマンドに対して、このフラグ以降のコマンド・ラインをすべて無視するように伝えます。このフラグは、キュー (**/etc/qconfig** ファイル内にある場合) が有効であるかどうかを調べるときに役に立ちます。例えば、ライン・プリンター **lp4** が有効なキューの場合、 `enq -P lp4 -Y` と入力すると終了値 0 が戻され、有効でなければ 0 以外の値が戻されます。このフラグは、**qdaemon** コマンドで **/etc/qconfig** ファイルを再度要約するよう強制するときにも役に立ちます。
-Z *Name*	リモート印刷ジョブの発信元を指定します。

印刷ジョブ優先順位オプション

項目	説明
-a *Number*	指定したジョブの優先順位を *Number* に変更します。このフラグを指定して **enq** コマンドを入力する前に、ジョブが出力用に実行依頼されている必要があります。優先順位の詳細については、 **-R** フラグのセクションを参照してください。ジョブ番号を指定する場合は、 **-#** フラグを使用してください。このフラグは、ローカル印刷ジョブに対してのみ有効です。
-#*JobNumber*	**enq -q** コマンド、または **enq -a** コマンドで使用されるジョブ番号を指定し、状況の出力に指定されているジョブだけを表示します。

注:

1. デフォルトの宛先プリンターを指定変更するには、 **-P** *Queue* を指定します。
2. ジョブ 1、2、3 がプリンター・キューにあり、ジョブ 1 が実行中であるにもかかわらずジョブ 3 の状況を要求することを指定した場合、ジョブ 3 だけでなくジョブ 1 の状況情報も出力されます。
3. 存在しないジョブ番号を指定した場合、エラー・メッセージの代わりに、現在キューにある現行ジョブ番号が表示されます。

状況表示オプション

項目	説明
-A	すべてのキューの状況を提供します。そのフラグの機能は、 **qconfig** ファイルのキューごとに、 **enq -q** コマンドを一度実行するのと同じです。
-e	**qdaemon** コマンドが制御していないキューからの状況情報を除外します。この種のキューの状況は、フォーマットが異なる場合があります。**-e** フラグは他のフラグと任意に組み合わせて使用することができます。
-L	詳しい状況を指定します。このフラグは、**-A**、**-q**、または **-W** のフラグと一緒に使用できます。**-L** フラグと **-W** フラグが同時に使用されると、印刷ジョブの詳しい状況がセミコロンで区切られた形式で表示されます。1 つの印刷ジョブの中で出力される複数のファイルを表示するには、**-L** フラグを使用します。
-q	デフォルトのキューの状況を表示します。**LPDEST** および **PRINTER** 環境変数は、デフォルトのプリンターの名前を制御します。 **LPDEST** 環境変数に値が入っていると、その値が常に最初に使用されます。**LPDEST** 変数に値が入っていない場合、**enq** コマンドは **PRINTER** 環境変数を使用します。 **PRINTER** 環境変数に値が入っていない場合は、**enq** コマンドはシステム・デフォルトを使用します。 **注:** 1. 特定のキューの状況を表示するには、 **-P** *Queue* フラグを、 **-q** フラグと一緒に使用してください。 2. 宛先コマンド・ラインのオプションはすべて、**LPDEST** と **PRINTER** の両方の環境変数を指定変更します。
-s	ファイルをリストせずに印刷キューの状況を入手します。
-u *Name*	印刷ジョブ状況のユーザー名を指定します。
-w *Seconds*	キュー状況の継続出力を指定し、指定した *Seconds* ごとに、キューが空になるまで、画面を更新します (**lpq** コマンドを参照)。キューが空になると、処理は停止します。このフラグは必ず、**-q** フラグ、 **-A** フラグ、または **-L** フラグのいずれかと一緒に使用します。
-W	幅の広い状況フォーマットをより長いキュー名、デバイス名、およびジョブ番号で指定します。このフラグは、**-A**、**-q**、または **-L** のフラグと一緒に使用できます。**-L** と **-W** のフラグが同時に使用されると、印刷ジョブの詳しい状況がセミコロンで区切られた形式で表示されます。

キューおよびキュー・デーモン状況の変更オプション

項目	説明
-d	**/etc/qconfig** ファイルで、 **digest** コマンドを実行します。要約 (digest) が完了すると、**/etc/qconfig** ファイルに加えられたすべての変更は、**/etc/qconfig.bin** ファイルに反映されます。このオプションを実行するには、root ユーザー権限を持っていなければなりません。

enq コマンドでは、すべてのユーザーが使用できる前述のフラグの他に、root ユーザー権限を持つユーザーによって入力される次のフラグを使用できます。 root ユーザー権限とは、その人が root であるか、または **printq** グループに属していることを意味します。

注**:** 以下のフラグは、ローカル印刷ジョブでのみ使用できます。

項目	説明
-D	即時停止。キューに関連しているデバイスの電源をオフにします。**qdaemon** プロセスは、デバイスにジョブを送信しなくなります。 **enq -q** コマンドを入力すると、「DOWN」という状況が表示されます。デバイスで実行中のジョブは終了されます。

項目	説明
-G	安全に停止。現在実行中のジョブが終了した後に、**qdaemon** プロセスを終了します。このフラグを使用する方法は、**qdaemon** プロセスを簡単に終了させる唯一の方法です。 **kill** コマンドを使用すると、キューにあるジョブがハングするなどの問題が発生することがあります。 **qdaemon** プロセスを **srcmstr** の制御下で実行している場合 (デフォルト構成)、 **enq -G** は、**qdaemon** が自動的に再始動するのを防ぎません。デフォルトの構成を変更し、**qdaemon** プロセスの自動再始動を防ぐ **chssys** コマンドを使用する必要があります。次のコマンド `chssys -s qdaemon -O` を **enq -G** コマンドの前に実行すれば、 **qdaemon** コマンドが自動的に再始動されるのを防ぐことができます。 次のコマンド `startsrc -s qdaemon` は、**qdaemon** プロセスを手動で再始動します。
-K	現在のジョブすべてが抹消されることを除いて、 **-D** フラグと同じ機能を実行します。抹消されたジョブはキューにそのまま残り、デバイスの電源が入れられたときに再度実行されます。
-L	詳しい状況を指定します。このフラグは、**-A**、**-q**、または **-W** のフラグと一緒に使用できます。1 つの印刷ジョブの中で出力される複数のファイルを表示するには、**-L** フラグを使用します。
-U	キューに関連するデバイスを立ち上げます。**qdaemon** プロセスは、キューに再度ジョブを送信し、**enq -q** コマンドを入力すると、キューの状況が ready として表示されます。

注**:** 1 つのキューが複数のデバイスを共用する場合、 **-D** フラグ、 **-K** フラグ、および **-U** フラグを使用するときには、キューだけでなくデバイスも指定する必要があります。例えば、`-P lp:lpd` と入力すると、そのキューに他のデバイスがない場合にだけ、同じデバイスを表します。

取り消しオプション

項目	説明
-X	ジョブの印刷を取り消します。ユーザーが root ユーザー権限を持っていれば、指定したキュー上のすべてのジョブが削除されます。このフラグは、ローカル印刷ジョブに対してのみ有効です。
-x*Number*	*Number* で指定した番号のジョブの印刷を取り消します。
-P *Printer*	すべてのジョブまたは特定のジョブのジョブ番号を取り消したいプリンター を指定します。

注意**:** root ユーザー権限を持っているユーザーがキューを指定しないと、すべてのキューのすべてのジョブが削除されます。

印刷ジョブの保留および解放オプション

項目	説明
-#*JobNumber*	保留または解放する印刷ジョブの番号を指定します。
-h	指定した印刷ジョブを保留します。
-H	*File* パラメーターで指定したファイルをキューに入れて保留します。
-p	指定した印刷ジョブを解放します。
-P *Queue*	保留または解放する印刷キューを指定します。
-u *User*	印刷ジョブを保留または解放するユーザーを指定します。

印刷ジョブの移動オプション

項目	説明
-#*JobNumber*	移動する印刷ジョブの番号を指定します。
-P *Queue*	移動する印刷キューを指定します。*Queue* 変数の値は、キュー名、またはキュー: デバイス名のフォーマットで指定できます。
-Q *NewQueue*	印刷ジョブの移動先となるターゲット・キューを指定します。*NewQueue* 変数の値は、キュー名、またはキュー:デバイス名の形式で指定できます。
-u *User*	移動する印刷ジョブのユーザーを指定します。

セキュリティー

監査イベント:

イベント	情報
ENQUE_admin	キュー名、デバイス名、ジョブ名、ユーザー名

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. ファイル `memo` をデフォルトのプリンター上で印刷するには、次のように入力します。

   ```
   enq memo
   ```

2. ファイル `prog.c` をページ番号付きで印刷するには、次のように入力します。

   ```
   pr prog.c | enq
   ```

   pr コマンドは、各ページの先頭に見出しを付けます。この見出しには、ファイルが最後に変更された日付、ファイル名、およびページ番号が表示されます。それから **enq** コマンドがファイルを印刷します。

3. 標準入力から読み込んで、ページ番号付きでファイルを印刷するには、次のように入力します。

   ```
   pr x | enq  -P bill  -n  -r fn1 - fn3
   ```

   特別なファイル名であるダッシュ (**-**) は、**enq** コマンドに標準入力から読み込むように伝えます。通常、コマンド・ラインにファイル名がある場合、**enq** コマンドは標準入力から読み込みを行いません。また、これは印刷する順序も示しています。 **pr** コマンドは、`x` ファイルのページ番号を付けたバージョンを作成し、そのファイルを **enq** コマンドに渡します。enq コマンドは、**/var/spool/qdaemon** ファイル内に、出力が入った一時ファイルを作成します。

   enq コマンドは、4 つのファイルを持つジョブを作成し、キュー `bill` に実行依頼します。 `fn1` ファイルを 2 回印刷します。次に、enq コマンドは **pr** コマンドの出力を、どのようなものでも印刷します。最後に、ファイル `fn3` を印刷します。4 つのファイルは、バースト・ページ用に、1 つのジョブとして扱われます。ジョブが完了すると、通知が送信されます (`-n` フラグ)。`-r` フラグが指定されているので、`fn1` と `fn3` ファイルはジョブの完了時に除去されます。ダッシュ (**-**) ファイルによって作成された一時ファイルは、常に削除されます。

   pr コマンドは、各ページの先頭に見出しを付けます。この見出しには、ファイルが最後に変更された日付、ファイル名、およびページ番号が表示されます。それから **enq** コマンドがファイルを印刷します。

4. ファイル report を、次に使用可能なプリンター (fred キューに設定しているプリンター) に出力するには、次のように入力します。

   ```
   enq -P fred report
   ```

5. プレフィックス sam で始まる複数のファイルを、次に使用可能なプリンター (fred キュー用に構成されているプリンター) に出力するには、次のように入力します。

   ```
   enq -P fred sam*
   ```

   プレフィックス sam で始まるすべてのファイルが、1 つの印刷ジョブに含まれます。**-T** フラグで他の値を指定していない限り、通常の状況コマンドでは、印刷ジョブのタイトル (この場合、キューの最初のファイルの名前) だけが表示されます。印刷ジョブの中のすべてのファイルの名前を表示するには、詳細状況コマンド **enq-A-L** を使用します。

6. ファイルが印刷されるのを待っているかどうかについて、印刷キューを調べるには、次のように入力します。

   ```
   enq  -q
   ```

   このコマンドは、ユーザーのデフォルト・キュー状況を表示します。ファイルがまだ印刷されていない場合は、キュー状況のリストに表示されます。システム・デフォルトのキューは、**/etc/qconfig[.bin]** ファイルの最初のキューとして定義されます。ユーザーは、**PRINTER** 環境変数を設定しエクスポートして、デフォルトを指定変更できます。

7. デフォルトでないキュー lp0 の状況を表示するには、次のように入力します。

   ```
   enq  -q  -P lp0
   ```

8. キューの詳細状況を表示するには、次のように入力します。

   ```
   enq  -L
   ```

9. すべてのキューについて状況を表示するには、次のように入力します。

   ```
   enq  -A
   ```

10. すべてのキューについて詳細状況を表示するには、次のように入力します。

    ```
    enq  -A  -L
    ```

11. デフォルトのキューの状況を幅広いフォーマットで表示するには、次のように入力します。

    ```
    enq  -W
    ```

12. すべてのキューの幅広い状況を表示するには、次のように入力します。

    ```
    enq  -W  -A
    ```

13. ジョブ (ジョブは 1 つ以上のファイルを指す) の印刷を停止するには、次のように入力します。

    ```
    enq  -x 413
    ```

    このコマンドは、ジョブの印刷の要求を取り消します。番号は、**enq -q** コマンドを入力して得たリストに登録されているものです。ジョブが印刷中である場合、プリンターは即座に停止します。ジョブがまだ印刷されていない場合、印刷されないようキューから除去されます。ジョブがキューにない場合、**enq** コマンドは次の出力のようなメッセージを表示します。

    ```
    no such request from you -- perhaps it's done?
    ```

14. プリンターをキュー・システムから切り離すには、次のように入力します。

```
enq  -P lp0:dlp0  -D
```

このコマンドを入力すると、 lp0 のキューを処理中のプリンターに対する、 **enq** コマンド要求の送信が停止されます。 ファイルが印刷中の場合は、終了するまで印刷されます。**enq** コマンドを実行するには、**qadm** コマンドが実行できなければなりません。

注**:** 指定されたキューを処理するプリンターは、**/etc/qconfig[.bin]** ファイルに記述されているデバイス・スタンザ名によって指定されます。

15. デフォルトのプリンターで、 **piobe** コマンドのバックエンドを使って、ページ番号付きでファイルを印刷するには、次のように入力します。

```
enq  -o -p filename
```

-p フラグは、**enq** コマンドに無視されます。-o フラグは次の項目 (引用符で囲むこともできます) を、変更されていないバックエンドに渡すように **enq** コマンドに伝えます。そのため、**enq** コマンドは -p フラグを **qdaemon** プロセスに引き渡し、次に **piobe** プロセスがバックエンドの piobe にそれを渡します。-p フラグは、文書にページ番号を付けた後で、**piobe** に **/usr/bin/pr** フィルターを実行させて、データをデバイスに渡します。複数のオプションを、1 つの **-o** フラグが前に付いている引用符で囲んで指定できます。また、複数のオプションを、引用符を使用せずに各オプションの前に **-o** フラグを付けて指定することもできます。

16. 次のような情報を持つ **qconfig** ファイルを仮定し、

```
qname:
               device = fred
fred:
               file = /tmp/hello
               backend = /usr/bin/sh /usr/bin/diff
```

さらに次のようなコマンドを指定します。

```
rm /tmp/hello
touch /tmp/hello
pr /etc/hosts|enq  -P qname:fred - /etc/hosts
```

qdaemon プロセスは、2 つの引数を持つ /usr/bin/diff プログラムを実行します。1 つは一時ファイル名で、もう 1 つは /etc/hosts ファイルです。この 2 つのファイルの唯一の違いは、一方のファイルが **pr** コマンドで実行されたものである点です。 /tmp/hello ファイルには、2 つのファイル間の違いが入っています。**qdaemon** プロセスは、/tmp/hello ファイルが存在しない場合にはそれを作成しません。

17. 次のコマンド

```
enq  -m'i want pink paper for this job' /etc/passwd
```

は、印刷ジョブが始まる直前に、指定したオペレーター・メッセージをオペレーターのコンソールに送信します。オペレーターは、そのジョブを継続するか取り消すかを、このメッセージに応答する必要があります。

```
enq  -M pink /etc/passwd
```

このコマンドも同じことを実行しますが、メッセージは pink というファイルに入っています。

18. キュー fred にあるすべてのジョブを取り消すには、次のように入力します。

```
enq  -X  -P fred
```

このコマンドを入力したユーザーが root ユーザー権限を持っている場合は、すべてのジョブがキュー fred から削除されます。ユーザーが root ユーザー権限を持っていない場合は、そのユーザーのジョブだけがそのキューから削除されます。

19. ファイル名 MyFile のファイルをキューに入れ、 MyFile のジョブ番号を **jdf** ファイルに戻すには、次のように入力します。

```
enq  -j MyFile
```

20. 印刷ジョブ番号 310 を保留にするには、次のように入力します。

```
enq  -h  -#310
```

印刷ジョブ番号 310 に対する保留を解放するには、次のように入力します。

```
enq  -p -#310
```

21. キュー lp0 に入っているすべての印刷ジョブを保留するには、次のように入力します。

```
enq  -h  -P lp0
```

lp0 キューを解放するには、次のように入力します。

```
enq  -p -P lp0
```

22. fred によって作成されたすべての印刷ジョブを保留するには、次のように入力します。

```
enq  -h  -u fred
```

fred によって作成された印刷ジョブを解放するには、次のように入力します。

```
enq  -p -u fred
```

23. ジョブ番号 318 を、キュー lp0 に移動するには、次のように入力します。

```
enq  -Q lp0  -#318
```

印刷ジョブの移動を制御するフラグは、印刷ファイルを保留するフラグと同様に機能します。保留フラグと変数は、上記の例に示してあります。

ファイル

項目	説明
/usr/sbin/qdaemon	キューイング・デーモン。
/etc/qconfig	キュー構成ファイル。
/var/spool/lpd/qdir/*	キュー要求。
/var/spool/lpd/stat/*	デバイスの状況に関する情報。
/var/spool/qdaemon/*	エンキューされたファイルの一時コピー
/etc/qconfig.bin	**/etc/qconfig** ファイルの要約バイナリー・バージョン。

関連情報:

mkque コマンド

キュー特性の変更または表示

プリンター固有の情報

プリンター・コロン・ファイルの規則

enroll コマンド

目的

セキュア通信チャネルをインプリメントするためのパスワードを設定します。

構文

enroll

説明

enroll コマンドは、パスワードを設定し、通信チャネルを確保するために使われます。この通信チャネルでは意図された受信者だけがメッセージを読み込むことができます。パスワードは、秘密メールを受信時に使用されます。

enroll コマンドは、**xsend** および **xget** コマンドと一緒に使用され、秘密メールの送受信を行います。**xsend** コマンドは秘密メールを送信するのに使用されます。 **xget** コマンドは受信者のパスワードを求め、それから秘密メールを渡します。

例

パスワードを設定するには、次のように入力します。

```
enroll
```

プロンプトが表示されたら、受信者は自分のパスワードを入力します。これでシステム上の他のユーザーが秘密メールを送信できるようになります。受信者は、**xget** コマンドを使ってその秘密メールを読むことができます。

ファイル

項目	説明
/var/spool/secretmail/User.key	ユーザー用に暗号化されたキーが入っています。
/usr/bin/enroll	**enroll** コマンドが入っています。

関連情報:

mail コマンド

xget コマンド

xsend コマンド

機密メールの送受信

enscript コマンド

目的

テキスト・ファイルを印刷用の PostScript フォーマットに変換します。

構文

enscript [**-1 -2 -c -g -k -l -m -o -q -r -B -G -K -R**] [**-b** *Header*] [**-f** *Font*] [**-f0** *CodeSet*:*Font*] [**-f1** *CodeSet*:*Font*] [**-p** *Out*] [**-F** *Hfont*] [**-F0** *CodeSet*:*Font*] [**-F1** *CodeSet*:*Font*] [**-L** *Lines*] [**-M** *MediaName*] [**-X** *CodesetName*] [*SpoolerOptions*] [*File ...*]

説明

enscript コマンドは、テキスト・ファイルを読み込んで、PostScript フォーマットに変換し、PostScript プリンターで印刷するためにスプールします。このコマンドを使用すると、フォント、見出し、限定フォーマット・オプション、スプーリング・オプションを指定できます。

次に例を示します。

```
enscript -daleph bubble.txt
```

は、aleph というプリンター上で **bubble.txt** ファイルを印刷します。

```
enscript -2r finder.c
```

は、デフォルトのプリンターで **finder.c** ファイルの二段組の横長リストを印刷します。

ENSCRIPT 環境変数は、デフォルトの指定に使用されます。**ENSCRIPT** の値は、引数がコマンド・ラインに表示される前に、引数の文字列として解釈されます。次に例を示します。

```
ENSCRIPT='-fTimes-Roman8'
```

これにより、デフォルトの本体の書体のサイズとフォントを 8 ポイントの Times Roman に設定します。

psdit コマンドと **enscript** コマンドに対する各種メディア・サイズを含む情報は、**/usr/lib/ps/MediaSizes** ファイルに入っています。

MediaSizes ファイルの各エントリーに必要な情報は、TranScript で使用される PostScript プリンターに一致する **PostScript Printer Description (PPD)** ファイルから得られます。 **PPD** ファイルは、Adobe Systems 社から入手できます。 **PPD** ファイルから取り出されたサイズは、ポイントと呼ばれるプリンターの尺度で表示されます。プリンターのポイントは、1 インチの 72 分の 1 です。

ASCII 文字の * (アスタリスク) で始まる **MediaSizes** ファイル内の行は、コマンド・ラインで **enscript** コマンドおよび **psdit** コマンドに提供されているメディア・サイズ名に一致する場合、すべて無視されます。

MediaSizes ファイルの各エントリーには、8 つまたは 9 つのフィールドが含まれます。最初の 8 つのフィールドは、すべてのエントリーに必須です。9 番目のフィールドはオプションです。フィールドはホワイト・スペースで区切られます。各エントリーのフィールドは、次のとおりです。

フィールド名	説明
EntryName	**enscript** コマンドまたは **psdit** コマンドを使って、**-M** フラグで指定されるメディア名と照合する文字列が入っています。
MediaWidth	ポイント数で表したメディアの幅を指定します。
MediaDepth	ポイント数で表したメディアの上下幅を指定します。
ImageableLLX	ポイント数で表したイメージ可能な左下隅の x 座標を指定します。
ImageableLLY	ポイント数で表したイメージ可能な左下隅の y 座標を指定します。
ImageableURX	ポイント数で表したイメージ可能な右上隅の x 座標を指定します。
ImageableURY	ポイント数で表したイメージ可能な右上隅の y 座標を指定します。
PageRegionName	イメージ可能領域のサイズを識別する特定のプリンター用の PostScript シーケンスを指定します。
PaperTrayName	特定の用紙/メディア・トレイを選択する特定のプリンター用の PostScript シーケンスを指定します。このフィールドはオプションです。 **注:** シーケンスは、PageRegionName フィールドと PaperTrayName の両フィールドについて、複数の PostScript 演算子やワードにすることができます。このようなシーケンスを指定するには、ASCII キャラクターの " (二重引用符) を使用して、シーケンス全体を区切ります。

次の表は、**MediaSizes** ファイルのフィールド・エントリーの例です。

名前	フィールド値
Letter	**Width** 612 縦の長さ 792 **llx** 18 **lly** 17 **urx** 597 **ury** 776 ページ名/領域名 Letter 用紙名/トレイ名 Letter
Legal	**Width** 612 縦の長さ 1008 **llx** 18 **lly** 17 **urx** 597 **ury** 992 ページ名/領域名 Legal 用紙名/トレイ名 Legal

PostScript フォント情報

次のトランスクリプト用 PostScript フォントの表は、enscript コマンドに使用できるフォントを示しています。フォント名は、**-F** および **-f encscipt** コマンド・フラグで指定します。英字は大文字と小文字が区別されます。

トランスクリプト用 PostScript フォント

フォント名	フォント・ファミリー
AvantGarde-Book	AvantGarde
AvantGarde-Demi	AvantGarde
AvantGarde-DemiOblique	AvantGarde
AvantGarde-BookOblique	AvantGarde
Bookman-Demi	Bookman
Bookman-DemiItalic	Bookman
Bookman-Light	Bookman
Bookman-LightItalic	Bookman
Courier	Courier
Courier-Bold	Courier
Courier-BoldOblique	Courier
Courier-Oblique	Courier
Garamond-Bold	Garamond

トランスクリプト用 PostScript フォント

フォント名	フォント・ファミリー
Garamond-BoldItalic	Garamond
Garamond-Light	Garamond
Garamond-LightItalic	Garamond
Helvetica	Helvetica
Helvetica-Bold	Helvetica
Helvetica-Oblique	Helvetica
Helvetica-BoldOblique	Helvetica
Helvetica-Narrow	Helvetica
Helvetica-Narrow-Bold	Helvetica
Helvetica-Narrow-BoldOblique	Helvetica
Helvetica-Narrow-Oblique	Helvetica
LubalinGraph-Book	Lubalin
LubalinGraph-BookOblique	Lubalin
LubalinGraph-Demi	Lubalin
LubalinGraph-DemiOblique	Lubalin

フォント名	フォント・ファミリー
Miryam-Iso	Miryam Iso
Miryam-IsoBold	Miryam Iso
Miryam-IsoBoldItalic	Miryam Iso
Miryam-IsoItalic	Miryam Iso
NarkissimIso	Narkissim Iso
NarkissimIso-Bold	Narkissim Iso
NarkissimIso-BoldItalic	Narkissim Iso
NarkissimIso-Italic	Narkissim Iso
NarkissTamIso	Narkiss Tam Iso
NarkissTamIso-Bold	Narkiss Tam Iso
NarkissTamIso-BoldItalic	Narkiss Tam Iso
NarkissTamIso-Italic	Narkiss Tam Iso
NewCenturySchlbk	NewCentury
NewCenturySchlbk-Bold	NewCentury
NewCenturySchlbk-Italic	NewCentury
NewCenturySchlbk-Roman	NewCentury
Optima	Optima
Optima-Bold	Optima
Optima-BoldOblique	Optima
Optima-Oblique	Optima
Palatino-Bold	Palatino
Palatino-BoldItalic	Palatino
Palatino-Italic	Palatino
Palatino-Roman	Palatino
Rokaa	Rokaa
Rokaa-Bold	Rokaa
Rokaa-BoldItalic	Rokaa
Rokaa-Italic	Rokaa

フォント名	フォント・ファミリー
Setting	Setting
Setting-Bold	Setting
Setting-BoldItalic	Setting
Setting-Italic	Setting
ShalomIso	ShalomIso Iso
ShalomIso-Bold	ShalomIso Iso
ShalomIso-BoldItalic	ShalomIso Iso
ShalomIso-Italic	ShalomIso Iso
Souvenir-Demi	Souvenir
Souvenir-DemiItalic	Souvenir
Souvenir-Light	Souvenir
Souvenir-LightItalic	Souvenir
Times-Bold	Times
Times-BoldItalic	Times
Times-Italic	Times
Times-Roman	Times
Typing	Typing
Typing-Bold	Typing
Typing-BoldItalic	Typing
Typing-Italic	Typing
Symbol	(none)
ZapfChancery-MediumItalic	Zapf
ZapfDingbats	(none)

パラメーター

項目	説明
SpoolerOptions	印刷ファイルをスプールするためのオプションを提供します。*SpoolerOptions* フラグには次のフラグがあります。 **{-d | -P}***Queue* 出力を指定したキューに入れます。 **-n** *Number* 指定した数のコピーを作成します。デフォルトは 1 です。 **-t***Title* 最初のバナー・ページに使用するジョブ・タイトルを設定します。 *File* PostScript フォーマットに変換されるテキスト・ファイルを指定します。このパラメーターをブランクのままにすると、**enscript** コマンドは標準入力から読み取ります。

フラグ

項目	説明
-1	1 段組に設定します (デフォルト)。
-2	2 段組に設定します。
-c	ページ幅より長い行を切り捨てます。通常、長い行はそのページの次の行に折り返されます。
-g	このフラグを指定しても何も実行されませんが、**-g** フラグは下位互換性を保つために引き続き使用できるようになっています。
-k	用紙をあらかじめフィードできるようにします (プリンターがサポートしている場合)。これによって、ページ間でプリンターを作動させられるので、単純なドキュメント (1 つのフォントでプログラムのリストを作成するなど) をいくらか速く印刷することができます。
-l	ライン・プリンターをシミュレートして、ページを 66 行の長さにし、ヘッダーを省略します。
-m	ファイルが印刷された後に、メールを送信します。
-o	**enscript** コマンドがあるフォントのセット内で文字を見つけることができなければ、その文字を表示します。
-q	**enscript** コマンドに実行中の内容を報告させないようにします。 **enscript** コマンドは、ページ数、送信先、省略された文字などに関して報告を行いません。致命的エラーだけは、標準エラー出力に報告されます。
-r	出力を 90 度回転させます (横長モード)。このフラグは、幅広いページを必要とする出力や、**-2** フラグと一緒に使用するプログラム・リストの作成などに使用します。次の例は、プログラム・リストを作成する方法を示しています。 `enscript -2r File . . .`
-B	ページ見出しを省略します。
-G	装飾モードで印刷します。ページ見出し、日付、ページ番号が目立つスタイルで印刷されます (パフォーマンスは多少低下します)。
-K	ページをあらかじめフィードする機能を使用不可にします (デフォルト)。
-R	ポートレイト・モード (回転しない) で印刷します。これはデフォルトです。
-b*Header*	ページ見出しに使用する文字列を *Header* 変数の値に設定します。デフォルトのヘッダーは、ファイル名、最終修正日、およびページ番号から構成されています。
-f*Font*	各ページの本文に使用されるフォントを設定します。 2 段組の回転モードが使用されない場合のデフォルトは、Courier10 で、使用される場合のデフォルトは Courier7 となります。 **注:** 1. PostScript フォント名 (Times-Roman、Times-BoldItalic、Helvetica、Courier など)。 2. ポイント・サイズ (1 ポイント = 1/72 インチ)。フォントは次のように設定されます。 Courier-Bold8 は 8 ポイントの Courier Bold で、Helvetica12 は 12 ポイントの Helvetica を表します。
-f0 *Codeset*:*Font*	PostScript ファイルに書き込まれる文字コード・セット名と、各ページの本文に使用される SBCS フォントを設定します。デフォルト値は、各ロケールの構成ファイル **/usr/lib/ps/transcript.conf** によって決まります。
-f1 *Codeset*:*Font*	PostScript ファイルに書き込まれる文字コード・セット名と、各ページの本文に使用される MBCS フォントを設定します。デフォルト値は、各ロケールの構成ファイル **/usr/lib/ps/transcript.conf** によって決まります。
-p*Out*	PostScript ファイルを、印刷のためにスプールせず、指定したファイルへ書き込むようにします。特別なケースとして、次のように入力すると、PostScript ファイルが標準出力に送信されます。 `-p -`
-F*Hfont*	ページ見出しに使用されるフォントを設定します。デフォルトは Courier Bold10 です。 **注:** フォントの指定は次の 2 つの部分に分かれています。 • PostScript フォント名 (Times-Roman、Times-BoldItalic、Helvetica、Courier など)。 • ポイント・サイズ (1 ポイント = 1/72 インチ)。フォントは次のように設定されます。 Courier-Bold8 は 8 ポイントの Courier Bold で、Helvetica12 は 12 ポイントの Helvetica を表します。
-F0 *Codeset*:*Font*	PostScript ファイルに書き込まれる文字コード・セット名と、各ページのヘッダーに使用される SBCS フォントを設定します。デフォルト値は、各ロケールの構成ファイル **/usr/lib/ps/transcript.conf** によって決まります。
-F1 *Codeset*:*Font*	PostScript ファイルに書き込まれる文字コード・セット名と、各ページのヘッダーに使用される MBCS フォントを設定します。デフォルト値は、各ロケールの構成ファイル **/usr/lib/ps/transcript.conf** によって決まります。

項目	説明
-L*Lines*	1 ページに印刷する最大行数を設定します。 **enscript** コマンドは、通常、1 ページに印刷する行数をポイント・サイズに基づいて計算します (1 ページにつき、**-L** フラグで要求された値よりも少ない行数が印刷される場合もあります)。
-M*MediaName*	用紙上のイメージ可能領域の量を決定するのに使用するメディアの名前を指定します。与えられた名前は、**MediaSizes** ファイルのエントリーと一致します。例えば、**-M legal** は、イメージ可能範囲としてリーガル・サイズを要求します。このフラグを使用しない場合、デフォルト・サイズはレターサイズとなります。レターサイズは横 8.5 インチ、縦 11.0 インチ (横 21.6 センチ、縦 27.9 センチ) です。
-X*CodesetName*	入力データのコード・セットを指定します。デフォルトでは、入力コード・セットは、**nl_langinfo** サブルーチンによって決定されます。このフラグが使用される場合、コード・セットは *CodesetName* によって決定されます。

各国文字サポート

フォント内で見つからない文字はすべて ? (疑問符) で置き換えられます。フォント内に見つからない文字の完全なリストは、 **-o** フラグを使えば得られます。**NLSvec** ファイルでは、文字変換に関する情報が得られます。

環境変数

項目	説明
ENSCRIPT	**enscript** コマンドで使用されるオプションの文字列を指定します。
LPDEST	プリンターの送り先を指定します。**-d** スプーラー・オプションは、この環境変数を指定変更します。
PSLIBDIR	**enscript** コマンド・プロローグおよびフォント・メトリック・ファイルのために、**/usr/lib/ps** ディレクトリーの代わりに使用するディレクトリーのパス名を指定します。
PSTEMPDIR	スプールされた一時ファイルの **/var/tmp** ディレクトリーの代わりに使用する一時ディレクトリーのパスを指定します。
TRANSCRIPT	MBCS 処理のために構成ファイル **/usr/lib/ps/transcript.conf** の代わりに使用するファイルの絶対パス名を指定します。

ファイル

項目	説明
/usr/lib/ps/*.afm	Adobe フォント・メトリック (AFM) ファイルが入っています。
/usr/lib/ps/font.map	フォント名とその省略形のリストが入っています。
/usr/lib/ps/enscript.pro	**enscript** コマンド・ファイルのためのプロローグが入っています。
/usr/lib/ps/MediaSizes	メディア・サイズに使用するデフォルト・ファイルが入っています。

関連情報:

managefonts コマンド

pic コマンド

refer コマンド

troff コマンド

---

entstat コマンド

目的

イーサネット・デバイス・ドライバーとデバイス統計情報を表示します。

構文

entstat [**-d -r -t**] *Device_Name*

説明

entstat コマンドは、指定したイーサネット・デバイス・ドライバーによって収集される統計を表示します。オプションとして一般的なデバイスの統計の他に、デバイス固有の統計を表示するように指定できます。フラグを指定しないと、一般的なデバイスの統計のみが表示されます。

このコマンドは、**-v** フラグを指定して **netstat** コマンドを実行するときにも呼び出されます。**netstat** コマンドは **entstat** コマンド・フラグを発行しません。

無効な *Device_Name* を指定すると、**entstat** コマンドはデバイスに接続できないことを示すエラー・メッセージを生成します。

フラグ

項目	説明
-d	デバイス固有の統計を含め、すべての統計を表示します。
-r	すべての統計を初期値にリセットします。このフラグは、特権ユーザーだけが発行できます。
-t	一部のデバイス・ドライバーでデバッグ・トレースを切り替えます。

パラメーター

項目	説明
Device_Name	**ent0** のようなイーサネット・デバイス名。

統計フィールド

> 注**:** アダプターによっては、特定の統計情報がサポートされない場合もあります。サポートされない統計フィールドの値は、常に 0 です。

entstat コマンドの出力に表示される統計フィールドとその説明は次のとおりです。

タイトル・フィールド

項目	説明
Device Type	アダプター・タイプの記述が表示されます。
Hardware Address	現在デバイスで使用中のイーサネット・ネットワーク・アドレスが表示されます。
Elapsed Time	最後に統計がリセットされてから経過した実時間が表示されます。ハードウェア・エラーが検出されると、エラー・リカバリー処理中に統計情報の一部がデバイス・ドライバーによってリセットされることがあります。このような状況が発生した場合は、2 つの統計情報の時間差を反映するために、出力の途中でもう 1 つの Elapsed Time が表示されます。

送信統計フィールド

項目	説明
Packets	デバイスによって正常に送信されたパケットの数。
Bytes	デバイスによって正常に送信されたバイト数。
Interrupts	ドライバーがアダプターから受け取った送信割り込みの数。
Transmit Errors	このデバイス上で発生した出力エラーの数。このフィールドは、ハードウェア/ネットワーク・エラーにより失敗した送信回数のカウンターです。
Packets Dropped	送信のためにデバイス・ドライバーが受け入れ、デバイスには (何らかの理由で) 提供されなかったパケットの数。
Max Packets on S/W Transmit Queue	ソフトウェア送信キューに入れられたことのある発信パケットの最大数。
S/W Transmit Queue Overflow	ソフトウェア送信キューからあふれた発信パケット数。
Current S/W+H/W Transmit Queue Length	ソフトウェア送信キューまたはハードウェア送信キュー上で保留されている発信パケットの数。
Broadcast Packets	エラーなしで送信されたブロードキャスト・パケット数。
Multicast Packets	エラーなしで送信されたマルチキャスト・パケット数。
No Carrier Sense	キャリアなし検知エラーにより失敗した送信回数。
DMA Underrun	DMA アンダーラン・エラーにより失敗した送信回数。
Lost CTS Errors	送信可信号が失われたことによるエラーのために失敗した送信回数。
Max Collision Errors	衝突が多すぎるために失敗した送信回数。衝突の発生回数が、アダプター上の再試行回数を超えました。
Late Collision Errors	後発衝突エラーにより失敗した送信回数。
Deferred	送信中に遅延された発信パケット数。遅延とは、アダプターがフレームを送信しようとして、延期しなければならなかったことを意味します。このような状況は、アダプターが送信できる状態になっていても、ネットワークがビジー状態である場合に起こります。アダプターは、最初の試行についてのみパケットの送信を遅延します。それ以降、アダプターはチェックしないでパケットを送信します。依然としてネットワークがビジー状態である場合は、衝突が記録されます。
SQE Test	送信中に正常に実行された「信号品質エラー」検査 (つまり、ハートビート) の回数が入っています。
Timeout Errors	アダプターがタイムアウト・エラーを報告したことにより失敗した送信回数。
Single Collision Count	送信中に衝突が 1 回発生した発信パケット数。
Multiple Collision Count	送信中に衝突が複数回 (2 から 15) 発生した発信パケット数。
Current HW Transmit Queue Length	現在ハードウェア送信キューに入っている発信パケット数。
CRC Errors	チェックサム (FCS) エラーがある受信パケット数。
DMA Overrun	DMA オーバーラン・エラーがある受信パケット数。
Alignment Errors	位置合わせエラーがある受信パケット数。
No Resource Errors	リソースなしエラーによりハードウェアがドロップした受信パケット数。通常、このエラーは、アダプター上の受信バッファーがすべて使い果たされたことが原因で発生します。アダプターによっては、受信バッファーのサイズが構成可能なパラメーターとなっている場合もあるので、調整に関して、デバイスの構成属性 (または smit のヘルプ) を調べてみてください。
Receive Collision Errors	受信中に衝突エラーが発生した受信パケット数。
Packet Too Short Errors	パケット・サイズがイーサネットの最小パケット・サイズよりも小さいことを示す長さエラーが付いた受信パケット数。
Packet Too Long Errors	パケット・サイズがイーサネットの最大パケット・サイズよりも大きいことを示す長さエラーが付いた受信パケット数。
Packets Discarded by Adapter	何からの原因でハードウェアがドロップした受信パケット数。
Receiver Start Count	アダプター上のレシーバー (受信装置) が始動された回数。

受信統計フィールド

項目	説明
Packets	デバイスによって正常に受信されたパケット数。
Bytes	デバイスによって正常に受信されたバイト数。
Interrupts	ドライバーがアダプターから受け取った受信割り込みの数。
Receive Errors	このデバイス上で発生した入力エラーの数。このフィールドは、ハードウェア/ネットワーク・エラーにより失敗した受信回数のカウンターです。
Packets Dropped	デバイス・ドライバーがこのデバイスから受信し、ネットワーク・デマルチプレクサーには (何らかの理由で) 提供されなかったパケットの数。
Bad Packets	デバイス・ドライバーによって受信 (つまり保管) された正しくないパケットの数。
Broadcast Packets	エラーなしで受信されたブロードキャスト・パケットの数。
Multicast Packets	エラーなしで受信されたマルチキャスト・パケットの数。
CRC Errors	チェックサム (FCS) エラーがある受信パケット数。
DMA Overrun	DMA オーバーラン・エラーがある受信パケット数。
Alignment Errors	位置合わせエラーがある受信パケット数。
No Resource Errors	リソースなしエラーによりハードウェアがドロップした受信パケット数。
Receive Collision Errors	受信中に衝突エラーが発生した受信パケット数。
Packet Too Short Errors	パケット・サイズがイーサネットの最小パケット・サイズよりも小さいことを示す長さエラーが付いた受信パケット数。
Packet Too Long Errors	パケット・サイズがイーサネットの最大パケット・サイズよりも大きいことを示す長さエラーが付いた受信パケット数。
Packets Discarded by Adapter	何からの原因でハードウェアがドロップした受信パケット数。
Receiver Start Count	アダプター上のレシーバー (受信装置) が始動された回数。

一般統計フィールド

項目	説明
No mbuf Errors	mbufs をデバイス・ドライバーが利用できなかった回数。これは、通常、インバウンド・パケットを処理するためにドライバーで mbuf バッファーを取得しなければならない場合、受信操作時に発生します。要求されたサイズの mbuf プールが空の場合には、パケットが廃棄されます。 **netstat -m** コマンドを使用すると、これを確認できます。
Adapter Reset Count	アダプターが再始動 (再初期化) された回数。
Adapter Data Rate	Mbps 単位 (メガビット/秒) の、アダプターの最大データ転送速度。
Driver Flags	現在オンになっているデバイス・ドライバーの内部状況フラグ。

デバイス固有の統計情報フィールド

この表示部分は、アダプターのタイプごとに異なる場合があります。アダプター固有の情報と、一般統計に含まれていなかった拡張統計が含まれていることがあります。アダプターによっては、デバイス固有の統計がないものもあります。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. **ent0** の一般統計を表示するには、次のように入力します。

   ```
   entstat ent0
   ```

これにより次の出力が生成されます。

```
ETHERNET STATISTICS (ent0) :
Device Type: Ethernet High Performance LAN Adapter
Hardware Address: 02:60:8c:2e:d0:1d
Elapsed Time: 0 days 0 hours 8 minutes 41 seconds

Transmit Statistics:     Receive Statistics:
--------------------     -------------------
Packets: 3               Packets: 2
Bytes: 272               Bytes: 146
Interrupts: 3            Interrupts: 2
Transmit Errors: 0       Receive Errors: 0
Packets Dropped: 0       Packets Dropped: 0
Max Packets on S/W       Bad Packets: 0
Transmit Queue:0
S/W Transmit Queue
Overflow: 0
Current S/W+H/W Transmit
Queue Length: 0

Broadcast Packets: 2     CRC Errors: 0
Multicast Packets: 0     Broadcast Packets: 1
No Carrier Sense: 0      Multicast Packets: 0
DMA Underrun: 0          DMA Overrun: 0
Lost CTS Errors: 0       Alignment Errors: 0
Max Collision Errors: 0  No Resource Errors: 0
Late Collision Errors: 0 Receive Collision Errors: 0
Deferred: 0              Packet Too Short Errors: 0
SQE Test: 0              Packet Too Long Errors: 0
Timeout Errors: 0        Packets Discarded by Adapter: 0
Single Collision         Receiver Start Count: 1
Count: 0
Multiple Collision Count: 0
Current HW Transmit Queue
Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 2000
Driver Flags: Up Broadcast Running Simplex
```

2. **ent0** に関して、イーサネット・デバイスの一般統計とイーサネット・デバイス固有の統計を表示するには、次のように入力します。

```
entstat -d ent0
```

これにより次の出力が生成されます。

```
ETHERNET STATISTICS (ent0) :
Device Type: Ethernet High Performance LAN Adapter
Hardware Address: 02:60:8c:2e:d0:1d
Elapsed Time: 0 days 2 hours 6 minutes 30 seconds

Transmit Statistics:      Receive Statistics:
--------------------      -------------------
Packets: 3                Packets: 2
Bytes: 272                Bytes: 146
Interrupts: 3             Interrupts: 2
Transmit Errors: 0        Receive Errors: 0
Packets Dropped: 0        Packets Dropped: 0
Max Packets on S/W        Receiver Start Count: 1
Transmit Queue:0
Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0
```

```
Broadcast Packets: 0      Broadcast Packets: 0
Multicast Packets: 0      Multicast Packets: 0
No Carrier Sense: 0       CRC Errors: 0
DMA Underrun: 0           DMA Overrun: 0
Lost CTS Errors: 0        Alignment Errors: 0
Max Collision Errors: 0   No Resource Errors: 0
Late Collision Errors: 0  Receive Collision Errors: 0
Deferred: 0               Packet Too Short Errors: 0
SQE Test: 0               Packet Too Long Errors: 0
Timeout Errors: 0         Packets Discarded by Adapter: 0
Single Collision Count: 0 Receiver Start Count: 1
Multiple Collision Count: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 2000
Driver Flags: Up Broadcast Running Simplex

Ethernet High Performance LAN Adapter Specific Statistics:
----------------------------------------------------------
Receive Buffer Pool Size: 37
Transmit Buffer Pool Size: 39
In Promiscuous Mode for IP Multicast: No
Packets Uploaded from Adapter: 0
Host End-of-List Encountered: 0
82586 End-of-List Encountered: 0
Receive DMA Timeouts: 0
Adapter Internal Data: 0x0 0x0 0x0 0x0 0x0
```

関連資料:

529 ページの『fddistat コマンド』

関連情報:

netstat コマンド

tokstat コマンド

---

env コマンド

目的

現在の環境を表示し、コマンド実行用の環境を設定します。

構文

複数の環境変数を表示する

env [**-i** | **-**] [*Name*=*Value*]... [*Command* [*Argument* ...]]

単一の環境変数を表示する

env [*Name*]

説明

env コマンドを使用すると、現在の環境を表示するか、または変更後の環境で指定したコマンドを実行できます。

フラグまたはパラメーターを指定しないと、**env** コマンドは現在の環境を *Name*=*Value* の組として 1 行に 1 つずつ表示します。

フラグ

項目	説明
-i	継承した環境を無視し、*Name*=*Value* パラメーターで指定した環境で、*Command* パラメーターで指定したコマンドを起動します。

パラメーター

項目	説明
Name=*Value*	1 つ以上の *Name*=*Value* パラメーターを指定すると、現行環境の変更バージョン内でコマンドを実行できます。現行環境全体を、指定した *Name*=*Value* パラメーターで置換したい場合は、**-i** フラグを使用します。どちらの場合も、環境の変更は指定したコマンドの実行中にのみ有効です。
コマンド	*Command* パラメーターには、オプションの *Argument* 変数が付いています。指定したコマンドが Korn シェルの特殊組み込みコマンドであれば、結果は未指定になります。 **ksh** シェルの組み込みコマンドについては、ksh コマンドのセクションを参照してください。

終了状況

Command パラメーターが指定されると、**env** コマンドの終了状況は *Command* パラメーターで指定したコマンドの終了状況となります。その他の場合、**env** コマンドの終了状況は次のいずれかです。

項目	説明
0	**env** コマンドは正常終了しました。
1-125	**env** コマンドでエラーが発生しました。
126	*Command* パラメーターで指定したコマンドは見つかりましたが、起動することはできません。
127	*Command* パラメーターで指定したコマンドが見つかりません。

例

1. **date** コマンドの実行中に **TZ** 環境変数を変更するには、次のように入力します。

   ```
   TZ=MST7MDT date
   ```

   または

   ```
   env TZ=MST7MDT date
   ```

   これらのコマンドは、山岳地帯標準時間と現在の日付を表示します。上の 2 つのコマンドは等価です。 **date** コマンドが終了すると、前の **TZ** 環境変数の値が再び有効となります。

2. **PATH**、**IDIR**、および **LIBDIR** 環境変数の定義のみで構成される環境で **make** コマンドを実行するには、次のように入力します。

   ```
   env -i PATH=$PATH IDIR=/$HOME/include LIBDIR=/$HOME/lib make
   ```

   シェルが **make** コマンドを見つけられるように、**PATH** 環境変数を指定する必要があります。 **make** コマンドが終了すると、前の環境が有効となります。

ファイル

項目	説明
/usr/bin/env	**env** コマンドが入っています。

関連情報:

printenv コマンド

environment コマンド

プロファイル・ファイル・フォーマット

プロファイル概要

---

epkg コマンド

目的

暫定修正マネージャー **emgr** でインストールできる暫定修正パッケージを作成します。

構文

epkg [**-w** *WorkDirectory*] [**-a** *APARrefFile*] [**-p** *PrerequisiteFile*] [**-d** *DescriptionFile*] [**-e** *interimfixControlFile*] [**-g** *PrerequisiteFile*] [**-l** *LockFile*] [**-S** *SupersedeFile*] [**-u** {**y**|**n**}]] [**-r** {**y**|**n**|**o**}] [**-s**] [**-T** {**y**|**n**}] [**-X**] [**-v**] *interimfixLabel*

説明

epkg ツールは、2 つの異なるモード (対話モード と テンプレート・ベース・モード) で実行することができます。対話モードでは、幾つかの質問がプロンプトされ、その応答に基づいて暫定修正パッケージが作成されます。テンプレート・ベース・モードでは、暫定修正制御ファイルが使用されます。この制御ファイルには、対話モードで尋ねられる質問に対する応答が用意されています。暫定修正 パッケージは、暫定修正 マネージャーによってインストールされます。マネージャーは、**emgr** コマンドによって始動されます。

対話モード

epkg コマンドは、デフォルトでは対話モードで実行されます。唯一の必須パラメーターは暫定修正ラベルです。**epkg** セッションを中断すると、暫定修正制御ファイルが保管されます。同じ暫定修正ラベルを使用して新規セッションを開始すると、前の暫定修正制御ファイルを用いて作業を続けたいかどうか尋ねられます。対話式 **epkg** セッションを開始する前にこの情報を提供するには、 **-u** フラグを指定した **epkg** を実行します。

epkg コマンドは質問順序のレコードを保持します。このコマンドにより、ユーザーはサブコマンドを使用して各質問の間をナビゲートすることができます。また、 **epkg** コマンドは、ユーザーが提供した前の応答を覚えており、その応答をデフォルトの応答として設定します。 **epkg** のサブコマンドについては、『サブコマンド』のセクションで説明しています。

すべての質問への応答後に、**epkg** コマンドは暫定修正制御ファイルを検査し、 **emgr** コマンドを使用してインストールできる圧縮された tar パッケージを作成します。

制御ファイル・テンプレートの使用

暫定修正制御ファイルをテンプレートとして使用して、暫定修正パッケージを非対話形式で作成することができます。以下は、完了した暫定修正制御ファイルの例を示したものです。

```
# 暫定修正 control file complete example
ABSTRACT=This is a test of epkg.
PRE_INSTALL=/tmp/pre_install
POST_INSTALL=.
PRE_REMOVE=/tmp/pre_remove
POST_REMOVE=.
REBOOT=yes
PREREQ=.
DESCRIPTION=/tmp/description
EFIX_FILES=2
APARREF=/tmp/aparref
LKU_CAPABLE=no

EFIX_FILE:
        EFIX_FILE_NUM=1
        SHIP_FILE=/home/test/ls
        TARGET_FILE=/usr/bin/ls
        TYPE= 1
        INSTALLER= 1
        ACL= DEFAULT
        AR_MEM=.

EFIX_FILE:
        EFIX_FILE_NUM=2
        SHIP_FILE=/home/test/mystrcat.o
        TARGET_FILE=/usr/ccs/lib/libc.a
        TYPE= 2
        INSTALLER= 1
        ACL= root:system:555
        AR_MEM=strcat.o
```

暫定修正制御ファイルの値は、次のとおりです。

ABSTRACT
暫定修正パッケージを簡潔に記述します。abstract は 38 バイト以内に制限されています。

PRE_INSTALL
インストールのプレビュー後およびすべての暫定修正ファイルのインストール前に実行されるスクリプトのロケーションを指定します。PRE_INSTALL スクリプトに障害が生じると、暫定修正パッケージのインストールが異常終了します。このコンポーネントはオプションです。

POST_INSTALL
すべての暫定修正ファイルが正常にインストールされた後で実行されるスクリプトのロケーションを指定します。このコンポーネントはオプションです。

PRE_REMOVE
除去のプレビュー後およびすべての暫定修正ファイルが除去操作時に除去される前に実行されるスクリプトのロケーションを指定します。このコンポーネントはオプションです。

POST_REMOVE
暫定修正ファイルが除去操作時に除去された後で実行されるスクリプトのロケーションを指定します。このコンポーネントはオプションです。

REBOOT
この暫定修正にリブートが必要かどうかを指定します。指定可能な値は yes または no です。この値が yes に設定されると、**emgr** コマンドは必要に応じてブート・イメージに変更を加え、インストール後のリブートを求めるメッセージを出します。

PREREQ
installp の前提条件を含むファイルのロケーションを指定します。このコンポーネントはオプションです。

APARREF

この暫定修正に関連付けられた APAR の参照番号および要約を含むファイルのロケーションを指定します。このファイルの各行には、APAR 参照番号、APAR 番号、および APAR の要約が含まれています。ファイルのフォーマットは、次のとおりです。

```
APAR reference|:|APAR number|:|APAR abstract
```

有効な APARREF ファイルを作成するには、すべてのフィールドが必要なわけではありません。特定のフィールドが不明または不要である場合は、単純に NONE を指定するか、そのフィールドをブランクにします。有効な APARREF ファイルの例をいくつか以下に示します。

例 **1**

```
123456|:|IV12345|:|This is the APAR abstract
789012|:|IV67890|:|This is another APAR abstract
```

例 **2**

```
123456|:|NONE|:|NONE
789012|:|NONE
345678
```

例 **3**

```
NONE|:|IV12345|:|This is the APAR abstract
```

例 **4**

```
NONE
```

APAR 参照ファイルを APAR 参照番号と一緒に提供すると、**installp** コマンドによる自動除去機能が、暫定修正に対して有効になります。この **installp** コマンドによる自動除去とは、**installp** が適用中のテクノロジー・レベル、Service Pack、あるいは PTF に該当の修正が含まれている場合に、暫定修正を自動的に除去する機能を指します。 APAR 参照フィールドに NONE がリストされている場合、暫定修正に対して自動除去機能は有効になりません。

DESCRIPTION

インストールされる暫定修正パッケージの詳細記述を含むファイルのロケーションを指定します。

EFIX_FILES

暫定修正内のファイルの総数を指定します。

EFIX_FILE_NUM

暫定修正内のファイルの番号を指定します。暫定修正内の各ファイルは、1 から 400 までの固有の番号をもっていなければなりません。**epkg** コマンドでは、暫定修正ごとに最大 400 個のファイルをサポートすることができます。

SHIP_FILE

epkg が暫定修正パッケージ内にアーカイブするファイルのロケーションを指定します。このファイルに絶対パスまたは相対パスを指定することができます。

TARGET_FILE

SHIP_FILE がインストールされるロケーションを指定します。このロケーションは、暫定修正パッケージがインストールされるシステム上にあります。このファイルへの絶対パスを指定する必要があります。このファイルが登録済みパッケージ (例えば、RPM Package Manager (RPM) または **installp** パッケージ) の一部である場合は、追跡されるロケーションを指定する必要があります。

TYPE インストールされるファイルのタイプを指定します。有効な選択項目は次のとおりです。

1 ファイル (標準ファイルまたは実行可能ファイル)

2 ライブラリーまたはアーカイブ・メンバー

INSTALLER
暫定修正パッケージを追跡するインストーラー (存在する場合) のタイプを指定します。有効な選択項目は次のとおりです。

1 **installp** によって現在追跡される

2 RPM によって現在追跡される

3 **ISMP** によって現在追跡される

4 別のインストーラーによって現在追跡される

5 これは **installp** によって追跡される新規ファイルです

6 これは RPM によって追跡される新規ファイルです

7 これは **ISMP** によって追跡される新規ファイルです

8 これは別のインストーラーによって追跡される新規ファイルです

9 どのインストーラーによっても追跡されない

ACL ファイルのアクセス属性 (モードおよび所有権) を指定します。この属性が **DEFAULT** に設定されていると、**emgr** コマンドは、置換されるファイルの現行アクセス権を保持します。ただし、ターゲット・ファイルが新規ファイルの場合またはユーザーが **-v** フラグを使用してアクセス権を指定したい場合は、次のような構文 *Owner*:*Group*:*OctalModes* を使用して **ACL** 属性を入力することができます。

```
ACL= root:system:555
```

AR_MEM
アーカイブ・メンバーの名前を指定します。このオプションは、 **TYPE**=2 の場合にのみ有効です。この場合、 **SHIP_FILE** は出荷されるアーカイブ・メンバーのローカル・ロケーションであり、 **TARGET_FILE** はターゲット・アーカイブであり、さらに **ACL** はアーカイブ・メンバーに適用されます。例えば、以下の値を設定すると、ローカル・ファイル **myshr.o** はターゲット・アーカイブ **/usr/ccs/lib/libc.a** 内のメンバー **shr.o** になります。

```
TYPE=2
SHIP_FILE=/home/myshr.o
TARGET_FILE=/usr/ccs/lib/libc.a
AR_MEM=shr.o
```

BUILD_BOOT_IMAGE
ブート・イメージを再作成する必要があるかどうかを指定します。指定可能な値は yes または no です。このフィールドを yes に設定した場合は、リブートが必要です。このフィールドを yes に設定し、かつ、**REBOOT** フィールドを no に設定した場合は、**epkg** はエラーを戻します。

E2E_PREREQ
暫定修正制御ファイル内の暫定修正前提条件ファイルの場所を指定します。

PKGLOCKS
暫定修正制御ファイル内のパッケージ・ロック・ファイルのローカル・ファイル場所を指定します。

SUPERSEDE
暫定修正制御ファイル内の置き換えられるファイルのローカル・ファイル場所を指定します。

FIXTESTED
この暫定修正がテストされたかどうかを指定します。指定できる値は yes または no です。

LKU_CAPABLE

この暫定修正がLive Update操作と互換性があることを指定します。この属性の値は yes または no です。理想的には、すべての暫定修正がLive Update対応としてマーク付けされている必要があります。この互換性は、複数の暫定修正を 1 つのグループとしてインストールするために必要です。暫定修正がLive Update操作に適さない場合、LKU_CAPABLE 属性の値は no に設定されますが、ほとんどの暫定修正では、この属性の値は yes に設定されます。

置き換えのためのサポート

パッケージャーは、epkg がインストールされるときに置き換えられるべき暫定修正のラベル名を含むファイルを指定できます。これによって、**emgr** コマンドは、このファイルに指定されている暫定修正ラベルがあれば (それらがインストールされている場合)、暫定修正パッケージをインストールする前にそれらをすべて除去します。インストール済みの置き換えられる暫定修正の除去に失敗すると、暫定修正パッケージのインストールは異常終了します。置き換えられるラベルは、最大 32 までサポートされます。パッケージャーは、**epkg** コマンドで置き換えファイルを次のように指定できます。

- ファイル場所は **-S** ***supersede_file*** フラグで指定します。次に例を示します。

  ```
  epkg -S /tmp/superseded.epkg myefix
  ```

- **epkg** コマンドは、拡張オプション・フラグ (**-v**) が対話モードで使用されていれば、置き換えられるファイルを求めるプロンプトを出します。次に例を示します。

  ```
  Enter the location for the supersede file or "." to skip.
  -> /tmp/superseded.epkg
  ```

- 暫定修正制御ファイル内の置き換えられるファイルのローカルのファイル場所に、 SUPERSEDE 属性を設定します。次に例を示します。

  ```
  SUPERSEDE=/tmp/superseded.epkg
  ```

置き換えられるファイルのフォーマットでは、置き換えられる暫定修正ラベル 1 つが 1 行に示されます。# 記号で始まるコメントと、先頭の空白は無視されます。次に例を示します。

```
# Requisites for efix myefix3
myefix1
myefix2
```

prereq および **xreq** のサポート

パッケージャーは、暫定修正パッケージをインストールするための必要条件としての暫定修正の暫定修正ラベル名を含むファイルを指定することができます。これによって、**emgr** コマンドは、その暫定修正ラベルがインストールされているか検査します (PREREQ)。この必要条件がインストールされていないと、**emgr** コマンドは暫定修正パッケージのインストールを異常終了します。ユーザーは、また、XREQ 暫定修正ラベルも指定できます。これによって、**emgr** コマンドは、指名された xreq 暫定修正がインストールされている場合はその暫定修正をインストールしない ようになります。

パッケージャーは、次のようにして **epkg** コマンドにより暫定修正前提条件ファイルを指定できます。

- ファイル場所を **-g** ***efix_prereq_file*** フラグで指定します。次に例を示します。

  ```
  epkg -g /tmp/efixprereq.epkg myefix
  ```

- **epkg** コマンドは、拡張オプション・フラグ (**-v**) が対話モードで使用されていれば、暫定修正前提条件ファイルを求めるプロンプトを出します。次に例を示します。

  ```
  Enter the location for the efix prerequisite file or "." to skip.
  -> /tmp/efixprereq.epkg
  ```

- 暫定修正制御ファイル内の暫定修正前提条件ファイルのローカル・ファイル場所に、E2E_PREREQ 属性を設定します。次に例を示します。

```
E2E_PREREQ=/tmp/efixprereq.epkg
```

暫定修正前提条件ファイル・エントリーのフォーマットは、次のとおりです。

```
EfixLabel  RequisiteType: PREREQ/XREQ
```

次に例を示します。

```
oldefix1 PREREQ  # Make sure oldefix1 is already installed

oldefix4 XREQ    # Make sure oldefix4 is NOT installed
```

暫定修正前提条件は、最大 32 までサポートされます。

installp による暫定修正の自動除去のサポート

パッケージャーは、APAR 参照ファイルに APAR 参照番号を含めるように指定することができます。APAR 参照番号を使用することにより、修正が出荷されたすべてのテクノロジー・レベルについて、**installp** は暫定修正から APAR にマップし直すことができます。適用されるテクノロジー・レベル、Service Pack、あるいは PTF にこの暫定修正が含まれていると **installp** が判断した場合には、**installp** は更新を適用するまえにこの暫定修正を自動的に除去します。

出力とトポロジー

emgr -d フラグは、暫定修正パッケージの内容とトポロジーを表示します。**-d** オプションは、**-v** 詳細オプションとともに働きます。詳細の有効なレベルは 1 から 3 です。

詳細レベル 1 (デフォルト) が表示するのは、次のものです。

- LABEL
- EFIX FILES
- TARGET LOCATION

詳細レベル 2 が表示するのは、次のものです。

- レベル 1 のすべての出力
- ABSTRACT
- REBOOT
- PRE-REQUISITES
- PRE_INSTALL
- POST_INSTALL
- PRE_REMOVE
- POST_REMOVE
- FILE TYPE

詳細レベル 3 が表示するのは、次のものです。

- レベル 2 のすべての出力
- PACKAGING DATE
- VUID
- SIZE
- ACL

- CKSUM
- PACKAGE
- EFIX DESCRIPTION
- CONTENTS OF INSTALL SCRIPTS (テキスト・ファイルの場合)

次に例を示します。

- 暫定修正パッケージ test.102403.epkg.Z についてレベル 1 の詳細出力を取得するには、次のように入力します。

  ```
  emgr -d test.102403.epkg.Z
  ```

- 暫定修正パッケージ test.102403.epkg.Z についてレベル 3 の詳細出力を取得するには、次のように入力します。

  ```
  emgr -v3 -d test.102403.epkg.Z
  ```

追加のパッケージ・ロックのサポート

パッケージャーは、**emgr** コマンドでロックすべきパッケージ名およびファイル所有権に基づいて自動的にロックされるパッケージ名を含むファイルを指定できます。パッケージャーは、パッケージの名前、インストーラー、およびパッケージ・ロック・アクションのタイプ (ALWAYS/IFINST) を指定する必要があります。パッケージャーは、このパッケージ・ロック・ファイルを次のようにして **epkg** コマンドで指定できます。

- ファイル場所を **-l** ***pkg_locks_file*** フラグで指定します。次に例を示します。

  ```
  epkg -l /tmp/pkglock.epkg myefix
  ```

- **epkg** コマンドは、拡張オプション・フラグ (**-v**) が使用されていれば、パッケージ・ロック・ファイルを求めるプロンプトを出します。次に例を示します。

  ```
  Enter the location for the package locks file or "." to skip.
  -> /tmp/pkglock.epkg
  ```

- 暫定修正制御ファイル内のパッケージ・ロック・ファイルのローカル・ファイル場所に、PKGLOCKS 属性を設定します。次に例を示します。

  ```
  PKGLOCKS=/tmp/pkglock.epkg
  ```

パッケージ・ロック・ファイルのフォーマットは、次のとおりです。

```
PackageName PackageAction PackageType
```

ここで、*PackageName* はロックされるパッケージの名前、*PackageAction* は以下のいずれかです。

項目	説明
ALWAYS	このパッケージを常時ロックしようとします。パッケージのロックに失敗すると、その結果、インストールは失敗します。
IFINST	パッケージがインストールされている場合のみ、このパッケージをロックしようとします。インストール済み パッケージのロックに失敗すると、その結果、インストールは失敗します。

PackageType は installp (デフォルト)、rpm、ISMP、other です。

注**:** installp のロックのみがサポートされています。

パッケージ・ロックは、最大 32 までサポートされます。

例:

```
bos.rte.lvm ALWAYS installp
bos.games   IFINST installp
```

上記の例では、**emgr** コマンドは、インストール時は常に **bos.rte.lvm** をロックし、除去時にはそれをアンロックします。また、**emgr** コマンドは、**bos.games** については、これがインストールされている場合のみロックし、(ロックした場合は) 除去時にそれをアンロックします。

bosboot オプションのサポート

epkg コマンドの reboot オプションには、ブート・イメージを再ビルドしないリブートが含まれています。

ユーザーは、以下のようにして、bosboot なしのリブートを指定できます。

- **epkg -r** フラグの o 引数は、リブート「のみ」が必要であって、**emgr** コマンドは bosboot を呼び出さない (すなわちブート・イメージを再ビルドしない) ことを指示します。
- 対話モードのリブート・プロンプトは、次の選択肢を示します。

  ```
  Select reboot policy for this efix package:
  1) Reboot is NOT required.
  2) Reboot is required. ブート・イメージが再ビルドされます。
  3) Reboot is required. The boot image will NOT be rebuilt.
  ```

- 暫定修正制御ファイル内で BUILD_BOOT_IMAGE と REBOOT の属性を「yes」または「no」に設定します。以下の REBOOT および BUILD_BOOT_IMAGE オプションがサポートされています。

項目	説明
REBOOT=no & BUILD_BOOT_IMAGE=no	リブートは必要ない。
REBOOT=yes & BUILD_BOOT_IMAGE=yes	リブートが必要。ブート・イメージが再ビルドされます。
REBOOT=yes & BUILD_BOOT_IMAGE=no	リブートが必要。ブート・イメージは再ビルドされません。

注: `REBOOT=no & BUILD_BOOT_IMAGE=yes` は、**epkg** コマンドでエラーとなります。

フラグ

項目	説明
-a *APARrefFile*	APAR 参照番号を含むファイルを指定します。
-d *DescriptionFile*	暫定修正記述を含むファイルを指定します。
-e *interimfixControlFile*	暫定修正の作成方法を制御する暫定修正制御ファイルを指定します。
-g *PrerequisiteFile*	暫定修正ラベル名を含む暫定修正前提条件ファイルの場所を指定します。これらのラベルは、暫定修正パッケージのインストール前に必要です。
-l *LockFile*	パッケージ名が入っているロックされたファイルの位置を指定します。これらのパッケージは、**emgr** コマンドによってロックされるか、またはファイル所有権に基づいて自動的にロックされます。
-p *PrerequisiteFile*	installp の前提条件を含むファイルを指定します。
-r {y|n|o}	**epkg REBOOT** 属性を設定します。この値が yes に設定されると、 **emgr** コマンドは必要に応じてブート・イメージに変更を加え、インストール後のリブートを指示するメッセージを出します。 **y** 引数は、リブートおよび **bosboot** が必要であることを指定します。 **n** 引数は、リブートが不要であることを指定するためのものです。**o** 引数はリブートが必要であることを示しますが、**emgr** は **bosboot** を呼び出すべきではありません。
-S *SupersedeFile*	暫定修正ラベル名を含む暫定修正置換ファイルの場所を指定します。これらのラベルは、**epkg** のインストール時に置き換えられます。
-s	**epkg** コマンドがスクリプトに関する質問および前提条件ファイルをスキップするようにします。
-T	この暫定修正がテストされたかどうかを指定します。指定可能な値は yes または no です。デフォルトは no です。
-u {**yes**|**no**}	既存の暫定修正制御ファイルを使用するか否かを指定します。

項目	説明
-v	**epkg** コマンドが拡張オプションに関してさらに質問を加えます。これには、すべての暫定修正ファイルに関するアクセス権の指定が含まれます。
-w *WorkDirectory*	**epkg** コマンドが使用する代替作業ディレクトリーを指定します。デフォルトの作業ディレクトリーは **$HOME/epkgwork** です。
-X	暫定修正のインストール時に **emgr** コマンドがファイルシステムを自動的に拡張します (スペースが必要とされ、拡張が可能な場合)。

パラメーター

暫定修正*Label*

この暫定修正パッケージを一意的に識別する文字列を指定します。暫定修正ラベルの最大長は 10 バイトです。

注**:** 暫定修正マネージャーでは、システム上の各暫定修正ラベルが固有である必要があります。

サブコマンド

b! 前の質問に戻ります。

s! 現行の暫定修正制御ファイルの状況を表示します。

q! 暫定修正制御ファイルを保管せずに終了します。(Ctrl+C キー・シーケンスを使用すると、 **epkg** コマンドは、暫定修正制御ファイルを保管したいかどうかを尋ねます。)

h! 現行の質問のヘルプ情報を表示します。

終了状況

0 **epkg** コマンド操作が正常に完了しました。

>0 エラーが発生しました。

例

1. **epkg** コマンドを対話モードで実行し、暫定修正ラベル **myfix** をもつ暫定修正パッケージを作成するには、次のように入力します。

   ```
   epkg myfix
   ```

2. **/tmp/ecfile** という名前の既存の暫定修正制御ファイルを使用して暫定修正ラベル **myfix** をもつ暫定修正パッケージを作成するには、次のように入力します。

   ```
   epkg -e /tmp/ecfile myfix
   ```

3. 暫定修正ラベル **myfix** をもつ暫定修正パッケージを作成し、前提条件ファイル **/tmp/prereq**、記述 **/tmp/description**、および拡張オプションを指定するには、次のように入力します。

   ```
   epkg -v -p /tmp/prereq -d /tmp/description myfix
   ```

ファイル

項目	説明
/usr/sbin/epkg	**epkg** コマンドが入っています。

関連資料:

371 ページの『emgr コマンド』

関連情報:

オプションのソフトウェア・プロダクトおよび保守更新のインストール

---

eqn コマンド

目的

troff コマンドの数式テキストをフォーマットします。

構文

eqn [**-d** *Delimiter1Delimiter2*] [**-f** *Font*] [**-p** *Number*] [**-s** *Size*] [**-T** *Name*] [**—**] [*File ...* | **-**]

説明

eqn コマンドは、フォトタイプセッターまたはそれと同等のデバイス上に数式テキストを作成するための **troff** プリプロセッサーです。次に示すように、**eqn** コマンドの出力は普通は **troff** コマンドにパイプ接続されます。

eqn [*Flag...*] *File...* | **troff** [*Flag...*] | [*Typesetter*]

eqn コマンドは、*File* パラメーターによって指定されたファイルを読み込みます。 - (ハイフン) が最終パラメーターに指定されている場合は、このコマンドは標準入力を読み込みます。 **.EQ** マクロで始まる行により、数式テキストの開始を表します。また、**.EN** マクロで始まる行により、数式テキストの終了を表します。これらの行は、**troff** コマンドでは変更されないので、例えば、センタリングやナンバリングなどの補足的なフォーマット化機能を備えるようマクロ・パッケージで定義することができます。

キーワード

次のキーワードは **eqn** コマンドと **neqn** コマンドの両方によって認識されます。

```
above     dot      gsize     over      tdefine
back      dotdot   hat       pile      tilde
bar       down     italic    rcol      to
bold      dyad     lcol      right     under
ceiling   fat      left      roman     up
ccol      floor    lineup    rpile     vec
col       font     lpile     size
cpile     from     mark      sqrt
define    fwd      matrix    sub
delim     gfont    ndefine   sup
```

eqn コマンドによって認識されるキーワードは、スペース、タブ、改行文字、中括弧、二重引用符、ティルド、脱字記号などとは別に設定できます。 { } (中括弧) は、グループ化に使用します。X などの単一文字を使用して、中括弧で囲んだ複雑な構造の代わりにすることができます。~ (ティルド) は出力でのフル・スペースを表し、^ (脱字記号) はハーフ・スペースを表します。

添え字と肩文字は、キーワード **sub** と **sup** で作成します。分数はキーワード **over** を使って作成します。平方根はキーワード **sqrt** で作成します。

下限/上限は、キーワード **from** と **to** で作成します。正しい高さの区切り文字 (左右の大括弧や中括弧など) は、キーワード **left** と **right** で作成します。キーワード **left** と **right** の後に指定できる文字は、中括弧、大括弧、縦線、最高と最低を示す **c** と **f**、まったく何もないことを示す **" "** (二重引用符) (右側のみの括弧に便利) です。 **left** 文字にはそれに対応する **right** 文字は必要ありませんが、**right** 文字はそれに対応する **left** 文字がなければなりません。

垂直方向のリスト (積み上げ) は、**pile**、**lpile**、**cpile**、**rpile** の各キーワードで作成されます。積上げは、任意の数のエレメントで作成できます。キーワード **lpile** は左そろえ、キーワード **pile** およびキーワード **cpile** は中央そろえ (ただし、行間スペースが異なる)、キーワード **rpile** は右そろえです。マトリックスはキーワード **matrix** で作成されます。さらに、列を右そろえにするキーワード **rcol** があります。

区別的発音符は、**dot**、**dotdot**、**hat**、**tilde**、**bar**、**vec**、**dyad**、**under** の各キーワードで作成されます。

ポイント・サイズとフォントは、**size** *Number* (または **size +/-***Number*)、**roman**、**italic**、**bold**、および **font** *Number* の各キーワードで変更できます。**gsize** *Number* キーワードと **gfont** *Number* キーワードを使用するか、コマンド・ラインの **-s***Number* フラグと **-f***Number* フラグを使用すると、ドキュメント全体でポイント・サイズとフォントを変更できます。

通常、添え字と肩文字は、元のサイズから 3 ポイント減らされます。これは、コマンド・ラインで、**-p***Number* フラグを指定すると、変更できます。

連続表示するパラメーターを縦方向に整列させることができます。キーワード **mark** を最初の数式の希望する整列ポイントの前に置いて、キーワード **lineup** をその後の数式で縦に整列する場所に置きます。

define キーワードを使うと、省略表現を定義したり、既存のキーワードの再定義ができます。次に例を示します。

```
define  Thing%Replacement%
```

上記の例は、*Thing* という新しいトークンを定義しています。このトークンは、その後表示される際に常に *Replacement* に置き換えられます。 % (パーセント記号) は、*Replacement* 内に含まれていない文字であれば何でもかまいません。

sum、**int**、**inf** のようなキーワードと、**>=**、**!=**、および **->** などの省略形が認識されます。ギリシャ文字は、**alpha** や **GAMMA** のように望ましい大文字/小文字で表示されます。 **sin**、**cos**、**log** などの数学関係の語は、自動的に Roman 体となります。二重剣標を作成する **¥(dd** のような **troff** コマンドの四文字エスケープは、どこでも使用できます。 **" "** (二重引用符) で囲まれた文字列は、処理されないまま渡されます。これによって、キーワードをテキストとして入力でき、常に **troff** コマンドと通信するのに使用できます。

フラグ

項目	説明
-d*Delimiter1Delimiter2*	2 つの ASCII 文字 *Delimiter1* と *Delimiter2* を **.EQ** マクロと **.EN** マクロで囲まれた入力に加えて、**eqn** コマンドで処理するテキストの区切り文字として設定します。これらの区切り文字の間のテキストは、**eqn** コマンドへの入力として扱われます。 注**:** ファイル内では、**delim** *Delimiter1Delimiter2* コマンドを使用して、**eqn** テキストの区切り文字を設定することもできます。これらの区切り文字は、**delim off** コマンドでオフにすることができます。 **.EQ** マクロと **.EN** マクロの間にないすべてのテキストは、処理されずにパスされます。

項目	説明
-f*Font*	**eqn** コマンドで処理した全テキストのフォントを、 *Font* 変数で指定した値に変更します。 *Font* の値 (フォント名または位置) は、1 文字または 2 文字の ASCII 文字でなければなりません。
-p *Number*	添え字と肩文字のサイズを指定のポイント数分縮小します (デフォルトは 3)。
-s *Size*	**eqn** コマンドで処理した全テキストのポイント・サイズを、*Size* 変数で指定した値に変更します。
-T*Name*	指定のプリンターへの出力を準備します。フォトタイプセッターまたはそれと同等のデバイスの端末名によって、*Name* 変数が与えられます。デフォルトは **ibm3816** です。
-	入力を標準入力から強制的に読み取ります。
—	(二重ダッシュ) フラグの終わりを示します。

ファイル

項目	説明
/usr/share/lib/pub/eqnchar	特殊文字の定義が入っています。

関連情報:

mvt コマンド

neqn コマンド

troff コマンド

eqnchar ファイル・フォーマット

---

errclear コマンド

目的

エラー・ログからエントリーを削除します。

構文

errclear [**-d** *ErrorClassList*] [**-i** *File*] [**-J** *ErrorLabel* [*,Errorlabel*]] | [**-K** *ErrorLabel* [*,Errorlabel*]] [**-l** *SequenceNumber*] [**-m** *Machine*] [**-n** *Node*] [**-N** *ResourceNameList*] [**-R** *ResourceTypeList*] [**-S** *ResourceClassList*] [**-T** *ErrorTypeList*] [**-y** *FileName*] [**-j** *ErrorID* [**,***ErrorID*]] | [**-k** *ErrorID* [**,***ErrorID*]] *Days*

説明

errclear コマンドは、*Days* パラメーターで指定した日数よりも古いエラー・ログ・エントリーを削除します。すべてのエラー・ログ・エントリーを削除するには、*Days* パラメーターの値を **0** に指定します。

errclear コマンドで **-i** フラグを指定しないと、 **errclear** によってクリアされるエラー・ログ・ファイルは、エラー・ログ構成データベース内で指定したファイルになります。 (エラー・ログ構成データベース内の情報を表示するには、 **errdemon** コマンドを使用します。)

> 注**: errclear** コマンドは、指定したエントリーをクリアしますが、エラー・ログ・ファイルのサイズは減らしません。

System Management Interface Tool (SMIT) の **smit errclear** 高速パスを使用して、このコマンドを実行できます。

フラグ

項目	説明
-d *List*	*List* 変数で指定したエラー・クラス内のエラー・ログ・エントリーを削除します。 *List* 変数の値は，(コンマ) で区切るか、" " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。有効な *List* 変数の値は、**H** (ハードウェア)、 **S** (ソフトウェア)、**0** (**errlogger** メッセージ)、および **U** (不定) です。
-i *File*	*File* 変数で指定したエラー・ログを使用します。このフラグを指定しないと、**errclear** コマンドはエラー・ログ構成データベースからの値を使用します。
-j *ErrorID*[,*ErrorID*]	*ErrorID* (エラー ID) で指定したエラー・ログ・エントリーを削除します。 *ErrorID* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-J *ErrorLabel*	*ErrorLabel* で指定したエラー・ログ・エントリーを削除します。*ErrorLabel* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-k *ErrorID*[,*ErrorID*]	*ErrorID* (エラー ID) で指定したエラー・ログ・エントリーを除き、すべてのエラー・ログ・エントリーを削除します。 *ErrorID* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-K *ErrorLabel*	*ErrorLabel* で指定したエラー・ログ・エントリーを除き、すべてのエラー・ログ・エントリーを削除します。 *ErrorLabel* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-l *SequenceNumber*	指定したシーケンス番号が付いているエラー・ログ・エントリーを削除します。 *SequenceNumber* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-m *Machine*	*Machine* 変数で指定したマシンのエラー・ログ・エントリーを削除します。**uname -m** コマンドは、*Machine* 変数の値を戻します。
-n *Node*	*Node* 変数で指定したノードのエラー・ログ・エントリーを削除します。**uname -n** コマンドは、*Node* 変数の値を戻します。
-N *List*	*List* 変数で指定したリソース名に関するエラー・ログ・エントリーを削除します。 *List* 変数は、エラーを検出したリソース名のリストです。ソフトウェア・エラーの場合は、エラーを検出したリソース名です。ハードウェア・エラーの場合は、デバイス名またはシステム・コンポーネントの名前です。これは、コンポーネントに欠陥があるとか、置換が必要であることを示しているのではありません。むしろ、エラー分析のために適切な診断モジュールを決定するのに使用されます。 *List* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-R *List*	*List* 変数で指定したリソース・タイプに関するエラー・ログ・エントリーを削除します。ハードウェア・エラーの場合、*List* 変数はデバイス・タイプです。ソフトウェアの場合、*List* 変数の値は **LPP** です。*List* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-S *List*	*List* 変数で指定したリソース・クラスに関するエラー・ログ・エントリーを削除します。ハードウェア・エラーの場合、*List* 変数はデバイス・クラスです。 *List* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-T *List*	*List* 変数で指定したエラー・タイプに関するエラー・ログ・エントリーを削除します。有効な *List* 変数値は、**PERM**、**TEMP**、**PERF**、 **PEND**、**INFO**、および **UNKN** です。 *List* 変数の値は，(コンマ) で区切るか、 " " (二重引用符) で囲んで，(コンマ) かスペース文字で区切ることができます。
-y *FileName*	*FileName* 変数で指定したエラー・レコード・テンプレート・ファイルを使用します。

セキュリティー

アクセス制御: root ユーザーだけがこのコマンドを実行できます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. エラー・ログからすべてのエントリーを削除するには、次のように入力します。

   ```
   errclear 0
   ```

2. ソフトウェア・エラーと分類されたエラー・ログ内の、すべてのエントリーを削除するには、次のように入力します。

   ```
   errclear -d S 0
   ```

3. 代替エラー・ログ・ファイル、 `/var/adm/ras/errlog.alternate` からすべてのエントリーを消去するには、次のように入力します。

   ```
   errclear -i /var/adm/ras/errlog.alternate 0
   ```

4. 代替エラー・ログ・ファイル、 `/var/adm/ras/errlog.alternate` から、すべてのハードウェア・エントリーを消去するには、次のように入力します。

   ```
   errclear -i /var/adm/ras/errlog.alternate -d H 0
   ```

ファイル

項目	説明
/etc/objrepos/SWservAt	エラー・ログ構成データベースであるソフトウェア・サービス支援属性オブジェクト・クラスが入っています。

関連資料:

434 ページの『errlogger コマンド』

関連情報:

uname サブルーチン

errlog コマンド

エラー・ログの概要

---

errctrl コマンド

目的

システム・コンポーネントのエラー検査属性を変更したり表示したりします。永続属性値は、まだ作成されていないコンポーネントに対しても指定できます。

構文

errctrl [**-nru**] *ComponentSelector ... subcommand ...*

errctrl -p [**-ru**] *ComponentSelector ... subcommand ...*

errctrl -P [**-ru**] *ComponentSelector ... subcommand ...*

errctrl -x { **-P** | **-p** } [**-ru**] *ComponentSelector ...*

errctrl -q [**-rupP**] {*ComponentSelector ...*}

errctrl {**-h** | **-?**}

errctrl -P {**errcheckon** | **errcheckoff**}

説明

errctrl コマンドは、一部またはすべてのコンポーネントのエラー検査属性値を変更したり表示したりします。コンポーネントは、名前、別名、またはタイプやサブタイプで選択されます。

ComponentSelector のサポートされる値は次のとおりです。

-c componentPatternList

-l aliasPatternList

-t typePatternList

これらのリストは、引用符で囲まれたスペースまたはコンマで区切られた 1 つ以上のパターンから構成されます。パターンには **fnmatch** サブルーチンで記述されている特殊文字を指定できます。パターンの文字では、疑問符 (?)、アスタリスク (*)、大括弧 ([ ]) がサポートされていますが、大括弧 ([ ]) の中で文字クラスと照合シーケンスを使用することはできません。**-c all** を指定すると、すべてのコンポーネントが選択されます。ほかに指定がない場合は、*ComponentSelector* が使用されます。

errctrl コマンドを **-p** または **-P** フラグと一緒に使用すると、永続属性のカスタマイズを指定できます。永続属性のカスタマイズについて詳しくは、『永続カスタマイズ』を参照してください。

すべてのコンポーネントに対して、エラー検査を即時および永続的に使用可能または使用不可にするには、**errcheckon** または **errcheckoff** サブコマンドを **-P** フラグと一緒に指定します。この形式のコマンドでは、他のフラグまたはサブコマンドは使用できません。再始動後も設定を永続的にするには、**bosboot** コマンドが必要です。

変更される属性は、コマンド・ラインに指定されるサブコマンドにより決まります。1 つの呼び出しに複数のサブコマンドを指定することができます。以下のサブコマンドが使用可能です。

項目	説明
errcheckon	エラー検査をオンにします。
errcheckoff	エラー検査をオフにします。
errcheckminimal	エラー検査レベルを 1 に設定します。
errchecknormal	エラー検査レベルを 3 に設定します。
errcheckdetail	エラー検査レベルを 7 に設定します。
errchecklevel={0-9}	エラー検査レベルを指定の値に設定します。
lowsevdisposition={disp}	重大度が低いエラーの後処理を指定の値に設定します。
medsevdisposition={disp}	重大度が中程度のエラーの後処理を指定の値に設定します。

disp エラー後処理は、以下のいずれかの値になります。

- ignore (または 48)
- log (または 64)
- livedump (または 80)
- isolate (または 96)
- sysdump (または 112)

他のサブコマンドは、個々のコンポーネントで認識されます。コンポーネントで認識されないサブコマンドは無視されます。

現行の属性値は、**-q** フラグを使用して表示できます。*ComponentSelector* が使用されていない場合は、エラー検査がサポートされているすべてのコンポーネントに対して属性値が表示されます。

ネットワーク・メモリーのメモリー・オーバーレイ検出システムは、netmalloc コンポーネントに詳細なエラー・レベルを設定して使用可能にすることができます。すべてのネットワーク・メモリー割り当てとフリー・イベントに関する詳細なネットワーク・メモリー・ポリス・バッファー情報を収集するには、netmalloc コンポーネントのエラー・レベルを 5 以上 (デフォルト: 3 (normal)) に上げます。エラー・レベルを 7 (detail) 以上に上げても、ネットワーク・メモリーのオーバーレイ検出システムを使用可能にすることができます。すべてのネットワーク・メモリー割り当てとフリー・イベントに対して **net_malloc_police** オプションとアウトスタンディング・メモリー割り当て (OSTD) ロギングのみを使用可能にするには、エラー・レベルを 5 に上げます。

エラー・レベルの変更の詳細については、 426 ページの『例』を参照してください。トレース・レベルを上げて netmalloc コンポーネントでトレース・データを収集する場合、その詳細については **ctctrl** コマンドを参照してください。

このコマンドを使用すると、可能性 (頻度) の値を次の **netmalloc** 関数に設定できます。

- police_frequency
- frag_mask

可能性は 1024 からの分子 (10%: 102、5%: 51、1%: 10、0.1%: 1 など) です。

永続カスタマイズ

-p フラグと **-P** フラグにより、まだ作成されていないシステム・コンポーネントに属性値を指定できます。このため、新規に作成されるコンポーネントの属性は、コンポーネントがアクティブになる前にカスタマイズすることができます。**-p** フラグは、AIX の再始動前に将来的に作成されるコンポーネントのカスタマイズを指定するために使用します。**-P** フラグは、次回の再始動後に有効になるカスタマイズを指定するために使用します。これらのカスタマイズは、**/var/adm/ras/raspertune** ファイルに追加されます。これらのカスタマイズをブート・イメージに保管するには **bosboot** コマンドを実行し、カスタマイズを有効にするには AIX を再始動する必要があります。

ComponentSelector にはパターン・マッチング文字を使用できます。このように、永続的なカスタマイズを複数のコンポーネントに適用することができます。また、複数の異なる *ComponentSelector* を使用することで、複数のカスタマイズを同一のコンポーネントに適用できます。競合する属性値が複数のカスタマイズに指定されている場合は、最後のカスタマイズが優先されます。指定された *ComponentSelector* にカスタマイズが既に存在する場合は、新しいカスタマイズで古いカスタマイズが置き換えられます。

永続カスタマイズが指定される場合は複数の *ComponentSelector* を使用できます。ただし、すべてのケースにおいて、複数のセレクターを使用することは、複数のコマンドを指定してそれぞれに 1 つのコンポーネント・セレクターを指定することと同じです。例えば、`"errctrl -p -l hdisk0 -l hdisk1 errchecknormal"` のカスタマイズは、次の 2 つのカスタマイズと同等です。

```
errctrl -p -l hdisk0 errchecknormal
errctrl -p -l hdisk1 errchecknormal
```

-p または **-P** フラグで指定されたカスタマイズは、その使用後に削除されません。従って、1 つのカスタマイズを複数の新規コンポーネントに作用させることができます。永続カスタマイズは **-x** フラグで削除できます。*ComponentSelector* は、カスタマイズが作成されたときに指定されたとおりに指定する必要があります。例えば、カスタマイズが *ComponentSelector* `-l hdisk0` で作成された場合、そのカスタマイズは *ComponentSelector* `-l hdisk[0]` で削除することはできません。これは両方の *ComponentSelector* が同じコンポーネントの別名と一致する場合でも同様です。永続カスタマイズが削除される場合、そのカスタマイズがアクティブなときに作成されたコンポーネントの属性は変更されません。

-x と **-P** フラグで削除された永続カスタマイズは、**bosboot** コマンドを実行して AIX を再始動しない限り有効のままです。**-P** フラグで作成された永続カスタマイズは、**-x** と **-p** フラグを使用することで再始動後に削除されます。この場合は、AIX を再始動すると、そのカスタマイズが再度アクティブになります。

設定されたカスタマイズが不明な場合に、デフォルトのシステム設定を復元したい場合は、以下のいずれかを実行できます。

- **/var/adm/ras/raspertune** ファイルで、カスタマイズに関連する行を削除し、**bosboot** コマンドを実行して AIX を再始動します。
- **/var/adm/ras/raspertune** ファイルで、指定されている該当のフラグとパラメーターを見つけます。6 (427 ページ) の例のように、**-x** フラグを使用してそのカスタマイズを削除します。**bosboot** コマンドを実行し、AIX を再始動します。

永続カスタマイズを指定する際、**-r** フラグと **-u** フラグを使用できます。1 つのフラグを使用すると、指定されたコンポーネント・セレクターに別のネームスペースが指定されます。2 つのフラグを同時に使用すると、2 つの別々のコマンド呼び出しにそれぞれ 1 つのフラグが指定される場合と同じ結果になります。例えば、"errctrl -p -l hdisk0 -u -r errcheckdetail" の永続カスタマイズは、次の 2 つの別々のカスタマイズと同等です。

```
errctrl -p -l hdisk0 -u errcheckdetail
errctrl -p -l hdisk0 -r errcheckdetail
```

次の永続的なカスタマイズはすべて別個のもので、独立して変更または削除できます。

```
errctrl -p -l hdisk0 errcheckdetail
errctrl -p -l hdisk0 -r errcheckdetail
errctrl -p -l hdisk0 -u errcheckdetail
```

再帰的に下るカスタマイズ (**-r** フラグにより指定) は、他のすべてのカスタマイズよりも優先されます。この場合、再帰的に下るカスタマイズではない他のカスタマイズに対してそれらが指定される順序は関係ありません。

永続カスタマイズを照会するには、**-q** フラグを **-P** または **-p** フラグと一緒に使用します。**-q** フラグを **-P** フラグと一緒に指定すると、**/var/adm/ras/raspertune** ファイルからの行が表示されます。**-q** フラグを **-p** および **-r** フラグと一緒に指定すると、元々 **-r** フラグで指定された永続カスタマイズが表示されます。**-r** フラグを指定しないと、**-q** と **-p** フラグにより、**-u** フラグで指定された永続カスタマイズと -u フラグなしで指定された永続カスタマイズが表示されます。

永続カスタマイズでは、複数のサブコマンドを指定できます。競合するサブコマンドが使用される場合は、最後のサブコマンドが使用されます。例えば、**errchecknormal** サブコマンドと **errcheckdetail** サブコマンドが同じエラー検査属性に対して異なる値を指定している場合は、最後に指定されたサブコマンドが使用されます。

フラグ

項目	説明
-c *ComponentList*	コンポーネント名のコンマ区切りリストまたはスペース区切りリストを指定します。 **-c all** フラグを使用すると、すべてのコンポーネントが選択されます (このフラグだけが *ComponentSelector* である場合)。
-h または **-?**	使用方法のメッセージを表示します。
-l *aliasList*	コンポーネントの別名のコンマ区切りリストまたはスペース区切りリストを指定します。
-n	サブコマンドを即時に適用します。**-p** フラグまたは **-P** フラグのいずれも使用しない場合は、このフラグがデフォルトです。
-P	再始動後も持続するサブコマンドを指定します。これらのサブコマンドをアクティブにするには、**bosboot** コマンドを実行し、AIX を再始動する必要があります。

項目	説明
-x	指定したコンポーネントに対する永続的なカスタマイズを削除します。 *ComponentSelector* は、カスタマイズが元々指定されたときに入力されたとおりに入力する必要があります。
-p	永続サブコマンドを指定します。指定されるサブコマンドは、新規に作成されるコンポーネントに適用されます。
-q	選択されたコンポーネントの属性設定を照会します。このフラグを **-p** または **-P** フラグと一緒に使用すると、永続カスタマイズを表示できます。
-r	選択済みコンポーネントのすべてのサブコンポーネントに、サブコマンドを繰り返し適用します。
-t *type_subtypeList*	*type* 名または *type_subtype* 名のスペース区切りリストまたはコンマ区切りリストを指定します。有効な *type* 名は、`device`、`filesystem`、`network`、`services`、`storage`、および `ui` です。*type* 名と *type_subtype* 名の詳細なリストは、 **/usr/include/sys/ras_base.h** ヘッダー・ファイル内にあります。
-u	指定したコンポーネントの上位に、サブコマンドを繰り返し適用します。

注**: -u** フラグと **-r** フラグは同時に使用できます。コマンド・ラインで複数の **-c**、**-l**、および **-t** フラグを使用することができます。

終了状況

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. すべての JFS2 ユーザー・データ・コンポーネントに対して詳細なエラー検査をオンにするには、次のように入力します。

   ```
   errctrl -c 'jfs2.filesystem.*.userdata' errcheckdetail
   ```

2. 新規 JFS2 ファイルシステムのユーザー・データ・コンポーネントに対して永続カスタマイズを指定するには、次のように入力します。

   ```
   errctrl -p -c 'jfs2.filesystem.*.userdata' errcheckminimal
   ```

   現在のユーザー・データ・コンポーネントに対しては無効です。

3. 再始動後も持続するカスタマイズを指定するには、次のように入力します。

   ```
   errctrl -P -c 'jfs2.filesystem.*.userdata' errcheckminimal
   ```

   bosboot コマンドを実行して、AIX を再始動すると、最低限のエラー検査がすべての JFS2 ユーザー・データ・コンポーネントに対して有効になります。

4. すべての現行および将来的な JFS2 ユーザー・データ・コンポーネントに対して最低限のエラー検査を設定するには、次のように入力します。

   ```
   errctrl -npP -c 'jfs2.filesystem.*.userdata' errcheckminimal
   ```

5. イーサネット・コンポーネントに対して複数の永続属性値を指定するには、次のように入力します。

   ```
   errctrl -P -c ethernet errcheckminimal medsevdisposition=80
   ```

6. 例 2 で指定されたカスタマイズを削除するには、次のように入力します。

   ```
   errctrl -p -x -c 'jfs2.filesystem.*.userdata'
   ```

7. すべての永続的、再帰的な属性のカスタマイズをリストするには、次のように入力します。

   ```
   errctrl -q -p -r
   ```

8. JFS2 コンポーネントとその子孫に対して現行のエラー検査属性値をリストするには、次のように入力します。

   ```
   errctrl -q -c jfs2 -r
   ```

9. ネットワーク・メモリーのメモリー・オーバーレイ検出システム (MODS) を使用可能にするには (netmalloc コンポーネントに対してエラー・レベルを詳細レベルに上げる)、次のように入力します。

   ```
   errctrl errcheckdetail -c netmalloc
   ```

   または

   ```
   errctrl errchecklevel=7 -c netmalloc
   ```

   注**:** これにより、すべてのネットワーク・メモリー割り当てとフリー・イベントに対する **net_malloc_police** オプションも使用可能になります。

10. すべてのネットワーク・メモリー割り当てとフリー・イベントに対して **net_malloc_police** オプションを使用可能にするには、netmalloc コンポーネントのエラー・レベルを 5 以上に上げます。次のように入力します。

    ```
    errctrl errchecklevel=5 -c netmalloc
    ```

    このコマンドにより、ネットワーク・メモリーに対するアウトスタンディング・メモリー割り当て (OSTD) ロギングも使用可能になります。

11. netmalloc のポリス割り当てとフリー・イベントの頻度を 25% に変更するには、police_frequency 関数の可能性を 256 に変更します。次のように入力します。

    ```
    errctrl police_frequency=256 –c netmalloc.police
    ```

位置

/usr/sbin/errctrl

ファイル

項目	説明
/var/adm/ras/raspertune	再始動後に適用される永続属性のカスタマイズを含むファイル (**bosboot** コマンドが最初に実行される場合)。

関連資料:

289 ページの『dumpctrl コマンド』

関連情報:

ras_register および ras_unregister

ras_control コマンド

/var/adm/ras/raspertune コマンド

errdead コマンド

目的

システム・ダンプまたはライブ・ダンプからエラー・レコードを抽出します。

構文

/usr/lib/errdead [**-i** *FileName*] *DumpFile* [*UnixFile*]

説明

errdead コマンドは、**/dev/error** ファイルによって保守されている内部バッファーを含むシステム・ダンプまたはライブ・ダンプから、エラー・レコードを抽出します。**errdead** コマンドは、エラー・レコードからダンプ・ファイルを抽出し、エラー・ログにエラー・レコードを直接追加します。

errdead コマンドは、エラー・ログ・デーモンが起動されていないときでも実行できます。

フラグ

項目	説明
-i *FileName*	抽出したエラー・レコードを、*FileName* 変数で指定したエラー・ログ・ファイルに追加します。このファイルが存在しない場合、**errdead** コマンドはこのファイルを新しく作成します。このフラグを指定しないと、 エラー・ログ構成データベースからの値が使用されます。

パラメーター

項目	説明
DumpFile	操作を行うダンプ・イメージを指定します。
UnixFile	システム・ダンプまたはライブ・ダンプをとるときに使用する UNIX ファイルを指定します。これはダンプ先の同じシステムで **errdead** コマンドを使用する場合は必要ありません。

セキュリティー

アクセス制御: root ユーザーだけがこのコマンドを実行できます。

例

`/var/adm/ras/vmcore.0` ファイルにあるダンプ・イメージからエラー・ログ情報を収集するには、次のように入力します。

```
/usr/lib/errdead /var/adm/ras/vmcore.0
```

エラーが発生したときに **errdemon** デーモンが実行されていない場合、エラー・ログ情報はダンプ・イメージにあります。

ファイル

項目	説明
/etc/objrepos/SWservAt	エラー・ログ構成データベースであるソフトウェア保守援助プログラム属性オブジェクト・クラスが入っています。

関連資料:

420 ページの『errclear コマンド』

431 ページの『errinstall コマンド』

関連情報:

errlog コマンド

エラー・ログの概要

---

errdemon デーモン

目的

エラー・ログ・デーモン (**errdemon**) を始動して、エラー・ログにエントリーを書き込みます。

構文

errdemon [[**-B** *BufferSize*] [**-d** | **-D**] [**-i** *File*] [**-s** *LogSize*] [**-t** *Time*] [**-m** *MaxDups*] | **-l**]

説明

エラー・ログ・デーモンは、**/dev/error** フィールドからエラー・レコードを読み込んで、システム・エラー・ログ内にエラー・ログ・エントリーを作成します。このデーモンは、エラーが記録されるたびにシステム・エラー・ログにエントリーを書き込む一方、エラー・ログ・デーモンは、エラー通知データベース内で指定したとおりにエラー通知を実行します。 **/etc/objrepos/errnotify** ファイルは、エラー通知データベースです。デフォルトのシステム・エラー・ログは、**/var/adm/ras/errlog** ファイル内で保守されます。最後のエラー・エントリーは、不揮発性ランダム・アクセス・メモリー (NVRAM) に書き込まれます。システム始動中には、エラー・ログ・デーモンが始動したときに、この最後のエラー・エントリーは NVRAM から読み取られ、エラー・ログに追加されます。

エラー・ログ・デーモンは、エラー・レコード・テンプレートが、`Log=FALSE` と指定している場合は、記録されたエラーに関するエラー・ログ・エントリーを作成しません。

エラー・ログ・デーモンをフラグなしで使用すると、システムは、 エラー・ログ構成データベースに格納されている値を使用して、エラー・ログ・デーモンを再始動します。デフォルトでは、**errdemon** デーモンは、重複するエラー・ログ・エントリーの記録速度が非常に高速である場合、それらを除去します。これによって、暴走するエラー・ロギングがシステム・パフォーマンスに悪影響を与えるのを防ぐことができます。重複するエントリーの数は、詳細エラー・レポートで参照することができます。

PowerHA® pureScale® エラー・ロギングが使用可能になっている場合、エラー・ログ・エントリーは、ローカル・システム・エラー・ログに加えて PowerHA pureScale ログ・ストリームにも送信されます。PowerHA pureScale エラー・ロギング状況およびログ・ストリーム名は、エラー・ログ構成データベースの **errlg_pscale_enabled** および **errlg_pscale_logstream** 値で指定されます。 PowerHA pureScale クライアント・ファイルセットをシステムにインストールする必要があり、**CentralizedLogService** という名前のサービスのバインディング情報をセットアップする必要があります。PowerHA pureScale ログ・ストリームとして指定されたログ・スペースおよびログ・ストリーム・オブジェクトが存在している必要があります。

システム・エラー・ログからエントリーを除去するには、**errclear** コマンドを使います。

重要: 一般に、エラー・ログ・デーモンは、システムの初期化時に始動されます。エラー・ログ・デーモンを停止すると、内部バッファーに一時的に格納されているエラー・データが、エラー・ログ・ファイルに記録される前に上書きされることがあります。

フラグ

項目	説明
-B *BufferSize*	エラー・ログ・デバイスのメモリー内バッファーに、*BufferSize* パラメーターで指定したバイト数を使用します。指定したバッファー・サイズは、エラー・ログ構成データベースに保存されます。*BufferSize* パラメーターが現在使用中のバッファー・サイズよりも大きければ、メモリー内のバッファーのサイズが即座に増加されます。 *BufferSize* パラメーターが現在使用中のバッファー・サイズよりも小さければ、システムがリブートされた後で次回にエラー・ログ・デーモンが始動されると、新しいサイズが有効になります。バッファーのサイズとして、ハードコーディングされたデフォルト値の 8KB よりも小さい値を指定することはできません。 このパラメーターを指定しないと、エラー・ログ・デーモンはエラー・ログ構成データベースに入っているバッファー・サイズを使用します。 指定したサイズは、メモリー・ページ・サイズ (4KB) の倍数の次の整数に繰り上げられます。エラー・ログ・デバイス・ドライバーのメモリー内バッファーに使用するメモリーは、他のプロセスには使用できません (このバッファーは固定されています)。バッファーを大きくしすぎて、システムのパフォーマンスに影響を与えないように注意してください。また、バッファーを小さくしすぎると、エラー・エントリーが高速で着信した場合にバッファーから読み込んでログに入れられなくなると、バッファーがいっぱいになることがあります。バッファーがいっぱいになると、バッファー内のスペースが使用可能になるまで新しいエントリーが破棄されます。この状況が発生すると、エラー・ログ・デーモンはエラー・ログ・エントリーを作成して、問題をユーザーに通知します。この問題は、バッファーのサイズを大きくすれば解決できます。
-d	重複するエラー・ログ・エントリーを除去できないように指定します。 **-D** フラグによって指定されるデフォルトの動作では重複を除去します。
-D	重複するエントリーを除去するように指定します。これはデフォルトです。
-i *File*	*File* 変数で指定したエラー・ログ・ファイルを使用します。指定したファイル名は、エラー・ログ構成データベースに保存され、即座に使用されます。
-l	エラー・ログ構成データベースから、エラー・ログ・ファイル名、ファイル・サイズ、バッファー・サイズ、および重複処理の値が表示されます。
-m *MaxDups*	許可される重複エントリーの最大数を指定します。この数を超えると重複エラーが出ます。デフォルトは 1000 です。*MaxDups* で指定した回数だけエラーが重複すると、固有のエラーが記録される場合と同様に、重複エラーが書き込まれます。 *MaxDups* で許可される値は、1 から 2147483647 です。
-s *LogSize*	エラー・ログ・ファイルの最大サイズとして、*LogSize* 変数で指定したサイズを使用します。指定したログ・ファイルのサイズ制限は、エラー・ログ構成データベースに保存され、即座に使用されます。ログ・ファイルのサイズ制限が現在使用中のログ・ファイルのサイズよりも小さければ、エラー・ログ・デーモンはファイル名に **.old** を追加して、現行ログ・ファイルの名前を変更します。エラー・ログ・デーモンは、指定したサイズ制限を使用して新しいログ・ファイルを作成します。 **errpt** コマンドの **-i** フラグを使用して、旧ログ・ファイルからレポートを生成してください。 このパラメーターを指定しないと、エラー・ログ・デーモンは、エラー・ログ構成データベースからのログ・ファイル・サイズを使用します。
-t *Time*	直前のエラーと同じエラーが発生した場合に、それが重複と見なされる時間間隔 (ミリ秒) の概数を指定します。この時間間隔が経過した後に発生したエラーは、直前のエラーと同じであっても重複とは見なされません。デフォルトの間隔は 10000 または 10 秒です。 Time に使用できる値は 1 から 2147483647 です。 注: このフラグは、エラー・ロガーが高速で同じエラーを記録する場合は、重複するエントリーを除去します。この状態は、通常は、ループ状態に陥っていることを示しています。エラー通知オブジェクトがある可能性のある重複エラーを、すべてキャッチするようにはなっていません。この値を十分大きくしておくと、エラーが多すぎるときには除去することによって、エラー通知を少なくすることができます。エラー・レポートでの重複エラーの除去については、 **errpt** コマンドを参照してください。

セキュリティー

アクセス制御: このデーモンは、root ユーザーだけが実行できます。

例

1. エラー・ログ・デーモンを始動するには、次のように入力します。
   ```
   /usr/lib/errdemon
   ```
2. 現行の最大エラー・ログ・サイズを表示するには、次のように入力します。
   ```
   /usr/lib/errdemon -l
   ```
3. 現行の最大エラー・ログ・サイズを 1MB から 64KB に変更するには、次のように入力します。
   ```
   /usr/lib/errdemon -s 65536
   ```
4. ここ 10 ミリ秒以内に記録されるエラーだけを重複と見なすには、次のように入力します。
   ```
   /usr/lib/errdemon -t 10
   ```

ファイル

項目	説明
/dev/error	エラー・レコードのソース。
/var/adm/ras/errtmplt	エラー・テンプレート・リポジトリーが入っています。
/usr/lib/errdemon	**errdemon** デーモンが入っています。
/etc/objrepos/SWservAt	エラー・ログ構成データベースであるソフトウェア保守援助プログラム属性オブジェクト・クラスが入っています。

関連資料:

437 ページの『errpt コマンド』

関連情報:

エラー・ログ

errlog サブルーチン

エラー・ログの概要

---

errinstall コマンド

目的

エラー・ログ・メッセージ・セットにメッセージをインストールします。

構文

errinstall [**-c**] [**-f**] [**-q**] [**-z** *FileName*] *File*

説明

errinstall コマンドは、エラー・ログ・メッセージ・カタログ内のエラーの説明、推定原因、ユーザー原因、インストール原因、障害原因、推奨処置、詳細データ ID メッセージのセットの追加または置換に用いられるインストール用の補助機能です。

File パラメーターは、追加または置換されるメッセージが入っている入力ファイルを指定します。 *File* パラメーターを指定しないか、または **-** (負符号) を指定すると、 **errinstall** コマンドは標準入力から読み込みます。

注**:** ライセンス・プログラムおよび企業内アプリケーションでは、エラー・ログ・メッセージ・セットに入っている事前定義メッセージを使用するようにしてください。事前定義メッセージは、**errmsg -w** コマンドを使って表示します。サード・パーティーのソフトウェア・ベンダーが新しいメッセージを追加する場合は、 IBM デベロッパー・ソリューション (Developer Solutions) に連絡して、新規メッセージを登録する必要があります。企業内アプリケーションについては、企業内アプリケーション作成中に **errmsg** コマンドを使って新しいメッセージを追加できます。ただし、他の企業内アプリケーションに追加されたメッセージと同じものがないよう注意が必要です。

取り消し機能

errinstall コマンドは、現行ディレクトリー内に *File*.**undo** という元に戻すファイルを作成します (**errinstall** コマンドが標準入力から読み込んでいる場合、undo ファイルの情報は標準出力に書き出されます)。 *File*.**undo** ファイルを **errinstall** コマンドへの入力として使用し、**errinstall** コマンドが行ったばかりの変更を取り消すことができます。変更を取り消すには、**-f** フラグを指定して **errinstall** コマンドを実行し、*File* パラメーターには *File*.**undo** ファイルを指定します。

入力ファイル **(**または標準入力**)** ファイル・フォーマット

エラー・ログ・メッセージ・カタログ内の単一メッセージに追加または置換を行うには、2 つの別々の行に情報を入れる必要があります。 1 つのファイルに、複数の追加や置換を入れることができます。最初の行には、メッセージが追加または置換されるメッセージ・セットを指定します。入力は次のフォーマットを使用します。

```
SET MessageSetID
```

この場合、*MessageSetID* パラメーターは、次のいずれかの 1 文字です。

項目	説明
E	エラーの説明を指定します。
P	推定原因を指定します。
U	ユーザー原因を指定します。
I	インストール原因を指定します。
F	障害原因を指定します。
R	推奨処置を指定します。
D	詳細データを指定します。

2 行目は、追加または置換を行うエラー・メッセージのエラー・メッセージ ID をリストします。この行は、メッセージ・セットを指定する単一行の後に少なくとも 1 つは必要であり、複数あってもかまいません。上記のとおり、メッセージ ID を入手するには、最寄りのサービス技術員に連絡してください。ただし、社内アプリケーションのみのためにメッセージが必要な場合を除きます (この場合は、事前定義エラー・メッセージ ID を使わず、**errmsg** コマンドを使ってエラー・メッセージをインストール)。

エラー・メッセージ ID とエラー・メッセージの間には、スペースを 1 つ入れ、次のように、エラー・メッセージのテキストを二重引用符で囲まなければなりません。

```
message ID "message text"
```

2 つの必須情報行の他に、コメント行も追加できます。コメント行には、最初の桁にドル記号 ($) またはアスタリスク (*) 演算子が含まれていなければなりません。アスタリスクのほうが適しています。

注**:** エラーの説明、推定原因、詳細データ ID メッセージ・セットに追加されるメッセージは、40 字以内の長さに納める必要があります。ユーザー原因、インストール原因、障害原因、および推奨処置メッセージ・セットに追加されるメッセージは、 128 字以内の長さに納めなければなりません。メッセージがこれらの長さを超えると、**errinstall** コマンドは警告メッセージを表示します。メッセージはコード・ポイント・カタログに追加されます。これらのメッセージは、要約 errpt コマンドによって表示されるときに切り捨てられます。

フラグ

項目	説明
-c	入力 *File* パラメーターの構文エラーを検査します。
-f	ID が重複しているメッセージを置き換えます。既に使用されているメッセージ ID を使用してメッセージを追加しようとすると、 **errinstall** コマンドは **-f** フラグによって、古いメッセージ・テキストを新しいメッセージ・テキストに置き換えます。 **-f** フラグを指定しないと、古いメッセージ・テキストは置き換えられず、警告メッセージが標準エラーに書き出されます。また、メッセージのインストールをやり直す場合にも、**-f** フラグが必須です。
-q	**undo** ファイルの作成を抑止します。
-z *FileName*	*FileName* パラメーターで指定したエラー・ログ・メッセージ・カタログを使用します。

セキュリティー

アクセス制御: root ユーザーだけがこのコマンドを実行できます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. ライセンス・プロダクト、 lpp に対するエラー・ログ・メッセージをインストールするには、次のように入力します。

   ```
   errinstall -f /tmp/lpp.desc
   ```

2. **errinstall** コマンドの上記の例で、エラー・ログ・メッセージ・カタログに行った変更を取り消すには、次のように入力します。

   ```
   errinstall -f /tmp/lpp.desc.undo
   ```

3. 推定原因メッセージ・セットにエラー・メッセージをインストールするには、次のように入力します。

   ```
   errinstall

   * Add a probable cause for widget failure:
   SET P
   E100 "widget adapter"
   ```

4. メッセージを推定原因メッセージ・セット内の重複する ID と置き換えるには、次のように入力します。

   ```
   errinstall -f

   * Replace the message associated with ID E100 in the
   * Recommended Action message set
   SET R
   E100 "Replace disk drive"
   ```

5. 入力ファイルに **in_file** と名前を付け、それを使って新しいエラー・メッセージをインストールしたい場合は、次のように入力します。

```
errinstall in_file
```

6. メッセージ・セット内の既存のエラー・メッセージを上書きするには、次のように、既に定義されている **in_file** 内の ID 番号を使い、 **errinstall** コマンドで **-f** フラグを指定します。

```
errinstall -f in_file
```

7. 次の例は、インストールされる入力ファイルの内容例を示しています。

```
*
* Add these error messages to the Detailed Data message set:
*
SET D
8105 "Logical channel number"
8106 "Timer reference stamp"
*
* Add these error messages to the Probable Cause message set:
*
SET P
E861 "Bad memory card"
E865 "Unexpected System Halt"
E876 "Fiber Optic Cable"
*
* Add this message to the Recommended Action message set:
*
SET R
E850 "Install updated driver code"
```

ファイル

項目	説明
/usr/lib/nls/msg/$LANG/codepoint.cat	エラー・ログ・メッセージ・カタログが入っています。米国では、**$LANG** 環境変数の値は **En_US** です。

関連資料:

429 ページの『errdemon デーモン』

関連情報:

errsave コマンド

errlog コマンド

エラー・ログ

---

errlogger コマンド

目的

オペレーター・メッセージのログを作成します。

構文

errlogger *Message*

説明

errlogger コマンドは、長さが 1024 文字までのオペレーター・メッセージを含む、オペレーター・エラー・ログ・エントリーを作成します。

セキュリティー

アクセス制御: root ユーザーだけがこのコマンドを実行できます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

システム・ドライブ再構成のためのオペレーター・メッセージを作成するには、次のように入力します。

```
errlogger system drive reconfigured
```

関連資料:

437 ページの『errpt コマンド』

関連情報:

errsave コマンド

errlog コマンド

エラー・ログの概要

---

errmsg コマンド

目的

エラー・ログ・メッセージ・カタログにメッセージを追加します。

構文

errmsg [**-c**] [**-z** *FileName*] [**-w** *Set_List* | *File*]

説明

errmsg コマンドは、エラーの説明、推定原因、ユーザー原因、インストール原因、障害原因、推奨処置、詳細データ ID のメッセージ・セットなどを含むエラー・ログ・メッセージ・カタログの更新と表示に使用されます。

メッセージが追加または削除されるメッセージ・セットは、入力 *File* パラメーター内に次のように指定されます。

項目	説明
* または **$**	コメント行には、最初の桁に * (アスタリスク) または $ (ドル記号) コメント演算子を付けなければなりません。* のほうが適しています。
+	追加されるメッセージの前には、+ (正符号) を付けなければなりません。
-	削除されるメッセージの前には、- (負符号) を付けなければなりません。
SET	メッセージ・セット ID。
"Message Text"	メッセージ・テキストは、二重引用符で囲まれなければなりません。
Message ID	削除されるメッセージのメッセージ ID。

エラーの説明、推定原因、詳細データ ID のメッセージ・セットに追加されるメッセージは、40 字以内の長さに納めなければなりません。ユーザー原因、インストール原因、障害原因、および推奨処置メッセー

ジ・セットに追加されるメッセージは、 128 字以内の長さに納めなければなりません。最大で 2047 のユーザー定義メッセージが、各メッセージ・セットに追加できます。

errmsg コマンドは、エラー・レコード・テンプレート・リポジトリー内で使用される新しいメッセージを作成するために、アプリケーション開発者が使用します。既存メッセージは、可能な限り使用してください。

コマンド・ラインで何のフラグも指定しないと、デフォルトの操作は更新です。更新は、入力 *File* パラメーターで指定します。入力 *File* パラメーターを指定しないか、または *File* パラメーターの代わりに **-** (負記号) を指定すると、**errmsg** コマンドは標準入力から読み込みます。追加するメッセージごとに、**errmsg** コマンドは ID を割り当てます。メッセージをメッセージ・カタログに追加する他に、**errmsg** コマンドは ID とメッセージ・テキストを *File***.out** ファイルに書き込みます。また、メッセージ・カタログを削除するときにも、*File***.out** ファイルが作成されます。 **errmsg** コマンドが標準入力から読み込む場合、ID とメッセージ・テキストは標準出力に書き出されます。

フラグ

項目	説明
-c	構文エラーがないか入力ファイルを調べます。
-w *Set_List*	*Set_List* 変数で指定したエラー・ログ・メッセージ・セットを表示します。このオプションは、エラー・ログ・メッセージ・セットに含まれたメッセージと ID を表示します。出力は標準出力に書き出されます。*Set_List* 変数は、コンマで区切るか、または二重引用符で囲んでコンマまたはブランクで区切ることができます。 *Set_List* 変数はメッセージ・セット ID です。*Set_List* 変数の値に **all** を指定すると、すべてのエラー・ログ・メッセージ・セットの内容が表示されます。 *Set_List* 変数の有効な値は次のとおりです。 **all** すべてのメッセージ・セットが表示されます。 **D** 詳細データ ID メッセージ・セットが表示されます。 **E** エラーの説明メッセージ・セットが表示されます。 **F** 障害原因メッセージ・セットが表示されます。 **I** インストール原因メッセージ・セットが表示されます。 **P** 推定原因メッセージ・セットが表示されます。 **R** 推奨処置メッセージ・セットが表示されます。 **U** ユーザー原因メッセージ・セットが表示されます。
-z *Filename*	*Filename* 変数で指定したエラー・ログ・メッセージ・カタログを使用します。

セキュリティー

アクセス制御: root ユーザーだけがこのコマンドを実行できます。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 推定原因メッセージ・セットからメッセージを削除するには、次のように入力します。

```
errmsg
* Delete messages FF1A, FF1B, and FF1C from the Probable Cause
* message set
```

```
SET P
- FF1A
- FF1B
- FF1C
```

2. ウィジェット障害エラーに関する推定原因メッセージ・セットに、メッセージを追加するには、次のように入力します。

```
errmsg
* Add a Probable Cause for Widget Failure
SET P
+ "WIDGET ADAPTER"
```

ファイル

項目	説明
/usr/lib/nls/msg/$LANG/codepoint.cat	エラー・ログ・メッセージ・カタログが入っています。米国では、**$LANG** の値は **En_US** です。

関連資料:

434 ページの『errlogger コマンド』

関連情報:

errsave サブルーチン

errlog サブルーチン

エラー・ログ・スペシャル・ファイル

---

errpt コマンド

目的

ログに記録されたエラーのレポートを生成します。

構文

エラー・ログからレポートを処理する

errpt [**-@** *wpar_name*] [**-a**] [**-A**] [**-c**] [**-d** *ErrorClassList*] [**-D**] [**-e** *EndDate*] [**-g**] [**-i** *File*] [**-I** *File*] [**-j** *ErrorID* [**,***ErrorID*]] | [**-k** *ErrorID* [**,***ErrorID*]] [**-J** *ErrorLabel* [**,***ErrorLabel*]] | [**-K** *ErrorLabel* [**,***ErrorLabel*]] [**-l** *SequenceNumber*] [**-m** *Machine*] [**-n** *Node*] [**-s** *StartDate*] [**-F** *FlagList*] [**-N** *ResourceNameList*] [**-P**] [**-R** *ResourceTypeList*] [**-S** *ResourceClassList*] [**-T** *ErrorTypeList*] [**-y** *File*] [**-z** *File*]

エラー・レコード・テンプレート・リポジトリーからのレポートを処理する

errpt [**-a**] [**-A**] [**-I** *File*] [**-t**] [**-d** *ErrorClassList*] [**-j** *ErrorID* [**,***ErrorID*]] | [**-k** *ErrorID* [**,***ErrorID*]] [**-J** *ErrorLabel* [**,***ErrorLabel*]] | [**-K** *ErrorLabel* [**,***ErrorLabel*]] [**-F** *FlagList*] [**-P**] [**-T** *ErrorTypeList*] [**-y** *File*] [**-z** *File*]

説明

errpt コマンドは、エラー・ログ内のエントリーからエラー・レポートを作成します。 errpt コマンドには、特定の基準に一致するエラーを選択するためのフラグが含まれています。デフォルト条件を使用すると、エラー・ログ・エントリーを、発生時に記録したのとは逆の順序で表示することができます。 **-c** (concurrent) フラグを使用すると、エラーを発生した時点の状態で表示することができます。 **errpt** コマ

ンドで **-i** フラグを使用しなければ、**errpt** によって処理されるエラー・ログ・ファイルは、エラー・ログ構成データベースで指定したファイルになります。 (エラー・ログ構成データベース内の情報を表示するには、 **errdemon** コマンドを使用します。)

デフォルトの要約レポートには、エラーごとに 1 行ずつデータが入っています。フラグを使用すると、別のフォーマットでレポートを生成できます。

> 注**: errpt** コマンドは、エラー・ログ分析を実行しません。分析するには、 **diag** コマンドを使用します。ただし、エラー・ログ分析を実行すると、診断によっては、診断情報が再びエラー・ログに追加される場合があります。このような情報は、それに対応するエラー・ログ・エントリーの明細データに続けて表示されます。

System Management Interface Tool (SMIT) の **smit errpt** 高速パスを使用して、このコマンドを実行できます。

フラグ

項目	説明
-@ *wpar_name*	指定された WPAR 名についてエラー・エントリーを選択します。
-a	エラー・ログ・ファイル内のエラー情報を詳細なフォーマットで表示します。 **- t** フラグと一緒に使用すると、テンプレート・ファイルのすべての情報が表示されます。
-A	**-a** フラグが作成する明細レポートの簡易バージョンを表示します。 **-A** フラグは、**-a**、**-g**、または **-t** フラグと同時には使用できません。報告される項目は、ラベル、日時、型、リソース名、説明、および明細データです。このフラグの出力例は次のような形式になります。 `LABEL: STOK_RCVRY_EXIT` `Date/Time: Tue Dec 14 15:25:33` `Type: TEMP` `Resource Name: tok0` `Description` `PROBLEM RESOLVED` `Detail Data` `FILE NAME` `line: 273 file: stok_wdt.c` `SENSE DATA` `0000 0000 0000 0000 0000 0000` `DEVICE ADDRESS` `0004 AC62 25F1`
-c	各エラー・エントリーを同時に (つまりログの時点で) フォーマットして表示します。ログ・ファイル内の既存のエントリーは、記録された順に表示されます。
-d *ErrorClassList*	エラー・レポートを、有効な *ErrorClassList* 変数で指定したタイプのエラー・レコードに制限します。この変数は、**H** (ハードウェア)、**S** (ソフトウェア)、 **0** (**errlogger** コマンド・メッセージ)、および **U** (不定) です。 *ErrorClassList* 変数のエラー・レコードは、 **,** (コンマ) で区切るか、あるいは、 **" "** (二重引用符) で囲んで **,** (コンマ) またはスペース文字で区切ることができます。
-D	重複するエラーを統合します。詳細エラー・レポートは、 **-a** フラグで得られ、重複の数および、最初と最後の時刻を報告します。「プログラミングの一般概念*:* プログラムの作成およびデバッグ」の『エラー・ログの概要』を参照してください。 **注: -D** フラグは、**-c**、**-g**、**-l**、**-t**、および **-P** フラグと同時には使用できません。
-e *EndDate*	*EndDate* 変数と同時か、これよりも前に追加されたすべてのレコードを指定します。この場合、 *EndDate* 変数のフォーマットは、*mmddhhmmyy* (月、日、時、分、年) です。

項目	説明
-g	フォーマットされていないエラー・ログ・エントリーの ASCII 表現を表示します。このフラグの出力は次のフォーマットになります。 **el_sequence** エラー・ログ・スタンプ番号 **el_label** エラー・ラベル **el_timestamp** エラー・ログ・エントリーのタイム・スタンプ **el_crcid** 固有の巡回冗長検査 (CRC) エラー ID **el_machineid** マシン ID 変数 **el_nodeid** ノード ID 変数 **el_class** エラー・クラス **el_type** エラー・タイプ **el_resource** リソース名 **el_rclass** リソース・クラス **el_rtype** リソース・タイプ **el_vpd_ibm** IBM 重要プロダクト・データ (VPD) **el_vpd_user** ユーザー VPD **el_in** デバイスのロケーション・コード **el_connwhere** ハードウェア接続 ID (スロット番号など、特定のデバイス上の位置) **et_label** エラー・ラベル **et_class** エラー・クラス **et_type** エラー・タイプ **et_desc** エラーの説明 **et_probcauses** 推定原因 **et_usercauses** ユーザー原因 **et_useraction** ユーザー処置

項目	説明
	et_instcauses インストール原因
	et_instaction インストール処置
	et_failcauses 障害原因
	et_failaction 障害処置
	et_detail_length 詳細データ・フィールド長
	et_detail_descid 詳細データ ID
	et_detail_encode 詳細データ入力フォーマットの記述
	et_logflg ログ・フラグ
	et_alertflg 警告可能なエラー・フラグ
	et_reportflg エラー・レポート・フラグ
	el_detail_length 詳細データ入力長
	el_detail_data 詳細データ入力
-F *FlagList*	テンプレートの `Alert`、`Log`、または `Report` のフィールドの値に従って、エラー・レコード・テンプレートを選択します。 *FlagList* 変数は、 , (コンマ) で区切るか、あるいは、 " " (二重引用符) で囲んで , (コンマ) またはスペース文字で区切ることができます。**-F** フラグは、**-t** フラグと一緒に使用する必要があります。 *FlagList* 変数の有効な数値は次のとおりです。 **alert=0** `Alert` フィールドが「False」に設定されたエラー・レコード・テンプレートを選択します。 **alert=1** `Alert` フィールドが「True」に設定されたエラー・レコード・テンプレートを選択します。 **log=0** `Log` フィールドが「False」に設定されたエラー・レコード・テンプレートを選択します。 **log=1** `Log` フィールドが「True」に設定されたエラー・レコード・テンプレートを選択します。 **report=0** `Report` フィールドが「False」に設定されたエラー・レコード・テンプレートを選択します。 **report=1** `Report` フィールドが「True」に設定されたエラー・レコード・テンプレートを選択します。
-i *File*	*File* 変数で指定したエラー・ログ・ファイルを使用します。このフラグを指定しないと、エラー・ログ構成データベースからの値が使用されます。
-I *File*	*File* で指定した診断ログ・ファイルを使用します。このフラグを指定しない場合は、デフォルトのパス名、 **/var/adm/ras/diag_log** が使用されます。
-j *ErrorID*[**,***ErrorID*]	*ErrorID* (エラー ID) 変数で指定したエラー・ログ・エントリーだけを指定します。 *ErrorID* 変数は、 , (コンマ) で区切るか、あるいは、 " " (二重引用符) で囲んで , (コンマ) またはスペース文字で区切ることができます。**-t** フラグと組み合わせると、エントリーはエラー・テンプレート・リポジトリーから処理されます。 (その他の場合、エントリーはエラー・ログ・リポジトリーから処理されます。)

項目	説明
-J *ErrorLabel*	*ErrorLabel* で指定したエラー・ログ・エントリーを指定します。 *ErrorLabel* 変数は、コンマで区切るか、または二重引用符で囲んでコンマまたはブランクで区切ることができます。 **-t** フラグと組み合わせると、エントリーはエラー・テンプレート・リポジトリーから処理されます。 (それ以外の場合、エントリーはエラー・ログ・リポジトリーから処理されます。)
-k *ErrorID*[**,***ErrorID*]	*ErrorID* (エラー ID) 変数で指定したエラー・ログ・エントリーを除外します。 *ErrorID* 変数は、 **,** (コンマ) で区切るか、あるいは、 " " (二重引用符) で囲んで **,** (コンマ) またはスペース文字で区切ることができます。**-t** フラグと組み合わせると、エントリーはエラー・テンプレート・リポジトリーから処理されます。 (その他の場合、エントリーはエラー・ログ・リポジトリーから処理されます。)
-K *ErrorLabel*	*ErrorLabel* 変数で指定したエラー・ログ・エントリーを除外します。 *ErrorLabel* 変数は、コンマで区切るか、または二重引用符で囲んでコンマまたはブランクで区切ることができます。 **-t** フラグと組み合わせると、エントリーはエラー・テンプレート・リポジトリーから処理されます。 (それ以外の場合、エントリーはエラー・ログ・リポジトリーから処理されます。)
-l *SequenceNumber*	*SequenceNumber* 変数で指定した、独自のエラー・ログ・エントリーを選択します。このフラグは、エラー通知オブジェクト・クラス内の方式によって使用されます。 *SequenceNumber* 変数は、 **,** (コンマ) で区切るか、あるいは、 " " (二重引用符) で囲んで **,** (コンマ) またはスペース文字で区切ることができます。
-m *Machine*	指定した *Machine* 変数に関するエラー・ログ・エントリーを指定します。 **uname -m** コマンドは、*Machine* 変数の値を戻します。
-n *Node*	指定した *Node* 変数に関するエラー・ログ・エントリーを指定します。 **uname -n** コマンドは、*Node* 変数の値を戻します。
-N *ResourceNameList*	*ResourceNameList* 変数で指定したリソース名に関するレポートを作成します。 *ResourceNameList* 変数は、エラーを検出したリソース名のリストです。ソフトウェア・エラーの場合、*ResourceNameList* 変数は、エラーを検出したリソース名を表示します。ハードウェア・エラーの場合は、デバイス名またはシステム・コンポーネントの名前を表示します。これは、コンポーネントに欠陥があるとか、置換が必要であることを示しているのではありません。むしろ、エラー分析のために適切な診断モジュールを決定するのに使用されます。 *ResourceNameList* 変数の名前は、 **,** (コンマ) で区切るか、あるいは、 " " (二重引用符) で囲んで **,** (コンマ) またはスペース文字で区切ることができます。
-P	直前のエラーと重複するエラーだけを表示します。**-P** フラグは、エラー・ログ・デバイス・ドライバーが生成する重複エラーにのみ適用されます。これらのエラーは、**errdemon** デーモンの、 **-t** フラグが制御する、 **errlg_duptime** エラー・ロギング属性により指定された、おおよその時間間隔内に発生した重複エラーです。 **-P** フラグは、**-D** フラグと同時には使用できません。
-R *ResourceTypeList*	*ResourceTypeList* 変数で指定したリソース・タイプに関するレポートを作成します。ハードウェア・エラーの場合、*ResourceTypeList* 変数はデバイス・タイプです。ソフトウェア・エラーの場合、これは **LPP** 値です。 *ResourceTypeList* 変数の項目は、それぞれ、 **,** (コンマ) で区切るか、あるいは、 " " (二重引用符) で囲んで **,** (コンマ) またはスペース文字で区切ることができます。
-s *StartDate*	*StartDate* 変数と同時か、それよりも後に追加されたすべてのレコードを指定します。この場合、*StartDate* 変数の形式は、 *mmddhhmmyy* (月、日、時、年) です。
-S *ResourceClassList*	*ResourceClassList* 変数で指定したリソース・クラスに関するレポートを作成します。ハードウェア・エラーの場合、*ResourceClassList* 変数はデバイス・クラスです。リソース・クラスは、それぞれ、 **,** (コンマ) で区切るか、あるいは、 "" (二重引用符) で囲んで **,** (コンマ) またはスペース文字で区切る必要があります。
-t	エラー・ログの代わりにエラー・レコード・テンプレート・リポジトリーを処理します。 **-t** フラグを使用すれば、レポート形式でエラー・レコード・テンプレートを表示できます。
-T *ErrorTypeList*	エラー・レポートを、有効な *ErrorTypeList* 変数で指定したエラー・タイプ **INFO**、**PEND**、**PERF**、**PERM**、**TEMP**、および **UNKN** に制限します。エラー・タイプは、それぞれ、**,** (コンマ) で区切るか、あるいは、 "" (二重引用符) で囲んで **,** またはスペース文字で区切ることができます。
-y *File*	*File* 変数で指定したエラー・レコード・テンプレート・ファイルを使用します。 **-t** フラグと組み合わせると、エントリーは指定したエラー・テンプレート・リポジトリーから処理されます。 (それ以外の場合、指定したエラー・テンプレート・リポジトリーが使用され、エントリーはエラー・ログ・リポジトリーから処理されます。)
-z *File*	*File* 変数で指定したエラー・ログ・メッセージ・カタログを使用します。 **-t** フラグと組み合わせると、エントリーはエラー・テンプレート・リポジトリーから処理されます。 (それ以外の場合、エントリーはエラー・ログ・リポジトリーから処理されます。)

例

1. 完全な要約レポートを表示するには、次のように入力します。

   ```
   errpt
   ```

2. 完全な詳細レポートを表示するには、次のように入力します。

   ```
   errpt  -a
   ```

3. エラー ID E19E094F に関してログに記録されたすべてのエラーの詳細レポートを表示するには、次のように入力します。

   ```
   errpt  -a  -j E19E094F
   ```

4. 過去 24 時間以内にログに記録されたすべてのエラーの詳細なレポートを表示するには、次のように入力します。

   ```
   errpt  -a  -s mmddhhmmyy
   ```

   この場合、mmddhhmmyy は、現在の月、日、時、分、年、マイナス 24 時間に相当します。

5. エラー・ログ・エントリーに対してログ記録がオフにされているエラー・レコード・テンプレートを表示するには、次のように入力します。

   ```
   errpt  -t  -F log=0
   ```

6. 代替エラー・ログ・ファイル /var/adm/ras/errlog.alternate からのすべてのエントリーを表示するには、次のように入力します。

   ```
   errpt  -i /var/adm/ras/errlog.alternate
   ```

7. 代替エラー・ログ・ファイル /var/adm/ras/errlog.alternate からのすべてのハードウェア・エントリーを表示するには、次のように入力します。

   ```
   errpt  -i /var/adm/ras/errlog.alternate -d H
   ```

8. エラー・ラベル ERRLOG_ON に関してログに記録されたすべてのエラーの詳細レポートを表示するには、次のように入力します。

   ```
   errpt  -a  -J ERRLOG_ON
   ```

9. すべてのエラーおよびグループ重複エラーの詳細レポートを表示するには、次のように入力します。

   ```
   errpt -aD
   ```

10. 8 月中のエラー・ラベル DISK_ERR1 および DISK_ERR2 に関してログに記録されたすべてのエラーの詳細レポートを表示するには、次のように入力します。

    ```
    errpt -a -J DISK_ERR1,DISK_ERR2 -s 0801000004 -e 0831235904"
    ```

ファイル

項目	説明
/etc/objrepos/SWservAt	エラー・ログ構成データベースであるソフトウェア保守援助プログラム属性オブジェクト・クラスが入っています。

関連資料:

144 ページの『diag コマンド』

関連情報:

uname コマンド

errlog サブルーチン

詳細エラー・レポートの例

---

errstop コマンド

目的

エラー・ログ・デーモンを終了します。

構文

errstop

説明

重要**:** **errstop** コマンドを実行すると、診断および回復機能が使用できなくなります。通常、**errdemon** コマンドは、システムの初期化時に自動的に始動され、システムの終了時に停止されます。エラー・ログは、通常の処理中に停止しないでください。 **errstop** コマンドは、やむをえない場合に限り、結果を十分に理解したうえで使用してください。

errstop コマンドは、**errdemon** コマンドで始動されたエラー・ログ・デーモンを停止します。

セキュリティー

アクセス制御: このコマンドは、root ユーザーだけが実行できます。

例

errdemon デーモンを終了するには、次のように入力します。

```
/usr/lib/errstop
```

関連資料:

428 ページの『errdead コマンド』

関連情報:

errsave コマンド

errlog コマンド

エラー・ログの概要

errupdate コマンド

目的

エラー・レコード・テンプレート・リポジトリーを更新します。

構文

errupdate [**-c**] [**-f**] [**-h**] [**-n**] [**-p**] [**-q**] [**-y** *FileName*] [*File*]

説明

errupdate コマンドは、エラー・レコード・テンプレート・リポジトリーのエントリーを追加または削除したり、既存のエントリーのログやレポート、警告特性を変更します。 **errupdate** コマンドは、 *File* パラメーターに指定したファイルから読み取ります。 *File* パラメーターを指定しないと、**errupdate** コマンドは標準入力から読み取り、標準出力に書き出します。

追加、削除、変更する各エントリーの前には、演算子を付けなければなりません。有効な演算子は次のとおりです。

項目	説明
+	エントリーを追加します (追加演算子)。
-	エントリーを削除します (削除演算子)。
=	エントリーのログ、レポート、警告特性を変更します。

入力ファイルのエントリーは、ブランク行で区切らなければなりません。

入力ファイル内のコメントは、テンプレート間に入れ、1 桁目に * (アスタリスク) を付けて示すことができます。

エラー・テンプレートで X/Open Portability Guide Issue 4 メッセージを使用する場合は、メッセージ・カタログを指定する必要があります。これは、次のようなフォーマットの 1 行を用いて行うことができます。

```
<*!catalog-name>
```

次に例を示します。

```
*!mycat.cat
```

指定されたカタログは、別の "*!" カタログ指定子が検出されるまで、次のテンプレートで検出される XPG4 メッセージに適用されます。また、"*!" 指定子を、「catname」キーワードを指定した個々のテンプレート単位で指定変更することもできます。

カタログに対して絶対パス名が指定されない限り、メッセージ・カタログを検索する通常の規則が用いられます。例えば、上記の例で、mycat.cat は **/usr/lib/nls/msg/%L** 内にあると見なされます。

追加するエントリーは、特定のフォーマットで定義する必要があります。エラー・レコード・テンプレートの一般的なフォーマットは次のとおりです。

```
Error Record Template

        + LABEL:
                        Comment=
                        Class=
                        Log=
                        Report=
```

```
Alert=
Err_Type=
Err_Desc=
Prob_Causes=
User_Causes=
User_Actions=
Inst_Causes=
Inst_Actions=
Fail_Causes=
Fail_Actions=
Detail_Data= <data_len>, <data_id>,
<data_encoding>
```

さらに、XPG4 メッセージのカタログ名は、以下のように入力して指定することができます。

```
catname = <catalog>
```

XPG4 メッセージ、catname キーワード、9 個以上の詳細なデータ項目を含むテンプレートはすべて、XPG4 テンプレートと呼ばれます。 XPG4 テンプレートには警報を出すことはできず、エラー ID の場合若干異なる計算が使用されます。

エラー・レコード・テンプレート・フィールドの説明は、次のとおりです。

項目	説明
Alert	エラー・ログ・エントリーを SNA Generic Alert Architecture に準拠するプロセスで処理できることを示します。 Alert フィールドは「True」または「False」に設定することができます。このフィールドをテンプレートから省略すると、値はデフォルトの「False」になります。 Alert フィールドを「True」に設定すると、 Err_Desc、Inst_Actions、 Fail_Cause、および Detail_Data data_id フィールドの内容が、「SNA Generic Alert Architecture」(資料番号 GA27-3136) によって認識される値でない限り、**errupdate** コマンドはテンプレートを追加しません。使用する値のどれも SNA Generic Alert Architecture によって認識されないか、テンプレートが XPG4 テンプレートであり、かつ Alert フィールドが「True」に設定されている場合は、**-p** フラグを指定して、テンプレートを追加もしくは更新する必要があります。
Class	ハードウェアまたはソフトウェア内で発生したエラーが、オペレーター・メッセージであるか、または不定であるかを示します。次のいずれかのクラス・ディスクリプターを指定する必要があります。 **H** エラーがハードウェア障害であることを示します。 **O** エラーがオペレーター・メッセージであることを示します。 **S** エラーがソフトウェア障害であることを示します。 **U** エラーが不定であることを示します。
Comment	エラー ID メッセージ・セット用に作成された **#define** ステートメントにコメントを含めるよう指定します。コメントは 40 字以内で、二重引用符で囲まれていなければなりません。 40 字を超える長さのコメントは、自動的に切り捨てられます。 **errupdate** コマンドは、コメントを C 言語のコメント区切り文字の /* (スラッシュ、アスタリスク) と */ (アスタリスク、スラッシュ) で囲みます。

項目	説明
Detail_Data	エラーが発生するときにエラーと一緒にログに記録される検出モジュール名、センス・データ、戻りコードなどの詳細なデータを示します。エラーと一緒に詳細なデータが何も記録されない場合、このフィールドはブランクのままにするか、**data_len** の値として 0 を指定することによって、詳細データ ID メッセージ・セットからのメッセージを表示できます。各 **Detail_Data** フィールドには、次の 3 つの値をコンマで区切って指定する必要があります。 **data_len** **data_id** 値に関連するデータのバイト数。 **data_len** の値は、10 進数として解釈されます。環境に依存するサイズを指定するには、「W」を使用します。「W」は、エラーが 64 ビット環境で記録される場合は 8 バイトとして処理され、それ以外の場合は 4 バイトとして処理されます。 **注:** 詳細データの長さを計算している間は、「W」はそれぞれ 8 バイト長として処理され、大文字小文字は区別されません。 **data_id** 詳細データの前のエラー・レポートに出力される、詳細データ ID メッセージ・セット「D」からのテキスト・メッセージを表します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。 **data_encoding** 詳細データがエラー・レポートにどのように出力されるかを示します。有効な値は次のとおりです。 **ALPHA** 詳細データは、印刷可能な ASCII 文字列です。 **DEC** 詳細データは、整数値のバイナリー表現と、10 進法に相当する値が出力されます。 **LDEC** 詳細データは、64 ビット値のバイナリー表現と、10 進法に相当する値が出力されます。 **HEX** 詳細データは、16 進法で出力されます。 テンプレートごとに、16 個までの Detail_Data エントリーを指定することができます。1 つのエラーに対してログに記録されるデータ量は、 **/usr/include/sys/err_rec.h** ファイルに定義されたエラー・レコードの最大の長さ (ERR_REC_MAX) を超えることはできません。エラー・ログ・エントリーに入りきらないエラー・データは、他の場所に保存してください。エラー・ログ・エントリーの詳細データには、エラー・データとエラー・ログ・エントリーとの関連付けに使用できる情報を入れてください。
Err_Desc	発生したエラーを説明します。このフィールドには、エラーの説明メッセージ ID を指定します。この値は、エラーの説明メッセージ・セット「E」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
Err_Type	発生したエラーの重大度を示します。次に示すフィールド・ディスクリプターのいずれか 1 つを指定する必要があります。 **PERF** デバイスまたはコンポーネントの性能が許容水準以下に低下している状態 (パフォーマンス) **PERM** 回復不可能な状態 (永続) **PEND** デバイスまたはコンポーネントを使用できないことが差し迫っている状態 (切迫) **TEMP** 何回かの試行後、回復できた状態 (一時) **UNKN** エラーの重大度が判断できない状態 (不明) **INFO** 情報としてのエラー・ログ・エントリーの条件

項目	説明
Fail_Actions	障害に起因するエラーを訂正するための推奨処置を説明します。最高 4 個までの、コンマで区切られた推奨処置メッセージ ID のリストを指定できます。この値は、推奨処置メッセージ・セット「R」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。このフィールドは、Fail_Causes フィールドがブランクの場合は、ブランクでなければなりません。 リストされる推奨処置の順位は、処置に要する費用の大きさとエラーを訂正できる可能性によって決定されます。費用がほとんどかからないか、あるいはまったくかからない処置 (システムに対する影響がほとんどないか、あるいはまったくない処置) を、常にリストの最初に挙げてください。次に、エラーを訂正できる可能性が同じかほとんど同じものを費用のかからないものの順に列挙します。同様に残りの処置も、この可能性の降順に従って並べてください。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
Fail_Causes	発生したエラーの障害原因を示します。障害原因は、リソース・エラーの結果、そうなった状態として定義されます。このフィールドには、コンマで区切られた最高 4 個までの障害原因メッセージ ID を含むリストを指定できます。この値は、障害原因メッセージ・セット「F」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。障害原因は、エラーを訂正できる可能性の降順に列挙します。発生したエラーに適用できない場合は、このフィールドをブランクのままにすることができます。このフィールドがブランクの場合は、User_Causes フィールドか Inst_Causes フィールドのどちらかがブランクであってはなりません。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
Inst_Actions	インストール原因のエラーを訂正するための推奨処置を示します。このフィールドでは、最高 4 個までのメッセージ ID をコンマで区切ってリストすることできます。この値は、推奨処置メッセージ・セット「R」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。 Inst_Causes フィールドがブランクにされていた場合は、このフィールドもブランクでなければなりません。リストされる推奨処置の順位は、処置に要する費用の大きさとエラーを訂正できる可能性によって決定されます。費用がほとんどかからないか、あるいはまったくかからない処置 (システムに対する影響がほとんどないか、あるいはまったくない処置) を、常にリストの最初に挙げてください。次に、エラーを訂正できる可能性が同じかほとんど同じものを、費用のかからないものの順に列挙します。同様に残りの処置も、可能性の高いものから低いものの順に並べます。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
Inst_Causes	発生したエラーのインストール原因を示します。インストール原因は、初期インストールまたはリソース設定の結果、そうなった状態として定義されます。最高 4 個までの、コンマで区切られたインストール原因メッセージ ID を含むリストを指定できます。この値は、インストール原因メッセージ・セット「I」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。インストール原因は、可能性の高いものから低いものの順番に列挙してください。このフィールドは、発生したエラーに適用できない場合は、ブランクのまま残すことができます。このフィールドをブランクのままに残す場合は、 User_Causes フィールドか、 Fail_Causes フィールドは非ブランクでなければなりません。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
LABEL	エラー・ログ・テンプレートごとに必要な、最高 19 文字の固有ラベルを指定します。「#define #ERRID_label Error_ID」を含む文字列 (ここで、**Error_ID** 値はエラー・レコード・テンプレートに割り当てられた固有 ID) は、**-h** フラグがコマンド・ラインに指定されていれば、標準出力に書き出されます。 **注:** LABEL フィールドが 19 文字を超える場合、最初の 19 文字が受け入れられます。
Log	エラー発生時に、そのエラーに関するエラー・ログ・エントリーを作成すべきかどうかを指定します。ログ・フィールドは、「True」または「False」に設定できます。このフィールドをテンプレートから省略すると、その値はデフォルトの「True」になります。このフィールドが「False」に設定されると、Report および Alert フィールドは、無視されます。
Prob_Causes	発生したエラーに関する 1 つ以上の推定原因を示します。最高 4 個までの、コンマで区切られた推定原因メッセージ ID を含むリストを指定できます。この値は、推定原因メッセージ・セット「P」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。推定原因は、可能性の高いものから低いものの順番に列挙してください。少なくとも 1 つは推定原因が必要です。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
Report	このエラーについてログに記録されたオカレンスを、エラー・レポートの出力時に報告すべきかどうかを指定します。 Report フィールドは、「True」または「False」に設定することができます。このフィールドをテンプレートから省略すると、その値はデフォルトの「True」になります。

項目	説明
User_Actions	ユーザー原因のエラーを訂正するための推奨処置を示します。最高 4 個までの、コンマで区切られた推奨処置メッセージ ID のリストを指定できます。この値は、推奨処置メッセージ・セット「R」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。 User_Causes フィールドがブランクのままに残された場合は、このフィールドもブランクのままに残す必要があります。リストされる推奨処置の順位は、エラーに要する費用の大きさとエラーを訂正できる可能性によって決定されます。費用がほとんどかからないか、あるいはまったくかからない処置 (システムに対する影響がほとんどないか、あるいはまったくない処置) を、常にリストの最初に列挙してください。次に、エラーを訂正できる可能性が同じかほとんど同じものを、費用のかからないものの順に列挙します。同様に残りの処置も、可能性の高いものから低いものの順に並べます。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。
User_Causes	発生したエラーのユーザー原因を説明します。ユーザー原因は、サービス部門に問い合わせなくても訂正できる状態として定義されます。最高 4 個までの、コンマで区切られたユーザー原因メッセージ ID を含むリストを指定できます。この値は、ユーザー原因メッセージ・セット「U」から、エラー発生時に表示されるテキスト・メッセージを指定します。長さが 4 桁までの値は、無符号の 16 進数と解釈されます。ユーザー原因は、可能性の高いものから低いものの順に列挙します。このフィールドは、発生したエラーに適用できない場合は、ブランクのまま残すことができます。このフィールドをブランクのままに残す場合は、 Inst_Causes フィールドか、 Fail_Causes フィールドは非ブランクでなければなりません。このフィールドも XPG4 スタイル・メッセージを指定することができます。これについては後述します。

catname (カタログ名) は、現行テンプレートの XPG4 メッセージの検索に使用するメッセージ・カタログを指定する場合に使用されます。 catname は、前の "*!"カタログ指定子によって指定されたカタログを指定変更します。 XPG4 メッセージが含まれているテンプレートには、catname または "*!" で指定されたカタログが必要です。カタログ名は、引用符で囲む必要があります。カタログに対して絶対パス名が指定されない限り、メッセージ・カタログを検索する通常の規則が用いられます。

例えば、

```
catname = "mycat.cat"
```

が指定されると、mycat.cat は、 **/usr/lib/nls/msg/%L** 内にあると見なされます。

エラーの説明、推定原因、ユーザー原因、インストール原因、障害原因、推奨処置、および詳細データ ID メッセージは、エラー・ログ・メッセージ・カタログ内に保持されたエラー・メッセージ ID、または XPG4 メッセージでなければなりません。

エラー・メッセージ ID は 4 桁までの 16 進数字から構成され、前に "0x" は付きません。例えば、1234 または ABCD です。これらのメッセージを ID と一緒に表示する場合は、 **errmsg -w** コマンドを使用することができます。新規メッセージを追加する場合は、 **errmsg** コマンドを使用することができます。

XPG4 メッセージは、以下のフォーマットで指定されます。

```
{<set>, <number>, <"default text">}
```

セット、番号、およびデフォルト・テキストは、すべて必須です。記号メッセージ参照は、サポートされません。また、XPG4 メッセージが含まれているテンプレートに警報を出すことはできません。

XPG4 メッセージには、メッセージ・カタログを指定する必要があります。これは、"*!" カタログ指定子、もしくは catname キーワードを用いて行われます。

エラー・ログは、通常のエラー・メッセージ交換の機能のすべてをサポートするとは限りません。エラー・ログ・テンプレートに使用する文字列は、いくつかの制約事項に適合していなければなりません。

- 変数代入は、サポートされません。例えば、この文字列は、値を出力する形式指定子として使用することはできません。文字列に含めることができるのは、書式制御文字の「¥t」および「¥n」のみです。

- デフォルトのテキスト文字列は 1KB、すなわち 1024 バイトを超えることはできません。
- エラーの説明は、非詳細レポートの 40 文字域に出力されることに注意する必要があります。これらのレポートでは文字列のフォーマットは行われず、先頭 40 文字だけが出力されます。
- 文字列には、末尾の改行を入れてはなりません。これは errpt によって指定されます。

追加されるエントリーごとに、errupdate コマンドは *File*.h で指定したヘッダー・ファイルに書き込まれる固有の Error ID を割り当てます (この場合、*File* パラメーターは、errupdate コマンド入力ファイルの名前です)。 errupdate コマンドが標準入力から読み込んでいる場合は、#define ステートメントが標準出力に書き出されます。Class、Err_Desc、Err_Type、Fail_Actions、Fail_Causes、 Inst_Actions、Inst_Causes、Prob_Causes、User_Actions、User_Causes フィールドに指定された値、および Detail_Data .data_id 値は、そのエラーの固有のエラー ID の計算に使用されます。XPG4 テンプレートでは、Label も計算に含まれます。

Log、Report、および Alert フィールドの内容は、固有のエラー ID の計算には含まれません。したがって、特定エラーのログ、レポート、および警告の各特性は、エラー・レコード・テンプレート・リポジトリーに格納されたエラー・エントリーの定義内で、**errupdate** コマンドを用いていつでも変更することができます。また、詳細データ・フィールドの data_len および data_encode の部分も使用されないことに注意してください。

errupdate コマンドは、*File*.**undo** という名前の現行ディレクトリー内の取り消しファイルの作成も行います。 **errupdate** コマンドが標準入力から読み込んでいる場合は、**undo** ファイルは **errids.undo** ファイルに書き込まれます。 **undo** ファイルには、**errupdate** コマンドが行った変更を取り消すための、**errupdate** コマンドへの入力が含まれています。

エラー・レコード・テンプレート・リポジトリーの内容を表示する場合は、**errpt -t** コマンドを使用することができます。テンプレートは、実際のエラー・レポートのとおりに処理され出力されます。

> 注意**:** エラー・テンプレートを変更する場合は、これらのテンプレートが以後の更新で変更される可能性のあることに注意してください。すべての変更の記録を保持し、システムを更新した場合は、その変更を再度適用する必要があります。これは、通常は、新しいレベルのオペレーティング・システムに移行するなど、大幅なシステム更新を行った場合にのみ必要です。また、そのような記録をとっておくと、再インストールの場合にテンプレートを変更することもできます。記録を保持する最も簡単な方法は、テンプレートの変更を、常に 1 つの errupdate ソース・ファイルから行うことです。

フラグ

項目	説明
-c	構文エラーがないか入力ファイルを調べます。
-f	入力テンプレートのエラー ID と同一のエラー ID を持つテンプレートを含むすべてのテンプレートを更新します。
-h	エラー・テンプレートに割り当てられたエラー ID ごとに、#define ステートメントを作成します。コマンド・ラインにファイル名が与えられている場合は、そのファイル名に **.h** を追加すれば、ヘッダー・ファイル名となります。コマンド・ラインにファイル名が与えられていない場合は、#define ステートメントが標準出力に書き込まれます。
-n	エラー・レコード・テンプレート・リポジトリーへエラー・レコード・テンプレートが追加されないようにします。
-p	Alert フィールドが「True」に設定され、「SNA Generic Alert Architecture」(英文番号 GA27-3136) では認められない、エラーの説明、推定原因、ユーザー原因、ユーザー処置、インストール原因、インストール処置、障害原因、障害処置、または詳細データのデータ ID 値が入っている、テンプレートを追加または更新します。**errupdate** コマンドは、このフラグを指定しなければ、これらの特性をもつテンプレートを追加できません。
-q	undo ファイルの作成を抑止します。

項目	説明
-y *FileName*	*FileName* パラメーターで指定したエラー・レコード・テンプレート・ファイルを使用します。

セキュリティー

アクセス制御: なし。ただし、変更するテンプレート・ファイル (デフォルトでは /var/adm/ras/errtmplt) への書き込み権限を持っていなければなりません。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. エントリーを追加するには、次のように入力ファイルでエントリーを定義します。

```
+ CDROM_ERR22:
        Comment=        "Temporary CDROM read error"
        Class=  H
        Log=            True
        Report= True
        Alert=          False
        Err_Type=       TEMP
        Err_Desc=       E801
        Prob_Causes=    5004
        Fail_Causes=    E800, 6312
        Fail_Actions=   1601, 0000
        Detail_Data=    120, 11, HEX
        Detail_Data=    4, 8058, DEC
        Detail_Data=    4, 8059, DEC
```

   データを入力するには、次のように入力します。

```
errupdate <input file>
```

2. エントリー 99999999 のログ、レポート、警告の各特性を変更するには、次のように修正演算子 = (等号) を指定し、その後に固有のエラー ID と、変更されるエントリーの新しい特性を指定します。

```
errupdate
 =99999999:
 Report = False
 Log = True
```

3. エラー・レコード・テンプレート・リポジトリーから、エントリー 99999999 を削除するには、次のように削除演算子 - (負符号) を指定し、その後に削除するエントリーの固有のエラー ID を指定します。

```
errupdate
 -99999999:
```

4. この入力ストリームに指定された XPG4 メッセージ・カタログを「*!」で指定変更するには、「catname」キーワードを使用します。

```
*!mycat.cat
```

   * mycat.cat は、これ以降、すべての XPG4 メッセージに使用されます。

   * ただし、この 1 つを除きます。

```
+ CDROM_ERR23:
         Comment=        "Temporary CDROM read error"
         catname= "othercat.cat"
```

```
Class=  H
Log=            True
Report= True
Alert=          False
Err_Type=       TEMP
Err_Desc=       {1, 1, "CD ROM is broken"}
Prob_Causes=    {2, 1, "cause 1"},¥
               {2, 2, "Cause 2"}
Fail_Causes=    E800, 6312
Fail_Actions=   1601, 0000
Detail_Data=    120, 11, HEX
Detail_Data=    4, 8058, DEC
Detail_Data=    4, 8059, DEC
```

カタログ othercat.cat は CDROM_ERR23 テンプレートのみに使用されます。

注**:** テンプレートには、XPG4 メッセージと、従来のエラー ID またはコード・ポイントの両方を含めることができます。

ファイル

項目	説明
/usr/include/sys/errids.h	エラー ID を含むヘッダー・ファイルが入っています。
/usr/include/sys/err_rec.h	エラーのログを記録するための構造を含むヘッダー・ファイルが入っています。

関連資料:

437 ページの『errpt コマンド』

関連情報:

errlog コマンド

エラー・ログの概要

---

ethchan_config コマンド

目的

アダプターをイーサチャンネルに追加、またはイーサチャンネルから除去します。

構文

ethchan_config { **-a** [**-b**] | **-d** } [**-p** *ParentName*] *EtherChannel Adapter*

ethchan_config -c [**-p** *ParentName*] *EtherChannel Attribute NewValue*

ethchan_config -f [**-p** *ParentName*] *EtherChannel*

説明

このコマンドは、アダプターをイーサチャンネルに追加、またはイーサチャンネルから除去します。また、このコマンドを使用して、イーサチャンネル の属性を変更することができます。こうした追加、削除、および変更は、イーサチャネルのインターフェースを構成中のときでも行うことができます。つまり、イーサチャネルのインターフェースを切り離さなくても、アダプターを追加または除去したり、イーサチャネルのほとんどの属性を変更することができます。

フラグ

項目	説明
-a	指定した *Adapter* を、指定した *EtherChannel* に追加します。アダプターをバックアップ・アダプターとして追加する必要がある場合は、**-b** フラグを指定する必要があります。
-b	*Adapter* をバックアップ・アダプターとして追加することを指定します。このフラグは、**-a** フラグとともに使用する場合のみ有効です。
-c	指定した *EtherChannel* 属性の指定した *Attribute* を、指定した *NewValue* に変更します。
-d	指定した *Adapter* を、指定した *EtherChannel* から削除します。 **-b** フラグを **-d** フラグとともに使用すべきではありません。
-f	指定した *EtherChannel* のフェイルオーバーを強制します。フェイルオーバーは、アイドル・チャネルのアダプターが機能している場合にのみ発生します。アイドル・チャネルのアダプターが停止している場合、*EtherChannel* はアクティブなアダプター上で動作し続け、フェイルオーバーは行われません。
-p	イーサチャネルの親アダプターを指定します。イーサチャネル上に共用イーサネット・アダプター (SEA) が構成されている場合、イーサチャネルの属性を変更するには (例えば、アダプターの追加や削除)、このフラグを他のフラグと一緒に使用する必要があります。

パラメーター

項目	説明
Adapter	追加または削除するアダプターを指定します。
Attribute	指定したイーサチャンネルの属性を指定します。
EtherChannel	イーサチャンネルを指定します。
NewValue	指定したイーサチャンネルの指定した属性の新規の値を指定します。
ParentName	イーサチャネルの親アダプターを指定します。

終了状況

項目	説明
0	コマンドは正常に完了しました。
>0	エラーが発生しました。

例

1. アダプター `ent0` を `ent7` というイーサチャンネルに、バックアップ・アダプターとして追加するには、次のコマンドを入力します。

   ```
   /usr/lib/methods/ethchan_config -a -b ent7 ent0
   ```

2. `ent7` というイーサチャンネルの ping するアドレスの属性を、`10.10.10.10` に変更するには、次のコマンドを入力します。

   ```
   /usr/lib/methods/ethchan_config -c ent7 netaddr 10.10.10.10
   ```

3. `ent7` というイーサチャンネルのフェイルオーバーを、現在アクティブになっているチャネルからアイドル・チャネルに強制するには、次のコマンドを入力します。

   ```
   /usr/lib/methods/ethchan_config -f ent7
   ```

4. `ent32` という SEA に属する `ent18` というイーサチャネルから、アダプター `ent13` を削除するには、次のコマンドを入力します。

   ```
   /usr/lib/methods/ethchan_config -d -p ent32 ent18 ent13
   ```

制限

use_jumbo_frame 属性の使用をこのコマンドで変更することはできません。この属性を変更しようとすると、このコマンドはエラー・メッセージを出力します。

位置

/usr/lib/methods

---

ewallevent コマンド

目的

ログインしているすべてのユーザーにイベントまたはリアーム・イベントをブロードキャストします。

構文

ewallevent [-c] [-h]

説明

ewallevent スクリプトは、イベントまたはリアーム・イベントの発生時点でホストにログインしているすべてのユーザーに対して、イベントまたはリアーム・イベントに関するメッセージをブロードキャストします。イベントまたはリアーム・イベントの情報は、イベントまたはリアーム・イベントが発生すると、イベント応答リソース・マネージャーから生成される環境変数で、イベント応答リソース・マネージャーによって収集され、追加されます。このスクリプトは、イベント応答リソースによって実行されるアクションとして使用できます。また、その他のユーザー定義アクションを作成するためのテンプレートとして使用することもできます。このスクリプトは常に、メッセージを英語で戻します。

メッセージは、このスクリプトが応答アクションであるイベントまたはリアーム・イベントが発生したときにログインしていたすべてのユーザーのコンソールに、次のフォーマットで表示されます。

```
Broadcast message from user@host (tty) at hh:mm:ss...

severity event_type occurred for Condition condition_name
on the resource resource_name of resource_class_name at hh:mm:ss mm/dd/yy
The resource was monitored on node_name and resided on {node_names}.
```

ERRM 環境変数に関するイベント情報が戻されます。また、イベント情報には以下も含まれます。

Local Time
: イベントまたはリアーム・イベントが監視される時刻。ERRM から提供される実際の環境変数は、ERRM_TIME です。この値は、現地時間に変換され、表示される前に読み取り可能なフォームに変換されます。

このスクリプトは、環境変数値を収集し、**wall** コマンドを使用して、その時点でログインしているユーザーのコンソールへメッセージを書き込みます。

フラグ

-c
: **ewallevent** に、ERRM イベントの **ERRM_VALUE** をブロードキャストするよう指示します。 **-c** フラグが指定されると、**ewallevent** は SNMP トラップ・メッセージをブロードキャストします。

-h
: スクリプトの使用方法に関するステートメントを標準出力に書き込みます。

パラメーター

log_file
: イベント情報を記録するファイルの名前を指定します。*log_file* パラメーターには、絶対パスを指定する必要があります。

log_file は、循環ログとして処理され、サイズは 64KB の固定です。*log_file* がいっぱいになると、新しいエントリーによって既存の最も古いエントリーが上書きされます。

log_file が既に存在する場合、イベント情報はそのファイルに追加されます。*log_file* が存在しない場合は、ファイルが作成されて、イベント情報はそこに書き込まれます。

終了状況

0 スクリプトが正常に実行されました。

1 スクリプトの実行時にエラーが発生しました。

制限

1. このスクリプトは、ERRM が実行されているノード上で実行する必要があります。
2. **wall** コマンドは、その時点でログインしているユーザーのコンソールにメッセージを書き込むために使用されます。**wall** コマンドの詳細については、**wall** のマニュアル・ページを参照してください。

標準出力

-h フラグを指定すると、スクリプトの使用方法に関するステートメントが標準出力に書き込まれます。

例

1. **ewallevent** スクリプトが、重大な通知応答の中の事前定義アクションであり、リソース **/var** 上で使用中の **/var** スペース条件と関連付けられているとします。この条件に定義されたイベント式のしきい値に達すると、イベントが発生します。重大な通知応答が発生し、**ewallevent** が実行されます。ログインしているすべてのユーザーのコンソールに、次のメッセージが表示されます。

```
Broadcast message from joe@neverland.com (pts/6) at 18:42:03...

Critical event occurred for Condition /var space used
on the resource /var of filesys of IBM.FileSystem at 18:41:50 03/28/02
The resource was monitored on c174n05 and resided on {c174n05}.
```

2. リソース **/var** 上で使用中の **/var** スペース条件についてのリアーム・イベントが発生すると、ログインしているすべてのユーザーのコンソールに、次のメッセージが表示されます。

```
Broadcast message from joe@neverland.com (pts/6) at 18:42:03...

Critical rearm event occurred for Condition /var space used
on the resource /var of filesys of IBM.FileSystem at 18:41:50 03/28/02
The resource was monitored on c174n05 and resided on {c174n05}.
```

位置

/opt/rsct/bin/ewallevent

---

ex コマンド

目的

テキスト・ファイルのエディターです。

構文

ex[**-c** *Subcommand*] [**-l**] [**-R**] [**-s**] [**-t***Tag*] [**-V**] [**-w***Number*] [**-v**| **-**] [**+**[*Subcommand*]] [**-r**[*File*]] [*File...*]

説明

ex コマンドは、ex エディターを始動します。 ex エディターは、**edit** コマンド・エディターを含むエディター・ファミリーの一部で、気軽に使える初心者向けの edit エディターと、フルスクリーン編集エディターである **vi** コマンド・エディターがあります。vi エディターを直接呼び出せば、画面編集用の環境変数が設定できます。ex エディターは、vi エディターのサブセットであり、vi エディターの画面編集機能にアクセスできるので、単純な行エディターよりもはるかに強力です。

File パラメーターでは編集する 1 個または複数のファイルを指定します。複数のファイル名を指定すると、ex エディターはそれらのファイルを指定された順番に編集します。

注**:**

1. ユーザーのワークステーションの効率を決定するために、ex エディターは、ワークステーションの機能データベースである **terminfo** と、ユーザーが **TERM** 環境変数で指定したワークステーションのタイプを使用します。
2. **ex** コマンドは、特別な配慮をしなければ、現在行に影響を及ぼします。ファイルのいろいろの部分を処理するには、ファイル内の行のアドレッシング方法を理解している必要があります。
3. 標準入力が端末デバイスでない場合は、**-s** フラグを指定したかのようになります。

フラグ

項目	説明
-c *Subcommand*	編集を始める前に、ex エディター・サブコマンドを実行します。 `-c ''` のような null オペランドが入力されると、エディターは現在行をファイルの最後に配置します。(通常、ex エディターは、現在行をファイルの先頭、あるいは指定されたタグまたはパターンに設定します。)
-l	LISP コードを適切に字下げし、( ) (左右の小括弧)、{ } (左右の中括弧)、[[ ]] (左右 2 つずつの大括弧) の文字を、vi サブコマンドと解釈せずにテキストとして受け入れます。このフラグは、ビジュアルとオープンの各モードでアクティブになります。
-R	ファイルが変更されないよう **readonly** オプションを設定します。
-s	すべての対話型ユーザー・フィードバックを抑止します。このフラグを使用すると、ファイルの入力エラーと出力エラーが発生しても、有効なエラー・メッセージは生成されません。このフラグは、**-** フラグと同様に機能します。TERM の値と実装のデフォルトの端末タイプを無視し、その端末がオープン・モードまたはビジュアル・モードをサポートするタイプであると仮定します。
-t *Tag*	*Tag* パラメーターで指定されるタグの入ったファイルをロードし、そのタグの位置を現在行にしてエディターを起動します。このフラグを使うには、最初に **ctags** コマンドを使って関数名とその位置からなるデータベースを作成する必要があります。
-w*Number*	デフォルトのウィンドウ・サイズを *Number* で指定された数に設定します。
-v	vi エディターを起動します。 注**: -v** フラグを選択すると、画面編集機能やカーソル移動機能などのサブコマンドの拡張セットが使用可能になります。**vi** コマンドのセクションを参照してください。
-V	エディターを詳細モードで起動します。
-	すべての対話型ユーザー・フィードバックを抑止します。このフラグを使用すると、ファイルの入出力エラーが発生しても、有効なエラー・メッセージは生成されません。このフラグは、**-s** フラグと同様に機能します。
+[*Subcommand*]	指定されたエディター検索またはサブコマンドで編集を始めます。パラメーターを入力しないと、**+***Subcommand* はファイルの最後に現在行を配置します。通常、ex エディターは現在行をファイルの先頭、あるいは指定されたタグまたはパターンの位置に設定します。
-r [*File*]	エディターかシステムがクラッシュした後でファイルを回復させます。 *File* パラメーターを指定しないと、保存されている全ファイルのリストが表示されます。

終了状況

次の終了値が戻されます。

項目	説明
0	正常終了。
>0	エラーが発生しました。

ファイル

項目	説明
/usr/lbin/exrecover	回復サブコマンド
/usr/lbin/expreserve	保管サブコマンド
$HOME/.exrc	エディター始動ファイル
./.exrc	エディター始動ファイル
/var/tmp/Ex*nnnnn*	エディター一時ファイル
/var/tmp/Rx*nnnnn*	バッファーの一時ファイル
/var/preserve	保管ディレクトリー

関連資料:

299 ページの『ed または red コマンド』

337 ページの『edit コマンド』

関連情報:

ctags コマンド

vi コマンド

TERM 環境変数

---

execerror コマンド

目的

標準エラーにエラー・メッセージを書き出します。

構文

execerror

説明

execerror コマンドは、実プログラムが正しくロードされない場合に、**exec** サブルーチンによって実行されます。 execerror コマンドには、実行中のファイルの名前と、ゼロ個または 1 個以上のローダー・エラー・メッセージの文字列が渡されます。個々のローダー・エラー・メッセージの文字列には、エラー番号と、それに続くエラー・データが含まれます。

例

execerror コマンドは次のように使用します。

```
char *buffer[1024];
buffer[0] = "execerror" ;
buffer[1] = "name of program that failed to load";
loadquery(L_GETMESSAGES, &buffer[2], sizeof buffer -8);
execvp("/usr/sbin/execerror",buffer);
```

このサンプル・コードでは、アプリケーションは、メッセージが標準エラーに書き出された後に終了します。

ファイル

項目	説明
/usr/sbin/execerror	**execerror** コマンドが入っています。

関連情報:

exec コマンド

loadquery コマンド

---

execrset コマンド

目的

rset に付加されたプログラムまたはコマンドを実行します。

構文

execrset [**-P**] [**-F**] **-c** *CPUlist* [**-m** *MEMlist*] **-e** *command* [*parameters*]

または

execrset [**-P**] [**-F**] [**-S**] *rsetname* [**-e**] *command* [*parameters*]

説明

execrset コマンドは、**rset** に付加されたコマンドを実行します。これにより、指定されたコマンドは、rset に含まれているプロセッサーおよびメモリー領域だけに実行が制限されます。システム・レジストリー内の **rset** 名を使用すると、コマンドが使用を許可されるプロセッサーおよびメモリー領域を指定できます。また、指定されたプロセッサーおよびメモリー領域が含まれている **rset** は、プロセスに付加することもできます。

フラグ

項目	説明
-F	**execrset** コマンドを強制的に実行します。このフラグにより、コマンドの発行前に、プロセス内の bindprocessor バインドおよびすべてのスレッドの **rset** が除去されます。 **-P** フラグも指定すると、コマンドの発行前に、プロセスから有効 **rset** およびすべてのスレッドの **rset** が切り離されます。
-P	区画 rset として **rset** を付加します。
-c *CPUlist*	プログラムまたはコマンドを実行するプロセスに付加する **rset** に組み込まれる CPU のリスト。これは 1 つ以上の CPU または CPU の範囲です。
-m *MEMlist*	**rset** に組み込まれるメモリー領域のリスト。これは 1 つ以上のメモリー領域または範囲です。
-e *command* [*parameters*]	実行するコマンドおよびパラメーターを指定します。 **-e** フラグは、このコマンド内で最後に指定する必要があります。
-S	このプロセスを単一スレッド化モードで実行するようにスケジュールする必要があることを示すヒント。指定された rset に組み込まれている各物理プロセッサーのハードウェア・スレッドの 1 つだけが、ジョブをスケジュールするのに使用されます。物理プロセッサーのすべてのハードウェア・スレッドが、指定された rset に組み込まれていない場合は、そのプロセッサーは無視されます。指定された rset は排他的 rset でなければなりません。そうでないと、コマンドは失敗します。このフラグを指定すると、ジョブを単一スレッド動作で実行することができます。

パラメーター

項目	説明
rsetname	プログラムまたはコマンドを実行するプロセスに付加されるシステム・レジストリー内の **rset** の名前。

セキュリティー

ユーザーは、root 権限または **CAP_NUMA_ATTACH** 機能が必要です。ユーザーがコマンドのプロセスに区画 rset を付加するためには (**-P** フラグ)、root 権限が必要です。

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. CPU 0 から 7 で test1 プログラムを実行するには、次のように入力します。

   ```
   execrset -c 0-7 -e test1
   ```

2. **test/cpus0to15** という名前の rset に付加して「test2 parm1 parm2」プログラムを実行するには、次のように入力します。

   ```
   execrset test/cpus0to15 test parm1 parm2
   ```

3. CPU 0 で **ls -l** コマンドを実行するには、次のように入力します。

   ```
   execrset -c 0 -e ls -l
   ```

ファイル

項目	説明
/usr/bin/execrset	**execrset** コマンドが含まれています。

関連資料:

110 ページの『detachrset コマンド』

関連情報:

attachrset コマンド

lsrset コマンド

mkrset コマンド

expand コマンド

目的

タブをスペースに変更して標準出力に書き出します。

構文

expand [**-t** *TabList*] [*File ...*]

expand [*-tabstop*] | [*-tab1,tab2,...,tabn*] [*File ...*]

説明

expand コマンドは、指定したファイルまたは標準入力を標準出力に書き出して、タブ文字を 1 つ以上のスペース文字に置き換えます。バックスペース文字は出力にコピーされ、タブ・ストップ計算に使用される桁位置のカウントが減少します。桁位置のカウントがゼロよりも小さくなることはありません。

注**:** *File* パラメーターは、テキスト・ファイルでなければなりません。

フラグ

項目	説明
-t *TabList*	タブ・ストップの位置を指定します。タブ・ストップのデフォルト値は 8 桁目です。 *TabList* 変数は、1 桁または複数桁の 10 進整数で構成する必要があります。複数の整数は昇順で指定し、コンマまたはブランク文字で区切り、整数の前後を引用符で囲まなければなりません。単一の *TabList* 変数は、タブ・ストップを個々の桁位置に等しい数値に設定します。複数の *TabList* 変数は、タブ・ストップを *TabList* 変数内の整数に対応する桁位置に設定します。 **expand** コマンドが、*TabList* 変数内で最後に指定した桁位置を超えてタブ・ストップを処理すると、タブ・ストップは出力内で単一スペース文字に置き換えられます。

パラメーター

項目	説明
tabstop	単一引数として指定します。デフォルトの 8 の代わりに、*tabstop* SPACE 文字を別に設定します。
tab1, tab2,..., tabn	*-tab1,tab2,...,tabn* によって指定された列に TAB 文字を設定します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. text.fil 内で、タブ・ストップを等間隔に調整するには、次のように入力します。

   ```
   expand -t 3 text.fil
   ```

   text.fil の内容が以下のとおりとします。

   ```
   1       2        3456789
   ```

   expand コマンドは次のように表示します。

   ```
   1  2        3456789
   ```

2. text.fil 内のタブ・ストップを、可変間隔に調整するには、次のように入力します。

   ```
   expand -t 3,15,22  text.fil
   ```

   または

   ```
   expand -t "3 15 22" text.fil
   ```

   text.fil の内容が以下のとおりとします。

```
1       2       3       456789
```

expand コマンドは次のように表示します。

```
1  2       3       456789
```

ファイル

項目	説明
/usr/bin/expand	**expand** コマンドが入っています。

関連情報:

newform コマンド

tab コマンド

unexpand コマンド

入出力リダイレクト

---

expfilt コマンド

目的

フィルター規則をエクスポート・ファイルにエクスポートします。

構文

expfilt [**-p**] [**-q**] [**-r**] [**-v 4** | **6**] **-f** *directory* [**-l** *filt_id_list*]

説明

expfilt コマンドは、フィルター規則を、エクスポート・テキスト・ファイル (1 つ以上) にエクスポートする場合に使用します。この規則は、**impfilt** コマンドで使用できます。これは、複数のマシンに、類似した規則を定義したい場合は便利です。

注**:** マシンのフィルターの説明は、別のマシンでは何も意味を持たなかったり、誤操作を招いたりする場合があります。このフィールドはエクスポートされません。

このコマンドの IPsec フィルター規則は、**genfilt** コマンドまたは IPsec smit (IP バージョン 4 または IP バージョン 6) を使用して構成することができます。

フラグ

項目	説明
-f*directory*	エクスポートされるテキスト・ファイルを作成するディレクトリーを指定します。ディレクトリーがない場合は、作成されます。
-l *filt_id_list*	エクスポートしたいフィルター規則の ID を表示します。フィルター規則の ID は、「,」もしくは「-」で分離することができます。このフラグが使用されない場合は、適用できる IP バージョンのフィルター規則テーブルで定義されたフィルター規則がすべて、エクスポートされます。
-p	事前定義規則を許可します。
-q	抑止モードを指定します。**stdout** への出力を抑止します。

項目	説明
-r	未加工モードを指定します。フィルター規則を現状のままエクスポートし、ルールに関する指示を変えたりはしません。このフラグは、フィルター規則を現状のままエクスポートしたりインポートしたりする場合に使用します。例えば、他のマシンに構成を保管したり、その複製を作成したりする場合です。 **-r** フラグを指定すると、トラフィックの方向が保存されます。例えば、ホスト 10.0.0.1 に、 10.0.0.2 からのインバウンド・トラフィックを許可するという規則があるとすると、 expfilt に -r フラグを指定することにより、同じフィルター規則が書き込まれます。 **-r** フラグを省略すると、エクスポート・ファイルの方向がインバウンドからアウトバウンドに切り替えられます。
-v	エクスポートしたいフィルター規則の IP バージョン。値 **4** は、IP バージョン 4 を指定し、値 **6** は IP バージョン 6 を指定します。このフラグが使用されないときは、IP バージョン 4 と IP バージョン 6 の両方の規則がエクスポートされます。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

関連情報:

impfilt コマンド

---

explain コマンド

目的

対話式のシソーラスを提供します。

構文

explain

説明

explain コマンドは、**diction** コマンドで検索される英語の語句について、対話式シソーラスを提供します。**explain** コマンドを使う前に、**diction** コマンドを使って言葉遣いに誤りのある語句のリストを作成しておいてください。 **explain** コマンドを使うと、システムは語句を入力するようにプロンプトを表示し、構文的に正しい語句にして応答します。その後で、続けて語句を入力できます。また、Ctrl-D キー・シーケンスにより終了します。

他のコマンド・ラインのパラメーターは無効です。

ファイル

項目	説明
/usr/lib/explain.d	シソーラスが入っています。

関連資料:

151 ページの『diction コマンド』

---

explore コマンド

目的

WebExplorer World Wide Web (WWW) ブラウザーを始動します。

構文

explore [**-i***FileName*] [**-t***Number*] [**-q**] [[**-url**] *URL*]

説明

explore コマンドは、ホーム・ページの文書を見るために WebExplorer メインウィンドウをオープンし、Uniform Resource Locator (URL) に接続します。

フラグ

項目	説明
-i *FileName*	代替初期化ファイルを指定します。この場合、*FileName* には、デフォルトの **$HOME/.explore-preferences** の代わりに使用するファイルの絶対パス名を指定します。これによって、一連の代わりのユーザー設定によって WebExplorer を始動できます。
-t *Number*	イメージのロードに使用するスレッドの数を指定します。この場合、*Number* はイメージ・ローダー・スレッドの数です。各スレッドは、メインウィンドウの状況領域内に表示されます。最高 8 個までのスレッドを指定できます。デフォルトは 4 です。
-q	抑止モードを指定します。これによって、アプリケーションの始動時には WebExplorer のタイトル・ウィンドウが抑止され、終了時には確認ウィンドウが略されます。
-url *URL*	WebExplorer の始動時に特定の文書をロードするように指定します。この場合、*URL* はロードする文書の URL です。 WebExplorer でホーム・ページの文書が定義されている場合には、この URL はそれを指定変更します。 URL の前には **-url** フラグを付けなくてもかまいません。 URL のみを指定しても、WebExplorer は受け入れます。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: 全ユーザー

監査イベント: 該当なし

例

タイトル・ウィンドウを表示せずに、ブラウザーを始動して、直接 Dilbert Zone の URL に進むには、次のように入力します。

```
explore -q  http://www.unitedmedia.com/comics/dilbert/
```

または

```
explore -q  -url http://www.unitedmedia.com/comics/dilbert/
```

ファイル

項目	説明
/usr/lpp/explorer/bin/explore	**explore** コマンドが入っています。
$HOME/.explore-preferences	使用する色の数などのユーザーによる設定を指定するための初期化ファイルが入っています。
$HOME/.mailcap	mimetype を外部ビューアーにマップするための構成ファイルが入っています。
$HOME/.mimetypes	mimetype を外部ビューアーにマップするためのユーザー定義の構成ファイルが入っています。この構成ファイルは、**Configure Viewers** ダイアログによって設定されます。このファイルは、**.mailcap** 設定を指定変更します。

exportfs コマンド

目的

ディレクトリーを NFS クライアントにエクスポートおよびアンエクスポートします。

構文

/usr/sbin/exportfs [**-a**] [**-v**] [**-u**] [**-i**] [**-f***File*] [**-F**] [**-o***Option* [*,Option ...*]] [**-V** *Exported Version*] [*Directory*]

説明

exportfs コマンドは、ネットワーク・ファイルシステム (NFS) のクライアントがローカル・ディレクトリーをマウントできるようにします。このコマンドは、通常、システム始動時に **/etc/rc.nfs** ファイルによって起動され、**/etc/exports** ファイル内の情報を使用して、1 つ以上のディレクトリー (必ず絶対パス名で指定) をエクスポートします。

/etc/xtab ファイルには、現在エクスポート中のディレクトリーのリストが入っています。このファイルを表示するには、フラグまたは引数を指定しないで **exportfs** コマンドを実行します。ファイルの変更、または、そのファイルのディレクトリーの特性変更を行うために、root ユーザーは **/etc/exports** ファイルを編集して、**exportfs** コマンドを実行できます。この操作は常時行えます。 **/etc/xtab** ファイルを直接編集することは、絶対に避けてください。

注**:**

1. 現在エクスポートされているディレクトリーと同一のファイルシステム内に存在し、かつ、そのディレクトリーの親ディレクトリーまたはサブディレクトリーのいずれかであるディレクトリーは、エクスポートできません。
2. NFS バージョン 2 および 3 では、ディレクトリーもファイルもエクスポートできます。 NFS バージョン 4 のアクセスでは、ディレクトリーだけがエクスポートできます。

3. **/etc/exports** ファイル内に、異なるバージョン 2 (または 3) と 4 を持つ同一ディレクトリーに対して 2 つのエントリーが存在する場合、**exportfs** コマンドは両方のエントリーをエクスポートします。
4. NFS バージョン 2 (または 3) と 4 のオプションが 1 つのディレクトリーに対して同一である場合、`-vers=3:4` を指定する **/etc/exports** ファイルに存在するエントリーは 1 つとなります。

フラグ

項目	説明
-a	エクスポート・ファイルにリストされているすべてのディレクトリーをエクスポートします。
-v	エクスポートまたはアンエクスポートされる各ディレクトリーの名前を出力します。
-u	指定したディレクトリーをアンエクスポートします。 **-a** フラグと併用すると、エクスポートされたすべてのディレクトリーがアンエクスポートされます。 **-a** および **-f** のフラグと一緒に使用すると、指定のエクスポート・ファイル内のすべてのディレクトリーがアンエクスポートされます。
-i	エクスポート・ファイルに指定されていないディレクトリーをエクスポートできるようにするか、あるいはエクスポート・ファイルに記述されているオプションを無視します。 **-f** フラグを使用して代替ファイルを指定する場合を除き、 **exportfs** コマンドは通常、エクスポートされたディレクトリーに関連付けられているオプションについて、**/etc/exports** ファイルを調べます。
-f *File*	**/etc/exports** ファイルの代わりに、エクスポート可能なディレクトリーのリストが入っているエクスポート・ファイルを指定します。このファイルは、**/etc/exports** ファイルと同じフォーマットにしてください。注 : この代替ファイルは、システムと NFS の始動時にディレクトリーを自動的にエクスポートするために使用されることはありません。 **/etc/exports** ファイルは、システムの始動時にエクスポートするディレクトリーを指定するためにサポートされる唯一のファイルです。
-F	強制アンエクスポートの実行を指定します。このフラグは、**-u** フラグを指定する場合のみ、一緒に使用します。このフラグは、V2/V3 エクスポートのアンエクスポート時には効果がありません。V4 のアンエクスポートでは、関連する状態のために失敗するおそれがあります。このフラグは、V4 エクスポートに関連する任意の状態を強制的に解除します。

項目	説明
-o*Options*	エクスポートされるディレクトリーにオプションの特性を指定します。コンマで区切れば、複数の変数を入力できます。*Client* パラメーターをとるオプションの場合は、*Client* でホスト名、ドット付きの IP アドレス、ネットワーク名、またはサブネット指定子を指定することができます。サブネット指定子は @ *host* /*mask* の形式をとります。ここで、*host* はホスト名またはドット付き IP アドレスのどちらかであり、*mask* はアクセスをチェックする際に使用するビットの数を指定します。 *mask* が指定されていない場合は、全マスクが使用されます。例えば、指定子 `@client.group.company.com/16` は、`company.com` サブネット上にあるすべてのクライアントと一致します。指定子 `@client.group.company.com/24` は、`group.company.com` サブネット上のクライアントとのみ一致します。次のオプションから選択します。 **ro** ディレクトリーを読み取り専用許可でエクスポートします。指定されていない場合は、ディレクトリーは読み取り/書き込み許可でエクスポートされます。 **ro=***Client[:Client]* ディレクトリーを読み取り専用許可で指定のクライアントにエクスポートします。ディレクトリーを読み取り/書き込み許可でリストに指定されていないクライアントにエクスポートします。読み取り/書き込みリストが指定されている場合は、読み取り専用リストは指定できません。 **rw** ディレクトリーを読み取り/書き込み許可ですべてのクライアントにエクスポートします。 **rw=***Client [:Client]* ディレクトリーを読み取り/書き込み許可で指定されたクライアントにエクスポートします。ディレクトリーを読み取り専用でリストにないクライアントにエクスポートします。読み取り専用リストが指定されている場合は、読み取り/書き込みリストは指定できません。 **anon =***UID* root ユーザーから要求があれば、実効ユーザー ID として *UID* 値を使用します。 このオプションのデフォルト値は -2 です。 NFS バージョン 2 および NFS バージョン 3 では、*anon* オプションの値を -1 に設定すると、匿名アクセスが使用不可になります。したがって、デフォルトでは、保護されている NFS は無保護の要求を無名として受け入れます。高レベルの保護を望むユーザーは、*anon* を -1 に設定することで、この機能を使用不可にできます。 **root=***Client[:Client]* リスト内の特定のクライアントからの root アクセスを可能にします。root リスト内にホストを入れても、他のオプションの意味体系は変更されません。例えば、このオプションは、root リスト内にはあるが、アクセス・リスト内にはないホストからのマウント・アクセスを拒否します。 **access=***Client[:Client,...]* 登録された各クライアントにマウント・アクセス権を与えます。クライアントには、ホスト名またはネットグループ名を使用できます。 **/etc/netgroup** データベースのリスト内の個々のクライアントが最初に検査され、次に **/etc/hosts** データベースのリスト内の個々のクライアントが検査されます。デフォルト値の場合は、どのコンピューターにでも、指定されたディレクトリーをマウントできます。 **secure** クライアントに対し、ディレクトリーにアクセスする際により安全なプロトコルの使用を要求します。

項目	説明
-o *Options* (続き)	**sec=***flavor[:flavor...]* このオプションは、エクスポートされるディレクトリーの下のファイルにアクセスするために使用できるセキュリティー・メソッドのリストを指定するために使用されます。ほとんどの exportfs オプションは、**sec** オプションを使用してクラスター化することができます。 **sec** オプションに続くオプションは、その前の **sec** オプションに属すものと見なされます。 **sec** スタンザは任意の数だけ指定できますが、各セキュリティー・メソッドはそれぞれ 1 回しか指定できません。各 **sec** スタンザ内では、**ro**、**rw**、**root** および **access** オプションは一度だけ指定できます。 **public**、**anon** および **vers** オプションだけは、エクスポートに関してグローバルなオプションであると考えられます。いずれかのセキュリティー・メソッドを指定するために **sec** オプションを使用する場合は、すべてのセキュリティー・メソッドの指定にこのオプションを使用する必要があります。 **sec** オプションの指定がない場合は、すべての種類の認証が許可されます。 使用できるフレーバー値には、次のものがあります。 **sys** UNIX 認証。これはデフォルトのメソッドです。 **dh** DES 認証。 **none** マウント要求がエクスポートで指定されていない認証フレーバーを使用している場合、マウント要求を無名のクリデンシャルで進行することを許可します。 **krb5** Kerberos。認証専用。 **krb5i** Kerberos。認証と保全性。 **krb5p** Kerberos。認証、保全性、およびプライバシー。 **secure** オプションは指定できますが、**sec** オプションと一緒には指定できません。**secure** オプションの使用は推奨できません。これは除かれる可能性があります。代わりに **sec=dh** を使用してください。 **vers=***version_number[:version_number...]* エクスポートされるディレクトリーにアクセスすることが許可される NFS のバージョンを指定します。有効なバージョンは 2、3 および 4 です。バージョン 2 と 3 は、どちらか片方だけを選択することはできません。バージョン 2 またはバージョン 3 を指定すると、NFS バージョン 2 と NFS バージョン 3 の両方によるアクセスが許可されます。バージョン 4 は排他的に選択することができます。デフォルトでは、NFS プロトコルのバージョン 2 と 3 を使用したアクセスが許可されます。 **exname=***external-name* 指定された外部名によってディレクトリーをエクスポートします。外部名は、**nfsroot** 名から始まらなければなりません。 **nfsroot** 名の説明については、**/etc/exports** ファイルの記述を参照してください。このオプションは、NFS バージョン 4 プロトコルによるアクセス用にエクスポートされるディレクトリーにだけ使用できます。 **deleg={yes | no}** 指定されたエクスポートに対して、ファイル委任を使用可能または使用不可にします。このオプションは、このエクスポートについてシステム全体にわたる委任有効化を指定変更します。システム全体にわたる有効化は、**nfso** によって行われます。

項目	説明
-o *Options* (続き)	**refer=***rootpath@host***[+***host***][:***rootpath@host***[+***host***]]** ネームスペース委託は、指定されたパスで作成されます。この委託は、操作を続行可能な指定された代替ロケーションにクライアントを経路指定します。委託は特殊なオブジェクトです。委託以外のオブジェクトが指定のパスに存在する場合は、エクスポートは不許可となり、エラー・メッセージが表示されます。指定のパスに何も存在しない場合は、1 つの委託オブジェクト (このオブジェクトに至るパス名ディレクトリーを含む) がその指定のパス上に作成されます。複数の委託を 1 つのファイルシステム内で作成することができます。委託を **nfsroot** に対して指定することはできません。名前 `localhost` を *hostname* として使用することはできません。**refer** オプションは、バージョン 4 のエクスポートにのみ使用できます。エクスポート指定によりバージョン 2 またはバージョン 3 のアクセスが許可される場合は、エラー・メッセージが表示され、エクスポートは不許可になります。委託オブジェクトのアンエクスポートは、委託オブジェクトから委託ロケーション情報を除去する効果があります。このオブジェクト自体は、アンエクスポートによって除去されません。このオブジェクトを除去するには、**rm** を使用してください。管理者は、適切なデータが委託サーバーで使用可能となるようにする必要があります。このオプションは、AIX 5L バージョン 5.3 (5300-03 推奨メンテナンス・パッケージ適用) 以降でのみ使用できます。 **注:** 委託エクスポートは、複製がサーバー上で使用可能な場合にのみ作成することができます。複製を使用可能にするには、**chnfs -R on** を使用します。

項目	説明
-o *Options* (続き)	**replicas=***rootpath@host***[+***host***][:***rootpath@host***[+***host***]]** レプリカ・ロケーション情報は、エクスポート・パスに関連付けられます。現行サーバーが使用不可になった場合には、NFS バージョン 4 のクライアントが、このレプリカ情報を使用して、指定された代替ロケーションに操作をリダイレクトできます。管理者は、適切なデータがレプリカ・サーバーで使用可能となるようにする必要があります。レプリカ情報はファイルシステム全体に適用されるため、指定されたパスはファイルシステムのルートでなければなりません。パスがファイルシステムのルートでない場合は、エクスポートは不許可となり、エラー・メッセージが表示されます。名前 `localhost` を *hostname* として使用することはできません。この **replicas** オプションは、バージョン 4 のエクスポートにのみ使用できます。このオプションがバージョン 2 またはバージョン 3 のアクセスを許可するエクスポートで使用される場合は、操作は許可されますが、レプリカ情報はバージョン 2 およびバージョン 3 のサーバーによって無視されます。エクスポートしようとするディレクトリーがレプリカ・リスト内にない場合は、エントリー *exported directory@current host* は、最初のレプリカ・ロケーションとして追加されることになります。このオプションは、AIX 5.3 (5300-03 適用) 以降でのみ使用できます。レプリカ・エクスポートを作成可能なのは、複製がサーバー上で使用可能な場合にのみです。デフォルトでは、複製は使用不可にされます。レプリカ・エクスポートをシステムのブート時に作成することになる場合は、**chnfs -R on** コマンドを使用して複製を使用可能にする必要があります。レプリカ・ロケーションを **nfsroot** に対しても指定することができます。この指定は、**chnfs -R** *host*[*+host*] によってのみ行うことができます。現在のホストがリスト内に指定されていない場合は、このホストは最初のレプリカ・ホストとして追加されます。 **nfsroot** は指定されたホストの **nfsroots** に対してのみ複製されるため、この場合の *rootpath* は不要または不許可となります。 **chnfs** プログラムを使用して、複製を使用可能または使用不可にすることができます。NFS バージョン 4 のエクスポートがすべてアクティブでない場合は、複製モードの変更を実施できるだけです。サーバーの複製モードを変更した場合は、前の複製モード時にサーバーが発行したファイル・ハンドルを、サーバーは受け入れません。これにより、旧ファイル・ハンドルを保持するクライアントでアプリケーション・エラーが生じる可能性があります。サーバーの複製モードを変更する際は、注意してください。可能であれば、サーバーへのマウントがあるすべてのクライアントは、サーバーの複製モードが変更される前に、それらをアンマウントする必要があります。ディレクトリーに関連付けられたレプリカ・ロケーション情報は、レプリカ・リストの変更およびディレクトリーの再エクスポートにより変更できます。新規レプリカ情報が旧レプリカ情報と置き換わります。NFS クライアントは、レプリカ情報を定期的にリフレッシュすることを求められます。サーバーがエクスポート用のレプリカ情報を変更する場合は、クライアントが認識するのに時間がかかる可能性があります。このことは、新規レプリカ・ロケーションの追加時には、大した問題ではありません。その理由は、これは、古い情報を保持するクライアントが引き続き正しい (たとえ不完全でも) レプリカ情報をもっているためです。レプリカ情報を除去すると、問題が生じる可能性があります。これは、クライアントが一定期間、誤ったレプリカ情報を保持することになる可能性があるためです。クライアントが新規情報を検出するのを援助するために、**exportfs** は複製されたディレクトリーへのタッチを試みます。これにより、ディレクトリーのタイム・スタンプが変更されるため、クライアントがディレクトリーの属性を再フェッチするようになります。ただし、複製されたファイルシステムが読み取り専用の場合には、この操作が不可能な場合があります。あるディレクトリー用のレプリカ情報の変更時は、情報の変更と新規情報に対するクライアントの認識との間に若干の待ち時間が生じる可能性があることに注意してください。

項目	説明
-o *Options* (続き)	**noauto** レプリカの指定を現状のまま受け入れます。レプリカ・ロケーションの 1 つとして 1 次ホスト名を自動挿入しません (指定されていない場合)。 **scatter** **refer** または **replicas** オプションで指定されたサーバーから代替ロケーション・リストをどのように生成するかを定義します。**noauto** オプションを使用しない場合、代替ロケーション・リストにはレプリカ・ロケーションの 1 つとして 1 次ホスト名も含まれます。**scatter** オプションは、NFS バージョン 4 プロトコルによるアクセス用にエクスポートされるディレクトリーにのみ適用されます。 **scatter** オプションには次の 3 つの許容値があります。 **full** 代替ロケーションの組み合わせを形成するために、すべてのサーバーが分散されます。 **partial** すべての組み合わせの最初のロケーションが、**refer** または **replicas** オプションで指定された最初のサーバーに固定されます。残りのロケーションと最初のロケーションは分散され、`scatter=full` 方式を使用して分散されたようになります。 **none** 分散は使用されません。この値は、使用可能になっている分散を使用不可にすることもできます。 クライアントの属性が変わったときは、そのクライアントをパラメーターとして含むすべてのエクスポート・エントリーは、再びエクスポートしてください。クライアントの属性を変更する可能性のあるイベントとしては、ネットグループの変更またはクライアントの IP アドレスの変更などがあります。この再エクスポートに失敗すると、サーバーは古いクライアント情報を使用することになるおそれがあります。
-V *Exported Version*	バージョン番号を指定します。有効なバージョン番号は 2、3 および 4 です。

Solaris の互換性

exportfs コマンドは、**share**、**shareall**、**unshare**、または **unshareall** として呼び出すことができます。 **exportfs** コマンドを **share** または **shareall** として呼び出す場合は、その機能性はそれぞれ、**exportfs** と **exportfs -a** と同等ですが、セキュリティー・メソッドを指定するために **sec** オプションの使用が必要である点が異なります。 **exportfs** コマンドを **unshare** または **unshareall** として呼び出す場合は、その機能性はそれぞれ、**exportfs -u** と **exportfs -u -a** と同等です。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. **/etc/exports** ファイル内の、すべてのディレクトリーをエクスポートするには、次のように入力します。

   ```
   exportfs -a
   ```

2. **/etc/exports** ファイルから、ディレクトリーを 1 つエクスポートするには、次のように入力します。

   ```
   exportfs /home/notes
   ```

   この例では、`/home/notes` ディレクトリーがエクスポートされます。

注**:** このコマンドを機能させるためには、**/etc/exports** ファイルで **/home/notes** ディレクトリーを指定する必要があります。

3. ディレクトリーをアンエクスポートするには、次のように入力します。

   ```
   exportfs -u /home/notes
   ```

   この例では、/home/notes ディレクトリーのエクスポートが解除されます。

4. 現在エクスポート中のディレクトリーの名前を表示するには、次のように入力します。

   ```
   exportfs -v
   ```

5. **/etc/exports** ファイルに指定されていないディレクトリーをエクスポートするには、次のように入力します。

   ```
   exportfs -i /home/zeus
   ```

   この例では、/home/zeus ディレクトリーが無制限にエクスポートされます。

6. ディレクトリーをエクスポートして、ネットグループ・メンバーにこのディレクトリーへのアクセス権を与えるには、次のように入力します。

   ```
   exportfs access=cowboys:oilers /home/notes -o
   ```

   この例では、/home/notes ディレクトリーがエクスポートされ、cowboys と oilers というホスト・コンピューターのユーザーに、アクセス権が与えられます。

7. **/etc/exports** ファイルから、別のオプションを持つディレクトリーをエクスポートするには、次のように入力します。

   ```
   exportfs -i -o root=zorro:silver /directory
   ```

   この例では、/directory ディレクトリーがエクスポートされ、**/etc/exports** ファイルに指定した権限とは無関係に、root ユーザーに zorro と、silver の各ホスト・コンピューターへのアクセスが許可されます。

8. **/common/docs** ディレクトリーを、Kerberos 認証を使用しているクライアントには書き込み許可とともに、また、UNIX 認証を使用しているクライアントには読み取り専用許可とともにエクスポートするためには、**/etc/exports** ファイルに次のテキストを追加してください。

   ```
   /common/docs -sec=krb5,rw,sec=sys,ro
   ```

   次に、このエクスポートを実行するために exportfs /common/docs を入力します。

9. ホスト infoserver 上の **/usr/info** ディレクトリーに対して **/usr/info** で委託を作成するには、次の行を **/etc/exports** に追加してから、**/usr/info** をエクスポートします。

   ```
   /usr/info -vers=4,refer=/usr/info@infoserver
   ```

10. ホスト backup1 および backup2 の **/common/info** ディレクトリー用のレプリカを指定するには、次の行を **/etc/exports** に追加してから、**/common/info** をエクスポートします。

    ```
    /common/info -vers=4,replicas=/common/info@backup1:/common/info@backup2,<other options>
    ```

11. **/common/docs** ディレクトリーをバージョン 3 とバージョン 4 の両方でエクスポートするには、次のコマンドを入力します。

    ```
    exportfs -V 3:4 /common/docs
    ```

12. **/etc/exports** ファイルにあるバージョン 4 のエントリーをすべてエクスポートするには、次のコマンドを入力します。

    ```
    exportfs -a -V 4
    ```

13. **/common/docs** ディレクトリーをバージョン 3 に対してのみアンエクスポートするには、次のコマンドを入力します。

```
exportfs -u -V 3 /common/docs
```

14. **/etc/xtab** ファイルにあるバージョン 3 のエントリーをすべてアンエクスポートするには、次のコマンドを入力します。

```
exportfs -ua -V 3
```

15. s1、s2、および s3 という名前のホストで **/common/docs** ディレクトリーの委託を指定し、それらをすべて分散するには、次の行を **/etc/exports** ファイルに追加して、**/common/docs** ディレクトリーをエクスポートします。

```
/common/docs -vers=4,refer=/common/docs@s1:/common/docs@s2:/common/docs@s3,scatter=full
```

16. s1、s2、s3、および s4 という名前のホストで **/common/docs** ディレクトリーのレプリカを指定し、それらの一部を分散するには (最初のフェイルオーバー・サーバーはすべての組み合わせで s1 です)、次の行を **/etc/exports** ファイルに追加して、**/common/docs** ディレクトリーをエクスポートします。

```
/common/docs -vers=4,noauto,replicas=/common/docs@s1:/common/docs@s2:/common/docs@s3:/common/docs@s4,scatter=partial
```

ファイル

項目	説明
/etc/exports	サーバーがエクスポートできるディレクトリーのリストが入っています。
/etc/xtab	現在エクスポートされているディレクトリーのリストが入っています。
/etc/hosts	ネットワーク上の各ホストのエントリーが入っています。
/etc/netgroup	ネットワーク上の各ユーザー・グループについての情報が入っています。
/etc/rc.nfs	NFS デーモンと NIS デーモンの起動スクリプトが入っています。

関連情報:

chnfsexp コマンド

mknfsexp コマンド

セキュア NFS を使用したファイルシステムのエクスポート

NFS コマンドのリスト

---

exportvg コマンド

目的

ボリューム・グループの定義を物理ボリュームのセットからエクスポートします。

構文

exportvg *VolumeGroup*

説明

exportvg コマンドは、*VolumeGroup* パラメーターで指定されたボリューム・グループの定義をシステムから除去します。ボリューム・グループとその内容に関するシステム上の情報がすべて除去されるので、エクスポートされたボリューム・グループにアクセスすることができなくなります。 **exportvg** コマンドは、ボリューム・グループのいかなるユーザー・データも変更することはありません。

ボリューム・グループは、システム内部の非共用リソースです。したがって、現行プロセッサーから明示的にエクスポートされて別のプロセッサーにインポートされるまでは、ボリューム・グループ別のプロセッサーからアクセスしないでください。 **exportvg** コマンドの第 1 の用途は、**importvg** コマンドと併用し

て、移送可能なボリュームをプロセッサー間で交換できるようにすることです。エクスポートできるのはボリューム・グループ全体だけで、個々の物理ボリュームはエクスポートできません。

exportvg コマンドと **importvg** コマンドを使用すると、 2 つのプロセッサー間で共有される物理ボリューム上のデータの所有権を切り替えることができます。

注**:** このコマンドを使用するには、root 権限を持つか、**system** グループのメンバーでなければなりません。

System Management Interface Tool (SMIT) の **smit exportvg** 高速パスを使用して、このコマンドを実行できます。

注**:**

1. ページング・スペース・ボリュームを持つボリューム・グループは、ページング・スペースがアクティブの状態のときはエクスポートできません。アクティブなページング・スペース・ボリュームを使ってボリューム・グループをエクスポートする前に、ページング・スペースがシステムの初期化時に自動的にアクティブになっていないことを確認してから、システムをリブートします。
2. 論理ボリュームのマウント・ポイント情報が 128 文字を超える場合、LVCB (論理ボリューム制御ブロック) から脱落してしまいます。 128 文字を超える長さのマウント・ポイントについてはメモをとっておいてください。**importvg** コマンドを実行してこのボリューム・グループを完全にインポートするために、**/etc/filesystems** ファイルを手動で編集する必要があるためです。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

ボリューム・グループ vg02 をシステムから除去するには、次のように入力します。

```
exportvg vg02
```

注**:** ボリューム・グループは、エクスポートする前にオフに設定変更する必要があります。

vg02 の定義がシステムから除去され、ボリューム・グループにアクセスすることはできません。

ファイル

項目	説明
/usr/sbin	**exportvg** コマンドが入っているディレクトリー。

関連情報:

importvg コマンド

varyonvg コマンド

論理ボリューム・ストレージ

System Management Interface Tool

---

expr コマンド

目的

引数を式として評価します。

構文

expr *Expression*

説明

expr コマンドは、*Expression* パラメーターを読み取り、それを評価して、結果を標準出力に書き出します。

Expression パラメーターに対しては、次の規則が適用されます。

- 各項目はブランクで区切ります。
- シェル専用の文字の前には、 ¥ (円記号) を付けます。
- ブランクやその他の特殊文字を含む文字列は、引用符で囲みます。

整数の前には、単項のハイフンを付けることができます。内部では、整数は 32 ビットの 2 の補数として処理されます。

注**: expr** コマンドは、 null 列ではなく、ゼロの値を示す 0 を戻します。

次の項目は、*Expression* パラメーターの演算子およびキーワードを記述しています。エスケープする必要のある文字の前には、¥ (円記号) が付いています。項目のリストは、優先順位が高い順に並んでおり、同じ優先順位の演算子は {} (中括弧) で囲んでグループ化されています。

項目	説明
Expression1 **¥|** *Expression2*	null でも 0 でもない場合は、*Expression1* を戻します。それ以外の場合は、*Expression2* を戻します。
Expression1 **¥&** *Expression2*	Expression1 と Expression2 の値が null でも 0 でもない場合は、*Expression1* を戻します。それ以外の場合には値 0 を戻します。
Expression1 **{ =, ¥>, ¥>=, ¥<, ¥<=, != }** *Expression2*	両方の式が整数の場合、整数の比較結果を戻します。その他の場合は、文字列の比較結果を戻します。
Expression1 **{+, - }** *Expression2*	整数値の引数を加算または減算します。
Expression1 **{ ¥*, /, % }** *Expression2*	乗算、除算を行うか、または整数値の引数の除算の剰余を算出します。

項目	説明
Expression1 **:** *Expression2*	*Expression1* の計算結果を *Expression2* の計算結果から得られた正規表現パターンと比較します。正規表現の構文は **ed** コマンドの場合と同じですが、すべてのパターンは文字列の先頭に置かれていなければなりません (つまり、文字列の先頭文字から始まるシーケンスのみが、正規表現によってマッチングされます)。したがって、^ (脱字記号) は、このコンテキストでは特殊文字ではありません。 一般に、照合演算子は、一致した文字の数を戻します (突き合わせが失敗すると 0)。パターンに次のサブ式 `¥( Expression ¥)` が含まれている場合は、実際に一致した文字が入っている文字列が戻されます。 照合シーケンスでは、文字範囲で使用する等価クラスを定義できます。照合シーケンスと等価クラスについて詳しくは、「ナショナル・ランゲージ・サポート ガイドおよびリファレンス」の『ロケール環境変数について』のセクションを参照してください。

注**:** 次の引数の文字列は、標準の引数を超えた拡張機能です。その動作はオペレーティング・システムによって異なる場合があります。これらの引数の文字列は移植可能ではありません。

項目	説明
match *String1 String2*	*Expression1* **:** *Expression2* と同じです。
length *String1*	*String1* の長さを戻します。
index *String1 String2*	*String2* 内のいずれかの文字が存在する *String1* 内の最初の桁を戻します。
substr *String1 StartPosition Length*	*String1* 内の *StartPosition* の桁にある文字で始まり、*Length* の文字数まで続く文字列を戻します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	*Expression* パラメーターは、null でも 0 でもありません。
1	*Expression* パラメーターは、null か 0 です。
2	*Expression* パラメーターは無効です。
>2	エラーが発生しました。

注**:** シェルがパラメーターを処理した後は、**expr** コマンドは値による場合を除いて演算子とオペランドを区別することができません。したがって、`$a` の値が `j` の場合、コマンド

```
expr $a = j
```

は、シェルがこの引数を expr コマンドを渡した後、

```
expr j = j
```

のようになります。また、次のコマンドも真になります。

```
expr X$a = Xj
```

例

1. シェル変数を変更するには、次のように入力します。

   ```
   COUNT=`expr $COUNT + 1`
   ```

   これによって、シェル変数 $COUNT に 1 が追加されます。**expr** コマンドがバッククォートで囲まれることにより、**expr** コマンドからの標準出力は、シェルによって COUNT= コマンドに置換されます。$COUNT 変数は使用前に初期化されなければなりません。

2. シェル変数 **$STR** の長さを知るには、次のように入力します。

   ```
   LENGTH=`expr $STR : ".*"`
   ```

   これによって、LENGTH 変数が: (コロン) 演算子で指定された値に設定されます。パターン .* (ドット、アスタリスク) は初めから終わりまでどの文字列とも一致します。したがって、コロン演算子は一致した文字の数として $STR 変数の長さを与えます。シェルが .* (ドット、アスタリスク) をパターン・マッチング文字として処理しないようにするために、 .* を引用符で囲まなければならないことに注意してください。引用符はパターンの一部ではありません。

   $STR 変数が null 文字列に設定されるか、ホワイト・スペース (ブランクまたはタブ) を含んでいると、コマンドは、「expr: syntax error」というエラー・メッセージを表示します。これは、シェルが、通常、null 文字列をコマンドに渡さないことが原因で発生します。この場合、**expr** コマンドでは、次の内容しか認識できません。

   ```
   :.*
   ```

   シェルは単一引用符も除去します。コロン演算子は 2 つの値を必要とするので、これは機能しません。この問題は、シェル変数を次のように二重引用符で囲めば解決できます。

   ```
   LENGTH=`expr "$STR" : ".*"`
   ```

   これで、$STR 変数の値が null の場合、LENGTH 変数の値は 0 に設定されます。一般的には、シェル変数を二重引用符で囲むことをお勧めします。シェル変数を単一引用符で囲まないように注意してください。

3. 文字列の一部を使用するには、次のように入力します。

   ```
   FLAG=`expr "$FLAG" : "-*¥(.*¥)"`
   ```

   これによって、先行ハイフンがシェル変数 $FLAG から除去されます。コロン演算子によっては ¥(と ¥) (円記号と左小括弧、および円記号と右小括弧) の文字で囲まれたサブ式と一致する、FLAG 変数の一部が得られます。¥(と ¥) のサブ式の文字を省略すると、コロン演算子は一致した文字の数を与えます。

   $FLAG 変数が - (ハイフン) に設定されている場合、コマンドは、構文エラー・メッセージを表示します。これは、**expr** コマンドを実行する前に、シェルが $FLAG 変数の値を置き換えるために発生します。**expr** コマンドはハイフンが変数の値であることを認識しません。 expr コマンドには次のようにしか見えません。

   ```
   - : -*¥(.*¥)
   ```

   そして、最初のハイフンを減算の演算子と解釈します。この問題を解決するには、次のように入力します。

   ```
   FLAG=`expr "x$FLAG" : "x-*¥(.*¥)"`
   ```

4. **if** ステートメントで、 **expr** コマンドを使用する場合は、次のように入力します。

```
if expr "$ANSWER" : "[yY]" >/dev/null
then
echo ANSWER begins with "y" or "Y"
fi
```

$ANSWER 変数が y、または Y で始まる場合は、 **if** ステートメントの then 部分が実行されます。マッチングが正常に行われると、式の結果は 1 になり、**expr** コマンドは終了値 0 を戻します。**if** ステートメントはこれを論理値が真であると認識します。マッチ・ファイルが失敗すると、結果は 0 となり、終了値 1 (偽) が戻されます。

expr コマンドの標準出力を **/dev/null** スペシャル・ファイルにリダイレクトすると、式の結果は破棄されます。リダイレクトしなければ、結果は標準出力に書き出されます。通常、標準出力はワークステーションのディスプレイです。

5. 次の式について考えてみましょう。

   ```
   expr "$STR" = "="
   ```

   $STR 変数に = (等号記号) が含まれている場合、シェルがこのコマンドを処理した後、**expr** コマンドは式を次のように認識します。

   ```
   = = =
   ```

   expr コマンドは、これを行内の 3 つの = 演算子と解釈して、構文エラー・メッセージを表示します。シェル変数の値が **expr** 演算子の 1 つと同じであると、このメッセージが必ず表示されます。この問題は、次のように式を記述すれば解決できます。

   ```
   expr "x$STR" = "x="
   ```

6. $SHELL 環境変数 /usr/bin/ksh の長さを戻すには、次のように入力します。

   ```
   expr length $SHELL
   ```

   次のように表示されます。

   ```
   12
   ```

7. "abcdef" の中の文字列 "de" のどちらか最初にある文字の桁を戻すには、次のように入力します。

   ```
   expr index abcdef de
   ```

   次のように表示されます。

   ```
   4
   ```

8. "abcdef" の中の文字列 "fd" のどちらか最初にある文字の桁を戻すには、次のように入力します。

   ```
   expr index abcdef fd
   ```

   次のように表示されます。

   ```
   4
   ```

9. 文字列 "Goodnight Ladies" の中で、11 桁目から始まり長さが 6 の文字列を戻すには、次のように入力します。

   ```
   expr substr "Goodnight Ladies" 11 6
   ```

   次のように表示されます。

   ```
   Ladies
   ```

ファイル

項目	説明
/usr/bin/expr	**expr** コマンドが入っています。

関連資料:

299 ページの『ed または red コマンド』

関連情報:

bsh コマンド

csh コマンド

ナショナル・ランゲージ・サポートの概要

---

exptun コマンド

目的

トンネル定義、および任意選択で、そのトンネルに関連付けられたすべてのユーザー定義フィルター規則をエクスポートします。トンネル・エクスポート・ファイルおよび、トンネル・パートナーに使用できる、任意選択のフィルター規則のエクスポート・ファイルを作成します。

構文

exptun [-v 4|6] -f *directory* **[-t** *tid_list***] [-r] [-l manual]**

説明

exptun コマンドは、トンネル・コンテキスト・エクスポート・ファイル、および任意選択で、トンネル・パートナーがインポートがするフィルター規則付加ファイルを作成するために使用します。このコマンドはトンネルを起動するわけではなく、単にトンネル・パートナーの必須ファイルを作成するだけです。

注**:** 生成されたエクスポート・ファイルには、トンネルで使用したキーが入っています。オペレーティング・システムのファイルシステム記憶保護機構を使用して、これらのファイルを保護します。

フラグ

項目	説明
-f	エクスポート・ファイルを書き込むディレクトリーを定義します。ディレクトリーがない場合は、作成されます。これで、エクスポート・ファイルは、インポートされるトンネル・パートナーに送ることができます。各トンネル・パートナーのエクスポート・ファイルには、異なるディレクトリーを指定することをお勧めします。
-l	エクスポートしたいトンネルのタイプ。手動が指定されると、手動 IBM トンネルのみがエクスポートされます。
-r	トンネル (1 または複数) に関連するすべてのユーザー定義フィルター規則をエクスポートします。このフラグが使用されない場合は、トンネル定義のみがエクスポートされます。
-t	エクスポート・ファイルに使用されるトンネル ID のリストを指定します。リストは、「,」または「**-**」(1、3、10、50 から 55) で分離した一連のトンネル ID として指定することができます。このフラグが使用されない場合は、トンネル・データベースのすべてのトンネル定義がエクスポートされます。
-v	エクスポートされるトンネルの IP バージョン。値 **4** は IP バージョン 4 トンネルを指定します。値 **6** は IP バージョン 6 トンネルを指定します。このフラグが使用されない場合は、IP バージョン 4 と IP バージョン 6 の両方のトンネル定義がエクスポートされます。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティ

ー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

関連資料:

689 ページの『gentun コマンド』

関連情報:

chtun コマンド

lstun コマンド

mktun コマンド

---

extendlv コマンド

目的

ボリューム・グループ内から割り当て解除されている物理区画を追加することによって、論理ボリュームのサイズを拡大します。

構文

使用可能な物理区画を追加する

extendlv [**-a** *Position*] [**-e** *Range*] [**-u** *Upperbound*] [**-s** *Strict*] *LogicalVolume Partitions* [*PhysicalVolume ...*]

特定の物理区画を追加する

extendlv [**-m***MapFile*] *LogicalVolume Partitions*

説明

extendlv コマンドは、*Partitions* パラメーターで表される数の論理区画を新たに割り当てることによって、*LogicalVolume* に割り当てられる論理区画の数を増やします。 *LogicalVolume* パラメーターでは、論理ボリューム名または論理ボリューム ID を使用できます。特定の物理ボリュームへの割り当てを制限するには、*PhysicalVolume* パラメーターで 1 つ以上の物理ボリュームの名前を指定します。制限を行わないと、ボリューム・グループのすべての物理ボリュームを、新しい物理区画の割り当てに使用できます。

デフォルトでは、論理ボリュームは lslv コマンドを使用する際表示される既存の特性によって拡張されます。新しい区画に対してのみ、これらの既存の特性を一時的に指定変更するには、フラグを使用して、これらの特性に別の値を指定します。

デフォルトでは、論理ボリュームの区画の最大数は 512 です。論理ボリュームを 512 を超える論理区画に拡張するには、その前に **chlv** コマンドを使用してデフォルト値を大きくしてください。

デフォルトの割り当てポリシーは、1 つの論理ボリューム・コピーに最小数の物理ボリュームを使用し、コピーに属している物理区画を、できるだけ連続的に設定し、 **-a** フラグで指定された適切な領域に、物理区画を設定することです。また、デフォルトでは、論理区画の個々のコピーは、別々の物理ボリュームに配置されます。

extendlv コマンドの使用時には、512 ブロック/KB/MB/GB 単位で論理ボリューム・サイズを指定できます。(480 ページの『例』を参照してください。)

注**:**

1. ストライプ化論理ボリュームを拡張する場合は、区画数をストライプ幅の偶数倍にする必要があります。
2. 多数の区画 (800MB を超える場合) を使用する論理ボリュームは、段階的に拡張するようにしてください。
3. 論理ボリュームに加えた変更は、ファイルシステムには反映されません。ファイルシステムの特性を変更するには、**chfs** コマンドを使用します。
4. このコマンドを使用するには、root ユーザー権限を持っているか、あるいは **system** グループのメンバーでなければなりません。
5. **extendlv** コマンドは、スナップショット・ボリューム・グループには使用できません。

System Management Interface Tool (SMIT) の **smit extendlv** 高速パスを使用して、このコマンドを実行できます。

フラグ

注**: -e** および **-s** の各フラグは、ストライプ化論理ボリュームでは無効です。

項目	説明
-a *Position*	物理ボリューム間の割り当てポリシー (物理ボリューム上の論理区画の位置) を設定します。 *Position* 変数は、次のいずれかです。 **m** 各物理ボリュームの外部中央部分に論理区画を割り当てます。これがデフォルトの位置です。 **c** 各物理ボリュームの中央部分に論理区画を割り当てます。 **e** 各物理ボリュームの外側の端の部分に論理区画を割り当てます。 **ie** 各物理ボリュームの内側の端の部分に論理区画を割り当てます。 **im** 各物理ボリュームの内部中央部分に論理区画を割り当てます。
-e *Range*	物理ボリューム間の割り当てポリシー (最良の割り当てができるボリュームを使用して、物理ボリュームの数を、物理ボリューム間に渡って拡張する) を設定します。 *Range* 変数の値は、 *Upperbound* 変数 (-u フラグで設定) によって制限され、次のいずれかを指定できます。 **x** 物理ボリュームの最大数に論理区画を割り当てます。 **m** 物理ボリュームの最小数に論理区画を割り当てます。
-m *MapFile*	割り当てる物理ボリューム区画を正確に指定します。区画は、*MapFile* パラメーターで指定されたファイルに示されている順序で使用します。コピーに属するすべての物理区画が割り当てられてから、次のコピーに割り当てられます。 *MapFile* のフォーマットは次のとおりです。 **PVname:PPnum1[-PPnum2]** この場合、*PVname* は物理ボリュームの名前 (`hdisk0` など) です。この名前には、物理区画または連続する物理区画の 1 範囲につき 1 レコードが与えられます。 **PVname** システムで指定された物理ボリュームの名前。 **PPnum** 物理区画番号。 **重要:** マップ・ファイルを使用する場合は、すべての LV 割り当てパラメーター (厳密さ、上限、ストライプ幅など) を理解してそれらに準拠しなければなりません。マップ・ファイルを使用すると、LVM 割り当てルーチンで行われるチェックがバイパスされます。これは、ストライプ幅に準拠している通常のストライプ済み割り当てパターンを持つストライプ済み LV では重要です。

項目	説明
-s *Strict*	厳密な割り当てポリシーを決定します。論理区画のコピーは、同一物理ボリュームを共用するように割り当てることも、共用しないように割り当てることもできます。 *Strict* 変数は、以下のいずれかによって表されます。 **y** 厳密な割り当てポリシーを設定します。したがって、論理区画のコピーは、同一物理ボリュームを共用しません。 **n** 厳密な割り当てポリシーを設定しません。したがって、論理区画のコピーは、同一物理ボリュームを共用できます。 **s** 非常に厳密な割り当てポリシーを設定するために、1 つのミラーに割り当てられた区画は別のミラーの区画と物理ボリュームを共有することができません。 **注:** 非常に厳密な論理ボリュームでないものを非常に厳密な論理ボリュームへ変更する場合は、物理ボリュームを指定するか、**-u** フラグを使用する必要があります。
-u *Upperbound*	新規割り当て用の物理ボリュームの最大数を設定します。 *Upperbound* 変数の値は、1 と物理ボリュームの合計数の間の数である必要があります。非常に厳密を使用する場合は、上限は各ミラー・コピーごとに許可された物理ボリュームの最大数を示します。ストライピングされた論理ボリュームを使用するときは、上限は *Stripe_width* の倍数である必要があります。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. `lv05` というディレクトリーによって表される論理ボリュームのサイズを、論理区画 3 つ分だけ増加させるには、次のように入力します。

   ```
   extendlv lv05 3
   ```

2. 最小サイズが 10MB である `lv05` という名前の論理ボリュームを要求するには、次のように入力します。

   ```
   extendlv lv05 10M #
   ```

   extendlv コマンドは、最小サイズ以上の論理ボリュームを作成するために必要な区画の数を判別します。

   次のように大文字と小文字の両方を使用できます。

   ```
   B/b       512 byte blocks
   K/k       KB
   M/m       MB
   G/g       GB
   ```

ファイル

項目	説明
/usr/sbin/	**extendlv** コマンドが入っているディレクトリー。

関連情報:

chfs コマンド

lslv コマンド

論理ボリューム・ストレージ

PowerHA SystemMirror の管理

---

extendvg コマンド

目的

物理ボリュームをボリューム・グループに追加します。

構文

extendvg [**-f**] [**-p** *mirrorpool*] *volumegroup physicalvolume ...*

説明

extendvg コマンドは、1 つ以上の *physicalvolume* を追加することで *volumegroup* のサイズを増やします。

物理ボリュームが既に別のボリューム・グループに存在していないことが確認されます。システムは、その物理ボリュームが追加しようとしているボリューム・グループに属していると判断すると、終了します。しかし、まだ追加されていないボリューム・グループであることをその記述領域で検出すると、システムはコマンドの実行を続行するかどうかの確認をユーザーに求めます。物理ボリューム内の以前の内容は失われます。したがって、この指定変更機能を使用する際は、十分に注意してください。

注**:** このコマンドを使用するには、root 権限を持つか、**system** グループのメンバーでなければなりません。

AIX 5.3 より前に作成されたボリューム・グループ、または AIX 5.3 で作成されたが **varyonvg -M** フラグを指定してオンに変更されたボリューム・グループの場合は、その物理ボリュームの最大転送サイズがボリューム・グループの論理トラック・グループ・サイズよりも小さいと、**extendvg** は失敗します。 AIX 5.3 で作成され、**varyonvg -M** フラグを指定せずにオンに変更されたボリューム・グループの場合、物理ボリュームの最大転送サイズがボリューム・グループの論理トラック・グループ・サイズよりも小さいと、**extendvg** はそのボリューム・グループの論理トラック・グループ・サイズを動的に小さくします。

注**: extendvg** コマンドは、スナップショット・ボリューム・グループでは使用できません。

System Management Interface Tool (SMIT) の **smit extendvg** 高速パスを使用して、このコマンドを実行できます。

注**:** このコマンドは、ディスクがサード・パーティーのボリューム・グループによって管理されていることを示す場合には、ボリューム・グループにディスクを追加できません。サード・パーティーのボリューム・マネージャーのディスクを上書きおよびクリアするには、**chpv -C** *HDiskName* を使用します。

注**:** 並行ボリューム・グループ (VG) を拡張する場合は、VG に追加される新しいディスクごとに物理ボリューム ID (PVID) が割り当てられていることを確認し、オブジェクト・データ・マネージャー (ODM) に

保管されている PVID がすべてのノードで同じものであることを確認する必要があります。VG の拡張に Cluster Single Point of Control (C-SPOC) ユーティリティーを使用すると、このチェックは自動的に行われます。

注**:** 既存の PV タイプの制約事項があるかどうかを判別するために VG が検査されます。そのような制約事項が存在する場合、制約事項に適合していることを確認するために、**extendvg** コマンド・ラインの物理ボリューム・リストが検査されます。1 つ以上のディスクが PV タイプの制約事項に適合していないことが検出された場合、コマンドは失敗します。

注**:** 4 KB ブロック・サイズの物理ボリューム (PV) を他のサイズの PV ブロックと混用することはできません。ボリューム・グループ内のすべての PV のブロック・サイズは同じでなければなりません。

フラグ

項目	説明
-f	物理ボリュームが、デバイス構成データベースの別のボリューム・グループ、またはアクティブなボリューム・グループのメンバーでない場合、その物理ボリュームを指定されたボリューム・グループに追加させます。
-p *mirrorpool*	追加される各物理ボリュームを指定のミラー・プールに割り当てます。ミラー・プールがボリューム・グループで使用可能になると、そのボリューム・グループはミラー・プールをサポートしない AIX のバージョンにはインポートされません。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

物理ボリューム `hdisk3` と `hdisk8` をボリューム・グループ `vg3` に追加するには、次のように入力します。

```
extendvg vg3 hdisk3 hdisk8
```

注**:** ボリューム・グループは、拡張する前にオンに設定変更する必要があります。

制限

extendvg コマンドは、スナップショット・ボリューム・グループでは実行できません。

ファイル

項目	説明
/usr/sbin/extendvg	**extendvg** コマンドが入っています。

関連情報:

reducevg コマンド

論理ボリューム・ストレージ

chpv コマンド

PowerHA SystemMirror の管理

f

以下の AIX コマンドは文字 f で始まります。

f コマンド

目的

ユーザー情報を表示します。このコマンドは **finger** コマンドと同じです。

構文

{ **f** | **finger** } [[**-b**] [**-h**] [**-l**] [**-p**]] | [**-i**] [**-q**] [**-s**] [**-w**]]

[**-f**] [**-m**] [*User* | *User* **@***Host* | **@***Host*]

説明

/usr/bin/f コマンドは、現在ホストにログインしているユーザーの情報を表示します。出力形式は、表示される情報の選択によって異なります。

デフォルトのフォーマット

デフォルトの形式には次の項目が入ります。

- ログイン名
- フル・ユーザー名
- 端末名
- 書き込み状況 (端末名の前の * (アスタリスク) は、書き込み許可が拒否されていることを表します)

ホスト上のユーザーの場合、デフォルト情報のリストには、次の項目も含まれます。

- アイドル時間 (アイドル時間は、単一の整数であれば分、 : (コロン) があれば時間と分、「d」があれば日付と時間です。)
- ログイン時刻
- サイト特定の情報

  サイト特定の情報は、 **/etc/passwd** ファイルの gecos フィールドから検索されます。gecos フィールドには、後ろにコンマが付いた完全なユーザー名が入っています。finger コマンドによって、コンマの後ろにすべての情報がサイト特定の情報と一緒に表示されます。

長形式

ユーザーの名前のリストが指定されると、 **f** コマンドは必ず、長形式を使用します。(ユーザーの姓名のほかに、アカウント名も使用できます。) このフォーマットは複数行で設定され、次に示す情報の他に上記の情報をすべて含みます。

- ユーザーの **$HOME** ディレクトリー
- ユーザーのログイン・シェル

- ユーザーの **$HOME** ディレクトリー内の、 **.plan** ファイルの内容
- ユーザーの **$HOME** ディレクトリー内の、 **.project** ファイルの内容

f コマンドはリモート・システム上のユーザーの検索にも使用できます。フォーマットは、ユーザーを *User@Host* として指定します。ユーザー名を省略すると、**f** コマンドは標準フォーマットのリストをリモート・システム上に表示します。

好みのテキスト・エディターを使って、 **.plan** ファイルと **.project** ファイルを作成し、それらのファイルを **$HOME** ディレクトリーに入れてください。 **f** コマンドは、**.plan** ファイルと **.project** ファイルの内容を表示するときに、**toascii** サブルーチンを使用して、通常の ASCII 文字範囲外の文字を変換します。**f** コマンドにより、変換された文字の前には `M-` が表示されます。

User パラメーターでユーザーを指定する場合は、ユーザーの姓、名、もしくはアカウント名を指定します。ユーザーを指定すると、**f** コマンドは指定したホストで、指定ユーザーの情報を長形式で戻します。

f コマンドのその他の情報については、「ネットワークおよびコミュニケーションの管理」の『TCP/IP のインストール』のセクションを参照してください。

フラグ

項目	説明
-b	簡潔な長形式をリストします。
-f	出力時にヘッダー行の印刷 (表示中のフィールドを定義する最初の行) を抑止します。
-h	**.project** ファイルを、長形式および簡潔な形の長形式では表示しません。
-i	アイドル時間と一緒に高速でリストします。
-l	長形式をリストします。
-m	*User* パラメーターは、ユーザーのログイン名ではなく、ユーザー ID (任意アクセス制御に使用される) であると見なされます。
-p	**.plan** ファイルを、長形式と簡潔な形の長形式では表示しません。
-q	高速でリストします。
-s	短形式をリストします。
-w	幅の狭い、短形式をリストします。

パラメーター

項目	説明
@Host	リモート・ホスト上のすべてのログイン・ユーザーを指定します。
User	**/etc/passwd** ファイル内で指定した、ローカル・ユーザー ID (任意アクセス制御に使用される) またはローカル・ログイン名を指定します。
User@Host	長形式で表示されたリモート・ホスト上のユーザー ID を指定します。

例

1. ホスト `alcatraz` にログインしている、すべてのユーザーの情報を得るには、次のように入力します。

   ```
   f @alcatraz
   ```

   次のような情報が表示されます。

   ```
   [alcatraz.austin.ibm.com]
   Login     Name         TTY Idle         When      Site Info
   brown     Bob Brown    console   2d   Mar 15 13:19
   smith     Susan Smith     pts0  11:   Mar 15 13:01
   jones     Joe Jones       tty0     3   Mar 15 13:01
   ```

   ユーザー `brown` は `console` でログインし、ユーザー `smith` は、疑似テレタイプ・ライン `pts0` からログインし、ユーザー `jones` は、 `tty0` からログインしています。

2. alcatraz のユーザー brown に関する情報を得るには、次のように入力します。

```
f brown@alcatraz
```

次のような情報が表示されます。

```
Login name: brown
Directory: /home/brown    Shell: /home/bin/xinit -L -n Startup
On since May 8 07:13:49 on console
No Plan.
```

3. ローカル・ホスト上で、ユーザー brown の情報を簡易書式で得るには、次のように入力します。

```
f -q brown
```

次のような情報が表示されます。

```
Login           TTY             When
brown           pts/6           Mon Dec 17 10:58
```

ファイル

項目	説明
/usr/bin/f	**f** コマンドが入っています。
/etc/utmp	現在ログインされているユーザーのリストが入っています。
/etc/passwd	ユーザー・アカウント、ユーザー名、ホーム・ディレクトリーを定義します。
/etc/security/passwd	ユーザーのパスワードを定義します。
/var/adm/lastlog	最後にログインした時刻が入っています。
$HOME/.plan	ユーザー・プランの 1 行単位の説明が入っているオプションのファイル。
$HOME/.project	ユーザー・プロジェクトの割り当てが入っているオプションのファイル。

関連資料:

801 ページの『hostname コマンド』

578 ページの『finger コマンド』

関連情報:

rwho コマンド

ログイン・ユーザーに関する情報を表示するコマンド

---

factor コマンド

目的

数を因数に分解します。

構文

factor [*Number*]

説明

factor コマンドは、 *Number* パラメーターに値を指定しないで呼び出すと、ユーザーが 1E14 (100,000,000,000,000) 未満の正の数字を入力するまで待機します。次に、その数字の素因数を標準出力に書き出します。適切な回数だけ個々の因数を表示します。終了するには、0 または数字以外の任意の文字を入力します。

factor コマンドは、引数付きで呼び出されると、*Number* パラメーターの素因数を判別し、結果を標準出力に書き出してから、終了します。

例

123 の素因数を計算するには、次のように入力します。

```
factor 123
```

次のように表示されます。

```
123
     3
     41
```

ファイル

項目	説明
/usr/bin/factor	**factor** コマンドが入っています。

関連情報:

bc コマンド

---

true または **false** コマンド

目的

ゼロの終了値 (真) を戻します。またはゼロ以外の終了値 (偽) を戻します。

構文

true

false

説明

true コマンドはゼロの終了値を戻します。 **false** コマンドはゼロ以外の終了値を戻します。この 2 つのコマンドは、多くの場合シェル・スクリプトの一部として使用します。

例

1 分ごとに一度ずつ日時を表示するループを作成するには、シェル・スクリプトで次のコードを使用します。

```
while true
do
   date
   sleep 60
done
```

---

reboot または **fastboot** コマンド

目的

システムをリブートします。

構文

{ **reboot** | **fastboot** } [**-l**] [**-n**] [**-q**] [**-t** *mmddHHMM* [*yy*]]

説明

ほかのユーザーがシステムにログインしていなければ、**reboot** コマンドを使用してリブート操作を行うことができます。**lsattr** コマンドを使用し、`lsattr -D -l sys0` と入力します。デフォルト値は **true** です。autorestart 属性値を **false** にリセットするには、**/var/adm/wtmp** ログイン・アカウント・ファイルを使用します。 **-l**、 **-n**、または **-q** フラグが指定されていれば、これらのアクションは禁止されます。

fastboot コマンドは、**reboot** コマンドを呼び出すことによってシステムをリブートさせます。**fsck** コマンドがシステムの始動時に実行され、ファイルシステムを検査します。このコマンドには BSD 互換性があります。

フラグ

項目 説明

-l リブートをログに記録すること、またはアカウント・ファイルにシャットダウン・レコードを書き込むこともしません。**-l** フラグは、アカウント・ファイルの更新を抑制しません。**-n** フラグおよび **-q** フラグは **-l** を暗黙指定します。

-n **sync** コマンドを実行しません。このフラグを使うと、ファイルシステムが損傷することがあります。

-q 最初に実行中のプロセスをシャットダウンせずにリブートします。
注: -q フラグを使用すると、ファイルシステムの同期化は起こりません。ファイルシステムを同期化させたい場合には、**sync** コマンドを手動で実行するか、または **shutdown -r** コマンドを実行します。

-t システムを直ちにシャットダウンし、その後、指定された日付にシステムを再始動します。有効な日付のフォーマットは以下のとおりです。

mmddHHMM [*yy*]

それぞれの意味は次のとおりです。

mm 月を指定します。

dd 日を指定します。

HH 時間を指定します。

MM 分を指定します。

yy 年を指定します (オプション)。 2 桁の数値は現行の世紀の年数の値 (システム時刻に基づく) を表しています。例えば、システム時刻に基づく現行年が 1985 年である場合、99 は 1999 年を意味し、現行年が 2005 年である場合、99 は 2099 年を、04 は 2004 年を意味します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意: このコマンドは特権操作を実行できます。特権命令を実行できるのは特権ユーザーのみです。権限および特権について詳しくは、「セキュリティー」の『特権コマンド・データベース』を参照してください。このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

リブートをログに記録せずにシステムをシャットダウンするには、以下のように入力します。

```
reboot  -l
```

ファイル

項目	説明
/etc/rc	システム始動スクリプトを指定します。
/var/adm/wtmp	ログイン・アカウント・ファイルを指定します。

fc コマンド

目的

コマンドのヒストリー・リストを処理します。

構文

エディターをオープンして以前に入力したコマンドを変更し、再実行する

fc [**-r**] [**-e** *Editor*] [*First* [*Last*]]

以前に入力したコマンドのリストを生成する

fc -l [**-n**] [**-r**] [*First* [*Last*]]

以前に入力したコマンドのリストを実行時刻付きで生成する

fc -t [**-n**] [**-r**] [*First* [*Last*]]

以前に入力したコマンドを再実行する

fc -s [*Old***=** *New*] [*First*]

説明

fc コマンドは、ユーザーのコマンド・ヒストリー・ファイルの内容を表示したり、エディターを起動して、シェルで以前に入力したコマンドを変更したり再実行したりします。

コマンド・ヒストリー・ファイルでは、コマンドが番号別に表示されます。リスト内の最初の番号は任意に選択されます。番号とそのコマンドとの関係は変化しませんが、ユーザーがログインするときに他のプロセスがそのリストにアクセスしていない場合は除きます。その場合、システムは番号付けをリセットして、コマンドを古い順に 1 から番号付けを始めます。

コマンド・ヒストリー・ファイルの番号が、**HISTSIZE** 環境変数の値と 32767 の、いずれか大きい方の値を超えて制限に達すると、シェルが 1 に折り返します。このオプションの番号折り返し機能に関係なく、**fc** コマンドはコマンドの時刻順シーケンスを維持します。例えば、3 つのコマンドの番号が順々に 32766、32767 および 1 (折り返し後の番号) となっている場合、コマンド 32767 は引き続きコマンド 1 の前にあるものと見なされます。

ヒストリー・ファイルの中にあるコマンドは、**-l** (小文字の L) フラグを使用すると表示できます。 **-l** フラグを指定せず、**-e** *Editor* フラグを使用してコマンドを編集すると、その結果生成された行はヒストリー・ファイルの終わりに入力されてから、シェルによって再度実行されます (**fc -e** *Editor* コマンドは、コマンド・ヒストリー・リストには入力されません)。エディターが 0 以外の終了状況を戻すと、ヒストリー・ファイル内の入力とコマンドの再実行が抑止されます。

fc コマンドで使用されるコマンド・ラインの変数代入またはリダイレクト演算子は、前のコマンドをもう一度起動し、**fc** コマンドと前のコマンドの両方の標準エラーを抑止します。例えば次のとおりです。

```
fc -s -- -1 2>/dev/null
```

フラグ

項目	説明
-e *Editor*	指定したエディターを使用してコマンドを編集します。*Editor* パラメーターはコマンド名です。コマンドを配置するには、**PATH** 環境変数を使用します。**-e** フラグを指定していない場合は、デフォルトとして **FCEDIT** 環境変数の値が使用されます。**FCEDIT** 環境変数が null、または設定されていない場合は、ed エディターが使用されます。
-l	(小文字の L) ヒストリー・ファイル内のコマンドを表示します。コマンドを変更するためのエディターは起動されません。各コマンドは、*First* および *Last* パラメーターで指定された順序で、**-r** フラグの影響を受け、各コマンドの前にコマンド番号が付いた状態で書き込まれます。
-n	**-l** フラグと一緒に使用すると、コマンド番号の表示を抑止します。
-r	コマンドのリスト順序 (**-l** フラグと一緒に使用した場合) を反転、またはコマンドの編集順序 (**-l** フラグを指定しない場合) を反転させます。
-s	エディターを起動せずにコマンドを再度実行します。 *First* パラメーターも指定していない場合は、**-s** フラグは直前のコマンドを再度実行します。
-t	コマンドを、それが実行された時刻とともにヒストリー・ファイルにリストします。この機能は **-l** フラグと似ていますが、コマンドの実行時刻が表示されます。 **注:** 時刻フィールドが以前 `EXTENDED_HISTORY=ON` を設定して記録されている場合は、フォーマット済みの時刻フィールドが表示されます。設定されていなければ「?」と表示されます。

パラメーター

項目	説明
First または *Last*	リストまたは編集するコマンドを選択します。アクセス可能な、以前のコマンドの数は **HISTSIZE** 環境変数の値によって決定されます。 *First* および *Last* パラメーターは、次のいずれかの値でなければなりません。 **[+]** *Number* 特定のコマンド番号を表します。コマンド番号は、**-l** フラグを使用して表示できます。+ (正符号) はデフォルトの値です。 **-***Number* 以前実行されたコマンドを表します。ヒストリー・リスト内でバックアップするコマンドの数により指定されます。例えば、-1 は直前のコマンドを示します。 *string* 指定した文字列で始まるコマンドのうち、最後に入力したコマンドを示します。 **-s** フラグを付けずに *Old*=*New* パラメーターを指定すると、*First* パラメーターからの文字列に **=** (等符号) を入れることはできません。 **-s** フラグを使用する場合に、*First* パラメーターを省略すると、直前のコマンドが使用されます。

-s フラグが指定されていない場合は、次の規則が適用されます。

- **-l** フラグを使用する場合に、 *Last* パラメーターを省略すると、デフォルトの設定は直前のコマンドになります。
- **-r**、**-n**、および **-e** フラグを使用する場合に、 *Last* パラメーターを省略すると、デフォルトの設定は *First* パラメーターになります。
- *First* パラメーターと、 *Last* パラメーターの両方を省略すると (**-l** フラグを使用するかどうかに応じて)、以前の 16 個のコマンドが表示されるか、直前のコマンドが 1 つ編集されます。
- *First* パラメーターと、 *Last* パラメーターの両方を指定すると、すべてのコマンドがリストされる (**-l** フラグを指定した場合) か、編集されます (**-l** フラグを指定しない場合)。複数のコマンドを編集するには、コマンドそれぞれを新しい行で始め、一度にすべてのコマンドをエディターに対して指定します。

First パラメーターが *Last* パラメーターよりも新しいコマンドを表す場合は、各コマンドが逆の順でリストまたは編集されます。これは、**-r** フラグを使用した場合と同じです。例えば、次の 1 行目のコマンドは、2 行目の対応するコマンドと同じです。

```
fc  -r  10  20          fc      30  40
fc      20  10          fc  -r  40  30
```

- コマンドの範囲を使用する場合は、ヒストリー・リスト内にない、 *First* または *Last* の値を指定しても、エラーではありません。**fc** コマンドは、リスト内の最も古いコマンドまたは最も新しいコマンドのうち、該当する値を代用します。例えば、ヒストリー・リスト内に 1 から 10 の番号付けられた 10 個のコマンドだけが入っている場合に、次のコマンドを入力したとします。

  ```
  fc  -l
  fc   1   99
  ```

  この場合、10 個のコマンドがすべて個別にリストされ、編集されます。

項目	説明
Old=*New*	再度実行されるコマンド内で、古い文字列の最初のオカレンスを新しい文字列に置換します。

環境変数

次の環境変数は、**fc** コマンドの実行に影響を与えます。

項目	説明
EXTENDED_HISTORY	ヒストリー・ファイルにおけるコマンド実行時刻の記録を制御するために使用されます。この変数を `ON` に設定すると、時刻が記録されます。それ以外の場合は、記録されません。
FCEDIT	シェルによって拡張されると、**-e** *editor* 変数のデフォルトの設定の値を決定します。 **FCEDIT** 環境変数の値が null、または設定されていない場合は、ed エディターがデフォルトになります。
HISTDATEFMT	**fc –t** コマンドが表示する時刻のフォーマットを制御するために使用されます。例えば、`HISTDATEFMT=%Y` とすれば、コマンドの実行時に **fc -t** は年を表示します。このフォーマット設定は、**date** コマンドで行われる設定と同様です。
HISTFILE	コマンド・ヒストリー・ファイルのパス名を決定します。 **HISTFILE** 環境変数が設定されていない場合、シェルはユーザーのホーム・ディレクトリー内の **.sh_history** ファイルにアクセスまたは作成する場合があります。
HISTSIZE	アクセス可能な以前のコマンド数の制限を表す 10 進数を決定します。この変数が設定されていない場合は、デフォルトの値である 128 が使用されます。

終了状況

次の終了値が戻されます。

項目	説明
0	リスト表示が正常終了しました。
>0	エラーが発生しました。

上記以外の終了状況は、**fc** コマンドで実行されたコマンドの終了状況です。

例

1. 最後に実行したコマンドで、 **FCEDIT** 環境変数により定義されたエディター (デフォルトのエディターは、 **/usr/bin/ed**) を起動するには、次のように入力します。

   ```
   fc
   ```

   このコマンドは、編集が終わると実行されます。
2. 以前に実行した 2 つのコマンドを表示するには、次のように入力します。

```
fc -l -2
```

3. cc で始まるコマンドを検索し、 foo を bar に変更し、コマンドを表示して実行するには、次のように入力します。

```
fc -s foo=bar cc
```

4. 以前実行されたコマンドを、その実行時刻とともにリストするには、次のように入力します。

```
fc –t
```

ファイル

項目	説明
/usr/bin/ksh	Korn シェルの **fc** 組み込みコマンドが入っています。
/usr/bin/fc	**fc** コマンドが入っています。

関連情報:

ksh コマンド

---

fccheck コマンド

目的

First Failure Data Capture (FFDC) ユーティリティーに関して、基本問題判別を実行します。

構文

/opt/rsct/bin/fccheck [**-q**] | [**-h**]

説明

fccheck は、First Failure Data Capture (FFDC) ユーティリティーに関して、基本問題判別を実行します。コマンドは、ローカル・ノード上で以下の状態および情報を検査します。

- 現行の処理環境で、FFDC エラー・スタックの使用が使用不可になっているかどうかを検査する。
- FFDC がローカル・ノードを識別する際に現在使用する IP アドレスを入手する。
- **/var/adm/ffdc/stacks** が使用可能かどうかを検査する。また、使用可能の場合は、このディレクトリーが存在するファイルシステムで、どれだけのスペースが使用可能であるかを検査する。FFDC エラー・スタックを作成するスペース量が不十分でないかどうかを検査します。
- **/var/adm/ffdc/dumps** が使用可能かどうかを検査する。また、使用可能の場合は、このディレクトリーが存在するファイルシステムで、どれだけのスペースが使用可能であるかを検査する。

「抑制」オプションを指定しない限り、上記のテストの結果は標準出力に表示されます。 **fccheck** は、テストの実行中に検出した最も重大な状態を示す終了状況値を設定します。

フラグ

-h 標準出力にヘルプおよび使用方法に関する情報を表示します。その他の処理は実行されません。

-q 「抑制」モードを指定します。コマンドは各テストの結果を標準出力に表示しません。テストの結果を判別するために、コマンドの終了状況を使用する必要があります。 複数の状態が検出された場合は、**fccheck** が検出した最も重大な状態が終了状況に反映されます。

終了状況

このコマンドでは、以下の整数の終了状況コードが生成されます。

0 **fccheck** がテストしたすべての状態が、正常稼働パラメーターでした。

2 ヘルプ情報が正常に表示されました。これ以上の処理は実行されません。

12 検査が行われませんでした。このコマンドに無効なオプションが指定されました。

19 ディレクトリー **/var/adm/ffdc/stacks** がマウントされていないか、存在しません。

20 パス **/var/adm/ffdc/stacks** にある 1 つ以上のディレクトリーをアクセスできないか、検査できません。アクセスを防ぐために、このパスにある 1 つ以上のディレクトリーについて、権限が変更された可能性があります。

24 パス **/var/adm/ffdc/dumps** にある 1 つ以上のディレクトリーをアクセスできないか、検査できません。アクセスを防ぐために、このパスにある 1 つ以上のディレクトリーについて、権限が変更された可能性があります。

32 ディレクトリー **/var/adm/ffdc/dumps** がマウントされていないか、存在しません。

40 ローカル・ノードで FFDC エラー・スタックを作成するための十分なスペースが、**/var/adm/ffdc/stacks** ディレクトリーにありません。

41 オペレーティング・システムからファイルシステム情報を入手できません。オペレーティング・システム自体に問題がある可能性があります。

42 この処理環境で、FFDC エラー・スタックの作成および使用が使用不可になっています。

例

ローカル・ノードでの FFDC ユーティリティーの問題を検査するには、次のようにします。

```
fccheck
fccheck Status: All tests completed
```

ローカル・ノードで FFDC エラー・スタックの作成が使用不可になっている場合は、 **fccheck** は、この問題を次のように表示します。

```
fccheck

fccheck Status: Creation and use of FFDC Error Stacks has been expressly
disabled in the current execution environment. Any processes created in
the current execution environment cannot create their own FFDC Error Stacks
or inherit use of existing FFDC Error Stacks.

fccheck Status: All checks completed. Examine the previous status output for
possible FFDC problem conditions and take the recommended actions listed in
these messages.
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

495 ページの『fcclear コマンド』

502 ページの『fcinit コマンド』

fcclear コマンド

目的

ローカル・ノードから FFDC エラー・スタックのファイルおよび詳細データ・ファイルを除去します。

構文

/opt/rsct/bin/fcclear -h | [**-d** *filename* [*,filename,...*]] [**-D** *filename* [*,filename,...*]] [**-f** *FFDC_Failure_ID* [*,FFDC_Failure_ID,...*]] [**-F** *FFDC_Failure_ID* [*,FFDC_Failure_ID,...*]] [**-s** *file_name*[*,filename,...*]] [**-S** *file_name* [*,filename,...*]] [**-t** *days*]]

説明

fcclear は、問題判別用として必要でなくなった FFDC エラー・スタック・ファイルをローカル・ノードから除去する際に使用します。特定の FFDC エラー・スタック・ファイル、および特定の FFDC 障害 ID のレコードを含む FFDC エラー・スタック・ファイルを除去することができます。FFDC エラー・スタックの個々のエントリーを除去することはできません。

-t オプションを使用すると、**fcclear** は、特定の日数より前の FFDC エラー・スタック・ファイルを除去する際に使用できます。 **fcclear** を自動的に実行して、不必要な FFDC エラー・スタックを消去するには、コマンドの実行を自動化するための **cron** コマンドを参照してください。

ローカル・ノードからすべての FFDC エラー・スタックを除去するには、日数オプションの引数としてゼロ (0) の値を指定します。

フラグ

-d 1 つ以上の詳細データ・ファイル名のリストを指定して、詳細データ・ファイルを除去します。これらのファイル名は、絶対パス名または **/var/adm/ffdc/dumps** ディレクトリーに対する相対パス名のどちらも使用することができます。これらのファイルがローカル・ノードに存在すれば、除去されます。リモート・ノードのファイルをこのコマンドを使用して除去することはできません。複数のファイル名を指定する場合は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

-D 1 つ以上の詳細データ・ファイル名のリストを指定して、詳細データ・ファイルを保持します。これらのファイル名は、絶対パス名または **/var/adm/ffdc/dumps** ディレクトリーに対する相対パス名のどちらも使用することができます。これらのファイルがローカル・ノードに存在すれば、保持されます。リモート・ノードのファイルをこのコマンドを使用して保持することはできません。複数のファイル名を指定する場合は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

-f 1 つ以上の FFDC 障害 ID のリストを指定して、FFDC エラー・スタック・ファイルを除去します。これらの FFDC エラー ID に関連した FFDC エラー・スタックがローカル・ノードに存在すれば、検出され、除去されます。リモート・ノードの FFDC エラー・スタックは除去されません。複数の FFDC 障害 ID を指定する場合は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

-F 1 つ以上の FFDC 障害 ID のリストを指定して、FFDC エラー・スタック・ファイルを保持します。これらの FFDC エラー ID に関連した FFDC エラー・スタックがローカル・ノードに存在すれば、検出され、保持されます。リモート・ノードの FFDC エラー・スタックは保持されません。複数の FFDC 障害 ID を指定する場合は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

-h 標準出力デバイスにヘルプおよび使用方法に関する情報を表示します。その他の処理は実行されません。

-s 1 つ以上の FFDC エラー・スタック・ファイル名のリストを指定して、 FFDC エラー・スタック・ファイルを除去します。これらのファイル名は、絶対パス名または **/var/adm/ffdc/stacks** ディレクトリーに対する相対パス名のどちらも使用できます。これらのファイルがローカル・ノードに存在すれば、除去されます。リモート・ノードの FFDC エラー・スタックをこのコマンドを使用して除去することはできません。複数のファイル名を指定する場合は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

-S 1 つ以上の FFDC エラー・スタック・ファイル名のリストを指定して、 FFDC エラー・スタック・ファイルを除去します。これらのファイル名は、絶対パス名または **/var/adm/ffdc/stacks** ディレクトリーに対する相対パス名のどちらも使用できます。これらのファイルがローカル・ノードに存在すれば、除去されます。リモート・ノードの FFDC エラー・スタックをこのコマンドを使用して除去することはできません。複数のファイル名を指定する場合は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

-t ローカル・ノードから、特定の日数より前の FFDC のエラー・スタック・ファイルおよび詳細データ・ファイルを除去するように指示します。この選択基準は、他の選択基準からは独立したものです。

終了状況

fcclear は、完了すると以下の終了状況を生成します。

0 コマンドが正常終了しました。FFDC のエラー・スタック・ファイルまたは詳細データ・ファイルが選択基準に一致しない場合は、コマンドが正常に終了します。

2 ヘルプ情報が正常に表示されました。これ以上の処理は実行されません。

10 ローカル・システムからファイルが削除されませんでした。このコマンドに必須オプションが指定されませんでした。

11 ローカル・システムからファイルが削除されませんでした。**-t** オプションの引数が数字ではありません。

12 ローカル・システムからファイルが削除されませんでした。呼び出し側から不明なオプションが指定されました。

19 ディレクトリー **/var/adm/ffdc/stacks** が存在しないか、マウントされていません。

26 ローカル・システムからファイルが削除されませんでした。同じオプションが複数回指定されました。

28 システムからファイルが削除されませんでした。呼び出し側が、コマンドに対して、同じファイルの除去および保持の両方を行うよう指示するオプションを指定しました。この状態が発生するのは、コマンド・ユーザーが、名前で指定された FFDC エラー・スタック・ファイルに記録されている FFDC 障害 ID をコマンドで指定した場合です。

例

ローカル・ノードから 7 日より前のすべての FFDC のエラー・スタック・ファイルおよび詳細データ・ファイルを除去するには、次のようにします。

```
fcclear -t 7
```

FFDC 障害 ID **/3Iv04ZVVfvp.wtY0xRXQ7.....** に関する情報を含む FFDC エラー・スタックは保持するが、7 日より前のそれ以外のすべての FFDC エラー・スタック・ファイルおよび詳細データ・ファイルを除去するには、次のコマンドを実行します。

```
fcclear -t 7 -F /3Iv04ZVVfvp.wtY0xRXQ7...................
```

FFDC 障害 ID **/3Iv04ZVVfvp.wtY0xRXQ7.....** に関するレコードを含む FFDC エラー・スタック・ファイルを除去するには、次のコマンドを実行します。

```
fcclear -f /3Iv04ZVVfvp.wtY0xRXQ7...................
```

システムから、FFDC エラー・スタック・ファイル **myprog.14528.19990204134809** および **a.out.5134.19990130093256**、および詳細データ・ファイル **myprog.14528.19990204135227** を除去するには、次のようにします。

```
fcclear -s myprog.14528.19990204134809,a.out.5134.19990130093256
    -d myprog.14528.19990204135227
```

上記のコマンドを拡張して、指定したファイルおよび 14 日より前のすべての FFDC のエラー・スタック・ファイルおよび詳細データ・ファイルを除去するには、次のようにします。

```
fcclear -s myprog.14528.19990204134809,a.out.5134.19990130093256
    -d myprog.14528.19990204135227 -t 14
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

493 ページの『fccheck コマンド』

520 ページの『fcreport コマンド』

525 ページの『fcstkrpt コマンド』

fcdecode コマンド

目的

First Failure Data Capture (FFDC) 障害 ID を標準フォームからコンポーネント部分に変換し、この情報を人間が読める形式で標準出力デバイスに表示します。

構文

/opt/rsct/bin/fcdecode *FFDC_Failure_ID* [*,FFDC_Failure_ID,...*] | **-h**

説明

fcdecode は、42 文字の FFDC 障害 ID をコンポーネント部分にデコードし、これらの部分を人間が読める形式で表示します。このコマンドの出力では、 FFDC 障害 ID から抽出された以下の情報が表示されます。

- このレポートが存在するノードのネットワーク・アドレス (ASCII フォーマット)
- この記録が作成された時刻 (現在アクティブなタイムゾーン設定を使用して表現)
- 情報が記録されている場所に対応した、以下のどれかの情報
- この記録を作成するのに使用された AIX エラー・ログ・テンプレート ID (記録がそのノードの AIX エラー・ログにファイルされている場合)、または
- この記録を含む FFDC エラー・スタック・ファイルの名前 (記録が FFDC エラー・スタックにファイルされており、FFDC エラー・スタックがこのノードに存在する場合)
- FFDC 障害 ID に関連した特別なレポートを入手する際に使用できる推奨コマンド

フラグ

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

パラメーター

FFDC_Failure_ID

FFDC 障害 ID。**fcpushstk** コマンドおよび **fclogerr** コマンドに対する以前の呼び出しによって戻されたもの、あるいは、**fc_push_stack** サブルーチンまたは **fc_log_error** サブルーチンに対する以前の呼び出しによって戻されたもの。この ID は、障害またはその他の注目すべき問題をレポートするために作成されたエントリーを示します。引数として複数の FFDC 障害 ID をこのコマンドに指定できます。ただし、ID は、間にホワイト・スペースを入れずに、コンマ (,) で区切る必要があります。

終了状況

fcdecode は、完了すると、以下の整数の状況コードのどれかを戻します。

0 FFDC 障害 ID が正常にデコードされました。

2 ヘルプ情報が表示され、処理が終了しました。

10 FFDC 障害 ID が引数としてこのコマンドに指定されませんでした。

12 無効なオプションまたはサポートされていないオプションが、このコマンドに指定されました。

27 標準出力デバイスに情報が書き込まれませんでした。FFDC 障害 ID の引数が無効でした。

例

FFDC 障害 ID は、右から左に読み取られる、base-64 の値によって表されています。各ドットは先行ゼロを表します。FFDC 障害 ID **.3Iv04ZVVfvp.wtY0xRXQ7**..... をコンポーネント部分にデコードするには、次のようにします。

```
fcdecode .3Iv04ZVVfvp.wtY0xRXQ7...................

Information for First Failure Data Capture identifier
        .3Iv04ZVVfvp.wtY0xRXQ7...................
        Generated by the local system
        Generated Thu Sep 3 11:40:17 1998 EDT
        Recorded to the AIX Error Log using template 460bb505
        To obtain the AIX Error Log information for this entry, issue
        the following command on the local system:
        TZ=EST5EDT errpt -a -j 460bb505 -s 0903114098 | more
```

```
        Search this output for an AIX Error Log entry that contains
        the following ERROR ID code:
         .3Iv04ZVVfvp.wtY0xRXQ7...................
```

別のノードで同じコマンドを実行すると、以下のような結果になります。

```
fcdecode .3Iv04ZVVfvp.wtY0xRXQ7...................

Information for First Failure Data Capture identifier
        .3Iv04ZVVfvp.wtY0xRXQ7...................
        Generated on a remote system with the following Internet address:
          9.114.55.125
        Generated Thu Sep 3 11:40:17 1998 EDT
        Recorded to the AIX Error Log using template 460bb505
        TZ=EST5EDT errpt -a -j 460bb505 -s 0903114098 | more
        Search this output for an AIX Error Log entry that contains
        the following ERROR ID code:
         .3Iv04ZVVfvp.wtY0xRXQ7...................
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

『fcdispfid コマンド』

520 ページの『fcreport コマンド』

525 ページの『fcstkrpt コマンド』

---

fcdispfid コマンド

目的

First Failure Data Capture 障害 ID (FFDC 障害 ID) を標準エラー・デバイスに表示します。

構文

/opt/rsct/bin/fcdispfid [-q]*FFDC_Failure_ID* | **-h**

説明

このコマンドは、FFDC 障害 ID の値を標準エラー・デバイスに表示するために、スクリプトで使用します。このインターフェースが提供されているのは、スクリプト・プログラムには、終了状況コード、シグナル、標準出力、および標準エラーを使用する以外に、クライアントにデータを戻すための仕組みがないためです。このような環境で FFDC 障害 ID をクライアントに戻すタスクを実現するために、**fcdispfid** は、XPG/4 カタログ式メッセージ番号 **2615-000** を使用して、この情報を標準エラー・デバイスに表示します。スクリプトのクライアントは、スクリプトから、標準エラー情報をキャプチャー、特定のメッセージ番号を検索し、FFDC 障害 ID を入手することができます。

スクリプトでは、スクリプトが生成するすべての FFDC 障害 ID を、スクリプト・ユーザー文書の形で標準エラー・デバイスに送信するよう指示する必要があります。デフォルトで、クライアントがこの動作を知っていると考えることはできません。

フラグ

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

-q このコマンドからの警告メッセージを抑止します。このオプションを指定しない場合は、無効な FFDC 障害 ID が検出されると、このコマンドがメッセージを表示します。

パラメーター

FFDC_Failure_ID

FFDC 障害 ID を指定します。**fcpushstk** または **fclogerr** に対する以前の呼び出しによって戻された ID であり、スクリプトによって検出された障害をレポートするために作成されたエントリーを示します。この ID は、FFDC メッセージ **2615-000** を使用して、標準エラー・デバイスに書き込まれます。

終了状況

0 FFDC 障害 ID が標準エラーに表示されました。

2 ヘルプ情報が表示され、処理が終了しました。

12 標準エラー・デバイスに情報が書き込まれませんでした。無効なオプションが指定されました。

27 標準エラー・デバイスに情報が書き込まれませんでした。*FFDC_Failure_ID* 引数が有効なフォーマットになっていません。

例

標準出力デバイスを使用して FFDC 障害 ID をクライアントに表示するには、次のようにします。

```
   FID=$(fclogerr -e FFDC_ERROR -t ERRID_SP_FFDCEXMPL_ER -i /usr/lpp/ssp/inc/
myprog.h -r myprog -s myprog.ksh -p $LINEPOS -v "1.1" -l PSSP -d $MINUSDOPTS -x
$MINUSXOPTS -y $MINUSYOPTS -b "myprog Configuration Failure - Exiting")
    RC=$?
    if ((RC == 0))
    then
         fcdispfid $FID
         return 1
    else
         :
    fi
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

497 ページの『fcdecode コマンド』

『fcfilter コマンド』

525 ページの『fcstkrpt コマンド』

---

fcfilter コマンド

目的

ファイルまたは標準入力にあるすべての First Failure Data Capture (FFDC) 障害 ID を見つけ、表示します。複数のファイルを指定することができます。

構文

/opt/rsct/bin/fcfilter [*file_name*] [. . .]

説明

このコマンドは、First Failure Data Capture (FFDC) 障害 ID の引数としてリストされているすべてのファイルをスキャンします。引数としてファイル名を指定しない場合、このコマンドは、標準入力を調べて FFDC 障害 ID を探します。FFDC 障害 ID を検出した場合、**fcfilter** は、それ自身の行の標準出力にその ID を表示します。

fcfilter は、標準エラー・デバイスを通じて子プロセスによって戻された FFDC 障害 ID を抽出するために、スクリプトで使用することができます。

fcfilter が入力で複数の FFDC 障害 ID を検出した場合は、見つかったすべての FFDC 障害 ID を、それぞれ別々の出力行に表示します。

パラメーター

file_name

FFDC 障害 ID の検索対象となるファイルの名前。複数のファイルを指定することができます。ファイル名を指定しない場合、**fcfilter** は標準入力から読み取ります。

終了状況

fcfilter は、完了すると、以下の整数の状況コードを戻します。

0 **fcfilter** の実行が完了しました。この終了状況は、必ずしも、すべての FFDC 障害 ID が検出されたことを意味するものではありません。

> 0 **fcfilter** はシグナルによって中断または停止されました。終了状況は、コマンドを停止したシグナルの整数の値です。

例

FFDC 障害 ID は、右から左に読み取られる、base-64 の値によって表されています。各ドットは先行ゼロを表します。コマンド *mycmd* の実行によって生成されたすべての FFDC 障害 ID のリストを入手するには、次のようにします。

```
 mycmd 2> /tmp/errout
 fcfilter /tmp/errout
/.00...JMr4r.p9E.xRXQ7.....................
/.00...JMr4r.pMx.xRXQ7.....................
```

親スクリプトの子プロセスから FFDC 障害 ID を入手するには、次のように、スクリプトで **fcfilter** コマンドを使用することができます。

```
RESULTS=$(mychild 2> /tmp/errout)
if (($? != 0))               # mychild ended in failure, get FFDC ID
then
    cat /tmp/errout | fcfilter | read FIRST_FFDCID
else
    rm -f /tmp/errout
fi
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

499 ページの『fcdispfid コマンド』
520 ページの『fcreport コマンド』
525 ページの『fcstkrpt コマンド』

---

fcinit コマンド

目的

First Failure Data Capture 実行環境を確立し、継承します。

構文

Bourne シェルおよび Korn シェルの場合:

/opt/rsct/bin/fcinit.sh [[**-l**] [**-s** {**c** | **i** }]] | [**-h**]

C シェルの場合:

source /opt/rsct/bin/fcinit.csh [[**-l**] [-s { **c** | **i** }]] | [**-h**]

説明

スクリプト・プログラムで FFDC インターフェースを使用して、AIX エラー・ログ、BSD システム・ログ、または FFDC エラー・スタックに情報を記録したい場合は、このインターフェースを使用する必要があります。

アプリケーションで FFDC 環境を確立したいときの理由を以下に示します。

- スクリプトで、AIX エラー・ログに情報を記録したい場合。スクリプトで **fcinit** を使用して、基本 FFDC 環境を確立できます。
- スクリプト自身、およびスクリプト自身によって作成されたすべての子孫プロセスまたは子プロセスを使用して、障害情報を FFDC エラー・スタックに記録したい場合。この場合、このスクリプトは、自分自身を、複数の「下位レベル」アプリケーションを作成する「トップレベル」アプリケーションとみなします。また、「トップレベル」アプリケーションが成功するかどうかは、これらの「下位レベル」アプリケーションが成功するかどうかにかかっています。 **fcinit** をこのように使用する場合、このプロセスは、FFDC エラー・スタック環境を確立 または 作成 すると言われます。
- 障害情報またはトレース情報を FFDC エラー・スタックまたは FFDC トレースに記録させたい祖先プロセスによって呼び出されたときだけ、スクリプトでこれらのデバイスを使用する場合。つまり、それ以外のときは、スクリプトでこれらのデバイスを使用したくない場合です。 **fcinit** をこのように使用する場合、このプロセスは、 FFDC エラー・スタック環境を継承 すると言われます。

FFDC インターフェースを使用して、情報を AIX エラー・ログまたは BSD システム・ログに記録したいすべてのプロセスは、FFDC 環境を確立する必要があります。プロセスが FFDC エラー・スタックを利用したくない場合、このプロセスは、 FFDC エラー・スタックを利用しない基本 FFDC 環境を確立することができます。プロセスが、プロセス自身、プロセスが作成するすべてのスレッド、およびプロセスが作成するすべての子孫プロセスからの障害情報を、FFDC エラー・スタックに記録したい場合は、FFDC エラー・スタックを含む FFDC エラー・スタック環境がこのプロセスによって確立されます。プロセスが、祖

先プロセスから依頼があったときだけ、障害情報を FFDC エラー・スタック・ファイルに記録したい場合 (これ以外のときは、 FFDC エラー・スタックに障害情報を記録しない) は、 FFDC エラー・スタックを含む *FFDC* エラー・スタック環境 がこのプロセスによって継承されます。

FFDC エラー・スタックを含む FFDC エラー・スタック環境では、FFDC エラー・スタック・ファイルが確保されるため、障害情報は、**/var/adm/ffdc/stacks** ディレクトリー内のファイルに記録されます。これらのファイルでは、***script_name.PID.date_and_time*** の命名フォーマットが使用されます。*script_name* はスクリプト自身の名前、*PID* はスクリプトのプロセス ID、*date_and_time* は、スクリプトが実行されたときの日付/時刻です。このスクリプトまたはこのスクリプトの子プロセスが障害情報を FFDC エラー・スタックに記録するときは、障害情報は必ずこのファイルに記録されます。

FFDC エラー・スタックに情報を記録するには、プロセスは、**fcpushstk** FFDC インターフェースを使用しなければならず、確立された FFDC エラー・スタック環境内で実行されていなければなりません。FFDC エラー・スタック環境が存在しない場合、あるいは FFDC エラー・スタック環境が存在しても、**fcpushstk** インターフェースが使用されない場合、情報は、そのプロセスによって FFDC エラー・スタックに記録されません。この機能を使用すると、障害デバッグ情報が不必要なときに、プロセスを通常モードまたは「サイレント」モードで実行することができます。ただし、プロセスが特別なデバッグ環境内で呼び出されたときには、この情報を入手することもできます。

fcinit コマンドで、適切に FFDC 環境をスクリプトに設定するには、このコマンドを FFDC クライアントのプロセス環境 (ソース) 内で実行する必要があります。**fcinit** をクライアント・プロセス・イメージ内で実行するには、このコマンドを使用するスクリプト・ベースの FFDC クライアントが、このコマンドの送信元でなければなりません。これを行わないと、FFDC インターフェースはそれ自身のプロセス・イメージ内で実行されます。また、FFDC インターフェースが完了すると、 FFDC 環境のすべての設定が失われます。スクリプト・ベースのアプリケーションがどのように **fcinit** コマンドの送信元であるかを示すため、 Korn シェル・プログラムは次の指示を出します。

```
. fcinit.sh <options and arguments>
```

C シェル・スクリプトは次のような指示を出します。

```
source fcinit.csh <options and arguments>
```

fclogerr FFDC インターフェースを使用するプロセスは、 *FFDC* 環境 を確立する必要があります。プロセスが **fclogerr** インターフェースだけを使用したい場合は、 FFDC エラー・スタックのない *FFDC* 環境を確立することができます。

作成しようとしたときに FFDC 環境が既に存在していた場合、スクリプトは、FFDC 環境を作成する代わりに、既存の FFDC 環境を継承します。

フラグ

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

-l プロセスが AIX エラー・ログだけを利用したいことを指示します。AIX エラー・ログは FFDC エラー・スタック環境内で使用することができます。このため、**-s** オプションを指定する場合は、このオプションは必要ありません。

-s FFDC エラー・スタック環境を確立することを指示します。**fcpushstk** インターフェースを使用したいアプリケーションは、このフラグを指定する必要があります。このコマンドが正常に完了する

と、FFDC エラー・スタック・ファイルが、スクリプト用に **/var/adm/ffdc/stacks** ディレクトリーに確保されます。このフラグは、指定可能な次の 2 つのオプションのどちらか 1 つと一緒に指定する必要があります。

c FFDC エラー・スタック環境を作成する ように要求します。 FFDC エラー・スタック環境が祖先プロセスによって作成されていなかった場合は、作成されます。この環境が祖先プロセスによって既に作成されている場合、このプロセスは、**i** オプションが指定されているかのように、 FFDC エラー・スタック環境を継承 します。

i FFDC エラー・スタック環境が祖先プロセスによって既に確立されている場合は、それを継承 するように指定します。FFDC エラー・スタック環境がこれまでに祖先プロセスによって確立されていない場合、このプロセスに対して、この環境は確立されません。したがって、このプロセスは FFDC エラー・スタックを利用できません (ただし、AIX エラー・ログおよび BSD システム・ログを利用することはできます)。

パラメーター

file_name

FFDC 障害 ID の検索対象となるファイルの名前。複数のファイルを指定することができます。ファイル名を指定しない場合、**fcfilter** は標準入力から読み取ります。

終了状況

fcinit は、完了すると、以下の終了状況コードを戻します。

0 FFDC 環境が正常に確立されました。

1 FFDC 環境が正常に継承されました。

2 ヘルプ情報が表示され、処理が終了しました。

fcinit は、障害を検出すると、以下の終了状況コードを戻します。

12 FFDC 環境が確立または継承されませんでした - 認識できない関数パラメーターが指定されました。

13 FFDC エラー・スタック環境が確立または継承されませんでした - 呼び出し側が、 FFDC 環境の作成および継承の両方を行うように指示しました。

14 この呼び出しで FFDC 環境が確立されませんでした - 呼び出し側は、自分のために、既に FFDC 環境を確立しています - 複数回このルーチンが実行されている可能性があります。

15 FFDC エラー・スタック環境が確立または継承されませんでした - FFDC エラー環境が存在しませんでした。また FC_INHERIT オプションが指定されました。

16 FFDC 環境が確立または継承されませんでした - このルーチンが、クライアントのプロセス環境を変更できませんでした。

17 FFDC 環境が確立または継承されませんでした - FFDC 環境が破壊されていると判断されました。使用不能とみなされます。

18 FFDC 環境が確立または継承されませんでした - このルーチンが、クライアントのプロセス環境の変更に必要なメモリーを割り当てることができませんでした。

19 FFDC エラー・スタック環境が確立または継承されませんでした - 呼び出しプロセスに対して、FFDC エラー・スタック・ファイルを確保できません。- FFDC エラー・スタック・ディレクトリーが存在しないか、使用できません。

21 FFDC エラー・スタック環境が確立または継承されませんでした - 呼び出しプロセスに対して、FFDC エラー・スタック・ファイルを確保できません。- このファイルが既に存在しています。

42 FFDC エラー・スタック環境が確立または継承されませんでした - システム管理者が、FFDC エラー・スタックの作成および使用を使用不可にしています。スクリプトが確立できるのは、AIX エラー・ログおよび BSD システム・ログを利用する基本 FFDC 環境だけです。

99 FFDC 環境が確立または継承されませんでした - **fcinit** 内で予期せぬ内部障害が発生しました。この状態のときは、お客様は注意する必要があります。また、アプリケーション・サポート・サービスが必要となる場合があります。

例

Korn シェル・スクリプトで AIX エラー・ログおよび BSD システム・ログだけを使用する (FFDC エラー・スタックは使用または確保しない) 基本 FFDC 環境を確立するには、次のようにします。

```
# Set up an FFDC Environment to use the AIX Error Log only. An FFDC Error
# Stack is not needed for this script.
. fcinit.sh -1
rc=$?
if ((rc != 0))
    then
         print "fcinit failed with exit code of $rc"
         exit 1
fi
# Normal processing starts
```

スクリプトおよびすべての子孫プロセスが、障害情報を FFDC エラー・スタックに記録するようになる FFDC エラー・スタック環境を Korn シェル・スクリプトで確立するには、次のようにします。

```
# Set up FFDC Environment to record failure information to the FFDC Error
# Stack
. fcinit.sh -sc
rc=$?
if ((rc != 0))
then
    print "fcinit failed with a code of $rc"
    exit 1
fi
# Normal processing starts
```

注**:** FFDC エラー・スタック環境が、**fcinit** 呼び出しによって作成される代わりに、継承されました、という表示を FFDC クライアントが受け取る場合があります。このようになるのは、祖先プロセスによって既に FFDC エラー・スタック環境が確立されている場合です。

プロセスの親プロセスから FFDC エラー・スタック環境を継承するには、次のようにします。

```
# Inherit an FFDC Environment from parent process if it exists - otherwise,
# operate in a normal "silent" mode
. fcinit.sh -si
rc=$?
if ((rc != 0))
then
    print "fcinit failed with a code of $rc"
    exit 1
fi
# Normal processing starts
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

493 ページの『fccheck コマンド』

『fclogerr コマンド』

513 ページの『fcpushstk コマンド』

527 ページの『fcteststk コマンド』

---

fclogerr コマンド

目的

障害または注目すべき状態に関する情報を AIX エラー・ログおよび BSD システム・ログに記録します。

構文

/opt/rsct/bin/fclogerr { **-e** *event* **-t** *error_template_label* **-i** *error_template_headerfile* **-r** *resource* **-s** *source_filename* **-p** *line_of_code_pos* **-v** *sidlevel* **-l** *lpp_name* **-a** *assoc_fid* { [**-d** *detail_data_item*[,*detail_data_item*,...] **-x** *detail_data_type*[,*detail_data_type*,...] **-y** *detail_data_len*[,*detail_data_len*,...]] | [**-f** *detail_data_file*] } **-b** *BSD_syslog_message_text* } | **-h**

説明

このインターフェースは、情報を AIX エラー・ログおよび BSD システム・ログに記録したいスクリプト・プログラムで使用します。このデバイスに書き込まれた情報は、システム管理者またはシステム・オペレーターが、注意を必要とする、どのような障害状態またはその他の注意すべき状態がシステムで発生したかを判別するときに使用します。AIX エラー・ログおよび BSD システム・ログの目的は、ある状態に関する十分な情報を記録することです。レポートがあれば、どのような状態がどこで発生したかを調べるために、その状態をもう一度発生させる必要なしに、その状態の内容、影響、および応答を判別できます。なんらかの直接の介入が行われるまで続く永続的な障害状態が発生するソフトウェア、またはシステム管理者が注意しなければならない状態が発生するソフトウェアは、 **fclogerr** を使用して、この情報を AIX エラー・ログおよび BSD システム・ログに記録する必要があります。

スクリプトは、**fclogerr** を使用する前に、基本 FFDC 環境または FFDC エラー・スタック環境を作成するか継承して、この環境を確立しておく必要があります。**fclogerr** は、AIX エラー・ログおよび BSD システム・ログの環境が確立されていなくても、これらに情報を記録します。しかし、このインターフェースは、これらの環境のどちらかが存在しないと、FFDC 障害 ID を生成することができません。

FFDC エラー・スタックを使用するように設計されているプロセスも **fclogerr** インターフェースを利用できます。また、解決するための管理者の注意または介入が必要な状態が発生する場合は、このインターフェースを利用する必要があります。

この状態およびその状態の検出場所を正しく識別するために、FFDC ポリシーでは、**fclogerr** を、スクリプトのソース・コード・モジュール内でインラインで呼び出すこと、および状態が検出されたらすぐに呼び出すことを推奨しています。 **fclogerr** は、この状態が発生したソース・コードを識別および探索できるように、ソース・コード・ファイル名およびコード行の情報を記録します。この情報を記録する必要がある場合は、**fclogerr** をサブルーチンまたは自動ロードされたルーチンから呼び出すことができます。ただし、この外部ルーチンが、場所に関する情報および障害に関する必要な詳細情報のすべてを入手できる場合に限られます。外部記録ルーチンは、問題が検出された実際の場所を記録する必要があります。

fclogerr は、AIX エラー・ログおよび BSD システム・ログの両方に情報を報告しますが、それぞれの記録デバイスごとに、異なるオプションをこのインターフェースに指定する必要があります。AIX エラー・ログに記録される詳細データ情報は、 BSD システム・ログにも記録されるわけではありません。BSD システム・ログ情報は、別のコマンド・オプションによって提供されます。このため、場合によって、**fclogerr** ユーザーは、この呼び出しで一部の情報を複写する必要があります。

フラグ

-a 障害状態の FFDC 障害 ID を含みます。この障害状態は、今回、障害情報を記録させる原因となったか、それに影響を与えたアプリケーションが使用するソフトウェアによってレポートされたものです。この ID は、ソフトウェアの結果指示の一部として、このアプリケーションに戻されたはずです。呼び出し側は、この ID をここで指定します。この結果、 FFDC エラー・スタックは、今回作成している障害レポートを、以前記録された障害レポートに関連付けることができるようになります。これにより、問題判別者は、このアプリケーションやその他のアプリケーションのさまざまな症状から、その他のソフトウェアの根本原因へと、障害の原因をトレースできるようになります。その他のソフトウェア障害がこの状態の原因でない場合、あるいは、その他のソフトウェアが、結果情報の一部として FFDC 障害 ID を戻さなかった場合は、このオプションを省略する必要があります。

-b BSD システム・ログに書き込まれる次のメッセージを指定します。

-d 状態に関する詳細情報を提供する 1 つ以上のデータ項目。この項目は、AIX エラー・ログ・エントリーに詳細データを提供するために使用されます。情報の詳細が長すぎる場合は、これらの詳細をファイルに書き込むことができます。このファイルの名前は、*detail_data_file* パラメーターとして指定します。詳細データ・ファイル名を指定する場合は、このオプションを省略する必要があります。*detail_data* パラメーターも *detail_data_file* パラメーターも指定しない場合、あるいはそれらが無効と判断された場合は、AIX エラー・ログの詳細データとして null 情報が記録されます。

このオプションを使用して、複数のデータ項目を指定することができます。各データ項目は、間にホワイト・スペース文字を入れずに、コンマ (,) で区切る必要があります。データ項目にホワイト・スペース文字が組み込まれている場合は、二重引用符 (") でこのデータ項目を囲む必要があります。データ項目自身にコンマ (,) を入れることはできません。コマンドは、コンマをフィールド区切り文字として解釈するからです。

このオプションには、**-x** および **-y** のオプションを付ける必要があります。

-e FFDC ログ・イベント・タイプを指定します。現在の有効値は FFDC_EMERG、FFDC_ERROR、FFDC_STATE、FFDC_TRACE、FFDC_RECOV、および FFDC_DEBUG です。このコードは、ログされているイベント・タイプ (緊急状態、永続状態、情報イベントの通知、デバッグ情報など) および状態の重大度の一般的記述になっています。このオプションを指定しない場合は、イベント・タイプ FFDC_DEBUG がこの問題レコードに割り当てられます。

-f 報告されている状態の詳細を含むファイルの名前。このオプションを使用するのは、詳細が長すぎて、**fclogerr** がアプリケーションに残した、100 バイトの残りの詳細データ情報内に記録できない場合、あるいは詳細情報を分析できるユーティリティーがある場合です。このファイルの内容は **/var/adm/ffdc/dumps** ディレクトリーにコピーされます。また、ファイルの新しいロケーションは、 AIX エラー・ログ・エントリーの詳細データとして記録されます。

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

-i **-l** オプションで指定した *error_template_label* に対応するエラー・ロギング・テンプレート識別番号を含むヘッダー・ファイル (.h) の絶対パス名を指定します。このテンプレートは、ノードのエ

ラー・ロギング・テンプレート・リポジトリー (**/var/adm/ras/errtmplt**) にもなければなりません。このヘッダー・ファイルは、**errupdate** コマンドによって、ソース・コードの構築手順の一部として生成されました。また、ソフトウェアと一緒にノードにインストールされる LPP パッケージに含まれているはずです。このオプションを指定しないか、スクリプトが実行されたときにヘッダー・ファイルが見つからない場合は、**fclogerr** は、独自のデフォルト・エラー・テンプレート (ラベル FFDC_DEF_TPLT_TR、ID コード 2B4F5CAB) を使用して障害情報を記録します。

-l 出荷の際にこのソフトウェアが入っていたライセンス・プログラミング製品の短縮名を指定します。この値は LPP の受け入れ可能名として、お客様およびアプリケーション・サポート・サービスの両者に認識できるものでなければなりません。この値の例として PSSP、 GPFS™、LoadLeveler®、および RSCT があります。このオプションが指定されていないか、無効と判断された場合は、文字列 **PPS_PRODUCT** が使用されます。

-p ソース・コード・モジュール内の、状態がレポートされているコード行の場所を指定します。指定した値は、有効な整数値でなければなりません。状態を正しく識別し、位置指定するために、この値は、状態を検出したコード行にできるだけ近いものでなければなりません。Korn シェル・スクリプトでは **$LINENO** の値を使用することができます。特殊な行カウント変数を提供していないスクリプト言語では、ここにシンボル値を指定することができます。この値を使用すると、開発者は、**fclogerr** が使用されている、ソース・コード内のスポットを探し出すことができます。このオプションが無効であるか、指定されていない場合は、 **0** の値が使用されます。

-q このコマンドからの警告メッセージの生成を抑止します。警告が生成されるのは、コマンドが、見つからない情報の代わりにデフォルト情報を使用しなければならない場合、またはコマンドが、*detail_data_file* を **/var/adm/ffdc/dumps** ディレクトリーにコピーできない場合です。

-r ソフトウェア・コンポーネント名を指定します。この名前は、レポートを作成するソフトウェアのシンボル名であり、お客様およびアプリケーション・サポート・サービスの両者に認識できる名前でなければなりません。文字列は 16 文字に制限されています。

-s レポートされている状態が発生したコード行を含むソース・ファイル名を指定します。Korn シェルおよび Borne シェルのスクリプトでは、このオプションに対する引数を **$0** に設定する必要があります。C シェル・スクリプトでは、この引数を **${0}** に設定します。このオプションが指定されないか、無効の場合は、文字列 **unknown_file** が使用されます。

-t エラー・ログ・リポジトリーで AIX エラー・ロギング・テンプレートに与えられたシンボル・ラベルを示します。エラー・ロギング・テンプレートを作成する **errupdate** コマンドは、このラベルを整数コードにマップするマクロを生成します。このラベルは **ERRID_** の文字で始まり、最大長は 19 文字です。このオプションを指定しないか、スクリプトが実行されるときにヘッダー・ファイルが見つからない場合、**fclogerr** は、**errlogger** を呼び出して、OPMSG テンプレートを使用して AIX エラー・ログにメッセージを作成します。

-v 記録されている状態を検出したソース・コード・モジュールの SCCS バージョン番号を示します。SCCS の制御下で構築されたソース・コードの場合、この番号を **"1.1"** (二重引用符が必要) に設定する必要があります。このオプションが指定されないか、無効の場合は、 **unknown** の文字列が使用されます。

-x **-d** オプションで指定されたデータ項目の情報を AIX エラー・ログに記録する際に、このデータ項目をどのように解釈するかを指示します。データ項目のタイプは、**-t** オプションで指定した AIX エラー・ロギング・テンプレートの対応するフィールドに一致する必要があります。各タイプは、**-d** リストの対応するデータ項目をどのように解釈するかを示します。このオプションの受け入れ可能な値は ALPHA、HEX、および DEC です。 **-d** リストのそれぞれの引数ごとに、一致するタイプを **-x** 引数にリストする必要があります。

-d オプションを指定した場合は、このオプションを指定する必要があります。

-y **-d** オプションで指定したデータ項目 (バイト) の長さを示します。これらの長さは、**-t** オプションで指定した AIX エラー・ロギング・テンプレートの対応するフィールドに一致する必要があります。 **-d** リストのそれぞれの引数ごとに、一致するタイプを **-y** 引数にリストする必要があります。

-d オプションを指定した場合は、このオプションを指定する必要があります。

パラメーター

file_name

FFDC 障害 ID の検索対象となるファイルの名前。複数のファイルを指定することができます。ファイル名を指定しない場合、**fcfilter** は標準入力から読み取ります。

終了状況

fclogerr は、正常に完了すると、以下の終了状況コードを戻します。

0 情報が、AIX エラー・ログおよび BSD システム・ログに書き込むためのキューに正常に入りました。レコードの FFDC 障害 ID は標準出力に表示されます。この値を入手するには、呼び出し側は標準出力をキャプチャーする必要があります。

2 ヘルプ情報が表示され、処理が終了しました。

12 AIX エラー・ログに情報が記録されませんでした。コマンドによって FFDC 障害 ID は提供されません。コマンド・ユーザーが、このコマンドに無効なオプションを指定しました。

AIX 以外の AIX プラットフォームでは、障害が発生すると、**fclogerr** は、次の終了状況コードを戻します。

38 この問題について、レコードを BSD システム・ログに入れることができませんでした。システム・ログに障害が発生しています。AIX システムでは、AIX エラー・ログにレポートが記録されます。その他のシステムでは、障害とみなされます。

不完全な情報と一緒に **fclogerr** を指定すると、このコマンドは、欠落情報の代わりにデフォルト情報を使用し、FFDC エラー・スタックでレコードを作成するようにします。これらの場合、警告が生成されます。また、**-q** オプションが指定されていない限り、警告メッセージが生成されます。複数の警告状態が検出された場合、コマンドは、最も重大と判断した状態の終了状況コードを戻します。警告状態が検出された場合、 **fclogerr** によって次の終了状況コードが戻されます。

10 コマンド・ユーザーが、このコマンドに **-i** オプションを指定できませんでした。あるいは、-i オプションの引数として指定するヘッダー・ファイル名が見つかりませんでした。この場合、コマンドは First Failure Data Capture のデフォルト・テンプレート (ラベル FFDC_DEF_TPLT_TR、ID コード 2B4F5CAB) を使用して、一般情報を AIX エラー・ログに記録します。

26 詳細データ・ストリングおよび詳細データ・ファイルの両方がこのルーチンに指定されました。このルーチンは、詳細データ・ストリングを選択し、詳細データ・ファイルを無視しました。

28 問題を検出するリソースの名前が指定されませんでした。存在しないリソース名の代わりに、デフォルト・リソース名 **ffdc** が使用されます。

29 検出アプリケーション情報 (ソース・コード・ファイル名、ソース・コード・ファイル・バージョン、LPP 名、コード行の位置) のコンポーネントが、1 つ以上、指定されませんでした。欠落情報の代わりにデフォルト情報が使用されます。

32 *detail_data_file* パラメーターで指定されたファイルを **/var/adm/ffdc/dumps** ディレクトリーにコピ

ーできませんでした。 FFDC エラー・スタック・エントリーには、このファイルの元のバージョンが引用されます。このファイルの元のコピーを廃棄しないでください。

33 **-e** オプションが指定されなかったか、このオプションで、有効な FFDC イベント・タイプが指定されませんでした。この問題レコードには、イベント・タイプ FFDC_DEBUG が割り当てられています。

34 *format* パラメーターにメッセージが指定されませんでした。この結果、この問題では、一般的なメッセージが BSD システム・ログに記録されました。

35 この問題についての詳細な情報が提供されませんでした。問題の詳細を示すこれらの詳細情報がないと、以後の問題分析が困難になる場合があります。

36 詳細データ・ストリングの長さが、AIX エラー・ログ・エントリーの制限を超えました。使用可能スペースに適合するように、詳細データが切り捨てられました。この切り捨てによって、問題の一部の情報が失われた可能性があります。

37 このルーチンが作成したレポートの FFDC エラー ID を作成できませんでした。 FFDC 障害 ID は標準出力に書き込まれていません。しかし、この問題の情報は AIX エラー・ログおよび BSD システム・ログに記録されました。

38 この問題について、レコードが BSD システム・ログで作成されませんでした。システム・ログが使用可能でないか、システム・ログに問題が発生している可能性があります。AIX システムでは、AIX エラー・ログにレポートが記録されます。その他のシステムでは、障害とみなされます。

例

この例では、Korn シェル・スクリプトがファイルの構成情報へのアクセスを試みます。この試みが失敗すると、このコードは、次のテンプレート・ソース・コードを使用して AIX エラー・ログに失敗を記録します。

```
*! mymesgcat.cat
+ SP_FFDCEXMPL_ER:
    Comment        = "Configuration Failed - Exiting"
    Class          = S
    Log       = true
    Report         = true
    Alert          = false
    Err_Type  = PERM
    Err_Desc  = {3, 10, "CONFIGURATION FAILURE - EXITING"}
    Prob_Causes    = E89B
    User_Causes    = E811
    User_Actions   = 1056
    Fail_Causes    = E906, E915, F072, 108E
    Fail_Actions   = {5, 14, "VERIFY USER HAS CORRECT PERMISSIONS TO ACCESS FILE"},
             {5, 15, "VERIFY CONFIGURATION FILE"}
    Detail_Data    = 46, 00A2, ALPHA
    Detail_Data    = 42, EB2B, ALPHA
    Detail_Data    = 42, 0030, ALPHA
    Detail_Data    = 16, EB00, ALPHA
    Detail_Data    = 16, 0027, ALPHA
    Detail_Data    = 4, 8183, DEC
    Detail_Data    = 4, 8015, DEC
    Detail_Data    = 60, 8172, ALPHA
```

この定義によって、次の AIX エラー・ロギング・テンプレートが生成されます。

```
LABEL:            ERRID_SP_FFDCEXMPL_ER
IDENTIFIER:        <calculated by errupdate during source code build>

Date/Time:         <filled in by AIX Error Log subsystem>
Sequence Number:   <filled in by AIX Error Log subsystem>
Machine Id:        <filled in by AIX Error Log subsystem>
Node Id:           <filled in by AIX Error Log subsystem>
Class:             S
Type:              PERM
Resource Name:     <filled in by -r option to fclogerr>

Description
CONFIGURATION FAILURE - EXITING

Probable Causes
COULD NOT ACCESS CONFIGURATION FILE

User Causes
USER CORRUPTED THE CONFIGRATION DATABASE OR METHOD

    Recommended Actions
    RE-CREATE FILE

Failure Causes
COULD NOT ACCESS CONFIGURATION FILE
PERMISSIONS ERROR ACCESSING CONFIGURATION DATABASE
FILE READ ERROR
FILE IS CORRUPT

    Recommended Actions
    VERIFY USER HAS CORRECT PERMISSIONS TO ACCESS FILE
    VERIFY CONFIGURATION FILE

Detail Data
DETECTING MODULE
<filled in by fclogerr options>
ERROR ID
<The FFDC Failure Identifier created by fclogerr>
REFERENCE CODE
<The -a option value to fclogerr>
FILE NAME
<Must be supplied as part of -d option list to fclogerr>
FUNCTION
<Must be supplied as part of -d option list to fclogerr>
RETURN CODE<Must be supplied as part of -d option list to fclogerr>
ERROR CODE AS DEFINED IN sys/errno.h
<Must be supplied as part of -d option list to fclogerr>
USER ID<Must be supplied as part of -d option list to fclogerr>
```

最初の 3 つの詳細データ・フィールドは、パラメーターで渡された情報から、**fclogerr** ルーチンによって作成されます。残りの詳細データは **-d** オプションを使用して指定する必要があります。また、指定したデータ・タイプを **-x** オプションによって指示する必要もあります。下記のソース・コード・セグメントの例は、このテンプレートがどのように実行されるか、および AIX エラー・ログおよび BSD システム・ログに情報を記録するために、**fclogerr** がどのように起動されるかを示しています。

```
typeset CONFIG_FNAME
typeset INBUF
typeset MINUSDOPTS
typeset MINUSXOPTS
typeset MINUSYOPTS
typeset FID
integer MYCLIENT
integer RC
    :
```

```
MYCLIENT=$$
CONFIG_FNAME="/configfile.bin"
exec 3< $CONFIG_FNAME
    :
read -u3 INBUF
RC=$?
if ((RC != 0))
then
    # Create Detail Data Memory Block for AIX Error Log Template
    # Need to know the EXACT structure of the Template to do this correctly.
    #    Field 1 - filled in by fc_log_error
    #    Field 2 - filled in by fc_log_error
    #    Field 3 - filled in by fc_log_error
    #    Field 4 - name of configuration file being used - 16 bytes
    #    Field 5 - name of function call that failed - 16 bytes
    #    Field 6 - return code from failing function - 4 byte integer
    #    Field 7 - errno from failing function call (unused) - 4 byte integer
    #    Field 8 - user ID using this software - remaining space (62 bytes)
    # This source code supplied fields 4 through 8 in the "-d" option, and
    # describes the data types for each in the "-x" option.
    MINUSDOPTS=$CONFIG_FNAME
    MINUSXOPTS="ALPHA"
    MINUSYOPTS="16"
    MINUSDOPTS="$MINUSDOPTS,read"
    MINUSXOPTS="$MINUSXOPTS,ALPHA"
    MINUSYOPTS="$MINUSYOPTS,16"
    MINUSDOPTS="$MINUSDOPTS,$RC"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,0"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,$MYCLIENT"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,60"
    FID=$(fclogerr -e FFDC_ERROR -t ERRID_SP_FFDCEXMPL_ER -i /usr/lpp/ssp/inc/
myprog.h -r myprog -s myprog.ksh -p $LINEPOS -v "1.1" -l PSSP -d $MINUSDOPTS -x
$MINUSXOPTS -y $MINUSYOPTS -b "myprog Configuration Failure - Exiting")
    RC=$?
    if ((RC == 0))
    then
         fcdispfid $FID
         return 1
    else
         :
    fi
fi
```

ここでは、上記とは少しだけ違う例を取り上げます。同じ AIX エラー・ロギング・テンプレートを使用しますが、今回は、外部コマンドを使用して、このソース・コードが提供するファイルから構成データを入手します。このコマンドは非ゼロの終了状況で終了し、障害状態を検出すると、 FFDC 障害 ID を標準出力に表示します。また、**-d** リストでの二重引用符の使用法を示すために、構成ファイルは名前に組み込みスペースをもちます。

```
typeset CONFIG_FNAME
typeset INBUF
typeset MINUSDOPTS
typeset MINUSXOPTS
typeset MINUSYOPTS
typeset FID
typeset OUTPUT
integer MYCLIENT
integer RC
    :
```

```
MYCLIENT=$$
CONFIG_FNAME="This is a test"
OUTPUT=$(configdabeast $CONFIG_FNAME)
RC=$?
if ((RC != 0))
then
    # Create Detail Data Memory Block for AIX Error Log Template
    # Need to know the EXACT structure of the Template to do this correctly.
    #    Field 1 - filled in by fc_log_error
    #    Field 2 - filled in by fc_log_error
    #    Field 3 - filled in by fc_log_error
    #    Field 4 - name of configuration file being used - 16 bytes
    #    Field 5 - name of function call that failed - 16 bytes
    #    Field 6 - return code from failing function - 4 byte integer
    #    Field 7 - errno from failing function call (unused) - 4 byte integer
    #    Field 8 - user ID using this software - remaining space (62 bytes)
    # This source code supplied fields 4 through 8 in the "-d" option, and
    # describes the data types for each in the "-x" option.
    MINUSDOPTS="¥""$CONFIG_FNAME"¥""
    MINUSXOPTS="ALPHA"
    MINUSYOPTS="16"
    MINUSDOPTS="$MINUSDOPTS,configdabeast"
    MINUSXOPTS="$MINUSXOPTS,ALPHA"
    MINUSYOPTS="$MINUSYOPTS,16"
    MINUSDOPTS="$MINUSDOPTS,$RC"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,0"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,$MYCLIENT"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,60"
    FID=$(fclogerr -e FFDC_ERROR -t ERRID_SP_FFDCEXMPL_ER -i /usr/lpp/ssp/inc/
myprog.h -r myprog -s myprog.ksh -p $LINEPOS -v "1.1" -l PSSP -d $MINUSDOPTS -x
$MINUSXOPTS -y $MINUSYOPTS -a $OUTPUT -b "myprog Configuration Failure - Exiting")
    RC=$?
    if ((RC == 0))
    then
         fcdispfid $FID
         return 1
    else
         :
    fi
fi
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

520 ページの『fcreport コマンド』

関連情報:

ct_ffdc.h ファイル

---

fcpushstk コマンド

目的

障害または注目すべき状態に関する情報を First Failure Data Capture エラー・スタックに記録します。

構文

/opt/rsct/bin/fcpushstk { **[-a** *assoc_fid*] **-c** *message_catalog_name* **-m** *message_set* **-n** *message_number* **[-o** *message_param* [*,message_param,...*]] **-l** *lpp_name* **-p** *line_of_code_pos* **-r** *resource* **-s** *source_filename* **-v** *sidlevel* {**[-d** *detail_data*] | **[-f** *detail_data_file*]} } *default_message* | **-h**

説明

fcpushstk は、障害情報を FFDC エラー・スタックに記録するためにスクリプトで使用します。スクリプトは、後の問題判別に使用するために、記述情報およびデバッグ・データを FFDC エラー・スタックに記録します。

FFDC エラー・スタックは、共通タスクを実行するために 1 つのノード上で複数の関連プロセスまたはスレッドを一緒に実行しているときに発生する障害状態を理解する際に使用します。 1 つ以上のスレッドまたはサブプロセスを作成し、次に、それら自身も、スレッドまたはサブプロセスを作成するようなアプリケーションの場合にこのデバイスは最適です。 FFDC エラー・スタックを使用するために、スクリプトは、**fcinit** インターフェースを使用して *FFDC* エラー・スタック環境 を確立します。この環境が確立されれば、アプリケーションおよびそのすべての子孫が FFDC エラー・スタックを利用できるようになります。

すべてのソフトウェア・アプリケーションが FFDC エラー・スタック環境を確立できるわけではありません。ただし、FFDC エラー・スタック環境を確立した他のアプリケーションまたはスクリプトが、この環境を確立できないアプリケーションを呼び出すことができます。このような場合、このソフトウェアを呼び出すスクリプトまたはアプリケーションが、このソフトウェアから障害情報を取り込んで、呼び出した他のソフトウェアからの他の障害情報と一緒に分析し、障害に何らかの関連またはパターンがあるかどうかを調べたいことがあります。このため、通常の動作条件では、普通、FFDC エラー・スタックを利用できないソフトウェアは、 FFDC エラー・スタックを使用するクライアントによって呼び出される場合、少なくとも、FFDC エラー・スタックの使用をサポートする必要があります。これは、**fcinit** インターフェースを使用して、親プロセスから FFDC エラー・スタック環境を継承する ことによって可能になります。

fcpushstk は、注目すべき状態に関する記述および詳細を FFDC エラー・スタックに記録します。作成または継承によって *FFDC* エラー・スタック環境 が確立されていない場合は、**fcpushstk** は情報を記録せず、制御を呼び出し側に戻します。この処理によって、デバッグ情報が要求されない場合に、スクリプトを通常の「サイレント」モードで実行できるようになります。また、デバッグ情報が要求されたときに、スクリプトが FFDC エラー・スタックの使用をサポートできるようになります。

FFDC エラー・スタック環境が確立したら、スクリプトは、明示的に **fcpushstk** を呼び出して、情報を FFDC エラー・スタックに記録する必要があります。障害データを記録できるようにするには、単に環境を確立するだけでは不十分です。**fclogerr** コマンドは、FFDC エラー・スタックにレコードを作成しません。

その状態およびその状態の検出場所を正しく識別するために、**fcpushstk** をスクリプトのソース・コード・モジュール内でインラインで呼び出すこと、および状態が検出されたらすぐに呼び出すことが必要です。**fcpushstk** は、この状態が発生したソース・コードを識別および探索できるように、ソース・コード・ファイル名およびコード行の情報を記録します。この情報を記録する必要がある場合は、**fcpushstk** をサブルーチンまたは自動ロードされたルーチンから呼び出すことができます。ただし、この外部ルーチンが、ロケーション情報および障害に関する必要な詳細情報のすべてを入手できる場合に限られます。外部記録ルーチンは、問題が検出された実際のロケーションを記録する必要があります。

FFDC エラー・スタック・エントリーの最大サイズは、**<rsct/ct_ffdc.h>** ヘッダー・ファイルの FC_STACK_MAX 定義で指定します。FC_STACK_MAX ではバイトで長さを定義します。この値は大まかなガイドとしてだけ使用する必要があります。この長さには、検出ファイル情報、記述情報、および

FFDC 障害 ID 情報を記録する **fcpushstk** によって使用されるデータが含まれるためです。
FC_STACK_MAX バイトより長いレコードは、FC_STACK_MAX 制限に適合するように切り捨てられます。

フラグ

-a 障害状態の FFDC 障害 ID を指定します。この障害状態は、今回記録されている障害の原因となったか、またはそれに影響を与えたアプリケーションが使用するソフトウェアによってレポートされたものです。この ID は、ソフトウェアの結果指示の一部として、このアプリケーションに戻されたはずです。呼び出し側は、この ID をここで指定します。この結果、 FFDC エラー・スタックは、今回作成している障害レポートを、以前記録された障害レポートに関連付けることができるようになります。これにより、問題判別者は、このアプリケーションやその他のアプリケーションのさまざまな症状から、その他のソフトウェアの根本原因へと、障害の原因をトレースできるようになります。その他のソフトウェア障害がこの状態の原因でない場合、あるいは、その他のソフトウェアが、結果情報の一部として FFDC 障害 ID を戻さなかった場合は、 **-a** オプションを指定する必要はありません。

-c 記録されている障害の記述が入った XPG/4 準拠のメッセージ・カタログの名前を指示します。この名前は、**/usr/lib/nls/msg/$LANG** ディレクトリーに対する相対名です。メッセージ・カタログが見つからない場合は、障害を説明する *default_message* が表示されます。 *default_message* は、ロケール間では変換されないことに注意してください。

-d 状態に関する詳細情報を提供する文字列。これは、AIX エラー・ログが使用する詳細データの概念に似たものです。情報の詳細が長すぎる場合は、これらの詳細をファイルに書き込むことができます。このファイルの名前は、**-f** オプションの引数として指定します。**-d** オプションと **-f** オプションを同時に指定することはできません。**-d** オプションまたは **-f** オプションのどちらも指定しないか、どちらも無効と判断された場合は、文字列 **no detail data** が記録されます。

-f AIX エラー・ログが使用する詳細データの概念に似た、レポートされている状態の詳細を含むファイルの名前を指定します。このオプションを使用するのは、詳細が長すぎて FFDC エラー・スタック自身に記録できない場合、あるいは詳細情報を分析できるユーティリティーがある場合です。このファイルの内容は **/var/adm/ffdc/dumps** ディレクトリーにコピーされます。また、ファイルの新しいロケーションは、FFDC エラー・スタックの詳細データとして記録されます。状態の詳細を含むファイルが存在しない場合は、このオプションは指定しません。**-d** オプションと **-f** オプションを同時に指定することはできません。

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

-l このソフトウェアが添付されたライセンス・プログラムの名前の省略形を指定します。この値は、ライセンス・プログラムに付ける妥当な名前として、カスタマー・サポート・サービスとアプリケーション・サポート・サービスの両方で認識できるものでなければなりません。このオプションが指定されないか、無効である場合は、文字列 **PPS_PRODUCT** が使用されます。

-m メッセージ・カタログ・ファイルの障害を説明するメッセージが入ったメッセージ・セットを指定します。このメッセージ・セットが見つからない場合は、障害を説明する *default_message* が表示されます。 **default_message** は、ユーザーのロケールに変換されないことに注意してください。

-n 記録されている障害を示すメッセージ番号を指定します。このメッセージが見つからない場合は、障害を説明する *default_message* が表示されます。 *default_message* は、ユーザーのロケールに変換されないことに注意してください。

-o **-n** オプションで指示したメッセージ内の置換パラメーターのリストを指定します。**fcpushstk** はシェル操作環境にあるため、置換パラメーター (%s) として文字列しかサポートしません。複数の

置換パラメーターを指定する場合は、それぞれのパラメーターをコンマ (,) で区切る必要があります。これらの置換パラメーターのうちで組み込みホワイト・スペースを含むものは、二重引用符 ("") で囲む必要があります。

-q このコマンドからの警告メッセージの生成を抑止します。警告が生成されるのは、コマンドが、見つからない情報の代わりにデフォルト情報を使用しなければならない場合、またはコマンドが、*detail_data_file* を **/var/adm/ffdc/dumps** ディレクトリーにコピーできない場合です。

-r ソフトウェア・コンポーネント名を指定します。この名前は、レポートを作成するソフトウェアのシンボル名であり、お客様およびアプリケーション・サポート・サービスの両者に認識できる名前でなければなりません。

-p ソース・コード・モジュール内の、状態がレポートされているコード行の場所を指定します。指定した値は、有効な整数値でなければなりません。状態を正しく識別し、位置指定するために、この値は、状態を検出したコード行にできるだけ近いものでなければなりません。Korn シェル・スクリプトでは **$LINENO** の値を使用することができます。特殊な行カウント変数を提供していないスクリプト言語では、ここにシンボル値を指定することができます。開発者はこの値を使用して、**fcpushstk** が使用されている、ソース・コード内のスポットを探し出すことができます。このオプションが無効であるか、指定されていない場合は、 **0** の値が使用されます。

-s レポートされている状態が発生したコード行を含むソース・ファイル名を指定します。Korn シェルおよび Borne シェルのスクリプトでは、このオプションに対する引数を **$0** に設定する必要があります。C シェル・スクリプトでは、この引数を ${0} に設定します。このオプションが指定されないか、無効の場合は、文字列 **unknown_file** が使用されます。

-v 記録されている状態を検出したソース・コード・モジュールの SCCS バージョン番号を示します。SCCS 制御下のソース・コードの場合、この番号を **"1.1"** (二重引用符が必要) に設定する必要があります。このオプションが指定されないか、無効の場合は、 **unknown** の文字列が使用されます。

パラメーター

default_message

-c、**-m**、および **-n** の各オプションで指定したメッセージ・カタログ情報から情報を取り出せない場合に、障害の説明として使用するデフォルト・メッセージを指示します。このストリングに定位置パラメーターが含まれる場合は、すべての定位置パラメーターを文字列 (%s) として指定する必要があります。メッセージに組み込みホワイト・スペースが含まれる場合は、メッセージを二重引用符で囲む必要があります。 **fcpushstk** では、このストリングの全体の長さは 72 文字に制限されています。

終了状況

fcpushstk は、正常に完了すると、以下の終了状況コードを戻します。

0 FFDC エラー・スタック環境が存在し、障害情報が FFDC エラー・スタックに正常に記録されました。レコードの FFDC 障害 ID は標準出力に表示されます。この値を入手するには、呼び出し側は標準出力をキャプチャーする必要があります。

2 ヘルプ情報が表示され、処理が終了しました。

fcpushstk は、障害が発生すると、次の終了状況コードを戻します。

11 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。オプションを使用するために要求されたクライアントがこのリリースの FFDC ソフトウェアでサポートされていません。

12 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。インターフェースに不明な関数パラメーターが指定されています。

15 FFDC エラー・スタック環境が存在しません。FFDC エラー・スタックに情報が記録されませんでした。FFDC 障害 ID はこのコマンドによって生成されません。 **fcinit** を使用して継承する FFDC エラー・スタック環境が存在しない場合、これが FFDC クライアントに対する通常の戻りコードです。

17 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。FFDC エラー・スタック環境が破壊されていると判断されました。使用不能とみなされます。

19 FFDC エラー・スタックに情報が記録されませんでした。- FFDC エラー・スタック・ディレクトリーが存在しないか、使用できません。 FFDC 障害 ID はこのコマンドによって提供されません。

20 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。FFDC エラー・スタック・ファイルにアクセスできません。ファイルが削除されているか、FFDC エラー・スタックへのアクセスを禁止するために、ファイルまたはディレクトリーに対する権限が変更されている可能性があります。

22 FFDC エラー・スタックに情報が記録されませんでした。- このインターフェース専用に FFDC エラー・スタック・ファイルをロックできませんでした。このファイルをロックする試みを繰り返しましたが、すべての試みが失敗しました。別のプロセスがこのファイルをロックしており、このファイルを解放できないか、他のプロセスがハングアップしていて、別のプロセスが FFDC エラー・スタックを使用できないようにしている可能性があります。 FFDC 障害 ID はこのコマンドによって提供されません。

24 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。FFDC エラー・スタックが壊れていると判断されました。クライアントは FFDC エラー・スタック環境を使用不能とみなします。

25 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。FFDC エラー・スタック・ファイル名はディレクトリー名に設定されます。FFDC エラー・スタック環境は破壊されているか、使用不能とみなされます。

32 ダンプ・ファイルを **/var/adm/ffdc/dumps** ディレクトリーにコピーできませんでした。**/var/adm/ffdc** ディレクトリーが入ったファイルシステムのスペース不足です。**fcclear** コマンドを使用して、不必要な FFDC エラー・スタックおよびダンプ・ファイルを削除する必要があります。あるいは、システム管理者がファイルシステムにより多くのスペースを追加する必要があります。 FFDC 障害 ID はこのコマンドによって提供されません。

40 FFDC エラー・スタックに情報が記録されませんでした。- FFDC エラー・スタックに情報を記録できませんでした。 **/var/adm/ffdc** ディレクトリーが入ったファイルシステムのスペース不足です。**fcclear** コマンドを使用して、不必要な FFDC エラー・スタックおよびダンプ・ファイルを削除する必要があります。あるいは、システム管理者がファイルシステムにより多くのスペースを追加する必要があります。 FFDC 障害 ID はこのコマンドによって提供されません。

41 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID は提供されません。FFDC エラー・スタックから制御情報を読み取るときか、FFDC エラー・スタックに問題情報を書き込むときに、障害が発生しました。クライアントは、この問題についてのエントリーが記録されなかったと判断する必要があります。

99 FFDC エラー・スタックに情報が記録されませんでした。このコマンドによって FFDC 障害 ID

は提供されません。**fc_push_stack** ルーチンで予期せぬ内部障害が発生しました。アプリケーション・サポート・サービスは、この問題に注意しなければならない場合があります。

不完全な情報と一緒に **fcpushstk** を指定すると、このコマンドは、欠落情報の代わりにデフォルト情報を使用し、FFDC エラー・スタックにレコードを作成するようにします。これらの場合、警告が生成されます。また、**-q** オプションが指定されていない限り、警告メッセージが標準エラー・デバイスに表示されます。複数の警告状態が検出された場合、コマンドは、検出した最も重大な警告状態に対応する終了状況コードを生成します。警告状態が検出された場合、**fcpushstk** によって次の終了状況コードが戻されます。

26 詳細データ・ストリングおよび詳細データ・ファイルの両方がこのルーチンに指定されました。このルーチンは、詳細データ・ストリングを選択し、詳細データ・ファイルを無視しました。

28 問題を検出するリソースの名前が指定されませんでした。存在しないリソース名の代わりに、デフォルト・リソース名が使用されます。

29 検出アプリケーション情報 (ソース・コード・ファイル名、ソース・コード・ファイル・バージョン、LPP 名、コード行の位置) のコンポーネントが、1 つ以上、指定されませんでした。欠落情報の代わりにデフォルト情報が使用されます。

30 問題の内容を説明するデフォルト・メッセージが指定されませんでした。記述メッセージを含む XPG/4 メッセージ・カタログが見つからない場合、 **fcstkrpt** コマンドによって、この状態の説明は表示されません。

31 問題の内容を説明するメッセージが提供されませんでした。あるいは、XPG/4 情報のコンポーネント (カタログ・ファイル名、メッセージ・セット番号、メッセージ番号) が提供されませんでした。**fcstkrpt** コマンドによって、この状態の説明を表示することができません。

32 *detail_data_file* パラメーターで指定されたファイルを **/var/adm/ffdc/dumps** ディレクトリーにコピーできませんでした。 FFDC エラー・スタック・エントリーには、このファイルの元のバージョンが引用されます。このファイルの元のコピーを廃棄しないでください。

35 この問題についての詳細な情報が提供されませんでした。問題の詳細を示すこれらの詳細情報がないと、以後の問題分析が困難になる場合があります。

37 このルーチンが作成したレポートの FFDC 障害 ID を作成できませんでした。 FFDC 障害 ID はこのコマンドによって提供されません。しかし、問題についての情報は FFDC エラー・スタックに記録されました。

44 このコマンドに提供された情報が原因となって、FFDC エラー・スタック・レコードが FC_STACK_MAX 制限を超えました。システム制限内に記録できるように、レコードが切り捨てられました。切り捨て処理の間に、障害に関する重要情報が失われた可能性があります。スクリプトを変更して、より少ない情報を提供させます。あるいは、代わりに詳細データ・ファイルに情報を記録させ、詳細データ・ファイル名をこのコマンドにサブミットさせます。

例

プロセスによって FFDC 環境が確立または継承されたときに、障害に関する情報を FFDC エラー・スタックに記録するには、以下のようにします。

```
#!/bin/ksh
    :
    :
cp /tmp/workfile $FILENAME
RC=$?
if ((RC != 0))
then
```

```
    FFDCID=$(fcpushstk -c mymsg.cat -m2 -n10 -o$FILENAME -r myprog
         -d"cp exit status $RC - file being copied /tmp/workfile" -s$0
         -p$LINENO -v"1.1" -lPSSP "Cannot update configuration file %1$s")
    if (($? == 0))
    then
         fcdispfid $FFDCID
         return 1
    fi
fi
    :
    :
```

使用可能なコード行変数をもたないスクリプト言語から同じ記録を作成するには、以下のようにします。

```
#!/bin/bsh
    :
    :
CODESCTN=14             # Used to identify where in the script code we are
cp /tmp/workfile $FILENAME
RC=$?
if test $RC -ne 0
then
    FFDCID=`fcpushstk -c mymsg.cat -m2 -n10 -o$FILENAME -r myprog
         -d"cp exit status $RC - file being copied /tmp/workfile" -s$0
         -p$CODESCTN -v"1.1" -lPSSP "Cannot update configuration file %1$s"`
    if test $? -eq 0
    then
         fcdispfid $FFDCID
         return 1
    fi
fi
CODESECTION=15               # New code section begins - a different task starts
    :
    :
```

FFDC を活用するアプリケーションによって以前 FFDC エラー・スタックに記録された別の障害状態に関連する障害状態についての情報を記録するには、以下のようにします。

```
#!/bin/ksh
    :
    :
ASSOC_FID=$(/usr/lpp/ssp/bin/somecmd -a -b)
RC=$?if ((RC != 0))
then
    FFDCID=$(fcpushstk -a$ASSOC_FID -c mymsg.cat -m2 -n10 -o$FILENAME -r myprog
         -d"cp exit status $RC - file being copied /tmp/workfile" -s$0
         -p$LINENO -v"1.1" -lPSSP "Cannot update configuration file %1$s")
    if (($? == 0))
    then
         fcdispfid $FFDCID
         return 1
    fi
fi
    :
    :
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

502 ページの『fcinit コマンド』

fcreport コマンド

目的

障害およびその障害に関連するすべての障害のレポートを検出し、表示します。

構文

/opt/rsct/bin/fcreport { [**-a**] *FFDC_Failure_ID* } | **-h**

説明

fcreport は FFDC 障害 ID をデコードし、その ID で識別できる障害に関するレポートを入手します。このコマンドは、FFDC 障害 ID に関連している障害があるかどうかについても検出し、ある場合は、その障害についてのレポートを入手します。コマンドは、次の条件のどれかが満たされるまで、検出した各障害のレポートを調べて関連する障害を探し、その障害に関するレポートを入手し続けます。

- これ以上の関連する障害が検出されない。
- 関連する障害についてのレポートが見つからない。関連する障害のレポートが、この時に検出することのできないリモート・ノード上にあるか、障害のレコードが、それがあったノードから削除された場合です。

このコマンドを使用すると、ユーザーは、特定の障害の原因となった障害のリスト全体のレポートを入手することができます。**fcreport** は、このコマンドに指定された初期障害によって引き起こされた可能性のある障害についてのレポートを検出することはできません。このコマンドが入手できるのは、この障害を引き起こした障害のレポートだけです。

フラグ

-a 障害のレポートに含まれるすべての情報を表示します。デフォルトでは、障害レポートが生成されたノードのネットワーク・アドレス、障害レポートのタイム・スタンプ、および障害レポートに記録された問題の説明が表示されます。

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

パラメーター

FFDC_Failure_ID

レポートを開始する障害の FFDC 障害 ID を指定します。 **fcreport** は、この障害についての障害情報、およびこのレポートによって関連する障害としてリストされるすべての障害についての障害情報を入手しようとします。このコマンドには FFDC 障害 ID を 1 つしか指定できません。

セキュリティー

fcreport は **rsh** を使用して、リモート・ノードにある障害レポートを入手します。ユーザーは、これらのリモート・ノードに対して **rsh** コマンドを実行できる十分な特権をもっていなければなりません。ユーザーがこの許可をもっていない場合、**fcreport** は、ローカル・ノード上の関連する障害のリストをトレースすることしかできません。

終了状況

fcreport は、完了すると、以下の終了状況コードのどれかを生成します。

0 指定された FFDC 障害 ID について、障害レポートが検出され、表示されました。ゼロ以上の関連する障害レポートも検出され、表示された可能性があります。

2 ヘルプ情報が表示され、処理が終了しました。

10 必須のオプションまたは必須の引数が指定されていません。

11 このコマンドに指定された FFDC 障害 ID は、FFDC ソフトウェアのより新しいリリースによって生成されました。コマンドがこの ID を正しく解釈できません。

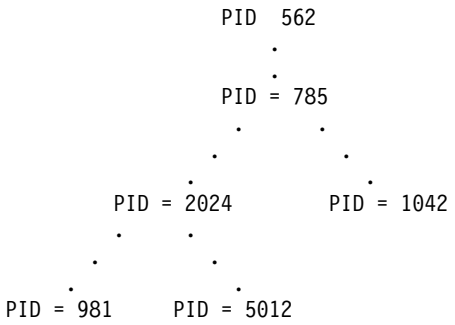
12 このコマンドに不明なオプションが指定されました。

20 FFDC 障害 ID は、このシステム上の FFDC エラー・スタックに作成されたエントリーを示しますが、FFDC エラー・スタック・ファイルにアクセスすることができません。ファイルが削除されているか、ファイルへのアクセスを防止するために、ファイルに関する権限が変更されている可能性があります。

27 このコマンドに指定された FFDC 障害 ID は無効な ID です。

例

次の親と子の順番でいくつかのプロセスが作成された場合を説明します。



この例で、プロセス 785 は FFDC 障害 ID **.3Iv04ZVVfvp.wtY0xRXQ7.....** を生成しました。次にそれをプロセス 562 に戻しました。FFDC 障害 ID **.3Iv04ZVVfvp.wtY0xRXQ7.....** およびこの特定の障害につながる以前の障害についての詳細レポートを取得するには、次のようにします。

```
$ fcreport -a .3Iv04ZVVfvp.wtY0xRXQ7...................
```

このレポートには、指定した FFDC 障害 ID の詳細、およびそれを引き起こした可能性のある、プロセス 2024、1042、981、および 5012 のすべての障害の詳細が含まれます。このレポートには、プロセス 785 の障害の結果として引き起こされた可能性のある、プロセス 562 のすべての障害は含みません。

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

495 ページの『fcclear コマンド』

506 ページの『fclogerr コマンド』

525 ページの『fcstkrpt コマンド』

fcstat コマンド

目的

指定されたファイバー・チャネル・デバイス・ドライバーが収集した統計情報を表示します。

構文

fcstat [**-z** [**-d** | **-c**] | **-d** | **-e** [**-d** | **-c**] | **-c**] *Device_Name*

説明

fcstat コマンドは、指定されたファイバー・チャネル・デバイス・ドライバーが収集した統計情報を表示します。オプションとして一般的なデバイスの統計情報のほかに、デバイス固有の統計情報を表示するように指定できます。フラグを指定しないと、**fcstat** コマンドは一般的なデバイスの統計情報のみを表示します。**fcstat** コマンドは、次の手順を使用して統計情報を収集します。

1. **fcstat** のメッセージ・カタログをオープンし、パラメーター・リストを検査する。
2. 選択されたアダプターに関連した情報を検索するために、オブジェクト・データ・マネージャー (ODM) データベースにアクセスする。
3. 選択されたアダプターのポートに関連した情報を検索するために、ODM データベースにアクセスする。
4. アダプター統計情報をオープンし、アクセスする。
5. **-z** フラグが指定されている場合は、一部の統計情報をリセットする。
6. 統計情報を報告して、終了する。

無効な *Device_Name* を指定すると、**fcstat** コマンドは ODM データベース内でデバイスを検索できないことを示すエラー・メッセージを戻します。

また、**fcstat** コマンドは、指定された *Device_Name* がネットワークに接続されていない (つまり、リンクが切断されている) 場合にも、**-d** フラグを使用して診断モードでデバイスをオープンすることにより統計情報を報告します。リンクが切断されていて、デバイスが診断モード以外のモードでオープンされると、**fcstat** コマンドによる出力の生成が遅れます。**-c** フラグを使用して、この遅延をなくすことができます。デバイスが既にオープンされていて、**fcstat** コマンドが **-d** フラグで開始すると、デバイスのオープン操作は EACCESS エラーで失敗します。

fcstat コマンドは、指定された *Device_Name* から統計情報を取り出すことができない場合も、ODM データベースから取り出した情報は報告します。

フラグ

項目	説明
-c	デバイスが診断モード以外のモードでオープンされ、リンクが切断されている場合に、出力の生成での遅延をなくします。
-d	診断モードでアダプターをオープンして統計情報を表示します。
-e	デバイス固有の統計情報 (ドライバー統計、リンク統計、FC4 タイプ) を含む、すべての統計情報を表示します。
-z	一部の統計情報をリセットして初期値に戻します。このフラグを発行できるのは、特権ユーザーのみです。

パラメーター

項目	説明
Device_Name	ファイバー・チャネル・デバイスの名前。例: `fcs0`。

統計フィールド

注**:** アダプターによっては、特定の統計情報をサポートしない場合があります。サポートされない統計フィールドの値は、常に 0 です。**fcstat** コマンドを **–z** フラグと一緒に使用すると、アスタリスク (*) の付いたパラメーターはすべてリセットされて初期値に戻ります。

fcstat コマンドの出力に表示される統計フィールドとその説明は次のとおりです。

	項目	説明
	Device Type	アダプターの記述を表示します。
	Serial Number	アダプターのシリアル番号を表示します。
	Option ROM Version	アダプターのオプション ROM のバージョンを表示します。
	ZA	アダプターの VPD から ZA フィールドを表示します。
	Node WWN	アダプターの worldwide name を表示します。
	Port FC ID	アダプターの SCSI ID を表示します。
	Port Type	アダプターの接続タイプを表示します。
	Port Speed	アダプターのスピードを表示します。
	Port WWN	ポートの worldwide name を表示します。
	Seconds Since Last Reset	アダプター上の統計情報の最終リセット以後の秒数を表示します。
*	Frames	送信および受信されたフレームの数を表示します。
*	Words	送信および受信されたワードの数を表示します。
*	LIP Count	LIP カウントを表示します。
*	NOS Count	NOS カウントを表示します。
	Error Frames	エラーのあるフレームの数を表示します。
*	Dumped Frames	ダンプされたフレームを表示します。
	Link Failure Count	リンク障害カウントを表示します。
	Loss of Sync Count	同期が失われた回数を表示します。
	Loss of Signal	シグナルが失われた回数を表示します。
	Primitive Seq Protocol Err Count	プリミティブ・シーケンスのエラー回数を表示します。
	Invalid Tx Word Count	無効な転送の発生数を表示します。
	Invalid CRC Count	CRC エラーの発生数を表示します。
	FC SCSI Adapter Driver Information: No DMA Resource Count	DMA リソースが使用不可であった回数を表示します。
	FC SCSI Adapter Driver Information: No Adapter Elements Count	使用可能なアダプター・エレメントがなかった回数を表示します。
	FC SCSI Adapter Driver Information: No Command Resource Count	使用可能なコマンド・リソースがなかった回数を表示します。
*	FC SCSI Traffic Statistics: Input Requests	入力要求の数を表示します。
*	FC SCSI Traffic Statistics: Output Requests	出力要求の数を表示します。
*	FC SCSI Traffic Statistics: Control Requests	制御要求の数を表示します。
*	FC SCSI Traffic Statistics: Input Bytes	入力バイトの数を表示します。
*	FC SCSI Traffic Statistics: Output Bytes	出力バイトの数を表示します。
	Adapter Effective Max Transfer Value	実効最大転送値を表示します。
	FC4 Types : Supported ULP	サポートされる ULP を表示します。
	FC4 Types : Active ULP	アクティブな ULP を表示します。

終了状況

項目	説明
0	コマンドは正常に完了しました。
>0	エラーが発生しました。

例

1. ファイバー・チャネル・デバイス・ドライバー fcs0 の統計情報を表示するには、次のように入力します。

   ```
   fcstat fcs0
   ```

   次のような出力が表示されます。

   注: さまざまな AIX コマンドの出力フォーマットは常に静的とは限りません。**fcstat** コマンドの出力が常に次のようになるという予想でプログラムを作成しないようにしてください。

   ```
   FIBRE CHANNEL STATISTICS REPORT: fcs0

       Device Type: FC Adapter (df1000f9)
       Serial Number: 1E313BB001
       Option ROM Version: 02C82115
       ZA: B1F2.10A5
       Node WWN: 20000000C9487B04
       Port WWN: 10000000C9416DA4

       FC4 Types
         Supported: 0x0000010000000000000000000000000000000000000000000000000000000000
         Active:    0x0000010000000000000000000000000000000000000000000000000000000000
       Class of Service: 4
       Port FC ID: 011400
       Port Speed (supported): 2 GBIT
       Port Speed (running):   1 GBIT
       Port Type: Fabric

       Seconds Since Last Reset: 345422

       Transmit Statistics    Receive Statistics
       -------------------    ------------------
       Frames: 1              Frames: 1
       Words: 1               Words: 1

       LIP Count: 1
       NOS Count: 1
       Error Frames:  1
       Dumped Frames: 1
       Link Failure Count: 1
       Loss of Sync Count: 1
       Loss of Signal: 1
       Primitive Seq Protocol Err Count: 1
       Invalid Tx Word Count: 1
       Invalid CRC Count: 1

       FC SCSI Adapter Driver Information
         No DMA Resource Count: 0
         No Adapter Elements Count: 0
         No Command Resource Count: 0

       FC SCSI Traffic Statistics
         Input Requests:   16289
         Output Requests:  48930
         Control Requests: 11791
         Input Bytes:  128349517
         Output Bytes: 209883136
   ```

位置

/usr/sbin/fcstat

関連資料:

529 ページの『fddistat コマンド』

関連情報:

netstat コマンド

tokstat コマンド

---

fcstkrpt コマンド

目的

FFDC エラー・スタック・ファイルの内容を表示します。

構文

/opt/rsct/bin/fcstkrpt { **[-a]** **[-p** | **-r]** { **-f** *FFDC_Failure_Identifier* [**-i**] | **-s** *FFDC_Error_Stack_File_Name* } } | **[-h**]

説明

fcstkrpt は、既存の FFDC エラー・スタック・ファイルを読み取り、その内容を標準出力デバイスに表示します。FFDC エラー・スタック・ファイルの指示には、ファイル自身の名前、またはそのファイル内の特定のレコードを参照する FFDC 障害 ID を使用します。

FFDC エラー・スタックからの情報は、*related failure conditions* (デフォルト) または *software layer* の 2 つのフォーマットのどちらかで表示することができます。

フラグ

-a FFDC エラー・スタック内のエントリーに関するすべての情報を表示するように指示します。デフォルト・アクションでは、レコードのタイム・スタンプおよび問題の説明が表示されます。

-f FFDC エラー・スタックの検索に使用する FFDC 障害 ID を指定します。**fcstkrpt** は FFDC 障害 ID をデコードし、その FFDC 障害 ID に関連した FFDC エラー・スタックを探し出し、次に FFDC エラー・スタックを処理します。このフラグでは、FFDC 障害 ID を 1 つしか指定できません。

-h ヘルプ・メッセージを標準出力に表示し、終了します。指定したオプションにかかわりなく、それ以外の処理は実行されません。

-i **-f** フラグで指定した特定の障害レポートに関連した情報だけを表示します。デフォルトでは、FFDC エラー・スタック内のすべてのレコードが表示されます。

-p プロセスの方向に従って FFDC エラー・スタックからの情報を表示します。出力は、プロセスが作成された順序 (親と子のプロセスの関係) に並べられます。最初に子プロセスの情報、次に親プロセスの情報が表示されます。この表示は、最初にどの問題が発生したか、後でそれらの問題が原因でどの問題が発生したかを知るために使用します。

-r 問題の関係に従って FFDC エラー・スタックからの情報を表示します。問題は、関連する問題と一緒に表示されます。この表示は、他のどの問題の発生が原因となって問題が発生したかを知るために使用します。これはデフォルトです。

-s 検査対象の FFDC エラー・スタックの名前を指定します。この名前は、FFDC エラー・スタックの絶対パス名または相対パス名のどちらでもかまいません。このフラグでは、FFDC エラー・スタック・ファイル名を 1 つしか指定できません。相対ファイル名を使用すると、ファイルは、それが存在するノードの **/var/adm/ffdc/stacks** ディレクトリーにあるものとされます。

パラメーター

FFDC_Failure_ID
レポートを開始する障害の FFDC 障害 ID を指定します。 **fcreport** は、この障害についての障害情報、およびこのレポートによって関連する障害としてリストされるすべての障害についての障害情報を入手しようとします。このコマンドには FFDC 障害 ID を 1 つしか指定できません。

セキュリティー

fcreport は **rsh** を使用して、リモート・ノードにある障害レポートを入手します。ユーザーは、これらのリモート・ノードに対して **rsh** コマンドを実行できる十分な特権をもっていなければなりません。ユーザーがこの許可をもっていない場合、**fcreport** は、ローカル・ノード上の関連する障害のリストをトレースすることしかできません。

終了状況

fcstkrpt は、完了すると、以下の整数の終了状況コードを出します。

0 FFDC エラー・スタック・ファイルを検出することができました。内容を標準出力デバイスに表示しました。

2 ヘルプ情報が表示され、処理が終了しました。

12 無効なオプションが指定されました。

14 標準出力デバイスに情報が書き込まれませんでした。**-f** オプションが使用されましたが、*FFDC Error Identifier* の引数が無効でした。

20 標準出力デバイスに情報が書き込まれませんでした。**-s** オプションが使用されましたが、*FFDC Error Stack File* の引数が見つかりませんでした。

27 標準出力デバイスに情報が書き込まれませんでした。呼び出し側は有効な *FFDC Failure Identifier* を指定しましたが、FFDC 障害 ID によって参照されるファイルがこのノードに記録されていません。 **fcdecode** コマンドを使用して、FFDC エラー・スタックのあるノードを検出してください。

81 標準出力デバイスに情報が書き込まれませんでした。標準出力に情報を書き込んでいるときに、障害が発生しました。アプリケーションは、標準出力が出力を受け付けることができないと判断します。

85 標準出力デバイスに情報が書き込まれませんでした。呼び出し側は有効な FFDC 障害 ID を指定しましたが、FFDC 障害 ID によって参照されるファイルが存在しません。

例

FFDC エラー・スタック・ファイル **/var/adm/ffdc/stacks/myprog.562.19981001143052** に保管されている情報の短いレポートを取得するには、次のようにします。

```
$ fcstkrpt -r -s myprog.562.19981001143052
```

FFDC 障害 ID **.3Iv04ZVVfvp.wtY0xRXQ7.....** が記録されている FFDC エラー・スタックにある情報の詳細なレポートを取得するには、次のようにします。

```
$ fcstkrpt -p -f .3Iv04ZVVfvp.wtY0xRXQ7...................
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

495 ページの『fcclear コマンド』

513 ページの『fcpushstk コマンド』

520 ページの『fcreport コマンド』

---

fcteststk コマンド

目的

First Failure Data Capture エラー・スタック環境の存在を検査します。

構文

/opt/rsct/bin/fcteststk [-q] | [-h]

説明

fcteststk は、FFDC エラー・スタックを使用して、これらの機能が活動化されているかどうかを検査したいすべてのアプリケーションから呼び出すことができます。このテストを行うことにより、アプリケーションは、*FFDC 環境* が確立されていないのに障害情報を収集するというパフォーマンス負荷を回避することができます。このインターフェースは、主に、ライブラリー・ルーチンが使用するために提供されています。ライブラリー・ルーチンは、クライアント・アプリケーションが *FFDC 環境* を確立または継承したかどうかを知ることができないからです。

プロセスが、プロセス自身、プロセスが作成するすべてのスレッド、およびプロセスが作成するすべての子孫プロセスからの障害情報を FFDC エラー・スタックに記録したい場合、 *FFDC* エラー・スタック環境はプロセスによって確立されます。祖先プロセスから要求があったときだけ、プロセスが障害情報を FFDC エラー・スタック・ファイルに記録したい場合 (これ以外のときは、FFDC エラー・スタックに障害情報を記録しない) は、*FFDC* エラー・スタック環境 はプロセスによって継承されます。プロセスは **fcinit** を使用して、FFDC エラー・スタック環境を確立または継承します。

FFDC エラー・スタック環境では、FFDC エラー・スタック・ファイルが確保されるため、障害情報は、**/var/adm/ffdc/stacks** ディレクトリー内のファイルに記録されます。これらのファイルでは、***script_name.PID.date_and_time*** の命名フォーマットが使用されます。*script_name* はスクリプト自身の名前、*PID* はスクリプトのプロセス ID、*date_and_time* は、スクリプトが実行されたときの日付/時刻です。このスクリプトまたはこのスクリプトの子プロセスが障害情報を FFDC エラー・スタックに記録するときは、障害情報は必ずこのファイルに記録されます。

アプリケーションは **fcpushstk** インターフェースを使用して、障害情報を FFDC エラー・スタックに記録します。ただし、この情報を記録する前に、アプリケーションは情報をさまざまなロケーションから収集する必要があります。また、この情報の取得によって、アプリケーションの全体のパフォーマンスに影響がでる可能性があります。*FFDC* エラー・スタック環境 が確立または継承されていない場合は、アプリケーションはこの情報を収集する必要はありません。パフォーマンスに対するこの影響を回避するために、アプリケーションは **fcteststk** を実行して、*FFDC* エラー・スタック環境 が使用可能かどうかを調べ、使用可

能な場合は、障害情報の収集を開始することができます。 *FFDC* エラー・スタック環境 が存在しなければ、アプリケーションはこの情報の収集を回避することができます。

fclogerr FFDC インターフェースを使用するプロセスは、 FFDC エラー・スタックが *FFDC* 環境 によって使用されているかどうかにかかわりなく、*FFDC* 環境 が存在すれば、**fclogerr** を使用することができます。 **fclogerr** を使用する場合は、FFDC エラー・スタックが確保されているかどうかにかかわりなく、障害情報が必ず AIX エラー・ログおよび BSD システム・ログに記録されます。**fclogerr** インターフェースを使用して情報を記録するアプリケーションは、FFDC エラー・スタックが使用中かどうかにかかわりなく、常に 障害情報を収集し、記録する必要があります。

フラグ

項目	説明
-h	このコマンドの使用方法に関するメッセージを表示します。これ以上の処理は実行されません。
-q	このコマンドからの、FFDC 環境が確立されたかどうかを説明する出力を抑制します。コマンド・ユーザーは、FFDC 環境がこの処理のために確立されたかどうかを判別するために、コマンドの終了状況を検査する必要があります。

パラメーター

FFDC_Failure_ID

レポートを開始する障害の FFDC 障害 ID を指定します。 **fcreport** は、この障害についての障害情報、およびこのレポートによって関連する障害としてリストされるすべての障害についての障害情報を入手しようとします。このコマンドには FFDC 障害 ID を 1 つしか指定できません。

セキュリティー

fcreport は **rsh** を使用して、リモート・ノードにある障害レポートを入手します。ユーザーは、これらのリモート・ノードに対して **rsh** コマンドを実行できる十分な特権をもっていなければなりません。ユーザーがこの許可をもっていない場合、**fcreport** は、ローカル・ノード上の関連する障害のリストをトレースすることしかできません。

終了状況

0 FFDC エラー・スタック環境が存在します。

2 ヘルプ情報が表示され、処理が終了しました。

12 処理が実行されませんでした。無効なオプションが指定されました。

15 この時点で FFDC エラー・スタック環境はクライアントによって確立または継承されていません。

17 FFDC エラー・スタック環境が破壊されていると判断されました。この環境は使用不能とみなされます。

例

アプリケーションに対して FFDC エラー・スタック環境が存在するかどうかを検査するには、次のようにします。

```
fcteststk -q
if (($? == 0))
then
    # Collect failure information
```

```
            :
            :
        # Use fcpushstk to record failure info
            :
            :
    fi
```

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

502 ページの『fcinit コマンド』

497 ページの『fcdecode コマンド』

---

fddistat コマンド

目的

FDDI デバイス・ドライバーとデバイスの統計情報を表示します。

構文

fddistat [**-r -t**] *Device_Name*

説明

fddistat コマンドでは、指定した FDDI デバイス・ドライバーによって収集された統計が表示されます。フラグを指定しない場合は、デバイス・ドライバーの統計のみが表示されます。このコマンドは、**-v** フラグを指定して **netstat** コマンドを実行するときにも呼び出されます。**netstat** コマンドでは、**fddistat** コマンドのフラグは発行されません。

無効な *Device_Name* を指定すると、**fddistat** コマンドによって、デバイスに接続できなかったことを通知するエラー・メッセージが生成されます。

フラグ

項目	説明
-r	すべての統計を初期値にリセットします。このフラグは、特権ユーザーだけが発行できます。
-t	一部のデバイス・ドライバーでデバッグ・トレースを切り替えます。

パラメーター

項目	説明
Device_Name	FDDI デバイス名 (例えば、**fddi0**)

統計フィールド

> 注**:** アダプターによっては、特定の統計情報がサポートされないものがあります。サポートされない統計フィールドの値は、常に 0 です。

以下は、**fddistat** コマンドの出力に表示される統計フィールドとその説明です。

タイトル・フィールド

項目	説明
Elapsed Time	最後に統計がリセットされてから経過した実時間が表示されます。ハードウェア・エラーが検出されると、エラー・リカバリー中に統計の一部がデバイス・ドライバーによってリセットされる場合があるので、このような状況が発生した場合は、2 つの統計の時間差を反映するために、出力の途中でもう 1 つの Elapsed Time が表示されます。

送信統計フィールド

項目	説明
Packets	デバイスによって正常に送信されたパケットの数。
Bytes	デバイスによって正常に送信されたバイト数。
Interrupt	ドライバーがアダプターから受け取った送信割り込みの数。
Transmit Errors	このデバイス上で発生した出力エラーの数。このフィールドは、ハードウェア/ネットワーク・エラーにより失敗した送信回数のカウンターです。
Packets Dropped	送信のためにデバイス・ドライバーが受け入れ、デバイスには (何らかの理由で) 提供されなかったパケットの数。
Max Packets on S/W Transmit Queue	ソフトウェア送信キューに入れられたことのある発信パケットの最大数。
S/W Transmit Queue Overflow	ソフトウェア送信キューからあふれた発信パケットの数。
Current S/W+H/W Transmit Queue Length	ソフトウェア送信キューまたはハードウェア送信キュー上で保留されている発信パケットの数。
Broadcast Packets	エラーなしで送信されたブロードキャスト・パケットの数。
Multicast Packets	エラーなしで送信されたマルチキャスト・パケットの数。

受信統計フィールド

項目	説明
Packets	デバイスによって正常に受信されたパケットの数。
Bytes	デバイスによって正常に受信されたバイト数。
Interrupts	ドライバーがアダプターから受け取った受信割り込みの数。
Receive Errors	このデバイス上で発生した入力エラーの数。このフィールドは、ハードウェア/ネットワーク・エラーにより失敗した受信回数のカウンターです。
Packets Dropped	デバイス・ドライバーがこのデバイスから受信し、ネットワーク・デマルチプレクサーには (何らかの理由で) 提供されなかったパケットの数。
Bad Packets	デバイス・ドライバーによって受信 (つまり保管) された正しくないパケットの数。
Broadcast Packets	エラーなしで受信されたブロードキャスト・パケットの数。
Multicast Packets	エラーなしで受信されたマルチキャスト・パケットの数。

一般統計フィールド

項目	説明
No mbuf Errors	mbufs をデバイス・ドライバーが利用できなかった回数。これは、通常、インバウンド・パケットを処理するためにドライバーで mbuf バッファーを取得しなければならない場合、受信操作時に発生します。要求されたサイズの mbuf プールが空の場合には、パケットが廃棄されます。 **netstat -m** コマンドを使用すると、これを確認できます。
SMT Error Word	アダプターの SMT エラー状況。
SMT Event Word	アダプターの SMT イベント状況。
Connection Policy Violation	アダプターのリングへの接続状況。
Port Event	アダプターのポート状況。
Set Count	現在の設定カウント値。
Adapter Check Code	アダプターの最新のアダプター・チェック状況。
Purged Frames	使用可能なディスクリプターの欠如によりアダプターから削除された受信フレーム。
ECM State Machine	エンティティー調整管理状態のマシン。
PCM State Machine: Port A	1 次アダプター用物理接続管理の状態マシン。
PCM State Machine: Port B	2 次アダプター用物理接続管理の状態マシン。
CFM State Machine: Port A	1 次アダプター用構成管理の状態マシン。
CFM State Machine: Port B	2 次アダプター用構成管理の状態マシン。
CF State Machine	全体構成の状態マシン。
MAC CFM State Machine	MAC 用構成管理の状態マシン。
RMT State Machine	リング管理の状態マシン。
Driver Flags	現在オンになっているデバイス・ドライバーの内部状況フラグ。

例

fddi0 のデバイス・ドライバー統計を表示するには、次のように入力します。

```
fddistat fddi0
```

このコマンドにより次の出力が生成されます。

```
------------------------------------------------------------
FDDI STATISTICS (fddi0) :
Elapsed Time: 0 days 0 hours 1 minutes 3 seconds

Transmit Statistics:                       Receive Statistics:
--------------------                       -------------------
Packets: 100                               Packets: 100
Bytes: 113800                              Bytes: 104700
Interrupts: 100                            Interrupts: 100
Transmit Errors: 0                         Receive Errors: 0
Packets Dropped: 0                         Packets Dropped: 0
Max Packets on S/W Transmit Queue: 0       Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 0                       Broadcast Packets: 0
Multicast Packets: 0                       Multicast Packets: 0

General Statistics:
-------------------
No mbuf Errors: 0
SMT Error Word: 00040080                   SMT Event Word: 000004a0
Connection Policy Violation: 0000          Port Event: 0000
Set Count Hi: 0000                         Set Count Lo: 0003
Adapter Check Code: 0000                   Purged Frames: 0

ECM State Machine:        IN
PCM State Machine Port A: CONNECT
PCM State Machine Port B: ACTIVE
CFM State Machine Port A: ISOLATED
```

```
CFM State Machine Port B: CONCATENATED
CF State Machine:         C_WRAP_B
MAC CFM State Machine:    PRIMARY
RMT State Machine:        RING_OP

Driver Flags: Up Broadcast Running
        Simplex DualAttachStation
```

関連資料:

522 ページの『fcstat コマンド』

402 ページの『entstat コマンド』

関連情報:

netstat コマンド

tokstat コマンド

---

fdformat コマンド

目的

fdformat コマンドは、ディスケットをフォーマットします。

構文

fdformat [*Device*] [**-h**]

説明

重要**:** ディスケット、または読み取り/書き込み光ディスクをフォーマットすると、格納されている既存のデータがすべて破棄されます。

fdformat コマンドは、**-h** フラグを指定しないと、ディスケットを低密度用にフォーマットします。

まったく新しい空のディスケットは、フォーマットしないと使用できません。

fdformat コマンドは、ディスケットまたは読み取り/書き込み光ディスクをフォーマットする前に、確認を求めるプロンプトを表示します。 これで操作を確実に終了させることができます。

フラグ

項目	説明
-h	強制的に高密度でフォーマットします。このフラグは **fdformat** コマンドと一緒に使用しなければなりません。

パラメーター

項目	説明
Device	フォーマットしたいディスケットが入っているデバイスを指定します。デフォルトは、デバイス **/dev/rfd0** (ドライブ 0) です。

例

fdformat コマンドを使ってディスケットを高密度でフォーマットするには、次のように入力します。

```
fdformat -h
```

ファイル

項目	説明
/usr/sbin/fdformat	**fdformat** コマンドが入っています。
/dev/rfd*	デバイス・パラメーターを指定します。
/dev/fd*	デバイス・パラメーターを指定します。
/dev/romd*	デバイス・パラメーターを指定します。
/dev/omd*	デバイス・パラメーターを指定します。

関連資料:

584 ページの『flcopy コマンド』

595 ページの『format コマンド』

関連情報:

fd コマンド

---

fdpr コマンド

項目	説明
-analyse_asm_csects	アセンブリーで書かれた csects を分析します (使用するときは、-1 および -3 フェーズの両方で指定する必要があります)。
-extra_safe_analysis	手書きアセンブリー・コードを含む非標準 csects の分析を試みません (使用するときは、-1 および -3 フェーズの両方で指定する必要があります)。
-ignore_info	コンパイル時に **-qfdpr** オプションを使用して作成された **.info** セクションを無視します (使用するときは、-1 および -3 フェーズの両方で指定する必要があります)。
-align *bytes*	頻繁に実行されるコードを、指定されたバイト数に基づいて位置合わせをします。これは、コード・プリフェッチ・バッファー率を改善することが目的です。このオプションを省略すると、**fdpr** コマンドは、変数のデフォルト・バイト数を使用してコードの位置合わせを行います。
-lr_opt	頻繁に実行されるプロシージャー内のリンク・レジスターの保管と復元を除去します。
-bt_csect_anchor_removal	コードに含まれるブランチ・テーブルの使用に関連したロード命令を除去します。
-dead_code_removal	到達不能コードを除去します。
-selective_inline	単一ドメイン呼び出しサイトから頻繁に呼び出される関数について、選択的インライン化を行います。
-sid_fac *percent*	選択的インライン最適化における支配ファクターとしてパーセンテージを設定します。可能な範囲は、50 から 100 です (**-selective_inline** フラグを使用する場合のみ適用可能)。
-inline_small_funcs *size*	バイト数で指定されたサイズより小さいか等しいすべての関数をインライン化します。
-inline_hot_funcs *percent*	指定されたパーセンテージより実行頻度が大きいか等しいすべての関数をインライン化します。入力できるパーセント範囲は、0 から 100 です。
-inline	**-selective_inline** を使用して **-inline_small_funcs 12** を実行します。
-hco_resched	頻繁に実行されるコードの命令を、まれにしか実行されないコード域に再配置します (可能な場合)。
-dcbt_opt	**dcbt** 命令を挿入してデータ・キャッシュのパフォーマンスを改善します。

項目	説明
-killed_regs	頻繁に実行される関数呼び出しの後に終了させられる (上書きされる) レジスターの保管および復元を除去します。
-tb	リオーダーされたコードでトレースバック・テーブルを再構成させます。 **-tb** オプションを省略すると、Try & Catch 機構を使用する C++ アプリケーションの場合は、トレースバック・テーブルが自動的に復元されます。
-pc	リオーダーされたコードで CSECT の境界を保持します。
-pp	リオーダーされたコードで関数の境界を保持します。
-RD	静的データのリオーダーを実行します。
-dpnf *factor*	データ配置正規化要因 (0 または 1)。0 を設定すると静的変数がそのサイズに無関係にリオーダーされ、1 を設定するとサイズの小さい変数から配置されます (**-RD** フラグを使用する場合のみ適用可能)。
-dpht *threshold*	データ配置頻度しきい値 (0 または 1)。0 を設定すると静的変数が制御フローに基づいて大規模グループにリオーダーされ、1 を設定すると変数はそれぞれのアクセス頻度に基づいて非常に小規模のグループにリオーダーされます (**-RD** フラグを使用する場合のみ適用可能)。
-build_dcg	拡張データ・リオーダー用の DCG (データ接続性グラフ) を作成します (**-RD** フラグ使用する場合のみ適用可能)。
-tocload	tocload 最適化を実行します。
-reduce_toc *removal_factor*	除去ファクター (0 または 1) に基づいて、TOC エントリーの除去を実行します。0 を設定すると非アクセス TOC エントリーのみ除去され、1 を設定するとすべての非エクスポート TOC エントリーが除去されます。
-strip	出力ファイルをストリップします (作成されたものがある場合)。
-ptrgl_opt	レジスターを介した間接呼び出し命令を直接ジャンプと置き換えることによって、最適化します。
-no_ptrgl_r11	_ptrgl CSECT の R11 ロード命令の除去を実行しません (デフォルトで **-ptrgl_r11** 最適化が適用されます)。
-O	ブランチ予測ビットの設定、ブランチの折り畳み、および NOOP 命令の除去を行ってコードのリオーダーを実行します。デフォルトで **-O** フラグが適用されます。
-O2	アグレッシブでないすべての最適化フラグをオンにします。
-O3	アグレッシブなすべての最適化フラグをオンにします。
-O4	アグレッシブなすべての最適化フラグをオンにします。

目的

ユーザー・レベルのリンク後アプリケーション・プログラムの実行時間と実メモリーの使用率を改善するためのパフォーマンス調整ユーティリティー。

構文

最も一般的な使用方法**:**

fdpr -p *ProgramFile* **-x** *WorkloadCommand*

詳細な使用方法**:**

fdpr -p *ProgramFile* **[-M** *SegNum* **] [-fd** *Fdesc* **] [-o** *OutputFile* **] [-armember** *ArchiveMemberList* **] [** OptimizationFlags **] [-map] [-disasm] [-disasm_data] [-disasm_bss] [-profcount] [-quiet] [-v] [-1 | -2 | -3 | -12 | -23 | -123] [-x** *WorkloadCommand* **]**

最適化フラグ

[-tb] [-pc] [-pp] [-O][-O2] [-O3] [-O4] [-selective_inline] [-sid_fac *percent*] [**-inline_small_funcs** *size*] [**-inline_hot_funcs** *percent*] [**-hco_resched] [-killed_regs] [-lr_opt] [-align** *bytes*] [**-RD] [-dpnf** *factor*] [**-dpht** *threshold*] [**-build_dcg] [-tocload] [-ptrgl_opt] [-no_ptrgl_r11] [-dcbt_opt] [-ignore_info] [-dead_code_removal] [-bt_csect_anchor_removal] [**

-strip] [**-analyse_asm_csects**] [**-extra_safe_analysis**] [**-inline**] [**-reduce_toc** *removal_factor*]

説明

fdpr コマンド (Feedback Directed Program Restructuring) は、ユーザー・レベルのアプリケーション・プログラムの実行時間と実メモリーの使用率の改良を促進できるパフォーマンス調整ユーティリティーです。 **fdpr** プログラムは、一般的なワークロードでプログラムを使用している間にそのプログラムの動作に関する情報を収集することによって、プログラムの実行可能イメージを最適化してから、そのワークロードに対して最適化された新しいバージョンのプログラムを作成します。**fdpr** によって新しく作成されるプログラムは、一般に以前よりも実行が速く、実メモリーの使用量が減少します。

重要**:** **fdpr** コマンドは、複数プログラムの振る舞いが期待通りでないことの原因になっている可能性があるその中の 1 つのプログラムに対して、拡張最適化手法を適用します。このツールを使用して最適化したプログラムは、十分な注意を払って使用する必要があります。期待通りに機能するかどうかを調べるためには、少なくとも元のプログラムの検査に使用したのと同じ方法で綿密に再検査する必要があります。最適化されたプログラムはサポートされません。

fdpr コマンドは、最適化実行可能プログラムを次の 3 つのフェーズで作成します。

- フェーズ 1 (**-1** フラグ): インスツルメント実行可能プログラムと空のテンプレート・プロファイル・ファイルを作成します。
- フェーズ 2 (**-2** フラグ): インスツルメント・プログラムを実行してプロファイル・データを更新します。
- フェーズ 3 (**-3** フラグ): 最適化実行可能プログラム・ファイルを生成します。

この 3 つのフェーズは、別々に実行したり、部分的にまたはすべてを組み合わせて実行できますが、順番どおりに (つまり、**-1**、**-2**、 **-3** の順、または **-12**、 **-3** の順に) 実行する必要があります。デフォルトの設定では、3 つのフェーズすべてが実行されます。

注**:** 一般に、フェーズ 1 で作成され、フェーズ 2 で実行されるインスツルメント化実行可能プログラムは、実行速度が元のプログラムの数分の 1 遅くなります。インスツルメント化プログラムに必要な実行時間が増えるため、実行可能プログラムは実行時間を最小限に抑えるような方法で起動する必要がありますが、必要なコード領域は従来通り完全に実行されます。また、**fdpr** コマンドのユーザーは、プログラムの時間に左右される部分をできる限りなくすようにしてください。

フラグ

項目	説明
-1、**-2**、**-3**	実行するフェーズを指定します。デフォルトの設定は 3 つのフェーズすべて (**-123**) です。各フェーズを別個に実行するときには、連続するフェーズが必要な中間ファイルにアクセスできるように、**-s** フラグを使用しなければなりません。各フェーズは順番通りに (例えば、**-1**、 **-2**、**-3** の順、または **-1**、**-23** の順で)、実行されなければなりません。 **-2** のフラグは、**-x** の起動フラグと一緒に使用する必要があります。
-M *SegNum*	プロファイルを作成するために共用メモリーのマップ先を指定します。デフォルトの設定は **0x30000000** です。最適化されるプログラム、または **-x** フラグで起動されるワークロード・コマンド文字列によって、競合する共用メモリー・アドレスが使用される場合は、代替共用メモリー・アドレスを指定します。一般的な代替値は **0x40000000**、**0x50000000**、... から **0xC0000000** です。
-fd *Fdesc*	上記の共用メモリー領域にマップするプロファイル・ファイルに対し、どのファイル・ディスクリプター番号を使用するかを指定します。*Fdesc* のデフォルトは 1999 に設定されています。
-o *OutFile*	最適化プログラムからの出力ファイルの名前を指定します。デフォルトは、*program*.**fdpr** です。
-p *ProgramFile*	最適化の対象となる、実行可能プログラム・ファイルの名前、あるいは、共用オブジェクト/実行可能プログラムが入っている共用オブジェクト・ファイル、または共用ライブラリーの名前が入ります。このプログラムは、ストリップされていない実行可能プログラムでなければなりません。

項目	説明
-armember *ArchiveMemberList*	**-p** フラグで指定した共用アーカイブ・ファイル内の、最適化するアーカイブ・メンバーのリスト。 **-armember** が指定されていない場合は、アーカイブ・ファイルのすべてのメンバーが最適化されます。
-map	それぞれの古いアドレス -> 新規アドレスをもつ基本ブロックと静的変数のマップを、サフィックスが **.mapper** のファイルに出力します。
-disasm	出力の最適化インスツルメント・プログラムの逆アセンブルされたテキスト・セクションを、サフィックスが **.dis_text** のファイルに出力します。
-disasm_data	出力の最適化インスツルメント・プログラムの逆アセンブルされたデータ・セクションを、サフィックスが **.dis_data** のファイルに出力します。
-disasm_bss	出力の最適化インスツルメント・プログラムの逆アセンブルされた bss セクションを、サフィックスが **.dis_bss** のファイルに出力します。
-profcount	プロファイル作成カウンターを、サフィックスが **.ncounts** のファイルに出力します。
-quiet	quiet 出力モード。
-v	詳細出力。
-x *WorkloadCommand*	インスツルメント化プログラムの起動に使用するコマンドを指定します。 **-x** フラグの後のすべての引数は、起動のために使用されます。したがって、**-x** フラグはコマンド・ラインの最後になければなりません。 **-2** フラグを使用するときは、**-x** フラグが必要になります。

最適化フラグ

最適化

fdpr コマンドは、ブランチ予測ビットの設定、ブランチの折り畳み、コードの位置合わせ、および冗長 NOOP 命令の除去という最適化とともに、最大限レベルのコード・リオーダー最適化をデフォルトで実行します。 **-pc** フラグは、CSECT の境界を維持しながらコード全体をリオーダーするため、デフォルトのコード・リオーダーに比べ、パフォーマンスの改善度が小さくなる場合があります。同様に **-pp** フラグも、プロシージャーの境界を維持しながらコード全体をリオーダーします。

実行可能プログラム・ファイル全体への追加最適化も、上記の最適化フラグを使用して行うことができます。

-qfdpr IBM xl コンパイラー・フラグを使用して作成した実行可能プログラムには、**fdpr** がリオーダー・プログラムを生成するのに役立つ情報が含まれています。**-qfdpr** オプションを使用してコンパイルされなかったモジュールは、記号テーブル内のコンパイラー・シグニチャーに基づいてリオーダーされます。

リオーダーするプログラムを作成するときに、静的リンクを使用すると、さらにパフォーマンスを高めることができます。**fdpr** プログラムでは指定した実行可能プログラム内の命令がリオーダーされるだけなので、プログラムによって呼び出される動的にリンクされた共用ライブラリー・ルーチンは最適化されません。これらのライブラリー・ルーチンを実行可能プログラムに静的にリンクすると、プログラム内の命令とプログラムに使用されるすべてのライブラリー・ルーチンを最適化することができます。静的にリンクされるプログラムを構築する方法には、このほかにも長所と短所があります。

出力ファイル

fdpr コマンドによって作成されるすべてのファイルは、 **-x** フラグで指定したワークロード・コマンドを実行して作成されるファイルを除き、現行ディレクトリーに保管されます。最適化プロセスの間に、オリジナル・プログラムは名称変更されて保存され、最終フェーズが完了するとオリジナル・プログラム名に復元されるだけです。

プログラムを実行するためのスクリプトによって、プログラムの実行前に作業ディレクトリーが変更されることがあるので、**fdpr** コマンドによって作成されるプロファイル・ファイルでは、現行ディレクトリーの完全な名前が明示的に使用されます。

fdpr コマンドによって作成または使用されるファイルは次のとおりです。

項目	説明
program	最適化されるストリップされていない実行可能プログラムの名前。
program.**save**	オリジナルの実行可能プログラムの保管バージョン。
program.**nprof**	プロファイルのファイル名。
program.**instr**	プログラムのインスツルメント化バージョンの名前。
program.**fdpr**	最適化された実行可能出力ファイルのデフォルト名。
program.**instr.dis_text**	インスツルメント・フェーズ後に **-disasm** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**fdpr.dis_text**	最適化フェーズ後に **-disasm** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**instr.dis_data**	インスツルメント・フェーズ後に **-disasm_data** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**fdpr.dis_data**	最適化フェーズ後に **-disasm_data** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**instr.dis_bss**	インスツルメント・フェーズ後に **-disasm_bss** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**fdpr.dis_bss**	最適化フェーズ後に **-disasm_bss** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**instr.mapper**	インスツルメント・フェーズ後に **-map** フラグによって作成されたデフォルトの ASCII フォーマットのマッピング・ファイル。
program.**fdpr.mapper**	最適化フェーズ後に **-map** フラグによって作成されたデフォルトの ASCII フォーマットのマッピング・ファイル。
program.**ncounts**	**-profcount** フラグによって作成されたデフォルトの ASCII フォーマットのプロファイル・カウンター・ファイル。

拡張デバッグ機能

最適化プログラムにある程度のデバッグ機能を設けるために、**FDPR** は **.text** セクションで行われた変更を反映するように記号テーブルを更新します。

FDPR のリオーダーの際に再配置された記号のアドレスを指定する、記号テーブル内の入力フィールドは、**.text** セクションの新アドレスを指すように変更されます。

さらに、リオーダーの間に機能もしくはファイルが分割される場合、**FDPR** は、分割された機能もしくはファイルの新規の部分ごとに、シンボル・テーブル内に新規エントリーを作成します。同じ機能のこのような新規部分には、以下の命名規則に従って、記号テーブル内の新しいシンボル名が付けられます。

```
<original function name>__fdpr_<function's part number>
```

コードのリオーダー後、すべての新規エントリーに接尾辞として `__fdpr_` 文字列が付けられます。

例: 機能「main」は、もともとは記号テーブルに次のエントリーを持っていました。

```
 [Index] m   Value       Scn     Aux   Sclass    Type    Name
  [456]  m  0x00000230    2       1     0x02    0x0000   .main
```

コードのリオーダー後に機能 `main` が 3 分割されると、この機能は、次のように、記号テーブル内に 3 エントリー (分割された部分ごとに 1 つずつ) を持つことになります。

```
 [Index] m   Value       Scn     Aux   Sclass    Type    Name
  [456]  m  0x00000304    2       1     0x02    0x0000   .main
 [1447]  m  0x00003328    2       1     0x02    0x0000   .main__fdpr_1
 [1453]  m  0x000033b4    2       1     0x02    0x0000   .main__fdpr_2
```

例

次の各例は、**fdpr** コマンドの典型的な使用方法です。

1. この例では、3 つのフェーズをすべて実行できます。この例で、 test1 はストリップされていない実行可能プログラムであり、 test2 は、test1 を起動するシェル・スクリプトです。現在の作業ディレクトリーは **/tmp/fdpr** です。

   ```
   test2 script file:

   # code to exercise test1
   test1 -expand 100 -root $PATH file.jpg -quit
   # the end of test2
   ```

   fdpr コマンドを (デフォルトの最適化を使用して) 次のように実行します。

   ```
   fdpr -p test1 -x test2
   ```

   これにより、新たにリオーダーされた実行可能ファイル **test1.fdpr** が生成されます。
2. 各フェーズを一度に 1 つずつ実行するには、**fdpr** のフェーズ 1 を実行します。

   ```
   fdpr -1 -p test1
   ```

   このコマンド文字列は、test1.instr という名前のインスツルメント化バージョンと、空のテンプレート・プロファイル・ファイル test1.nprof を作成します。

   フェーズ 2 を実行するには、次のように入力します。

   ```
   fdpr -2 -p test1 -x test2
   ```

   このコマンド文字列によって、test1 のインスツルメント化バージョンを実行してプロファイル・データを収集するスクリプト・ファイル test2 が実行されます。

   フェーズ 3 を実行するには、次のように入力します。

   ```
   fdpr -3 -p test1
   ```

   これにより、もう一度、新たにリオーダーされた実行可能ファイル **test1.fdpr** が生成されます。
3. 最初の 2 つのフェーズに続けてフェーズ 3 を実行するには、次のように、フェーズ 1 と 2 を実行します。

   ```
   fdpr -12 -p test1 -x test2
   ```

   フェーズ 3 を最適化レベル 3 で実行します。

   ```
   fdpr -3 -O3 -p test1
   ```
4. **fdpr** 最適化プログラムを実行中にエラーが発生した場合は、**dbx** コマンドを次のように使用して、どのプロシージャーでエラーが発生したかを判別できます。

   ```
   dbx program.fdpr
   ```

   これにより、次のような出力が作成されます。

   ```
   Type 'help' for help.
   reading symbolic information ...warning: no source compiled with -g

   [using memory image in core]

   Segmentation fault in proc_d at 0x10000634
   0x10000634 (???) 98640000        stb   r3,0x0(r4)
   (dbx)
   ```

スタック・トレースバックは、プログラムが現行位置にどのように到着したかを判別するために使用するもので、次のように作成します。

```
(dbx) where
```

これにより、次のような出力が生成されます。

```
proc_d(0x0) at 0x10000634
proc_c(0x0) at 0x10000604
proc_b(0x0) at 0x100005d0
proc_a(0x0) at 0x1000059c
main(0x2, 0x2ff7fba4) at 0x1000055c
(dbx)
```

5. また、**dbx** サブコマンド **stepi** を使用すると、リオーダー後の実行可能プログラムの命令を次のように 1 つのステップでたどることができます。

```
(dbx) stepi
```

これにより、次のような出力が生成されます。

```
stopped in proc_d at 0x1000061c
0x1000061c (???) 9421ffc0       stwu   r1,-64(r1)
(dbx)
```

この例で、**dbx** は、プログラムが、リオーダーされたテキスト・セクションのアドレス `0x1000061c` のルーチン `proc_d` で中止したことを示しています。

実装上の固有な条件

ソフトウェアの製品/オプション: *AIX Performance Aide/ Local Performance Analysis & Control Commands*。

標準準拠: なし。

ファイル

項目	説明
/usr/bin/fdpr	**fdpr** コマンドが入っています。
program	最適化されるストリップされていない実行可能プログラムの名前。
program.**save**	オリジナルの実行可能プログラムの保管バージョン。
program.**nprof**	プロファイルのファイル名。
program.**instr**	プログラムのインスツルメント化バージョンの名前。
program.**fdpr**	最適化された実行可能出力ファイルのデフォルト名。
program.**instr.dis_text**	インスツルメント・フェーズ後に **-disasm** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**fdpr.dis_text**	最適化フェーズ後に **-disasm** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**instr.dis_data**	インスツルメント・フェーズ後に **-disasm_data** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**fdpr.dis_data**	最適化フェーズ後に **-disasm_data** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**instr.dis_bss**	インスツルメント・フェーズ後に **-disasm_bss** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**fdpr.dis_bss**	最適化フェーズ後に **-disasm_bss** フラグによって作成されたデフォルトの ASCII フォーマットの逆アセンブリー・ファイル。
program.**instr.mapper**	インスツルメント・フェーズ後に **-map** フラグによって作成されたデフォルトの ASCII フォーマットのマッピング・ファイル。
program.**fdpr.mapper**	最適化フェーズ後に **-map** フラグによって作成されたデフォルトの ASCII フォーマットのマッピング・ファイル。

項目	説明
program.**ncounts**	**-profcount** フラグによって作成されたデフォルトの ASCII フォーマットのプロファイル・カウンター・ファイル。

関連資料:

10 ページの『dbx コマンド』

関連情報:

fdpr プログラムによる実行可能プログラムの再構造化

---

fencevsd コマンド

目的

ノードまたはノード・グループで実行されているアプリケーションが、仮想共用ディスクまたは仮想共用ディスク・グループからアクセスされることを回避します。

構文

fencevsd {**-a** | **-v** *vsd_name_list*} **-n** *node_list*

説明

実際にはノードの操作が可能であるときに、ある環境下では、システムはノードが機能を停止したと判断することがあり、リカバリー手順が開始されます。ただし、同じアプリケーションを実行中の他のノードとの通信は遮断されます。この場合、リカバリーが完了するまで問題のあるノードから仮想共用ディスクに要求を出すことはできません。ただし、アプリケーションを実行中の他のノードは、問題のあるノードを操作可能とみなします。**fencevsd** コマンドは、問題ノードが仮想共用ディスクへの要求を満たせないようにします。

このコマンドは、リカバリー可能仮想共用ディスク・サブシステムが稼働している RSCT ピア・ドメイン内の任意のノードから実行することができます。

フラグ

-a すべての仮想共用ディスクを指定します。

-v *vsd_name_list*
コンマで区切られた、1 つ以上の仮想共用ディスク名を指定します。

-n *node_list*
コンマで区切られた、1 つ以上のノード番号を指定します。

パラメーター

logical_volume_name
仮想共用ディスクとして指定したい論理ボリュームの名前。この論理ボリュームは、指示されたグローバル・ボリューム・グループ上に常駐している必要があります。名前の長さは 15 文字以内でなければなりません。

global_group_name
vsdvg コマンドにより前回定義された、仮想共用ディスクとして指定したい、グローバルにアクセス可能なボリューム・グループ・グループの名前。名前の長さは 31 文字以内でなければなりません。

vsd_name

新規仮想共用ディスクに対して固有の名前を指定します。この名前は、RSCT ピア・ドメイン内で固有でなければなりません。また、今後の命名において競合の危険性を避けるために、クラスター全体に渡っても固有でなければなりません。推奨する命名規則は **vsdnn***gvg_name* です。名前の長さは 31 文字以内でなければなりません。

注**:** 他のデバイスで既に *vsd_name* が指定されている場合、**cfgvsd** コマンドは、仮想共用ディスクに対して失敗します。このエラーが起こるということは、その名前用に作成した特別の装置ファイルが、論理ボリュームのような他のデバイス・タイプを表している同じ名前のファイルをオーバーレイしたり、破棄したりしないことを裏付けています。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

制限

このコマンドは、アクティブなリカバリー可能仮想共用ディスク・サブシステムを持つ、ピア・ドメイン内のノードから発行する必要があります。

例

ノード 5 から仮想共用ディスク vsd1 および vsd2 を隔離するには、次のように入力します。

```
fencevsd -v vsd1,vsd2 -n 5
```

位置

/opt/rsct/vsd/bin/fencevsd

関連情報:

lsvsd コマンド

unfencevsd コマンド

---

ff コマンド

目的

ファイルシステムのファイル名と統計情報をリストします。

構文

ff [**-a** *Number*] [**-c** *Number*] [**-I**] [**-l**] [**-m** *Number*] [**-n** *File*] [**-o** *Options*] [**-p** *Prefix*] [**-s**] [**-u**] [**-V** *VFSName*] [**-i** *I-Number* [**,***I-Number ...*]] [*FileSystem* | *DeviceName*]

説明

ff コマンドは、*FileSystem* パラメーターで指定されたファイルシステムの i ノード番号を読み取って、i ノードに関する情報を標準出力に書き込みます。 *FileSystem* はファイルシステムと見なされるので、**/etc/filesystems** ファイルで参照されて、フラグで指定されたファイルの i ノード・データを保存します。

ff コマンドからの出力は、ユーザーがフラグを使用して要求できる他のファイル情報と、要求した個別の i ノード番号のパス名から構成されています。出力は i ノード番号順に表示され、すべてのフィールド間

にタブが付きます。**ff** コマンドで作成されるデフォルト行には、パス名フィールドと i ノード番号フィールドがあります。すべてのフラグを使用可能にすると、パス名、i ノード番号、サイズ、ユーザー ID (UID) が出力フィールドに設定されます。

Number パラメーターは、日数を指定する 10 進数です。このパラメーターの前には、+ または - (正符号または負符号) が付きます。したがって、+3 は 4 日以上を意味し、-3 は 2 日以内を意味し、3 は 3 日を意味します。1 日は 24 時間制で定義されます。

-l フラグを指定しない限り、 **ff** コマンドは、複数のリンクを持つ i ノードに対して、使用できるパス名の中から 1 つだけ表示します。 **-l** フラグを指定すると、 **ff** コマンドはすべてのリンクを表示します。

フラグ

項目	説明
-a *Number*	*Number* パラメーターで指定した日数以内に、ファイルがアクセスされていれば、そのファイルを表示します。
-c *Number*	*Number* パラメーターで指定した日数以内に、ファイルの i ノード番号が変更されていれば、そのファイルを表示します。
-i *I-Number*	*I-Number* パラメーターで指定された i ノード番号に対応するファイルを表示します。リストする i ノード番号は、コンマで区切らなければなりません。
-I	(このフラグは i の大文字) 各パス名の後ろの i ノードを表示しません。

項目	説明
-l	(このフラグは L の小文字) 複数のリンクを持つファイルのパス名のリストを追加表示します。
-m *Number*	*Number* パラメーターで指定した日数以内に、ファイルが変更されていれば、そのファイルを表示します。
-n *File*	*File* パラメーターで指定されたファイルより後に、ファイルが変更されていれば、そのファイルを表示します。
-o *Options*	仮想ファイルシステムに対してインプリメンテーション固有のオプションのコンマ区切りリストを指定します。 次のオプションは、拡張ジャーナル・ファイルシステム (JFS2) に固有です。 **-o snapshot=***snapName* **ff** コマンドの対象となる内部スナップショットの名前を指定します。このスナップショットを持つファイルシステムは、マウント済みでなければなりません。
-p *Prefix*	*Prefix* パラメーターで指定されたプレフィックスを、各パス名に追加します。デフォルトのプレフィックスは .(ドット) です。
-s	各パス名の後ろに、ファイル・サイズをバイト数で書きます。
-u	各パス名の後ろに、オーナーのログイン名を書きます。
-V *VFSName*	**ff** コマンドに、ファイルシステムのタイプが *VFSName* であることを前提とし、 **/etc/filesystems** ファイルの値を指定変更するよう指示します。

セキュリティー

RBAC ユーザーおよび Trusted AIX ユーザーへの注意: このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 指定されたファイルシステム内のすべてのファイルのパス名を表示するには、次のように入力します。

   ```
   ff     -I    /dev/hd0
   ```

   これによって、/dev/hd0 デバイス上のファイルのパス名が表示されます。 **-I** フラグを指定していない場合は、 **ff** コマンドは各ファイルの i ノード番号を表示します。

2. 最近変更されたファイルを表示するには、次のように入力します。

   ```
   ff    -m    -2    -u    /dev/hd0
   ```

   この入力で、最後の 2 日以内 (-m -2) に変更された、 /dev/hd0 デバイス上の各ファイルの、パス名、i ノード番号、およびオーナーのユーザー名 (-u フラグ) が表示されます。
3. 最近使用されていない ファイルを表示するには、次のように入力します。

   ```
   ff    -a    +30    /dev/hd0
   ```

   このコマンドにより、最後にアクセスされてから 31 日以上たっている (-a +30)、各ファイルのパス名と i ノードが表示されます。
4. 特定の i ノード番号のパス名を見つけるには、次のように入力します。

   ```
   ff    -l    -i    451,76    /dev/hd0
   ```

   これにより、451 と 76 の i ノード番号と関連のある、すべてのパス名 (-l) が表示されます。

ファイル

項目	説明
/etc/vfs	仮想ファイルシステムのタイプに関する記述が入っています。
/etc/filesystems	既知のファイルシステムをリストし、その特性を定義します。

関連資料:

568 ページの『find コマンド』

関連情報:

ncheck コマンド

ファイルシステム

---

fg コマンド

目的

ジョブをフォアグラウンドで実行します。

構文

fg [*JobID*]

説明

ジョブ制御が使用可能になっている場合 (「オペレーティング・システムおよびデバイスの管理」の『Korn シェルまたは POSIX シェルにおけるジョブ制御』のセクションを参照)、**fg** コマンドは、現在の環境内のバックグラウンド・ジョブをフォアグラウンドに移動します。フォアグラウンドで実行する特定のジョブを示すには、*JobID* パラメーターを使用します。このパラメーターを指定していない場合は、**fg** コマンドによって、最後に延期されたジョブ、バックグラウンドにあるジョブ、バックグラウンド・ジョブとして実行されるジョブのいずれかが使用されます。

JobID パラメーターは、プロセス ID 番号の場合もあります。または、次の記号の組み合わせのいずれか 1 つを指定することもできます。

項目	説明
%*Number*	ジョブ番号別にジョブを参照します。
%*String*	指定文字列で名前が始まるジョブを参照します。
%?*String*	指定文字列が名前に含まれているジョブを参照します。
%+ または **%%**	現行ジョブを参照します。
%-	直前のジョブを参照します。

fg コマンドを指定して、ジョブをフォアグラウンドに移動させると、現在のシェル環境で認識されるジョブのリストからそのジョブのプロセス ID が除去されます。

/usr/bin/fg コマンドは、このコマンド固有の実行環境で動作させると機能しません。これは、この環境では操作に適したジョブがないためです。したがって、**fg** コマンドは Korn シェル (POSIX シェル) の正規組み込みコマンドとして組み込まれています。

終了状況

次の終了値が戻されます。

項目	説明
0	正常終了。
>0	エラーが発生しました。

ジョブ制御が不可能な場合は、**fg** コマンドはエラーを出して終了し、ジョブはフォアグラウンドに移動しません。

例

jobs -l コマンドの出力に、バックグラウンドで実行される次のジョブが示された場合を考えてみます。

```
[1] + 16477RunningSleep 100 &
```

この場合、プロセス ID を使用し、次のように入力して `sleep 100 &` コマンドをフォアグラウンドで実行します。

```
fg 16477
```

画面には次のように表示されます。

```
sleep
```

ファイル

項目	説明
/usr/bin/ksh	Korn シェルの **fg** 組み込みコマンドが入っています。
/usr/bin/fg	**fg** コマンドが入っています。

関連情報:

bg コマンド

csh コマンド

wait コマンド

Korn シェルまたは POSIX シェルにおけるジョブ制御

fgrep コマンド

目的

ファイルを検索してリテラル文字列を探します。

構文

fgrep [**-h**] [**-i**] [**-s**] [**-u**] [**-v**] [**-w**] [**-x**] [**-y**] [[**-b**] [**-n**] | [**-c** | **-l** | **-q**]] [**-p** *Separator*] {*Pattern* | **-e** *Pattern* | **-f** *StringFile*} [*File...*]

説明

fgrep コマンドは、*File* パラメーター (デフォルトでは標準入力) で指定された入力ファイルを検索してパターンと一致する行を探します。**fgrep** コマンドは、特に固定長文字列の *Pattern* パラメーターを検索します。**fgrep** コマンドは、 *File* パラメーターに複数のファイルを指定した場合、一致する行が含まれているファイルを表示します。

fgrep コマンドは、式と一致するパターンを検索するのではなく、文字列を検索するという点で、**grep** および **egrep** コマンドとは異なります。**fgrep** コマンドは、高速でコンパクトなアルゴリズムを使用します。 **fgrep** コマンドでは、$、*、[、|、(、)、および ¥ 文字を文字どおりに解釈します。これらの文字 ($、*、[、|、(、)、および ¥) は、**grep** コマンドおよび **egrep** コマンド内で解釈されるので、正規表現の一部としては解釈されません。これらの文字は、シェルに対して特別な意味があるため、ストリング全体を単一引用符で囲む ('...') 必要があります。ファイルを指定しなければ、**fgrep** コマンドでは、標準入力を前提とします。通常、検索された各行は標準出力にコピーされます。入力ファイルが複数の場合は、ファイル名は検索された各行の前に出力されます。

注**:**

1. **fgrep** コマンドは、 **-F** フラグを指定した **grep** コマンドと、同じ機能を実行しますが、エラー・メッセージと使用方法メッセージ、および **-s** フラグの機能が異なります。
2. 各行は 2048 バイト以内に制限されています。
3. 現在、パラグラフ (**-p** フラグの指定) の長さは、5000 文字に限定されています。
4. 予測できない結果を生ずるので、**grep** コマンドをスペシャル・ファイル上で実行するのは避けてください。
5. 入力行には、NULL 文字を使用しないでください。
6. 入力ファイルは、改行文字で終らなければなりません。
7. 複数のフラグを同時に指定できますが、フラグの中には他のフラグを指定変更してしまうものがあります。例えば、**-l** と **-n** を同時に指定すると、ファイル名だけが標準出力に書き出されます。

フラグ

フラグ	説明
-b	各行の前に、その行が見つかったブロック番号を付けます。このフラグを使用すると、ディスク・ブロック番号をコンテキストで見つけるときに便利です。 **-b** フラグは、標準入力またはパイプからの入力と一緒には使用できません。
-c	一致した行の数のみを表示します。
-e *Pattern*	パターンを指定します。それは、単純パターンと同じ働きをしますが、パターンが **-** (ハイフン) で始まる場合に便利です。
-f *StringFile*	文字列を含むファイルを指定します。 **注:** 検索パターンを含んでいるファイルを入力に使用して、**fgrep** (または **grep -F**) コマンドのパフォーマンスを強化するには、**fgrep** コマンドを実行する前に、**ENABLE_FGREP_AC** 環境変数をエクスポートします。例えば、この環境変数をエクスポートするには、次のコマンドを実行することができます。 `export ENABLE_FGREP_AC=""`
-h	複数のファイルを処理したときに、ファイル名を抑止します。
-i	比較を行うときに、大文字と小文字の区別を無視します。
-l	一致している行のファイルの名前を表示します (1 回)。各ファイルの名前は改行文字で区切られます。
-n	各行の前に、ファイル内におけるその行の相対行番号を付けます。
-p *Separator*	一致する行を含んでいるパラグラフ全体を表示します。パラグラフは、*Separator* パラメーターで指定された、パラグラフ・セパレーターで区切られます。パラグラフ・セパレーターは検索パターンと同じ書式のパターンです。パラグラフ・セパレーターを含んでいる行は、セパレーターとしてのみ使用され、出力には含まれません。デフォルトのパラグラフ・セパレーターはブランク行となります。
-q	行の一致に関係なく、標準出力への書き出しをすべて抑止します。入力行を選択すると、終了して状況 0 を戻します。
-s	エラー・メッセージだけを表示します。それは、状況を検査するのに便利です。
-u	出力をバッファーしません。
-v	指定されたパターンに一致する行を除くすべての行を表示します。
-w	1 つのワードを検索します。
-x	余分な文字を含まず、パターンと完全に一致する行を表示します。
-y	比較を行うときに、大文字と小文字の区別を無視します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	一致が見つかりました。
1	一致が見つかりませんでした。
>1	構文エラーが見つかったか、(一致が見つかったとしても) ファイルにアクセスできませんでした。

例

1. いくつかのファイル内の単純文字列を検索するには、次のように入力します。

   ```
   fgrep strcpy *.c
   ```

   これによって、`.c` 文字ストリングで終わる名前を持つ現行ディレクトリーにあるすべてのファイル内のストリング `strcpy` が検索されます。
2. パターンと一致する行数をカウントするには、次のように入力します。

   ```
   fgrep -c "{" pgm.c
   fgrep -c "}" pgm.c
   ```

   それによって、左中括弧と右中括弧を含んでいる `pgm.c` 内の行の数が表示されます。

C プログラムで、1 行に複数の { (左中括弧) または } (右中括弧) を付けなかった場合、および左右の中括弧が対応している場合に、正しい条件を満たしていれば 2 つの数字が同じになります。この数字が同じでない場合には、次のように入力すれば、中括弧を含んでいる行を、ファイル内の順序で表示することができます。

```
egrep {¥|} pgm.c
```

3. パターンを含むファイル名を表示するには、次のように入力します。

```
fgrep -l strcpy *.c
```

それによって、.c で終わる現行ディレクトリー内のファイルが検索されて、文字列 strcpy を含んだファイルの名前が表示されます。

ファイル

ファイル	説明
/usr/bin/fgrep	**fgrep** コマンドが入っています。
/bin/fgrep	**fgrep** コマンドへのシンボリック・リンク。

関連資料:

358 ページの『egrep コマンド』

739 ページの『grep コマンド』

関連情報:

sed コマンド

入出力リダイレクト

---

file コマンド

目的

ファイル・タイプを判別します。

構文

ファイル・タイプを分類する

file [**-m** *MagicFile*] [**-d**] [**-h**] [**-i**] [**-M** *MagicFile*] [**-f** *FileList*] [*File...*]

マジック・ファイル内で形式エラーを検査する

file -c [**-m** *MagicFile*]

説明

file コマンドは、*File* パラメーターまたは *FileList* 変数で指定したファイルを読み取り、各ファイルで一連のテストを行い、各ファイルをタイプ別に分類します。次に、そのファイル・タイプを標準出力に書き出します。ファイル・タイプは、通常ファイル、ディレクトリー、FIFO (名前付きパイプ)、ブロック・スペシャル、文字スペシャル、シンボリック・リンク、ソケットのいずれかです。

- 通常ファイルで長さがゼロの場合は、空のファイルと見なされます。
- シンボリック・リンクの場合は、デフォルトでは、そのシンボリック・リンクが参照するファイルがそのリンクの後に続きます。

ファイルが ASCII フォーマットと見なされる場合は、 **file** コマンドで最初の 1024 バイトを検査して、そのファイルのタイプを判別します。ファイルが ASCII フォーマットでなければ、**file** コマンドは拡張文字が入っているテキスト・ファイルとバイナリー・データ・ファイルとを区別します。

File パラメーターで実行可能ファイルまたはオブジェクト・モジュール・ファイルを指定している場合、バージョン番号が 0 より大きいと、 **file** コマンドはバージョン・スタンプを表示します。 **ld** コマンドのセクションに、**a.out** ファイルの使用方法が説明されています。

言語環境が C プログラミング言語である場合、**file** コマンドは、**/etc/magic** ファイルを使用してマジック・ナンバーのような番号を持つファイル、つまり、タイプを示す数値または文字列定数が入っているファイルを識別します。

ただし、言語環境が C プログラミング言語以外の言語である場合、**file** コマンドは、**/usr/lib/nls/msg/<*language_env*.>/magic.cat** ファイルを使用してマジック・ナンバーを持つファイルを識別します。

そのファイルが存在しない場合、読み取りができない場合、あるいはファイル状況がその時点で判別できなかった場合、それは、終了状況に影響を与えるエラーとは見なされません。出力から、ファイルは処理されたがタイプの判別はできなかったということが分かります。

-i フラグを使用すると、 *file* を指定した各オペランドの識別に次の形式が使用されます。

```
"%s: %s¥n", file, type
```

type の値は指定されていません。ただし、 POSIX ロケールでは、*file* が次の表にリストされているいずれかのタイプとして識別される場合は、 *type* には対応する文字列が含まれます (しかし、対応する文字列に限定されません)。文字列内に示される各スペースは、正確に 1 つの *space* になります。

表 5. ファイル・ユーティリティー出力文字列

file が次に該当する場合	*type* には次の文字列が含まれます
ディレクトリー	ディレクトリー
FIFO	fifo
ソケット	socket
ブロック・スペシャル	block special
文字スペシャル	character special
実行可能バイナリー	executable
空の通常ファイル	empty
シンボリック・リンク	symbolic link to
ar アーカイブ・ライブラリー	アーカイブ
拡張 *cpio* フォーマット	cpio archive
拡張 *tar* フォーマット	tar archive
シェル・スクリプト	commands text
C 言語ソース	c program text
FORTRAN ソース	fortran program text

file がシンボリック・リンクとして識別される場合は、次の代替出力形式が使用されます。

```
"%s: %s %s¥n", file, type, contents of link"
```

file オペランドで指定されたファイルが存在しないかまたは読み取ることができない場合は、文字列 `cannot open` が *type* フィールドの一部として組み込まれますが、これは終了状況に影響を及ぼすエラーとは見なされません。 *file* オペランドで指定されたファイルのタイプが判別できない場合は、文字列 `data`

が、*type* フィールドの一部として組み込まれますが、これは終了状況に影響を及ぼすエラーとは見なされません。

フラグ

項目	説明
-c	指定されたマジック・ファイル (デフォルトでは **/etc/magic** ファイル) に形式エラーがないかどうか検査します。この妥当性検査は、通常は行われません。このフラグを使うと、ファイルの分類は行われません。
-d	任意のデフォルトのシステム・テストをファイルに適用します。
-f *FileList*	指定されたファイル・リストを読み取ります。ファイルは、1 行に 1 つずつ表示し、先行スペースまたは後書きスペースが入っていてはなりません。
-h	シンボリック・リンクが検出されると、そのファイルをシンボリック・リンクとして識別します。 **-h** フラグが指定されないで、かつ *file* が、存在しないファイルを参照するシンボリック・リンクである場合は、 *file* は、**-h** フラグが指定されている場合と同様に、そのファイルをシンボリック・リンクとして識別します。
-i	ファイルが通常ファイルの場合は、ファイルのタイプの分類は行われません。ただし、 547 ページの『説明』で指定されているファイルは識別されます。
-m *MagicFile*	マジック・ファイル (デフォルトでは **/etc/magic** ファイル) のファイル名を指定します。
-M *MagicFile*	ファイルを分類するためにファイルに適用されるテストを含むファイルの名前を指定します。デフォルトのシステム・テストは適用されません。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. ファイルに入っている情報のタイプを表示するには、次のように入力します。

   ```
   file myfile
   ```

   このコマンドにより、*myfile* のタイプのファイル (ディレクトリー、データ、ASCII テキスト、C プログラム・ソース、アーカイブなど) が表示されます。
2. ファイル名のリストに登録されている、それぞれのファイルのタイプを表示するには、次のように入力します。

   ```
   file -f filenames
   ```

   このコマンドにより、`filenames` リストに登録されている各ファイルのタイプが表示されます。ファイル名は、それぞれ 1 行につき 1 つでなければなりません。

注**: file** コマンドからカスタマイズされたメッセージを取得するには、 **-m** オプションを指定した個別のマジック・ファイルを使用します。読み取り専用の **/etc/magic** ファイルを編集することはお勧めできません。

ファイル

項目	説明
/usr/bin/file	**file** コマンドが入っています。
/etc/magic	ファイル・タイプ・データベースが入っています。

関連資料:

568 ページの『find コマンド』

関連情報:

ld コマンド

Files コマンド

ファイルとディレクトリーのアクセス・モード

---

filemon コマンド

目的

ファイルシステムのパフォーマンスをモニターして、論理ファイル、および仮想記憶セグメント、論理ボリューム、および物理ボリュームのために I/O アクティビティーを報告します。

構文

filemon [**-d**] [**-i** *Trace_File* **-n** *Gensyms_File*] [**-o** *File*] [**-O** ***Levels***] [**-w**] [**-I** ***count:interval***] [**-P**] [**-T** *n*] [**-u**] [**-v**] [**-@** [*WparList* | **ALL**] [**-r** *RootString*] [**-A** **-x** *User_Command*]

説明

filemon コマンドでは、ファイルシステムと入出力システム・イベントのトレースがモニターされ、その期間のファイルと入出力アクセスのパフォーマンスについて報告されます。

通常モードでは、**filemon** コマンドはバックグラウンドで実行され、1 つ以上のアプリケーション・プログラムまたはシステム・コマンドを実行し、モニターします。**filemon** コマンドは自動的に始動され、プログラムのファイルシステムと入出力イベントのトレースをリアルタイムでモニターします。デフォルトでは、トレースは即時に開始します。オプションでユーザーが **trcon** コマンドを発行してからトレースを開始するようにも設定できます。**filemon** コマンドの実行中に **trcoff** コマンドと **trcon** コマンドを実行して、モニターを必要に応じてオフにしたりオンにしたりできます。トレースが **trcstop** コマンドによって中止されると、**filemon** コマンドは入出力アクティビティー・レポートを生成して、終了します。

また、**filemon** コマンドでは、トレース機能によって以前に記録されたトレース・ファイルを処理することもできます。ファイルと入出力アクティビティーの報告は、そのファイルに記録されているイベントに基づきます。

アプリケーションのファイルシステムのパフォーマンスについて詳細な情報を提供するために、**filemon** コマンドでは次の 4 つのレベルでファイルと入出力アクティビティーをモニターします。

項目	説明
Logical file system (論理ファイルシステム)	**filemon** コマンドでは、論理ファイル上の論理入出力操作をモニターします。モニターされる操作は **read**、 **write**、 **open**、および **lseek** の各システム呼び出しです。これらのシステム呼び出しによって、ファイルが既にメモリーにバッファーされているかどうかに応じて、実際の物理入出力が行われるかどうかが決まります。入出力統計はファイル単位で保持されます。非同期入出力システム呼び出しは **filemon** コマンドでモニターされないため、**filemon** 論理ファイル・レポートには非同期入出力 (AIO) 要求は組み込まれません。
Virtual memory system (仮想記憶システム)	**filemon** コマンドでは、セグメントとディスク上のそれらのイメージ間の物理入出力操作 (つまりページング) をモニターします。入出力統計はセグメント単位で保持されます。
Logical volumes (論理ボリューム)	**filemon** コマンドでは、論理ボリューム上の入出力操作をモニターします。入出力統計は論理ボリューム単位で保持されます。
Physical volumes (物理ボリューム)	**filemon** コマンドでは、物理ボリューム上の入出力操作をモニターします。このレベルでは、物理リソースの使用状況が取得されます。入出力統計は物理ボリューム単位で保持されます。

コマンド・ライン・フラグで指定したとおりに、これらの 4 つのレベルを組み合わせてモニターすることができます。デフォルトでは、**filemon** コマンドによって、仮想記憶、論理ボリューム、物理ボリュームのレベルでの入出力操作だけがモニターされます。これらのレベルはすべて、実際のディスク入出力に対する要求に関連するものです。

filemon コマンドは、ファイル、論理ボリューム、および物理ボリュームについての稼働状況レポートも生成します。この稼働状況レポートは、**–O** hot オプションを使用して生成できます。このレポートは、自動オフライン・モードと手動オフライン・モードでのみサポートされています。稼働状況レポートには、ファイル、論理ボリューム、および物理ボリュームに関する入出力操作統計が含まれます。このレポートを使用すると、ファイル/論理ボリュームの稼働状況に基づくさまざまな入出力特性を使用して、どのファイルまたは論理ボリュームをどのドライブに移動するかを決定できます。この稼働状況は、読み取り操作の数、読み取り操作ごとに読み取られたバイトの平均数、読み取りシーケンスの数、およびシーケンスの平均の長さに基づいて判別されます。

filemon コマンドにより、その報告が標準出力または指定ファイルに書かれます。デフォルトでは、この報告には、モニターしている各レベルの入出力アクティビティーの要約が含まれます。詳細レポートの印刷は、**-O detailed** フラグが使用可能になっている場合のみ行われます。要約と詳細レポートの内容については、『報告』のセクションで説明します。

注**:**

1. **filemon** コマンドによって生成されるレポートは、長くなる場合があります。したがって、出力ファイルにレポートを書き込むには、**-o** オプションを使用します。アプリケーションによって物理デバイスがオープンされ、直接アクセスされるときには、完全な 512 バイト単位のブロックの読み取りおよび書き込みだけが、報告に反映されます。デバイス・コマンドを実行してデバイスの状況を読み取るためにデバイス・ドライバーによって使用される「短い」読み書きは、無視されます。 CD-ROM は、ハード・ファイルとは異なり、同心円上の「トラック」も「シリンダー」もありません。(ら旋上のトラックが 1 つあるだけです。)したがって、CD-ROM の場合は、シリンダーに関して距離の統計情報を報告することは不可能です。
2. **-u** フラグは、**trace** デーモンの始動前にオープンされたファイルに関するレポートを生成する場合に使用されます。このデータには役に立つものもありますが、そのほとんどはデーモンやほかの無関係のアクティビティーに該当します。このバックグラウンド情報は、大きなシステムの場合、特に多くなります。 **/unix** ファイルと実行中のカーネルが同じでない場合、カーネル・アドレスが正しくないので、

filemon コマンドは終了します。**filemon** コマンドをシェル・スクリプト内から使用する場合は、**filemon** 出力ファイルの内容を表示するまでに少し遅延が発生します。**filemon** コマンドでこのレポートを作成するのに、数秒かかることがあります。

3. 入出力処理プログラムで相対パスを指定してファイルの読み取りまたは書き込みを行う場合、**filemon** コマンドはこの相対パスを **filemon** コマンドが実行されたディレクトリーとして解釈します。このような場合、入出力アクティビティーの報告には、そのファイルのボリューム情報 (i ノード) が正しく表示されない可能性があります。この問題を回避するには、すべての入出力処理プログラムで完全なパスを使用してください。
4. **filemon** コマンドは、ソリッド・ステート・ドライブ (SSD) ディスクをサポートしていません。このため、**filemon** コマンドは SSD ディスクの統計情報を報告しません。

システム・トレース機能

filemon コマンドでは、システム・トレース機能を使って、未加工の入出力パフォーマンス・データを獲得します。現在、トレース機能では 1 つの出力ストリームしかサポートしていません。したがって、**filemon** またはトレース・プロセスは 1 つずつ実行されます。別の **filemon** またはトレース・プロセスが既に実行中の場合、**filemon** コマンドは次のようなメッセージで応答します。

```
/dev/systrace: Device busy
```

入出力が集中するアプリケーションをモニターする場合、**filemon** コマンドでは、トレース・イベントをリアルタイムに (トレース・イベントが生成されるのと同じ速さで) 処理できないことがあります。この場合は、次のようなエラー・メッセージが表示されます。

```
Trace kernel buffers overflowed, N missed entries
```

このメッセージは stderr に表示されて、トレース・バッファーがいっぱいになったときに失われたトレース・イベントの数が示されます。**filemon** コマンドでは、入出力アクティビティーのモニターが継続されますが、報告の正確性はある程度失われます。オーバーフローを避ける 1 つの方法として、ファイルと入出力サブシステムをモニターするレベルを少なくする方法があります。生成されるトレース・イベントの数は、モニターするレベルの数に比例するためです。さらに、**-T** オプションを使ってトレース・バッファー・サイズを増やし、オーバーフローが発生する前にトレース・イベントのバーストを調整する方法もあります。ただし、トレース・バッファー・サイズを増やすことは、固定されたメモリーが増えるので、入出力やページングの動作に影響を及ぼすことがあるので注意してください。

メモリー制約のある (必要なメモリーが足りない) 環境では、**-P** オプションを使用すると、リアルタイムの **filemon** プロセスのテキストとデータ・ページをメモリーに固定できるので、ページがスワップアウトされません。 **-P** オプションを使用しないと、**filemon** プロセスがスワップアウトされるので、**filemon** コマンドの進行が遅れ、十分な速度でトレース・イベントを処理できないことがあります。このような状況が発生すると、トレース・バッファーは前述のようにオーバーフローしてしまいます。その結果、このプロセスを固定すると、アプリケーションで使用できるメモリーは減少します (**filemon** コマンドは大きいプログラムではありませんが、そのプロセス・イメージによって使用されるメモリーは 500KB までになります)。

-i *Trace_File* フラグと **-n** *Gensyms_File* フラグを指定すると、**trace** コマンドで作成されたトレース・データ・ファイルを **filemon** コマンドによってオフライン処理することができます。これらのフラグのいずれか一方が存在している場合には、もう一方も指定しなければなりません。これらのフラグは、リモート・マシンからトレース・ファイルのポストプロセスをする場合や、ある時点でトレース・データ収集を実行し、別の時点でそのポストプロセスをする場合に役に立ちます。システム負荷が大きく、**filemon** がトレース・フックを見逃している場合にも、これらのフラグは役に立ちます。これらのフラグは、自動オフライン・モードに使用することができます。

-r *RootString* フラグを使用する場合、**-i** *Trace_File* フラグと **-n** *Gensyms_File* フラグの使用は推奨されません。 **-r** *RootString* フラグはオフライン処理での使用のほかに、**-A** フラグと一緒に使用して自動オフライン・モードを使用可能にすることができます。

gensyms ファイル (ファイルシステム情報が含まれています) は、トレース元のマシンから使用する必要があります。また、システム・トレース・ファイルが作成されるのとほぼ同時に、 **gensyms** を実行して、両者のシステム構成が同じになるようにするのも賢明な方法です。

filemon に関係のあるトレース・フックは、**trace** コマンドにより収集され、**trace -j** フラグで指定されます。関係のあるトレース・フックが表示されるのは、**-v** フラグを使用して **filemon** を開始した場合です。次に **-F** オプションを指定した **gensyms** コマンドが実行され、出力が *Gensyms_File* に保管されて、**filemon** の追加情報を収集します。gensyms コマンドに **-F** オプションを指定すると、物理ボリュームおよび論理ボリュームに関するデバイス情報が収集されます。このオプションは、オフラインの **filemon** によって使用される仮想ファイルシステム情報を収集する場合にも使用されます。その後で、このファイルと *Gensyms_File* は、 **filemon** に提供されることもあります。

報告

filemon コマンドによって生成される各報告には、日付、マシン ID、モニター期間 (秒) を示すヘッダーがあります。また、モニター期間中のプロセッサー使用率も報告されます。

次に、モニターする各ファイルシステム・レベルごとに要約報告が生成されます。デフォルトでは、論理ファイルと仮想記憶についての報告は、転送されたデータの総量を調べて最もアクティブであった 20 個のファイルとセグメントに制限されます。 **-v** フラグを指定すると、すべてのファイルとセグメントについてのアクティビティーが報告されます。各ファイル、セグメント、ボリュームについて、それぞれ 1 行に報告されます。 4 つの要約報告の各行の列を、次のリストで説明します。

最もアクティブなファイルの報告

列	説明
#MBS	ファイルとの間で転送された合計メガバイト数。このフィールドに従って、行は降順にソートされます。
#opns	測定期間中にファイルがオープンされた回数。
#rds	ファイルに対して行われた読み取りシステム呼び出しの回数。
#wrs	ファイルに対して行われた書き込みシステム呼び出しの回数。
file	ファイルの名前 (詳細レポートでは絶対パス名)。
volume:inode	ファイルが入っているボリュームの名前とそのファイルの i ノード番号。このフィールドを使って、ファイルと、仮想記憶入出力報告に表示される対応する永続セグメントを関連付けることができます。このフィールドはブランクになっている場合があります (実行時に生成されてから削除された一時ファイルなどの場合)。

最もアクティブなセグメントの報告

項目	説明
列	説明
#MBS	セグメントに転送された、またはファイルから転送されたデータの合計数 (MB 単位)。このフィールドに従って、行は降順にソートされます。
#rpgs	ディスクからセグメントに読み取られた (つまり、ページインされた) 4096 バイト単位のページ数。
#wpgs	セグメントからディスクに書き込まれた (ページアウトされた) 4096 バイト単位のページ数。
segid	セグメントの内部 ID。
segtype	セグメントのタイプ: 作業セグメント、永続セグメント (ローカル・ファイル)、クライアント・セグメント (リモート・ファイル)、ページ・テーブル・セグメント、システム・セグメント、ファイルシステムのデータが入っている特殊永続セグメント (ログ、ルート・ディレクトリー、.inode、.inodemap、.inodex、.inodexmap、.indirect、.diskmap) のいずれか。

項目	説明
volume:inode	永続セグメントの場合、関連ファイルを含むボリュームの名前とそのファイルの i ノード番号。このフィールドは、永続セグメントをファイル入出力報告で表示される永続セグメントに対応するファイルに関連付けるために、使用します。永続セグメントでない場合、このフィールドはブランクになります。 **注:** 仮想メモリー分析ツール **svmon** を使用すると、セグメント ID (segid) を次のように指定して、セグメントに関する詳細を表示することができます。 `svmon -S <segid>`

最もアクティブな論理ボリュームの報告

項目	説明
列	説明
util	ボリュームの使用状況 (使用率)。このフィールドに従って、行は降順にソートされます。
#rblk	ボリュームから読み取られた 512 バイト単位のブロックの数。
#wblk	ボリュームに書き込まれた 512 バイト単位のブロックの数。
KB/sec	合計の転送スループット (K バイト/秒)。
volume	ボリュームの名前。
description	ボリュームの内容: ファイルシステム名または論理ボリューム・タイプ (paging、jfslog、boot、または sysdump)。また、ファイルシステムがフラグメント化されているか、圧縮されているかについても示されます。

最もアクティブな物理ボリュームの報告

項目	説明
列	説明
util	ボリュームの使用状況 (使用率)。このフィールドに従って、行は降順にソートされます。
#rblk	ボリュームから読み取られた 512 バイト単位のブロックの数。
#wblk	ボリュームに書き込まれた 512 バイト単位のブロックの数。
KB/sec	合計のボリューム・スループット (K バイト/秒)。
volume	ボリュームの名前。
description	ボリュームのタイプ。例えば、120MB disk、355MB SCSI、または CDROM SCSI など。 **注:** 論理ボリューム入出力要求は、物理ボリューム入出力要求の前に始まり、物理ボリューム入出力要求の後で終了します。そのため、論理ボリュームの総使用状況は、物理ボリュームの総使用状況より多く見えます。

最もアクティブなファイルのプロセスに関する報告

項目	説明
列	説明
#MBS	ファイルに転送された、またはファイルから転送されたデータの合計数 (MB 単位)。このフィールドに従って、行は降順にソートされます。
#opns	測定期間中にファイルがオープンされた回数。
#rds	ファイルに対して行われた読み取りシステム呼び出しの回数。
#wrs	ファイルに対して行われた書き込みシステム呼び出しの回数。
file	ファイルの名前 (詳細レポートでは絶対パス名)。
PID	ファイルをオープンしたプロセスの ID。
Process	ファイルをオープンしたプロセスの名前。
TID	ファイルをオープンしたスレッドの ID。

最もアクティブなファイルのスレッドに関する報告

項目	説明
列	説明
#MBS	ファイルに転送された、またはファイルから転送されたデータの合計数 (MB 単位)。このフィールドに従って、行は降順にソートされます。
#opns	測定期間中にファイルがオープンされた回数。
#rds	ファイルに対して行われた読み取りシステム呼び出しの回数。
#wrs	ファイルに対して行われた書き込みシステム呼び出しの回数。
file	ファイルの名前 (詳細レポートでは絶対パス名)。
TID	ファイルをオープンしたスレッドの ID。
Process	ファイルをオープンしたプロセスの名前。
PID	ファイルをオープンしたプロセスの ID。

最後に、モニターしている各ファイルシステム・レベルごとに詳細レポートが生成されます。デフォルトでは、論理ファイルと仮想記憶についての報告は、転送されたデータの総量を調べて最もアクティブであった 20 個のファイルとセグメントに制限されます。 **-v** フラグを指定すると、すべてのファイルとセグメントについてのアクティビティーが報告されます。各ファイル、セグメント、またはボリュームについて、それぞれ 1 つのエントリーに報告されます。

フィールドのなかには、1 つの値が報告されるものと、たくさんの値の分布を示す統計が報告されるものがあります。例えば、応答時間統計には、モニターされた読み取りまたは書き込みの要求すべてについての統計が保持されます。平均、最小、最大の応答時間、および応答時間の標準偏差が報告されます。標準偏差は、個別の応答時間が平均からかけ離れている度合いを示すために使用されます。抽出した応答時間の約 2/3 が `average - standard deviation` と `average + standard deviation` の間にあります。応答時間の分布が広範囲に広がると、標準偏差は平均応答時間と比較して大きくなります。 4 つの明細報告を、次のリストで説明します。

ファイル統計の詳細レポート

項目	説明
列	説明
FILE	ファイルの名前。可能であれば、絶対パス名が表示されます。
volume	ファイルが入っている論理ボリューム/ファイルシステムの名前。
inode	ファイルシステム内でのファイルの i ノード番号。
opens	モニター中にファイルがオープンされた回数。
total bytes xfrd	ファイルから読み取られた、またはファイルに書き込まれたバイトの合計数。
reads	ファイルに対して行われた読み取りコールの数。
read sizes (bytes)	バイト単位の読み取り転送サイズの統計 (avg/min/max/sdev)。
read times (msec)	ミリ秒単位の読み取り応答時間の統計 (avg/min/max/sdev)。
writes	ファイルに対して行われた書き込みコールの数。
write sizes (bytes)	書き込み転送サイズの統計。
write times (msec)	書き込み応答時間の統計。
seeks	**lseek** サブルーチン呼び出しの回数。

VM セグメント統計の詳細レポート

項目	説明
列	説明
SEGMENT	内部セグメント ID。
segtype	セグメントの内容のタイプ。
segment flags	各種のセグメント属性。
volume	永続セグメントの場合は、対応するファイルが入っている論理ボリュームの名前。
inode	永続セグメントの場合は、対応するファイルの i ノード番号。
reads	セグメントに読み込まれた (つまり、ページインされた) 4096 バイト単位のページ数。
read times (msec)	ミリ秒単位の読み取り応答時間の統計 (avg/min/max/sdev)。
read sequences	読み取りシーケンスの数。シーケンスとは、連続して読み取られた (ページインされた) ページの文字列のことです。読み取りシーケンスの数は、順次アクセスの総計を示します。
read seq. lengths	読み取りシーケンスの長さを記述した統計 (ページ単位)。
writes	セグメントから書き込まれた (つまり、ページアウトされた) ページ数。
write times (msec)	書き込み応答時間の統計。
write sequences	書き込みシーケンスの数。シーケンスとは、連続して書き込まれた (ページアウトされた) ページの文字列のことです。
write seq.lengths	書き込みシーケンスの長さを記述した統計 (ページ単位)。

論理/物理ボリューム統計の詳細レポート

項目	説明
列	説明
VOLUME	ボリュームの名前。
description	ボリュームの記述。(論理ボリュームを扱う場合は内容を記述し、物理ボリュームを扱う場合はタイプを記述する。)
reads	ボリュームに対して行われた読み取り要求の回数。
read sizes (blks)	512 バイト・ブロック単位の、読み取り転送サイズの統計 (avg/min/max/sdev)。
read times (msec)	ミリ秒単位の読み取り応答時間の統計 (avg/min/max/sdev)。
read sequences	読み取りシーケンスの数。シーケンスとは、連続して読み取られた 512 バイト単位のブロックのことで、順次アクセスの総数を示します。
read seq. lengths	読み取りシーケンスの長さを記述した統計 (ブロック単位)。
writes	ボリュームに対して行われた書き込み要求の回数。
write sizes (blks)	書き込み転送サイズの統計。
write times (msec)	書き込み応答時間の統計。
write sequences	書き込みシーケンスの数。シーケンスとは、連続して書き込まれた 512 バイト単位のブロックの文字列のことです。
write seq. lengths	書き込みシーケンスの長さを記述した統計 (ブロック単位)。
seeks	読み取りまたは書き込み要求の前に行われたシークの数。シークを必要とした読み取りと書き込みの総数に対するパーセントで表されます。
seek dist (blks)	シーク距離の統計 (512 バイトのブロック単位)。通常の統計 (avg/min/max/sdev) に加え、初期シーク動作の距離 (ブロック 0 が開始位置と仮定する) が個別に報告されます。このシーク距離の統計は大きくなる場合があるので、ほかの統計を壊さないように、個別に報告されます。
seek dist (cyls)	(ハード・ディスクのみ) シーク距離の統計 (ディスク・シリンダー単位)。
time to next req	ボリュームに対する連続した読み取りまたは書き込み要求間の時間の長さを記述した、ミリ秒単位の統計 (avg/min/max/sdev)。この列は、ボリュームのアクセス率を示しています。
throughput	合計のボリューム・スループット (K バイト/秒)。
utilization	ボリュームが使用中の時間。このフィールドに従って、この報告内のエントリーは降順にソートされます。

プロセスに関する統計の詳細レポート

項目	説明
列	説明
Process Id	ファイルをオープンしたプロセスの ID。
Name	オープンされたファイルの名前 (パスを含む)。
Thread Id	ファイルをオープンしたスレッドの ID。
Total Bytes	読み取られたバイトまたは書き込まれたバイトの合計数。
# of seeks	シークの数。
# of reads	読み取り操作の数。
read errors	読み取りエラーの数。
# of writes	書き込み操作の数。
Bytes Read	読み取られたバイト数。
	min 一度に読み取られた最小バイト数。
	avr 一度に読み取られた平均バイト数。
	max 一度に読み取られた最大バイト数。
Bytes Written	書き込まれたバイト数。
	min 一度に書き込まれた最小バイト数。
	avr 一度に書き込まれた平均バイト数。
	max 一度に書き込まれた最大バイト数。
Read Time	読み取り操作に要する時間。
Write Time	書き込み操作に要する時間。

スレッドに関する統計の詳細レポート

項目	説明
列	説明
Thread Id	ファイルをオープンしたスレッドの ID。
Name	オープンされたファイルの名前 (パスを含む)。
Process Id	ファイルをオープンしたスレッドの ID。
Total Bytes	読み取られたバイトまたは書き込まれたバイトの合計数。
# of seeks	シークの数。
# of reads	読み取り操作の数。
read errors	読み取りエラーの数。
# of writes	書き込み操作の数。
Bytes Read	読み取られたバイト数。
	min 一度に読み取られた最小バイト数。
	avr 一度に読み取られた平均バイト数。
	max 一度に読み取られた最大バイト数。
Bytes Written	書き込まれたバイト数。
	min 一度に書き込まれた最小バイト数。
	avr 一度に書き込まれた平均バイト数。
	max 一度に書き込まれた最大バイト数。
Read Time	読み取り操作に要する時間。
Write Time	書き込み操作に要する時間。

照合報告フォーマット

項目	説明
ID	**process** 読み取りまたは書き込み操作を行ったプロセスの ID。 **thread** 読み取りまたは書き込み操作を行ったスレッドの ID。 **CPU** 読み取りまたは書き込み操作が実行された CPU の ID。
transaction type	トランザクションのタイプ (SCSI や SSA など)。
time	**bstart event** **bstart** イベントが開始した時間。 **iodone event** 入出力操作が完了した時間。 **duration** 入出力操作の合計所要時間。
read/write	操作のタイプ (読み取りまたは書き込み)。
physical block address	物理ブロック・アドレス。
access pattern	アクセス・パターンのタイプ (順次またはランダム)。
physical block size	物理ブロック・サイズ。
volume name or address	**physical** 物理ボリューム名またはアドレス。 **logical** 論理ボリューム名またはアドレス。
Transaction index	トランザクションを識別する固有の ID。
time	**event** イベントが開始した時間。 **extend** イベントが拡張された時間。
ID	**process** トランザクションを実行したプロセスの ID。 **thread** トランザクションを実行したスレッドの ID。 **CPU** トランザクションが実行された CPU の ID。
protocol stage	イベントの破壊を表示します。
name	デバイス、バッファー、またはブロックの名前、あるいはバイト数。
address/count	デバイス、バッファー、またはブロックのアドレス、あるいはバイト数。
access pattern	アクセス・パターンのタイプ (順次またはランダム)。
label	ボリューム・タイプまたは転送フラグ。
values	ボリューム名またはフラグ値。

稼働状況レポート

稼働状況レポートは、情報セクション、集計セクション、および稼働状況レポート・セクションの 3 つのセクションで構成されています。情報セクションには、システム・モデル、使用された **filemon** コマンド、および使用された **trace** コマンドが含まれています。集計セクションには、読み取りまたは書き込み操作の総数、要した時間の合計、読み取りまたは書き込みが行われたデータの合計、および CPU 使用率が含まれています。

ホット・ファイル・レポート

項目	説明
列	説明
Name	ファイルの名前。
Size	ファイルのサイズ。デフォルトの単位は MB です。デフォルトの単位は、**–O** unit オプションで指定された単位によって指定変更されます。
CAP_ACC	アクセスされた容量。この値は、ファイル内のアクセスされた固有データです。デフォルトの単位は MB です。デフォルトの単位は、**–O** unit オプションで指定された単位によって指定変更されます。
IOP/#	アクセスされたデータの単位当たりの入出力操作数。データの単位は、**–O** unit オプションから取得されます。デフォルト値は MB です。この列の値の例として 2560/T、256/G、0.256/M、0.000/K があります。文字 K、M、G、および T は、KB、MB、GB、および TB を表します。
LV	ファイルが属する論理ボリュームの名前。この情報を取得できないと、**"-"** が報告されます。
#ROP	ファイルで発生した読み取り操作の総数。
#WOP	そのファイルで発生した書き込み操作の総数。
B/ROP	読み取り操作ごとに読み取られたバイトの <最小、平均、最大> 数。
B/WOP	書き込み操作ごとに読み取られたバイトの <最小、平均、最大> 数。
RTIME	読み取り操作ごとに要した <最小、平均、最大> 時間 (ミリ秒単位)。
WTIME	書き込み操作ごとに要した <最小、平均、最大> 時間 (ミリ秒単位)。
Seqlen	読み取りシーケンスの <最小、平均、最大> 長。
#Seq	読み取りシーケンスの数。シーケンスとは、一連の 4K ページのことであり、このページが連続して読み取られます(ページインされる)。読み取りシーケンスの数は、順次アクセスの総計を示します。

ホット論理ボリューム・レポート

項目	説明
列	説明
Name	論理ファイルの名前。
Size	論理ボリュームのサイズ。デフォルトの単位は MB です。デフォルトの単位は、–O unit オプションで指定された単位によって指定変更されます。この値を取得できないと、"-" が報告されます。
CAP_ACC	アクセスされた容量。この値は、ファイル内のアクセスされた固有データです。デフォルトの単位は MB です。デフォルトの単位は、**–O** unit オプションで指定された単位によって指定変更されます。
IOP/#	アクセスされたデータの単位当たりの入出力操作数。データの単位は、**–O** unit オプションから取得されます。デフォルト値は MB です。この列の値の例として 2560/T、256/G、0.256/M、0.000/K があります。文字 K、M、G、および T は、それぞれ KB、MB、GB、および TB を表します。
#Files	この論理ボリューム内でアクセスされたファイル数。
#ROP	論理ボリュームで発生した読み取り操作の総数。
#WOP	その論理ボリュームで発生した書き込み操作の総数。
B/ROP	読み取り操作ごとに読み取られたバイトの <最小、平均、最大> 数。
B/WOP	書き込み操作ごとに読み取られたバイトの <最小、平均、最大> 数。
RTIME	読み取り操作ごとに要した <最小、平均、最大> 時間 (ミリ秒単位)。
WTIME	書き込み操作ごとに要した <最小、平均、最大> 時間 (ミリ秒単位)。
Seqlen	読み取りシーケンスの <最小、平均、最大> 長。
#Seq	読み取りシーケンスの数。シーケンスとは、一連の 4K ページのことであり、このページが連続して読み取られます(ページインされる)。読み取りシーケンスの数は、順次アクセスの総計を示します。

ホット物理ボリューム・レポート

項目	説明
列	説明
Name	物理ボリュームの名前。
Size	物理ボリュームのサイズ。デフォルトの単位は MB です。デフォルトの単位は、**–O** unit オプションで指定された単位によって指定変更されます。
CAP_ACC	アクセスされた容量。この値は、ファイル内のアクセスされた固有データです。デフォルトの単位は MB です。デフォルトの単位は、**–O** unit オプションで指定された単位によって指定変更されます。
IOP/#	アクセスされたデータの単位当たりの入出力操作数。データの単位は、**–O** unit オプションから取得されます。デフォルト値は MB です。この列の値の例として 2560/T、256/G、0.256/M、0.000/K があります。文字 K、M、G、および T は、それぞれ KB、MB、GB、および TB を表します。
#ROP	物理ボリュームで発生した読み取り操作の総数。
#WOP	その物理ボリュームで発生した書き込み操作の総数。
B/ROP	読み取り操作ごとに読み取られたバイトの <最小、平均、最大> 数。
B/WOP	書き込み操作ごとに読み取られたバイトの <最小、平均、最大> 数。
RTIME	読み取り操作ごとに要した <最小、平均、最大> 時間 (ミリ秒単位)。
WTIME	書き込み操作ごとに要した <最小、平均、最大> 時間 (ミリ秒単位)。
Seqlen	読み取りシーケンスの <最小、平均、最大> 長。
#Seq	読み取りシーケンスの数。シーケンスとは、連続して読み取られた 512 バイト単位のブロックの文字列のことです。読み取りシーケンスの数は、順次アクセスの総計を示します。

前述の各稼働状況レポートは、ソート・フィールドに基づいて複数回繰り返されます。

種々のソート・フィールドに基づいて、下記のような稼働状況レポートが出力されます。

1. キー・ファクターでソートされた稼働状況レポート
2. CAP_ACC でソートされた稼働状況レポート
3. IOP/# でソートされた稼働状況レポート
4. #ROP でソートされた稼働状況レポート
5. #WOP でソートされた稼働状況レポート
6. RTIME でソートされた稼働状況レポート
7. WTIME でソートされた稼働状況レポート

各レポートは、該当するソート・フィールドの降順にソートされます。

–O hot=r オプションを指定すると、読み取り操作に基づくレポートおよびキー・ファクターに基づくレポートのみが生成されます。すなわち、レポート番号 1、4、および 6 が生成されます。

–O hot=w オプションを指定すると、書き込み操作に基づくレポートおよびキー・ファクターに基づくレポートのみが生成されます。すなわち、レポート番号 1、5、および 7 が生成されます。

キー・ファクターは、次の列値 (**#ROP**、**B/ROP**、**Seqlen**、および **#Seq**) によって判別されます。

フラグ

項目	説明
-i *Trace_File*	リアルタイム・トレース・プロセスからではなく、指定した *Trace_File* から入出力トレース・データを読み取ります。**filemon** 報告では、トレース・ファイルに表されたシステムと期間についての入出力アクティビティーが要約されます。このオプションは推奨されていません。代わりに **-r** *RootString* フラグを使用してください。 報告の正確性を高めるために、トレース・ファイルには、**filemon** コマンドに必要なすべてのフックが入っている必要があります。 **-n** オプションも指定しなければなりません。
-n *Gensyms_File*	オフライン・トレース処理のために *Gensyms_File* を指定します。このファイルは、次のように、**-f** オプションを指定して gensyms コマンドを実行し、出力をファイルにリダイレクトすると作成されます。 `gensyms -F > file` **-i** オプションも指定しなければなりません。 **-n** フラグは廃止されました。代わりに **-r** *RootString* フラグを使用してください。
-o *File*	**stdout** ファイルにではなく、指定した *File* に入出力アクティビティーの報告を書き込みます。
-d	**filemon** コマンドを開始しますが、ユーザーが **trcon** コマンドを実行するまでトレースを開始しません。デフォルトでは、トレースは即時に開始されます。
-T *n*	カーネルのトレース・バッファー・サイズを *n* バイトに設定します。デフォルトのサイズは、CPU ごとに 64 000 バイトです。必要であれば、バッファー・サイズを増やして、イベントのバーストを調整できます。(典型的なイベント・レコード・サイズは 30 バイトです。) **注:** カーネル内のトレース・ドライバーではダブル・バッファリングが使用されます。したがって、実際には、サイズ *n* バイトには 2 つのバッファーが割り当てられます。また、これらのバッファーはメモリーに固定されているので、ページングの対象にはなりません。バッファーを大きくすると、ページングやほかの入出力のパフォーマンスに影響を及ぼす場合があります。
-P	モニター・プロセスをメモリーに固定します。 **-P** フラグを指定すると、モニター期間中、**filemon** コマンドのテキストとデータ・ページがメモリーに固定されます。このフラグは、メモリー制約のある環境で実行するときに、リアルタイム **filemon** プロセスがページアウトされないようにするためのものです。
-v	エクストラ情報を報告に出力します。 **-v** フラグの最も重要な機能は、最もアクティブな論理ファイルとセグメントが最大 20 までという制限がなく、アクセスされたすべての論理ファイルとセグメントが入出力アクティビティーの報告に含まれることです。
-A -x *User_Command*	自動オフライン・モードをオンにします。**-x** フラグを **-A** フラグと一緒に使用する必要があります。この場合、トレースは指定のユーザー・コマンドがその実行を終了するまで収集されます。ユーザー・コマンドの標準的な例は `sleep 10` です。
-r *RootString*	このフラグを **-A** フラグと組み合わせると、**filemon** コマンドはトレース・データを **RootString.trc** ファイルに保管し、**gensyms** ファイルを生成し、それを **RootString.syms** ファイルに保管します。このオプションが **-A** フラグなしで使用可能になっている場合、**filemon** コマンドは **RootString.trc** ファイルと **RootString.syms** ファイルを後処理してオフライン報告を生成します。このオプションを使用する場合、既存の **-n** フラグと **-i** フラグの使用は推奨されません。**filemon** コマンドでは、バイナリー互換性のために **-i** フラグと **-n** フラグを引き続きサポートしています。

項目	説明
-O *Levels*	指定したファイルシステム・レベルだけをモニターします。有効なコンマ区切りのオプションは次のとおりです。 **abbreviated** 簡略フォーマットでトランザクションのリスト (トランザクションごとに 1 行) を生成します (既存の subpar ツールに取って代わります)。このオプションはオフライン・モードでのみサポートされ、他の **-O** オプションと組み合わせることはできません。 **collated** 照合フォーマットでトランザクションのリストを生成します。イベントはトランザクションごとにまとめて収集されます。このオプションはオフライン・モードでのみサポートされ、他の **-O** オプションと組み合わせることはできません。 **detailed** 詳細レポートが統計要約モードとともに生成されます。**abbreviated** オプションまたは **collated** オプションと組み合わせることはできません。 **lf=num** 指定された数だけの論理ファイル・エントリーを表示します。**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。 **vm=num** 指定された数だけの仮想メモリー・エントリーを表示します。**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。 **lv=num** 指定された数だけの論理ボリューム・エントリーを表示します。**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。 **pv=num** 指定された数だけの物理ボリューム・エントリーを表示します。**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。 **hot=r|w** 稼働状況レポートを生成します。**hot=r** が指定されると、読み取り操作のみに基づく稼働状況レポートが生成されます。**hot=w** が指定されると、書き込み操作のみに基づく稼働状況レポートが生成されます。 **sz=num** 稼働状況レポート内で報告される、アクセスされたファイルの最大サイズを指定します。この値の単位は、**–O** unit オプションによって指定されます。デフォルトの単位は MB です。**unit={KB|MB|GB|TB} sz** オプションで使用される単位と、稼働状況レポートの **CAP_ACC** フィールドおよび Size フィールドで使用される単位を指定します。 **th=num** 指定された数だけのスレッド統計エントリーを表示します。**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。 **pr=num** 指定された数だけのプロセス統計エントリーを表示します。**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。 **all=num** **lf=num**、**vm=num**、**lv=num**、**pv=num**、**ts=num** を設定し、**lf**、**vm**、**lv**、**pv**、**th**、および **pr** のオプションの既存値を上書きします。このオプションは、**-O abbreviated** フラグまたは **-O collated** フラグと組み合わせることはできません。*num* 引数が指定されない場合は、すべてのエントリーが表示されます。これがデフォルトのオプションです。

項目	説明
	vm、**lv**、および **pv** レベルは、**filemon -O** コマンドを **-@** フラグなしのグローバル WPAR で実行する場合はデフォルトで暗黙です。**lf** レベルは、**filemon -O** コマンドを WPAR で実行するか、**-@** フラグを使用する場合はデフォルトで暗黙です。 *num* 引数が指定されない場合は、デフォルトでそのセクションのすべてのエントリーが表示されます。*num* 引数は、**abbreviated** フォーマットと **collated** フォーマットではサポートされておらず、統計要約でのみサポートされています。**-O detailed** フラグが指定されている場合は、オンラインとオフラインの両方のモードの詳細レポートを含む統計要約フォーマットの報告になります。**filemon** コマンドの操作のデフォルト・モードは、要約および詳細の統計報告から要約のみの統計報告に変更されています。 **filemon** コマンドがオプションなしで呼び出されたり、**-O** フラグと **lf**、**vm**、**lv**、**pv**、**pr**、**th**、または **all** オプションの組み合わせで呼び出された場合は、**-O detailed** フラグが指定されていない限り、要約報告のみが表示されます。
-u	**trace** デーモンの始動前にオープンされたファイルについて報告します。ファイル名の代わりに、プロセス ID (PID) とファイル・ディスクリプター (FD) が使用されます。 **注:** PID と FD は再使用可能なので、異なるファイルが同じ名前フィールドで報告される場合があります。
-w	稼働状況レポートを幅広フォーマットで印刷します。このオプションは、**–O** hot オプションが指定されている場合にのみ有効です。
-I *count:interval*	マルチスナップショット・トレースに使用される回数と間隔を指定します。このオプションを指定すると、**count** 回のトレース・スナップショットが収集されます。この場合、あるスナップショットから次のスナップショットの間隔は **interval** 秒で指定します。このオプションは、**–O** hot オプションが指定された、自動オフライン・モードでのみ有効です。
-@ [*WparList* | **ALL**]	レポートは、引数で渡された WPAR のリストに制限されます。

例

1. ファイルシステムの仮想記憶、論理ボリューム、物理ボリュームの各レベルで物理入出力アクティビティーをモニターするには、次のように入力します。

   ```
   filemon
   ```

   filemon コマンドによって自動的にシステム・トレースが開始され、コマンド自体はバックグラウンドに入ります。このコマンドの後に、この時点で実行するアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcstop
   ```

   trcstop コマンドを実行すると、入出力アクティビティーの報告が標準出力に表示されます (ただし、画面がスクロールされて全体が表示されない場合もあります)。仮想記憶の入出力の報告は、多くの入出力が関連する 20 個のセグメントに制限されます。
2. すべてのファイルシステム・レベルでアクティビティーをモニターし、`fmon.out` ファイルに報告を書き込むには、次のように入力します。

   ```
   filemon -o fmon.out -O all
   ```

   filemon コマンドによって自動的にシステム・トレースが開始され、コマンド自体はバックグラウンドに入ります。このコマンドの後に、この時点で実行するアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcstop
   ```

   trcstop コマンドを実行すると、入出力アクティビティーの報告が `fmon.out` ファイルに書き込まれます。ファイルと入出力システムの 4 つのレベルすべて (論理ファイル、仮想記憶、論理ボリューム、物理ボリュームのすべてのレベル) がモニターされます。論理ファイルと仮想記憶の入出力の報告は、それぞれ、多くの入出力が関連する 20 個のファイルとセグメントに制限されます。

3. すべてのファイルシステム・レベルでアクティビティーをモニターし、fmon.out ファイルに詳細レポートを書き込むには、次のように入力します。

   ```
   filemon -v -o fmon.out -O all
   ```

   filemon コマンドによって自動的にシステム・トレースが開始され、コマンド自体はバックグラウンドに入ります。このコマンドの後に、この時点で実行するアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcstop
   ```

   この例は、fmon.out ファイル上で詳細レポートが生成される点を除き、前の例に似ています。主な違いは、**filemon** コマンドがトレースを開始するときに踏襲したステップが示されることと、要約と詳細レポートに 20 個のファイルとセグメントだけではなく、すべての入出力に関連する (おそらく多数の) すべてのファイルとセグメントが含まれることです。

4. 以前に記録したトレース・セッションから取り出された入出力アクティビティーを報告するには、次のように入力します。

   ```
   filemon -i trcfile | pg
   ```

   この例では、**filemon** コマンドによって、入力ファイル trcfile からファイルシステム・トレース・イベントが読み取られます。トレース・データは既にファイルに取り込まれているので、アプリケーション・プログラムの実行のために、**filemon** コマンドがそれ自体をバックグラウンドに入れることはありません。ファイルが完全に読み取られると、仮想記憶、論理ボリューム、物理ボリュームの各レベルの入出力アクティビティーの報告が標準出力に表示されます (この例では、pg に入力されています)。

5. **trcon** と **trcoff** コマンドを使ってモニター期間を制御しながら、論理ボリュームと物理ボリュームの入出力アクティビティーだけをモニターするには、次のように入力します。

   ```
   filemon -d -o fmon.out -O pv,lv
   ```

   filemon コマンドによって自動的にシステム・トレースが開始され、コマンド自体はバックグラウンドに入ります。このコマンドの後に、この時点で実行するモニターされないアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcon
   ```

   このコマンドの後に、この時点で実行したいモニター対象のアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcoff
   ```

   このコマンドの後に、この時点で実行するモニターされないアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcon
   ```

   このコマンドの後に、この時点で実行したいモニター対象のアプリケーション・プログラムとシステム・コマンドを入力してから、次のように入力します。

   ```
   trcstop
   ```

   この例では、**-O** フラグで、モニターを論理ボリュームと物理ボリュームだけに制限しています。論理ボリュームと物理ボリュームに関連するトレース・イベントだけが使用可能です。また、**-d** フラグを使用しているので、 **trcon** コマンドが実行されるまでモニターは開始されません。システム・トレースは、**trcoff** と **trcon** コマンドによって、断続的に行うことができます。したがって、特定の期間だけモニターできます。

6. オフライン・モードで **filemon** を実行するには、次のように、**trace** コマンドと **gensyms** コマンドを別々に実行し、次に、これらのコマンドからの出力を **filemon** コマンドへの入力として使用します。

   ```
   trace -a -T 768000 -L 10000000 -o trace.out -j 000,000,001,002,003,005,006,139,102,10C,106,00A,107,
   101,104,10D,15B,12E,130,163,19C,154,3D3,1BA,1BE,1BC,10B,221,1C9,222,228,232,45B
   ```

   モニターしたアプリケーション・プログラムとシステム・コマンドを実行してから、次のように入力します。

   ```
   trcstop
   ```

   gensyms ファイルを作成します。

   ```
   gensyms -F > gensyms.out
   ```

   次に、**-i** と **-n** フラグの両方を指定して **filemon** を実行します。

   ```
   filemon -i trace.out -n gensyms.out -O all
   ```

7. データの単位をメガバイトにして自動オフライン・モードで稼働状況レポートを生成するには、次のコマンドを使用します。

   ```
   filemon -O hot,unit=MB -r <rootstring> -A-x "<user command>"
   ```

8. 3 回のトレース・スナップショットを 5 秒間隔で収集して稼働状況レポートを生成するには、次のコマンドを実行します。

   ```
   filemon -O hot -r <rootstring> -A-x "<user command>" -I 3:5
   ```

9. 稼働状況レポートをオフライン・モードで生成するには、次のコマンドを使用します。

   ```
   filemon -r <rootstring> -O hot
   ```

関連情報:

svmon コマンド

trcrpt コマンド

lseek コマンド

ディスク入出力のモニター

---

fileplace コマンド

目的

論理または物理ボリューム内のファイル・ブロックの配置を表示します。

構文

fileplace [{ **-l** | **-p** [**-o** *FragOffset*] [**-n** *FragNumber*] }[**-i**] [**-v**]] *File* | [**-m** *LogicalVolumeName*]

説明

fileplace コマンドは、指定したファイルが入っている論理ボリュームまたは物理ボリューム内のファイルの位置を表示します。

デフォルトでは、**fileplace** コマンドによって、指定したファイルに割り当てられた論理ボリューム・フラグメントの範囲が標準出力に表示されます。論理ボリューム・フラグメントは、そのファイル内での順序どおりに表示されます。短いヘッダーは、ファイル・サイズ (バイト単位)、そのファイルが存在する論理ボ

リュームの名前、そのボリュームのブロック・サイズ (バイト単位)、フラグメント・サイズ (バイト単位)、およびファイルシステムが圧縮されているかどうかを示す圧縮状態を表します。

ファイルの一部がボリューム内のフラグメントにマッピングされない場合があります。このような領域のサイズは、フラグメントの合計数になっており、ファイルシステムによって暗黙にゼロで埋められています。**fileplace** コマンドは、ファイル内のどの領域にフラグメントが割り当てられていないかを示します。

オプションとして **fileplace** コマンドは、次の情報も表示します。

- ファイルがボリューム内でどのように拡散しているかを示す統計。
- ファイルの間接ブロック・アドレス。
- ファイルの各物理コピーの、物理ボリューム上 (論理ボリューム上ではない) の配置。

注**:**

1. **fileplace** コマンドでは、リモートのネットワーク・ファイルシステム (NFS) のファイルの位置を表示できません。リモート・ファイルを指定すると、**fileplace** コマンドによってエラー・メッセージが戻されます。しかし、**fileplace** コマンドをファイル・サーバー上で直接実行している場合は、リモート・ファイルの位置を表示できます。
2. **fileplace** コマンドでは、ディスク上の論理ボリュームから直接ファイルのブロックのリストが読み取られます。ファイルが新しく作成、拡張、または切り捨てられた場合は、**fileplace** コマンドを実行したときに、ファイルシステム情報がまだディスク上にない可能性があります。**sync** コマンドを使って、ファイル情報を論理ボリュームにフラッシュしてください。
3. JFS2 ファイルシステムには、間接/二重間接ブロックの概念がありません。ファイルは、エクステントという用語で表されます。そのため、最大エクステントのサイズは、集合ブロック・サイズによって決まります。集合ブロック・サイズが 512 バイト (最小許容サイズ) の場合、最大エクステントは 512*(2^ 24-1) バイト長 (8G 弱) になります。集合ブロック・サイズが 4096 バイト (最大許容サイズ) の場合、最大エクステントは 4096*(2^ 24-1) バイト長 (64G 弱) になります。

   これらの制限は、単一エクステントのみに適用されます。全体のファイル・サイズを制限することはありません。

フラグ

項目	説明
-i	ファイルの間接ブロックがある場合、それを表示します。指定されているのが **-l** フラグか **-p** フラグかによって、間接ブロックが論理ボリューム・ブロック・アドレスまたは物理ボリューム・ブロック・アドレスのいずれかで表示されます。
-l	ファイルが存在する論理ボリュームについて、論理ボリューム・フラグメントにおけるファイル位置を表示します。**-l** と **-p** フラグは相互排他です。 注**: -l** フラグも **-p** フラグも指定しなければ、デフォルトで **-l** フラグが暗黙指定されます。両方のフラグを指定すると、**-p** フラグが使用されます。
-m *LogicalVolumeName*	論理ボリュームの論理対物理マップを表示します。
-n *FragNumber*	先頭ブロックから *FragNumber* に相当するブロックまでの範囲にある論理ファイル・ブロックまたは物理ファイル・ブロックを表示します。
-o *FragOffset*	*fragoffset*+1 に相当するブロックから最終ブロックまでの範囲にある論理ファイル・ブロックまたは物理ファイル・ブロックを表示します。**fileplace** コマンドに **-n** フラグと **-o** フラグの両方を指定した場合には、特定フラグメントのアドレスを表示します。
-p	ファイルが存在する物理ボリュームについて、物理ボリュームにおけるファイル位置を表示します。ファイルが存在する論理ボリュームがミラーリングされている場合、物理位置は各ミラー・コピーについて表示されます。**-l** と **-p** フラグは相互排他です。

項目	説明
-v	ファイルとその位置についての、より詳細な情報を表示します。これには、ファイルがボリュームの中でどの程度拡散しているか、およびフラグメントの状態の程度についての統計が含まれます。**-l** フラグまたは **-p** フラグを指定してあると、論理または物理ボリューム・フラグメント番号についての統計が表示されます。 ファイル・スペース効率 は、null でないフラグメント数 (*N*) をファイルに割り当てられたフラグメントの範囲 (*R*) で除算し、その商を 100 で乗算、つまり (*N*/*R*) x 100 として計算されます。範囲は、(アドレスに割り当てられた最大数) - (アドレスに割り当てられた最小数) + 1、つまり、 *MaxBlk-MinBlk*+1 で計算されます。例えば、ファイルに対して書き込まれた論理ブロック数が 01550 から 01557 であれば、 *N* は 8 となります。範囲 *R* は、 (01557 - 01550 +1) で、これも 8 となります。このファイルのスペース効率は、100%、つまり 8/8 x 100 です。**-v** フラグ・メッセージは、式 (*N*/*R*)+100 の結果を表示します。 この効率計算方法により、32 KB よりも大きいファイルの場合は、その間接ブロックの使用状況の関係で効率が 100% になることはあり得ません。 順次効率 は、1 - (ギャップ数 (*nG*) / 適用可能なギャップ数 (*nPG*))、つまり、1 - (*nG*/*nPG*) と定義されます。適用可能なギャップ数は *N* から 1 を減算した値 (*nPG*=*N* - 1) に等しくなります。ファイルが 9 個のブロック (32 KB 超) に書き込まれ、論理フラグメント列に次のように表示されるとします。 `01550-01557` `01600` この場合、ファイルは適用可能な 9 個のフラグメントのうちの 2 個に格納されます。このファイルの順次効率計算は次のようになります。 *nG*=1 *nPG*=9-1=8 (1-1/8) x 100=87.5%

例

1. 論理ボリュームにおけるファイルの位置を表示するには、次のように入力します。

   ```
   fileplace data1
   ```

   この例では、フラグメントとファイル `data1` が入っている論理ボリュームのリストが表示されます。
2. ファイルの間接ブロックを表示するには、次のように入力します。

   ```
   fileplace -i data1
   ```

   デフォルトの論理ボリューム・フラグメントのリストに加えて、ファイル・ブロック・アドレスをファイルシステムに格納するための間接ブロック (ただし、存在する場合のみ) が表示されます。
3. ファイルの詳細な位置情報を表示するには、次のように入力します。

   ```
   fileplace -v data1
   ```

   デフォルトの論理ボリューム・フラグメントのリストに加えて、配置効率についての統計が表示されます。
4. 物理ボリュームにおけるファイルの位置についての情報をすべて表示するには、次のように入力します。

   ```
   fileplace -piv data1
   ```

   この例では、物理ボリュームにおけるファイルと間接ブロックのリストが表示され、配置効率についての統計も表示されます。

5. **/usr/lib/boot/unix_mp** ファイルの先頭 18 ブロックに関して、その基礎となる物理ボリュームの位置を表示するには、次のように入力します。

   ```
   fileplace -n 18 -p /usr/lib/boot/unix_mp
   ```

6. **/usr/lib/boot/unix_mp** ファイルの 18 番目のブロックから最終ブロックまでの基礎となる物理ボリュームの位置を表示するには、次のように入力します。

   ```
   fileplace  -p -o 17 /usr/lib/boot/unix_mp
   ```

7. **/usr/lib/boot/unix_mp** ファイルの 18 番目のブロックの基礎となる物理ボリュームの位置を表示するには、次のように入力します。

   ```
   fileplace -o 17 -n 1 -p /usr/lib/boot/unix_mp
   ```

ファイル

項目	説明
/dev/hd0, /dev/hd1, .../dev/hd*n*	論理ボリュームを指定します。

関連情報:

sync コマンド

ディスク入出力のモニター

論理ボリューム・ストレージ

---

find コマンド

目的

照合式に一致するファイルを検索します。

構文

find [**-H** | **-L**] *Path ...* [*Expression*]

説明

find コマンドは、ディレクトリー・ツリーの中で、指定されたそれぞれの *Path* パラメーターを再帰的に検索し、ブール式と一致するファイルを探します。ブール式は、以下のテキストに示す項を使用して記述されます。 **find** コマンドがディレクトリー構造を上から下へ再帰的に検索している場合は、現在の階層にシンボリック・リンクによってリンクしているディレクトリーを検索しません。 **find** コマンドからの出力は、*Expression* パラメーターで指定した項に応じて異なります。

find コマンドは 4.3 BSD 高速検索構文をサポートしません。

フラグ

項目	説明
-H	コマンド・ラインで検出されたシンボリック・リンクに対して評価されたファイル情報とファイル・タイプが、リンクで参照されるファイル (リンク自体ではない) のファイル情報とファイル・タイプになります。参照されるファイルが存在しない場合は、ファイル情報とファイル・タイプはリンク自体に関するものです。コマンド・ラインにないすべてのシンボリック・リンクに関するファイル情報は、リンク自体のファイル情報です。
-L	シンボリック・リンクに対して評価されたファイル情報とファイル・タイプが、リンクで参照されるファイル (リンク自体ではない) のファイル情報とファイル・タイプになります。

式の項

ブール式と変数は、**find** コマンドの検索境界を *Path* と *Expression* パラメーターの定義に従って記述します。

注**:** 以下の定義では、*n* 変数には **+***n* (*n* より大きい)、**-***n* (*n* より小さい)、または *n* (*n* と等しい) という表記で 10 進整数を指定し、*Number* 変数には **+***Number* (*Number* より大きい)、**-***Number* (Number より小さい)、または *Number* (*Number*-1 から *Number*) という表記で 10 進整数を指定します。

項目	説明
¥ (*Expression* **¥)**	括弧で囲まれている式が真であるとき、真となります。
-amin *n*	*n* の値は、以下のいずれかの値とすることができます。 ***n*** 初期設定時間から減算されたファイル・アクセス時間を 60 秒で割った値 (余りは切り捨てる) が *n* のとき、値は真となります。 ***-n*** 初期設定時間から減算されたファイル・アクセス時間を 60 秒で割った値 (余りは切り捨てる) が *n* より小さいとき、値は真となります。 ***+n*** 初期設定時間から減算されたファイル・アクセス時間を 60 秒で割った値 (余りは切り捨てる) が *n* より大きいとき (UNIX03 の場合は *n*+1 より大きいとき)、値は真となります。 例えば、**-amin 2** は、ファイルが 1 分から 2 分以内にアクセスされていれば、真です。 **注: find** コマンドの開始時間以降にアクセスしたファイルは考慮されません。ただし、UNIX03 以外の動作に対して **find** コマンドを単項 NOT 演算子の中で使用した場合、コマンドの開始時刻以降に変更されたファイルは値 *n* に到達するまで表示されます。

項目	説明
-atime *n*	*n* の値は、以下のいずれかの値とすることができます。 ***n*** 初期設定時間から減算されたファイル・アクセス時間を 86400 秒で割った値 (余りは切り捨てる) が *n* のとき、値は真となります。 ***-n*** 初期設定時間から減算されたファイル・アクセス時間を 86400 秒で割った値 (余りは切り捨てる) が *n* より小さいとき、値は真となります。 ***+n*** 初期設定時間から減算されたファイル・アクセス時間を 86400 秒で割った値 (余りは切り捨てる) が *n* より大きいとき (UNIX03 の場合は *n*+1 より大きいとき)、値は真となります。 **注: -atime** の定義が変更され、Single UNIX Specification バージョン 3 に準拠するようになりました。以前の動作では、24 時間の *n*-1 倍から *n* 倍までの間にファイルがアクセスされたとき、 **-atime** が真と評価されていました。デフォルトでは、`find -atime` は UNIX03 より前の働きと同様の働きをします。 UNIX03 の動作を可能にするには、環境変数 **XPG_SUS_ENV** を `ON`、**XPG_UNIX98** を `OFF` に設定します。 このオプションの以前の動作を可能にするには、 **XPG_UNIX98** 変数を `ON` に設定します。 **find** コマンドの開始時間以降にアクセスしたファイルは考慮されません。ただし、UNIX03 以外の動作に対して **find** コマンドを単項 `NOT` 演算子の中で使用した場合、コマンドの開始時刻以降に変更されたファイルは値 *n* に到達するまで表示されます。
-cmin *n*	*n* の値は、以下のいずれかの値とすることができます。 ***n*** 初期設定時間から減算されたファイルの i ノード変更時間を 60 秒で割った値 (余りは切り捨てる) が *n* のとき、値は真となります。 ***-n*** 初期設定時間から減算されたファイルの i ノード変更時間を 60 秒で割った値 (余りは切り捨てる) が *n* より小さいとき、値は真となります。 ***+n*** 初期設定時間から減算されたファイルの i ノード変更時間を 60 秒で割った値 (余りは切り捨てる) が *n* より大きいとき (UNIX03 の場合は *n*+1 より大きいとき)、値は真となります。 **注: find** コマンドの開始時刻より後に変更された i ノードがあるファイルは、考慮されません。ただし、UNIX03 以外の動作に対して **find** コマンドを単項 `NOT` 演算子の中で使用した場合、コマンドの開始時刻以降に i ノード変更したファイルは値 *n* に到達するまで表示されます。
-cpio *Device*	現行ファイルを指定されたデバイスに **cpio** コマンド・フォーマットで書き込みます。

項目	説明
-ctime *n*	*n* の値は、以下のいずれかの値とすることができます。 ***n*** 初期設定時間から減算されたファイルの i ノード変更時間を 86400 秒で割った値 (余りは切り捨てる) が *n* のとき、値は真となります。 ***-n*** 初期設定時間から減算されたファイルの i ノード変更時間を 86400 秒で割った値 (余りは切り捨てる) が *n* より小さいとき、値は真となります。 ***+n*** 初期設定時間から減算されたファイルの i ノード変更時間を 86400 秒で割った値 (余りは切り捨てる) が *n* より大きいとき (UNIX03 の場合は *n*+1 より大きいとき)、値は真となります。 **注: -ctime** の定義が変更され、Single UNIX Specification バージョン 3 に準拠するようになりました。ファイルが、24 時間の *n*-1 倍から *n* 倍までの間にアクセスされたとき、 **-ctime** の以前の動作は真となりました。デフォルトでは、`find -ctime` は UNIX03 より前の働きと同様の働きをします。 UNIX03 の動作を可能にするには、環境変数 **XPG_SUS_ENV** を `ON`、**XPG_UNIX98** を `OFF` に設定します。 このオプションの以前の動作を可能にするには、 **XPG_UNIX98** 変数を `ON` に設定します。 **find** コマンドの開始時間以降に i ノード変更したファイルは考慮されません。ただし、UNIX03 以外の動作に対して **find** コマンドを単項 `NOT` 演算子の中で使用した場合、コマンドの開始時刻以降に i ノード変更したファイルは値 *n* に到達するまで表示されます。
-depth	値は常に真となります。ディレクトリー階層を下位方向に検索していくので、ディレクトリー内のすべてのエントリーが、ディレクトリーそのものよりも先に影響を受けます。それは、**find** コマンドを **cpio** コマンドと一緒に使用して、書き込み許可を持たないディレクトリー内に含まれているファイルを転送する場合に便利です。
-ea	ファイルにアクセス制御リスト (ACL) または拡張属性 (EA) が設定されていれば、値は真と評価されます。
-exec *Command*	指定したコマンドが実行されて、終了状況として値 0 が戻されるとき、値は真となります。指定したコマンドの終わりには、引用符で囲んだセミコロン、エスケープされたセミコロン、または正符号を付ける必要があります。 2 つの文字 {} (中括弧) を含む引数の後には、指定したコマンドの終わりを示す正符号を付ける必要があります。コマンド・パラメーター {} (中括弧) は現行パス名に置き換えられます。
-follow	シンボリック・リンクおよびハード・リンクが後に続くようにします。
-fstype *Type*	ファイルが属しているファイルシステムが指定されたタイプであるとき、値は真となります。*Type* 変数の値には `jfs` (ジャーナル・ファイルシステム)、`nfs` (ネットワーク・ファイルシステム)、`jfs2` (拡張ジャーナル・ファイルシステム)、`procfs` (proc ファイルシステム)、または `namefs` (名前ファイルシステム) が入っています。
-group *Group*	ファイルが指定したグループに属するとき、値は真になります。 *Group* 変数の値が数値で、`/etc/group` ファイルに無い場合は、グループ ID と解釈されます。
-inum *n*	ファイルの i ノード番号が *n* 変数の値と一致するとき、値は真となります。
-links *n*	ファイル内に、指定した数のリンクがあるとき、値は真となります。リンクについての説明は、**ln** コマンドを参照してください。
-iregex *regular_expression*	ファイルのパス名全体が正規表現と一致する場合、値は真となります。このオプションは **-regex** オプションと似ていますが、突き合わせの際に大/小文字を区別しない点は異なります。
-long	**-ls** と組み合わせて使用すると、各ユーザー名/グループ名を、最初の 8 文字に切り捨てずに、使用可能なすべての文字が印刷されます。

項目 説明

-ls
値は常に真となります。現行パス名を関連する統計と一緒に表示します。これらの統計には、以下の値が含まれます。

- i ノード番号
- KB (1024 バイト) 単位のサイズ
- 保護モード
- ハード・リンク数
- ユーザー
- グループ
- バイト単位のサイズ
- 変更時間

ファイルがスペシャル・ファイルの場合、サイズ・フィールドにはメジャー・デバイス番号とマイナー・デバイス番号が含まれます。ファイルがシンボリック・リンクの場合、リンク先のファイルのパス名が -> (ハイフン、より大) シンボルに続いて出力されます。このフォーマットは、**ls -filds** コマンドのフォーマットと似ていますが、フォーマット化は、**ls** コマンドを実行せずに内部的に行われます。このため、**ls** コマンドの出力結果とは異なる場合があります (保護モードの出力の有無など)。

-mmin *n*
n の値は、以下のいずれかの値とすることができます。

n 初期設定時間から減算されたファイル変更時間を 60 秒で割った値 (余りは切り捨てる) が *n* のとき、値は真となります。

-n 初期設定時間から減算されたファイル変更時間を 60 秒で割った値 (余りは切り捨てる) が *n* より小さいとき、値は真となります。

+n 初期設定時間から減算されたファイル変更時間を 60 秒で割った値 (余りは切り捨てる) が *n* より大きいとき (UNIX03 の場合は *n*+1 より大きいとき)、値は真となります。

注: find コマンドの開始時間以降に変更したファイルは考慮されません。ただし、UNIX03 以外の動作に対して **find** コマンドを単項 NOT 演算子の中で使用した場合、コマンドの開始時刻以降に変更されたファイルは値 *n* に到達するまで表示されます。

-mtime *n*
n の値は、以下のいずれかの値とすることができます。

n 初期設定時間から減算されたファイル変更時間を 86400 秒で割った値 (余りは切り捨てる) が *n* のとき、値は真となります。86400 秒は 24 時間です。

-n 初期設定時間から減算されたファイルの変更時間を 86400 秒で割った値 (余りは切り捨てる) が *n* より小さいとき、値は真となります。

+n 初期設定時間から減算されたファイルの変更時間を 86400 秒で割った値 (余りは切り捨てる) が *n* より大きいとき (UNIX03 の場合は *n*+1 より大きいとき)、値は真となります。

注: -mtime の定義は、Single UNIX Specification, Version 3 に対応するように変更されました。ファイルが、24 時間の *n*-1 倍から *n* 倍までの間に変更されたとき、**-mtime** の以前の動作は真となりました。デフォルトでは、**find -mtime** は UNIX03 より前の働きと同様の働きをします。 UNIX03 の動作を可能にするには、環境変数 **XPG_SUS_ENV** を ON、**XPG_UNIX98** を OFF に設定します。

このオプションの以前の動作を可能にするには、 **XPG_UNIX98** 変数を ON に設定します。

find コマンドの開始時間以降に変更したファイルは考慮されません。ただし、UNIX03 以外の動作のために **find** コマンドを単項 NOT 演算子の中で使用した場合、コマンドの開始時刻より後に変更されたファイルは *n* の値まで表示されます。

項目	説明
-name *File*	*File* 変数の値がファイル名と一致するとき、値は真となります。通常のシェル・ファイル名生成文字 (**sh** コマンドを参照) が使用できます。パターンは、引用符またはエスケープ文字で囲む必要があります。**find** コマンドがシェルから使用される場合は、エスケープ文字が使用されます。円記号 (¥) は、パターン内ではエスケープ文字として使用されます。ワイルドカード (パターン・マッチング) 文字は、引用符で囲めば使用できます。ワイルドカード文字の使用方法について詳しくは、「オペレーティング・システムおよびデバイスの管理」の『ワイルドカードとメタキャラクターを使ったパターン・マッチング』のセクションを参照してください。 [a-z] のような式で、ハイフンは、現行の照合シーケンスによってからを意味します。照合シーケンスは文字範囲内で使用する等価クラスを定義できます。照合シーケンスと等価クラスについて詳しくは、「ナショナル・ランゲージ・サポート ガイドおよびリファレンス」の『ナショナル・ランゲージ・サポートの概要』を参照してください。
-newer *File*	現行ファイルが、*File* 変数で指定したファイルよりも後に変更された場合、値は真となります。
-nogroup	ファイルが /etc/group データベース以外のユーザーに属するとき、値は真となります。
-nouser	ファイルが /etc/passwd データベースに無いユーザーに属するとき、値は真となります。
-ok *Command*	**-exec** 式と同じですが、**find** コマンドは、指定されたコマンドを開始する必要があるかどうかをユーザーに確認します。ユーザーが肯定応答すると、そのコマンドを始動します。指定したコマンドの末尾には、引用符で囲んだセミコロンまたは ¥; (円記号 (エスケープ) とセミコロン) を付けなければなりません。
-perm [**-**] *OctalNumber*	ファイルの許可コードが *OctalNumber* パラメーターと正確に一致する場合、値は真となります。ファイルの許可については、**chmod** コマンドのセクションを参照してください。オプションの **-** (ハイフン) がある場合は、少なくともこれらの許可が設定されていれば、この式の値は真になります。 *OctalNumber* パラメーターには最高 9 個までの 8 進数を指定できます。 **注:** TCB 環境の一部であるファイルの場合、ファイルのアクセス権に追加のセキュリティー・ビットが付加されます。これらのファイルには S_ITCB ビットが設定されており、セキュリティー・ビット・セットは 0x010000000 と定義されます。よって、TCB 対応ファイルの 8 進数の許可の値には、ビット設定 100000000 がその他の許可ビットと一緒に含まれている必要があります。 例: TCB 環境の一部であるファイルをリストするには、find -perm 100000600 -print と入力します。それによって、オーナー読み取り許可とオーナー書き込み許可のみを持ち、TCB 環境の一部であるファイル名がリストされます。許可コードについての説明は、**chmod** コマンドを参照してください。

項目	説明
-perm [**-**] *Mode*	ファイル・モード・ビットを表すのに、mode 引数が使用されます。これは、**chmod** で説明されている <symbolicmode> オペランドと同じフォーマットで、次のように解釈されます。 最初に、テンプレートは、すべてのファイル・モード・ビットがクリアされているものと想定されます。オペランド・シンボル (-) には、以下の機能があります。 **+** 適切なモード・ビットをテンプレート内に設定する。 **-** 適切なビットをクリアする。 **=** プロセスのファイル・モード作成マスクの内容とは関係なく、適切なモード・ビットを設定する。 オペランド・シンボル **-** は、モードの先頭文字になることはできません。それにより、オプションを先導するハイフンとのあいまいさが避けられます。初期モードでは、すべてのビットがオフであるので、先頭文字として **-** を使用する必要のあるシンボル・モードはありません。 ハイフンが省略されている場合、ファイル許可ビットが、結果として生じるテンプレートの値と正確に一致すると、プライマリーは真として評価します。そうでない場合、つまり、モードに、ハイフンがプレフィックスとして付いている場合、少なくとも、結果として生じるテンプレート内のすべてのビットがファイル許可ビット内に設定されていれば、プライマリーは真として評価します。 *Mode* パラメーターのコマンド構文は、**chmod** コマンド構文と同一です。この式は、ファイルにこれらの許可が正確に付いていれば、値は真となります。オプションの **-** (ハイフン) がある場合は、少なくともこれらの許可が設定されていれば、この式の値は真になります。
-print	値は常に真となります。現行パス名を表示します。 **find** コマンドは、**-exec**, **-ls** または **-ok** のいずれかの式を指定しない限り、**-print** 式を想定します。
-prune	値は常に真となります。現行パス名がディレクトリーの場合は、そこから下位のパス名の検索は行われません。 **-depth** フラグを指定すると、**-prune** フラグは無視されます。
-size *n*	ファイルの長さが指定した *n* ブロック (1 ブロックは 512 バイト) であるとき、値は真となります。比較を行うために、ファイル・サイズはブロック単位に切り上げられます。
-regex *regular_expression*	ファイルのパス名全体が正規表現と一致する場合、値は真となります。このオプションでは正規表現の検索は行いませんが、ファイルの完全パス名と正規表現の突き合わせは行います。例えば、`./test` という名前のファイルを突き合わせる場合、`.*test.*` または `.*t.*t` という正規表現を使用することができますが、`t.*t` は使用できません。
-regextype *Type*	値は常に真となります。このオプションでは、**-regex** オプションと **-iregex** オプションにおける正規表現の構文のタイプを指定します。このオプションは、コマンド・ラインの後ろの方にある正規表現にも影響します。 *Type* 変数には、次のいずれかの値を指定できます。 **Basic** 基本的な正規表現の構文。 **Extended** 拡張的な正規表現の構文。 **注: -regextype** オプションを使用しない場合、正規表現は `basic` と解釈されます。
-size *n***c**	ファイルの長さが正確に、指定した *n* バイトであるとき、値は真となります。 *n* 変数の末尾に **c** を付けると、ファイル・サイズがブロックではなくバイト単位で測定されたことを示します。

項目	説明
-type *Type*	*Type* 変数が次のいずれかの値を示すと、値は真となります。 **b** ブロック・スペシャル・ファイル **c** キャラクター・スペシャル・ファイル **d** ディレクトリー **f** 普通ファイル **l** シンボリック・リンク **p** 先入れ先出し (名前付きパイプ) **s** ソケット
-user *User*	ファイルが指定したユーザーに属するとき、値は真となります。 *User* 変数の値が数字で、/etc/passwd ファイルにログイン名として記述されていない場合は、ユーザー ID と解釈されます。
-xdev	値は常に真となります。**find** コマンドは、*Path* パラメーターで指定されたファイルシステムをまたがって検索するのを防止します。

これらの式は、次の演算子を優先順位が高い方から順に使用して組み合わせることができます。

1. **(** *Expression***)** - 括弧で囲まれた式と演算子のグループ (括弧はシェル専用で、円記号のエスケープ・シーケンスが必要です)。
2. **!** *Expression* - 式の否定 (「**!**」は単項 NOT 演算子です)。
3. *Expression* [**-a**] *Expression* - 式の連結 (AND 演算は 2 個の式を並置して暗黙に指定するか、またはオプションの **-a** 演算子として明示的に記述することができます)。
4. *Expression* **-o** *Expression* - 式の論理和。**-o** は OR 演算子です。第 1 の式が真であれば、第 2 の式は計算されません。

注**: find** コマンドと **cpio** コマンドを一緒に使用する場合、-follow オプションと -L オプションを **cpio** コマンドで使用してください。これら 2 つのオプションを一緒に使用しなければ、望ましくない結果が生じます。式が存在しない場合は、-print がデフォルトの式の中で使用されます。例えば、指定された式がプライマリー -exec、-ok、および -print をどれも含んでいない場合、その式は (*given_expression*) -print によって置き換えられます。-user、-group、および -newer プライマリーは、それぞれの引数を 1 回だけ評価します。 -exec または -ok によって指定されたコマンドを使用しても、同じファイル上のその後のプライマリーに影響しません。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	すべての *Path* パラメーターが正常に処理されました。
>0	エラーが発生しました。

例

1. 指定した基本ファイル名を持つ、ファイルシステム内のすべてのファイルをリストするには、次のように入力します。

   ```
   find /  -name .profile -print
   ```

   このコマンドによって、ファイルシステム全体が検索され、.profile という名前のすべてのファイルの完全なパス名が書き出されます。 / (スラッシュ) は **find** コマンドに対して、ルート・ディレクト

リーとそのサブディレクトリーすべてを検索するよう指示します。時間の無駄を省くには、ファイルが入っていると思われるディレクトリーを指定して、検索範囲を限定するのが最善の方法です。

2. 現行ディレクトリー・ツリー内で、特定の許可コードの付いたファイルをリストするには、次のように入力します。

   ```
   find . -perm 0600  -print
   ```

   このコマンドによって、オーナー読み取り許可とオーナー書き込み許可のみを持ったファイルの名前がリストされます。 . (ドット) は **find** コマンドに対して現行ディレクトリーとそのサブディレクトリーを検索するよう指示します。許可コードについての説明は、**chmod** コマンドを参照してください。

3. いくつかのディレクトリー内で特定の許可コードの付いたファイルを検索するには、次のように入力します。

   ```
   find manual clients proposals -perm -0600  -print
   ```

   このコマンドによって、オーナー読み取り許可とオーナー書き込み許可、およびその他の考えられる許可の付いたファイル名がリストされます。manual、clients、および proposals の各ディレクトリーおよびそのサブディレクトリーが検索されます。前の例で示したように、-perm 0600 は、 0600 と正確に一致する許可コードの付いたファイルのみを選択します。この例では、-perm -0600 は、0600 で示されるアクセスと、それ以外の 0600 レベルより上のアクセスを許可する許可コードの付いたファイルを選択します。許可コード 0622 と 0744 も適合します。

4. この 24 時間以内に変更された現行ディレクトリー内のすべてのファイルをリストするには、次のように入力します。

   ```
   find .  -ctime 1  -print
   ```

5. 複数のリンクがある正規ファイルを検索するには、次のように入力します。

   ```
   find .  -type f  -links +1  -print
   ```

   このコマンドによって、複数のリンク (-links +1) を持つ通常のファイル (-type f) の名前がリストされます。

   注: どのディレクトリーにも、少なくとも 2 個のリンク、つまり親ディレクトリー内のエントリーと、そのディレクトリー自体の . (ドット) エントリーがあります。複数のファイル・リンクについては、**ln** コマンドのセクションで説明します。

6. パス名に **find** が含まれているすべてのアクセス可能なファイルを検索するには、次のように入力します。

   ```
   find .  -name '*find*' -print
   ```

7. 1 週間アクセスされておらず、かつ nfs を使用してマウントされていない a.out または *.o という名前のファイルをすべて除去するには、次のように入力します。

   ```
   find / ¥( -name a.out -o -name '*.o' ¥) -atime +7 ! -fstype nfs -exec rm {} ¥;
   ```

   注: **-atime** 式の中で使用されている数値が +7 になっています。これは、アクセスされなかった時間が 1 週間 (24 時間の期間が 7 つ) を超えるファイルに対してコマンドを実行する場合の正しいエントリーです。

8. SCCS という名前のディレクトリーまたは SCCS ディレクトリー内のファイルを除く、現行ディレクトリー内、またはその下のすべてのファイルのパス名を表示するには、次のように入力します。

   ```
   find .  -name SCCS  -prune -o -print
   ```

SCCS ディレクトリーの名前を含む、現行ディレクトリー内、またはその下のすべてのファイルのパス名を表示するには、次のように入力します。

```
find . -print -name SCCS -prune
```

9. 長さが正確に 414 バイトのファイルをすべて検索するには、次のように入力します。

   ```
   find . -size 414c -print
   ```

10. .c という接尾部の付いたホーム・ディレクトリー内のすべてのファイルを検索して除去するには、次のように入力します。

    ```
    find /u/arnold -name "*.c" -exec rm {} ¥;
    ```

    find コマンドが .c という接尾部でファイルを識別するたびに、**rm** コマンドがそのファイルを削除します。 **rm** コマンドは **-exec** 式に対して指定される唯一のパラメーターです。{ } (中括弧) は現行パス名を表します。

11. この例では、dirlink はディレクトリー dir に対するシンボリック・リンクです。コマンド行上のシンボリック・リンク dirlink を参照することによって dir 内のファイルをリストするには、次のように入力します。

    ```
    find -H dirlink -print
    ```

12. この例では、dirlink はディレクトリー dir に対するシンボリック・リンクです。 dirlink 内のファイルをリストし、シンボリック・リンクを含め、dir 下のファイル階層をトラバースするには、次のように入力します。

    ```
    find -L dirlink -print
    ```

13. シンボリック・リンク dirlink で参照されるファイル dir1 が dir2 より新しいかどうかを判別するには、次のように入力します。

    ```
    find -H dirlink -newer dir2
    ```

    注**: -H** フラグが使用されるので、時間データは dirlink からではなく dir1 から収集されます。dir1 は、シンボリック・リンクのトラバースによって検出されます。

14. 拡張ユーザー名とグループ名付きの **ls** フォーマットで現行ディレクトリー内のファイルのリストを出すには、次のように入力します。

    ```
    find . -ls -long
    ```

15. 現行ディレクトリー内の ACL/EA 設定のファイルをリストするには、次のように入力します。

    ```
    find . -ea
    ```

16. 60 分以内に変更されたファイルをリストするには、次のように入力します。

    ```
    find . -mmin -60
    ```

17. パス名にパターン afile が含まれているすべてのパス名を /home ディレクトリーで検索するには、次のコマンドを入力します。

    ```
    find /home -regextype basic -regex ".*afile.*"
    ```

18. パス名にパターン afile または cap が含まれているすべてのパス名を /home ディレクトリーで検索するには、次のコマンドを入力します。

    ```
    find /home -regextype extended -regex ".*afile.*|.*cap.*"
    ```

19. パス名にパターン afile、AFILE、cap、または CAp が含まれているすべてのパス名を /home ディレクトリーで検索するには、次のコマンドを入力します。

    ```
    find /home -regextype extended -iregex ".*afile.*|.*cap.*"
    ```

ファイル

項目	説明
/usr/bin/find	**find** コマンドが入っています。
/bin/find	**find** コマンドへのシンボリック・リンク。
/etc/group	すべての既知グループのリストが入っています。
/etc/passwd	すべての既知ユーザーのリストが入っています。

関連情報:

ln コマンド

Backup methods

ファイルのタイプ

入出力リダイレクト

シェル・コマンド

---

finger コマンド

目的

ユーザー情報を表示します。このコマンドは **f** コマンドと同じです。

構文

{ **finger** | **f** }[[**-b**][**-h**] [**-l**][**-p**]] | [**-i**][**-q**][**-s**][**-w**]]

[**-f**][**-m**][*User* | *User* **@***Host* | **@***Host*]

説明

/usr/bin/finger コマンドでは、現在ホストにログインしているユーザーの情報が表示されます。出力形式は、表示される情報の選択によって異なります。

デフォルトのフォーマット

デフォルトの形式には次の項目が入ります。

- ログイン名
- フル・ユーザー名
- 端末名
- 書き込み状況 (端末名の前の * (アスタリスク) は、書き込み許可が拒否されていることを表します)

ホスト上のユーザーの場合、デフォルト情報のリストには、次の項目も含まれます。

- アイドル時間 (アイドル時間は、単一の整数であれば分、 : (コロン) があれば時間と分、「d」があれば日付と時間です。)
- ログイン時刻
- サイト特定の情報

サイト特定の情報は、 **/etc/passwd** ファイルの gecos フィールドから検索されます。gecos フィールドには、後ろにコンマまたは / (スラッシュ文字) が付いたフル・ユーザー名が入っています。finger コマンドによって、コンマまたはスラッシュ文字の後にすべての情報がサイト特定の情報と一緒に表示されます。

長形式

長形式は、ユーザー名のリストが提供されると必ず **finger** コマンドによって使用されます。 (ユーザーの姓名のほかに、アカウント名も使用できます。) このフォーマットは複数行で設定され、次に示す情報の他に上記の情報をすべて含みます。

- ユーザーの **$HOME** ディレクトリー
- ユーザーのログイン・シェル
- ユーザーの **$HOME** ディレクトリー内の、 **.plan** ファイルの内容
- ユーザーの **$HOME** ディレクトリー内の、 **.project** ファイルの内容

finger コマンドは、リモート・システム上のユーザーの検索にも使用できます。フォーマットは、ユーザーを *User@Host* として指定します。ユーザー名を省略すると、**finger** コマンドによって標準フォーマットのリストがリモート・システム上に表示されます。

好みのテキスト・エディターを使って、 **.plan** ファイルと **.project** ファイルを作成し、それらのファイルを **$HOME** ディレクトリーに入れてください。 **finger** コマンドでは、**.plan** ファイルと **.project** ファイルの内容を表示するときに、**toascii** サブルーチンによって、通常の ASCII 文字範囲外の文字が変換されます。**finger** コマンドでは、変換された文字の前に `M-` が表示されます。

User パラメーターでユーザーを指定する場合は、ユーザーの姓、名、もしくはアカウント名を指定します。ユーザーを指定すると、指定したホストで、**finger** コマンドによってユーザーの情報が長形式で戻されます。

finger コマンドのその他の情報については、「ネットワークおよびコミュニケーションの管理」の『TCP/IP のインストール』のセクションを参照してください。

フラグ

項目	説明
-b	簡潔な長形式をリストします。
-f	出力時にヘッダー行の印刷 (表示中のフィールドを定義する最初の行) を抑止します。
-h	**.project** ファイルを、長形式および簡潔な形の長形式では表示しません。
-i	アイドル時間と一緒に高速でリストします。
-l	長形式をリストします。
-m	*User* パラメーターは、ユーザーのログイン名ではなく、ユーザー ID (任意アクセス制御に使用される) であると見なされます。
-p	**.plan** ファイルを、長形式と簡潔な形の長形式では表示しません。
-q	高速でリストします。
-s	短形式をリストします。
-w	幅の狭い、短形式をリストします。

パラメーター

項目	説明
@Host	リモート・ホスト上のすべてのログイン・ユーザーを指定します。
ユーザー	**/etc/passwd** ファイル内で指定した、ローカル・ユーザー ID (任意アクセス制御に使用される) またはローカル・ログイン名を指定します。
User@Host	長形式で表示されたリモート・ホスト上のユーザー ID を指定します。

例

1. ホスト alcatraz にログインしている、すべてのユーザーの情報を得るには、次のように入力します。

   ```
   finger @alcatraz
   ```

   次のような情報が表示されます。

   ```
   [alcatraz.austin.ibm.com]
   Login     Name         TTY Idle          When      Site Info
   brown    Bob Brown   console   2d   Mar 15 13:19
   smith    Susan Smith    pts0  11:   Mar 15 13:01
   jones    Joe Jones      tty0    3   Mar 15 13:01
   ```

   ユーザー brown は console でログインし、ユーザー smith は、疑似テレタイプ・ライン pts0 からログインし、ユーザー jones は、 tty0 からログインしています。

2. alcatraz のユーザー brown に関する情報を得るには、次のように入力します。

   ```
   finger brown@alcatraz
   ```

   次のような情報が表示されます。

   ```
   Login name: brown
   Directory: /home/brown    Shell: /home/bin/xinit -L -n Startup
   On since May 8 07:13:49 on console
   No Plan.
   ```

3. ローカル・ホスト上で、ユーザー brown の情報を簡易書式で得るには、次のように入力します。

   ```
   finger -q brown
   ```

   次のような情報が表示されます。

   ```
   Login           TTY             When
   brown           pts/6           Mon Dec1710:58
   ```

ファイル

項目	説明
/usr/bin/finger	**finger** コマンドが入っています。
/etc/utmp	現在ログインされているユーザーのリストが入っています。
/etc/passwd	ユーザー・アカウント、ユーザー名、ホーム・ディレクトリーを定義します。
/etc/security/passwd	ユーザーのパスワードを定義します。
/var/adm/lastlog	最後にログインした時刻が入っています。
$HOME/.plan	ユーザー・プランの 1 行単位の説明が入っているオプションのファイル。
$HOME/.project	ユーザー・プロジェクトの割り当てが入っているオプションのファイル。

関連資料:

801 ページの『hostname コマンド』

581 ページの『fingerd デーモン』

関連情報:

ログイン・ユーザーに関する情報を表示するコマンド

通信およびネットワークのセクション

fingerd デーモン

目的

finger コマンドにサーバー機能を提供します。

構文

注**: fingerd** デーモンは、通常、 **inetd** デーモンによって始動されます。また、システム・リソース・コントローラー (SRC) コマンドを使用して、コマンド・ラインから制御することもできます。

/usr/sbin/fingerd [**-s**] [**-f**]

説明

/usr/sbin/fingerd デーモンは、いくつかのネットワーク・サイトで **finger** コマンドへのインターフェースを提供する単純プロトコルです。**finger** コマンドは現行システムまたはユーザーに関する状況報告を戻します。 **fingerd** デーモンは、TCP (伝送制御プロトコル) のポート 79 での要求を、**/etc/services** ファイルと、 **/etc/inetd.conf** ファイルのリストに従って listen します。

デフォルトでは、**fingerd** デーモンは、個別のサイト・セキュリティーについて、**finger** 要求をほかのシステムへ転送しません。 **finger** 転送要求を受け取ると、 **fingerd** デーモンでは、 **finger** コマンドに対して、「`Finger forwarding service denied`」というメッセージで応答します。システム管理者は、**-f** フラグを使って **fingerd** デーモンを実行するときにデフォルトとして、finger の転送をオンにするオプションを指定できます。

fingerd デーモンの変更は、System Management Interface Tool (SMIT) または SRC を使用するか、**/etc/inetd.conf** ファイルまたは **/etc/services** ファイルを編集することによって行います。 `fingerd` をコマンド・ラインに入力するのはお勧めしません。デフォルトでは、**fingerd** デーモンは、**/etc/inetd.conf** ファイル内のコメントが解除されていれば始動されます。

inetd デーモンは、**/etc/inetd.conf** ファイルと **/etc/services** ファイルから情報を取り出します。

/etc/inetd.conf または **/etc/services** ファイルを変更した後、**refresh -s inetd** または **kill-1***InetdPID* コマンドを実行して、**inetd** デーモンにその構成ファイルに対する変更内容を通知します。

fingerd デーモンを実行するには、最低限の特権を持つユーザー ID が必要です。 **nobody** ID で、最低の許可を与えることができます。 **fingerd** デーモンに **nobody** ユーザー ID を与えることによって、fingerd デーモンがユーザーのホストで使用できるようになります。 **/etc/services** ファイルを、使用するユーザー ID を反映するように変更してください。

システム・リソース・コントローラーによる **fingerd** デーモンの操作

fingerd デーモンは、**inetd** デーモン (SRC) のサブシステムのサブサーバーです。**fingerd** デーモンは **tcpip** SRC サブシステム・グループのメンバーです。このデーモンは、**/etc/inetd.conf** ファイル内でコメントを外すと使用可能になり、次の SRC コマンドで操作できます。

項目	説明
startsrc	サブシステム、サブシステム・グループ、またはサブサーバーを始動します。
stopsrc	サブシステム、サブシステムのグループ、またはサブサーバーを停止します。
lssrc	サブシステム、サブシステムのグループ、またはサブサーバーの状況を得ます。

フラグ

項目	説明
-s	ソケット・レベルのデバッグをオンにします。

項目	説明
-f	この **fingerd** デーモンについて finger の転送をオンにします。

例

注**: fingerd** デーモンの引数は、SMIT を使用するか、 **/etc/inetd.conf** ファイルを編集して指定することができます。

1. **fingerd** デーモンを始動するには、次のように入力します。

   ```
   startsrc -t finger
   ```

   このコマンドによって、**fingerd** サブサーバーが始動します。
2. **fingerd** デーモンを停止するには、通常、次のように入力します。

   ```
   stopsrc -t finger
   ```

   このコマンドは、保留状態の接続をすべて開始し、既存の接続を完了します。しかし、新しい接続の開始は行いません。
3. **fingerd** デーモンとすべての **fingerd** 接続を強制的に停止するには、次のように入力します。

   ```
   stopsrc -f -t finger
   ```

   このコマンドによって、保留状態の接続と既存の接続がすべて即座に終了します。
4. **fingerd** デーモンについての簡略状況報告を表示するには、次のように入力します。

   ```
   lssrc -t finger
   ```

   このコマンドは、デーモンの名前、プロセス ID、および状態 (アクティブまたは非アクティブ) を戻します。

関連情報:

kill コマンド

startsrc コマンド

TCP/IP デーモン

/etc/inetd.conf コマンド

---

fish コマンド

目的

ゴー・フィッシュ・カード・ゲームをプレイします。

構文

fish

説明

ゴー・フィッシュ・ゲームの目的は、同じ数字の 4 枚のカードの組を集めることです。このゲームでは、プログラム (対戦相手) と交互に相手のカードを要求します。要求した数字のカードを対戦相手が持っていたら、対戦相手はそれらのカードを渡さなければなりません。持っていない場合は、対戦相手から「GO FISH!」というプロンプト指示が出されるので、配られていないカードの山から 1 枚カードを引きます。要求した数字のカードが引けた場合は、もう 1 枚引けます。4 枚そろった組は、卓上に置かれます。ゲームはカードがなくなるまで続きます。組数の多い方が勝ちです。**fish** コマンドは、勝者を知らせて終了します。

fish コマンドはゲームを始める前に「instructions?」というプロンプトを表示します。ゲームの説明を参照する場合は、「Y (はい)」を入力してください。

ゲームの最初に p を入力すると、プロ・レベルのゲームができます。デフォルトではアマチュア・レベルのゲームになります。

ゴー・フィッシュでは、相手が出す以下のプロンプトに対して自分の欲しいカードを入力します。

```
you ask me for:
```

上記のプロンプトに Enter キーだけを押すと、相手の手のカードの枚数と配られていないカードの枚数が表示されます。

このゲームでは、次の内容が表示されます。

- プレーヤーの現在の手 (既に 4 枚そろった組も含む)
- 「GO FISH!」というプロンプト
- 「GO FISH!」というプロンプト
- 対戦相手 (プログラム) が要求したカード
- (プレーヤーおよび対戦相手の) 完成した組
- プレーヤーまたは対戦相手が再度要求したときの要求のカード

例

次に、**fish** の画面表示の例を示します。

```
your hand is: A 5 5 7 10 J Q
you ask me for: 5
I say "GO FISH!"
You draw A
I ask you for: 5
Made a book of 5's
I get another guess
I ask you for 6
You say "GO FISH!"
your hand is: A A 7 10 J Q
you ask me for:
```

プレイの途中でゲームを終了するには、割り込みキー・シーケンス (Ctrl-C) を押します。

ファイル

項目	説明
/usr/games	システム・ゲームのロケーション。

関連情報:

arithmetic コマンド

bj コマンド

moo コマンド

quiz コマンド

---

flcopy コマンド

目的

ディスケットへのコピーと、ディスケットからのコピーを行います。

構文

flcopy [**-f** *Device*] [**-h** | **-r**] [**-t** *Number*]

説明

flcopy コマンドは、(**/dev/rfd0** としてオープンされた) ディスケットを、現行ディレクトリーに作成した **floppy** という名前のファイルにコピーしてから、「`Change floppy, hit return when done`」というメッセージを出力します。次に、**floppy** コマンドは **floppy** ファイルをディスケットにコピーします。フラグ **-f**、**-h**、**-r**、**-t***Number* を使用して、**flcopy** コマンドの動作を変更できます。

注**: flcopy** コマンドでは、サイズの異なるディスケット間でデータをコピーすることはできません。

フラグ

項目	説明
-f *Device*	**/dev/rfd0** 以外のドライブが指定できます。
-h	**flcopy** コマンドに現行ディレクトリー内の **floppy** ファイルをオープンさせ、**/dev/rfd0** にコピーさせます。
-r	ディスケットを現行ディレクトリー内の **flcopy** ファイルにコピーした後で終了するよう **floppy** コマンドに命令します。
-t *Number*	*Number* で指定した数のトラックをコピーします。常にディスケットの最初のトラックからコピーされます。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. **/dev/rfd1** を、現行ディレクトリー内の **floppy** ファイルにコピーするには、次のように入力します。

   ```
   flcopy -f/dev/rfd1 -r
   ```

2. ディスケットの最初の 100 トラックをコピーするには、次のように入力します。

```
flcopy -f/dev/rfd1 -t100
```

ファイル

項目	説明
/usr/sbin/flcopy	**flcopy** コマンドが入っています。

関連資料:

595 ページの『format コマンド』

関連情報:

fd コマンド

---

flush-secldapclntd コマンド

目的

flush-secldapclntd コマンドは、**secldapclntd** デーモン・プロセスのキャッシュをフラッシュします。

構文

//usr/sbin/flush-secldapclntd

説明

flush-secldapclntd コマンドは、**secldapclntd** デーモン・プロセスのキャッシュをクリアします。

例

1. **secldapclntd** デーモンのキャッシュをフラッシュするには、次のように入力します。

   ```
   /usr/sbin/flush-secldapclntd
   ```

ファイル

項目	説明
/etc/security/ldap/ldap.cfg	**secldapclntd** デーモンがサーバーに接続する際に必要な情報が入っています。

関連情報:

mksecldap コマンド

start-secldapclntd コマンド

restart-secldapclntd コマンド

/etc/security/ldap/ldap.cfg コマンド

---

fmt コマンド

目的

送信前にメール・メッセージをフォーマットします。

構文

/usr/bin/fmt [**-***Width*] [*File* ...]

説明

fmt コマンドは、連結された入力 *Files* (*Files* の指定がない場合は標準入力) を読み取るテキスト・フォーマッターを始動し、入力の行の長さを **-***Width* の値に設定したものを標準出力に生成します。 **-***Width* フラグによる値の指定がない場合は、72 文字のデフォルト値を使用します。入力行の先頭のスペーシングは、ブランク行や語間のスペーシングと同じように、出力内に保存されます。

fmt コマンドは通常、送信前のメール・メッセージをフォーマットしてメッセージの外観を整えるために使用します。しかし、**fmt** コマンドは単純なフォーマット作業にも有効です。例えば、vi エディターなどのテキスト編集プログラムのビジュアル・モードで **!}fmt** コマンドを実行すると、すべての行が **-***Width* フラグで指定した値に設定されるようにパラグラフがフォーマットされます。*-Width* フラグによる値の指定がない場合は、72 文字のデフォルト値を使用します。複雑なフォーマット操作の場合は、**fmt** より標準テキスト編集プログラムの方が適しています。

注**:** メッセージ内に組み込みメッセージまたは他のファイルのフォーマット済みの情報が含まれている場合は、**fmt** コマンドを使用しないでください。このコマンドは、組み込みメッセージの見出し情報をフォーマットします。フォーマット済みの情報のフォーマットが変更されてしまう場合があります。

フラグ

項目	説明
File	フォーマットするファイルの名前を指定します。
-*Width*	行の長さを指定します。*Width* のデフォルト値は 72 文字です。

例

1. メール・エディターを使って作成したメッセージをフォーマットするには、次のように入力します。

   ```
   ~| fmt
   ```

   ~| はメッセージの左マージンに入力します。 **~| fmt** コマンドを実行すると、メッセージがフォーマットされます。「(continue)」という語が表示され、さらに情報を入力するかメッセージを送信するよう指示されます。
2. ファイルをフォーマットし、出力を画面に表示するには、次のように入力します。

   ```
   fmt file1
   ```

   この例では、file1 というファイルがフォーマットされ、画面に表示されます。

ファイル

項目	説明
/usr/bin/fmt	**fmt** コマンドが入っています。

関連情報:

mail コマンド

nroff コマンド

vi コマンド

メール・アプリケーション

fold コマンド

目的

出力デバイスの幅が固定されている場合に、長い行を折り返します。

構文

fold [**-b**] [**-s**] [**-w** *Width*] [*File...*]

説明

fold コマンドは、出力デバイスの幅が限定されている場合に、長い行を折り返すフィルターです。デフォルトでは、このコマンドは行を 80 桁の幅になるように切って、標準入力の内容を折り返します。また、コマンドへの入力として 1 個または複数のファイルを指定できます。

fold コマンドは、個々の出力行が *Width* パラメーターで指定した値を超えない範囲でできるだけ広くなるように、入力行に改行文字を挿入します。 **-b** フラグを指定すると、行幅はバイトで計算されます。**-b** フラグを指定しない場合は、次のとおりになります。

- *Width* は、**LC_CTYPE** 環境変数で決定された桁数で計算されます。
- バックスペース文字は、出力行の長さを 1 ずつ減らします。
- タブ文字は、桁位置が 8 の倍数プラス 1 桁となる次の桁に進みます。

fold コマンドは、ファイルにタブが含まれていれば、8 の倍数の **-w** *Width* 値を受け入れます。ファイルにタブが含まれているときに他の幅値を使用する場合は、 **expand** コマンドを使用してから、 **fold** コマンドを使用してください。

注**:**

1. **fold** コマンドを使用すると、アンダーライン付きテキストが影響を受けることがあります。
2. **fold** コマンドは、 **-b** フラグを使用しているときでも、マルチバイト文字の間には改行文字を挿入しません。

フラグ

項目	説明
-b	*Width* をバイト単位で計算します。デフォルトでは桁単位で計算します。
-s	出力行セグメントに空白文字が含まれている場合に、*Width* の範囲内の右端のブランクの後に来る行を切ります。デフォルトでは、個々の出力行セグメントができるだけ広くなるように行を切ります。
-w *Width*	最大行幅を *Width* 変数の値として指定します。デフォルトは 80 です。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	すべての入力ファイルが正常に処理されました。
>0	エラーが発生しました。

例

longlines というファイルの行を幅 72 桁で折り返すには、次のように入力します。

```
fold -w 72 longlines
```

ファイル

項目	説明
/usr/bin/fold	**fold** コマンドが入っています。

関連資料:

458 ページの『expand コマンド』

関連情報:

tab コマンド

---

folder コマンド

目的

フォルダーとメッセージを選択してリストします。

構文

folder [**+** *Folder*] [*Message*] [**-all**] [**-nopack** | **-pack**] [**-nofast** | **-fast**] [**-norecurse** | **-recurse**] [**-print** | **-noprint**] [**-header** | **-noheader**] [**-nototal** | **-total**] [**-push** | **-pop**] [**-list** | **-nolist**]

説明

folder コマンドは、該当フォルダーの現行フォルダーと現行メッセージを設定し、ユーザーのフォルダーに関する情報をリストします。デフォルトでは、**folder** コマンドは、現行フォルダー名、メッセージ数、メッセージ番号の範囲、および現行メッセージを表示します。

+*Folder* フラグで指定したフォルダーが現行フォルダーになります。*Message* パラメーターで指定したメッセージが、現行フォルダーの現行メッセージになります。**-pack** フラグを指定すると、フォルダー内のメッセージの番号が付け直されます。

フラグ

項目	説明
-all	メール・ディレクトリー内の各フォルダーの情報を、1 行に 1 フォルダー分ずつ表示します。
-fast	フォルダーの名前のみを表示します。
+*Folder*	表示するフォルダー情報を指定します。
-header	フォルダー情報の列見出しを表示します。
-help	コマンド構文、使用可能なスイッチ (トグル)、およびバージョン情報を表示します。 注: メッセージ・ハンドラー (MH) の場合、このフラグ名は完全にスペルアウトしなければなりません。
-list	現行フォルダーの後に、フォルダー・スタックの内容を表示します。
Message	指定されたメッセージを現行メッセージに設定します。 **+***Folder* フラグを指定しないと、このコマンドでは指定したメッセージが現行フォルダーに設定されます。メッセージを指定するには、次の参照を使用します。 *Number* メッセージの番号。 **cur** または **.** (ピリオド) 現行メッセージ。これはデフォルトです。 **first** フォルダー内の最初のメッセージ。 **last** フォルダー内の最後のメッセージ。 **next** 現行メッセージの次のメッセージ。 **new** 新たに作成するメッセージ。 **prev** 現行メッセージの直前のメッセージ。
-nofast	各フォルダーの情報を表示します。このフラグはデフォルトです。
-noheader	フォルダー情報の列見出しを表示しません。このフラグがデフォルトです。
-nolist	フォルダー・スタックの内容を表示しません。このフラグがデフォルトです。
-nopack	フォルダー内のメッセージの番号を付け直しません。このフラグがデフォルトです。
-noprint	フォルダー情報を表示しません。**-push**、**-pop**、**-list** フラグを指定した場合は、**-noprint** フラグがデフォルトになります。
-norecurse	現行フォルダー内の最上位フォルダーの情報だけを表示します。サブフォルダーの情報は表示しません。このフラグはデフォルトです。
-nototal	メール・ディレクトリー構造内にあるすべてのメッセージとフォルダーの表示を行いません。**-all** フラグを指定した場合は **-total** フラグがデフォルトになり、それ以外の場合は **-nototal** フラグがデフォルトになります。
-pack	指定されたフォルダー内のメッセージの番号を付け直します。番号を付け直すと、メッセージが削除されたために生じたメッセージ番号の欠番がなくなります。
-pop	フォルダー・スタックの最上部からフォルダーを除去して、現行フォルダーにします。 **-pop** フラグと一緒に **+***Folder* フラグを指定することはできません。
-print	フォルダーの情報を表示します。**-push**、**-pop**、**-list** フラグを指定した場合は **-noprint** フラグがデフォルトになり、それ以外の場合は **-print** フラグがデフォルトになります。
-push	現行フォルダーをフォルダー・スタックの最上部に移して、指定されたフォルダーを現行フォルダーにします。フォルダーを指定せずに **-push** フラグを使用すると、フォルダー・スタックの最上部のフォルダーと現行フォルダーが交換されます。
-recurse	現行フォルダー内にあるすべてのフォルダーとサブフォルダーの情報を表示します。
-total	メール・ディレクトリー構造内にあるすべてのメッセージとフォルダーを表示します。**-recurse** フラグを指定しないと、**-total** フラグを指定してもサブフォルダーの情報は表示されません。**-all** フラグを指定した場合は、**-total** フラグがデフォルトになります。

プロファイル・エントリー

以下のエントリーは、*UserMhDirectory***/.mh_profile** ファイルに入力します。

項目	説明
Current-Folder:	デフォルトの現行フォルダーを設定します。
Folder-Protect:	新規フォルダー・ディレクトリーの保護レベルを設定します。
Folder-Stack:	フォルダー・スタックを指定します。
lsproc:	フォルダーの内容の表示に使用するプログラムを指定します。
Path:	ユーザーの MH ディレクトリーを指定します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 現在のフォルダーの情報を表示するには、次のように入力します。

   ```
   folder
   ```

   システムは次のようなメッセージで応答します。

   ```
   inbox+ has 80 messages (1-82); cur = 7; (others).
   ```

   この例では、現行フォルダーは inbox です。このフォルダーには、メッセージ 1 からメッセージ 82 までの、 80 メッセージが入っています。現行メッセージの番号は 7 です。
2. すべてのフォルダーの情報を表示するには、次のように入力します。

   ```
   folder -all
   ```

   システムは次のようなメッセージで応答します。

   ```
   Folder    # of messages (range); cur msg   (other files)
   inbox+ has 80 messages (1-82); cur= 7; (others).
   test   has   5 messages (1-5);   cur= 5; (others).

          Total= 85 messages in 2 folders
   ```

   この例では、2 つのフォルダーに合計 85 のメッセージが入っています。現行フォルダーは inbox です。これは、後ろに + (正符号) が付いていることから分かります。
3. test フォルダーを現行フォルダーにして、 test に関する情報を表示するには、次のように入力します。

   ```
   folder +test
   ```

   システムは次のようなメッセージで応答します。

   ```
   test+ has 5 messages (1-5); cur = 5; (others)
   ```
4. メッセージ 2 を、現行フォルダーの現在のメッセージにするには、次のように入力します。

   ```
   folder 2
   ```

   システムは次のようなメッセージで応答します。

   ```
   test+ has 5 messages (1-5); cur = 2; (others)
   ```
5. group というフォルダーを作成して、それを現行フォルダーにするには、次のように入力します。

   ```
   folder +group
   ```

システムは次のようなメッセージで応答します。

```
Create  folder  "/home/dawn/Mail/group"?  _
```

次のように入力します。

```
yes
```

システムは次のようなメッセージで応答します。

```
group+  has  no  messages.
```

6. 現行フォルダー内のメッセージの番号を付け直すには、次のように入力します。

```
folder   -pack
```

システムは次のようなメッセージで応答します。

```
inbox+  has  80  messages  (1-80);  cur=  7;  (others).
```

この例では、メッセージの番号が付け直され、メッセージが削除されたために生じたメッセージ番号の欠番がなくなります。

ファイル

項目	説明
$HOME/.mh_profile	MH ユーザー・プロファイルが入っています。
/usr/bin/folder	**folder** コマンドが入っています。

関連情報:

mhpath コマンド

refile コマンド

mh_profile ファイル

メール・アプリケーション

---

folders コマンド

目的

メール・ディレクトリー内のすべてのフォルダーとメッセージをリストします。

構文

folders [*+Folder*] [*Message*] [**-all**] [**-pack** | **-nopack**] [**-fast** | **-nofast**] [**-recurse** | **-norecurse**] [**-print** | **-noprint**] [**-header** | **-noheader**] [**-total** | **-nototal**] [**-push** | **-pop**] [**-list** | **-nolist**]

説明

folders コマンドは、メール・ディレクトリー内のすべてのフォルダーとメッセージをリストします。このコマンドは、**-all** フラグを指定して、 **folder** コマンドを実行するのと同じ効果があります。

フラグ

項目	説明
-all	メール・ディレクトリー内の各フォルダーの情報を、1 行に 1 フォルダー分ずつ表示します。
-fast	フォルダーの名前のみを表示します。
+*Folder*	表示するフォルダー情報を指定します。
-header	フォルダー情報の列見出しを表示します。このフラグがデフォルトです。
-help	コマンド構文、使用可能なスイッチ (トグル)、およびバージョン情報を表示します。 注: メッセージ・ハンドラー (MH) の場合、このフラグ名は完全にスペルアウトしなければなりません。
-list	現行フォルダーの後に、フォルダー・スタックの内容を表示します。
Message	指定されたメッセージを現行メッセージに設定します。 **+***Folder* フラグを指定しないと、このコマンドでは指定したメッセージが現行フォルダーに設定されます。メッセージを指定するには、次の参照を使用します。 *Number* メッセージの番号。 **cur** または **.** (ピリオド) 現行メッセージ。これはデフォルトです。 **first** フォルダー内の最初のメッセージ。 **last** フォルダー内の最後のメッセージ。 **next** 現行メッセージの次のメッセージ。 **new** 新たに作成するメッセージ。 **prev** 現行メッセージの直前のメッセージ。
-nofast	各フォルダーの情報を表示します。このフラグはデフォルトです。
-noheader	フォルダー情報の列見出しを表示しません。
-nolist	フォルダー・スタックの内容を表示しません。このフラグがデフォルトです。
-nopack	フォルダー内のメッセージの番号を付け直しません。このフラグがデフォルトです。
-noprint	フォルダー情報を表示しません。**-push**、**-pop**、**-list** フラグを指定した場合は、**-noprint** フラグがデフォルトになります。
-norecurse	メール・ディレクトリー内にあるフォルダーの情報を表示します。サブフォルダーの情報は表示しません。このフラグはデフォルトです。
-nototal	メール・ディレクトリー構造内のすべてのメッセージとフォルダーの表示を行いません。
-pack	フォルダー内のメッセージの番号を付け直します。番号を付け直すと、メッセージが削除されたために生じたメッセージ番号の欠番がなくなります。
-pop	フォルダー・スタックの最上部からフォルダーを除去して、現行フォルダーにします。
-print	各フォルダー内のメッセージ数、各フォルダーの現行メッセージ、現行フォルダーを表示します。 **-push**、**-pop**、**-list** フラグを指定した場合は **-noprint** フラグがデフォルトになり、それ以外の場合は **-print** フラグがデフォルトになります。
-push	現行フォルダーをフォルダー・スタックの最上部に移して、指定されたフォルダーを現行フォルダーにします。フォルダーを指定せずに **-push** フラグを使用すると、フォルダー・スタックの最上部のフォルダーと現行フォルダーが交換されます。
-recurse	メール・ディレクトリー構造内にあるすべてのフォルダーとサブフォルダーの情報を表示します。
-total	メール・ディレクトリー構造内にあるすべてのメッセージとフォルダーを表示します。**-recurse** フラグを指定しないと、**-total** フラグを指定してもサブフォルダーの情報は表示されません。**-total** フラグはデフォルトです。

プロファイル・エントリー

以下のエントリーは、*UserMhDirectory*/**.mh_profile** ファイルに入力します。

項目	説明
Current-Folder:	デフォルトの現行フォルダーを設定します。
Folder-Protect:	新規フォルダー・ディレクトリーの保護レベルを設定します。
Folder-Stack:	フォルダー・スタックを指定します。
lsproc:	フォルダーの内容の表示に使用するプログラムを指定します。
Path:	ユーザーの MH ディレクトリーを指定します。

例

1. すべてのフォルダーの情報を表示するには、次のように入力します。

   ```
   folders
   ```

   システムは次のようなメッセージで応答します。

   ```
   Folder  # of messages (range); cur msg  (other files)
   inbox+ has 80 messages (1-82); cur= 7; (others).
   test has 5 messages (1-6);   cur=  5; (others).

      Total= 85 messages in 2 folders.
   ```

   この例では、2 つのフォルダーに合計 85 のメッセージが入っています。現行フォルダーは inbox です。これは、後ろに + (正符号) が付いていることから分かります。
2. フォルダーの名前だけをすべてリストするには、次のように入力します。

   ```
   folders    -fast
   ```

   システムは次のようなメッセージで応答します。

   ```
   inbox
   test
   ```
3. すべてのフォルダー内のメッセージの番号を付け直すには、次のように入力します。

   ```
   folders    -pack
   ```

   システムは次のようなメッセージで応答します。

   ```
   inbox+ has 80 messages (1-80); cur= 7; (others).
   test has 5 messages (1-5);  cur= 5; (others).
   ```

   この例では、 inbox フォルダーと test フォルダー内のメッセージの番号が付け直され、メッセージを削除したために生じたメッセージ番号の欠番がなくなっています。

ファイル

項目	説明
$HOME/.mh_profile	MH ユーザー・プロファイルが入っています。
/usr/bin/folders	**folders** コマンドが入っています。

関連情報:

mhpath コマンド

packf コマンド

refile コマンド

mh_profile コマンド

forcerpoffline コマンド

目的

ピア・ドメインを強制的にオフラインにします。

構文

forcerpoffline **[-h]** *domain_name*

説明

注意**:** このコマンドを使用する際は、十分に注意してください。

forcerpoffline は、ノードがオンライン状態の保留中のときに、このノードを **startrpdomain** コマンドを使用してオンラインにすることができない場合に限り使用する必要があります。このシナリオは、ドメインがクォーラムで動作している場合にノードをオンラインにしようとすると発生する可能性があります。ノードがオンライン状態の保留中のままになっている原因がわからない場合は、**ctsnap** コマンドを実行してから **forcerpoffline** コマンドを使用してください。**forcerpoffline** コマンドの実行の結果、構成リソース・マネージャー・サブシステム (**IBM.ConfigRM**) と RMC サブシステム (**ctrmc**) がリサイクルされます。

パラメーター

domain_name
オフラインにする定義済みのピア・ドメインに対して名前を指定します。

フラグ

-h コマンドの使用方法のステートメントを標準出力に書き込みます。

ファイル

/var/ct/cfg/current_cluster ファイルと **/var/ct/cfg/default_cluster** ファイルが変更されます。

標準出力

-h フラグが指定されている場合は、このコマンドの使用状況ステートメントが標準出力に書き込まれます。

終了状況

0 コマンドが正常に実行されました。

1 潜在的な RMC エラーのためコマンドが強制終了しました。

2 コマンド・スクリプト内の潜在的なエラーのためコマンドが強制終了しました。

3 ユーザーが無効なフラグを指定したためコマンドが強制終了しました。

4 ユーザーが無効なパラメーターを指定したためコマンドが強制終了しました。

5 ユーザー・エラー (存在しないドメイン名の指定など) のためコマンドが強制終了しました。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

実装上の固有な条件

このコマンドは、**rsct.basic.rte** ファイルセット (AIX の場合) および **rsct.basic-3.1.0.0-0.***platform***.rpm** パッケージ (Linux、Solaris、および Windows の場合) の一部です。ここで、*platform* は **i386**、**ppc**、**ppc64**、**s390**、または **x86_64** です。

位置

/opt/rsct/bin/forcerpoffline

関連情報:

ctsnap コマンド

startrpdomain コマンド

stoprpdomain コマンド

---

format コマンド

目的

ディスケットまたは読み取り/書き込み光メディア・ディスクをフォーマットします。

構文

format [**-d** *Device*] [**-f**] [**-l**]

説明

重要**:** ディスケット、または読み取り/書き込み光ディスクをフォーマットすると、格納されている既存のデータがすべて破棄されます。

format コマンドは、*Device* パラメーターで指定されたディスケット・ドライブ内のディスケットをフォーマットします。**format** コマンドは次の中からデバイス・タイプを 1 つ決定します。

- 9 セクターずつのトラックが 40 x 2 個ある、 5.25 インチの低密度ディスケット (360KB)
- 15 セクターずつのトラックが 80 x 2 個ある、 5.25 インチの大容量ディスケット (1.2MB)
- 9 セクターずつのトラックが 80 x 2 個ある 3.5 インチの低密度ディスケット (720KB)
- 18 セクターずつのトラックが 80 x 2 個ある 3.5 インチの大容量ディスケット (1.44MB)
- 36 セクターずつのトラックが 80 x 2 個ある 3.5 インチの大容量ディスケット (2.88MB)

セクターのサイズはどのディスケット・タイプの場合も 512 バイトです。

format コマンドは、*Device* パラメーターで異なる密度を指定しない限り、ディスケット・ドライブがサポートする最大容量にディスケットをフォーマットします。

format コマンドで読み取り/書き込み光ディスクをフォーマットするには、ドライブで欠陥リスト・ヘッダーのフォーマット・オプション有効 (FOV) ビットを 0 に設定できることが条件になります。読み取り/書き込み光ディスクをフォーマットする場合は、**-d** フラグの後に読み取り/書き込み光ディスク・ドライブの名前 (**/dev/romd0** など) を指定します。詳細については、「*Technical Reference: Kernel and Subsystems, Volume 2*」の『scdisk SCSI Device Driver』のセクションの、 **ioctl** サブルーチンの **DKFORMAT** 操作の説明を参照してください。

format コマンドは、ディスケットまたは読み取り/書き込み光ディスクをフォーマットする前に、確認を求めるプロンプトを表示します。これで操作を確実に終了させることができます。

フラグ

項目	説明
-d *Device*	ディスケット・フォーマットに使用するデバイスを指定します。デバイス名の末尾に文字 **h** が付いていると、ドライブはディスケットを高密度用にフォーマットします。デバイス名の末尾に文字 **l** が付いていると、ドライブはディスケットを低密度用にフォーマットします。有効なデバイス・タイプについては、**fd** スペシャル・ファイルのセクションを参照してください。このフラグは **format** コマンドと一緒に使用しなければなりません。 注意**:** ディスケット・ドライブがサポートする容量がディスケットの最大容量を上回っている場合は、**format** コマンドの *Device* パラメーター (**-d** *Device* フラグ) でディスケットの容量を明示的に指定する必要があります。例えば、1MB のディスケットを 4MB のディスケット・ドライブでフォーマットするには、次のようにディスケット容量を **-d** フラグ内で指定します。 `-d /dev/fd0.9 for a 1MB diskette` 容量を指定しないと、読み取りエラーや書き込みエラーを引き起こす場合があります。
-f	不良トラックの検査をしないでディスケットをフォーマットします。この方法をとると、短時間でフォーマットできます。 このフラグはディスケットにのみ適用されます。読み取り/書き込み光ディスクには適用されません。このフラグは **format** コマンドと一緒に使用しなければなりません。
-l	(小文字の L) 5.25 インチ、1.2MB のディスケット・ドライブでは、360KB のディスケットをフォーマットします。3.5 インチ 1.4MB のディスケット・ドライブでは、720KB のディスケットをフォーマットします。 3.5 インチ 1.4MB のディスケット・ドライブでは、 720KB のディスケットをフォーマットします。このフラグはディスケットにのみ適用されます。読み取り/書き込み光ディスクには適用されません。このフラグは **format** コマンドと一緒に使用しなければなりません。 重要**:** 360KB のディスケット・ドライブでは、1.2MB のドライブでフォーマットした 360KB のディスケットを読み取ることはできません。

パラメーター

項目	説明
Device	フォーマットしたいディスケットが入っているデバイスを指定します。デフォルトは、デバイス **/dev/rfd0** (ドライブ 0) です。

例

1. ディスケットを **/dev/rfd0** デバイスでフォーマットするには、次のように入力します。
   ```
   format  -d  /dev/rfd0
   ```
2. 不良トラックを検査せずにディスケットをフォーマットするには、次のように入力します。
   ```
   format  -f
   ```
3. **/dev/rfd1** デバイスにある 5.25 インチ、 1.2MB のディスケット・ドライブで 360KB のディスケットをフォーマットするには、次のように入力します。
   ```
   format  -l  -d  /dev/rfd1
   ```
4. 3.5 インチ、低密度 (720KB) のディスケットをフォーマットするには、次のように入力します。
   ```
   format -d /dev/fd0.9
   ```
5. 3.5 インチ、大容量 (1.44MB) のディスケットをフォーマットするには、次のように入力します。
   ```
   format -d /dev/fd0.18
   ```
6. **/dev/romd0** デバイス内の、読み取り/書き込み光ディスクをフォーマットするには、次のように入力します。
   ```
   format -d /dev/romd0
   ```

ファイル

項目	説明
/usr/sbin/format	**format** コマンドが入っています。
/dev/rfd*	デバイス・パラメーターを指定します。
/dev/fd*	デバイス・パラメーターを指定します。
/dev/romd*	デバイス・パラメーターを指定します。
/dev/omd*	デバイス・パラメーターを指定します。

関連資料:

584 ページの『flcopy コマンド』

532 ページの『fdformat コマンド』

関連情報:

fd コマンド

---

fortune コマンド

目的

運勢のデータベースから無作為に運勢を表示します。

構文

fortune [**-**] [**-s** | **-l** | **-a** [**-w**]] [*File*]

説明

fortune コマンドは、**fortunes.dat** ファイルまたは *File* パラメーターで指定されたファイルから運勢を表示します。運勢を表示した後、**fortune** コマンドは終了します。

フラグ

項目	説明
-	使用方法の概要を表示します。
-a	両方のタイプの運勢を表示します。
-l	長期的な運勢だけを表示します。
-s	短期的な運勢だけを表示します。
-w	運勢を表示した後、ユーザーがそれを読む間待機します。

ファイル

項目	説明
/usr/games	システム・ゲームのロケーション。
/usr/games/lib/fortune/fortunes.dat	デフォルトの **fortune** データベースのロケーション。

関連資料:

778 ページの『hangman コマンド』

関連情報:

arithmetic コマンド

moo コマンド

ttt コマンド

forw コマンド

目的

メッセージを転送します。

構文

forw [**+** *Folder*] [**-draftfolder +***Folder* | **-nodraftfolder**] [*Message*] [**-draftmessage** *Message*] [**-digest** *Name* [**-issue** *Number*] [**-volume** *Number*]] [**-form** *FormFile*] [**-editor** *Editor* | **-noedit**] [**-whatnowproc** *Program* | **-nowhatnowproc**] [**-filter***File*] [**-annotate** [**-inplace** | **-noinplace**] | **-noannotate**] [**-format** | **-noformat**] [**-help**]

説明

forw コマンドは、メッセージを転送するためのインターフェースを始動します。デフォルトでは、**forw** コマンド・インターフェースは次のように機能します。

- *UserMhDirectory*/**draft** ファイルをオープンして、編集できるようにします。
- **/etc/mh/mhl.forward** ファイルに定義されたテンプレートに従って、転送情報を入力するよう求めるプロンプトを表示します。
- 転送するメッセージに添付するテキストを追加するよう求めるプロンプトを表示します。

UserMhDirectory/**draft** ファイルの編集を終了するには、Ctrl-D シーケンスを押します。 **forw** コマンドは、現行フォルダーの現行メッセージを **draft** ファイルに追加します。複数のメッセージを追加したい場合は、*Messages* パラメーターを使用します。

注**:** 送信時にメッセージが識別できるように、メッセージのヘッダーと本文の間に、ハイフンの行またはブランク行を 1 行入れておく必要があります。

エディターの終了時に、**forw** コマンドは「`What Now?`」というプロンプトを始動します。使用可能な **whatnow** サブコマンドのリストを参照するには、 Enter キーを押してください。サブコマンドを使うと、メッセージの編集を続けたり、メッセージを表示したり、メッセージの後処理を指示したり、**forw** コマンドの処理を終了できます。

forw コマンドに **-form** フラグを指定すると、転送するメッセージのフォーマットを変更できます。デフォルトでは、*UserMhDirectory*/**forwcomps** ファイルに保存されているデフォルトのメッセージ形式が使用されます。独自の **forwcomps** ファイルを定義していない場合は、**/etc/mh/forwcomps** ファイルが使用されます。

元のメッセージに転送情報を注釈として付けるには、**-annotate** フラグを使用します。確実に注釈を付けるには、**forw** コマンド・インターフェースが終了する前に、転送する注釈を送信します。

注**:** 同じドラフト上で **forw** コマンドを複数回実行する場合、**-annotate** フラグはそのたびごとに指定する必要があります。

フラグ

項目	説明
-annotate	転送するメッセージに次の注釈を付けます。 `Forwarded: Date` `Forwarded: Addresses` 強制的に適切な注釈を付けるには、**-inplace** フラグを使用します。このフラグを使用すると、注釈付きメッセージに対するリンクが保存されます。
-digest *Name*	ダイジェスト機能を使って、新しく発行する *Name* 変数で指定したダイジェストを作成します。**forw** コマンドは、(**repl** コマンドで使用されるのと同じフォーマット文字列機構を使用して) **components** ファイル内のフォーマット文字列を拡張し、標準のダイジェスト・カプセル化アルゴリズムを使ってドラフトを構成します。ドラフトの構成が終わると、**forw** コマンドはダイジェストのボリュームと発行のエントリーを書き出し、エディターを始動します。 **-form** フラグを指定しないと、 **forw** コマンドは、 *UserMhDirectory***/digestcomps** ファイル内のフォーマットを使用します。このファイルが存在しない場合は、**/etc/mh/digestcomps** ファイルに指定されているデフォルトを使用します。
-draftfolder +*Folder*	指定したフォルダーにドラフト・メッセージを入れます。このフラグを指定しないと、**forw** コマンドはメッセージ・ハンドラー (MH) プロファイル内の情報に従ってデフォルトのドラフト・フォルダーを選択します。**+***Folder* を指定しないと、`Current-Folder` が使用されます。**$HOME/.mh_profile** ファイルにデフォルトのデフォルト・フォルダーを定義できます。 **注: -draftfolder +***Folder* の後に *Message* パラメーターを指定すると、**-draftmessage** フラグを指定した場合と同じになります。
-draftmessage *Message*	ドラフト・メッセージを指定します。**-draftmessage** フラグを指定せずに **-draftfolder** を指定した場合のデフォルト・メッセージは `new` です。
-editor *Editor*	メッセージを作成するための初期エディターを指定します。
-filter *File*	転送する各メッセージのフォーマットを再設定して、ドラフト・メッセージに入れます。**-filter** フラグには、**mhl** コマンドで使用するフォーマットを使用できます。
+*Folder*	転送するメッセージが入っているフォルダーを指定します。フォルダーを指定しないと、`Current-Folder` が使用されます。
-form *FormFile*	**forw** コマンドの出力を、*FormFile* 変数で指定したフォーマットで表示します。**forw** コマンドは、指定されたファイル内の各行をフォーマット文字列として扱います。**-digest** フラグも同時に指定すると、**forw** コマンドは *File* 変数で指定されたフォーマットをダイジェストのフォーマットとして使用します。**-digest** フラグを指定して **-form** フラグを指定しないと、ダイジェスト・フィルター・ファイルがフォーマットのデフォルトになります。
-format	**mhl** コマンドとデフォルトのフォーマット・ファイルを使って、転送する各メッセージのフォーマットを再設定し、ドラフト・メッセージに入れます。*UserMhDirectory***/mhl.forward** ファイルがある場合は、その中にデフォルトのフォーマットが入っています。このファイルがない場合は、 **/etc/mh/mhl.forward** ファイルにデフォルトのフォーマットが入っています。
-help	コマンド構文、使用可能なスイッチ (トグル)、およびバージョン情報を表示します。 注**:** メッセージ・ハンドラー (MH) の場合、このフラグ名は完全にスペルアウトしなければなりません。
-inplace	強制的に適切な注釈を付けて、注釈付きメッセージとのリンクを保存します。
-issue *Number*	ダイジェストの発行番号を指定します。デフォルトの発行番号は、*UserMhDirectory***/context** ファイル内の、 `DigestName-issue-list` エントリーの現在の値より 1 大きい値です。

項目	説明
Message	メッセージを指定します。複数のメッセージ、メッセージの範囲、または単一のメッセージを指定できます。メッセージの指定には、次の参照を使用します。 *Number* メッセージの番号。 *Sequence* ユーザーが指定するメッセージのグループ。認識される値は次のとおりです。 **all** フォルダー内のすべてのメッセージ。 **cur** または **.** (ピリオド) 現行メッセージ。これはデフォルトです。 **first** フォルダー内の最初のメッセージ。 **last** フォルダー内の最後のメッセージ。 **new** 新たに作成するメッセージ。 **next** 現行メッセージの次のメッセージ。 **prev** 現行メッセージの直前のメッセージ。 デフォルトのメッセージは、現行フォルダー内の現行メッセージです。複数のメッセージを指定する場合は、最初に転送されるメッセージが現行メッセージになります。フォルダーを指定すると、指定したフォルダーが現行フォルダーになります。
-noannotate	元のメッセージに注釈を付けません。このフラグはデフォルトです。
-nodraftfolder	ドラフトを *UserMhDirectory*/**draft** ファイルに入れます。
-noedit	初期編集を抑制します。
-noformat	転送するメッセージを再フォーマットしません。このフラグがデフォルトです。
-noinplace	適切な注釈付けを強制しません。このフラグはデフォルトです。
-nowhatnowproc	**forw** コマンドの対話式処理を行いません。このフラグを指定すると、編集は行われません。
-volume *Number*	ダイジェストのボリューム番号を指定します。デフォルトのボリューム番号は、*UserMhDirectory*/**context** ファイル内の `DigestName-volume-list` エントリーの現行値です。
-whatnowproc *Program*	指定されたプログラムを始動して、転送作業を補助します。 **注:** *Program* に **whatnow** コマンドを指定すると、**forw** コマンドは **whatnow** というファイル名を持つプログラムではなく、内部 **whatnow** プロシージャーを始動します。

プロファイル・エントリー

以下のエントリーは、*UserMhDirectory*/**.mh_profile** ファイルに入力します。

項目	説明
`Current-Folder:`	デフォルトの現行フォルダーを設定します。
`Draft-Folder:`	ドラフトを入れるデフォルトのフォルダーを設定します。
`Editor:`	デフォルトのエディターを設定します。
`fileproc:`	メッセージをリファイルするプログラムを指定します。
`mhlproc:`	転送するメッセージのフィルターに使用するプログラムを指定します。
`Msg-Protect:`	新しいメッセージ・ファイル用の保護レベルを設定します。
`Path:`	*UserMhDirectory* を指定します。
`whatnowproc:`	「`What now?`」という質問のプロンプトを出すのに使用するプログラムを指定します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティ

ー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 現在のメッセージを転送するには、次のように入力します。

   ```
   forw
   ```

   システムは、ヘッダー・フィールドへの情報の入力を求めるプロンプトを表示します。フィールドをスキップするには、Enter キーを押します。`To:` フィールドへの情報の入力は必須です。システムは次のように応答します。

   ```
   ---------Enter initial text
   ```

   転送するメッセージのテキストの前に表示したいテキストを入力し、Ctrl-D キー・シーケンスを押します。転送するメッセージのテキストが表示され、「`What now?`」というプロンプトが表示されます。「`What now?`」プロンプトの後に `send` と入力すると、メッセージが転送されます。
2. `inbox` フォルダーからメッセージ `5` を転送するには、次のように入力します。

   ```
   forw    +inbox   5
   ```

ファイル

項目	説明
/etc/mh/digestcomps	**-digest** フラグを指定した場合の MH のデフォルトのメッセージ形式を定義します。
/etc/mh/mhl.forward	デフォルトの MH メッセージ・フィルターが入っています。
UserMhDirectory/**digestcomps**	**-digest** フラグを指定した場合のユーザーのデフォルトのメッセージ形式を定義します。(このフォーマットがある場合、MH のデフォルトのメッセージ・フィルターは指定変更されます。)
UserMhDirectory/**forwcomps**	ユーザーのデフォルトのメッセージ形式が入っています。
UserMhDirectory/**mhl.forward**	ユーザーのデフォルトのメッセージ・フィルターが入っています。(このフォーマットがある場合、MH のデフォルトのメッセージ・フィルターは指定変更されます。)
/usr/bin/forw	**forw** コマンドの実行可能フォームが入っています。
$HOME/.mh_profile	個々のユーザー用に MH をカスタマイズするファイルが入っています。
UserMhDirectory/**draft**	メッセージ編集用に作成されたドラフトが入っています。
/etc/mh/forwcomps	**forw** コマンドで作成するメッセージのコンポーネントを定義します。

関連情報:

anno コマンド

whatnow コマンド

mh_alias ファイル

メール・アプリケーション

---

fpm コマンド

目的

setuid または setgid アクセス権を持つ特権ユーザーが所有するコマンドとデーモンへのアクセス権を管理します。

構文

fpm [**-l** *level* [**-f** *file*] [[**-c**] [**-p**]] [**-v**]] | [**-s**] | [**-q**] | [**-?**]

説明

fpm コマンドを使用すると、管理者はオペレーティング・システムで多くのコマンドの setuid ビットと setgid ビットを使用不可にしてシステムを強化することができます。このコマンドは、特権ユーザーが所有するコマンドとデーモンから setuid アクセス権を除去することを目的としています。ただし、このコマンドをカスタマイズして、固有のコンピューター環境の特殊なニーズに対応することもできます。

基本 AIX オペレーティング・システム上の setuid プログラムは、強化のレベルのためにグループ化されています。このグループ化により、管理者はシステム環境に応じて強化のレベルを選択できます。また、**fpm** コマンドを使用して、環境内で使用不可にする必要があるプログラムのリストをカスタマイズできます。使用不可化のレベルを検討して、環境に適したレベルを選択する必要があります。

fpm コマンドでコマンドとデーモンの実行アクセス権を変更すると、非特権ユーザーがこれらのコマンドとデーモンまたはコマンドとデーモンの機能にアクセスできなくなります。また、これらのコマンドとデーモンを呼び出したり依存したりする他のコマンドも影響を受ける可能性があります。 **fpm** コマンドで変更されたアクセス権を持つコマンドとデーモンに依存するユーザー作成スクリプトは、非特権ユーザーがこれらを実行すると期待どおりに動作しません。コマンドとデーモンのデフォルトのアクセス権を変更する場合は、その効果と起こり得る影響についてよく考慮してください。

このコマンドを使用してクリティカルなコンピューター環境でコマンドとデーモンの実行アクセス権を変更する場合は、事前に適切なテストを実行する必要があります。実行アクセス権を変更した環境で問題が発生した場合は、デフォルトのアクセス権を復元し、このデフォルトの環境で問題を再作成して、その問題の原因が適切な実行アクセス権の不足ではないことを確認してください。

fpm コマンドには、**-l default** フラグを使用して元の AIX インストール済み環境のデフォルト・アクセス権を復元する機能があります。

また、**fpm** コマンドは、ファイルのアクセス権の状態をファイルの変更前にログに記録します。**fpm** ログ・ファイルは、**/var/security/fpm/log/*date_time*** ファイルに作成されます。必要に応じて、これらのログ・ファイルを使用して、前回保管されたログ・ファイルに記録されていたシステムのファイル・アクセス権を復元できます。

fpm コマンドがアクセス権を拡張したファイルで使用される場合、このコマンドはその拡張アクセス権を使用不可にします。ただし、**fpm** の呼び出し前に存在した拡張アクセス権のデータは拡張 ACL に保存されます。

カスタマイズされた構成ファイルを高、中、低、およびデフォルトの設定の一部として作成して設定することができます。ファイル・リストは、**/usr/lib/security/fpm/custom/high/*** ディレクトリー、**/usr/lib/security/fpm/custom/medium/*** ディレクトリー、および **/usr/lib/security/fpm/custom/default/*** ディレクトリーで指定できます。この機能を利用するには、**fpm** コマンドの内部リストに加えて、自動的に処理したいファイルのリストを含むファイルを作成します。**fpm** コマンドは実行されると、対応するカスタマイズ・ディレクトリー内のリストも処理します。カスタマイズされたファイルのフォーマットの例を確認するには、**/usr/lib/security/fpm/data/high_fpm_list** ファイルを表示します。デフォルトのフォーマットは、**/usr/lib/security/fpm/data/default_fpm_list.example** ファイルで表示できます。**-l low** フラグのカスタマイズの場合、**fpm** コマンドは **/usr/lib/security/fpm/custom/medium** ディレクトリーの同じファイルを読み取りますが、setgid アクセス権だけを除去します。その一方で、**-l medium** フラグは、setuid と setgid の両方のアクセス権を除去します。

fpm コマンドは、TCB 対応ホストでは実行できません。

フラグ

項目	説明
-l *level*	指定したレベルに応じてファイルのアクセス権を変更することを指定します。 **-l high** 高レベル・セキュリティー。このフラグは、コンピューター・システムに対して setuid と setgid のアクセス権を除去し、それらを高レベル・セキュリティーに分類します。このフラグは、デフォルトの入力として **/usr/lib/security/fpm/data/high_fpm_list** ファイルと **/usr/lib/security/fpm/custom/high/*.*** ファイル内のファイル・リストを使用します。ただし、**-f** フラグを使用して代替入力ファイルを選択できます。 **-l medium** 中レベル・セキュリティー。このフラグは、コンピューター・システムに対して setuid と setgid のアクセス権を除去し、それらを中レベル・セキュリティーに分類します。このフラグは、デフォルトの入力として **/usr/lib/security/fpm/data/med_fpm_list** ファイルと **/usr/lib/security/fpm/custom/med/*.*** ファイル内のファイル・リストを使用します。**-f** フラグを使用して代替入力ファイルを選択できます。 **-l low** 低レベル・セキュリティー。このフラグは、コンピューター・システムに対して setuid アクセス権のみを除去し、それらを低レベル・セキュリティーに分類します。このフラグは、デフォルトの入力として **/usr/lib/security/fpm/data/med_fpm_list** ファイルと **/usr/lib/security/fpm/custom/med/*.*** ファイル内のファイル・リストを使用します。**-f** フラグを使用して代替入力ファイルを選択できます。 **-l default** **fpm** コマンドで変更されたシステム・コマンドをデフォルトの出荷時のアクセス権に戻します (そのコマンドが高、中、または低のレベルを使用して変更された場合)。このオプションは、**/usr/lib/security/fpm/custom/default/*.*** ファイルを読み取り、そのファイルに定義されたアクセス権を設定します。
-s	**fpm** コマンドで前回行われた変更の状況を表示します。状況は **/usr/lib/security/fpm/data/status_fpm** ファイルに書き込まれます。セキュリティー・レベルは、1 から 5 (包括的) の整数で表されます。
-f *file*	ファイル・リストを指定してデフォルトの入力ファイルを上書きします。この場合、*file* パラメーターは、入力に使用されるファイルのリストを含むファイルの名前です。このフラグは、**-l high**|**medium**|**low**|**default** または **-c** フラグと一緒に使用する必要があります。高、中、低のレベルを使用する場合、入力ファイルのフォーマットは次のようになります。 *full_path*/*filename* 例: `/usr/sbin/foo` **-l default** フラグと一緒に使用される場合、入力ファイルのフォーマットは次のようになります。 *octet_permissions full_path*/*filename* *octet_permissions* 変数と *full_path* 変数の間にはスペースが 1 つ必要です。例: `0750 /usr/sbin/foo` **-f** フォーマットでは、影響を受けるファイル・リストの特定の制御が可能です。

項目	説明
-c	ファイルのアクセス権を検査します。ただし、アクションは実行しません。**fpm** コマンドは、非準拠のファイルが見つからなかった場合、0 を返します。1 つ以上のファイルに非準拠のアクセス権が含まれている場合、このオプションは非準拠のファイルをリストし、1 を返します。このフラグは、**-l** ***level*** オプションと一緒に使用する必要があります。例えば、**-c** フラグと **-l high** フラグを一緒に使用すると、**fpm** コマンドは **/usr/lib/security/fpm/data/high_fpm_list** ファイルにリストされているファイルを検査し、それらの setuid と setgid のアクセス権を除去します。**-f** ***file*** フラグは、**-c** オプションと一緒に使用することもできます。
-v	詳細出力。
-p	**fpm** コマンドが行う変更をプレビューします。ただし、アクションは実行しません。このフラグは、**-l** ***level*** フラグと一緒に使用する必要があります。
-q	終了モード。出力を最小限にして、警告を抑制します。
-?	使用方法に関するステートメントを表示します。

終了状況

項目	説明
0	成功です。
非ゼロ	失敗、または一部失敗。詳しくは、**-v** フラグを参照してください。

セキュリティー

fpm コマンドは、setuid と setgid アクセス権のコマンド数を削減します。

例

1. **fpm** コマンドの低レベル・セキュリティー設定を適用するには、次のように入力します。

   ```
   fpm –l low
   ```

   このコマンドは、**/usr/lib/security/fpm/custom/med/** ディレクトリー内のファイル・リストも処理します。
2. システム・コマンドが現在 **fpm** の低レベル・アクセス権に設定されているかを確認するには、次のように入力します。

   ```
   fpm –c –l low
   ```

   このコマンドは、不適合のアクセス権を持つファイルを報告します。
3. 元々の出荷時のデフォルト・アクセス権を復元するには、次のように入力します。

   ```
   fpm –l default
   ```

   このコマンドは、**/usr/lib/security/fpm/custom/default/** ディレクトリー内のファイル・リストも処理します。
4. ファイル・アクセス権を変更せずに、**fpm** コマンドの高レベル・セキュリティーにシステムを準拠させるために行うアクセス権の変更をリストまたはプレビューするには、次のように入力します。

   ```
   fpm -l high –p
   ```

   このコマンドは、**/usr/lib/security/fpm/custom/high/** ディレクトリー内のファイル・リストもプレビューします。
5. **fpm** コマンドの高レベル・セキュリティー設定を適用するには、次のように入力します。

   ```
   fpm –l high
   ```

このコマンドは、**/usr/lib/security/fpm/custom/high/** ディレクトリー内のファイル・リストも処理します。

6. **fpm** コマンドで変更されたシステムの現在の状況をリストするには、次のように入力します。

   ```
   fpm –s
   ```

7. **fpm -l** *level* コマンドが 2007 年 1 月 7 日 8:00 a.m. に実行された場合、影響を受けるファイルのアクセス権の状態は **fpm** コマンドによりその変更前に取り込まれています。ファイル・アクセス権を 2007 年 1 月 7 日 8:00 a.m. の状態に復元するには、次のように入力します。

   ```
   fpm –l default –f /var/security/fpm/log/01072007_08:00:00
   ```

ファイル

項目	説明
/usr/lib/security/fpm/data/default_list_example	デフォルトの出荷時のアクセス権とファイルが入っています。
/usr/lib/security/fpm/data/high_fpm_list	**-l high** フラグで変更可能なアクセス権を持つファイルのリストが入っています。
/usr/lib/security/fpm/data/med_fpm_list	**-l medium** または **-l low** フラグで変更可能なアクセス権を持つファイルのリストが入っています。
/usr/lib/security/fpm/custom/high/*	このディレクトリー内のファイルは、**-l high** レベルが選択された場合にユーザー構成の入力として使用できます。これらのファイルには、**fpm** コマンドが setuid と setgid のアクセス権を除去する元のファイルのリストが入っていなければなりません。
/usr/lib/security/fpm/custom/medium/*	このディレクトリー内のファイルは、高レベルのディレクトリーと同様に機能しますが、**-l medium** フラグおよび **-l low** フラグと一緒に使用されます。
/usr/lib/security/fpm/custom/default/*	このディレクトリー内のファイルは、高レベルのディレクトリーと同様に機能しますが、**-l default** フラグと一緒に使用されます。 **注:** これらのファイルは、**/usr/lib/security/fpm/data/default_list_example** ファイルと同じフォーマットでなければなりません。
/usr/lib/security/fpm/data/status_fpm	**fpm** コマンドの前回の実行で変更されたファイル・アクセス権の状況が入っています。
/var/security/fpm/log/*date_time*	**fpm** コマンドで変更された、コマンドの実行日時に対応するファイルのリストが入っています。このファイルは、**-f** フラグの入力ファイルとして使用してアクセス権をこのインスタンスに復元できます。

関連情報:

aixpert コマンド

Security Expert

---

asa、fpr コマンド

目的

ライン・プリンターの規則に従って FORTRAN ファイルを出力します。

構文

{ **asa** | **fpr** } [*File ...*]

説明

asa コマンドと **fpr** コマンドは、このオペレーティング・システムのライン・プリンター規則に従って FORTRAN ファイルを出力します。どちらのコマンドにもフィルターとしての機能があり、FORTRAN の紙送り制御規則に従ってフォーマットされたファイルを、ライン・プリンター規則に従ってフォーマットされたファイルに変換します。

File 変数は、**asa** コマンドおよび **fpr** コマンドが標準入力の代わりに読み取る入力ファイルの名前を指定します。 **asa** コマンドおよび **fpr** コマンドはこのファイルを読み取り、紙送り制御文字を認識可能なオペレーティング・システム文字で置換し、ファイルを標準出力に出力します。

どちらのコマンドも、入力ファイルの各行の最初の 1 文字を読み取り、その文字を解釈し、その文字の定義に従って行間隔を調整します。最初の文字が、ブランク、**0**、ダッシュ (**-**)、**1**、または正符号 (**+**) のいずれかであれば、どちらのコマンドも次の処理をします。

項目	説明
ブランク	キャリッジを 1 行先送りして入力行を出力します。
0	キャリッジを 2 行先送りして入力行を出力します。
-	キャリッジを 3 行先送りして入力行を出力します。
1	キャリッジを次ページの先頭に先送りします。
+	キャリッジを先送りしないで、出力ファイルの最初のスペースから入力行の出力を開始します。

これらのコマンドは、ブランク行は最初の文字がブランクの行であるものと解釈し、紙送り制御文字として現れるブランクを削除します。定義されている制御文字以外の文字で始まる行は、ブランク文字で始まるものとして扱われます。行の最初の 1 文字は出力されません。このような行が現れると、該当の診断が標準エラーに表示されます。

注**:** 170 文字を超える入力行の場合、結果は予測できません。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. 次のように **fpr** コマンドを使うと、 FORTRAN コンパイラーにより生成された `a.out` ファイル内の紙送り制御文字が、紙送り制御文字に変更され、結果のファイルが出力されます。

   ```
   a.out | fpr | qprt
   ```

2. 次のように **asa** コマンドを使うと、 **asa** コマンドにより `f77.output` ファイルが実行され、紙送り制御文字が FORTRAN からオペレーティング・システムに変更され、結果のファイルが出力されます。

   ```
   asa f77.output | qprt
   ```

ファイル

項目	説明
/usr/ucb/fpr	**fpr** コマンドが入っています。
/usr/bin/asa	**asa** コマンドが入っています。

---

frcactrl コマンド

目的

FRCA を制御し、構成します。

構文

frcactrl { **load** | **unload** }**frcactrl open** *Ip_Address Port* [*Virtual_Host*] *Server_Name Virtual_Root Log_File***frcactrl close** *Ip_Address Port* [*Virtual_Host*]**frcactrl loadfile** *Ip_Address Port* [*Virtual_Host*] *Document_Root File ...***frcactrl stats** [reset] [*Interval*]**frcactrl logging** *Ip_Address Port* [*Virtual_Host*] { on | off } [*Format*] [*CPU_Id*] **frcactrl** { **start** | **stop** } *Ip_Address Port* [*Virtual_Host*] **frcactrl revaltimeout** *Ip_Address Port* [*Virtual_Host*] [*Seconds*] **frcactrl pctonintr** [*Percentage*]**frcactrl set** { option=*value* }**frcactrl getfrcactrl default** [*option*]

説明

frcactrl コマンドは、FRCA カーネル拡張機能を制御し、構成します。カーネル拡張機能は、FRCA を使用したい Web サーバーを始動する前に読み込む必要があります。

サブコマンド

load FRCA カーネル拡張機能がロードされていない場合に、これをロードします。

unload
FRCA カーネル拡張機能がロードされている場合に、これをアンロードします。

open *Ip_Address Port* **[** *Virtual_Host* **]** *Server_Name Virtual_Root Log_File*
ポート *Port* 上に IP アドレスが *Ip_Address* で *Server_Name* という名前の FRCA インスタンスをオープンして構成します。 *Virtual_Root* パラメーターは、Web データが始動するディレクトリーを指定します。この要求は、*Log_File* で指定したファイルに記録されます。このファイル名は、完全修飾名にする必要があります。

注**:** FRCA は 1 つのログ・ファイルしかサポートしません。 FRCA を使用するシステムで複数の Web サーバーを実行する場合は、同一ファイルにすべての要求が記録されます。

close *Ip_Address Port* **[** *Virtual_Host* **]**
指定されている IP アドレスとポートに関連した FRCA インスタンスをクローズします。

loadfile *Ip_Address Port* **[** *Virtual_Host* **]** *Document_Root File* **...**
指定したファイルを FRCA またはネットワーク・バッファー・キャッシュにロードします。ここでは、以前に FRCA インスタンスがオープンされた IP およびポート番号を、ドキュメント・ルートおよびロードするファイルと共に指定します。

stats [reset] [*Interval* **]**
FRCA 統計情報を表示します。オプションの **reset** サブコマンドは、統計情報をゼロにクリアします。 *Interval* パラメーターで間隔 (秒数) を指定して、定期的に統計情報を表示することができます。

logging *Ip_Address Port* **[** *Virtual_Host* **] { on | off } [** *Format* **] [** *CPU_Id* **]**
指定した *Ip_Address* および *Port* にバインドされている FRCA インスタンスによってサービスされる要求のロギングをオンまたはオフにします。フォーマットは、CLF、V-CLF、ECLF (共通ログ・フォーマット、仮想ホスト & CLF、拡張 CLF) のいずれかです。マルチプロセッサー・マシンで、オプションの *CPU_Id* パラメーターを指定して、FRCA ロギング・スレッドを特定の CPU にバインドすることもできます。

start *Ip_Address Port* **[** *Virtual_Host* **]**
カーネル取得エンジンが、指定した IP およびポートに送信される要求をサービスすることができるようにします。

stop *Ip_Address Port* **[** *Virtual_Host* **]**
指定した IP およびポートについて、カーネル取得エンジンを使用不可にします。

revaltimeout *Ip_Address Port* **[** *Virtual_Host* **] [** *Seconds* **]**
指定したアドレスおよびポートで、FRCA インスタンスの妥当性再検査のタイムアウト値を変更します。タイムアウト値は秒で指定してください。

pctonintr [*Percentage* **]**
割り込みコンテキストに費やされる CPU 時間のパーセンテージを制御します。この値が低過ぎると、FRCA は、常に割り込みコンテキストで実行されるので、より頻繁に Web サーバーに要求を送信します。 100 以上の値を指定すると、FRCA は、FRCA キャッシュにキャッシュされているすべての要求を処理します。

set {option=*value***}**
指定された FRCA オプションに値を設定します。現在使用可能な唯一のオプションは **frca_hashsz** です。このオプションは、 FRCA ハッシュ・テーブル内のスロット数を指定された値に設定します。**frca_hashsz** のデフォルト値は 12841 です。この値を変更する場合は、ハッシュ・テーブル・エントリーのより均一な分散が行われるように、使用される値は素数でなければなりません。

get 使用可能なすべての FRCA オプションならびにその現在値を表示します。 **frca_hashsz** と呼ばれる唯一のオプションが現在存在します。

default [*option***]**
すべてのオプションの値を、オプション名を指定せずに使用する際にデフォルト値に設定します。オプション名を指定すると、指定されたオプションの値のみがデフォルト値に設定されます。

セキュリティー

RBAC ユーザーおよび Trusted AIX ユーザーへの注意: このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. **open** サブコマンドの使用例を次に示します。

   ```
   frcactrl open 9.1.1.1 80 ici imgcache01 /htdocs /logs/frca.log bin
   frcactrl open 9.1.1.2 80 ici imgcache02 /htdocs /logs/frca.log bin
   ```

   上の例で、「ici」は、ミラー imgcache01 または imgcache02 のいずれかにアクセスする際に使用される仮想ホスト名です。Web サーバーが特定の IP アドレスにバインドされていない場合は、IP アドレスは 0.0.0.0 の可能性があります。
2. IP アドレス 9.1.1.1 およびポート 80 に関連した FRCA インスタンスをクローズするには、次のように入力します。

   ```
   frcactrl close 9.1.1.1 80
   ```
3. URL /d および /e を使用して、ファイル /a/b/c/d および /a/b/c/e の内容をロードするには、次のように入力します。

   ```
   frcactrl loadfile /a/b/c /a/b/c/d e
   ```
4. FRCA 統計情報を表示するには、次のように入力します。

   ```
   frcactrl stats
   ```

これにより、FRCA 統計情報が表示されます。次のように表示されます。

```
      Total     Deferred       Cache       Cache     Resource
   Requests     Requests        Hits      Misses       Errors
-------------------------------------------------------------
 1024065396          227  1024065168           1            0
```

5. 次の例は、仮想ホスト「ici」に対する **start** サブコマンドの使用方法を示しています。

   ```
   frcactrl start 9.1.1.1 80 ici
   ```

   注**:** virtual host パラメーターはオプションです。
6. 仮想ホスト「ici」上の IP アドレス 9.1.1.1 のポート 80 に対してカーネル取得エンジンを使用不可にするには、次のように入力します。

   ```
   frcactrl stop 9.1.1.1 80 ici
   ```
7. 次の例は、IP アドレス 9.1.1.1 のポート 80 にある FRCA インスタンスの妥当性再検査タイムアウト値を 100 秒に設定します。

   ```
   frcactrl revaltimeout 9.1.1.1 80 100
   ```
8. CPU が割り込みコンテキストに CPU 時間の 98% を費やすようにするには、次のように入力します。

   ```
   frcactrl pctonintr 98
   ```
9. **frca_hashsz** オプションの値を 24499 に設定するには、次のように入力します。

   ```
   frcactrl set frca_hashsz=24499
   ```
10. **frca_hashsz** オプションの値をデフォルト値に設定するには、次のように入力します。

    ```
    frcactrl default frca_hashsz
    ```

ファイル

/usr/bin/frcactrl

---

from コマンド

目的

メールの送信者を判別します。

構文

from [**-d** *Directory*] [**-s** *Sender*] [*user*]

説明

from コマンドはメールボックス・ファイル内のメッセージ見出しを表示し、メールの送信者を表示します。 *user* を指定すると、自分のメールボックスではなく、*user* のメールボックスが検査されます (user のメールボックスの読み取り許可がある場合)。

フラグ

項目	説明
-d *Directory*	システム・メールボックス・ディレクトリーを指定します。
-s *Sender*	*Sender* が送信したメールのメッセージ・ヘッダーだけを出力します。

パラメーター

項目	説明
ユーザー	自分のメールボックスの代わりに検査される *user* のメールボックスを指定します (user のメールボックスの読み取り許可がある場合)。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. メールボックス内のメッセージ見出しを表示するには、次のように入力します。

   ```
   from
   ```

   送信者名とメッセージ日付が表示されます。
2. 特定のユーザーが送信したメールのメッセージ見出しを表示するには、次のように入力します。

   ```
   from -s dale
   ```

   この例では、ユーザー `dale` が送信したメッセージのメッセージ見出しだけを表示します。
3. 特定のユーザーのメールボックス内のメッセージ見出しを表示するには、次のように入力します。

   ```
   from   dawn
   ```

   この例では、ユーザー `dawn` のメールボックス内のメッセージ見出しを表示します (`dawn` のメールボックスの読み取り許可がある場合)。
4. `bob` が `jane` から受信したすべてのメッセージを表示するには、次のように入力します。

   ```
   from -d /var/spool/mail -s jane bob
   ```

   これにより、アクセス権がある場合は (例えば、root)、`bob` が `jane` から受信したすべてのメッセージを見ることができます。

ファイル

項目	説明
/var/spool/mail/*	すべてのユーザーが使うシステム・メールボックス。
/usr/bin/from	ユーザーのメールボックス・ファイル。

関連情報:

mail コマンド

メール・アプリケーション

fsck コマンド

目的

ファイルシステムの整合性を検査し、対話式に修復します。

構文

fsck [**-n**] [**-p**] [**-y**] [**-d***BlockNumber*] [**-f**] [**-i***i-NodeNumber*] [**-o** *Options*] [**-t***File*] [**-V** *VfsName*] [*FileSystem1 - FileSystem2 ...*]

説明

重要**:** システムが誤動作したときは、必ずファイルシステムに対して **fsck** コマンドを実行してください。修正作業を行うと、データの一部が失われることがあります。それぞれの整合性に関する修正のデフォルト・アクションは、オペレーターによる `yes` または `no` の入力を待機することです。該当するファイルシステムへの書き込み許可がない場合、実際の応答にかかわらず、**fsck** コマンドでは、デフォルトで `no` の応答になります。

注**:**

1. **fsck** コマンドは、マウント済みのファイルシステムの修正は行いません。
2. 修復以外の理由であれば、**fsck** コマンドをマウント済みのファイルシステムで実行できます。しかし、ファイルシステムがマウントされたときに誤ったエラー・メッセージが戻される場合があります。

fsck コマンドは整合性のないファイルシステムを検査して、対話式に修正します。ファイルシステムをマウントする場合は、事前にこのコマンドを実行してください。その場合、ファイルシステムが存在するデバイス・ファイルを読み取れなければなりません (例えば、**/dev/hd0** デバイス)。通常、ファイルシステムは整合性を持っているので、**fsck** コマンドは単にファイルシステム内のファイル数、使用ブロック数、フリー・ブロック数などを報告するだけです。ファイルシステムに不整合がある場合は、**fsck** コマンドは発見された不整合についての情報を表示し、修正の許可を求めるプロンプトを表示します。

fsck コマンドは修正作業にあたっては慎重で、正しいデータを失う恐れのあるアクションを避けようとします。しかし場合によっては、**fsck** コマンドが損傷のあるファイルの破壊を勧めることもあります。 **fsck** コマンドで必要な修復を実行できないようにすると、ファイルシステムの整合性が損なわれることがあります。整合性のないファイルシステムをマウントすると、システムがクラッシュすることがあります。

JFS2 ファイルシステムにスナップショットがある場合、**fsck** コマンドはそれを保存しようとします。このアクションが失敗した場合、スナップショットに、スナップされたファイルシステムの変更前のすべてのイメージが含まれていることは保証されません。 **fsck** コマンドは、スナップショットおよびスナップショット論理ボリュームを削除します。**fsck** コマンドがファイルシステムを変更した場合は、内部スナップショットが削除されます。

FileSystem パラメーターでファイルシステムを指定しないと、 **fsck** コマンドは **/etc/filesystems** ファイルに登録されたすべてのファイルシステムを検査して、 **check** 属性が真に設定されているかどうかを調べます。スタンザに次の 1 行を追加すると、この検査を実行できます。

```
check=true
```

ファイルシステムを **/etc/filesystems** ファイルにグループ化して、複数のファイルシステムで検査を行うこともできます。これを行うには、**/etc/filesystems** ファイルの中の check 属性を次のように変更してください。

```
check=Number
```

Number パラメーターは、**fsck** コマンドに、特定のファイルシステムがどのグループに含まれているかを通知します。共通ログ・デバイスを使用するファイルシステムは、同じグループに入れる必要があります。ファイルシステムの検査は、一度に 1 つずつ、グループ順に、次に **/etc/filesystems** ファイルに登録されている順に行われます。すべての `check=true` ファイルシステムはグループ 1 にあります。**fsck** コマンドは、コマンド・ラインまたは **/etc/filesystems** ファイル内で指定された順序に関係なく、どのファイルシステムよりも前にルート・ファイルシステムを検査しようとします。

fsck コマンドは次の不整合を検査します。

- 複数のファイルに割り当てられたブロックまたはフラグメント。
- 重複するブロック番号またはフラグメント番号がある i ノード。
- 範囲外のブロック番号またはフラグメント番号がある i ノード。
- ファイルに対するディレクトリーの参照数とファイルのリンク・カウントとの差。
- 不正に割り当てられているブロックまたはフラグメント。
- ディスク・マップ内で空きを示すマークが付いているブロック番号またはフラグメント番号が入っている i ノード。
- 破壊されたブロック番号またはフラグメント番号がある i ノード。
- i ノード内の最後のディスク・アドレスでないフラグメント。この検査は、圧縮されたファイルシステムには適用されません。
- フラグメントがある 32KB を超えるファイル。この検査は、圧縮されたファイルシステムには適用されません。
- サイズ検査:
  - 間違ったブロック数。
  - 512 バイトの倍数でないディレクトリー・サイズ。

  これらの検査は、圧縮されたファイルシステムには適用されません。
- ディレクトリー検査:
  - i ノード・マップ内で空きを示すマークが付いている i ノード番号があるディレクトリー・エントリー。
  - 範囲外の i ノード番号。
  - 存在していない、またはディレクトリー自身を指していないドット (.) リンク。
  - 存在していない、または親ディレクトリーを指していないドット・ドット (..) リンク。
  - 参照されていないファイル、または到達できないディレクトリー。
- 整合性のないディスク・マップ。
- 整合性のない i ノード・マップ。

孤立ファイルや孤立ディレクトリー (アクセス不可) は、指定により、ファイルシステムの / (ルート) ディレクトリー内の **lost+found** サブディレクトリーに入れると再接続できます。指定する名前は i ノード番号です。 **fsck** コマンドが孤立ファイルに再接続できるように指定しないと、コマンドは、ファイルを破棄する許可を要求します。

fsck コマンドは、メッセージの他に、終了値により検査と修復の結果を記録します。終了値は次の条件の合計になります。

項目	説明
0	検査済みファイルシステムにはすべて問題ありません。
2	検査または修復終了前に **fsck** コマンドが中断されました。
4	**fsck** コマンドがファイルシステムを変更しました。ユーザーは直ちにシステムを再始動させなければなりません。
8	ファイルシステムには未修復の損傷が含まれています。

fsck コマンドを実行するには、ファイル・システムの基礎となる論理ボリューム・デバイスへの排他的アクセスが必要です。基礎となるデバイスが使用不可であるために fsck が失敗した場合、デバイスが空いてオープンできるようになった後で fsck を再試行する必要があります。

システムをディスクからブートすると、ブート・プロセスは **/**、 **/usr**、**/var**、**/tmp** ファイルシステムで **-f** および **-p** フラグを指定した **fsck** コマンドを明示的に実行します。**fsck** コマンドがこのいずれかのファイルシステム上で失敗に終わると、システムはブートしません。このシステムをブートする前には、取り外し可能なメディアからブートし、保守作業を実行する必要があります。

/、**/usr**、**/var**、 **/tmp** において **fsck** コマンドが正常に実行されると、続いて通常のシステム初期設定が行われます。通常のシステム初期設定中に、 **-f** および **-p** フラグを指定した **fsck** コマンドが **/etc/rc** ファイルから実行されます。このコマンド・シーケンスは、**check** 属性が真に設定されている (`check=true`) すべてのファイルシステムを検査します。**/etc/rc** ファイルから実行された **fsck** コマンドでファイルシステムの整合性を保証できない場合でも、システムの初期設定は続行されます。ただし、整合性のないファイルシステムをマウントできないことがあります。マウントできなければ、システムの初期化が完了しないことがあります。

注**:** デフォルトでは、 **/**、**/usr**、**/var**、 **/tmp** ファイルシステムの **check** 属性は、 **/etc/filesystem** スタンザ内で、偽 (`check=false`) に設定されています。この属性は、次の理由で偽に設定されています。

1. ブート・プロセスは、**/**、 **/usr**、**/var**、 **/tmp** ファイルシステムに対して、明示的に **fsck** コマンドを実行します。
2. **/**、**/usr**、 **/var**、**/tmp** ファイルシステムは、 **/etc/rc** ファイルが実行されるときにマウントされます。**fsck** コマンドは、マウントされたファイルシステムを変更しません。また、マウントされたファイルシステム上で **fsck** コマンドを実行すると、信頼性のない結果が生成されます。

System Management Interface Tool (SMIT) の **smit fsck** 高速パスを使用して、このコマンドを実行できます。

フラグ

項目	説明
-d*BlockNumber*	指定されたディスク・ブロックへの参照を検索します。 **fsck** コマンドは、指定されたブロックを含むファイルを検出するたびに、関連する i ノード番号とすべてのパス名を表示します。 JFS2 ファイルシステムの場合、指定されたブロックを参照している i ノード番号は表示されますが、パス名は表示されません。
-f	高速検査を行います。通常は、正しい方法で電源を切らずにシステムを停止したために影響を受けるのは、そのときにマウントされているファイルシステムだけです。 **-f** フラグを指定すると、**fsck** コマンドは、正常にアンマウントされたファイルシステムの検査を行いません。 **fsck** コマンドは、ファイルシステム・スーパーブロック内の、 **s_fmod** フラグを検査することにより、正常にアンマウントされたファイルシステムを判別します。 このフラグはファイルシステムをマウントするたびに設定され、正常にアンマウントすると消去されます。ファイルシステムが正常にアンマウントされれば、何も問題が発生するようなことはありません。ほとんどのファイルシステムは正常にアンマウントされるため、このようなファイルシステムは検査されず、検査に要する時間が短縮できます。

項目	説明
-i*i-NodeNumber*	指定された i ノードへの参照を検索します。**fsck** コマンドは、指定された i ノードのディレクトリー参照を検出するたびに、参照先への絶対パス名を表示します。
-n	**fsck** コマンドのすべての質問に対して no という応答が与えられるものと見なします。指定したファイルシステムは書き込み用にオープンされません。
-o *Options*	**fsck** コマンドにコンマで区切られたオプションを渡します。以下のオプションは現在、JFS に対してサポートされます (これらのオプションは新規のファイルシステムでは古いものとなり、無視される可能性があります)。 **mountable** 対象のファイルシステムがマウント可能な (現在アンマウントされている) 場合、値 0 を戻して **fsck** コマンドを正常終了させます。ファイルシステムがマウント可能でない場合、**fsck** コマンドは値 8 を戻して終了します。 **mytype** 該当するファイルシステムが、 **/etc/filesystems** ファイル内で指定されたものか、コマンド・ライン上の **-V** フラグで指定されたものと同じタイプである場合に、 **fsck** コマンドを正常終了 (0) させます。その他の場合は 8 の値が戻されます。例えば、 **/** (ルート・ファイルシステム) がジャーナル・ファイルシステムであれば、 fsck -o mytype -V jfs / は 0 値で終了します。
-p	小さな問題についてのメッセージは表示せず自動的に修正します。このフラグは、**-y** フラグとは異なり、すべてをシステム側にまかせるわけではありません。システムの通常の始動時に自動検査を行う場合に役に立ちます。システムが自動的に実行される場合は常に、システム始動手続きの一部として、このフラグを使用する必要があります。1 次スーパーブロックが壊れている場合は、2 次スーパーブロックが確認され、1 次スーパーブロックにコピーされます。
-t*File*	**fsck** コマンドでテーブルを格納するための十分なメモリーが獲得できない場合に、*File* パラメーターを検査対象のファイルシステム以外のファイルシステム上のスクラッチ・ファイルとして指定します。 **-t** フラグを指定せず、 **fsck** コマンドがスクラッチ・ファイルを必要としている場合には、fsck コマンドはスクラッチ・ファイル名を求めるプロンプトを表示します。ただし **-p** フラグを指定している場合は、 **fsck** コマンドは正常に実行されません。スクラッチ・ファイルがスペシャル・ファイルでなければ、**fsck** コマンド終了時に除去されます。
-V *VfsName*	**/etc/filesystems** ファイルの代わりに、*VFSName* 変数で指定された仮想ファイルシステムの記述を使って、記述を決定します。コマンド・ラインに **-V** *VfsName* フラグを指定しないと、**/etc/filesystems** ファイルが検査され、一致するスタンザの **vfs=***Attribute* が、正しいファイルシステム・タイプであると見なされます。
-y	**fsck** コマンドが発行するすべての質問に対して、 yes という応答が与えられるものと仮定します。このフラグによって、 **fsck** コマンドは必要と考えられるすべての処理を行います。このフラグは、損傷の激しいファイルシステムだけに使用してください。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. すべてのデフォルトのファイルシステムを検査するには、次のように入力します。

   ```
   fsck
   ```

   このコマンドは、**/etc/filesystems** ファイル内の check=true のマークが付いているファイルシステムすべてを検査します。このフォーマットの **fsck** コマンドはファイルシステムに何らかの変更を行う前に、発行者に許可を求めます。
2. デフォルトのファイルシステムに関する小さな問題を自動的に修正するには、次のように入力します。

   ```
   fsck -p
   ```
3. 特定のファイルシステムを検査するには、次のように入力します。

```
fsck /dev/hd1
```

このコマンドは、**/dev/hd1** デバイス上でアンマウントされたファイルシステムを検査します。

ファイル

項目	説明
/usr/sbin/fsck	**fsck** コマンドが入っています。
/etc/filesystems	既知のファイルシステムをリストし、その特性を定義します。
/etc/vfs	仮想ファイルシステムのタイプに関する記述が入っています。
/etc/rc	システム始動時に実行されるコマンド (**fsck** コマンドなど) が入っています。

関連資料:

130 ページの『dfsck コマンド』

『fsdb コマンド』

関連情報:

filsys.h ファイル

ファイルシステム

---

fsck_cachefs コマンド

目的

CacheFS を用いてキャッシュされたデータの整合性を検査します。

構文

fsck_cachefs [**-m**] [**-o noclean**] *cache_directory*

説明

fsck コマンドの CacheFS バージョンは、キャッシュ・ディレクトリーの整合性を検査します。デフォルトでは、検出した CacheFS の問題を修正します。対話モードはありません。 CacheFS ファイルシステムで **fsck_cachefs** の呼び出しの最も可能性が高いのは、ブート時に **/etc/rc.nfs** のエントリーから実行される場合です。

フラグ

項目	説明
-m	検査はしますが、修復は行いません。
-o noclean	問題があると疑う理由がない場合でも、強制的にキャッシュを検査します。

例

強制的にキャッシュ・ディレクトリーの検査を行うには、次のように入力します。

```
fsck_cachefs -o noclean /cache3
```

---

fsdb コマンド

目的

ファイルシステムをデバッグします。

構文

fsdb *FileSystem* [**-**]

説明

fsdb コマンドを使用すると、*FileSystem* パラメーターで指定したファイルシステムを検査、変更、デバッグすることができます。このコマンドは、ブロック、i ノード、ディレクトリーなど、ファイルシステム・オブジェクトにアクセスできるようにします。 **fsdb** コマンドは、損傷したファイルシステムを検査し、補修します。ファイルシステムのキー・コンポーネントは、記号により参照することができます。この機能により、制御ブロック・エントリーを訂正したり、ファイルシステム・ツリーの階層を下降する手順を単純化できます。

ファイルシステムを検査するには、ブロック・デバイス名、ロウ・デバイス名、マウントされているファイルシステム名で指定します。マウントされているファイルシステム名を指定する場合、**fsdb** コマンドは、**/etc/filesystems** ファイルを読み取って、対応するファイルシステムを決定します。マウントされているファイルシステムを変更することはできません。

fsdb コマンドでは、 JFS ファイルシステムと JFS2 ファイルシステムにそれぞれ別のインターフェースが用意されています。以下で、JFS ファイルシステムでの **fsdb** の使用方法を説明します。 JFS2 サブコマンドの情報については、『JFS2 サブコマンド』のセクションを参照してください。

指定されているファイルシステムが JFS2 スナップショットである場合、**fsdb** コマンドは、スナップショット・スーパーブロック、スナップショット・マップ、ブロック・マップ xtree コピー、セグメント・ヘッダーの検査および修正を使用可能にします。 JFS2 スナップショット・サブコマンドについては、 JFS2 スナップショット・サブコマンドのセクションを参照してください。

fsbd コマンドのサブコマンドを使用すると、ファイルシステム内の情報にアクセスしたり、情報を表示または変更することができます。サブコマンドに入力する数値は、前に 8 進数を示す 0 または 16 進数を示す 0x を付けなければ、デフォルトで 10 進数と見なされます。すべてのアドレスは 16 進で出力されます。

fsdb fsdb コマンドは一度に 1 ブロックずつ読み書きするので、未加工の入出力にもブロック入出力にも機能します。

フラグ

項目	説明
-	i ノードおよびブロック・アドレスを検査するためのエラー検査ルーチンを使用不可にします。 **O** サブコマンドは、このルーチンをオンとオフの間で切り替えます。エラー検査ルーチンの実行中に、**fsdb** コマンドはスーパーブロックから重要なファイルシステム・データを読み取ります。取り出された情報により、**fsdb** コマンドは各種ファイルシステム・オブジェクトに正常にアクセスし、各種のエラー検査を実行することができます。

サブコマンド

fsdb サブコマンドは、ファイルシステム内の情報の位置を突き止め、表示または変更するための要求です。サブコマンドの主なカテゴリーは次のとおりです。

項目	説明
カテゴリー	機能
位置	ファイルシステム内の情報にアクセスします。
表示	ファイルシステム内の情報を表示します。
変更	ファイルシステム内の情報を変更します。

これ以外に、その他のサブコマンドがいくつかあります。

位置サブコマンド

位置サブコマンドには、次の 2 つのタイプがあります。

Number[**I** | **M** | **i** | **b**]

または

d*DirectorySlot*

第 1 のタイプは、番号と、その後にオプションとして続くアドレス指定で構成されています。アドレス指定は、先行の番号をどのように解釈するかを定義します。*Number* 変数の持つ 4 とおりの解釈に対して、4 つのアドレス指定があります。

項目	説明
I	i ノード・マップ・ブロック番号
M	ディスク・マップ・ブロック番号
i	i ノード番号
b	フラグメント番号

アドレス指定に応じて (またはアドレス指定がないことにより)、このタイプの位置サブコマンドは次の情報にアクセスします。

項目	説明
Number	*Number* 変数で指定された絶対バイト・オフセットにあるデータにアクセスします。
*MapBlockNumber***I**	*MapBlockNumber* 変数で指定された i ノード・マップ・ブロックにアクセスします。
*MapBlockNumber***M**	*MapBlockNumber* 変数で指定されたディスク・マップ・ブロックにアクセスします。
*InodeNumber***i**	*InodeNumber* 変数で指定された i ノードにアクセスします。
*FragmentNumber***b**	*FragmentNumber* 変数で指定されたファイルシステム・ブロックにアクセスします。フラグメント番号は、ブロック・アドレスとエンコードされた長さから成ります。完全なフラグメント・アドレスの長さは 32 ビットです。下位 28 ビットは開始フラグメント・アドレスです。このフラグメント長は、残りの 4 ビットでエンコードされます。すなわち、ブロック全体より少ないフラグメント数でエンコードされます。例えば、1024 バイトのフラグメントから成るファイルシステムでは、アドレス `0x2000010f` は 1KB のブロック番号 `0x10f` で始まり、長さが 2KB のブロックを指します。これに対して、512 バイトのフラグメントから成るファイルシステムでは、アドレス `0x2000010f` は 512 バイトのブロック 0x10f で始まり、長さが `0x10f` 3072 (512 * 6) バイトのブロックを指します。

第 2 のタイプの位置サブコマンドは、ディレクトリーのエントリーにアクセスするために使用します。このサブコマンドは、文字 **d** と後続のディレクトリー・スロット番号で構成されています。ディレクトリー・スロット番号は、それに関連する i ノードのブロックごとに 0 から始まります。

このタイプの位置サブコマンドは、次の情報にアクセスします。

項目	説明
d*DirectorySlot*	*DirectorySlot* 変数で索引付けされている現在の i ノードのディレクトリー・エントリーにアクセスします。割り当て済みのディレクトリー・エントリーのみを、この位置サブコマンドで操作することができます。

表示サブコマンド

アドレス指定に関する情報を表示するには、次のように表示機能と表示フォーマットをそれぞれ 1 つずつ合わせて構成される表示サブコマンドを使用します。

p[*Number*]{ **i** | **d** | **o** | **e** | **c** | **b** | **y** | **M** | **I** | **x** | **s** | **D** }

または

f[*Number*]{ **i** | **d** | **o** | **e** | **c** | **b** | **y** | **M** | **I** | **x** | **s** | **D** }

表示機能は次のとおりです。

項目	説明
p	汎用機能を示します。汎用表示サブコマンドを使用すると、現行アドレスに関するデータが表示されます。**p** 記号の後に数を入力すると、**fsdb** コマンドはエントリーの数を表示します。また、ブロック境界のオーバーフローを検出するための検査が実行されます。0 または * (アスタリスク) を入力すると、**fsdb** コマンドは現行フラグメントの終わりまでのすべてのエントリーを表示します。
f	ファイル機能を示します。ファイル表示サブコマンドを使用すると、現行 i ノードに関連するデータ・ブロックが表示されます。**f** 記号の後に数を入力すると、**fsdb** コマンドはファイル・ブロックを表示します。ブロック番号は 0 から始まります。ブロック番号によるフォーマットで表示されます。ブロック番号を指定せずに **f** を入力すると、**fsdb** コマンドはデフォルトで現行 i ノードのブロック 0 を表示します。

この 2 つの機能の表示フォーマットは次のとおりです。

項目	説明
i	i ノードとして表示
d	ディレクトリーとして表示
o	8 進ワードとして表示
e	10 進ワードとして表示
c	文字として表示
b	8 進バイトとして表示
y	16 進バイトとして表示
M	ディスク・マップ・エントリーとして表示
I	i ノード・マップ・エントリーとして表示
x	16 進ワードとして表示
S	単一間接ブロックとして表示
D	ダブル間接ブロックとして表示

選択した表示機能および表示フォーマットは、**fsdb** コマンドの処理中、明示的に変更されるまでは有効です。指定したアドレスが適切な境界内に入らなかった場合、境界が不適当であることを示すエラー・メッセージが表示されます。

Number、*MapBlockNumber***I**、*FragmentNumber* **b** 位置サブコマンドを使用して i ノード情報にアクセスすると、各バイト、ワード、ダブルワードを検査できます。次のサブコマンドのうちの 1 つを入力して希望する表示モードを選択します。

項目	説明
B	バイト・モードで表示を開始
D	ダブルワード・モードで表示を開始
W	ワード・モードで表示を開始

情報内を前後に動くことができます。境界は表示画面とともに前進し、最後に表示された項目のアドレスに残ります。INTERRUPT キーを押せば、いつでも出力を終了させることができます。次の記号を使うと、情報内を移動することができます。

項目	説明
+ *Number*	現在有効になっている指定した単位数だけ前方に移動します。
-*Number*	現在有効になっている指定した単位数だけ後方に移動します。

次の記号を使えば、現行アドレスを保存し、簡単にそのアドレスに戻ることができます。

項目	説明
>	現行アドレスを保管します。
<	以前に保管されたアドレスに戻ります。

サブコマンドの区切り文字としてドット、タブ、スペースを使えますが、16 進数として解釈される可能性のあるサブコマンドから 16 進数を区切る場合にのみ必要です。 Enter キーを押すと (ブランク行を入力する)、最後に表示されたデータ・タイプのサイズ分だけ、現行アドレスが増加されます。つまり、アドレスが次のバイト、ワード、ダブルワード、ディレクトリー・エントリー、または i ノードに設定され、ファイルシステムの領域を進むことができます。

fsdb コマンドは、データ・タイプに適切なフォーマットで情報を表示します。バイト、ワード、ダブルワードは 16 進アドレスとして表示され、その後にそのアドレスの位置にあるデータが 16 進数で表示され、さらに等価の 10 進数が括弧内に表示されます。 **fsdb** コマンドは、接尾部 **.B**、または **.D** を、バイトまたはダブル・ワードの表示値を示すアドレスの終わりに追加します。10 進 i ノード番号とエントリー名の文字表現の前に、ディレクトリー・スロット・オフセットとしてディレクトリーを表示します。また、各エレメントを記述するラベル付きフィールドとして i ノード番号を表示します。環境変数は日付フィールドおよび時間フィールドのフォーマットを制御します。

変更サブコマンド

フィールド指定 (i ノード内のフィールドとディレクトリー内のフィールドの場合) を使うと、アドレス指定に関連する情報を変更できます。新しい値を割り当てる一般的なフォーマットは *mnemonic operator new-value* です。ここで、*mnemonic* パラメーターは次のリストで説明するいずれかのフィールドを表します。

次のニーモニックは、i ノード番号のフィールド名に使用され、現在作業中の i ノードを指します。

項目	説明
md	許可モード
ln	リンク・カウント
uid	ユーザー番号
gid	グループ番号
sz	ファイル・サイズ
a*Number*	データ・ブロック番号 (0 から 8)。この場合の *Number* パラメーターは位置サブコマンドでもかまいません。
at	アクセス時間
mt	変更時間
maj	メジャー・デバイス番号

項目	説明
min	マイナー・デバイス番号

次のニーモニックは i ノードおよびディスク・マップに関するものです。

項目	説明
mf	未使用マップ数
ms	マップ・サイズ
mp	永久割り当てビットマップ
mw	作業中割り当てビットマップ

次のニーモニックはディレクトリー内のフィールド名に使用されます。

項目	説明
rl	ディレクトリー・エントリー・レコードの長さ
nl	ディレクトリー名の長さ
nm	ディレクトリー名

Operator パラメーターの有効な値は次のとおりです。

注**:** ファイルシステムは変更する前にアンマウントする必要があります。

項目	説明
=	*New-Value* パラメーターの値を、指定された *Mnemonic* パラメーターに割り当てします。
=+	*Mnemonic* パラメーターの値を、指定された *New-Value* パラメーターの値だけ増やします。デフォルトの *New-Value* パラメーターの値は 1 です。
=-	*Mnemonic* の値を指定された *New-Value* の値だけ、減らします。デフォルトの *New-Value* の値は 1 です。
="	*New-Value* パラメーターで指定された文字列を、指定された *Mnemonic* パラメーターに割り当てます。ディレクトリーの現在の表示フォーマットが、 **d** のアドレス指定である場合、ニーモニックが指定されていないと、ディレクトリー名が変更されます。新しいディレクトリー名は前のディレクトリー名よりも長くすることはできません。

その他のサブコマンド

その他のサブコマンドは次のとおりです。

項目	説明
q	終了します。

項目	説明
x*n*	ディレクトリーを *n* バイト拡張します。 *n* とディレクトリーの現行サイズの和が、現行ディレクトリーのフラグメントのサイズ (バイト) より大きくならないように指定します。
!	シェルにエスケープします。
O	エラー検査を切り替えます。

JFS2 サブコマンド

これらのサブコマンドは、その完全名を指定して、あるいは名前のサブセットを使用することによって、入力できます。少なくとも、太字の部分は入力しなければなりません。

項目	説明
a[lter] <block> <offset> <hex string>	ディスク・データを変更します。
b[map] [<block number>]	ブロック割り当てマップを表示します。
dir[ectory] <inode number> [<fileset>] [R]	ディレクトリー・エントリーを表示します。
d[isplay] [<block> [<offset> [<format> [<count>]]]]	データを表示します。
dt[ree] {<block number> ｜<inode number>{a ｜ f } }	dtree ノードを表示します。
h[elp] [<command>]	サブコマンド上でヘルプを提供します。
ia[g] [<IAG number>] [a ｜ <fileset>]	IAG ページを表示します。
im[ap] [a ｜ <fileset>]	i ノード割り当てマップを表示します。
i[node] [<inode number>] [a ｜ <fileset>]	i ノードを表示します。
q[uit]	fsdb を終了します。
su[perblock] [p ｜ s]	スーパーブロックを表示します。
x[tree] {<block number> ｜ <inode number>{a ｜ f}}	xtree ノードを表示します。

a[lter] <block> <offset> <hex string>

それぞれの意味は次のとおりです。

項目	説明
<block>	ブロック番号 (10 進数)
<offset>	ブロック内のオフセット (16 進数)
<hex string>	16 進数の文字列

ディスク・データを変更します。 <hex string> の桁数は、偶数でなければなりません。

b[map] [<block numbers>]

ブロック割り当てマップを表示します。

<block number> このブロック番号が記述されている **dmap** ページを表示します。

サブコマンド:

項目	説明
m	現行ノードを変更します。
u	上位レベル **bmap** ページに移動します。
l	左側の兄弟に移動します。
r	右側の兄弟に移動します。
w	wmap を表示します。
p	pmap を表示します。
s	stree を表示します。
x	サブコマンド・モードを終了します。

dir[ectory] <inode number> [<fileset>] [R]

項目	説明
<inode number>	ディレクトリーの i ノード番号 (10 進数)
<fileset>	番号。現在の番号はゼロでなければなりません。
R	すべてのサブディレクトリーを繰り返しリストします。

ディレクトリー・エントリーを表示します。

d[isplay] [<block> [<offset> [<format> [<count>]]]]

項目	説明
<block>	ブロック番号 (10 進数)
<offset>	ブロック内のオフセット (16 進数)
<format>	データを表示する形式 (下記を参照のこと)
<count>	表示するオブジェクトの番号 (10 進数)

各種の形式でデータを表示します。

形式は、以下のいずれかになります。

項目	説明	
a	ASCII	
i	i ノード	struct dinode
I	i ノード割り当てマップ	iag_t
s	スーパーブロック	struct superblock
x	16 進数	

dt[ree] {<block number> |<inode number>{a | f } }

項目	説明
<block number>	dtree ページがあるブロック番号
<inode number>	ディレクトリーの i ノード番号 (10 進数)
{a | f}	'a' は、i ノード番号が集約 i ノードであることを示します。'f' は、 i ノード番号がファイルセット i ノードであることを示します。

ディレクトリー btree の root を表示し、btree をナビゲートするサブコマンド・モードに入ります。

サブコマンド:

項目	説明
m	現在のノードを変更します。
f	フリー・リスト・エントリーを walk します。
s	指定されたスロット・エントリーを表示します。
[0-9]+	指定された stbl エントリーを表示します。
t	フォーマット済み stbl を表示します。
u	親ノード (親ディレクトリーではない) に移動します。
d	下位ノードに移動します。
x	サブコマンド・モードを終了します。

h[elp] [<command>]

項目	説明
<command>	コマンド名

ヘルプ・テキストを表示します。パラメーターがない場合はすべてのコマンドを表示します。

ia[g] [<IAG number>] [a | <fileset>]

項目	説明
<IAG number>	IAG 番号 (10 進数)
a	集約 i ノード・テーブルを使用します。
<fileset>	ファイルセット番号 (現在のファイルセット番号はゼロでなければなりません)。

iag 情報を表示し、サブコマンド・モードに入ります。

サブコマンド:

項目	説明
e	i ノード範囲マップを表示または変更します。
m	iag を変更します。
p	永続マップを表示または変更します。
w	作業マップを表示または変更します。

im[ap] [a | <fileset>]

項目	説明
a	集約 i ノード・テーブルを使用します。
<fileset>	ファイルセット番号 (現在のファイルセット番号はゼロでなければなりません)。

指定された i ノード・マップを表示し、サブコマンド・モードに入ります。

サブコマンド:

項目	説明
e	i ノード範囲マップを表示または変更します。
m	iag を変更します。
p	永続マップを表示または変更します。

i[node] [<inode number>] [a | <fileset>]

項目	説明
<inode number>	i ノード番号 (10 進数)
a	集約 i ノード・テーブルを使用します。
fileset	ファイルセット番号 (現在のファイルセット番号はゼロでなければなりません)。

i ノード情報を表示し、サブコマンド・モードに入ります。

サブコマンド:

項目	説明
m	i ノードを変更します。
t	i ノードの b-tree を表示または変更します。
e	i ノードの EA を表示または変更します。

注**: fsdb** コマンドは、**v1** と **v2** の両方の拡張属性フォーマットを理解します。 EA 表示時の動作は、i ノードが表示されるフォーマットによって異なります。

v1 の場合は、i ノードの EA の表示後、その pxdTable または eaDirectory エントリーを変更できます。変更オプションを指定してから、 pxdTable または eaDirectory インジケーターおよびテーブルに対するオフセットを指定してください。

v2 の場合は、EA は **dtree** サブコマンドのフォーマットを使用して表示されます。その後で、EA についてさらに処理できるように、すべての **dtree** サブコマンドが使用可能になります。

q[uit] fsdb を終了します。

su[perblock] [p | s]

項目	説明
p	1 次スーパーブロックを表示します。
s	2 次スーパーブロックを表示します。

スーパーブロック・データを表示します。

x[tree] {<block number> | <inode number>{a | f}}

項目	説明
<block number>	ブロック番号 (10 進数)
<inode number>	i ノード番号
{a | f}	'a' は、i ノード番号が集約 i ノードであることを示します。'f' は、 i ノード番号がファイルセット i ノードであることを示します。

xtree のノードの 1 つを表示し、xtree をナビゲートするサブコマンド・モードに入ります。

サブコマンド:

項目	説明
m	現在のノードを変更します。
u	親ノードに移動します。
d	下位ノードに移動します。
n	右側の兄弟に移動します。
p	左側の兄弟に移動します。
s	表示する xad エントリーを選択します。
x	サブコマンド・モードを終了します。

JFS2 スナップショット・サブコマンド

これらのサブコマンドは、その完全名を指定して、あるいは名前のサブセットを使用することによって、入力できます。少なくとも、太字の部分は入力しなければなりません。

項目	説明
a[lter] <block> <offset> <hex string>	ディスク・データを変更します。
b[map]	ブロック・マップ xtree コピーを表示します。
d[isplay] [<block> [<offset> [<format> [<count>]]]]	データを表示します。
h[elp] [<command>]	サブコマンド上でヘルプを提供します。
q[uit]	fsdb を終了します。
st[able] [<block number>]	要約スナップショット・テーブルを表示します。
s[map] <block number>	スナップショット・ビットマップを表示します。
su[perblock]	スーパーブロックを表示します。

a[lter] <block> <offset> <hex string>

それぞれの意味は次のとおりです。

項目	説明
<block>	ブロック番号 (10 進数)
<offset>	ブロック内のオフセット (16 進数)
<hex string>	16 進数の文字列

ディスク・データを変更します。 <hex string> の桁数は、偶数でなければなりません。

b[map]

ブロック・マップ xtree コピーを表示します。

d[isplay] [<block> [<offset> [<format> [<count>]]]]

項目	説明
<block>	ブロック番号 (10 進数)
<offset>	ブロック内のオフセット (16 進数)
<format>	データを表示する形式 (下記を参照のこと)
<count>	表示するオブジェクトの番号 (10 進数)

各種の形式でデータを表示します。

形式は、以下のいずれかになります。

項目	説明
a	ASCII
s	スナップショット・セグメント・ヘッダー
t	スナップショット・テーブル・ページ
x	xtree ページ

h[elp] [<command>]

項目	説明
<command>	コマンド名

サブコマンド上でヘルプを提供します。

q[uit] fsdb を終了します。

st[able] [<block number>]
それぞれの意味は次のとおりです。

項目	説明
<block number>	ブロック番号 (10 進数)

要約スナップショット・テーブルを表示します。

s[map] [<block number>]
それぞれの意味は次のとおりです。

項目	説明
<block number>	ブロック番号 (10 進数)

スナップショット・ビットマップを表示します。

su[perblock]

スーパーブロックを表示します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

次の例は、JFS ファイルシステム上で **fsdb** コマンドを始動した後に使用できる、サブコマンドを示したものです。

1. i ノードを表示するには、次のように入力します。

   ```
   386i
   ```

   このコマンドは、i ノード 386 を i ノード・フォーマットで表示します。これが現行 i ノードとなります。
2. 現在の i ノードのリンク・カウントを、値 4 に変えるには、次のように入力します。

   ```
   ln=4
   ```
3. 現在の i ノードのリンク・カウントを、値 1 だけ増やすには、次のように入力します。

   ```
   ln=+1
   ```
4. 現在の i ノードに関連するファイルの部分を表示するには、次のように入力します。

   ```
   fc
   ```

   このコマンドを使用すると、現在の i ノードに関連したファイルのブロック 0 が、 ASCII バイトで表示されます。
5. ディレクトリーのエントリーを表示するには、次のように入力します。

   ```
   2i.fd
   ```

   このコマンドは、現在の i ノードをルート i ノード (i-node 2) に変更して、その i ノードに関連する最初のブロック内のディレクトリー・エントリーを表示します。最後に表示される 1 つ以上のエントリーには、 i ノード番号 0 (ゼロ) が付いている場合があります。これらは、未使用のディレクトリー・ブロックです。このようなエントリーは、次の例のように操作することはできません。
6. ディレクトリー・ツリーの下位レベルに移動するには、次のように入力します。

   ```
   d5i.fc
   ```

   このコマンドは、現在の i ノードを、ディレクトリー・エントリー 5 に関連する i ノードに変更します。それから、そのファイルの最初のブロックを ASCII テキスト (fc) として表示します。ディレクトリー・エントリーは、0 から始まる番号が付きます。
7. ブロック番号が分かっているブロックを表示するには、次のように入力します。

   ```
   1b.p0o
   ```

   このコマンドは、ファイルシステムのスーパーブロック (ブロック 1) を、 8 進数で表示します。
8. ディレクトリー・エントリーの i ノードを変更するには、次のように入力します。

   ```
   2i.a0b.d7=3
   ```

このコマンドは、ルート・ディレクトリー (2i) 内の、ディレクトリー・エントリー 7 の i ノードを、 3 に変更します。この例では、複数の操作を 1 行にまとめる方法も示しています。

9. ディレクトリー・エントリーのファイル名を変更するには、次のように入力します。

```
d7.nm="chap1.rec"
```

このコマンドは、ディレクトリー・エントリー 7 の名前フィールドを、 chap1.rec に変更します。

10. 現在の i ノードに関連するファイルの任意のブロックを表示するには、次のように入力します。

```
a2b.p0d
```

このコマンドは、現在の i ノードのブロック 2 をディレクトリー・エントリーとして表示します。

11. ブロック 7 にある単一の間接ブロックの内容を表示するには、次のように入力します。

```
7b. p0S
```

このコマンドは、ブロック 7 に単一の間接ブロックを持っている i ノードに割り当てられたブロック数を表示します。

12. ディスク・マップの最初のページを表示するには、次のように入力します。

```
OM
```

13. 永続ブロック割り当てマップの最初の 10 ワードを 16 進数で表示するには、次のように入力します。

```
mp1.p10x
```

このコマンドは現行アドレスでの割り振りビットマップを表示します。例えば、0M。

以下の例では、JFS2 ファイルシステム上で使用できるサブコマンドの一部を示しています。

重要: JFS2 サブコマンドを使用してファイルシステムを変更しないでください。

1. i ノードを表示するには、次のように入力します。

```
inode 2
```

このコマンドは、i ノード 2 を i ノード・フォーマットで表示します。

2. ディレクトリーのエントリーを表示するには、次のように入力します。

```
dir 2
```

このコマンドは、i ノード 2 に関連付けられたディレクトリー・エントリーを表示します。

3. ブロック番号が 0x1000 のブロックを表示するには、次のように入力します。

```
display 0x1000
```

このコマンドは、ファイルシステムのブロックを 16 進形式で表示します。

ファイル

項目	説明
/usr/sbin	**fsdb** コマンドが入っています。
/etc/filesystems	ファイルシステムに関する情報が入っています。

関連資料:

130 ページの『dfsck コマンド』

関連情報:

dir コマンド

filsys.h ファイル

read サブルーチン

---

fsplit コマンド

目的

FORTRAN ソース・コードを別個のルーチン・ファイルに分割します。

構文

fsplit [**-e** *SubprogramUnit*] ... [*File*]

説明

fsplit コマンドは、FORTRAN ソース・コードが入っているファイルまたは標準入力を入力として取り、*name***.f** というフォーマットの別個のルーチン・ファイルに分割します。*name* はプログラム・ユニット (関数、サブルーチン、ブロック・データ、プログラムなど) の名前になります。

名前なしブロック・データ・サブプログラムの名前は、*blkdtaNNN***.f** というフォーマットになります。NNN は 3 桁の数字で、この名前のファイルはまだ存在しません。名前が指定されていないメインプログラムの場合、その名前は *mainNNN***.f** の形式となります。プログラム・ユニットの分類でエラーが発生するか、*name***.f** が既に存在する場合、そのプログラム・ユニットは *zzzNNN***.f** というフォーマットのファイルに入れられます。この場合、*zzzNNN***.f** はまだ存在しません。

注**: fsplit** コマンドは、サブプログラム名がサブプログラム・ユニットの最初の非コメント行にあることを前提としています。ソースが標準フォーマットでないと、コマンドが混乱し、予期しない結果になる恐れがあります。

フラグ

項目	説明
-e *SubprogramUnit*	指定したサブプログラム・ユニットだけを別個のファイルに分割します。通常、各サブプログラム・ユニットは別個のファイルに分割されます。 **-e** フラグは、名前付きの主プログラムおよびブロック・データ・サブプログラムにのみ使用できます。 **-e** オプションで指定した名前が見つからないと、標準エラーに診断が書き出されます。

例

次の **fsplit** コマンドは、サブプログラム `readit` および `doit` を別個のファイルに分割します。

```
fsplit -e readit -e doit prog.f
```

ファイル

項目	説明
/usr/bin/fsplit	**fsplit** コマンドが入っています。

関連情報:

asa コマンド

struct コマンド

---

ftp コマンド

目的

ローカル・ホストとリモート・ホスト間でファイルを転送します。

構文

ftp [**-d**] [**-D** *DataConnTimeOut*] [**-g**] [**-i**] [**-n**] [**-v**] [**-f**] [**-K**] [**-k** *realm*] [**-q**] [**-C**] [**-s**] [**-M**] [*HostName* [*Port*]] [**-H**]

説明

ftp コマンドは、ファイル転送プロトコル (FTP) を使って、ローカル・ホストとリモート・ホストの間で、または 2 つのリモート・ホストの間でファイルを転送します。 **ftp** コマンドのリモート実行はお勧めしません。

FTP プロトコルを使うと、異なるファイルシステムを使っているホスト間でデータ転送が可能になります。このプロトコルはデータ転送には高い柔軟性を持っていますが、それぞれのファイルシステムに固有のファイル属性 (ファイルの保護モードまたは変更時間など) を保存するようには設計されていません。さらに、このプロトコルはファイルシステムの構造全般についてほとんど仮定を行わず、また、サブディレクトリーを再帰的にコピーするような機能を提供することもありません。

注**:** システム間でファイルを転送する場合にファイル属性を保持したり、サブディレクトリーを再帰的にコピーする必要がある場合は、**rcp** コマンドを使用してください。

サブコマンドの発行

`ftp>` プロンプトで、サブコマンドを入力して、リモート・ディレクトリーのリスト表示、ローカルおよびリモートの現行ディレクトリーの変更、単一要求での複数ファイルの転送、ディレクトリーの作成と除去、シェル・コマンドの実行のためのローカル・シェルへのエスケープなどのタスクを実行することができます。各サブコマンドの説明については、 サブコマンドのセクションを参照してください。

ftp コマンドを実行して、リモート・ホストの *HostName* パラメーターを指定しないと、**ftp** コマンドは即時に `ftp>` プロンプトを表示し、**ftp** サブコマンドを待ちます。リモート・ホストに接続するには、 **open** サブコマンドを実行します。**ftp** コマンドは、リモート・ホストに接続すると、ログイン名とパスワードを入力するためのプロンプトを出し、その後 `ftp>` プロンプトを再度表示します。リモート・ホスト側でログイン名のパスワードが定義されていないと、**ftp** コマンドは失敗に終わります。

ftp コマンド・インタープリターは、`ftp>` プロンプトで入力されたすべてのサブコマンドを処理しますが、次のようなほとんどのファイル転送プログラムでは使用できない機能を備えています。

- **ftp** サブコマンドのファイル名パラメーターを処理する。

- サブコマンドのグループを収集して単一のサブコマンド・マクロを作成する。
- **$HOME/.netrc** ファイルからマクロをロードする。

これらの機能を使用すると、反復タスクを単純化して、**ftp** コマンドを不在モードで使用することができます。

このコマンド・インタープリターは、ファイル名パラメーターを次の規則に従って処理します。

- パラメーターとして - (ハイフン) を指定すると、読み取り操作には標準入力 (stdin)、書き込み操作には標準出力 (stdout) が使用されます。
- 上記の検査が適用されず、ファイル名拡張機能が使用可能であれば (**-g** フラグ、または **glob** サブコマンドのセクションを参照)、インタープリターは C シェルの規則に従ってファイル名を拡張します。グロビング機能が使用可能であり、単一のファイル名を期待するサブコマンドにパターン・マッチング文字が使用されていると、予期しない結果が起こることがあります。

  例えば、**append** サブコマンドと、 **put** サブコマンドは、ファイル名の拡張を行った後、最初に生成されたファイル名だけを使用します。 **cd**、 **delete**、 **get**、 **mkdir**、 **rename**、 **rmdir** など、その他の **ftp** サブコマンドは、ファイル名の拡張を行わず、パターン・マッチング文字をそのまま採用します。
- **get**、**put**、**mget**、および **mput** サブコマンドでは、インタープリターがローカルとリモートの異なるファイル名構文のスタイルを変換し、マップする機能を提供します (**case**、**ntrans**、および **nmap** サブコマンドを参照)、また、ローカル・ファイル名が固有でない場合、これを変更する機能も提供します (**runique** サブコマンドを参照)。 さらに、**ftp** コマンドは、リモート・ファイル名が固有のファイル名でない場合、リモート **ftpd** サーバーに指示を送って、この名前を変更することができます (**sunique** サブコマンドを参照)。
- ブランク文字を含むパラメーターを指定するには、二重引用符 (" ") を使用します。

  注**: ftp** コマンド・インタープリターはパイプをサポートしません。また、すべてのマルチバイト文字ファイル名がサポートされるとは限りません。

ftp セッションを対話式に実行しているときに終了するには、`ftp>` プロンプトで、**quit** サブコマンドまたは **bye** サブコマンドを使用するか、ファイルの終わり (Ctrl-D) キー・シーケンスを入力します。ファイル転送を完了前に終了するには、割り込みキー・シーケンスを押します。デフォルトの割り込みキー・シーケンスは Ctrl-C です。このキー・シーケンスを再定義するには、**stty** コマンドを使用します。

通常、**ftp** コマンドは (ローカル・ホストからリモート・ホストへ) 送られる転送を即座に停止します。**ftp** コマンドは、リモート・ホストから送信され、ローカル・ホストで受信されるデータ転送を停止するのに、リモート FTP サーバーに FTP ABOR 命令を送信し着信するファイル転送パケットをすべて廃棄して、リモート・サーバーが送信を停止するといった方法をとります。リモート・サーバーが ABOR 命令をサポートしていなければ、リモート・サーバーが要求されたファイルをすべて送信し終わるまで、**ftp** コマンドは `ftp>` プロンプトを表示しません。また、リモート・サーバーが予期しない動作を行う場合は、ローカル **ftp** プロセスを終了しなければならないことがあります。

セキュリティーと自動ログイン

標準が現行認証メソッドである場合

ftp コマンドはさらに、リモート・ホストにパスワードを送ってセキュリティーを提供し、自動ログイン、ファイル転送、ログアウトを許可します。

ftp コマンドを実行してリモート・ホストのホスト名 (*HostName*) を指定すると、**ftp** コマンドは直ちに指定されたホストへの接続を設定しようとします。 **ftp** コマンドは、正常に接続すると、現行ディレクトリ

ーまたはホーム・ディレクトリー内でローカル **$HOME/.netrc** ファイルを検索します。ファイルがあれば、**ftp** コマンドはそのファイル内で、ログイン処理を開始するエントリーと、リモート・ホストのコマンド・マクロ定義を検索します。 **$HOME/.netrc** ファイルまたは自動ログイン・エントリーが存在しない場合、またはユーザーのシステムが **securetcpip** コマンドによって保護されている場合は、**ftp** コマンドはユーザーに対してユーザー名とパスワードの入力を指示します。これは *HostName* パラメーターがコマンド・ラインで指定されているかどうかにかかわらず行われます。

注**:** キューイング・システムでは、マルチバイトのホスト名はサポートされません。

ftp コマンドは、指定されたホスト用の **$HOME/.netrc** 自動ログイン・エントリーを見つけると、そのエントリー内の情報を使用してリモート・ホストにログインしようとします。さらに、**ftp** コマンドはその自動ログイン・エントリー内で定義されたコマンド・マクロをすべてロードします。場合によっては (例えば、必要なパスワードが自動ログイン・エントリーにリスト表示されていない場合)、 **ftp** コマンドは、`ftp>` プロンプトを表示する前に、パスワードを求めるプロンプトを出します。

ftp コマンドは自動ログインを完了すると、**init** マクロが自動ログイン・エントリー内で定義されていれば、それを実行します。**init** マクロが存在しないか、その中に **quit** または **bye** サブコマンドが入っていない場合、**ftp** コマンドは `ftp>` プロンプトを表示して、サブコマンドを待ちます。

注**:** プロンプトまたは **$HOME/.netrc** ファイル内で指定するリモート・ユーザー名は、必ず存在し、かつリモート・ホストで定義されたパスワードを持っていなければなりません。そうでない場合、**ftp** コマンドは失敗に終わります。

Kerberos 5 が現行認証メソッドである場合

ftp コマンドは、ftp 仕様への拡張機能を、IETF ドラフト文書「draft-ietf-cat-ftpsec-09.txt」で定義された通りに使用します。 FTP セキュリティー拡張機能は、Generic Security Service API (GSSAPI) のセキュリティー・メカニズムを用いてインプリメントされます。GSSAPI は、基礎となるセキュリティーおよび通信メカニズムとは別のサービスを提供します。GSSAPI は rfc 1508 および 1509 で定義されます。

ftp コマンドは、**ftpd** デーモンを用いて認証する AUTH と ADAT コマンドを使用します。 AUTH と ADAT の両方のコマンドが Kerberos 認証をサポートしている場合、それらはローカル・ユーザー DCE 認証を用いて、リモート・システム上のユーザーの認証を行います。これが失敗し、両システムで標準の認証が構成される場合は、上記のプロセスが使用されます。

HostName パラメーターはファイルの転送先となるホスト・コンピューターの名前です。オプションの *Port* パラメーターにより、転送時に使われるポートの ID を指定します。(デフォルトのポートは **/etc/services** ファイルで指定されます。)

注**:** レジストリーの値が現行の認証方式に正しく設定されている場合、FTP 認証はアクティブなディレクトリー・パスワードを使用して機能します。レジストリーの値がヌルに設定されている場合は、ファイルのデフォルト値 (ローカル・ユーザー認証) が使用されます。

トランスポート層セキュリティーのサポート

ftp コマンドは、RFC 4217 で定義されているトランスポート層セキュリティー (TLS) をサポートします。TLS は、クライアントとサーバー間で安全な通信を行うための暗号プロトコルです。

ftp コマンドは、**AUTH TLS** と **PROT P** コマンドを使用して、**ftpd** デーモンとの通信を保護します。**AUTH TLS** と **PROT P** コマンドの両方が TLS プロトコルをサポートしていると、保護されたチャネルが確立されます。標準認証方式のみがサポートされています。

ftp コマンドの実行時に **-s** フラグが指定されていると、**ftp** コマンドはユーザーのホーム・ディレクトリー内でローカル **$HOME/.ftpcnf** ファイルを検索します。このファイルが見つかると、**ftp** コマンドは以下の構成パラメーターを使用して、サーバーとの TLS セッションをセットアップします。このファイルが見つからないか、構成パラメーターが欠落している場合、**ftp** コマンドは構成パラメーターを使用しないでサーバーへの接続を試行します。

CRL_PATH
CRL_PATH パラメーターは、証明書取り消しリスト・ファイルのパスを提供します。このファイルは Privacy Enhanced Mail (PEM) フォーマットでなければなりません。これを指定すると、サーバーが提供するデジタル証明書が証明書取り消しリストと照合されます。証明書が取り消されていた場合、TLS セッションは失敗します。このパラメーターが無指定の場合、デジタル証明書は証明書取り消しリストと照合されません。

CA_PATH
CA_PATH パラメーターは、認証局ファイルへのパスを提供します。このファイルは PEM フォーマットでなければなりません。これを指定すると、サーバー証明書が認証局と照合されます。サーバーが提供するデジタル証明書がセキュリティー機関によって署名されていなかった場合は、TLS セッションが失敗します。このパラメーターが無指定の場合、サーバーが提供するデジタル証明書は証明書取り消しリストと照合されません。

CIPHER_LIST
CIPHER_LIST パラメーターを指定すると、このリストが TLS セッション中に使用されます。これを指定しないと、デフォルトの暗号リストが使用されます。

DEPTH
CA_PATH 構成パラメーターを指定すると、デジタル証明書階層内の **ftpd** サーバーが提供する証明書を検証するために DEPTH 値が使用されます。これを指定しないと、デフォルト値の 9 が使用されます。

CERTIFICATE
CERTIFICATE パラメーターは、PEM フォーマットの有効なデジタル証明書チェーン・ファイルへのパスを提供します。このファイルは TLS セッションで使用されます。

CERTIFICATE_PRIVATE_KEY
CERTIFICATE_PRIVATE_KEY パラメーターには、証明書秘密鍵 (PEM 形式) へのパスが含まれます。これは TLS セッション中に使用されます。 TLS をサポートするには、AIX Web Download Pack Programs Web サイトから最新バージョンの OpenSSL ツールをインストールする必要があります。

Trusted AIX システムの場合

ユーザーにはデフォルト・ログイン機密ラベル (SL) と保全性ラベル (TL) が割り当てられ、これが正常なログイン後にユーザーのプロセスの有効な SL と TL になります。ユーザーがデフォルト・ログイン SL でログインしたくない場合は、ログイン時に **-e** オプションを使用して、別の SL を提供することを選択できます。ユーザーが提供する SL は、ユーザーの認可に従い、システム認定の範囲に含まれる必要があります。TL は、ユーザーがログイン時に指定できません。デフォルト・ログイン SL と TL は、各ユーザーごとのユーザー名および認可とともに、**/etc/security/user** ファイル内に定義されています。**-e** オプションを使用する場合は、サーバー側のカーネル・トラステッド・ネットワーク・ビットをオフにする必要があります。

注**:** ユーザー ID が 128 以下のユーザーは、リモートの Trusted AIX システムにログインできません。

フラグ

項目	説明
-C	**send_file** コマンドを使用して送信される発信ファイルをネットワーク・バッファー・キャッシュ (NBC) に入れなければならないことを指定します。このフラグは、**-q** フラグが指定されていなければ指定できません。このフラグは、ファイルがバイナリー・モードで保護されずに送信される場合にのみ適用されます。
-d	**ftp** コマンドの動作についてのデバッグ情報を **syslogd** デーモンに送ります。 **-d** フラグを指定した場合、**/etc/syslog.conf** ファイルを編集して、次のいずれか 1 つのエントリーを追加しなければなりません。 `user.info FileName` または `user.debug FileName` **注: syslogd** デーモンの debug レベルには、info レベルのメッセージが含まれます。 **/etc/syslog.conf** ファイルを編集しないと、メッセージはなにも生成されません。 **/etc/syslog.conf** ファイルを変更した後、 **refresh -s syslogd** または、 **kill-1***SyslogdPID* コマンドを実行して、 **syslogd** デーモンに、構成ファイルに加えた変更内容を通知してください。デバッグ・レベルの詳細については、**/etc/syslog.conf** ファイルを参照してください。また、**debug** サブコマンドも参照してください。
-D *DataConnTimeOut*	**ftp** コマンドがデータ接続を保持する最大秒数を指定します。デフォルト値は 300 秒です。値は 300 秒から 3600 秒の範囲で指定することができます。
-f	認証の転送を行います。このフラグは、Kerberos 5 が現行の認証メソッドでない場合は無視されます。
-g	ファイル名のメタキャラクターの拡張を使用不可にします。メタキャラクターの解釈は、ファイル名の拡張 (グロビングともいう) として参照されます。 **glob** サブコマンドを参照してください。
-H	`FILE_Unlink` イベントの監査ロギングをオンにします (このイベントがユーザーに対して有効になっている場合)。
-i	複数のファイルを転送する間、対話式のプロンプト表示をオフにします。複数のファイル転送時のプロンプトの説明については、 **prompt**、 **mget**、 **mput**、 **mdelete** サブコマンドを参照してください。
-K	制御とデータ接続の両方に関する **sys/socket.h** ファイルに定義された SO_KEEPALIVE オプションを使用不可にします。
-k *realm*	リモート端末のレルムがローカル・システムのレルムと異なる場合は、ユーザーがそれを指定できるようにします。このような目的では、レルムは DCE セルと同義です。このフラグは、 Kerberos 5 が現在の認証メソッドでない場合は無視されます。
-M	ローカル・ホストとリモート・ホスト間でファイルが転送された後に **ftp** コマンドがブロックされないようにします。
-n	最初の接続時の自動ログインを防止します。このフラグを指定しないと、**ftp** コマンドは、ログインおよびリモート・ホスト用の初期化処理を記述した **$HOME/.netrc** エントリーを探します。**user** サブコマンドを参照してください。
-q	ネットワーク上のファイルの送信に **send_file** サブルーチンを使用しなければならないことを指定します。このフラグは、ファイルがバイナリー・モードで保護されずに送信される場合にのみ適用されます。
-v	リモート・サーバーからの応答をすべて表示し、データ転送の統計を提供します。 **ftp** コマンドの出力がコンソールやディスプレイなどの端末に対して行われる場合、これがデフォルトの表示モードになります。 stdin が端末でない場合、**-v** フラグを指定して **ftp** コマンドを起動するか、または **verbose** サブコマンドを実行しない限り、**ftp** コマンドは詳細モードを使用不可にします。
-s	**AUTH TLS** コマンドと **PROT P** コマンドを **ftpd** デーモンに送信することで、サーバーとの TLS セッションを開始します。TLS セッションが確立され、ユーザーが標準認証方式を使用して認証されると、データとコマンドの転送は暗号化されます。

サブコマンド

以下の **ftp** サブコマンドは、`ftp>` プロンプトで入力することができます。ブランク文字を含むパラメーターを指定するには、二重引用符 (`" "`) を使用します。

項目	説明
![*Command* [*Parameters*]]	ローカル・ホスト上の対話型シェルを起動します。1 つ以上のオプション・パラメーターとオプション・コマンドをシェル・コマンドと一緒に指定することができます。
$*Macro* [*Parameters*]	**macdef** サブコマンドで定義済みの、指定されたマクロを実行します。パラメーターは拡張されません。
?[*Subcommand*]	サブコマンドの説明をするヘルプ・メッセージを表示します。 *Subcommand* パラメーターを指定しない場合は、**ftp** コマンドは既知のサブコマンドのリストを表示します。
account [*Password*]	リモート・ホストのリソースへのアクセスを認める前に、必要となる補足パスワードをリモート・ホストに送信します。コマンドと一緒にパスワードを指定しないと、パスワードを求めるプロンプトが表示されます。パスワードは画面には表示されません。
append *LocalFile* [*RemoteFile*]	リモート・ホスト上のファイルにローカル・ファイルを追加します。リモート・ファイル名を指定しないと、ローカル・ファイル名が使用され、 **ntrans** サブコマンド、または **nmap** サブコマンドで行われた設定で、変更されます。**append** サブコマンドは、ファイルの追加時に、**form**、 **mode**、 **struct**、 **type** サブコマンドの現在の値を使用します。
ascii	**type ascii** サブコマンドと同義。
bell	各ファイルの転送が終わるたびにベルを鳴らします。
binary	**type binary** サブコマンドと同義。
block	**mode block** サブコマンドと同義。
bye	ファイル転送セッションを終了し、**ftp** コマンドを終了します。 **quit** サブコマンドと同じように機能します。
carriage-control	**form carriage-control** サブコマンドと同義。
case	ファイル名の大文字と小文字の切り替えを設定します。**case** サブコマンドがオンになっていると、**ftp** コマンドは、すべて大文字で表示されているリモート・ファイル名を、ローカル・ディレクトリーに書き込むときに大文字から小文字に変更します。デフォルトはオフです (**ftp** コマンドにより大文字のリモート・ファイル名はローカル・ディレクトリーに大文字で書き込まれます)。
cd *RemoteDirectory*	リモート・ホスト上の作業ディレクトリーを、指定されたディレクトリーに変更します。
cdup	リモート・ホスト上の作業ディレクトリーを、現行ディレクトリーの親に変更します。
close	ファイル転送セッションを終了しますが、**ftp** コマンドは終了しません。定義済みのマクロは消去されます。 **disconnect** サブコマンドと同じように機能します。
copylocal	ローカル・コピーを切り替えます。**copylocal** のデフォルトはオフです。ftp はファイルをそのファイル自体に (同じホスト名、同じパス名などを) 転送してゼロ設定しなくてもよいようにします。 **copylocal** をオンにすると、この検査が省略されます。
cr	ASCII タイプのファイルの転送中、レコードの受信時に復帰/行送りシーケンスから復帰文字をストリップします。 (**ftp** コマンドは、ファイル転送時に、復帰と行送りを使って ASCII タイプの各レコードを終了させます。) 実行中のオペレーティング・システムとは別のオペレーティング・システムによる、リモート・ホスト上のレコードには、単一の行送りが組み込まれている場合があります。組み込まれた行送りをレコード区切り文字と区別するには、**cr** サブコマンドをオフに設定します。**cr** サブコマンドはオンとオフの間で切り替わります。

項目	説明
debug [0 | 1]	デバッグ・レコードの保存のオン/オフを切り替えます。 **debug** または **debug 1** を指定して、リモート・ホストに送られた各コマンドを出力し、再始動制御ファイルを保管します。デバッグ・レコードの保存を中止するには、**debug** を再度指定するか **debug 0** を指定します。また、Ctrl-C キー・シーケンスも再始動制御ファイルを保管します。 **debug** サブコマンドを指定すると、**ftp** コマンドの操作についてのデバッグ情報が **syslogd** デーモンに送られます。**debug** サブコマンドを指定した場合には、 **/etc/syslog.conf** ファイルを編集して次のいずれか 1 つのエントリーを追加しなければなりません。 `user.info FileName` または `user.debug FileName` **注: syslogd** デーモンの debug レベルには、info レベルのメッセージが含まれます。 **/etc/syslog.conf** ファイルを編集しないと、メッセージはなにも生成されません。**/etc/syslog.conf** ファイルを変更した後、 **refresh -s syslogd** または、 **kill-1***SyslogdPID* コマンドを実行して、 **syslogd** デーモンに、構成ファイルに加えた変更内容を通知してください。デバッグ・レベルの詳細については、**/etc/syslog.conf** ファイルを参照してください。 **ftp -d** フラグも参照してください。
delete *RemoteFile*	指定されたリモート・ファイルを削除します。
dir [*RemoteDirectory*][*LocalFile*]	指定されたリモート・ディレクトリー (*RemoteDirectory*) の内容のリストを指定されたローカル・ファイル (*LocalFile*) に書き込みます。 *RemoteDirectory* パラメーターの指定がない場合、**dir** サブコマンドは現行リモート・ディレクトリーの内容を出力します。 *LocalFile* パラメーターが指定されない場合または - (ハイフン) の場合には、**dir** サブコマンドはローカル端末装置上にリストを表示します。
disconnect	ファイル転送セッションを終了しますが、**ftp** コマンドは終了しません。定義済みのマクロは消去されます。 **close** サブコマンドと同じように機能します。
ebcdic	**type ebcdic** サブコマンドと同義。
exp_cmd	標準的プロトコル・コマンドと実験的プロトコル・コマンドを切り替えます。デフォルトはオフです。
file	**struct file** サブコマンドと同義。
form [carriage-control | non-print | telnet]	ファイル転送のフォーマットを指定します。**form** サブコマンドは、指定したフォーマットでファイル転送を行うように **type** サブコマンドを変更します。有効な引数は、**carriage-control**、**non-print**、および **telnet** です。 **carriage-control** ファイル転送のフォーマットを紙送り制御に設定します。 **non-print** ファイル転送のフォーマットを非印刷に設定します。 **telnet** ファイル転送のフォーマットを Telnet に設定します。Telnet は、システムへの接続をオープンする伝送制御プロトコル/インターネット・プロトコル (TCP/IP) です。
get *RemoteFile* [*LocalFile*]	リモート・ファイルをローカル・ホストにコピーします。 *LocalFile* パラメーターの指定がない場合は、リモート・ファイル名をローカルに使用し、**case**、 **ntrans**、 **nmap** サブコマンドによる設定によって、変更します。 **ftp** コマンドは、ファイル転送時には、 **type**、 **form**、 **mode**、 **struct** サブコマンドの、現在の設定を使用します。

項目	説明
glob	**mdelete**、 **mget**、 **mput** サブコマンドの、ファイル名拡張 (グロビング) を切り替えます。グロビングが使用不可の場合は、上記のサブコマンドのファイル名パラメーターは拡張されません。グロビング機能が使用可能であり、単一のファイル名を期待するサブコマンドにパターン・マッチング文字が使用されていると、予期しない結果が起こることがあります。 例えば、**append** サブコマンドと、 **put** サブコマンドは、ファイル名の拡張を行った後、最初に生成されたファイル名だけを使用します。 **cd**、 **delete**、 **get**、 **mkdir**、 **rename**、 **rmdir** など、その他の **ftp** サブコマンドは、ファイル名の拡張を行わず、パターン・マッチング文字をそのまま採用します。 **mput** サブコマンドのグロビングは、 **csh** コマンドの場合と同じ方法で、ローカルに行われます。**mdelete** サブコマンドと、 **mget** サブコマンドの場合には、各ファイル名はリモート・コンピューターで別々に拡張され、リストはマージされません。ディレクトリー名の拡張はリモート・ホストと **ftp** サーバーに依存しており、ファイル名の拡張とは異なる場合があります。 ディレクトリー名の拡張をプレビューするには、 **mls** サブコマンドを使用してください。 `mls RemoteFile` ファイルのディレクトリー・サブツリー全体を転送するには、 **mget** サブコマンドや、 **mput** サブコマンドを使わずに、サブツリーの **tar** アーカイブを、バイナリー・フォーマットで転送します。
hash	ハッシュ記号 (#) の表示を切り替えます。 **hash** サブコマンドがオンの場合、**ftp** コマンドは転送されるデータ・ブロック (1024 バイト) ごとにハッシュ記号を 1 個表示します。
help [*Subcommand*]	ヘルプ情報を表示します。 **?** サブコマンドを参照してください。
image	**type image** サブコマンドと同義。
lcd [*Directory*]	ローカル・ホスト上の作業ディレクトリーを変更します。ディレクトリーを指定しない場合は、**ftp** コマンドはユーザーのホーム・ディレクトリーを使用します。
local *M*	**type local** *M* サブコマンドと同義。
ls [*RemoteDirectory*] [*LocalFile*]	リモート・ディレクトリーの簡略ファイル・リストをローカル・ファイルに書き込みます。 *RemoteDirectory* パラメーターの指定がないと、**ftp** コマンドは現行リモート・ディレクトリーのリストを表示します。 *LocalFile* パラメーターが指定されていないか - (ハイフン) である場合、**ftp** コマンドはローカル端末にリストを表示します。
macdef *Macro*	サブコマンドのマクロを定義します。後に続く null 行までの行 (2 つ連続の行送り) はマクロのテキストとして保管されます。最高 4096 文字を含む 16 個までのマクロを定義できます。マクロは、再定義するか、 **close** サブコマンドを実行するまで、定義されたままです。 $ (ドル記号) と ¥ (円記号) は **ftp** マクロ内の特殊文字です。 $ 記号の後に 1 つ以上の数値を付けると、呼び出し行の対応するマクロ・パラメーターで置き換えられます (**$** サブコマンドを参照)。$ 記号の後に i という文字がくると、マクロがループすることを示し、$i 文字の組み合わせは各パス上の連続パラメーターに置き換えられます。 最初のマクロ・パラメーターは最初のパスで使用され、2 番目のマクロ・パラメーターは 2 番目のパスで使用されます (以下同様)。 ¥ 記号は次にくる文字が特殊な扱いを受けることを防止します。$ および ¥. (円記号、ピリオド) 記号の特殊な意味を無効にするには、¥ 記号を使用します。
mdelete *RemoteFiles*	リモート・ホストで *RemoteFiles* パラメーターによって指定されたファイルを拡張し、リモート・ファイルを削除します。

項目	説明
mdir [*RemoteDirectories LocalFile*]	リモート・ホストで *RemoteDirectories* パラメーターによって指定されたディレクトリーを展開し、拡張したディレクトリーの内容のリストを *LocalFile* パラメーター内で指定されたファイルに書き込みます。 *RemoteDirectories* パラメーターがパターン・マッチング文字を含んでいる場合、何も指定がなければ、**mdir** サブコマンドはローカル・ファイルを求めるプロンプトを表示します。 *RemoteDirectories* パラメーターがブランクで区切られたリモート・ディレクトリーのリストである場合、そのリストの最後の引数はローカル・ファイル名または - (ハイフン) のどちらかでなければなりません。 *LocalFile* パラメーターが - (ハイフン) の場合、**mdir** サブコマンドはローカル端末装置上でリストを表示します。対話式のプロンプトがオンになっていると (**prompt** サブコマンドを参照)、**ftp** コマンドはユーザーに対して、最後のパラメーターがローカル・ファイルであって、リモート・ディレクトリーではないことを確認するよう指示します。
mget *RemoteFiles*	リモート・ホスト上で *RemoteFiles* パラメーターを拡張し、指示されたリモート・ファイルをローカル・ホスト上の現行ディレクトリーにコピーします。ファイル名の詳細については、 **glob** サブコマンドを参照してください。リモート・ファイル名はローカルで使用され、 **case**、 **ntrans**、 **nmap** サブコマンドによる設定で変更されます。 **ftp** コマンドは、ファイルの転送時に、 **form**、 **mode**、 **struct**、 **type** サブコマンドの、現在の設定を使用します。
mkdir [*RemoteDirectory*]	*RemoteDirectory* パラメーターで指定されたディレクトリーをリモート・ホスト上で作成します。
mls [*RemoteDirectories LocalFile*]	リモート・ホストで *RemoteDirectories* パラメーターで指定されたディレクトリーを拡張し、指示されたリモート・ディレクトリーの簡略ファイル・リストをローカル・ファイルに書き込みます。 *RemoteDirectories* パラメーターがパターン・マッチング文字を含んでいる場合、何も指定がなければ、**mls** サブコマンドはローカル・ファイルを求めるプロンプトを表示します。 *RemoteDirectories* パラメーターがブランクで区切られたリモート・ディレクトリーのリストである場合、そのリストの最後の引数はローカル・ファイル名または - (ハイフン) のどちらかでなければなりません。 *LocalFile* パラメーターが - (ハイフン) の場合、**mls** サブコマンドはローカル端末装置上でリストを表示します。対話式のプロンプトがオンになっていると (**prompt** サブコマンドを参照)、**ftp** コマンドはユーザーに対して、最後のパラメーターがローカル・ファイルであって、リモート・ディレクトリーではないことを確認するよう指示します。
mode [**stream** | **block**]	ファイル転送モードを設定します。引数を指定しないと、デフォルトは **stream** です。 **block** ファイル転送モードをブロックに設定します。 **stream** ファイル転送モードをストリームに設定します。

項目	説明
modtime	指定したファイルの、最後に変更した時刻をリモート・マシン上に表示します。 **ftp** コマンドを実行する前にホストに接続していないと、**modtime** サブコマンドは、エラー・メッセージを出して終了します。 **ftp** コマンドは、最初のパラメーター以外のパラメーターを無視します。 *FileName* パラメーターを指定していない場合には、**ftp** コマンドはファイル名を要求します。ファイル名を指定しないと、**ftp** コマンドは標準出力に使用方法メッセージを送り、サブコマンドを終了します。 *FileName* パラメーターで指定した名前がリモート・ホスト上に存在し、その名前が 1 つのファイルを示す場合、**ftp** コマンドはそのファイルの最後の変更時刻を示すメッセージを標準出力に送信し、サブコマンドを終了します。 *FileName* にディレクトリーを指定した場合、**ftp** コマンドはエラー・メッセージを標準出力に送り、サブコマンドを終了します。 **注: modtime**modtime サブコマンドは、メタキャラクターを解釈します (許可されている場合)。
mput [*LocalFiles*]	*LocalFiles* パラメーターで指定されたファイルをローカル・ホスト上で拡張し、指示されたローカル・ファイルをリモート・ホストにコピーします。ファイル名の詳細については、 **glob** サブコマンドを参照してください。ローカル・ファイル名はリモート・ホストで使用され、 **ntrans**、 **nmap** サブコマンドによる設定で変更されます。 **ftp** コマンドは、ファイル転送時には、 **type**、 **form**、 **mode**、 **struct** サブコマンドの現在の設定を使用します。

項目	説明
nlist [*RemoteDirectory*][*LocalFile*]	指定されたリモート・ディレクトリー (*RemoteDirectory*) の内容のリストを指定されたローカル・ファイル (*LocalFile*) に書き込みます。 *RemoteDirectory* パラメーターが指定されていない場合は、**nlist** サブコマンドは現行のリモート・ディレクトリーの内容を表示します。 *LocalFile* パラメーターが指定されていないか、または - (ハイフン) である場合、**nlist** サブコマンドはローカル端末装置上にリストを表示します。
nmap [*InPattern OutPattern*]	ファイル名マッピング機能を設定、または解除します。パラメーターの指定がないと、ファイル名のマッピングはオフにされます。パラメーターを指定すると、**mget** サブコマンドと、 **mput** サブコマンド、また、宛先ファイルを指定しない場合は、 **get** サブコマンドと、 **put** サブコマンドで、ソース・ファイル名がマップされます。このサブコマンドは、ローカル・ホストとリモート・ホストで使用されるファイル命名の規則や慣行が異なる場合に便利です。マッピングは、*InPattern* パラメーターと *OutPattern* パラメーターによって設定されたパターンどおりに行われます。 *InPattern* パラメーターは、入力ファイル名のテンプレートを示します。このテンプレートは、**case**、および **ntrans** の設定値に従って、既に処理されている場合があります。テンプレート変数の `$1` から `$9` までを *InPattern* パラメーターに入れることができます。 *InPattern* パラメーター内の $ (ドル記号) と ¥$ (円記号、ドル記号) 以外のすべての文字は、文字どおりに処理され、*InPattern* 変数間の区切り文字として使用されます。例えば、*InPattern* パラメーターが `$1.$2` で、リモート・ファイル名が `mydata.dat` の場合、`$1` の値は `mydata` に、`$2` の値は `dat` になります。 *OutPattern* パラメーターは結果として生じるファイル名を決定します。 `$1` から `$9` までの変数は *InPattern* パラメーターから派生した値に置き換えられ、変数 `$0` は元のファイル名に置き換えられます。さらに、シーケンス [*Sequence1,Sequence2*] は、*Sequence1* の値が null でなければ *Sequence1* に置き換えられます。Sequence1 の値が null の場合は *Sequence2* に置き換えられます。例えば、サブコマンド `nmap $1.$2.$3 [$1,$2].[$2,file]` は、`myfile.data` または `myfile.data.old` から `myfile.data` を、 `myfile` から `myfile.file` を、そして `.myfile` から `myfile.myfile` を生成します。*OutPattern* パラメーター内で、$ (ドル記号)、[(左大括弧)、] (右大括弧)、および , (コンマ) に特別な意味を持たせないようにするには、¥ (円記号) を使用します。
non-print	**form non-print** サブコマンドと同義。
ntrans [*InCharacters* [*OutCharacters*]]	ファイル名文字変換メカニズムをオンまたはオフにします。パラメーターを指定しないと、文字変換はオフになります。パラメーターを指定した場合、 **mget** サブコマンド、および **mput** サブコマンドでは、ソース・ファイル名の文字が変換され、 **get** サブコマンド、および **put** サブコマンドでは、宛先ファイル名を指定しない場合にソース・ファイル名の文字が変換されます。 このサブコマンドは、ローカル・ホストとリモート・ホストで使用されるファイル命名の規則や慣行が異なる場合に便利です。文字変換は、*InCharacters* および *OutCharacters* パラメーターによって設定されたパターン通りに行われます。 *InCharacters* パラメーター内の文字に一致するソース・ファイル名の文字は、*OutCharacters* パラメーター内の対応する文字に置き換えられます。 *InCharacters* パラメーターで指定した文字列の長さが *OutCharacters* パラメーターで指定した文字列より長い場合、 *InCharacters* パラメーターの文字列の中で *OutCharacters* パラメーターの中の文字と対応しないものは削除されます。
open *HostName* [*Port*]	*HostName* パラメーターで指定されたホストで、FTP サーバーへの接続を確立します。オプションのポート番号が指定されると、**ftp** コマンドはそのポートでサーバーへの接続を試みます。自動ログイン機能を設定すると (つまり、コマンド・ラインで **-n** フラグを指定しない場合)、 **ftp** コマンドはユーザーを FTP サーバーにログインさせようとします。 また、正確な情報が与えられ、適正な許可が設定された **$HOME/.netrc** ファイルが必要です。**.netrc** ファイルはホーム・ディレクトリーに入っていなければなりません。
passive	ファイル転送用のパッシブ・モードに切り替えます。ファイル転送コマンド (**get**、**mget**、**put**、**mput** など) を、パッシブ・モードをオフにして起動すると、**ftp** サーバーは、クライアントに戻るデータ接続をオープンします。パッシブ・モードでは、クライアントは、データの送受信時にサーバーへのデータ接続をオープンします。

項目	説明
private	保護レベルをプライベートに設定します (認証方式が設定されている場合のみ)。このレベルでは、データの保全性と機密性が保護されます。
prompt	対話式プロンプトを切り替えます。対話式のプロンプト表示機能をオン (デフォルト) にしている場合、 **ftp** コマンドは、**mget**、 **mput**、 **mdelete** サブコマンドの実行時に、複数のファイルの検索、送信、削除を行う前に、確認のプロンプトを表示します。そうでない場合、**ftp** コマンドは、指定されたすべてのファイル上で状況に応じた操作を行います。
protect	このコマンドは、現行レベルの保護を戻します。
proxy [*Subcommand*]	**ftp** コマンドを 2 次制御接続上で実行します。このサブコマンドを使うと、**ftp** コマンドは、2 個のサーバー間でファイルを転送するために、2 個のリモート FTP サーバーに同時に接続できます。最初の **proxy** サブコマンドは、 2 次制御接続を設定する **open** サブコマンドでなければなりません。 2 次接続で実行可能なその他の **ftp** サブコマンドを参照するには、 **proxy ?** サブコマンドを入力します。 次のサブコマンドは、**proxy** サブコマンドを前に付けると、別の機能を持ちます。 • **open** サブコマンドは、自動ログイン・プロセス中に新しいマクロを定義しません。 • **close** サブコマンドは、既存のマクロ定義を消去しません。 • **get**、および **mget** サブコマンドは、 1 次接続のホストから 2 次接続のホストへのファイル転送を行います。 • **put**、 **mput**、 **append** サブコマンドは、 2 次接続のホストから 1 次接続のホストへの転送を行います。 • **restart** サブコマンドは、 **proxy** コマンドで取り扱うことができます。 • **status** サブコマンドは、正確な情報を表示します。 ファイル転送では、2 次接続上の FTP サーバーが PASV (パッシブ) 命令をサポートする必要があります。
put *LocalFile* [*RemoteFile*]	ローカル・ファイルをリモート・ホストに格納します。 *RemoteFile* パラメーターを指定しないと、**ftp** コマンドは、ローカル・ファイルの名前を使ってリモート・ファイルに名前を付けます。リモート・ファイル名は、 **ntrans**、および **nmap** サブコマンドでの設定によって、変更されます。 **ftp** コマンドは、ファイル転送時には、 **type**、 **form**、 **mode**、 **struct** サブコマンドの現在の設定を使用します。
pwd	リモート・ホスト上の現行ディレクトリーの名前を表示します。
quit	接続をクローズして **ftp** コマンドを終了します。 **bye** サブコマンドと同じように機能します。
quote *String*	*String* パラメーターで指定した文字列と同じ文字列をリモート・ホストに送信します。 *String* パラメーターに有効な値のリストを表示するには、 **remotehelp** または **quote help** サブコマンドを実行します。 **注:** データ転送を含む「Quoting (引用符で囲む)」コマンドを使用すると、予期しない結果になることがあります。
record	**struct record** サブコマンドと同義。
recv *RemoteFile* [*LocalFile*]	リモート・ファイルをローカル・ホストにコピーします。 **get** サブコマンドと同じように機能します。
reinitialize	すべての入出力をフラッシュして転送を完了させ、FTP セッションを再初期化します。リモート・ホストにログインせずに FTP セッションを開始した直後と同様に、すべてのデフォルト値をリセットします。
remotehelp [*Subcommand*]	リモート FTP サーバーからのヘルプを要求します。
rename *FromName ToName*	リモート・ホスト上のファイルの名前を変更します。
reset	応答キューを消去します。このサブコマンドはコマンドの構文解析を再同期します。
restart get | **put** | **append**	最後のチェックポイントが作成された位置からファイル転送を再開します。正常に実行するには、サブコマンドの構造、タイプ、フォーマットが、打ち切られたサブコマンドのものと同じでなければなりません。有効な引数は、**get**、**put**、**append** です。
rmdir *RemoteDirectory*	*RemoteDirectory* パラメーターで指定されたリモート・ディレクトリーを、リモート・ホストから除去します。

項目	説明
runique	(ReceiveUnique) **get**、および **mget** サブコマンドの実行時に、ローカル宛先ファイルの固有ファイル名を作成する機能を切り替えます。この機能がオフ (デフォルト) になっていると、**ftp** コマンドはローカル・ファイルを上書きします。この機能がオンの場合は、ローカル宛先ファイルに指定された名前と同じ名前を持つローカル・ファイルがあると、**ftp** コマンドはローカル宛先ファイルに指定された名前に **.1** を付けて変更します。その名前を使用しているローカル・ファイルが既に存在する場合、**ftp** コマンドは指定された名前に .2 という接尾語を付けます。この名前も既にローカル・ファイルで使用されている場合、**ftp** コマンドは、固有ファイル名が見つかるまで接尾部の増分を続けます。固有ファイル名が見つからないまま .99 に達すると、接尾部の増分をやめます。固有ファイル名が見つからなかった場合、**ftp** コマンドはエラーを報告し、転送は行われません。**runique** サブコマンドはシェル・コマンドから生成されたローカル・ファイル名には影響を与えません。
safe	保護レベルを「安全」に設定します。このレベルでは、データは保全性が保護されます。
send *LocalFile* [*RemoteFile*]	ローカル・ファイルをリモート・ホストに格納します。 **put** サブコマンドと同じように機能します。
sendport	FTP PORT 命令の使用を切り替えます。デフォルトでは、**ftp** コマンドは個々のデータ転送の接続を設定するときに、PORT 命令を使用します。PORT 命令が使用不可のときは、**ftp** コマンドはデータ転送に PORT 命令を使用しません。PORT 命令は、PORT 命令を受信したことを正しく表示せず、PORT 命令の実行を無視する FTP サーバーを扱う場合に便利です。
site *Args*	**chmod** コマンドを使って、アイドル・タイムアウト期間の表示または設定、ファイル作成 umask の表示または設定、あるいはファイルの許可の変更を行います。*Args* パラメーターに使用できる値は、**umask** と **chmod** です。
size *RemoteFile*	*RemoteFile* パラメーターで指定したリモート・ファイルのサイズをバイト数で表示します。
status	**ftp** コマンドの現行状況と、サブコマンドの状況を表示します。
stream	**mode stream** サブコマンドと同義。
struct [file | record]	データ転送構造タイプを設定します。有効な引数は **file** と **record** です。 **file** データ転送構造タイプをファイルに設定します。 **record** データ転送構造タイプをレコードに設定します。
sunique	(Send/Store Unique) **put**、および **mput** サブコマンドの実行時に、リモート宛先ファイルの固有ファイル名を作成する機能を切り替えます。この機能がオフ (デフォルト) であれば、**ftp** コマンドはリモート・ファイルを上書きします。それ以外の場合、リモート宛先ファイル用に指定したのと同じ名前がリモート・ファイルに付いていれば、リモート FTP サーバーがリモート宛先ファイル名を変更します。リモート・サーバーは必ず STOU 命令をサポートしていなければならないことに注意してください。
system	リモート・マシンで実行されているオペレーティング・システムのタイプを表示します。
telnet	**form telnet** サブコマンドと同義。
tenex	**type tenex** サブコマンドと同義。
trace	パケットのトレースを切り替えます。
type [ascii | binary | ebcdic | image | local *M* **| tenex]**	ファイル転送タイプを設定します。有効な引数は **ascii**、**binary**、**ebcdic**、**image**、**local** *M*、**tenex** です。引数が指定されない場合は、現行タイプを表示します。デフォルト・タイプは **ascii** です。**binary** タイプは **ascii** より効率的な場合があります。 **ascii** ファイル転送タイプをネットワーク ASCII に設定します。これがデフォルトとなります。バイナリー・イメージ転送によるファイル転送のほうが効率的な場合があります。詳細については **binary** 引数を参照してください。 **binary** ファイル転送タイプをバイナリー・イメージに設定します。このタイプのほうが ASCII 転送より効率的な場合があります。 **ebcdic** ファイル転送タイプを EBCDIC に設定します。 **image** ファイル転送タイプをバイナリー・イメージに設定します。このタイプのほうが ASCII 転送より効率的な場合があります。 **local** *M* ファイル転送タイプをローカルに設定します。*M* パラメーターは、機械語 1 語あたりのビット数を 10 進数で定義します。このパラメーターにはデフォルト値はありません。 **tenex** ファイル転送タイプを、TENEX マシンに必要なタイプに設定します。

項目	説明
user *User* [*Password*] [*Account*]	ローカル・ユーザー (*User*) をリモート FTP サーバーに知らせます。リモート・サーバーが *Password* や *Account* パラメーターを要求している場合にそれらが指定されないと、**ftp** コマンドはパスワードやアカウントの入力を求めるプロンプトをローカルに表示します。 *Account* パラメーターが必要な場合、**ftp** コマンドはリモート・ログイン処理が完了した後で Account パラメーターをリモート・サーバーに送信します。 **注:** コマンド・ラインに、 **-n** フラグを指定して、自動ログインを使用不可にしない限り、**ftp** コマンドは、リモート・サーバーへの初期接続時に自動的に、*User*、*Password*、*Account* パラメーターを送信します。また、自動ログインを発行するには、ホーム・ディレクトリー内に **.netrc** ファイルが必要です。
verbose	詳細モードを切り替えます。詳細モードがオン (デフォルト) の場合、**ftp** コマンドはリモート FTP サーバーからの応答をすべて表示します。また、**ftp** コマンドは、転送の完了時にすべてのファイル転送についての統計を表示します。

例

1. **ftp** コマンドを起動し、システム canopus にログインし、ローカル・ヘルプ情報、リモート・ヘルプ情報、および状況を表示し、**bell**、**prompt**、**runique**、**trace**、および **verbose** サブコマンドを切り替えてから、終了するには、次のように入力します。

```
$ ftp canopus
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
Name (canopus:eric): dee
331 Password required for dee.
Password:
230 User dee logged in.
ftp> help
Commands may be abbreviated. Commands are:
!          delete          mdelete          proxy          runique
$          debug           mdir             sendport       send
account    dir             mget             put            size
append     disconnect      mkdir            pwd            status
ascii      form            mls              quit           struct
bell       get             mode             quote          sunique
binary     glob            modtime          recv           system
bye        hash            mput             remotehelp     tenex
case       help            nmap             rstatus        trace
cd         image           nlist            rhelp          type
cdup       lcd             ntrans           rename         user
close      ls              open             reset          verbose
cr         macdef          prompt           rmdir          ?
clear      private         protect          safe
ftp> remotehelp
214-The following commands are recognized(* =>'s unimplemented).
 USER   PORT   RETR   MSND*  ALLO   DELE   SITE*  XMKD   CDUP
 PASS   PASV   STOR   MSOM*  REST*  CWD    STAT*  RMD    XCUP
 ACCT*  TYPE   APPE   MSAM*  RNFR   XCWD   HELP   XRMD   STOU
 REIN*  STRU   MLFL*  MRSQ*  RNTO   LIST   NOOP   PWD
 QUIT   MODE   MAIL*  MRCP*  ABOR   NLST   MKD    XPWD
 AUTH   ADAT   PROT   PBSZ   MIC    ENC    CCC
214 Direct comments to ftp-bugs@canopus.austin.century.com.
ftp> status
Connected to canopus.austin.century.com.
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
ftp> bell
Bell mode on.
```

```
ftp> prompt
Interactive mode off.
ftp> runique
Receive unique on.
ftp> trace
Packet tracing on.
ftp> verbose
Verbose mode off.
ftp> quit
$
```

2. **ftp** コマンドを起動して、システム canopus にログインし、作業ディレクトリーを表示、変更し、ファイル転送タイプを ASCII に設定し、ローカル・ファイルをリモート・ホストに送信し、作業ディレクトリーを親ディレクトリーに変更して終了するには、次のように入力します。

```
$ ftp canopus
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
Name (canopus:eric): dee
331 Password required for dee.
Password:
230 User dee logged in.
ftp> pwd
257 "/home/dee" is current directory.
ftp> cd desktop
250 CWD command successful.
ftp> type ascii
200 Type set to A.
ftp> send typescript
200 PORT command successful.
150 Opening data connection for typescript (128.114.4.99,1412).
226 Transfer complete.
ftp> cdup
250 CWD command successful.
ftp> bye
221 Goodbye.
$
```

3. (**.netrc** ファイルを使って) 自動ログインで、**ftp** コマンドを起動し、システム canopus とのセッションをオープンしてログインし、作業ディレクトリーを親ディレクトリーに変更して、作業ディレクトリーを表示し、現行ディレクトリーの内容をリストし、ファイルを削除して、現行ディレクトリーの内容のリストをローカル・ファイルに書き込み、セッションをクローズして終了するには、次のように入力します。

```
$ ftp canopus
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
331 Password required for dee.
230 User dee logged in.
ftp> cdup
250 CWD command successful.
ftp> pwd
257 "/home" is current directory.
ftp> dir
200 PORT command successful.
150 Opening data connection for /usr/bin/ls (128.114.4.99,1407)
(0 bytes).
total 104
drwxr-xr-x   2 system        32 Feb 23 17:55 bin
Drwxr-xr-x  26 rios        4000 May 30 17:18 bin1
drwxr-xr-x   2 system        32 Feb 23 17:55 books
drwxrwxrwx  18 rios        1152 Jun  5 13:41 dee
-r--r--r--   1 system      9452 May 17 12:21 filesystems
drwxr-xr-x   2 system        32 Feb 23 17:55 jim
drwxr-xr-x   5 system        80 Feb 23 17:55 krs
drwxrwxrwx   2 rios       16432 Feb 23 17:36 lost+found
```

```
-rwxr-xr-x   1 rios        3651 May 24 16:45 oldmail
drwxr-xr-x   2 system       256 Feb 23 17:55 pubserv
drwxrwxrwx   2 system       144 Feb 23 17:55 rein989
drwxr-xr-x   2 system       112 Feb 23 17:55 reinstall
226 Transfer complete.
ftp> delete oldmail
250 DELE command successful.
ftp> mdir /home/dee/bin binlist
output to local-file: binlist? y
200 PORT command successful.
150 Opening data connection for /usr/bin/ls (128.114.4.99,1408) (0 bytes).
226 Transfer complete.
ftp> close
221 Goodbye.
ftp> quit
$
```

ファイル

項目	説明
/usr/samples/tcpip/netrc	サンプルの **.netrc** ファイルが入っています。
/etc/syslog.conf	**syslogd** デーモンの構成情報が入っています。

関連資料:

『ftpd デーモン』

関連情報:

stty コマンド

telnet コマンド

tftp コマンド

syslogd コマンド

Fftp および rcp コマンドを使用するファイル転送

---

ftpd デーモン

目的

インターネット FTP プロトコルにサーバー機能を持たせます。

構文

注**: ftpd** デーモンは、通常、 **inetd** デーモンによって始動されます。また、SRC コマンドを使用してコマンド・ラインから制御することもできます。

/usr/sbin/ftpd [**-d**] [**-D** *DataConnTimeOut*] **[-e][** **-f**] [**-ff**] [**-k**] [**-l**] [**-U**] [**-t** *TimeOut*] [**-T** *MaxTimeOut*] [**-s**] [**-u** *OctalVal*] **[-q** **[-C]]** **[-c]** **[-H]**

説明

/usr/sbin/ftpd デーモンは、 DARPA インターネット・ファイル転送プロトコル (FTP) サーバー・プロセスです。 **ftpd** デーモンは、伝送制御プロトコル (TCP) を使用して、**/etc/services** ファイル内の、**ftp** コマンド・サービス仕様で指定されたポート上で listen します。

ftpd デーモンの変更は、System Management Interface Tool (SMIT) またはシステム・リソース・コントローラー (SRC) を使用して、**/etc/inetd.conf** ファイルまたは **/etc/services** ファイルを編集することによ

って行うことができます。 ftpd をコマンド・ラインに入力することはお勧めしません。**/etc/inetd.conf** ファイル内でコメントが解除されていれば **ftpd** デーモンはデフォルトで始動します。

inetd デーモンは、**/etc/inetd.conf** ファイルと **/etc/services** ファイルから情報を取り出します。

/etc/inetd.conf または **/etc/services** ファイルを変更する場合は、**refresh -s inetd** または **kill -1***InetdPID* コマンドを実行して、**inetd** デーモンにその構成ファイルへの変更を通知します。

ftpd デーモンは、ファイル名を **csh** コマンドの規則に従って拡張します。このコマンドを使用すると、* (アスタリスク)、?(疑問符)、[ ] (左大括弧と右大括弧)、{ } (左中括弧と右中括弧)、~ (ティルド) のようなメタキャラクターを使用できます。

ftpaccess.ctl File

/etc/ftpaccess.ctl ファイルでは、**allow:**、 **deny:**、**readonly:**、**writeonly:**、 **readwrite:**、**useronly:**、**grouponly:**、**herald:** および/または **motd:** で始まる行が検索されます。その他の行は無視されます。ファイルが無い場合には、すべてのホストに対して FTP アクセスが許可されます。 **allow:** および **deny:** 行は、ホストのアクセスを制限します。**readonly:**、**writeonly:** および **readwrite:** 行は、 FTP の読み取り (get) および書き込み (put) を制限します。**useronly:** および **grouponly:** 行は、無名ユーザーを定義します。 **herald:** および **motd:** 行は、ログイン前およびログイン後の複数行メッセージに関する指定を行います。

/etc/ftpaccess.ctl 内のすべての行の構文は、次のフォームです。

```
keyword: value, value, ...
```

キーワードごとに 1 つ以上の値を指定することができます。同じキーワードのものを複数行指定できます。**/etc/ftpaccess.ctl** の行は 1024 文字までに制限されています。1024 を超える文字は無視されます。

allow: および **deny:** 行の構文は、次のようになります。

```
allow: host, host, ...
deny: host, host, ...
```

allow: 行が指定されている場合、すべての **allow:** 行にリストされているホストのみが FTP アクセスを許可されます。他のすべてのホストは FTP アクセスを拒否されます。 **allow:** 行が無い場合には、**deny:** 行に指定されているホストを除くすべてのホストが FTP アクセスを許可されます。ホストは、ホスト名または IP アドレスのいずれかで指定します。

readonly:、**writeonly**:、**readwrite**: 行の構文は、次のようになります。

```
readonly: dirname, dirname, ...
writeonly: dirname, dirname, ...
readwrite: dirname, dirname, ...
```

readonly: 行は読み取り専用ディレクトリーをリストし、**writeonly:** 行は書き込み専用ディレクトリーをリストします。書き込み専用ディレクトリーでは読み取りアクセスは拒否され、読み取り専用ディレクトリーでは書き込みアクセスは拒否されます。 **readwrite:** 行が指定されている場合を除いて、他のすべてのディレクトリーはアクセスを認可されます。 **readwrite:** 行が指定されていると、**readwrite:** 行および/または **readonly:** 行にリストされているディレクトリーのみが読み取りのアクセスを認可されます。同様に、**readwrite:** 行および/または **writeonly:** 行にリストされているディレクトリーのみが書き込みのアクセスを認可されます。また、これらの行に「ALL」あるいは「NONE」という値を指定できます。

useronly:、**puseronly:**、**grouponly:**、および **pgrouponly:** 行の構文は、次のようになります。

```
useronly: username, username, ...
puseronly: username, username, ...
grouponly: groupname, groupname, ...
pgrouponly: groupname, groupname, ...
```

username は **/etc/passwd** にあり、groupname は **/etc/group** にあります。 **useronly:** および **puseronly:** 行は、無名ユーザーを定義します。**grouponly:** および **pgrouponly:** 行は、無名ユーザーのグループを定義します。これらの無名ユーザーは、ftp 操作が各自のホーム・ディレクトリーに限定されているという点でユーザー「anonymous」に似ています。**useronly:** および **grouponly:** 行は、パスワードで保護されない無名ユーザーを定義します。これはユーザー「anonymous」に似ています。**puseronly:** および **pgrouponly:** 行は、パスワードで保護される無名ユーザーを定義します。

注**: puseronly:** および **pgrouponly:** のユーザーの場合は、パスワードを作成し、ログインを使用不可にする必要があります。

herald: および **motd:** 行の構文は、次のようになります。

```
herald: path
motd: on|off
```

パスは、ログイン前に表示される複数行の herald が入っているファイルの絶対パス名です。 **motd:** 行の値が 'on' である場合は、**$HOME/motd** ファイルに、ログイン後に表示される複数行のメッセージが入っています。ユーザーが定義済み無名ユーザーである場合は、**/etc/motd** ファイルに、ログイン後に表示される複数行のメッセージが入っています。 (**/etc/motd** は、無名ユーザーの chroot が実行されるホーム・ディレクトリーです。) **motd:** 行のデフォルトは off です。

標準オペレーティング・システムの認証メソッドが現行認証メソッドである場合

ftpd デーモンは、クライアント・プロセスを認証してからでないと、クライアント・プロセス用にファイルを転送できません。 **ftpd** デーモンは次の規則に従ってクライアント・プロセスを認証します。

- ユーザーは、パスワード・データベース **/etc/security/passwd** に、パスワードを入れておく必要があります。(ユーザーのパスワードが null でない場合、クライアント・プロセスでそのパスワードを指定しなければなりません。)
- ユーザー名は、 **/etc/ftpusers** ファイル名に存在するものは、無効です。
- ユーザーの login shell は、**/etc/security/login.cfg** ファイルのシェル属性に存在するものでなければなりません。
- ユーザー名が anonymous、ftp または **/etc/ftpaccess.ctl** ファイルに定義されている無名ユーザーである場合は、無名の FTP アカウントをパスワード・ファイルに定義する必要があります。この場合、クライアント・プロセスはどのパスワードでもログインできます。規則では、パスワードにはクライアント・ホストの名前を使います。 **ftpd** デーモンは特別措置を講じてクライアント・プロセスが無名のアカウントへアクセスするのを制限します。

Kerberos 5 が現行認証メソッドである場合

以下の条件がすべて満たされた場合にのみ、**ftpd** デーモンはアクセスを認めます。

- ftp クライアントのローカル・ユーザーに、現行 DCE 認証がある。
- ローカルおよびリモートの両システムが、 **AUTH** コマンドをサポートしている。
- リモート・システムが DCE 認証を、リモート・アカウントにアクセスするのに十分なものとして受け入れる。詳細については、**kvalid_user** 機能のセクションを参照してください。

トランスポート層セキュリティーのサポート

ftpd デーモンは、RFC 4217 で定義されているトランスポート層セキュリティー (TLS) をサポートします。TLS は、クライアントとサーバー間で安全な通信を行うための暗号プロトコルです。

このインプリメンテーションの主な目的は、暗号化を使用して制御とデータ接続を保護するためです。クライアントは、他の方法で認証されなければなりません。サポートされている方法は、標準認証メソッドのみです。

TLS セッションを開始するための要求を受信すると、**ftpd** デーモンは **/etc/ftpd.cnf** ファイルの読み取りを行い、TLS セッションのセットアップに使用される以下の構成パラメーターをロードします。

項目	説明
CRL_PATH	*CRL_PATH* パラメーターは、証明書取り消しリスト・ファイルへのパスを提供します。このファイルは PEM フォーマットでなければなりません。これを指定すると、クライアントが提供するデジタル証明書が証明書取り消しリストと照合されます。ftp クライアントがデジタル証明書を使用していない場合は、接続が失敗します。クライアントがデジタル証明書を提供している一方で、その証明書が取り消されている場合は、TLS セッションが失敗します。このパラメーターを指定しない場合は、クライアントはデジタル証明書を提供する必要はありません。
CA_PATH	*CA_PATH* パラメーターは、認証局ファイルへのパスを提供します。このファイルは PEM フォーマットでなければなりません。これを指定すると、クライアント証明書が認証局と照合されます。クライアントがデジタル証明書を提供しない場合は、接続が失敗します。クライアントがデジタル証明書を提供している一方で、その証明書がセキュリティー機関によって署名されていない場合は、TLS セッションが失敗します。このパラメーターを指定しない場合は、クライアントはデジタル証明書を提供する必要はありません。
CIPHER_LIST	*CIPHER_LIST* パラメーターを指定すると、このリストが TLS セッション中に使用されます。これを指定しないと、デフォルトの暗号リストが使用されます。
DEPTH	*CA_PATH* 構成パラメーターを指定すると、デジタル証明書階層内の ftp クライアントが提供する証明書を検証するために *DEPTH* 値が使用されます。これを指定しないと、デフォルト値の 9 が使用されます。
CERTIFICATE	*CERTIFICATE* パラメーターは、PEM フォーマットの有効なデジタル証明書チェーン・ファイルへのパスを提供します。このファイルは TLS セッションで使用されます。このパラメーターは、TLS セッションを開始するために指定する必要があります。このパラメーターを指定しないと、ftpd サーバーはすべての TLS 要求を拒否します。
CERTIFICATE_PRIVATE_KEY	*CERTIFICATE_PRIVATE_KEY* パラメーターは、PEM フォーマットの証明書の秘密鍵へのパスを提供します。これは TLS セッション中に使用されます。このパラメーターは、TLS セッションを開始するために指定する必要があります。このパラメーターを指定しないと、ftpd サーバーはすべての TLS 要求を拒否します。
DH_PARAMETERS_DIR	*DH_PARAMETERS_DIR* パラメーターは、PEM フォーマットの *Diffie Helman* パラメーターを含むディレクトリーへのパスを提供します。このディレクトリーには、PEM フォーマットの *Diffie Helman* パラメーターを含むファイルが複数入ります。**ftpd** デーモンは、必要に応じて、使用するパラメーターを検索します。

TLS をサポートするには、AIX Web Download Pack Programs Web サイトから最新バージョンの OpenSSL ツールをインストールする必要があります。

ファイル転送プロトコル・サブツリーのガイドライン

無名 FTP ユーザーの処理では、サーバーは FTP ユーザー・アカウントのホーム・ディレクトリーで **chroot** コマンドを実行します。セキュリティーを強化するために、FTP サブツリーの構成時に次の規則を実行します。

項目	説明
~ftp	ホーム・ディレクトリーを root の所有とし、モードを r-xr-xr-x (555) にする。
~ftp/bin	このディレクトリーを root ユーザーの所有とし、他のユーザーからの書き込みができないようにする。リスト・コマンドをサポートするために、このディレクトリー内には **ls** プログラムがなければなりません。また、このプログラムはモード 111 を持つ必要があります。
~ftp/etc	このディレクトリーを root ユーザーの所有とし、他のユーザーからの書き込みができないようにする。
~ftp/pub	このディレクトリーのモードを 777 とし、FTP の所有とします。ユーザーはこのディレクトリー内の無名のアカウントを介してアクセス可能になるようなファイルを配置します。

注**:** シェル・スクリプト **/usr/samples/tcpip/anon.ftp** は、上記の規則に従って無名 FTP アカウントをセットアップします。

/etc/ftpaccess.ctl に定義されている無名 FTP ユーザーを処理する場合、サーバーは FTP ユーザー・アカウントのホーム・ディレクトリーで **chroot** コマンドを実行します。セキュリティーを強化するために、ユーザーのサブツリーの構成時に次の規則を実行します。

~user ホーム・ディレクトリーを root の所有とし、モードを r-xr-xr-x (555) にする。

~user/bin
このディレクトリーを root ユーザーの所有とし、他のユーザーからの書き込みができないようにする。リスト・コマンドをサポートするために、このディレクトリー内には **ls** プログラムがなければなりません。また、このプログラムはモード 111 を持つ必要があります。

~user/etc
このディレクトリーを root ユーザーの所有とし、他のユーザーからの書き込みができないようにする。

~user/pub
このディレクトリーのモードを 777 とし、ユーザーの所有とする。ユーザーはこのディレクトリー内の無名のアカウントを介してアクセス可能になるようなファイルを配置します。

注**:** シェル・スクリプト **/usr/samples/tcpip/anon.users.ftp** は、上記の規則に従って無名 FTP アカウントをセットアップします。

サーバーは特権ポート番号付きのソケットを作成するために、root ユーザーとして実行されなければなりません。サーバーはログイン済みの実効ユーザー ID を保持し、ソケットにアドレスをバインドするときだけ root ユーザーに変わります。

サポートされるファイル転送プロトコル要求

ftpd デーモンは現在次の FTP 要求をサポートしています。

項目	説明
ABOR	前のコマンドを終了させる。
ACCT	アカウントを指定する (無視される)。
ADAT	認証/セキュリティー・データを指定します。
ALLO	保存域を割り当てる (中は空白)。
APPE	ファイルに追加する。
AUTH	認証/セキュリティー・メカニズムを指定します。
CCC	コマンド・チャネル・クリアを指定します。
CDUP	現在の作業ディレクトリーの親ディレクトリーに変更する。
CWD	作業ディレクトリーを変更する。
DELE	ファイルを削除する。
ENC	プライバシー保護コマンドを指定します。
HELP	ヘルプ情報を提供する。

項目	説明
LIST	ディレクトリーにリスト・ファイルを提供する (**ls -lA** コマンドと同じ)。
MKD	ディレクトリーを作成する。
MDTM	ファイルを最後に変更した時刻を明示する。
MIC	保全性保護コマンドを指定します。
MODE	データ転送モードを指定する。
NLST	ディレクトリーにリスト・ファイルを提供する (**ls** コマンドと同じ)。
NOOP	何もしない。
PASS	パスワードを指定する。
PASV	サーバーからサーバーへの転送の準備をする。
PBSZ	保護バッファー・サイズを指定します。
PORT	データ接続ポートを指定する。
PROT	データ・チャネル保護レベルを指定します。
PWD	現在の作業ディレクトリーを印刷する。
QUIT	セッションを終了する。
RETR	ファイルを検索する。
RMD	ディレクトリーを除去する。
RNFR	改名前のファイル名を指定する。
RNTO	改名後のファイル名を指定する。
SITE	**SITE** 要求では、次の非標準コマンドまたは UNIX 固有コマンドがサポートされます。 **UMASK** umask を変更する (**SITE UMASK 002**)。 **IDLE** アイドル時間を設定する (**SITE IDLE 60**)。 **CHMOD** ファイルのモードを変更する (**SITE CHMOD 755** *FileName*)。 **HELP** ヘルプ情報を提供する (**SITE HELP**)。
SIZE	現行ファイルのサイズを戻す。
STAT	サーバーの状況を戻す。
STOR	ファイルを格納する。
STOU	固有のファイル名を使用してファイルを格納する。
STRU	データ転送の構造をファイル構造として指定する。
SYST	サーバー・システムのオペレーティング・システム・タイプを明示する。
TYPE	データ転送タイプを *Type* パラメーターで指定する。
USER	ユーザー名を指定する。
XCUP	現在の作業ディレクトリーの親ディレクトリーに変更する (通常は使用しない)。
XCWD	現行ディレクトリーを変更する (通常は使用しない)。
XMKD	ディレクトリーを作成する (通常は使用しない)。
XPWD	現在の作業ディレクトリーを表示する (通常は使用しない)。
XRMD	ディレクトリーを除去する (通常は使用しない)。

インターネット RFC 959 に定義された残りの FTP 要求は、認識されますが実行はされません。 **MDTM** および **SIZE** 要求は RFC 959 で指定されていませんが、次の改訂版 FTP RFC に登場する予定です。

STAT 要求がデータ転送中に受信されて、Telnet **IP** シグナルおよび **SYNCH** シグナルの後に来る場合には、転送状況は再実行されます。

ftpd デーモンは System Management Interface Tool (SMIT) を使うか、または **/etc/inetd.conf** ファイルを変更して制御します。 `ftpd` をコマンド・ラインに入力することはお勧めしません。

システム・リソース・コントローラーによる **ftpd** デーモンの操作

ftpd デーモンは、システム・リソース・コントローラー (SRC) サブシステムである、 **inetd** デーモンの、サブサーバーです。**ftpd** デーモンは **tcpip** SRC サブシステム・グループのメンバーです。このデーモンはデフォルトでは **/etc/inetd.conf** ファイルで使用可能になり、次の SRC コマンドで操作できます。

項目	説明
startsrc	サブシステム、サブシステム・グループ、またはサブサーバーを始動します。
stopsrc	サブシステム、サブシステムのグループ、またはサブサーバーを停止します。
lssrc	サブシステム、サブシステムのグループ、あるいはサブサーバーの状況を取得します。

フラグ

項目	説明
-C	**send_file** コマンドを使用して送信される発信ファイルをネットワーク・バッファー・キャッシュ (NBC) に入れなければならないことを指定します。このフラグは、**-q** フラグが指定されていなければ指定できません。このフラグは、ファイルがバイナリー・モードで保護されずに送信される場合にのみ適用されます。
-c	ホスト名のリバース・ルックアップを抑制します。
-d	**ftpd** デーモンの動作についてのデバッグ情報を **syslogd** デーモンに送信します。**-d** フラグを指定する場合には、**/etc/syslog.conf** ファイルを編集して次のエントリーを追加する必要があります。 `daemon.debug FileName` **注: syslogd** デーモンの `debug` レベルには、`info` レベルのメッセージが含まれます。 **/etc/syslog.conf** ファイルを編集しないと、メッセージはなにも生成されません。 **/etc/syslog.conf** ファイルを変更した後、 **refresh -s syslogd** コマンドまたは、 **kill-1***SyslogdPID* コマンドを実行して、 **syslogd** デーモンに、構成ファイルに加えた変更内容を通知してください。デバッグ・レベルの詳細については、**/etc/syslog.conf** ファイルを参照してください。
-D *DataConnTimeOut*	**ftpd** デーモンがデータ接続を保持する最大秒数を指定します。デフォルト値は 300 秒です。待機時間を無限にする場合にはこの値を 0 に指定します。*DataConnTimeOut* パラメーターの値は 0 から *MAXINT* までの範囲で指定できます。
-e	TLS 対応のクライアントのみがサーバーとの接続を確立できるようにします。
-f	クライアントがサーバーに特定ポートに戻る接続を要求した場合は、特権ポートの確認を使用不可にします。デフォルトでは、**ftpd** ではクライアントは、セキュリティー上の用心から特権ポートへの接続を要求することはできません。
-ff	クライアントがサーバーに、特定のクライアント・ポートに戻る接続を要求した場合は、特権ポートと、制御接続に使用されるアドレスと一致する IP アドレスの、両方の確認を使用不可にします。このフラグを使用すると、クライアントがサーバーに、代替ホストまたは代替インターフェースにデータを送信するよう要求できるようになります。デフォルトでは、**ftpd** はこのアクションを、セキュリティー上の用心から許可しません。
-H	`FILE_Rename`、`FS_Rmdir`、および `FILE_Unlink` の各イベントの監査ロギングをオンにします (これらのイベントが root ユーザーに対して有効にされている場合)。
-k	データ転送ソケット上の **sys/socket.h** ファイルに定義された **SO_KEEPALIVE** オプションを設定して、イベント TCP/IP がハングした場合にデータ転送がタイムアウトになるようにします。アイドル状態の間隔時間は、 **no** コマンドの、 `tcp_keepidle` オプションと、 `tcp_keepintvl` オプションで指定された、システム全体に関する値に依存します。このフラグがないと、**ftpd** データ転送はタイムアウトになりません。
-l	**ftpd** デーモンの動作についてのロギング情報を **syslogd** デーモンに送信します。**-l** フラグを指定した場合、**/etc/syslog.conf** ファイルを編集して、次のエントリーを追加しなければなりません。 `daemon.info FileName` **/etc/syslog.conf** ファイルを編集しないと、メッセージはなにも生成されません。 **/etc/syslog.conf** ファイルを変更した後、 **refresh -s syslogd** コマンドまたは、 **kill-1***SyslogdPID* コマンドを実行して、 **syslogd** デーモンに、構成ファイルに加えた変更内容を通知してください。デバッグ・レベルの詳細については、**/etc/syslog.conf** ファイルを参照してください。
-q	ネットワーク上のファイルの送信に **send_file** サブルーチンを使用しなければならないことを指定します。このフラグは、ファイルがバイナリー・モードで保護されずに送信される場合にのみ適用されます。
-t*TimeOut*	*TimeOut* 変数で指定された秒数後にアクティブでないセッションをログアウトします。デフォルトのリミットは 15 分 (900 秒) です。タイムアウトはデータ接続と制御接続の両方に適用されます。
-T *MaxTimeOut*	*MaxTimeOut* 変数で指定された最大秒数後にアクティブでないクライアント・セッションをログアウトします。デフォルトのリミットは 2 時間 (7200 秒) です。
-s	ソケット・レベルのデバッグをオンにします。

項目	説明
-u *OctalVal*	**ftpd** デーモンの umask を設定します。*OctalVal* 変数は、umask を定義する 8 進数として指定しなければなりません。デフォルトの umask は 027 という 8 進数で、これは rw-r--- というファイル許可になります。
-U	転送中にファイルをアンロックしたままにします。このフラグが **/usr/sbin/ftpd** と一緒に指定されている場合には、転送中でもファイルをオープンできます。

セキュリティー

ftpd デーモンは、サービス名が *ftp* の PAM 使用可能なアプリケーションです。認証に PAM を使用するようにシステム全体の構成を設定するには、root ユーザーとして **/etc/security/login.cfg** の **usw** スタンザにある **auth_type** 属性の値を PAM_AUTH に変更します。

PAM が使用可能であるときに使用される認証メカニズムは、**/etc/pam.conf** の **ftp** サービスに対する構成によって決まります。 **ftpd** デーモンは、**auth**、**account**、および **session** モジュール・タイプに対して **/etc/pam.conf** エントリーを必要とします。下記に、**ftp** サービスのための **/etc/pam.conf** に推奨される構成をリストします。

```
#
# AIX ftp configuration
#
ftp auth      required     /usr/lib/security/pam_aix

ftp account   required     /usr/lib/security/pam_aix

ftp session   required     /usr/lib/security/pam_aix
```

例

注**: ftpd** デーモンの引数は、SMIT を使用することによって、または **/etc/inetd.conf** ファイルを編集することによって指定できます。

1. **ftpd** デーモンを始動するには、次のように入力します。

   ```
   startsrc -t ftp
   ```

   -t フラグを指定して **startsrc** コマンドを実行すると、**ftpd** サブサーバーが始動されます。サブサーバーを指定するには、**-t** フラグを使用しなければなりません。このフラグを使用しないと、コマンドは正常に実行されません。
2. **ftpd** デーモンを停止するには、通常、次のように入力します。

   ```
   stopsrc -t ftp
   ```

   -t フラグを指定して **stopsrc** コマンドを実行すると、**ftpd** サブサーバーが停止されます。**stopsrc** コマンドを使用すると、保留中のすべての接続を始められ、既存の接続をすべて完了させ、新たな接続が始められるのを防ぐことができます。サブサーバーを指定するには、**-t** フラグを使用しなければなりません。このフラグを使用しないと、コマンドは正常に実行されません。
3. **ftpd** デーモンとすべての **ftpd** 接続を強制的に停止させるには、次のように入力します。

   ```
   stopsrc -f -t ftp
   ```

   -t フラグと **-f** フラグを指定して **stopsrc** コマンドを実行すると、**ftpd** サブサーバーが強制的に停止されます。また、保留中のすべての接続と既存の接続が即座に終了します。
4. **ftpd** デーモンについての簡略状況報告を表示するには、次のように入力します。

   ```
   lssrc -t ftp
   ```

-t フラグを指定した **lssrc** コマンドは、デーモンの名前、プロセス ID、および状態 (アクティブまたは非アクティブ) を戻します。サブサーバーを指定するには、**-t** フラグを使用しなければなりません。このフラグを使用しないと、コマンドは正常に実行されません。

ファイル

項目	説明
/etc/locks/ftpd	インターロックおよびプロセス ID (PID) ストレージが入っています。
/etc/group	グループ用のパスワードが入っています。
/etc/passwd	ユーザー用のパスワードが入っています。
/etc/security/login.cfg	ログインおよびユーザー認証のための構成情報が入っています。
/etc/security/passwd	暗号化されたパスワードが入っています。
/etc/syslog.conf	**syslogd** デーモンの構成情報が入っています。
/usr/samples/tcpip/anon.ftp	無名の FTP アカウントを設定するためのシェル・スクリプト例が入っています。このファイルにはその使い方の説明も入っています。
/etc/ftpd.cnf	TLS サポート用の構成パラメーターが入っています。

関連情報:

rlogin コマンド

telnet コマンド

syslogd コマンド

kvalid_user サブルーチン

/etc/ftpusers ファイル

---

fuser コマンド

目的

ファイルまたはファイル構造を使ってプロセスを識別します。

構文

fuser [[**-c** | **-C** | **-f**] [**-x**] |**-d**] [**-k** | **-K** { *SignalNumber* | *SignalName* }] [**-u**] [**-V**]*File* ...

説明

fuser コマンドは、*File* パラメーターで指定されたローカルまたはリモート・ファイルを使用しているローカル・プロセスのプロセス番号を表示します。ブロック特殊デバイスの場合は、コマンドがデバイス上のいずれかのファイルを使用するプロセスを表示します。

個々のプロセス番号の後には、そのプロセスのファイルの使い方を示す次のような文字が付けられます。

項目	説明
c	ファイルを現行ディレクトリーとして使用します。
e	ファイルをプログラムの実行可能なオブジェクトとして使用します。
r	ファイルをルート・ディレクトリーとして使用します。
s	ファイルを共用ライブラリー (またはその他のロード可能なオブジェクト) として使用します。

プロセス番号は、各プロセス番号の間にスペースを挿入して 1 行で標準出力に書き出されます。それぞれのファイル・オペランドごとに、改行文字が最後の出力の後で標準エラーに書き込まれます。その他の出力はすべて標準エラーに書き込まれます。

fuser コマンドは、関連ファイル・ディスクリプターがクローズされた mmap 領域を持つプロセスは検出しません。また、FIFO (名前付きパイプ) を使用するプロセスは、FIFO が完全にオープンするまで検出されません。例えば、オープン・システム呼び出しの完了を待っているプロセスは、**fuser** コマンドでは認識されません。

fuser コマンドは、ファイルシステムを使用しているプロセスを決定するのに使用します。このファイルシステムがネットワーク・ファイルシステム (NFS) であり、NFS サーバーが応答していない場合、**fuser** コマンドがハングすることがあります。この状態を回避するため、FUSER_VERSION 環境変数を 1 に設定することができます。

フラグ

項目	説明
-c	*File* が入っているファイルシステム内のすべてのオープン・ファイルについて報告します。
-C	**File** パラメーターが指定するディレクトリーにマウントされたファイルシステム内にオープン・ファイルがあれば、それらのすべてのファイルについて報告します。**File** パラメーターがマウント・ポイントではない場合、コマンドはエラーを報告します。
-d	*File* を含むファイルシステムからリンク解除された (削除された) すべてのオープン・ファイルについて報告します。**-V** フラグと一緒に使用された場合は、削除されたファイルの i ノード番号とサイズも報告します。
-f	*File* のオープン・インスタンスについてのみ報告します。
-K *SignalNumber* | *SignalName*	指定のシグナルを各ローカル・プロセスに送信します。 root ユーザーのみが他のユーザーのプロセスを強制終了できます。シグナルには *SignalName* (SIGKILL シグナルの KILL など) または *SignalNumber* (9 など) を指定できます。*SignalName* の有効な値は、**kill -l** コマンドで表示されるものです。
-k	**SIGKILL** シグナルを各ローカル・プロセスに送信します。root ユーザーのみが他のユーザーのプロセスを強制終了できます。 **注: fuser -k** または **-K** では、プログラムが実行のための始動直後に作成された新規プロセスは検出することも強制終了することもできないことがあります。
-u	プロセス番号の後に括弧付きでローカル・プロセス番号のログイン名を提供します。
-V	詳細出力を提供します。
-x	**-c** または **-f** と一緒に使用された場合は、標準の fuser 出力に加えて、実行可能で、かつ、ロード可能なオブジェクトを報告します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. **/etc/passwd** ファイルを使用している、ローカル・プロセスのプロセス番号を表示するには、次のように入力します。

   ```
   fuser /etc/passwd
   ```

2. **/etc/filesystems** ファイルを使用しているプロセスのプロセス番号とユーザー・ログイン名を表示するには、次のように入力します。

   ```
   fuser -u /etc/filesystems
   ```

3. 所定のファイルシステムを使用しているすべてのプロセスを停止するには、次のように入力します。

   ```
   fuser -k -x -u -c /dev/hd1
   ```

   または

```
fuser -kxuc /home
```

いずれのコマンドもプロセス番号とユーザー名をリストした後、**/dev/hd1 (/home)** ファイルシステムを使用しているプロセスを停止します。root ユーザーのみが他のユーザーに属するプロセスを停止できます。**/dev/hd1** ファイルシステムをアンマウントしようとしていて、**/dev/hd1** ファイルシステムにアクセスしているプロセスがこれを阻止するのであれば、このコマンドを使用します。

4. あるファイルシステムから削除されたファイルを使用している、すべてのプロセスを表示するには、次のように入力します。

```
fuser -d /usr
```

ファイル

項目	説明
/dev/kmem	システム・イメージに使用されます。
/dev/mem	同じく、システム・イメージに使用されます。

関連情報:

kill コマンド

mount コマンド

ps コマンド

セキュリティー

---

fwtmp コマンド

目的

標準入力から **wtmp** フォーマットのバイナリー・レコードを読み取り、フォーマットされた ASCII レコードに変換することによって、接続時アカウンティング・レコードを処理します。不良なレコードを編集するには ASCII バージョンを使用することができます。

構文

/usr/sbin/acct/fwtmp [**-i**] [**-c**] [**-X**] [**-L**]

説明

fwtmp コマンドは、標準入力から **wtmp** フォーマットのバイナリー・レコードを読み取り、それらをフォーマットされた ASCII レコードに変換して、アカウンティング・レコードを処理します。

フラグ

項目	説明
-i	**utmp** フォーマットの ASCII レコードを入力として受け入れます。
-c	出力を **utmp** フォーマットのバイナリー・レコードに変換します。
-ic	ASCII **utmp** フォーマットの入力レコードをバイナリー出力レコードに変換します。
-X	最初の 8 文字に切り捨てることをせずに、各ユーザー名の使用可能なすべての文字を印刷します。
-L	各ホスト名は、最初の 32 文字に切り捨てられずに、使用可能なすべての文字が印刷されます。

セキュリティー

アクセス制御: これらのコマンドは **adm** グループのメンバーにのみ実行 (x) アクセス権を与えます。

例

1. **wtmp** フォーマットのバイナリー・レコードを `dummy.file` という ASCII レコードに変換するには、次のように入力します。

   ```
   /usr/sbin/acct/fwtmp < /var/adm/wtmp > dummy.file
   ```

   バイナリー・ファイル **wtmp** の内容は、ダミーの ASCII ファイルにリダイレクトされます。

2. ASCII ファイル `dummy.file` を `/var/adm/wtmp` という **wtmp** フォーマットのバイナリー・ファイルに変換するには、次のように `-ic` スイッチを指定して `fwtmp` コマンドを入力します。

   ```
   /usr/sbin/acct/fwtmp  -ic < dummy.file  > /var/adm/wtmp
   ```

   ダミーの ASCII ファイルはバイナリー・ファイル **wtmp** にリダイレクトされます。

ファイル

項目	説明
/usr/sbin/acct/fwtmp	**fwtmp** コマンドが入っています。
/var/adm/wtmp	古い日付と新しい日付からなる日付変更レコードが入っています。
/usr/include/utmp.h	理由、日付、時刻からなるヒストリー・レコードが入っています。

関連情報:

runacct コマンド

wtmpfix コマンド

アカウンティング・システムのセットアップ

---

fxfer コマンド

目的

ローカル・システムと HCON で接続されたホスト・コンピューターとの間でファイルを転送します。

構文

中断されたファイル転送を再開する

fxfer -R [-n *SessionName* **]**

ファイルをホストからダウンロードする

fxfer [-n *SessionName* **] [-a** | **-r] [-d] [-c** | **-C] [-J] [-f** *FileName* **] [-F] [-H** *HostType* **][-I** *InputField* **] [-q] [-t [[-l] [-s] [-b]]** | **-T [[-l] [-s] [-b]]]**

[**-v**] [**-x** *HostLogin*] [**-e**] [**-X** *CodeSet*] *SourceFile DestFile*

ファイルをホストへアップロードする

fxfer [**-n** *SessionName*] [**-a** | **-r**] [**-u**] [**-c** | **-C**] [**-J**] [**-f** *FileName*] [**-H** *HostType*] [**-q**] [**-t** [[**-l**] [**-s**]] | **-T** [[**-l**] [**-s**]]] [**-l**] [**-s**] [**-v**] [**-x** *HostLogin*] [**-X** *CodeSet*] [**-F** | **-V** | **-U**] [**-B** *BlockSize*] [**-L** *LoglRecLength*] [**-I** *InputField*] [**-S** *NumberUnits* [**,***IncreaseUnits* | **,***IncreaseUnits*,*UnitType* | **,,***UnitType*]] [**-M** *Volume*] [**-N** *Unit*] [**-k**] *SourceFile DestFile*

ヘルプ画面を表示する

fxfer -h

説明

fxfer コマンドは、ローカル・システムと、ホスト接続プログラム (HCON) で接続されたメインフレーム・ホストとの間でファイルを転送します。ファイルは、ローカル・システムからホストへ (アップロード)、またはホストからローカル・システムへ (ダウンロード) 転送されます。 **fxfer** コマンドは、*SourceFile* パラメーターで指定したファイルを、 *DestFile* パラメーターで指定したファイルへ転送します。この転送は、特定のセッション・プロファイル、または既存のセッションを必要とする HCON セッションで行われます。

ホスト・オペレーティング・システムには、VM/CMS、MVS/TSO、CICS/VS (CICS/MVS または CICS/VSE の場合)、VSE/ESA、VSE/SP などがあり、対応するバージョンの 3270 ファイル転送プログラム (**IND$FILE** またはそれと等価なもの) が一緒にインストールされています。ホスト・ファイル転送プログラムのバージョンは、セッション・プロファイル内のファイル転送プログラム値によって決定されます。 **fxfer** コマンドは、テキスト・データとバイナリー・データのどちらの転送もサポートします。ファイルは、ホストへまたはホストから転送できます。このとき ASCII 変換や EBCDIC 変換は行われる場合と行われない場合があります。

セキュリティー機構によって無許可のアクセス、現存ファイルの破壊、データ消失が防止されます。 **fxfer** コマンドを実行するユーザーが HCON ユーザーでない場合は、コマンドは実行されません。 **fxfer** コマンドが、完了する前に中断された場合、転送の状態が RESTART ファイルに保存されます。

-h フラグを指定して、 **fxfer** コマンドを実行すると、ヘルプ画面が表示されます。 **-R** フラグを指定して実行すると、 **$HOME** ディレクトリー内で再始動ファイルが検索されます。再始動ファイルがあれば、再始動メニューが表示され、ファイル転送の再始動が可能になります。 **-h** も **-R** フラグも指定されていない場合は、コマンドは指定されたファイル転送を実行します。

fxfer コマンド情報は次のとおりです。

- フラグ
- ホスト・ファイル特性を示すフラグ
- 例
- ファイル

このコマンドには次のものが必要です。

- メインフレーム・ホストに接続するための 1 つ以上のアダプター。
- 次のメインフレーム・オペレーティング・システムのいずれか 1 つが、ホスト上にインストールされていること。
  - VM/SP CMS

  - VM/XA CMS
  - MVS/SP TSO/E
  - MVS/XA TSO/E
  - CICS/VS (CICS/MVS または CICS/VSE の場合)
  - VSE/ESA
- メインフレームの、ホスト・サポート・ファイル転送プログラム (**IND$FILE**、またはそれと等価のもの) がメインフレーム上にインストールされていること。

fxfer コマンドを使用するセッション・プロファイル

fxfer コマンドは HCON セッションと通信し、特定のセッション・プロファイルを要求することができます。セッション・プロファイルは次のものを定義します。
- ホストへの通信パス
- ホスト・タイプ
- デフォルトのファイル転送方向 (ダウンロードかアップロードか)
- リカバリー時間
- ファイル転送待ち期間

fxfer コマンドが自動ログオンを実行している時は、プロファイルはさらに次のものを定義することができます。
- ホスト・ログオン ID
- AUTOLOG ノード ID
- AUTOLOG トレースがオンになっているか
- AUTOLOG タイムアウト値

ユーザーは通常、**fxfer** コマンドの起動中にセッション・プロファイルを指定します。例外は、このコマンドが既存セッションのサブシェルから実行される場合です。この場合、ユーザーがセッション・プロファイルを指定しなければ、**fxfer** コマンドは既存セッションを使用します。適切なセッションが実行されていない場合、**fxfer** コマンドは新しいセッションを起動しようとします。

fxfer コマンドは次のような HCON セッションを検索します。
- **-n** *SessionName* フラグを指定しないで実行した場合:
  - **fxfer** コマンドが、既存セッションのサブシェルから実行されている場合、このコマンドは、サブシェル (**$SNAME** 環境変数によって定義) と関連したセッションを使用します。
  - コマンドが、エミュレーター・セッションのサブシェルから実行されていない 場合、 **fxfer** コマンドはエラー・メッセージを発行して終了します。
- **-n** *SessionName* フラグを指定して実行されている場合、ファイル転送は指定されたセッションで実行されます。指定されたセッションが存在しない場合、コマンドはそのセッションを求めてセッション・プロファイルを検索します。指定されたセッション・プロファイルが検出されない場合、**fxfer** コマンドはエラー・メッセージを発行して終了します。指定されたセッション・プロファイルが存在する場合、**fxfer** コマンドは、セッション・プロファイル内で定義された AUTOLOG 値または **-x** フラグで定義された値を使って、あるいはユーザーに必要なログイン情報を入力するよう指示して、ホストに自動ログオンを試みます。

ファイル転送の中断と再開

fxfer コマンドは完了前にオペレーターによって、また、回復不能な通信エラーによって中断される場合があります。中断された場合コマンドは、RESTART ファイルに転送状況を保存します。転送はデータを失わずに最初から再開されます。

転送の中断後に新しいファイルの転送を実行しようとすると、**fxfer** コマンドは RESTART ファイルが作成済みであることを通知し、次の選択肢を表示します。

- 中断したファイル転送を再開する。
- RESTART ファイルを保存し、ファイル転送プログラムを終了する。
- RESTART ファイルを削除し、ファイル転送プログラムを終了する。
- RESTART ファイルを削除し、現在の転送を続行する。

fxfer コマンドを、 **-R** フラグを指定して実行しても、中断されたファイル転送を再開できます。

自動ログオンで開始されたファイル転送中にホスト通信が失われるか接続が切断された場合には、ファイル転送はホストへの再接続や再ログオンを試みて転送を再開しようとします。リカバリー時間は、セッション・プロファイルの File Transfer Recovery Time 値により決定されます。ホスト接続が再設定されると、ファイル転送は最初から再開されます。通信が再設定できない場合は、ファイル転送プログラムが RESTART ファイルを生成します。

明示的なファイル転送がホストと通信できなくなった場合、ユーザーはファイル転送を再開する前にエミュレーター・セッションを再開し、ホストにログインし直す必要があります。

ソース・ファイルと宛先ファイル

fxfer コマンドには *SourceFile* および *DestFile* パラメーターが必要です。*SourceFile* パラメーターはファイル転送のためのソース・ファイルを指定します。 *DestFile* パラメーターはファイル転送のための宛先ファイルを指定します。ローカル・システムのファイル名は、標準のフォーマットどおりです。ホスト・ファイル名は次のいずれかのフォーマットのホスト命名規則に準拠します。

ホスト・タイプ	ファイル名のフォーマット
VM/CMS	**"***FileName FileType FileMode***"** **注:** ファイル転送を正しく行うために、すべての VM/CMS ファイル名に **" "** (二重引用符) が必要です。
MVS/TSO	**"['']***DataSetName* **[(***MemberName***)] [/***Password* **][']"** それぞれの意味は次のとおりです。 *DataSetName* 物理順次データ・セットまたは区画データ・セットを示します。 **(***MemberName***)** 既存の区画データ・セットのディレクトリー内にあるメンバー名を示します。 *MemberName* は **()** (小括弧) で囲む必要があります。 **/***Password* MVS/TSO データ・セット用にパスワード保護が指定されている場合に必要です。*Password* の前に **/** (スラッシュ) が必要です。 **注:** 1. ファイル転送を正しく行うために、すべての MVS/TSO ファイル名に **" "** (二重引用符) が必要です。 2. MVS/TSO ファイル名の完全なパス名を指定する場合は、**"** (二重引用符) の中に **'** (単一引用符) を使用します。単一引用符と二重引用符の間や引用符とファイル名の間には、スペースを挿入しないでください。
CICS/VS	**"***FileName***"**

ホスト・タイプ	ファイル名のフォーマット
VSE/ESA	**"***FileName FileType***"**

注:

1. ファイル転送を正しく行うために、CICS/VS、VSE/ESA、および VSE/SP のすべてのファイル名に **" "** (二重引用符) が必要です。
2. CICS/VS、VSE/ESA、および VSE/SP の各ファイル名の規則では、最高 8 文字までの長さのファイル名が使用できます。
3. DBCS 環境では、HCON は VSE ホストをサポートしません。

フラグ

注**:** 日本語プラス英語、日本語カタカナ、韓国語、中国語 (繁体字) のいずれかを含む 2 バイト文字セット (DBCS) のサポートの場合は、以下の注意事項が適用されます。

- DBCS の **-l** フラグ、または **-s** フラグを指定した場合は、変換フラグ (**-t**、**-T**、または **-J**) のいずれか 1 つも指定しなければなりません。指定しないと DBCS フラグは無視されます。
- **-M**、**-N**、および **-k** フラグは MVS/TSO ホストの場合にのみ使用されます。
- **-e** フラグは、CICS® プログラムでダウンロードを行う場合にのみ有効です。
- **-b** フラグはダウンロードにのみ使用できます。

項目	説明
-a	宛先ファイルが存在する場合、 *SourceFile* で指定したファイルを、 *DestFile* で指定したファイルに付加します。*DestFile* パラメーターで指定したファイルが存在しない場合、このフラグは無視され、宛先ファイルが作成されます。 **注: -a** フラグは、ファイルを CICS/VS ホストへアップロードするときには無効です。VSE/ESA の場合は、CICS 一時記憶域 (FILE=TS) にアップロードする場合にのみ **-a** フラグは有効です。
-b	**-t**、**-T**、**-c**、 **-C** フラグと一緒に使用される場合に、各レコードの終わりのブランクを保存します。**-b** フラグは DBCS 環境でのみサポートされます。
-c	DBCS 環境で、ファイル転送がアップロードの場合、**-c** フラグは、ファイルの LF (改行) コードを CRLF (復帰) に変更します。ダウンロード・ファイル転送の場合、**-c** フラグはファイルの CRLF コードを LF コードに変更します。
-C	DBCS 環境で、ファイル転送がアップロードの場合、**-C** フラグは、 PC-DOS ファイルの EOF (ファイルの終わり) コードの送信を禁止します。ファイル転送のダウンロードの場合は、**-C** フラグによって EOF コード `x'1A'` が PC-DOS ファイルの終わりに付加されます。
-d	ファイルをホストからローカル・システムに転送してダウンロードします。このフラグと **-u** フラグのどちらも指定しない場合、セッション・プロファイルのファイル転送方向特性によって転送の方向が決まります。 **注:** VSE/ESA ホスト・ファイル転送 (FILE=HTF) から変換されたファイルをダウンロードする場合、**-I** `"KEEP"` フラグを指定していないと、ファイルはホスト・システムから削除されます。
-e	ファイルの転送が完了した時点で、一時記憶域キューを削除します。このフラグは CICS ホストのダウンロードの場合以外には使用しないでください。 **-e** フラグは DBCS 環境でのみサポートされます。
-f *FileName*	*FileName* 変数で指定したファイルに、ファイル転送処理診断出力 (またはファイル転送状況) を入れます。 ある非同期転送に対して、**-f** フラグが指定されていない場合は、メッセージは **$HOME/hconerrors** ファイル内に入れられます。同期転送に対して **-f** フラグが指定されていない場合は、メッセージは標準出力に送られます。 ファイル転送パラメーターやファイル名の指定エラーまたはファイル転送の障害などによるメッセージは、標準出力 (ローカル・システムの画面の場合) または **$HOME/hconerrors** ファイル (標準がローカル・システムの画面でない場合) に直接出力されます。

項目　説明

-h　**fxfer** コマンドに応じてヘルプ画面を表示します。この画面の内容は、有効なコマンド・フラグとコマンド操作に関する要約です。このフラグが指定されると、その他のフラグは無視され、ファイルは転送されません。

注:

1. **-h** フラグを使うと、他のフラグはすべて無視されます。ファイル転送は行われません。
2. **fxfer** コマンドを既存の HCON セッションのサブシェルから起動しない場合は、**-h** フラグまたは **-n** フラグが必要です。

-H *HostType*　ホストのタイプを指定します。有効な *HostType* 変数は次の値のどれかです。

CMS　VM/SP CMS または VM/XA CMS

TSO　MVS/SP TSO または MVS/XA TSO

CICS　CICS/VS (CICS ホスト・タイプには CICS/VSE、CICS/MVS、 CICS/ESA、および CICS/MVS/ESA があります。)

VSE　VSE/ESA (DBCS 環境ではサポートされません。)

-H フラグが省略されると、Host Type 特性によりセッション・プロファイル内で指定された値が使われます。ユーザーは正しいホスト・オペレーティング・システムを指定しなければなりません。

注:

1. **CICS** または **VSE** の値を指定し、システムがエラーを戻した場合は、他の値を用いてコマンドを再実行してください。 CICS と VSE の **IND$FILE** プログラムは機能的互換性はありますが、両者のヘッダー・サイズに 6 バイトの相違があるため、CICS と VSE のバージョンの各 IND$FILE プログラムは運用的互換性はありません。宛先ホストのプログラムのバージョンが逆である可能性があります。
2. ファイルを MVS/TSO のホストに転送するには、転送を開始する前にセッション管理プログラム・モードから出る必要があります。

-I *InputField*　**IND$FILE** コマンド内に直接入れるホスト・ファイル転送オプションを指定します。 **IND$FILE** コマンドの) (右括弧) の後ろにコメントを記入することもできます。*InputField* 変数で指定する値は次のように引用符で囲みます。

```
-I  "FILE=TS)  This  is  a  comment"
```

注: -I フィールドは、DBCS 環境ではサポートされません。

-J　EBCDIC と ASCII 間でのデータ変換、および SI/SO 文字の正規化を可能にします。この変換は、次のように転送の方向によって決まります。

アップロード
: ファイルの 1 バイト文字を EBCDIC コードに変換します。DBCS を採用している国の場合は、拡張コードが適正な DBCS コードに変換されます。SO/SI 文字は、DBCS 文字が入っている DBCS フィールドに挿入されます。ファイルに制御コード 0x1E または 0x1F が入っている場合、それらのコードはそれぞれ、SO および SI 文字に置き換えられます。

ダウンロード
: EBCDIC コードをファイルの 1 バイト文字に変換します。DBCS を採用している国の場合は、DBCS コードが拡張コードに変換されます。DBCS フィールドから SO/SI 文字が削除されます。

注: -J フィールドは DBCS 環境でのみサポートされます。

-k　ファイルの転送が完了した時点で、データ・セット内の未使用のレコードを解放します。このフラグは MVS/TSO 環境以外では使用できません。 **-k** フラグは DBCS 環境でのみサポートされます。

-l　DBCS 環境でホスト言語を指定します。このオプションは、変換フラグ (**-t**、**-T**、**-J** のいずれか) と一緒に使用しなければなりません。 **-t**、**-T**、**-J** のいずれのフラグも設定していないと、**-l** フラグは無視されます。**-l** フラグを指定しないと、セッション・プロファイル内で定義されたホスト言語が使用されます。 **-l** フラグを指定すると、使用されるホスト言語はセッション・プロファイル内で定義した言語の代替言語となります。例えば、セッション・プロファイル内の言語特性が JPK (日本語のカタカナ) の場合、ファイル転送に使用されるホスト言語は日本語プラス英語となります。**-l** フラグは DBCS 環境でのみサポートされます。

-M *Volume*　データ・セット割り当てのためにホスト・ディスクのボリューム通し番号を指定します。このフラグは MVS/TSO 環境以外では使用できません。 **-M** フラグは DBCS 環境でのみサポートされます。

項目	説明
-n *SessionName*	ファイル転送を制御する特性を持つ、定義済みのセッション名を指定します。セッション名は a から z までの単一文字です。大文字は小文字として解釈されます。 **-n** *SessionName* フラグは必須です。ただし、ユーザーが既存のセッションのサブシェルから **fxfer** コマンドを開始する場合を除きます。この場合、**-n** フラグを使用しないと、**fxfer** コマンドはデフォルトにより既存セッションを使用します。 **注:** 1. 指定されたセッションは、**smit hcon** 高速パス・コマンドまたは **mkhcons** コマンドを使用して、それまでに定義しておく必要があります。 2. **fxfer** コマンドを既存の HCON セッションのサブシェルから起動しない場合は、**-h** フラグまたは **-n** フラグが必要です。
-N *Unit*	データ・セット割り当てのためにホスト・ディスクのデバイス・タイプを指定します。このフラグは MVS/TSO 環境以外では使用できません。 **-N** フラグは DBCS 環境でのみサポートされます。
-q	ファイル転送をバックグラウンド・プロセスとして非同期的に実行します。すべてのファイル転送が完了していないと、現行の転送要求はキューに入れられます。**-q** フラグを指定しないと、ファイル転送操作は同期で行われます。 **-f** フラグを指定しないと、診断出力と状況は **$HOME/hconerrors** ファイルに出力されます。 **注:** システムは、1 つのプロセス間通信 (IPC) メッセージ・キューで使用可能なバイト数を制限します。一度にキューに入れられる転送ファイルの最大数は、約 580 です。
-r	ホスト上に存在するファイルの置換 (アップロード) またはローカル・システム上に存在するファイルの置換 (ダウンロード) を指定します。ダウンロードでは、正常に転送された場合にのみ置換が行われます。これは、なんらかの理由で転送が完了しなかった場合に既存ファイルが消失または破壊されることを防ぐためです。 **-r** フラグが指定されていてもファイルが存在しないときは、転送時にファイルが作成されます。逆に、**-r** フラグの指定がなく、宛先ファイルが存在するときは、エラー・メッセージが出力されます。 アップロードの場合、MVS/TSO の PTF UR20455 または VM/CMS の PTF UR90118 以下のバージョンのホスト・ファイル転送プログラムを使うときは、**-r** フラグを指定しなければなりません。VSE および CICS では、**-r** フラグは無視されます。 **注:** ホスト・ファイル転送プログラムは通常、デフォルトでファイルを置換します。ホスト・ファイル転送プログラムがファイルを置換しない場合は、置換を指定するために、**fxfer** コマンドに、`-I "replace"` を追加します。 注意**:** ホスト上のファイルを置換する場合、既存のファイルの論理レコード長およびレコード・フォーマットと等しい、論理レコード長 (**-L** フラグ) と、レコード形式 (**-F**、または **-V** フラグ) を指定しなければなりません。指定しないと、データの破損が起こる場合があります。これは VSE/ESA の場合には適用されません。
-R	以前のファイル転送 (ユーザーまたはリカバリー試行の失敗によって中断されたもの) を、いずれかの RESTART ファイル (**$HOME/x_fxfer.r** ファイルまたは **$HOME/i_fxfer.r** ファイル) に保管されている情報を使って再開します。既存のセッションのサブシェルから起動されないファイル転送の場合は、使用するセッションを指定するため、**-n** *SessionName* フラグを指定する必要があります。**-R** フラグを指定した場合は、他のファイル転送フラグを指定しても、すべて無視されて、RESTART ファイル転送メニューが表示されます。 **注: -R** フラグを指定すると、**-n** *SessionName* フラグ以外のすべてのフラグが無視されます。RESTART ファイル転送メニューが表示されます。
-s	DBCS 環境のシフトイン・シフトアウト処理を指定します。 **-s** フラグは変更フラグ (**-t**、**-T**、**-J** のいずれか) と一緒に使用しなければなりません。 **-t**、**-T**、**-J** のいずれのフラグも指定してしないと、**-s** フラグは無視されます。 **-s** フラグを指定した場合、次の機能がファイル転送のために実行されます。 アップロード SO/SI 文字は DBCS フィールドに挿入されません。 ダウンロード SO/SI 文字は DBCS フィールド内で制御文字 (0x1E/0x1F) と置き換えられます。 **-s** フラグは DBCS 環境でのみサポートされます。

項目	説明
-t	ファイルの ASCII と EBCDIC 間の変換を実行します。**fxfer** コマンドは、ダウンロードの場合は EBCDIC から ASCII へ変換します。アップロードの場合、**fxfer** コマンドは ASCII を EBCDIC に変換します。使用言語は、そのセッション・プロファイルの Language 特性で指定します。 **-t** フラグで変換するファイルはテキスト・ファイルが前提です。改行文字は行区切り文字です。 **-t** フラグを他の DBCS サポート・フラグと一緒に DBCS 環境で使用した場合、**-t** フラグの働きは次のように変わります。 アップロード JISCII (日本語) または ASCII (韓国語、中国語 (繁体字)) を EBCDIC に変換します。SO/SI 文字を DBCS フィールドに挿入します。 ダウンロード EBCDIC を JISCII (日本語) または ASCII (韓国語、中国語 (繁体字)) に変換します。DBCS フィールドから SO/SI 文字が削除されます。
-T	ディスク・オペレーティング・システム・ファイルの ASCII と EBCDIC 間の変換が実行されます。文字シーケンス CRLF が行区切り文字として使用され、ディスク・オペレーティング・システム EOF (ファイルの終わり) 文字がダウンロード・ファイルの終わりに挿入されます。EBCDIC から ASCII への変換に使用される言語はセッション・プロファイル内の Language 特性によって指定されます。 **-T** フラグはディスク・オペレーティング・システム・ファイルの変換に使用されます。 **注: -T**、**-t**、**-J** フラグのいずれも指定しないと、ファイル転送で変換が行われず、情報がバイナリー・フォーマットで転送されます。
-u	ローカル・システムからホストへファイルを転送してそのファイルをアップロードします。このフラグと **-d** フラグのどちらも指定しない場合、セッション・プロファイルのファイル転送方向特性によって転送の方向が決まります。
-v	ファイル転送の現行状況を定期的に画面または **-f** フラグで指定された状況ファイルに書き込みます。その状況には、転送されたバイト数と、ファイル転送処理がデータ転送を開始してからの経過時間とが含まれます。
-x *HostLogin*	*HostLogin* 変数で指定されたログイン ID を使って、ホストへのログインを行います。ユーザーは、パスワードを入力するよう要求されます。 *HostLogin* 文字列は、そのホストのログイン ID、AUTOLOG ノード ID、その他のオプションの AUTOLOG 値で構成されます。文字列にブランクを使用することはできません。AUTOLOG のノード ID は必ず組み込まなければなりません。AUTOLOG 文字列のフォーマットは次のとおりです。 `UserID,AutologNodeID[,Trace,Time . . .]` **-x** フラグの指定がない場合は、 *HostLogin* 文字列に対する情報は、次のようにセッション・プロファイルから取り出されます。 • ホストのログイン ID がセッション・プロファイルに設定されていれば、パスワードの入力が要求されます。残りのパラメーターはプロファイルから検索されます。 • ホストのログイン ID がプロファイルに設定されていない場合、ホストのログイン文字列とパスワードの両方の入力が要求されます。 • プロンプトに応答すると、常にプロファイル・パラメーターが指定変更されます。例えば、プロファイルに AUTOLOG 時間が設定されていても、プロンプト上で異なる時間値を入力すると、その値が使われます。 ユーザーがホストのログイン文字列のパラメーターを入力していない場合は、パラメーターがプロファイルで定義されていれば、そこから取り出されます。例えば、AUTOLOG ノード ID、 AUTOLOG トレース、AUTOLOG 時間パラメーターをプロファイルに設定してあれば、プロンプトで入力する必要があるのは、ホストのログイン ID だけです。 ファイル転送プロセスは、ホストにログインし、**-n** フラグで指定されたセッション・プロファイルを使って、エミュレーション・セッションを設定します。いったんそのセッションが正常にログインされると、ファイル転送プログラムは実行を開始します。 セッション・プロファイルの File Transfer Wait Period パラメーターは、そのログイン・セッションが維持される時間数を決定します。このパラメーターを使うと、ホストのログイン・セッションは、以後のファイル転送にも使用されます。したがって、再度ログインする必要がなくなります。

項目	説明
-X *CodeSet*	ASCII と EBCDIC 間の変換で使用する代替コード・セットを指定します。**-X** フラグを指定しないと、システム・ロケールで指定したコード・セットが使用されます。次のコード・セットがサポートされます。 **Default** 現行システムの ASCII コード・ページを使用します。 **IBM-932** DBCS 環境での変換に IBM コード・ページ 932 を使用します。 **ISO8859-1** ISO 8859-1 ラテン文字 No.1 コード・ページを使用します。 **ISO8859-7** ISO 8859-7 ギリシャ文字を使用します。 **ISO8859-9** ISO 8859-9 トルコ文字を使用します。 **IBM-eucJP** 日本語環境での変換に IBM 拡張 UNIX コードを使用します。 **IBM-eucKR** 韓国語環境での変換に IBM 拡張 UNIX コードを使用します。 **IBM-eucTW** 中国語 (繁体字) 環境での変換に IBM 拡張 UNIX コードを使用します。

ホスト・ファイル特性を示すフラグ

次のフラグはホストのファイル特性を指定するもので、ファイルをアップロードするときにのみ使用されます (**-F** フラグのみは例外で、VSE ホストからダウンロードする時に使用されます)。

項目	説明
-B *BlockSize*	ホストのデータ・セットのブロック・サイズを指定します。**-B** フラグは、MVS/TSO 環境でかつ順次データ・セットに対してのみ使えます。 *BlockSize* 変数は、単一トラックの容量を超えられません。追加されていファイルの場合は、**-B** フラグは無視されます。ブロック・サイズを 0 にすると、エラーになります。
-F	固定長レコードを指定します。**-V**、 **-t**、 **-T**、 **-c**、 **-C** のいずれのフラグも指定しない場合は、これがデフォルトです。追加されているファイルの場合は、**-F** フラグは無視されます。 CICS または VSE のホストでは、変換フラグ (**-t** または **-T**) あるいは CRLF フラグ (**-c** または **-C**) を **-F** フラグと一緒に指定する必要があります。これは、CICS および VSE のホスト・ファイル転送プログラムでは、固定レコード長をサポートしていないためです。**-F** フラグを変換フラグと組み合わせると、転送プログラムはレコードを、その論理レコード長の最後までブランクで埋めます。デフォルトは 80 です。 **注: -F** フラグは、VSE ホストからダウンロードする場合、変換ファイルの後続ブランクが削除されるのを防ぎます。
-L *LoglRecLength*	ホスト・ファイルの論理レコード長をバイト数で指定します。新規のファイルに対するデフォルトは 80 バイトです。可変長レコードに対しては、*LoglRecLength* はレコードの最大サイズになります。追加されているファイルの場合は、**-L** フラグは無視されます。*LoglRecLength* 値を 0 にすると、エラーになります。 MVS™ のオーバーヘッドにより、MVS/TSO ホスト上の可変長レコード内に格納される実バイト数は、*LoglRecLength* 変数で指定した値よりも 4 バイト少なくなります。 CICS および VSE ホストのファイル転送プログラムは、論理レコード長をサポートしていません。 CICS ホストまたは VSE ホストとの間での転送では、**-L** フラグを **-F** フラグと一緒に指定する必要があります。**-F** フラグと **-L** フラグを組み合わせると、転送プログラムはレコードを論理レコード長の最後までブランクで埋めます。デフォルトは 80 です。 **注:** レコード長がデフォルトである 80 を超える場合、**-L** フラグを指定する必要があります。

項目	説明
-S *NumberUnits* [*,IncreaseUnits* | *,IncreaseUnits,UnitType* | *,,UnitType*]	TSO 上の新規の順次データ・セットに対して割り当てるスペースの量を指定します。大きな MVS ファイルに対しては、全ディスク・トラックを必ずいっぱいにするために、ホスト上の最大許容ブロック・サイズが使用されます。 **-S** フラグは MVS/TSO ホストの場合にのみ使用できます。 次の変数は **-S** フラグと一緒に使用できます。使用する場合、変数は定められた順序で指定し、コンマによって区切ります。ある変数の前の変数が省略される場合、1 つのコンマがプレースホルダーとして必要です。**-S** フラグと *NumberUnits* 変数の間には、スペースが 1 つ必要です。しかし、変数の文字列にはスペースを入れてはいけません。 *NumberUnits* 最初に追加されるスペースの単位数を指定します。*NumberUnits* 変数には 0 (ゼロ) およびマイナス値は指定できません。 *IncreaseUnits* 先に割り当てられたスペースがいっぱいになるたびに、データ・セットに追加するスペースの単位数を指定します (オプション)。 *UnitType* スペースの単位 (例えば、トラックは `T`、シリンダーは `C`)、またはデータ・セットに書き込まれたレコードの平均ブロック・サイズ (バイト単位) を指定する数を定義します。 *UnitType* 変数を指定しない場合、 **-B** フラグで指定した値が、デフォルトとして使用されます。**-B** *BlockSize* フラグの指定がない場合のデフォルト値は 80 です。 次に示すのは、**-S** フラグと一緒に使用される変数の有効な組み合わせです。 **-S** *NumberUnits*,*IncreaseUnits*,*UnitType* **-S** *NumberUnits*,*IncreaseUnits* **-S** *NumberUnits* **-S** *NumberUnits*,,*UnitType*
-U	定義されていないレコード長を指定します。**-U** フラグは MVS/TSO 環境でのみ使用できます。**-U** フラグは、ファイルが追加されている間は無視されます。
-V	可変長レコードを指定します。 **-F** フラグを指定せずに、 **-t**、 **-T**、 **-c**、 **-C** のいずれかのフラグを指定した場合は、これがデフォルトです。追加されているファイルの場合は、**-V** フラグは無視されます。 **-V** フラグは、CICS または VSE ホストでは可変レコード長がデフォルトとなるため、ファイル転送プログラムではサポートされません。

例

次の例では、セッション `a` に対するセッション・プロファイルを想定しています。

```
Session   type         DFT
Communication   device         3270c0
Language         English   (U.S.A.)
Host   type         CMS
File   transfer   direction         up
File   transfer   wait   period         10
File   transfer   recovery   time         30
```

それぞれの意味は次のとおりです。

- ホスト・タイプは VM/CMS です。
- 接続には DFT 3270 接続デバイスが使用されています。
- デフォルトのファイル転送方向は、アップロードです (セッション・プロファイル `a` を使用して、ファイルをダウンロードする場合は、**fxfer** コマンドで、**-d** フラグを指定する必要があります)。
- ファイル転送プロセスは、10 分間ログイン状態にあります。

- 転送が中断されると、このプロセスでは、 30 分間リカバリーを試みた後、以後の転送のために RESTART ファイルに情報を保管します。
- 変換言語は、U.S.A. ASCII-EBCDIC です。

1. samplefile ファイル (現行ディレクトリーにある) を、ホストにアップロードし、U.S.A. 変換テーブルを使って EBCDIC に変換するには、次のように入力します。

   ```
   fxfer -n a -t samplefile "test file a"
   ```

   – **-n** は、**fxfer** コマンドに、セッション a を使用してファイルを転送するよう指示します。
   – **-t** は、コマンドに改行文字を使用して変換を行うよう指示します。

   変換されたデータは、ホスト上の test file a に入れられます。ホスト・ファイル名にはスペースが含まれているので、ファイル名を囲む引用符が必要です。

2. file2 ファイルを、 VM/CMS ホストの test file b にアップロードするには、次のように入力します。

   ```
   fxfer -urv -L 132 -V -H CMS file2 "test file b"
   ```

   – **-u** は、 **fxfer** コマンドにファイルをアップロードするよう指示します。
   – **-H** は、ホスト・タイプが VM/CMS ホストであることを指示します。宛先ファイルが存在する場合は、そのファイルが転送されたファイルで置換されます (**-r** フラグが指定されているため)。
   – **-v** を指定すると、**fxfer** が転送されたデータのバイト数と経過時間を表示します。状況または診断出力が端末に表示されます。
   – ホスト・ファイルが存在しない場合は、ホスト・ファイルの最大論理レコード長が、 132 バイトに設定されます (**-L** フラグ)。
   – ホスト・ファイルのレコード形式は、可変です (**-V** フラグ)。変換は行われません。

3. ローカル・システムの **/etc/motd** ファイルを、エミュレーター・セッション a のサブシェルから、CICS の motdfile ホスト・ファイルに、変換とブランクの埋め込み処理をしてアップロードするには、次のように入力します。

   ```
   fxfer -utFH CICS -I ")This is a comment" /etc/motd "motdfile"
   ```

   – **-u** は、このコマンドにファイルをアップロードするよう指示します。
   – **-t** を指定すると、 ASCII から EBCDIC への変換が行われます。
   – **-F** を指定すると、転送プログラムによって、アップロードされたファイルが、 80 桁 (デフォルトのレコード長) までブランクで埋め込まれます。デフォルトの桁数を変更する場合は、 **-L** フラグに、異なるレコード長 (桁数) を付けて指定してください。
   – **-H** は、ホストをタイプ CICS として指定します。
   – **-I** は、*InputField* 値を **IND$FILE** コマンドに追加するよう指定します。

   この例で、「This is a comment」はホストのコメント・フィールドです。

   fxfer コマンドで、現在の環境とは別の TSO 環境との間でファイルのアップロードまたはダウンロードを行うには、他の環境に対する適切な権限が必要です。転送するファイル (またはデータ・セット) を、まず単一引用符 (') で、次に二重引用符 (" ") で完全に修飾しなければなりません。

4. 例えば、newfile というファイルを、修飾ファイル名 sys4.parmlib.samplefile を持つ TSO 環境にアップロードするには、次のように入力します。

   ```
   fxfer -urtvH TSO 'newfile' "sys4.parmlib.samplefile"
   ```

   – **-u** は、このコマンドにファイルをアップロードするよう指示します。

– sys4.parmlib.samplefile ファイルが存在する場合は、このファイルが newfile ファイル (**-t** フラグ) の、変換済みの内容で置換されます (**-r** フラグ)。
– **-v** は、**fxfer** コマンドに、ファイル転送状況を数秒ごとにローカル・マシンの画面に表示するよう指示します。
– **-H** は、ホストが MVS/TSO ホストであることを **fxfer** コマンドに指示します。

注**:** この例では、**fxfer** コマンドが確立されたセッション (セッションを確立するには、 **e789** コマンドを使用) のサブシェルから実行されることを前提としています。

5. ファイル spfuser.test を、 MVS/TSO ホストからローカル・システムへダウンロードするには、次のように入力します。

```
fxfer -n a -d -r -H TSO spfuser.test samplefile1
```

– **-n** は、**fxfer** コマンドに、セッション a を使用してファイルを転送するよう指示します。セッション a が確立されていない場合は、コマンドは自動ログインを行おうとします。ホスト・ログイン ID が指定されていないので、**fxfer** コマンドは、ログイン ID に対するセッション・プロファイルを検査します。セッション・プロファイルに指定がなければ、ユーザーにログイン ID とパスワードを入力するように要求します。
– **-d** は、デフォルトのファイル転送方向であるアップロードを指定変更します。
– samplefile1 ファイルが既に存在する場合は、このファイルがホストからダウンロードされたファイルで置換されます (**-r** フラグ)。
– **-H** は、ホストが VM/CMS ホスト (セッション・プロファイルのデフォルト) ではなく、MVS/TSO ホストであることを **fxfer** コマンドに指示します。

転送ファイルはローカル・システム上の samplefile1 に入れられます。このファイル転送は同期的に実行されます。

6. セッション・プロファイル a および自動ログインを用いて、 VM/CMS ホスト test file a をダウンロードし、それをローカル・システム mydir/samplefile ファイルに追加するには、次のように入力します。

```
fxfer -n a -dat -q -f status.out
-x laura,vm1,trace "test file a" mydir/samplefile
```

– **-n** は、**fxfer** コマンドに、セッション・プロファイル a を使用してファイルを転送するよう指示します。
– **-x** は、ホスト・ログイン ID を指定します。**fxfer** コマンドは、まず、セッションがローカル・システムで確立されているかどうかを確認します。確立されていれば、そのセッション上でファイルを転送します。セッション a が確立されていない場合は、**fxfer** コマンドがホスト・ログオン ID laura と AUTOLOG スクリプト vm1 を使用して自動ログインを実行し、ログイン・アクティビティーをトレースします。ユーザーにはパスワードの入力を要求します。コマンドはファイルを転送します。
– **-dat** は、**fxfer** コマンドに、ファイルをダウンロード (**-d** フラグ) して、 U.S.A 変換テーブル (セッション・プロファイルで定義) によって、データを EBCDIC から ASCII へ変換 (**-t** フラグ) し、変換されたファイルを、ローカル・システム上の mydir/samplefile ファイルに付加 (**-a** フラグ) します。mydir/samplefile ファイルが存在しなければ、**fxfer** コマンドは **-a** フラグを無視してファイルを作成します。
– 状況または診断の出力は、現行ローカル・ディレクトリーの、 status.out ファイルに格納されます (**-f** フラグ)。
– **-q** は、**fxfer** コマンドにファイルを非同期的に転送するよう指示します。

ユーザーがパスワードを入力すると、プロンプトに戻り、ファイル転送はバックグラウンドで実行されます。

別のファイル転送を同じファイル転送プロセスで実行するためにキューに入れるには、次のように入力します。

```
fxfer  -n  a  -daq  -f  status.out  "test  file  b"
mydir/samplefile
```

– **-n** は、**fxfer** コマンドに、セッション a を使用してファイルを転送するよう指示します。セッション a は、以前のコマンドで既に確立されているので、**fxfer** コマンドは、あらためてホストにログインする必要はありません。
– **-d** は、コマンドにホストからファイルをダウンロードするよう指示します。
– **-a** は、コマンドに、test file b ホスト・ファイルを、ローカル・システム 上の mydir/samplefile ファイルに追加するように指示します。
– **-q** は、**fxfer** コマンドにファイルを非同期的に転送するよう指示します。

fxfer コマンドは、状況情報を引き続きローカル・システム上の status.out ファイルに送信するよう指示します (**-f** フラグ)。

注**:**

a. **fxfer** コマンドに対して、テキストを画面の制限を超えて入力する場合、テキストは自動的に折り返されます。Enter キーを押してテキストを改行するとエラーになります。
b. キューに非同期転送が存在する場合に、同期転送を行おうとするとエラーになります。
c. セッションの実行中にホストにログインされた、 **dfxfer** プロセスが継続中である場合は、ユーザーはログイン ID およびパスワードの入力は要求されません。プロセスがログインしている時間の長さは、セッション・プロファイルの File Transfer Wait Period で決定されます。

7. 中断されたファイル転送を、エミュレーターのサブシェルから再開するには、次のように入力します。

```
fxfer  -R
```

-R は、**fxfer** コマンドに、RESTART ファイルのうちの 1 つに保存された情報でファイル転送を実行するよう指示します。RESTART ファイルは、 **$HOME/x_fxfer.r** 明示再始動ファイルであるか、または **$HOME/i_fxfer.r** 暗黙再始動ファイルのどちらかです。**-R** フラグが他のファイル転送フラグと一緒に指定されている場合、他のフラグは無視されます。RESTART ファイル転送メニューが表示されます。メニューを使用して中断されたファイルを転送するよう **fxfer** コマンドに命令します。

8. エミュレーターのサブシェルからではなく、コマンド・ラインからファイル転送を再開させるには、次のように入力します。

```
fxfer  -R  -n  a
```

-n フラグは、**fxfer** コマンドにセッション a を使用して再開した転送を実行するように指示します。

ファイル

項目	説明
/usr/bin/fxfer	**fxfer** コマンドが入っています。
/usr/bin/dfxfer	**dfxfer** プロセスが入っています。
$HOME/i_fxfer.r	自動ログイン・キューに関する RESTART 情報が入っています。**fxfer** コマンドにより作成される一時ファイルです。
$HOME/x_fxfer.r	手動ログイン・キューに関する RESTART 情報が入っています。**fxfer** コマンドにより作成される一時ファイルです。
$HOME/hconerrors	HCON 診断出力とファイル転送状況が入っています。HCON コマンドにより作成される一時ファイルです。
/usr/lib/libfxfer.a	プログラム化されたファイル転送処理のライブラリーが入っています。

関連情報:

smit コマンド

g

以下の AIX コマンドは文字 g で始まります。

gated デーモン

目的

RIP、RIPng、EGP、BGP、BGP4+、HELLO、IS-IS、ICMP、ICMPv6、および SNMP プロトコル用のゲートウェイ経路指定機能を提供します。

注**:** コマンド・ラインから **gated**デーモンを制御するには、SRC コマンドを使用してください。システム始動時に毎回デーモンを始動するには、**rc.tcpip** ファイルを使用します。

構文

/usr/sbin/gated [**-c**] [**-C**] [**-n**] [**-N**] [**-t** *TraceOptions*] [**-f** *ConfigFile*] [*TraceFile*]

説明

/usr/sbin/gated デーモンは、複数経路指定プロトコルを処理し、**routed** と、(HELLO) 経路指定プロトコルを使用する経路指定デーモンとを置換します。 **/usr/sbin/gated** デーモンが現在扱うのは、経路指定情報プロトコル (RIP)、経路指定情報プロトコル次世代 (RIPng)、Exterior Gateway Protocol (EGP)、 Border Gateway Protocol (BGP) と BGP4+、 Defense Communications Network Local-Network Protocol (HELLO)、および Open Shortest Path First (OSPF)、Intermediate System to Intermediate System (IS-IS)、および Internet Control Message Protocol (ICMP) / Router Discovery 経路指定プロトコルです。さらに、**gated** デーモンは、シンプル・ネットワーク管理プロトコル (SNMP) もサポートします。 **gated** プロセスは、これらのプロトコルのすべてを、もしくは任意の組み合わせを実行するように構成することができます。 **gated** デーモンのデフォルトの構成ファイルは、**/etc/gated.conf** ファイルです。**gated** デーモンは、デーモンのプロセス ID を **/etc/gated.pid** ファイルに格納します。

注**: gated** デーモンと、**routed** デーモンを、 1 つのホスト上で一緒に実行すると、予測できない結果になる場合があります。

コマンド・ラインに、トレース・ファイルを指定する、またはトレース・フラグを指定しないと、**gated** デーモンは、端末装置から切り離され、バックグラウンドで実行されます。トレース・ファイルを指定せずにトレース・フラグを指定すると、**gated** では、トレースが **stderr** に指定されていると想定し、フォアグラウンドでの実行を続けます。

注**:** IS-IS 経路指定プロトコルは、64 ビット・カーネル上では実行できません。

シグナル

gated サーバーは、**kill** コマンドによるシグナルの送信時に、次のアクションを実行します。

項目	説明
SIGHUP	設定が再読み取りされます。
	SIGHUP を使用すると、gated によって構成ファイルが再読み取りされます。 **gated** デーモンではまず、割り当てられたポリシー構造をすべて整理します。すべての BGP ピアと EGP ピアに消去を示すフラグが付けられ、構成ファイルが再解析されます。
	ファイルの再解析が成功すると、設定に入っていない、BGP ピアまたは EGP ピアはすべてシャットダウンされ、新しいピアが始動されます。 **gated** デーモンは既存のピアを変更するためにシャットダウンが必要かどうかを判別し、再始動しようとします。 注: OSPF (Open Shortest Path First) が使用可能な場合は、再構成は使用不可の状態です。
SIGINT	現在の状態のスナップショットが表示されます。
	ゲート指定されたすべてのタスク、タイマー、プロトコル、テーブルの現在の状態が、 **/var/tmp/gated_dump** に書き込まれます。
	これは、テーブル情報をダンプするサブプロセスを fork するという方法で実行されます。 **gated** デーモンの経路指定機能に影響を与えないようになっています。
SIGTERM	通常のシャットダウンが行われます。
	SIGTERM シグナルを受信した時点で、**gated** デーモンでは、通常のシャットダウンを実行しようとします。すべてのタスクとプロトコルがシャットダウンを要求します。このうち大部分はすぐに終了しますが、確認を待機している EGP ピアだけは例外となります。このプロセスに時間がかかりすぎるときは、**SIGTERM** を 1 から 2 回繰り返さなければならない場合があります。
	すべてのプロトコル経路は、**SIGTERM** が受信された時点で、カーネルの経路指定テーブルから除去されます。インターフェース経路、**RTF_STATIC** が設定されている経路 (サポートされている場合は **route** コマンドから設定)、**retain** を指定する静的経路は、すべてそのまま残されます。外部経路がそのまま残されている **gated** デーモンを終了するには、**SIGKILL** シグナルまたは **SIGQUIT** シグナル (メモリー・ダンプを作成する) を使用します。
SIGUSR1	トレースが切り替えられます。
	SIGUSR1 シグナルを受信した時点で、**gated** デーモンでは、トレース・ファイルをクローズします。次に **SIGUSR1** を受け取ると、このデーモンが再度オープンされます。これにより、ファイルを定期的に移動することができます。 注: トレース・ファイルを指定していない場合、または **stderr** にトレースされる場合、**SIGUSR1** シグナルは、使用できません。
SIGUSR2	インターフェースの変更を検査します。
	SIGUSR2 シグナルを受信すると、**gated** デーモンはカーネル・インターフェース・リストをスキャンし直して、変更の有無を調べます。

gated デーモンと **snmpd** デーモン

gated デーモンは、**snmpd** デーモンの SNMP 多重 (SMUX) プロトコル・ピア、すなわち代替エージェントとなるように内部で設定されます。詳しくは、「ネットワークおよびコミュニケーションの管理」の『SNMP デーモンの処理』のセクションを参照してください。

システム・リソース・コントローラーによる **gated** デーモンの操作

gated デーモンは、システム・リソース・コントローラー (SRC) で制御することができます。また、**gated** デーモンは SRC の **tcpip** システム・グループのメンバーです。このデーモンはデフォルトでは使用不可で、以下の SRC コマンドにより操作できます。

項目	説明
startsrc	サブシステム、サブシステム・グループ、またはサブサーバーを始動します。
stopsrc	サブシステム、サブシステムのグループ、またはサブサーバーを停止します。
refresh	サブシステムまたはサブシステム・グループに該当する構成ファイルを再読み取りします。
lssrc	サブシステム、サブシステムのグループ、あるいはサブサーバーの状況を取得します。

注**: startsrc** コマンドからの初期始動時には、**gated** デーモンは、すべての **gated** の初期化が完了するまで他の SRC コマンドに対する応答を開始しません。極端に大きい **/etc/gated.conf** ファイルの場合は、完全に解析するまでに 1 分以上かかることがあります。

フラグ

項目	説明
-c	構成ファイルに **gated** デーモンを終了させてしまう構文エラーがないかどうか解析するように指定します。エラーが発生しなければ、**gated** デーモンはダンプ・ファイルを **/var/tmp/gated_dump** ファイルに入れます。 **-c** フラグは、**-tgeneral,kernel,nostamp** フラグを暗黙指定します。**-c** フラグを指定すると、**gated** デーモンは構成ファイル内のすべての **traceoption** 節と **tracefile** 節を無視します。
-C	構成ファイルの構文エラーのみを解析するように指定します。 **gated** デーモンの状況は、何らかのエラーを検出した場合、**1** になり、エラーがない場合は、**0** の状況になります。 **-C** フラグは、**-tnostamp** フラグを暗黙指定します。
-f *ConfigFile*	代替構成ファイルを指定します。デフォルトでは、**gated** デーモンは **/etc/gated.conf** ファイルを使用します。
-n	**gated** デーモンによってカーネルの経路指定テーブルが変更されないように指定します。これは、実際の経路指定データを使って **gated** の設定を検査する場合に使用します。
-N	**gated** デーモンがデーモン化しないように指定します。一般に、**stderr** へのトレースを指定せず、親プロセス ID が 1 でなければ、**gated** デーモンはデーモン化します。このフラグを指定すると、プロセス ID が 1 でない **gated** デーモンをコールする **/etc/inittab** のようなメソッドを使用することができます。
-t*TraceOptions*	システム始動時に使用可能にするトレース・オプションを指定します。 *TraceOptions* 変数を付けずに使用すると、このフラグは **general** トレース・オプションを始動します。各トレース・オプションをコンマで区切ってください。フラグと最初のトレース・オプションの間には、スペースを挿入しないでください。 **-t** フラグは、インターフェース設定の判別およびカーネルからの経路指定の読み取りなどの、**/etc/gated.conf** ファイルが解析される前に起こるイベントをトレースするために使用しなければなりません。 **gated.conf** ファイルの項目に、使用可能なトレース・オプションを記述しています。

例

1. **gated** デーモンを始動するには、次のようなコマンドを入力します。

   ```
   startsrc -s gated -a "-tall /var/tmp/gated.log"
   ```

   このコマンドにより、**gated** デーモンが始動され、メッセージがログに記録されます。メッセージは、**/var/tmp/gated.log** ファイルに送られます。
2. **gated** デーモンを通常どおりに停止するには、次のように入力します。

   ```
   stopsrc -s gated
   ```

   このコマンドによって、このデーモンが停止されます。 **-s** フラグは、後に続くサブシステムを停止することを指定します。
3. **gated** デーモンから簡潔な状況情報を得るには、次のように入力します。

   ```
   lssrc -s gated
   ```

このコマンドは、デーモン名、デーモンのプロセス ID、デーモンの状態 (アクティブまたは非アクティブ) を戻します。

ファイル

項目	説明
/etc/gated.pid	**gated** のプロセス ID が入っています。
/var/tmp/gated_dump	メモリー・ダンプ・ファイルを指定します。
/var/tmp/gated.log	エラー・メッセージのログ・ファイルを指定します。

関連資料:

『gdc コマンド』

関連情報:

kill コマンド

gated.conf コマンド

gated デーモンの構成

---

gdc コマンド

目的

gated 用の操作可能ユーザー・インターフェースを提供します。

構文

gdc [**-q**] [**-n**] [**-c** *coresize*] [**-f** *filesize*] [**-m** *datasize*] [**-s** *stacksize*] [**-t** *seconds*] *Subcommands*

説明

gdc コマンドは、**gated** 経路指定デーモン操作のためのユーザー指向インターフェースを提供します。このコマンドは、以下のサポートを行います。

- デーモンの始動と停止
- デーモンの稼働中にそれを操作するシグナルの送達
- 構成ファイルの保守と構文検査
- 状態ダンプおよびメモリー・ダンプの実動と取り外し用

gdc コマンドは、**gated** の稼働状況を確実に判断して、エラー発生時に確かな終了状況を作成し、そして **gated** を扱うシェル・スクリプトで使用するためそれを使いやすいものにします。 **gdc** を用いて実行するコマンドや、それらの実行で作成されるエラー・メッセージ (オプション) は、 **gated** 自身が使用する同じ **syslogd** 機能によって記録され、そのデーモン上で行われた操作の監査証跡を提供します。

フラグ

項目	説明
-n	カーネル転送テーブルを変更せずに実行されます。これは、テストの場合および、転送を行わない経路指定サーバーとして動作するときに便利です。
-q	メッセージは出されません。通常、標準出力に出される情報メッセージは、このフラグで抑止され、エラー・メッセージは、標準エラー出力に出されるのではなく **syslogd** に記録されます。これは、シェル・スクリプトから **gdc** を実行中のときは便利です。
-t*seconds*	**gated** がある種の操作を (特に、終了時と始動時に) 完了するのを **gdc** が待つ時間を秒単位で指定します。デフォルトでは、この値は 10 秒に設定されています。
-c *coresize*	**gdc** で始動された **gated** が作成するメモリー・ダンプの最大サイズを設定します。これは、デフォルトの最大メモリー・ダンプ・サイズが小さ過ぎて **gated** がエラーに関してメモリー・ダンプを完全に作成できないシステムでは便利です。
-f *filesize*	**gdc** で始動された **gated** が作成する最大ファイル・サイズを設定します。これは、デフォルトの最大ファイル・ダンプ・サイズが小さ過ぎて、**gated** が要求時に全状態ダンプを作成できないシステムでは便利です。
-m *datasize*	**gdc** で始動された **gated** の最大データ・セグメント・サイズを設定します。これは、デフォルトのデータ・セグメント・サイズが小さ過ぎて、**gated** が実行できないシステムでは便利です。
-s *stacksize*	**gdc** で始動された **gated** の最大スタック・サイズを設定します。これは、デフォルトの最大スタック・サイズが小さ過ぎて、**gated** が実行できないシステムでは便利です。

サブコマンド

以下のサブコマンドを用いると、シグナルはさまざまな目的で **gated** に送達されます。

項目	説明
COREDUMP	**gated** に異常終了シグナルを送り、メモリー・ダンプを作成して終了させます。
dump	**gated** に、現行の状態をファイル **/var/tmp/gated_dump** にダンプするようシグナルを送ります。
interface	**gated** に、インターフェース構成を再検査するようシグナルを送ります。どのような場合でも、**gated** は通常これを定期的に行いますが、変更の発生がわかっているときは、この機能を使用してデーモンにインターフェース状況を即時に検査させることができます。
KILL	**gated** を異常状態で終了させます。
reconfig	**gated** に、その構成ファイルを再読み取りして現在の状態を適切に再構成するようシグナルを送ります。
term	**gated** に、すべての操作経路指定プロトコルを規定どおりに停止させた後、終了するようシグナルを送ります。このコマンドを 2 度目に実行した場合は、完全に停止していないプロトコルがあっても **gated** は終了します。
toggletrace	トレースを中断し、**gated** が現在ファイルをトレースしている場合は、そのトレース・ファイルをクローズします。 **gated** のトレースが現在中断中の場合は、このサブコマンドで、トレース・ファイルは再オープンし、トレースは開始されます。これは、トレース・ファイルを移動する場合に便利です。

以下のサブコマンドは、構成ファイルに関連した操作を行います。

項目	説明
checkconf	**/etc/gated.conf** を検査して、構文エラーを調べます。この検査は、構成ファイルの変更後でかつ現在実行中の **gated** に **reconfig** シグナルを出す前に行って、実行中の **gated** が再構成で終了する原因となるエラーが構成内にないことを確認するのに便利です。このコマンドを使用すると、**gdc** は、構文解析エラーの有無を示し、あれば検査のためにエラー出力をファイルに保存することを示す情報メッセージを出します。
checknew	新規 構成ファイル **/etc/gated.conf+** が検査されることを除けば **checkconf** と同じです。
newconf	**/etc/gated.conf+** ファイルを **/etc/gated.conf** として所定の場所に移動し、上記の旧バージョンのファイルを保存します。 **gdc** は、新規構成ファイルが存在しないか、存在しても疑わしい場合は、このコマンドが与えられていると場合何の処置もしません。
backout	構成ファイルをより新しい方向に回転し、実質的には古い構成ファイルを **/etc/gated.conf** に移動します。 **/etc/gated.conf-** が存在しないか、長さがゼロの場合、あるいはこの操作で既存の、ゼロ以外の長さの **/etc/gated.conf+** ファイルを削除する場合は、このコマンドは動作しません。
BACKOUT	**/etc/gated.conf+** が存在していて、長さがゼロ以外であっても、バックアウト操作を行います。
modeconf	すべての構成ファイルを、モード 664、オーナー・ルート、グループ・システムに設定します。

項目	説明
createconf	**/etc/gated.conf+** が存在しない場合は、ファイル・モードを 664、オーナー・ルート、グループ・システムに設定して、長さがゼロのファイルを作成します。

以下のサブコマンドは、**gated** の始動と停止、およびその実行状態の判別についてサポートします。

項目	説明
実行	**gated** が現在実行中かどうかを判別します。これは、**gated** がその PID が入っているファイルにロックをかけているかどうか、ファイル内の PID が目的にかなったものかどうか、そしてその PID に実行中のプロセスがあるかどうかをチェックして行われます。 **gated** が実行中ならゼロ状況で、そうでない場合はゼロ以外で、終了します。
start	**gated** を始動します。このコマンドは、**gated** が既に実行中ならばエラーを戻します。実行中でなければ、**gated** バイナリーを実行し、新たに始動されたプロセスが PID ファイルへのロックを入手するまで遅延間隔いっぱい (デフォルトでは 10 秒、そうでない場合は **-t** オプションの設定に従って) 待ちます。バイナリーの実行中にエラーが検出された場合や、あるいは指定の待ち時間内で PID ファイルのロックが入手されなかった場合は、ゼロ以外の終了状況が戻されます。
stop	可能であれば正常状態で、可能でなければ異常状態で **gated** を停止します。 **gated** が現在実行中でない場合、コマンドはエラー (ゼロ以外の終了状況で) を戻します。実行中の場合は、終了シグナルを **gated** に送り、プロセスが終了するまでの遅延間隔いっぱい (デフォルトでは 10 秒、そうでない場合は **-t** オプションの設定に従って) 待ちます。 **gated** が遅延間隔以内に終了しなかった場合は、それに 2 番目の終了シグナルで再度シグナルが送られます。 2 番目の遅延間隔の終わりまでに終了しなかった場合は、kill シグナルで 3 回目のシグナルが送られます。これによって、余程何かが壊れていない限り、即時終了が強制されます。このコマンドは、**gated** が終了したことを検出した場合はゼロ終了状況で、そうでない場合はゼロ以外の終了状況で終了します。
restart	**gated** が実行中である場合は、上記の **stop** コマンドで使用した手順と同じ手順で終了します。前の **gated** が、終了したとき、あるいはコマンドの実行前に実行中でなかった場合は、新しい **gated** プロセスが、上記の **start** コマンドで説明した手順を用いて実行されます。この手順のステップが失敗したようであれば、ゼロ以外の終了状況が戻されます。

以下のサブコマンドを用いると、上記のコマンドのどれかを実行して作成されたファイルを除去することができます。

項目	説明
rmcore	既存の **gated** メモリー・ダンプ・ファイルを除去します。
rmdump	既存の **gated** 状態ダンプ・ファイルを除去します。
rmparse	**checkconf** または **checknew** コマンドが実行され、構文エラーが検査中の構成ファイルで検出されたときに生成された、構文解析エラーを除去します。

デフォルトでは、**gated** は、通常 **/etc/gated.conf** というファイルからその構成を入手します。 **gdc** プログラムは、他の複数のバージョンの構成ファイル、特に以下の名前の構成ファイルも維持しています。

項目	説明
/etc/gated.conf+	新規構成ファイル。 **gdc** が、新規構成ファイルのインストールを要求されると、このファイルの名前は **/etc/gated.conf** に名前変更されます。
/etc/gated.conf-	旧構成ファイル。 **gdc** が、新規構成ファイルのインストールを要求されると、前の **/etc/gated.conf** の名前はこの名前に変更されます。
/etc/gated.conf—	実際に古い構成ファイル。 **gdc** は、前の古い構成ファイルを、この名前で保存します。

ファイル

項目	説明
/usr/sbin/gated	**gated** バイナリー。
/etc/gated.conf	現行 **gated** 構成ファイル。
/etc/gated.conf+	より新しい構成ファイル。
/etc/gated.conf-	より古い構成ファイル。
/etc/gated.conf—	さらに古い構成ファイル。
/etc/gated.pid	**gated** がその PID を格納する場所。
/var/tmp/gated_dump	**gated** の状態ダンプ・ファイル
/var/tmp/gated.log	構成ファイル構文解析エラーが入るファイル。

関連資料:

669 ページの『gated デーモン』

関連情報:

syslogd コマンド

---

gencat コマンド

目的

メッセージ・カタログを作成して、変更します。

構文

gencat *CatalogFile SourceFile ...*

説明

gencat コマンドは、メッセージ・テキスト・ソース・ファイル (通常は ***.msg**) からメッセージ・カタログ・ファイル (通常は ***.cat**) を作成します。**gencat** コマンドは、*SourceFile* パラメーターで指定したメッセージ・テキスト・ソース・ファイルを、*CatalogFile* パラメーターで指定したフォーマット済みのメッセージ・カタログにマージします。メッセージをソース・ファイルに入力した後で、**gencat** コマンドを使ってソース・ファイルを処理し、メッセージ・カタログを作成してください。**gencat** コマンドは、カタログ・ファイルが存在しなければ、カタログ・ファイルを作成します。カタログ・ファイルが存在する場合は、**gencat** コマンドは新しいメッセージをそのカタログ・ファイルに追加します。

メッセージ・テキスト・ソース・ファイルはいくつでも指定できます。**gencat** コマンドは、指定した順序で複数のソース・ファイルを 1 つずつ処理します。連続する各ソース・ファイルによってカタログが変更されます。セット番号とメッセージ番号が矛盾すると、*SourceFile* パラメーターに定義した新しいメッセージ・テキストによって、*CatalogFile* パラメーターに現在格納されている旧メッセージ・テキストが置き換えられます。メッセージ番号の範囲は 1 から **NL_MSGMAX** です。セット番号の範囲は 1 から **NL_SETMAX** です。

gencat コマンドでは、記号メッセージ ID は受け入れられません。記号メッセージ ID を使用する場合は、**mkcatdefs** コマンドを実行する必要があります。

注**:** *CatalogFile* パラメーターに、 **-** (ダッシュ) 文字を指定すると、標準出力が使用されます。*SourceFile* パラメーターに **-** (ダッシュ) 文字を指定すると、標準入力が使用されます。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

ソース・ファイル `test.msg` から `test.cat` カタログを生成するには、次のように入力します。

```
gencat test.cat test.msg
```

`test.msg` ファイルには記号 ID は含まれていません。

ファイル

項目	説明
/usr/bin/gencat	**gencat** コマンドが入っています。

関連資料:

232 ページの『dspcat コマンド』

関連情報:

runcat コマンド

catopen コマンド

メッセージ機能の概要

---

gencopy コマンド

目的

さまざまなパッケージ・フォーマット (installp、RPM、ISMP) のソフトウェア・プロダクトをコピーできるようにします。

構文

ソフトウェアをメディアからターゲット・ロケーションにコピーするには

gencopy -d *Media* [**-t** *TargetLocation*] [**-D**] [**-b** *bffcreateFlags*] [**-U**] [**-X**] **-f** *File* | *CopyList...* | all

ソフトウェア・プロダクトおよびパッケージをメディア上に表示するには

gencopy -L -d *Media* [**-D**]

説明

gencopy コマンドは、 **bffcreate** コマンドに対するラッパーです。このコマンドは、コピーする必要があるイメージを判別し、適切なコマンドを呼び出します。RPM、ISMP、または必要なファイルのリストが不明なイメージのその他のタイプでは、サブディレクトリー内の全ファイルがターゲット・ロケーションにコピーされます。

フラグ

項目	説明
-b *bffcreateFlags*	有効な **l**、**q**、**v**、 **w**、**S** フラグを指定します。
-d *Media*	インストール・イメージが存在するデバイスまたはディレクトリーを指定します。メディアはデバイス (**/dev/cd0**、**/dev/rmt0**) でも、ディレクトリーでもかまいません。
-D	デバッグ・モードを指定します。このフラグは、このスクリプトをデバッグするためのものです。これにより大量の出力が作成されるため、通常の操作にこれを使用すべきではありません。
-f *File*	ターゲット・ロケーションにコピーするイメージのリストを入れるファイルを指定します。 **installp**、RPM、および ISMP のイメージには、プレフィックスをそれぞれ **I:**、**R:**、および **J:** と付ける必要があります。暫定修正パッケージにはプレフィックス **E:** を付けてください。
-L	メディア上にインストール・パッケージをリストします。このリストはコロンで区切られ、次のような情報を含んでいます。 file_name:package_name:fileset:V.R.M.F:type:platform:Description bos.sysmgt:bos.sysmgt:bos.sysmgt.nim.client:4.3.4.0:I:R:Network Install Manager - Client Tools bos.sysmgt:bos.sysmgt:bos.sysmgt.smit:4.3.4.0:I:R:System Management Interface Tool (SMIT)
-t *TargetLocation*	インストール・イメージ・ファイルが保管されるディレクトリーを指定します。**-t** フラグを指定しない場合、ファイルは **/usr/sys/inst.images** ディレクトリーに保管されます。
-U	必要に応じて、宛先リポジトリーのディレクトリー構造を現行標準にアップグレードします。現行標準では、イメージがパッケージ・タイプおよびアーキテクチャーに応じて、サブディレクトリーに編成することが必要です。例えば、installp イメージは、***SaveDir*/installp/ppc** ディレクトリーにあります。この構造が入っているソースからコピーするときは、コピー先が適合する必要があります。 **-U** フラグを指定すると、**gencopy** コマンドは、適切なサブディレクトリー構造をユーザーのリポジトリーに作成し、既存のイメージがあればそれらを適切なロケーションに移動します。このフラグは、その後に無効な手動コピーが行われない限り、一度だけ使用すれば済みます。
-X	スペースが必要な場合に、自動的にファイルシステムを拡張します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

イメージをすべて、CD (**/dev/cd0**) から、 **LPP_SOURCE** (**/export/lpp_source/500**) にコピーするには、次のように入力します。

```
gencopy -d  /dev/cd0 -t  /export/lpp_source/500 all
```

ファイル

項目	説明
/usr/sbin/gencopy	
/usr/sys/inst.data/sys_bundles	
/usr/sys/inst.data/user_bundles	

関連情報:

bffcreate コマンド

gencore コマンド

目的

実行中のプロセスのコア・ファイルを生成します。

構文

gencore *ProcessID FileName*

説明

gencore コマンドは、プロセス ID *ProcessID* で指定されたプロセスのコア・ファイルを、そのプロセスを終了せずに、作成します。作成されたコア・ファイルには、プロセスのメモリー・イメージが入っています。これは、デバッグの際に **dbx** コマンドで使用できます。生成されたコア・ファイルは、*FileName* パラメーターで指定された名前になります。

gencore コマンドは、**chcore** または **syscorepath** コマンドで設定されたロケーションにはコア・ファイルを作成しません。コア・ファイルは、*FileName* パラメーターで指定されたパスに置かれます。*FileName* にファイルの名前のみが指定されている場合、コア・ファイルは現在の作業ディレクトリーに置かれます。

パラメーター

項目	説明
FileName	**gencore** コマンドが作成するコア・ファイルのファイル名を指定します。
ProcessID	**gencore** がコア・ファイルを作成する対象となるプロセスのプロセス ID を指定します。

終了状況

0 コア・ファイルは正常に作成されました。

>0 エラーが発生しました。部分的なコア・ファイルが作成されている可能性があります。

例

1. プロセス ID 1095 のプロセスについて「core.1095」という名前のコア・ファイルを生成するには、次のように入力します。

   ```
   gencore 1095 core.1095
   ```

   プロセスを終了せずに、コア・ファイルが作成されます。

ファイル

項目	説明
/usr/bin/gencore	**gencore** コマンドが入っています。

関連資料:

10 ページの『dbx コマンド』

genfilt コマンド

目的

フィルター規則を追加します。

構文

genfilt -v 4|6 [-n *fid*] **[-a D|P|I|L|E|H|S] -s** *s_addr* **-m** *s_mask* **[-d** *d_addr*] **[-M** *d_mask*] **[-g Y|N] [-c** *protocol*] **[-o** *s_opr*] **[-p** *s_port*] **[-O** *d_opr*] **[-P** *d_port*] **[-r R|L|B] [-w I|O|B] [-l Y|N] [-f Y|N|O|H] [-t** *tid*] **[-i** *interface*] **[-D** *description*] **[-e** *expiration_time*] **[-x** *quoted_pattern*] **[-X** *pattern_filename* **] [-C** *antivirus_filename*]

説明

genfilt コマンドを使用して、フィルター規則テーブルにフィルター規則を追加します。このコマンドによって生成されたフィルター規則は、手動フィルター規則と呼ばれます。 IPsec フィルター規則は、**genfilt** コマンドまたは IPsec smit (IP バージョン 4 または IP バージョン 6) を使用して構成することができます。

フラグ

項目	説明
-a *Action*	次の *Action* 値が使用できます。 • **D** (拒否) トラフィックをブロックします。 • **P** (許可) トラフィックを許可します。 • **I** これを IF フィルター・ルールにします。 • **L** これを ELSE フィルター・ルールにします。 • **E** これを ENDIF フィルター・ルールにします。 • **H** これを SHUN_HOST フィルター・ルールにします。 • **S** これを SHUN_PORT フィルター・ルールにします。 すべての IF ルールは、関連する ENDIF ルールととても緊密です。条件ルールはネストすることができますが、正しいネスティングとスコープを順守する必要があります。順守しないとそのルールは **mkfilt** コマンドを正しくロードしません。
-C *antivirus_filename*	アンチウィルスのファイル名を指定します。**-C** フラグは一部のバージョンの ClamAV ウィルス・データベース (http://www.clamav.net) を認識します。
-c *protocol*	有効な値は、**udp**、**icmp**、**icmpv6**、**tcp**、**tcp/ack**、**ospf**、**ipip**、**esp**、**ah**、および **all** です。値 **all** は、フィルター・ルールがすべてのプロトコルに適用されるように指定します。プロトコルは、数値 (1 から 252 の間) でも指定することができます。デフォルト値は **all** です。 **tcp/ack** の値は、ACK フラグが設定された TCP パケットをチェックすることを意味します。
-D *description*	フィルター・ルールの短い説明のテキストです。このフラグは、静的フィルター規則の場合はオプションで、動的フィルター規則には適用されません。
-d *d_addr*	宛先アドレスを指定します。IP アドレスまたはホスト名です。ホスト名を指定すると、そのホストのネーム・サーバーによって戻される最初の IP アドレスが使用されます。この値は、宛先サブネット・マスクとともに、IP パケットの宛先アドレスと比較されます。
-e *expiration_time*	満了時間を指定します。満了時間 (秒) はルールがアクティブである時間です。 *expiration_time* では、フィルター・ルールがデータベースから除去されません。*expiration_time* は、ネットワーク・トラフィックの処理中にフィルター・ルールがアクティブである時間に関連します。*expiration_time* を指定しないと、フィルター・ルールの存続時間は無限になります。 *expiration_time* を SHUN_PORT (**-a S**) または SHUN_HOST (**-a H**) フィルター・ルールと一緒に指定すると、これらのフィルター・ルール・パラメーターが満たされた場合に、これはリモート・ポートまたはリモート・ホストが拒否または回避される時間を表します。*expiration_time* を回避ルールと独立に指定すると、これは、フィルター・ルールがカーネルにロードされてネットワーク・トラフィック処理が開始したとき、フィルター・ルールが活動中であり続ける時間になります。
-f	フラグメント制御を指定します。このフラグは、すべてのパケット (**Y**)、フラグメント・ヘッダーとフラグメントされていないパケットのみ (**H**)、フラグメントとフラグメント・ヘッダーのみ (**O**)、フラグメントされていないパケットのみ (**N**) のいずれかにこのルールが適用されることを指定します。デフォルト値は **Y** です。

項目	説明
-g	発信元経路指定に適用しますか ? に **Y** (はい) か **N** (いいえ) を指定しなければなりません。 **Y** を指定すると、このフィルター・ルールが送信元経路指定を使用する IP パケットに適用されます。デフォルト値は、はい (**Y**) です。このフィールドは許可規則にしか適用されません。
-i *interface*	フィルター・ルールが適用される IP インターフェースの名前を指定します。名前の例としては、**all**、**tr0**、**en0**、**lo0**、および **pp0** があります。デフォルト値は **all** です。
-l	ログ制御を指定します。**Y** (はい) または **N** (いいえ) を指定する必要があります。**Y** を指定した場合、このフィルター・ルールに適合するパケットがフィルター・ログに取り込まれます。デフォルト値は **N** (いいえ) です。
-M	宛先サブネット・マスクを指定します。これは、IP パケットの宛先アドレスと、フィルター・ルールの宛先アドレスを比較する際に使用されます。
-m	送信元サブネット・マスクを指定します。これは、IP パケットの送信元アドレスと、フィルター・ルールの送信元アドレスを比較する際に使用されます。
-n	フィルター・ルール ID を指定します。新しい規則は、指定されたフィルター規則の前に追加されます。 IP バージョン 4 の場合、ID は 1 より大きくなければなりません。理由は、最初のフィルター規則はシステム生成規則であって、移動できないからです。このフラグが使用されない場合は、フィルター規則テーブルの末尾に新規規則が追加されます。
-O	宛先ポートまたは ICMP コード操作を指定します。これは、宛先ポートまたは ICMP コード (**-P** フラグ) を指定して、パケットの宛先ポート/ICMP コード間の比較を行う際に使用される操作です。有効な値は、**lt**、**le**、**gt**、**ge**、**eq**、**neq**、および **any** です。デフォルト値は **any** です。 **-c** フラグが **ospf** の場合、この値は **any** でなければなりません。
-o	発信元ポートまたは ICMP タイプ操作を指定します。これは、このフィルター規則に送信元ポートまたは ICMP タイプ (**-p** フラグ) を指定してパケットの送信元ポート/ICMP タイプ間を比較する際に使用されるオペレーションです。有効な値は、**lt**、**le**、**gt**、**ge**、**eq**、**neq**、および **any** です。デフォルト値は **any** です。 **-c** フラグが **ospf** の場合、この値は **any** でなければなりません。
-p	発信元ポートまたは ICMP タイプを指定します。これは、IP パケットの送信元ポート (または ICMP タイプ) と比較される値/タイプです。
-P	宛先ポートまたは ICMP コードを指定します。これは、IP パケットの宛先ポート (または ICMP コード) に相当する値/コードです。
-r	経路指定。これによって、規則が、転送パケット (**R**)、ローカル・ホストからの予定されたパケットか発信されたパケット (**L**)、あるいはその両方 (**B**) のいずれに適用されるかが指定されます。デフォルト値は **B** です。
-s *s_addr*	送信元アドレスを指定します。IP アドレスまたはホスト名です。ホスト名を指定すると、そのホストのネーム・サーバーによって戻される最初の IP アドレスが使用されます。この値は、送信元サブネット・マスクとともに、IP パケットの送信元アドレスと比較されます。
-t	このフィルター・ルールに関連したトンネルの ID を指定します。このフィルター・ルールに一致するすべてのパケットが、指定されたトンネルを経由する必要があります。このフラグを指定しないと、この規則は非トンネル・トラフィックのみに適用されます。
-v	フィルター・ルールの IP バージョンを指定します。有効な値は **4** と **6** です。
-w *Direction*	ルールが、受信パケット (I)、発信パケット (O)、または両方 (B) に適用されるかどうかを指定します。デフォルト値は B です。発信方向 (O) を **-x**、**-X**、または **-C** パターン・オプションと一緒に使用することは無効です。両方向 (B) をパターン・オプションと一緒に指定することは有効ですが、受信パケットだけが検査されます。
-X *pattern_filename*	パターン・ファイル名を指定します。複数のパターンがこのフィルター・ルールに関連している場合は、 1 つのパターン・ファイル名を使用しなければなりません。そのパターン・ファイル名は、行あたり 1 パターンのフォーマットである必要があります。パターンは引用符で囲まれていない文字列です。このファイルは、フィルター・ルールが活動化されるときに一度読み取られます。詳しくは、**mkfilt** コマンドを参照してください。
-x *pattern*	引用符で囲まれた文字列またはパターンを指定します。指定されたこの文字列は、前に `0x` が付かない限り、ASCII 文字列として解釈されます。0x が前に付いた場合は、この文字列は 16 進数文字列として解釈されます。 **-x** *pattern* フラグは、ネットワーク・トラフィックと比較されます。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティ

ー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

geninstall コマンド

目的

さまざまなパッケージ・フォーマットのソフトウェア製品をインストールする汎用インストーラー。例えば、installp、RPM、SI、ISMP などです。

構文

geninstall -d *Media* [**-I** *installpFlags*] [**-E** | **-T**] [**-t** *ResponseFileLocation*] [**-e** *LogFile*] [**-p**] [**-F**] [**-Y**] [**-Z**] [**-D**] { **-f** *File* | *Install_List*] | all}

または

geninstall -k [**-d** *Media*] [**-p**] [**-Y**] [**-f** *File* | *install_list* | `all` | `update_all`]

または

geninstall -u [**-e** *LogFile*] [**-E** | **-T**] [**-t** *ResponseFileLocation*] [**-D**] {**-f** *File* | *Uninstall_List*...}

または

geninstall -L -d *Media* [**-e** *LogFile*] [**-D**]

説明

現在の **installp** フラグをすべて受け入れ、それらを **installp** に渡します。フラグの一部 (例えば **-L**) は、メディア上のすべての製品を表示するために、過負荷になっています。 ISMP パッケージ製品に関係のないフラグは無視されます。これにより、プログラム (NIM など) は、 **installp** フラグを常に **geninstall** に送信し続けることができますが、使用されるのは関係のあるフラグだけです。

geninstall コマンドは、**/etc/check_config.files** にリストされている構成ファイルに対して、どのような変更が行われたかを調べる簡単な方法を提供します。 **geninstall** インストールまたは更新操作中にこれらのファイルが変更されると、旧ファイルと新ファイル間の違いが **/var/adm/ras/config.diff** に記録されます。**/etc/check_config.files** で旧ファイルを保管するように要求している場合は、旧ファイルは **/var/adm/config** ディレクトリーにあります。

/etc/check_config.files は編集することができ、変更された旧構成ファイルを保管するか (`s` で示す)、または削除するか (`d` で示す) どうかを指定するために使用することができます。フォーマットは次のとおりです。

```
d /etc/inittab
```

geninstall コマンドのインストール・アクティビティーの要約が、 **/var/adm/sw/geninstall.summary** に保管されています。このファイルには、**installp** でインストールしたファイルセットと、ISMP でインストールしたコンポーネントのリストが、コロンで区切られて入っています。これは主として、サイレント・インストールの要約情報を提供するために使用されます。

注**:** ISMP パッケージのインストールおよび応答ファイルの使用についての詳細は、 **/usr/lpp/bos** ディレクトリーにある **README.ISMP** ファイルを参照してください。また、**geninstall** コマンドは、並行更新

を含む暫定修正のファイルをインストールすることができます。並行更新を含む暫定修正は、インストール・イメージを含むディレクトリー内の **cupdates** というサブディレクトリーに置かれなければなりません。これで **geninstall** コマンドは更新を適切にインストールします。

フラグ

項目	説明
-d *Device* または *Directory*	インストールするイメージが入ったデバイスまたはディレクトリーを指定します。 **geninstall** コマンドは、以下のパスにあるイメージを検索します。 • */mount_point*/installp/ppc (installp パッケージ) • */mount_point*/RPMS/ppc (RPM パッケージ) • */mount_point*/emgr/ppc (AIX 用の暫定修正パッケージ) • */mount_point*/ISMP/ppc (AIX 用の ISMP パッケージ) パスが存在しない場合、**geninstall** コマンドは特定のデバイスのベース・ディレクトリーにあるイメージを検索します。イメージ名の前にイメージ・タイプを示すプレフィックスが付いていない場合、**geninstall** コマンドはイメージのタイプを識別します。 パスが存在し、どのパスにもイメージがなく、しかもイメージがプレフィックスを含んでいない場合、イメージは RPM フォーマットのイメージとして扱われます。
-D	デバッグ・モードを指定します。このフラグは、このスクリプトをデバッグするためのものです。これにより大量の出力が作成されるため、通常の操作にこれを使用すべきではありません。
-e *LogFile*	イベント・ロギングを使用可能にします。**-e** フラグを指定することにより、 **geninstall** コマンド出力の特定部分を *LogFile* 変数で指定したファイルに追加することができます。*LogFile* 変数には、既存の書き込み可能なファイルを指定する必要があります。また、そのファイルがあるファイルシステムに、ログを保管するのに十分なスペースがなければなりません。ログ・ファイルは循環しません。
-E	デフォルトのロケーションに ISMP 応答記録ファイルを作成します。これは製品インストール・ファイルを含むディレクトリーです。このオプションを指定する際は、 ISMP インストールまたはアンインストールを対話式で完全に実行する必要があります。結果の応答ファイルを使用すると、それ以降同じ製品をインストールまたはアンインストールする際に同じオプションが提供されます。応答記録ファイルを作成することにより、結果的に製品がインストールまたはアンインストールされます。
-f *File*	ターゲット・ロケーションにコピーするイメージのリストを入れるファイルを指定します。**installp**、RPM、および ISMP のイメージには、プレフィックスをそれぞれ **I:**、**R:**、および **J:** と付ける必要があります。暫定修正パッケージにはプレフィックス **E:** を付けてください。
-F	既にインストールされているパッケージを再インストールしたり、現在インストールされているバージョンより古いパッケージをインストールします。
-I *installpFlags*	**installp** コマンドを呼び出す際に使用する、 **installp** フラグを指定します。 **installp** のインストール操作時に使用されるフラグは、 **a**、**b**、**c**、 **D**、**e**、**E**、 **F**、**g**、**I**、 **J**、**M**、**N**、**O**、**p**、**Q**、 **q**、**S**、**t**、 **v**、**V**、**w**、 **X** フラグです。インストール時に使用されない **installp** フラグは、 **C**、**i**、**r**、 **z**、**A**、および **l** フラグです。これらの機能を実行するには、 **installp** コマンドを直接呼び出さなければなりません。 **-u**、**-d**、**-L**、 **-f** フラグは、**-I** フラグの外側で指定してください。

項目	説明
-k	このフラグを IBM AIX 7.2 テクノロジー・レベル 1 以降で使用すると、AIX Live Update操作を使用して、Service Pack (SP)、テクノロジー・レベル (TL)、グループ更新または個別更新、および LU CAPABLE としてマーク付けされている暫定修正をアップグレードすることができます。暫定修正が LU CAPABLE であるかどうかを判別するために、インストールのプレビューを実行できます。Live Update操作を開始する前に、システム上に既存の更新があれば、それらをコミットしておく必要があります。Live Update操作は、プロセス中に適用されたすべての更新を常にコミットし、必要なファイル・システムを展開して、必要なソフトウェアをすべてインストールします。Live Update操作中、他のすべてのインストール・プロセスは開始されません。 **注:** Live Update操作に更新のインストールが含まれている場合は、Live Update操作を開始する前に、実行可能なシステム・バックアップを実行しておく必要があります。Live Update操作では、バックアップ・イメージは作成されません。 ご使用の環境で以下のタスクを実行している場合、Live Update操作は失敗します。 • ワークロード・パーティション (wpar) 環境、multibos 環境、または代替ディスク環境などの特殊な環境で暫定修正をインストールする。 • オペレーティング・システムをインストールする。 • 必須オペレーティング・システム論理ボリューム (/、/var、/opt、/usr、/etc) およびブート論理ボリュームの論理ボリューム名が、デフォルトの論理ボリューム名として使用されていない。 Live Update のオペレーションには、/var/adm/ras/liveupdate/lvupdate.data ファイルで追加入力を指定する必要があります。このファイルについて詳しくは、/var/adm/ras/liveupdate/lvupdate.templateを参照してください。 更新が正常に適用されてコミットされ、Live Update・プロセスが失敗した場合は、**-k** フラグだけを指定して **geninstall** コマンドを実行することにより、Live Update・プロセスを再実行できます。このシナリオでは、コマンドにデバイスとソフトウェアを含める必要はありません。 AIX 7200-01 以降では、SMIT または何らかの好みの方法を使用して更新や暫定修正を適用およびコミットし、**-d** フラグを使用せずにLive Update操作を実行することができます。これにより、システムを再始動せずに済みます。 all オプションは、**-d** フラグで指定されたデバイスまたはディレクトリーにあるすべてのソフトウェアをインストールします。このプロセスは、そのソフトウェアが以前にシステムにインストールされていなかった場合でも、**-d** フラグによって指定されたソフトウェア・ソースにあるすべてのソフトウェアをインストールします。update_all オプションは、既にシステムにインストールされている、指定されたソフトウェアの上位バージョンだけをインストールします。update_all オプションは、既存のソフトウェアの新規バージョンが新規ソフトウェアを必要とする場合にのみ、新規ソフトウェアをインストールします。 **注:** Live Update操作を実行するには、bos.liveupdate.rte ファイルセットがインストールされている必要があります。
-L	メディアの内容を表示します。出力フォーマットは **installp -Lc** フォーマットと同じで、ISMP および RPM フォーマット製品の末尾にはフィールドが追加されています。
-p	指定されたアクションについてすべての事前インストール検査を実行して、アクションのプレビューを行います。
-t *ResponseFileLocation*	応答ファイルまたは応答ファイル・テンプレートの代替ロケーションの指定を許可します。デフォルトのロケーションは、製品インストール・ファイルを含むディレクトリーです。このフラグは、応答ファイルの記録またはテンプレートを、別のロケーションに作成するために使用できます。 *ResponseFileLocation* は、ファイル名でもディレクトリー名でもかまいません。 *ResponseFileLocation* がディレクトリーの場合は、このディレクトリーは既存のものでなければなりません。*ResponseFileLocation* が既存のディレクトリーでない場合は、ファイル名を指定したものと見なされます。

項目	説明
-T	デフォルトのロケーションに ISMP 応答ファイル・テンプレートを作成します。このロケーションは製品インストール・ファイルを含むディレクトリーです。作成されたテンプレートを使って、希望するオプションを指定し、同じ製品の今後のインストールまたはアンインストール用の応答ファイルを作成することができます。応答ファイル・テンプレートを作成することによって、結果的に製品がインストールまたはアンインストールされることはありません。
-u	指定されたソフトウェアのアンインストールを実行します。ISMP 製品の場合、ベンダーのデータベースにリストされているアンインストーラーが呼び出され、プレフィックス「J:」が付けられます。
-Y	ソフトウェアのインストールに必要なソフトウェア・ライセンス契約に同意します。このフラグは、**-I** オプションを付けた、**installp** フラグとしても受け入れられます。
-Z	**geninstall** に、インストールをサイレント・モードで起動するよう指示します。

例

1. ドライブ cd0 の CD メディア上にある製品をすべてインストールするには、次のように入力します。

   ```
   geninstall -d /dev/cd0 all
   ```

   メディア上に ISMP イメージがある場合には、グラフィカル・インターフェースが提供されます。**installp**、SI、または RPM イメージの場合は、**installp** イメージが複数の CD にまたがっていない限り、プロンプトなしでインストールされます。
2. /images/emgr/ppc ディレクトリーにある IV12345.160101.epkg.Z という名前の暫定修正をインストールするには、次のコマンドを入力します。

   ```
   geninstall -d /images IV12345.160101.epkg.Z
   ```

   注: /images/emgr/ppc ディレクトリーが存在しても、パッケージが /images ディレクトリーに入っていると (/images/IV12345.160101.epkg.Z)、**geninstall** コマンドはそのパッケージを暫定修正と見なさず、そのパッケージを RPM フォーマットのイメージとしてインストールしようとします。詳しくは、**-d** フラグを参照してください。

ファイル

- /usr/sbin/gencopy
- /usr/sys/inst.data/sys_bundles
- /usr/sys/inst.data/user_bundles

関連情報:

installp コマンド

Live Update のインストール

制限

さまざまなフォーマットのソフトウェア・パッケージのインストール

genkex コマンド

目的

genkex コマンドは、現在システムにロードされているカーネル拡張機能のリストを抽出して、リストの各カーネル拡張機能のアドレス、サイズ、およびパス名を表示します。

構文

genkex [**-dh**]

説明

カーネルは、システムにロードされているカーネル拡張について、ローダー・エントリーと呼ばれるデータ構造で構成されるリンク・リストを保持します。ローダー・エントリーには、拡張名、開始アドレス、およびそのサイズが入っています。この情報は、**genkex** コマンドによって収集され報告されます。

フラグ

項目	説明
-d	テキスト・セクションのアドレスとサイズに加えて、データ・セクションのアドレスとサイズを示します。
-h	使用方法に関するステートメントを表示します。

例

ロードされているカーネル拡張のリストを生成するには、次のように入力します。

```
genkex
```

関連資料:

『genkld コマンド』

686 ページの『genld コマンド』

関連情報:

コマンドとサブルーチンのモニターおよびチューニング

---

genkld コマンド

目的

genkld コマンドは、現在システムにロードされている共用オブジェクトのリストを抽出して、リストの各オブジェクトのアドレス、サイズ、およびパス名を表示します。

構文

genkld [**-dh**]

説明

システムにロードされた共用オブジェクトの場合、カーネルはローダー・エントリーと呼ばれるデータ構造から成るリンク・リストを保持します。ローダー・エントリーには、オブジェクト名、開始アドレス、そのサイズが入っています。この情報は、**genkld** コマンドによって収集され報告されます。

フラグ

項目	説明
-d	テキスト・セクションのアドレスとサイズに加えて、データ・セクションのアドレスとサイズを示します。
-h	使用方法に関するステートメントを表示します。

例

ロードされている共用オブジェクトのリストを取得するには、次のように入力します。

```
genkld
```

関連資料:

684 ページの『genkex コマンド』

『genld コマンド』

関連情報:

コマンドとサブルーチンのモニターおよびチューニング

---

genld コマンド

目的

genld コマンドは、システム上で現在実行されているすべてのプロセスのリストを収集し、オプションで、それぞれのプロセスに対応するロード済みオブジェクトのリストを報告します。

構文

genld [**-h** | **-l** [**-d**]] [**-a** *Area*] [**-u**]

説明

genld コマンドは、現在実行されている各プロセスについて、そのプロセス ID とプロセス名、およびそのプロセスのロード済みオブジェクトのリスト (オプション) で構成されるレポートを表示します。オブジェクトのアドレスとパス名が表示されます。ライブラリーのメンバーは、ブラケットで囲んで示されます。例えば、`/usr/lib/libc.a[shr.o]` は、`shr.o` が **libc.a** ライブラリーのロード済みメンバーであることを意味します。

-u フラグを使用して **genld** コマンドの出力をフィルターに掛け、ロード済みオブジェクトの古いバージョンがあるプロセスを表示することができます。オブジェクトは、そのオブジェクト・イメージが現在ファイル・システムにインストールされているイメージと異なる場合、古いオブジェクトと見なされます。**-u** フラグは、更新を適用した後に、新しいバイナリーとライブラリーを使用するために再始動操作を必要とするプロセスをリストする目的で使用されます。

注**:**

- 特権のないユーザーは、ロード済みオブジェクトをそのプロセスでのみ見ることができます。
- ロード済みオブジェクトへのフルパス名を判別できない場合、**genld** コマンドは、そのオブジェクトがジャーナル・ファイル・システム (JFS2) 以外のファイル・システム上に置かれていると、そのオブジェクトに対する更新を報告しないこともあります。オブジェクトが同一のコピーによって置き換えられた場合も、オブジェクトが `updated` として報告されることがあります。

フラグ

項目	説明
-a *Area*	*Area* パラメーターで指定された共用ライブラリー領域を使用するプロセスのみがリストされます。
-d	テキスト・セクションのアドレスとサイズに加えて、データ・セクションのアドレスとサイズを示します。このオプションは、**-l** フラグがなければ効果がありません。
-h	使用方法に関するステートメントを表示します。
-l	現在システム上で実行されている各プロセスのロード済みオブジェクトのリストを報告します。
-u	ロード済みオブジェクトの古いバージョンがあるプロセスだけをリストします。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

- 実行されている各プロセスのロード済みオブジェクトのリストを取得するには、次のコマンドを入力します。

  ```
  genld -l
  ```

- ロード済みオブジェクトの古いバージョンがあるプロセスのリストを取得するには、次のコマンドを入力します。

  ```
  genld -u
  ```

- 古いバージョンの **libcrypt.a** ライブラリーがロードされているプロセスのリストを取得するには、次のコマンドを入力します。

  ```
  genld -lu | grep -p libcrypt.a
  ```

関連資料:

684 ページの『genkex コマンド』

685 ページの『genkld コマンド』

関連情報:

コマンドとサブルーチンのモニターおよびチューニング

---

gennames コマンド

目的

filemon コマンドと **netpmon** コマンドをオフライン・モードで実行するのに必要なすべての情報を収集します。

構文

gennames[-f]

説明

gennames コマンドは、 **filemon** および **netpmon** コマンドがオフライン・モードで動作するのに必要な名前とアドレスのマッピング情報を収集します。収集される情報には、次のようなものがあります。

- すべてのロード済みカーネル拡張機能のリスト。 **genkex** コマンドが報告するものと同様です。
- すべてのロード済み共用ライブラリーのリスト。 **genkld** コマンドが報告するものと同様です。

- すべてのロード済みプロセスのリスト。 **genld** コマンドが報告するものと同様です。
- **/unix** およびすべてのカーネル拡張機能およびライブラリーでは、**stripnm -z** コマンドの出力が収集されます。

フラグ

項目	説明
-f	物理ボリュームおよび論理ボリュームのデバイス情報を収集します。このオプションは、オフラインの **filemon** によって使用される仮想ファイルシステム情報も出力します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

filemon コマンドをオフライン・モードで実行する際に必要な情報を収集するには、次のように入力します。

```
gennames -f > gen.out
```

関連資料:

684 ページの『genkex コマンド』

関連情報:

netpmon コマンド

stripnm コマンド

コマンドとサブルーチンのモニターおよびチューニング

---

gensyms コマンド

目的

curt、**splat**、**tprof** の各コマンドをオフライン・モードで実行するために必要なすべての情報を収集します。

構文

gensyms [**-o**] [**-f**] [**-F**] [**-h**] [**-s**] [**-g**] [**I**] [**-N**] [**-k** *kernel*] [**-i** *file*] [**-b** *binary*[,*binary*[,...]]] [**-P** *pid*[,*pid*[,...]]] [**-S** *path*]

説明

gensyms コマンドは、**curt**、**splat**、**tprof** の各コマンドがオフライン・モードで動作するのに必要な、名前からアドレスへのマッピング情報を収集します。収集される情報には、以下の項目が含まれます。

- すべてのロード済みカーネル・エクステンションのリスト。
- すべてのロード済み共用ライブラリーのリスト。
- すべてのロード済みプロセスのリスト。

- **/unix**、すべてのカーネル・エクステンション、ライブラリー、およびプロセスに対応するすべてのオブジェクト・ファイルでは、**stripnm** コマンドの出力が収集されます。

フラグ

項目	説明
-b *binary*	シンボルの検索対象のバイナリーのオプショナル・リストを指定します。
-f	ソース・ファイル名が出力されるのを抑止します。
-F	物理ボリュームおよび論理ボリュームのデバイス情報を収集します。
-g	シンボル名をデコードします。
-h	ヘルプ・メッセージを出します。
-i *file*	指定したファイルからシンボルを読み取ります。
-I	シンボルのバイナリー命令を出力します。
-k *kernel*	カーネル・イメージの名前を指定します (デフォルトは **/unix**)。
-N	シンボルのソース行番号を出力します。
-o	アドレスの代わりにオフセットを出力します。
-P *pid*[,*pid*[,...]]	指定されたプロセスによってロードされた従属モジュールのシンボルを出力します。このフラグはオプションです。
-s	**-k** フラグおよび **-b** フラグによって指定されたファイルのシンボルだけを検出します。
-S *path*	検索パス・リストを指定します。このリストはバイナリーの検索に使用されます。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. ユーザー・プログラム・テストのプロファイルを使用して、**tprof** コマンドをオフライン・モードで実行するために必要な情報を収集するには、次のコマンドを入力します。

   ```
   gensyms > test.syms
   ```

2. 指定されたプロセス ID およびその従属プロセスに関する情報を収集するには、次のコマンドを入力します。

   ```
   gensyms -P pid > test.syms
   ```

関連資料:

687 ページの『gennames コマンド』

関連情報:

splat コマンド

tprof コマンド

コマンドとサブルーチンのモニターおよびチューニング

---

gentun コマンド

目的

トンネル・データベースにトンネル定義を作成します。

構文

gentun -s *src_host_IP_address* **-d** *dst_host_IP_address* **-v 4|6** [**-t** *tun_type*] [**-m** *pkt_mode*] [**-t** *IBM*] [**-t** *manual*] [**-m** *tunnel*] [**-m** *transport*] [**-f** *fw_address*] [**-x** dst_mask]] [**-e** [*src_esp_algo*]] [**-a** [*src_ah_algo*]] [**-p** *src_policy*] [**-A** [*dst_ah_algo]*] [**-P** *dst_policy*] [**-k** *src_esp_key*] [**-h** *src_ah_key*] [**-K** *dst_esp_key*] [**-H** *dst_ah_key*] [**-n** *src_esp_spi*] [**-u** *src_ah_spi*] [**-N** *dst_esp_spi*] [**-U** *dst_ah_spi*] [**-b** *src_enc_mac_algo*] [**-c** *src_enc_mac_key*] [**-B** *dst_enc_mac_algo*] [**-C** *dst_enc_mac_key*] [**-g**] [**-z**] [**-E**]

説明

gentun コマンドは、ローカル・ホストとトンネル・パートナー・ホスト間のトンネルの定義を作成します。トンネルのための関連自動生成フィルター規則は、このコマンドによってオプションとして生成することができます。

フラグ

項目	説明
-a	IP パケットの認証のために送信元ホストが使用する認証アルゴリズムです。 **-a** の有効な値は、ホストにインストールされている認証アルゴリズムによって異なります。**ipsecstat -A** コマンドを出せば、すべての認証アルゴリズムのリストを表示することができます。 デフォルト値は、**manual** トンネルの場合は HMAC_MD5 です。
-A	(**manual** トンネルのみ) IP パケット認証のために宛先が使用する認証アルゴリズムです。**-A** の有効な値は、ホストにインストールされている認証アルゴリズムによって異なります。**ipsecstat -A** コマンドを出せば、すべての認証アルゴリズムのリストを表示することができます。このフラグを使用しないと、**-a** フラグが使用する値が使用されます。
-b	(**manual** トンネルのみ) 送信元 ESP の認証アルゴリズムです (新しいヘッダー・フォーマットのみ)。**-b** の有効な値は、ホストにインストールされている認証アルゴリズムによって異なります。 **ipsecstat -A** コマンドを出せば、すべての認証アルゴリズムのリストを表示することができます。
-B	(**manual** トンネルのみ) 宛先 ESP の認証アルゴリズムです (新しいヘッダー・フォーマットのみ)。 **-B** の有効な値は、ホストにインストールされている認証アルゴリズムによって異なります。**ipsecstat -A** コマンドを出せば、すべての認証アルゴリズムのリストを表示することができます。このフラグを使用しないと、**-b** フラグと同じ値に設定されます。
-c	(**manual** トンネルのみ) 送信元 ESP の認証キーです (新しいヘッダー・フォーマットのみ)。この値は「0x」で始まる 16 進文字列でなければなりません。このフラグを使用しないと、システムがフラグを生成します。
-C	(**manual** トンネルのみ) 宛先 ESP の認証キーです (新しいヘッダー・フォーマットのみ)。この値は「0x」で始まる 16 進文字列でなければなりません。このフラグを使用しないと、**-c** フラグと同じ値に設定されます。
-d	宛先ホストの IP アドレス。ホスト - ホストの接続の場合、これは、トンネルが使用する宛先ホスト・インターフェースの IP アドレスです。ホスト - ファイアウォール - ホストの接続の場合は、これはファイアウォールの後ろにある宛先ホストの IP アドレスです。ホスト名も有効であり、そのホスト名のネーム・サーバーによって戻される最初の IP アドレスが使用されます。
-e	IP パケットの暗号化のために送信元が使用する暗号化アルゴリズムです。 **-e** の有効な値は、ホストにインストールされている暗号化アルゴリズムによって異なります。すべての暗号化アルゴリズムのリストを表示するには、**ipsecstat -E** コマンドを発行します。
-E	(**manual** トンネルのみ) IP パケット暗号化のために宛先が使用する暗号化アルゴリズムです。**-E** の有効な値は、ホストにインストールされている暗号化アルゴリズムによって異なります。すべての暗号化アルゴリズムのリストを表示するには、**ipsecstat -E** コマンドを発行します。このフラグを使用しないと、**-e** フラグが使用する値が使用されます。
-f	送信元ホストと宛先ホスト間にあるファイアウォールの IP アドレスです。このホストとファイアウォール間には、トンネルが確立されます。したがって、対応するトンネル定義はファイアウォール・ホスト上に作成しなければなりません。このフラグにはホスト名が使用される場合もあり、そのホスト名のネームサーバーにより戻される最初の IP アドレスが使用されます。
-g	システム自動生成フィルター規則フラグです。このフラグを使用しないと、コマンドはトンネルのための 2 つのフィルター規則を自動的に生成します。自動生成フィルター規則では、トンネルの 2 つのエンドポイント間で IP トラフィックがトンネルを通過することができます。 **-g** フラグを指定すると、コマンドはトンネル定義を作成するだけで、トンネルを作動させるには、ユーザーがユーザー定義のフィルター規則を追加しなければなりません。
-h	これは、**manual** トンネル用の AH キー文字列です。この入力は「0x」で始まる 16 進文字列でなければなりません。このフラグを使用しないと、システムが乱数生成プログラムを用いてキーを生成します。

項目	説明
-H	(**manual** トンネルのみ) 宛先 AH 用のキー文字列です。この入力は「0x」で始まる 16 進文字列でなければなりません。このフラグを使用しないと、システムが乱数生成プログラムを用いてキーを生成します。
-k	これは、**manual** トンネル用の ESP キー文字列です。これは、送信元がトンネルを作成するのに使用します。この入力は「0x」で始まる 16 進文字列でなければなりません。このフラグを使用しないと、システムが乱数生成プログラムを用いてキーを生成します。
-K	(**manual** トンネルのみ) 宛先 ESP 用のキー文字列です。この入力は「0x」で始まる 16 進文字列でなければなりません。このフラグを使用しないと、システムが乱数生成プログラムを用いてキーを生成します。
-l	分単位で指定されるキー・ライフタイムです。 **manual** トンネルの場合は、この値は、トンネルが満了するまでの動作可能時間を示します。 **manual** トンネルの有効な値は 0 から 44640 です。値 0 は、**manual** トンネルが満了になることがないことを示します。**manual** トンネルのデフォルト値は 480 です。
-m	セキュア・パケット・モード。この値は、**tunnel** または **transport** として指定する必要があります。デフォルト値は **tunnel** です。トンネル・モードは IP パケット全体をカプセル化しますが、トランスポート・モードは IP パケットのデータ部分しかカプセル化しません。ホスト - ファイアウォール - ホストのトンネルを生成するとき (ファイアウォールの後ろにホストがある場合) は、このフラグに **tunnel** の値を使用しなければなりません。 **-f** フラグを指定すると、-m フラグは、強制的にデフォルト値 (**tunnel**) を使用するようになります。
-n	(**manual** トンネルのみ) 送信元 ESP のセキュリティー・パラメーター索引です。これは、宛先 IP アドレスと一緒になって、ESP を使用するパケットに使用するセキュリティー・アソシエーションを識別する数値です。このフラグを使用しないと、代わりにシステムが SPI を生成します。
-N	(**manual** トンネルのみ) 宛先 ESP のセキュリティー・パラメーター索引です。 **-P** フラグに指定されたポリシーに ESP が含まれている場合は、 **manual** トンネルでこの索引を組み込まなければなりません。このフラグは、**IBM** トンネルには適用されません。
-p	このホストが IP パケット認証または暗号化 (あるいはその両方) をどのように使用するかを識別する送信元ポリシーです。**ea** と指定すると、IP パケットは認証前に暗号化されます。 **ae** と指定された場合は IP パケットは認証後に暗号化されますが、**e** だけの指定もしくは **a** だけの指定は、IP パケットが暗号化されるだけか、あるいは認証されるだけと同じです。このフラグのデフォルト値は、**-e** と **-a** フラグが指定されたかどうかによって決まります。**-e** と **-a** フラグのどちらかが指定されたか、どちらも指定されなかったかにより、デフォルト・ポリシーは **ea** になります。それ以外では、ポリシーは **-e** と **-a** フラグのどちらが指定されたかを反映します。
-P	(**manual** トンネルのみ) 宛先ポリシー。宛先が IP パケット認証または暗号化 (あるいはその両方) をどのように使用するかを識別します。**ea** と指定すると、IP パケットは認証前に暗号化されます。これを **ae** と指定すると、認証後に暗号化されます。ただし、**e** または **a** と指定した場合には、暗号化のみ、または、認証のみが IP パケットに対して行われます。**-E** と **-A** フラグのどちらかが指定されたか、どちらも指定されなかったかにより、デフォルト・ポリシーは **ea** になります。それ以外では、ポリシーは **-E** と **-A** フラグのどちらが指定されたかを反映します。
-s	送信元ホスト IP アドレス。トンネルが使用するローカル・ホスト・インターフェースの IP アドレスです。ホスト名も有効であり、そのホスト名のネーム・サーバーによって戻される最初の IP アドレスが使用されます。
-t	トンネルのタイプです。 **manual** と指定する必要があります。 **manual** トンネルを使用するときは、初期トンネル・キーとそれ以降のキー更新を手動で行う必要があります。キーを手動でインストールしたら、それを手動で変更するまで、すべてのトンネル操作に同じキーが使用されます。 **manual** トンネル値は、IBM 以外のセキュリティー・ホストまたは IP バージョン 6 エンドポイントでトンネルを構成したいときに選択する必要があります。この場合、IP トンネルの新しい IP セキュリティーのカプセル化フォーマットに関して、エンドポイントは RFC 1825-1829 か IETF ドラフトをサポートしています。
-u	(**manual** トンネルのみ) 送信元 AH のセキュリティー・パラメーター索引です。AH に使用するセキュリティー・アソシエーションを決めるため、SPI と宛先 IP アドレスを使用してください。このフラグを使用しないと、-n SPI の値が使用されます。
-U	(**manual** トンネルのみ) 宛先 AH のセキュリティー・パラメーター索引です。このフラグを使用しないと、-N spi の値が使用されます。
-v	トンネルを作成する場合の IP のバージョンです。 IP バージョン 4 トンネルの場合は **4** の値を使用します。IP バージョン 6 トンネルの場合は **6** の値を使用します。

項目	説明
-x	ファイアウォールの後ろにあるセキュア・ネットワークのネットワーク・マスクです。宛先ホストはセキュア・ネットワークのメンバーです。**-d** と **-x** を組み合わせると、送信元ホストは、送信元とファイアウォール間のトンネルを介してセキュア・ネットワーク内の複数のホストと通信できます (ただし、セキュア・ネットワークはトンネル・モードでなければなりません)。 このフラグは、**-f** フラグが使用されている場合に限り有効です。
-y	(**manual** トンネルのみ) 再生防止フラグです。再生防止は、ESP または AH ヘッダーが新しいヘッダー・フォーマットを使用している場合に限り有効です (**-z** フラグを参照)。**-y** フラグの有効な値は、Y (はい) および N (いいえ) です。このトンネルで使用されるすべてのカプセル化 (AH、ESP、送信、および受信) は、このフラグの値が Y の場合に、再生フィールドを使用します。デフォルト値は N です。
-z	(**manual** トンネルのみ) 新しいヘッダー・フォーマット・フラグです。新規ヘッダー・フォーマットは、再生防止のために ESP および AH ヘッダー内にフィールドを保存し、また、ESP 認証も許可します。再生フィールドが使用されるのは、再生フラグ (**-y**) が Y に設定される場合だけです。**-z** フラグの有効な値は Y (はい) と N (いいえ) です。**-z** フラグが使用されないときのデフォルト値は、トンネルに選んだアルゴリズムによって決まります。**-a** か **-A** フラグで KEYED_MD5 以外のアルゴリズムが使用されるか、あるいは **-b** か **-B** フラグが使用される場合以外は、この値のデフォルト値は N です。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

関連資料:

477 ページの『exptun コマンド』

関連情報:

chtun コマンド

imptun コマンド

lstun コマンド

---

genxlt コマンド

目的

lconv ライブラリーで使用するコード・セット変換テーブルを生成します。

構文

genxlt [OutputFile]

説明

genxlt コマンドでは、標準入力からソース・コード・セット変換テーブル・ファイルが読み取られて、コンパイル済みバージョンが *OutputFile* パラメーターで指定したファイルに書き込まれます。 *OutputFile* パラメーターに値を指定しないと、標準出力が使用されます。ソース・コード・セット変換テーブル・ファイルには、**genxlt** コマンドによって作動する、コンパイル済みバージョンを生成するための指示が入っています。

コード・セット変換テーブル・ソース・ファイルのフォーマットは次のとおりです。

- 最初のホワイト・スペース以外の文字が # (ポンド記号) である行は、コメント行として扱われます。

- ヌル行およびホワイト・スペース文字だけで構成されている行は、コメント行として扱われます。
- コメント以外の行は、次のような形式でなければなりません。

```
%token <blank>   # <tab> and <space>
%token <hex>     # <zero>, <one>, <two>, <three>, <four>,
                 # <five>, <six>, <seven>, <eight>, <nine>,
                 # <a>, <b>, <c>, <d>, <e>, <f>,
                 #  <A>, <B>, <C>, <D>, <E>, <F>,
%token <any>     # any character but '¥n'
line    :  offset  blank  value  blank  comment  '¥n'
        | 'SUB'  blank  value  blank  comment  '¥n'
        ;

blank    : <blank>
        | blank <blank>
        ;

offset   :  '0x'  <hex>
        |  offset  <hex>
        ;


value    :  offset
        |  'invalid'
        |  'substitution'
        ;


comment  :  '#'  <any>
        |  comment  <any>
        ;
```

オフセットが 'SUB' の行は、デフォルトの置換文字を指定するために使用します。

テーブルが 'substitution' に設定されると、このテーブルを使用する **iconv** コンバーターによってこのオフセットの SUB 値が使用されます。

値が 'invalid' に設定されると、このテーブルを使用する **iconv** コンバーターによって、オフセットのエラーが戻されます。

オフセットがソース・コード・セット変換テーブル・ファイル内で何度も検出されると、変換テーブルのコンパイルで最後のエントリーが使用されます。

オフセットと値は、0x00 から 0xff の範囲内に限定されます。

次に、コード・セット変換テーブルから一部抜粋したものを紹介します。

```
SUB     0x1a     substitute character
0x80    0xc7     C  cedilla
0x81    0xfc     u  diaeresis
0x82    0xe9     e  acute
0x83    0xe2     a  circumflex
0x84    0xe4     a  diaeresis
0x85    0x40     a  grave
0x9F            substitution
0xff            invalid
```

うまくいくと、**genxlt** コマンドは値 0 を出して終了します。出力ファイルをオープンできない場合は、**genxlt** コマンドの実行は失敗に終わり、戻り値 1 を出して終了します。入力ストリーム内で構文エラーが見つかると、**genxlt** コマンドは戻り値 2 を出して直ちに終了し、構文エラーが発生した行の番号が標準エラーに書き出されます。

genxlt コマンドによって作成されるファイルの名前は、**iconv** サブシステムがそのファイルを変換ファイルとして認識できるように、次の命名規則に準じたものでなければなりません。

```
fromcode: "IBM-850"
tocode: "ISO8859-1"
conversion table file: "IBM-850_ISO8859-1"
```

変換テーブル・ファイル名は、fromcode ファイル名と tocode ファイル名を下線で連結して作成されます。

例

英語以外の言語によるユーザー定義のコード・セット変換テーブルを生成するには、次のように入力します。

```
cp  /usr/lib/nls/loc/iconvTable/ISO8859-1_IBM-850_src  $HOME
vi  $HOME/ISO8859-1_IBM-850_src
genxlt  <  $HOME/ISO8859-1_IBM-850_src  >  cs1_cs2
```

関連情報:

iconv コマンド

iconv_open コマンド

iconv_close コマンド

プログラミングのためのコンバーターの概要

---

get コマンド

目的

指定されたバージョンのソース・コード制御システム (SCCS) のファイルを作成します。

構文

SCCS ファイルの読み取り専用バージョンを作成する

get [**-g**] [**-m**] [**-n**] [**-p**] [**-s**] [**-c** *Cutoff*] [**-i** *List*] [**-r** *SID*] [**-t**] [**-x** *List*] [**-w** *String*] [**-l** [**p**]] [**-L**] *File ...*

SCCS ファイルの編集可能バージョンを作成する

get [**-e**] [**-k**] [**-b**] [**-s**] [**-c** *Cutoff*] [**-i** *List*] [**-r** *SID*] [**-t**] [**-x** *List*] [**-l** [**p**]] [**-L**] *File ...*

説明

get コマンドでは、指定したバージョンのソース・コード制御システム (SCCS) ファイルが読み取られ、指定したフラグに従って ASCII テキスト・ファイルが作成されます。次に、**get** コマンドによって各テキスト・ファイルが、オリジナルの SCCS ファイルと同じ名前で **s.** プレフィックス (**g** ファイル) が付いていないファイルに書き込まれます。

フラグとファイルは任意の順序で指定できます。また、すべてのフラグは、指定したすべてのファイルに適用されます。 *File* パラメーターのディレクトリーを指定すると、**get** コマンドによって、要求したアクションが、ディレクトリー内の **s.** プレフィックスで始まるすべてのファイルに対して実行されます。 *File*

パラメーターの値として **-** (負符号) を指定すると、**get** コマンドによって標準入力が読み取られ、各行が SCCS ファイル名として解釈されます。 **get** コマンドでは、ファイル終わり文字が読み取られるまで入力が読み取られます。

実効ユーザーが SCCS ファイルが入っているディレクトリー内で書き込み許可を持っていても、実ユーザーが書き込み許可を持っていなければ、**-e** フラグを使うときに 1 つのファイルしか命名できません。

注**: get** コマンドでは、 **w** フラグを使って指定するファイル名と文字列のデータについて、マルチバイト文字セット (MBCS) がサポートされます。

読み取り専用ファイル・バージョンの作成

get コマンドでは、読み取り専用バージョンと編集可能バージョンのファイルが作成されます。アプリケーションでファイル内容の変更が必要でない場合は、読み取り専用バージョンのファイルを使用してください。読み取り専用バージョンのソース・コード・ファイルはコンパイルできます。また、読み取り専用バージョンからテキスト・ファイルを表示したり印刷したりすることができます。

識別キーワードを使用するときには、編集可能バージョンと読み取り専用バージョンの違いが重要です。識別キーワード とは、**get** コマンドによってファイルが読み取り専用として検索されるときに、一種のテキスト値に展開される記号です。編集可能バージョンでは、キーワードは展開されません。識別キーワードは SCCS ファイル内のどこにでも入れることができます。識別キーワードについての詳細は、**prs** コマンドを参照してください。

SCCS ファイル

s. プレフィックス (**s** ファイル) のほかに、 **get** コマンドでは複数の補助ファイル、つまり **g** ファイル、**l** ファイル、**p** ファイル、**z** ファイルが作成されます。これらのファイルはタグで識別されます。このタグとは、ハイフンの前の文字のことです。 **get** プログラムでは、 SCCS ファイル名の先頭の **s.** が正しいタグに置き換えられて、補助ファイルが命名されます。ただし、**g** ファイルの場合は、 **s.** プレフィックスが除去されて命名されます。したがって、ファイル名が **s.sample** であれば、補助ファイル名は **sample**、**l.sample**、 **p.sample**、**z.sample** となります。

これらのファイルの用途は次のとおりです。

項目	説明
s-file	オリジナル・ファイルのテキストと、ファイルに対して行われた変更内容 (デルタ) が定義されています。また、ファイル内容を変更できるユーザー、変更したユーザー、変更時期、変更内容に関する情報も入っています。このファイルは読み取り専用なので、ユーザーが直接編集することはできません。ただし、ユーザーが編集できる **g** ファイルを作成するときに SCCS コマンドに必要となる情報が入っています。
g-file	**-r** フラグで指定した SCCS ファイル・バージョン (または、デフォルトでは最新のトランク・バージョン) のテキストが入っている ASCII テキスト・ファイルです。このファイルはユーザーが直接編集できます。すべての変更を行った後でファイルの新しいデルタを作成する場合は、そのファイルに対して **delta** コマンドを実行できます。 **get** コマンドでは、現行ディレクトリー内に **g** ファイルが作成されます。 **-g** フラグまたは **-p** フラグを指定しなければ、**get** コマンドの実行時には必ず g ファイルが作成されます。ファイルを所有するのは実ユーザーです (実効ユーザーではありません)。 **-k** フラグまたは **-e** フラグを指定しなければ、ファイルは読み取り専用となります。 **-k** フラグまたは **-e** フラグを指定すると、オーナーは **g** ファイルの書き込み許可を持つことになります。 **g** ファイルを作成するには、現行ディレクトリー内への書き込み許可が必要です。

項目	説明
l-file	**-l** フラグを指定すると、**get** コマンドによって **l** ファイルが作成されます。 **l** ファイルは読み取り専用ファイルです。このファイルには、**g** ファイルの生成時に適用されたデルタを示すテーブルが入っています。 **l** ファイルを作成するには、現行ディレクトリー内の書き込み許可が必要です。 **l** ファイル内の行は次のような書式になっています。 • デルタが適用されればブランク文字、適用されなければアスタリスク。 • デルタが適用された場合、または適用されずに無視された場合は、ブランク文字。デルタが適用されず、かつ無視されなかった場合は、アスタリスクが表示されます。 • デルタが適用された特別な理由、または適用されなかった特別な理由を示す次のコード。 *Blankspace* 通常どおり組み込まれたか、または除外された **I** **-i** フラグを使って組み込まれた **X** **-x** フラグを使って除外された **C** **-c** フラグを使ってカットオフされた • SID • ファイルの作成日時 • デルタを作成したユーザーのログイン名 コメントと変更要求 (MR) データは、水平タブ文字の 1 字分字下げされて次行に続きます。ブランク行で各エントリーを終了します。例えば、**-c** フラグを使ったデルタのカットオフの場合、l ファイル内のエントリーは次のようになります。 `**C 1.3 85/03/13 12:44:16 pat` また、最初のデルタに関するエントリーは次のようになります。 `1.1 85/02/27 15:42:20 pat` `date and time created 85/02/27 15:42:20 by pat`
p-file	**-e** フラグまたは **-k** フラグを指定すると、**get** コマンドによって **p** ファイルが作成されます。 **p** ファイルでは、**get -e** コマンドからの結果情報が **delta** コマンドに渡されます。また、それ以降は **delta** コマンドが実行されるか、SCCS ファイル内で結合編集キー文字 (**j**) を設定するまで、**p** ファイルは同じ SID について **get -e** コマンドが実行されないようにします。**j** キー文字を使うと、同じ SID に対して複数の **get** コマンドを実行できます。 **p** ファイルは、SCCS ファイル が入っているディレクトリー内で作成します。 SCCS ディレクトリー内で **p** ファイルを作成するには、そのディレクトリーの書き込み許可が必要です。 **p** ファイルの許可コードは、コードのオーナー以外の全ユーザーに対しては読み取り専用で、実効ユーザーが所有します。 **p** ファイルをオーナーが直接編集することはできません。 **p** ファイルには次の情報が入っています。 • 現行 SID • 新たに作成されるデルタの SID • ユーザー名 • **get** コマンドの日時 • **-i** フラグ (ただし、存在する場合のみ) • **-x** フラグ (ただし、存在する場合のみ) **p** ファイルには、ファイルについて保留中の各デルタに関する優先情報を持つエントリーが入っています。 2 つの行が同一の新しいデルタ SID に入ることはありません。
z-file	**z** ファイルは、同時更新に対するロック機構です。 **z** ファイルには、このファイルを作成した **get** コマンドの 2 進プロセス番号が入っています。このファイルは、SCCS ファイルが入っているディレクトリー内で作成し、**get** コマンドを実行中にだけ存在します。

get コマンドを使用すると、アクセス中の SID と SCCS ファイルから作成された行数が表示されます。 **-e** フラグを指定すると、SID がアクセスされてから行数が作成されるまでに実行されたデルタの SID が表示されます。複数ファイル、ディレクトリー、標準入力のいずれかを指定すると、**get** コマンドによって各ファイルの処理前にファイル名が表示されます。 **-i** フラグを指定すると、**get** コマンドによって、ワー

ド Included の下に組み込まれているデルタが表示されます。**-x** フラグを指定すると、**get** コマンドによって、ワード Excluded の下に排他デルタが表示されます。

次の表に、**get** コマンドによって取り出されるファイルの SID と保留中の SID の両方がどのように判別されるかを示します。「指定された SID」の欄には、**-r** フラグを使って SID を指定するさまざまな方法を示します。最初の欄にはまた、**get -e** コマンドと一緒に **-b** フラグを使用するかどうかを含め、考えられるさまざまな条件を示します。「取り出された SID」の欄には、**g** ファイルを構成するファイルの SID を示してあります。「作成されるデルタの SID」の欄には、**delta** コマンドを適用したときに作成されるバージョンの SID を示してあります。

SID の判別

指定された **SID**	検索された **SID**	作成されるデルタの **SID**
なし [1] **-b** を使用したか**?** no その他の条件 R のデフォルトは mR [2]	mR.mL	mR.(mL+1)
なし [1] **-b** を使用したか**?** はい その他の条件 R のデフォルトは mR	mR.mL	mR.mL.(mB+1).1
R **-b** を使用したか**?** no その他の条件 R>mR	mR.mL	R.1 [3]
R **-b** を使用したか**?** no その他の条件 R=mR	mR.mL	mR.(mL+1)
R **-b** を使用したか**?** はい その他の条件 R>mR	mR.mL	mR.mL.(mB+1).1
R **-b** を使用したか**?** はい その他の条件 R=mR	mR.mL	mR.mL.(mB+1).1
R **-b** を使用したか**?** N/A その他の条件 R<mR で、R は存在しない	hR.mL [4]	hR.mL.(mB+1) .1

SID の判別

指定された **SID**	検索された **SID**	作成されるデルタの **SID**
R **-b** を使用したか**?** N/A その他の条件 リリース内のトランク後続オペレーション > R で R が存在する	R.mL	R.mL.(mB+1).1
R.L. **-b** を使用したか**?** no その他の条件 トランク後続オペレーションなし	R.L.	R.(L+1)
R.L. **-b** を使用したか**?** はい その他の条件 トランク後続オペレーションなし	R.L.	R.L(mB+1).1
R.L. **-b** を使用したか**?** N/A その他の条件 リリース内のトランク後続オペレーション > R または = R	R.L.	R.L.(mB+1).1
R.L.B. **-b** を使用したか**?** no その他の条件 分岐後続オペレーションなし	R.L.B.mS	R.L.B.(mS+1)
R.L.B. **-b** を使用したか**?** はい その他の条件 分岐後続オペレーションなし	R.L.B.mS	R.L.(mB+1).1
R.L.B.S. **-b** を使用したか**?** no その他の条件 分岐後続オペレーションなし	R.L.B.S.	R.L.B.(S+1)
R.L.B.S. **-b** を使用したか**?** はい その他の条件 分岐後続オペレーションなし	R.L.B.S.	R.L.(mB+1).1

SID の判別

指定された **SID**	検索された **SID**	作成されるデルタの **SID**
R.L.B.S. **-b** を使用したか**?** N/A その他の条件 分岐後続オペレーション	R.L.B.S.	R.L.(mB+1).1

注**:** SID 判別テーブルで、文字 R、L、B、および S はそれぞれ SID のリリース、レベル、分岐、シーケンス・コンポーネントです。文字 *m* は最大を表します。

[1] ファイル内に **-d** (デフォルト SID) フラグが存在しない場合にのみ適用されます (**admin** コマンドを参照してください)。

[2] mR は既存の最新のリリースを示します。

[3] 新しいリリース内で最初のデルタが強制的に作成されます。

[4] hR は、指定された、存在しないリリース R よりも前の既存の最新リリースです。

識別キーワード

識別情報は、発生時に識別キーワードがその値に置き換えられることによって、SCCS ファイルから取り出されたテキストに挿入されます。 SCCS ファイルに格納されたテキストには、次のキーワードを使用できます。

キーワード	値
%M%	モジュール名: ファイル内の **m** フラグの値、またはフラグが存在しない場合は、**s.** が除去された SCCS ファイルの名前。
%I%	取り出されたテキストの SCCS 識別コード (SID) (*%R%.%L%* または *%R%.%L%.%B%.%S%*)
%R%	リリース
%L%	レベル
%B%	分岐
%S%	シーケンス
%D%	*YY/MM/DD* フォーマットの現在の日付
%H%	*MM/DD/YY* フォーマットの現在の日付
%T%	*HH:MM:SS* フォーマットの現在の時刻
%E%	最新の適用デルタが作成された *YY/MM/DD* フォーマットの日付
%G%	最新の適用デルタが作成された *MM/DD/YY* フォーマットの日付
01:51:20	最新の適用デルタが作成された *HH:MM:SS* フォーマットの時刻
%Y%	モジュール・タイプ: SCCS ファイル内の **t** フラグの値
%F%	SCCS ファイル名
%P%	SCCS の絶対パス名
%Q%	ファイル内の **-q** フラグの値
%C%	現在行の番号。このキーワードは、「`this should not have happened`」というエラー・メッセージのような、プログラムによって出力されるメッセージを識別するためのものです。**%C%** は、すべての行にシーケンス番号を付けるためのものではありません。
%Z%	*what* による認識が可能な 4 文字の文字列、 `@(#)`
%W%	*what* ストリングを作成するための省略表現: `%W% = %Z%%M%<tab>%I%`
%A%	*what* 文字列を作成するための別の省略表現: `%A% = %Z%%Y% %M% %I%%Z%`

フラグ

項目	説明
-b	作成されるデルタに新しい分岐内の SID を持たせるように指定します。新しい SID には、SID 判別表に示された規則に従って番号が付けられます。**-b** フラグは、**-e** フラグとのみ併用できます。このフラグが必要なのは、リーフ・デルタ (後続オペレーションがないデルタ) から分岐するときだけです。リーフ以外のデルタでデルタを作成しようとすると、**b** ヘッダー・フラグが設定されていなくても、自動的に分岐することになります。SCCS ファイル内で **b** ヘッダー・フラグを指定しなければ、このファイルは分岐できないので、**-b** フラグは **get** コマンドによって無視されます。
-c *Cutoff*	カットオフの日時を *YY[MM[DD[HH[MM[SS]]]]]* のフォーマットで指定します。 **get** コマンドでは、g ファイル内で指定されたカットオフ後に作成された SCCS ファイルには、デルタが組み込まれません。 *Cutoff* 変数に指定されていない項目のデフォルト値は、許容最大値になります。したがって、年 (YY) だけを使ってカットオフ日時を指定すると、その年の最後の月、日、時間、分、秒を指定したことになります。 *Cutoff* 変数の日時の項目を 2 桁ずつ区切るには、数値以外の文字を使用します。これにより、次のようにさまざまな方法で日時を指定できます。 `-c85/9/2,9:00:00` `-c"85/9/2 9:00:00"` `"-c85/9/2 9:00:00"`
-e	作成中の g ファイルが、**get** コマンドを適用するユーザーによって編集されることを示します。変更内容は、後で **delta** コマンドを使って記録します。 **get -e** コマンドは、**delta** コマンドを実行するまでは、ほかのユーザーがさらに **get -e** コマンドを実行して同じ SID の第 2 の g ファイルを編集できないように、p ファイルを作成します。このファイルのオーナーは、**admin** コマンドと **-fj** フラグを使って、同じ SID で共同編集できるようにすることで、この制限を無効にできます。許可を持つほかのユーザーは、**-e** フラグを付けずに **get** コマンドを使うことによって、読み取り専用コピーを作成できます。 **get -e** コマンドでは、SCCS ファイル内の上限、下限、許可ユーザーのリストで指定された SCCS ファイルの保護が強制的に実行されます。 **admin** コマンドを参照してください。 **注: get -e** コマンドで作成した g ファイルを誤って破壊してしまった場合は、**get -k** コマンドでそのファイルを再作成することができます。
-g	**g** ファイルが実際に作成されるのを抑止します。 **-g** フラグは主に、**l** ファイルを作成するときや、特定の SID の存在を検証したりするときに使用します。このフラグを **-e** フラグと併用しないでください。
-i *List*	**g** ファイルの作成時に組み込まれるデルタのリストを指定します。 SID リストのフォーマットは、個々の SID をコンマで区切ったものと、2 つの SID をハイフンで区切って示される SID 範囲との組み合わせです。次の 2 つのコマンド・ラインはいずれも同じ SID を指定します。 `get -e -i1.4,1.5,1.6 s.file` `get -e -i1.4-1.6 s.file` 前述の表の「指定された SID」の欄の書式を使って、デルタの SCCS 識別コードを指定できます。 **get** コマンドは、部分的な SID が「取り出された SID」欄に記載されているとおりに解釈します。
-k	**g** ファイル内の識別キーワードがその値に置き換えられるのを抑止します。 **-k** フラグは **-e** フラグによって暗黙指定されます。 **get -e** コマンドを使って作成した **g** ファイルを誤って破壊してしまった場合は、**-e** フラグの代わりに **-k** フラグを付けて **get** コマンドを再実行すると、 g ファイルを作成し直すことができます。
-l[p]	デルタの要約を l ファイルに書き込みます。 **-lp** を指定すると、デルタの要約は標準出力に書き出され、**get** コマンドによって **l** ファイルは作成されません。このフラグは、現在使用中の g ファイルを作成するときにどのデルタを使ったかを判別するときに使用します。 **l** ファイルのフォーマットについては、**sccsfile** ファイルを参照してください。また、**-L** フラグも参照してください。
-L	デルタの要約を標準出力に書き出します。 **-L** フラグの指定は、**-lp** フラグを使用するのと同じです。
-m	**g** ファイルの各テキスト行の前に、その行を SCCS ファイルに挿入したデルタの SID を書き込みます。フォーマットは次のとおりです。 `SID tab line of text`
-n	**g** ファイル内の各テキスト行の前に **%M%** キーワードの値を書き込みます。書式は、**%M%** の値の後に水平タブが付き、その後にテキスト行が続きます。 **-m** フラグと **-n** フラグの両方を使用すると、書式は次のようになります。 `%M% value tab SID tab line of text`
-p	SCCS ファイルから作成されたテキストを標準出力に書き出しますが、g ファイルを作成しません。 **-s** フラグと **-p** フラグを共に使用しなければ、通常は標準出力に送られるすべての情報出力が、標準エラーに送られます。この場合、通常は標準出力に送られる出力はどこにも表示されません。
-r *SID*	作成される SCCS ファイル・バージョンの SCCS 識別文字列 (SID) を指定します。 SID 判別表には、作成されるファイルのバージョンと、指定された SID の機能として保留中のデルタの SID を示してあります。

項目	説明
-s	通常は標準出力に書き込まれるすべての出力を抑止します。エラー・メッセージ (標準エラー出力に書き込まれる) は、影響を受けません。
-t	指定されたリリース、または指定されたリリースとレベルで最も新しく作成されたデルタにアクセスします。
-w *String*	編集用でない g ファイル内の **%W%** キーワードを *String* 値で置き換えます。
-x *List*	**g** ファイルの作成時にデルタの指定されたリストを除外します。 SID リストの形式については、 **-i** フラグを参照してください。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

次の説明と例は、読み取り専用バージョンのファイルと編集可能バージョンのファイルとの違いを示しています。

1. ファイル内の現在の日付と SID を印刷するには、ファイルに次の記号を入れます。

   ```
   %H% %I%
   ```

   %H% は現在の日付を表す記号で、**%I%** は SID を表す記号です。**get** コマンドでは、ファイルを編集可能バージョンとして取り出すと、記号をファイル内に残して、テキスト値には置き換えません。
2. 次の **get** コマンドの例では、ファイルのバージョンを指定しないので、最新の SID を持つバージョンが作成されます。

   ```
   $ ls
   s.test.c
   $ get s.test.c
   3.5
   59 lines
   $ ls
   s.test.c test.c
   ```
3. 次の 2 つの例では、**-r** フラグを使って、取得されるバージョンを指定します。

   ```
   $ get -r1.3 s.test.c
   1.3
   67 lines

   $ get -r1.3.1.4 s.test.c
   1.3.1.4
   50 lines
   ```
4. SID のリリース番号のみを指定すると、 **get** コマンドによって、そのリリース番号の範囲内で最新のレベルのファイルが検出されます。

   ```
   $ get -r2 s.test.c
   2.7
   21 lines
   ```
5. 指定した SID が既存の最高 SID より大きい場合は、 **get** コマンドによって、既存の最新 SID が取得されます。指定した SID が既存の最低 SID より小さい場合は、SCCS によってエラー・メッセージが書き込まれます。次の例では、リリース 7 が既存の最新リリースです。

   ```
   $ get -r9 s.test.c
   7.6
   400 lines
   ```

6. **-t** フラグによって、指定したリリースまたはレベル内で最上位のバージョンが取得されます。最上位のバージョンとは、位置に関係なく最も新しく作成されたデルタのことです。次の例では、リリース 3 内で既存の最上位デルタは 3.5 で、最も新しく作成されたデルタは 3.2.1.5 です。

   ```
   $ get -t -r3 s.test.c
   3.2.1.5
   46 lines
   ```

7. これまでの例では、 **get** コマンドを使って読み取り専用ファイルを取得しました。編集して新しいデルタの作成に使えるファイルのコピーを作成するには、**get** コマンドで **-e** フラグを使用します。 **get -e** コマンドの効果を取り消して、デルタを作成する前にファイルに対して行った変更内容を廃棄するには、**unget** を使用します。次の例は、**-e** フラグの使用方法を示します。

   ```
   $ ls
   s.test.c
   $ get -e s.test.c
   1.3
   new delta 1.4
   67 lines
   $ ls
   p.test.c s.test.c test.c
   ```

   作業ファイルは test.c です。このファイルを編集し、その変更内容を **delta** コマンドを用いて保存すると、SCCS は SID が 1.4 の新規デルタを作成します。ファイル p.test.c は、ファイル・バージョンをトラックするために SCCS が使用する一時ファイルです。

   前の例では、**-r** フラグを使って特定のバージョンを取得しました。リリース 1 が既存の最新リリースであり、デルタ 1.3 が既に存在し、リリース内の最上位デルタであると想定すると、次の 3 つの **get** コマンドではいずれも同じ処理が実行されます。

   ```
   $ get -e s.test.c
   $ get -e -r1 s.test.c
   $ get -e -r1.3 s.test.c
   ```

8. 新しい (値が最大の) リリース番号を使い始めるには、 **-r** フラグを使ってファイルを取得し、既存の最新リリース番号より大きいリリース番号を指定します。次の例では、リリース 2 がまだ存在していません。

   ```
   $ get -e -r2 s.test.c
   1.3
   new delta 2.1
   67 lines
   ```

   delta コマンドによって SCCS ファイルに変更内容が格納される場合、**get** コマンドによって、作成される新しいデルタのバージョンが示されることに注目してください。

9. 分岐デルタを作成するには、**-r** フラグを使って、分岐を作成するリリースとレベルを指定します。次の例では、デルタ 1.3 と 1.4 が既に存在しています。

   ```
   $ get -e -r1.3 s.test.c
   1.3
   new delta 1.3.1.1
   67 lines
   ```

   これと同じ方法を使って、分岐上でデルタを作成します。

   ファイルを編集するには、**get -e** コマンドを使ってファイル・バージョンを取得し、**delta** コマンドを使って変更内容を保管します。複数の異なる編集可能バージョンの SCCS ファイルを存在させることができますが、その場合は各バージョンが異なるディレクトリー内になければなりません。 **delta** コマンドを使用せずに (**get** コマンドを使って) 同じ編集可能ファイル・バージョンを同じディレクトリーに複数入れようとすると、SCCS によってエラー・メッセージが書き出されます。

同じ編集可能ファイル・バージョンを複数回取得するには、**admin** コマンドを使って SCCS ファイル内に **j** ヘッダー・フラグを設定します。次に、**-f** フラグを使って **j** オプションを設定します。これにより、各 **get** コマンドごとに別々のファイルを作成して、異なるディレクトリーから同じ SID を複数回取得できます。各ファイルの起点は 1 つの SID ですが、SCCS によってそれぞれに固有の新しい SID が提供されます。

10. 次の例では、 **pwd** コマンドによって現行ディレクトリーが表示されます。次に、**admin** コマンドを使って **j** オプションが設定されます。

    注**:** この例でコマンドを実行するためには、両方のディレクトリーへの書き込みアクセス権が必要です。

    ```
    $ pwd
    /home/marty/sccs
    $ admin -fj s.test.c
    ```

11. 次に **get** コマンドを使って、ファイルの最新バージョンを取り出します。

    注**:** この例でコマンドを実行するためには、両方のディレクトリーへの書き込みアクセス権が必要です。

    ```
    $ get -e s.test.c
    1.1
    new delta 1.2
    5 lines
    ```

12. /home/new ディレクトリーに移動して、再び **get** コマンドを実行します。

    注**:** この例でコマンドを実行するためには、両方のディレクトリーへの書き込みアクセス権が必要です。

    ```
    $ cd /home/new
    $ get -e /home/marty/sccs/s.test.c
    1.2
    new delta 1.1.1.1
    5 lines
    ```

    SCCS によって、1 つのオリジナル・ファイル・バージョン 1.1 から、 2 つのデルタ、1.2 と、1.1.1.1 が作成されることに注目してください。**p.test.c** ファイルを確認してください。このファイルは、現在使用中の各バージョンごとに個別のエントリーを表示します。 **delta** コマンドまたは **unget** コマンドを使って両方のファイル・バージョンを処理するまで、**p.test.c** ファイルはそのままディレクトリー内に残ります。

ファイル

項目	説明
/usr/bin/get	**get** コマンドが入っています。

関連情報:

admin コマンド

unget コマンド

what コマンド

ソース・コード制御システム (SCCS) の概要

getconf コマンド

目的

システム構成変数値を標準出力に書き出します。

構文

getconf [**-v** *specification*] [*SystemwideConfiguration* | *PathConfiguration PathName*] [*DeviceVariable DeviceName*]

getconf -a

説明

SystemwideConfiguration パラメーターを指定して起動される **getconf** コマンドは *SystemwideConfiguration* パラメーターに指定された変数の値を標準出力に書き出します。

PathConfiguration パラメーターおよび *Pathname* パラメーターを指定して起動される **getconf** コマンドは *PathName* パラメーターにより指定されたパスにある *PathConfiguration* パラメーターにより指定された変数の値を標準出力に書き出します。

-a フラグを指定して **getconf** コマンドを起動すると、すべてのシステム構成変数が標準出力に書き込まれます。

DeviceVariable および *DeviceName* パラメーターを指定して **getconf** コマンドを起動すると、 *DeviceName* パラメーターによって指定されたデバイス・パスのディスク・デバイス名またはロケーションの値が標準出力に書き込まれます。

指定した変数がシステム上で定義され、その値が **confstr** サブルーチンから使用可能であると記述されている場合、指定変数の値は次の構文で書き込まれます。

```
"%s¥n", <value>
```

それ以外の場合で、指定変数がシステム上で定義されていると、その値は次の構文で書き込まれます。

```
"%d¥n", <value>
```

指定した変数が有効であっても、システム上に定義されていなければ、標準出力に次のように書き出されます。

```
"undefined¥n"
```

変数名が無効であるか、またはエラーが発生すると、診断メッセージが標準エラーに書き出されます。

フラグ

項目	説明
-v *specification*	構成変数を判別する特定の仕様とバージョンを指定します。このフラグを指定しないと、戻される値は、インプリメント・デフォルトの XBS5 準拠コンパイル環境に対応します。
-a	すべてのシステム構成変数の値を標準出力に書き込みます。

パラメーター

項目	説明
PathName	*PathConfiguration* パラメーターのパス名を指定します。
SystemwideConfiguration	システム構成変数を指定します。
PathConfiguration	システム・パス構成変数を指定します。
DeviceName	デバイスのパス名を指定します。
DeviceVariable	デバイス変数を指定します。

次の表の最初の列に表示されるシンボルを **system_var** オペランドとして使用すると、2 番目の列の値と一緒に呼び出したときに **getconf** コマンドによって **confstr** と同じ値が生成されます。

注**: confstr** に対するパラメーターとして使用される **_CS_AIX_ARCHITECTURE** および **_CS_AIX_BOOTDEV** 変数は、 root ユーザーのみに適用されます。

system_var	**confstr** 名前値
BOOT_DEVICE	_CS_AIX_BOOTDEV
MACHINE_ARCHITECTURE	_CS_AIX_ARCHITECTURE
MODEL_CODE	_CS_AIX_MODEL_CODE
PATH	_CS_PATH
XBS5_ILP32_OFF32_CFLAGS	_CS_XBS5_ILP32_OFF32_CFLAGS
XBS5_ILP32_OFF32_LDFLAGS	_CS_XBS5_ILP32_OFF32_LDFLAGS
XBS5_ILP32_OFF32_LIBS	_CS_XBS5_ILP32_OFF32_LIBS
XBS5_ILP32_OFF32_LINTFLAGS	_CS_XBS5_ILP32_OFF32_LINTFLAGS
XBS5_ILP32_OFFBIG_CFLAGS	_CS_XBS5_ILP32_OFFBIG_CFLAGS
XBS5_ILP32_OFFBIG_LDFLAGS	_CS_XBS5_ILP32_OFFBIG_LDFLAGS
XBS5_ILP32_OFFBIG_LIBS	_CS_XBS5_ILP32_OFFBIG_LIBS
XBS5_ILP32_OFFBIG_LINTFLAGS	_CS_XBS5_ILPBIG_OFF32_LINTFLAGS
XBS5_LP64_OFF64_CFLAGS	_CS_XBS5_LP64_OFF64_CFLAGS
XBS5_LP64_OFF64_LDFLAGS	_CS_XBS5_LP64_OFF64_LDFLAGS
XBS5_LP64_OFF64_LIBS	_CS_XBS5_LP64_OFF64_LIBS
XBS5_LP64_OFF64_LINTFLAGS	_CS_XBS5_LP64_OFF64_LINTFLAGS
XBS5_LPBIG_OFFBIG_CFLAGS	_CS_XBS5_LPBIG_OFFBIG_CFLAGS
XBS5_LPBIG_OFFBIG_LDFLAGS	_CS_XBS5_LPBIG_OFFBIG_LDFLAGS
XBS5_LPBIG_OFFBIG_LIBS	_CS_XBS5_LPBIG_OFFBIG_LIBS
XBS5_LPBIG_OFFBIG_LINTFLAGS	_CS_XBS5_LPBIG_OFFBIG_LINTFLAGS

環境変数

次に示す環境変数は、**getconf** コマンドの実行に影響を与えます。

項目	説明
LANG	設定されていない、または null である国際化対応変数のデフォルト値を指定します。 LANG を設定しなかった場合、または null の場合は、インプリメント依存またはデフォルトのロケールから対応する値が使用されます。国際化対応変数のいずれかに無効な値が設定されると、ユーティリティーは、変数が 1 つも定義されていないものとして動作します。
LC_CALL	空文字列以外の値に設定する場合は、ほかのすべての国際化対応変数の値を指定変更してください。
LC_CTYPE	テキスト・データのバイト・シーケンスを文字として解釈するのに使用するロケールを決めます (例えば、引数の複数バイト文字に対する単一バイト文字)。
LC_MESSAGES	標準エラーに書き出される診断メッセージの書式と内容に影響を与えるロケールを決めます。
NLSPATH	LC_MESSAGES の処理に使用するメッセージ・カタログの位置を指定します。

全システム構成変数

SystemwideConfiguration パラメーターは、システム全体で有効なシステム構成変数を指定します。システム構成変数の値には次の 2 種類があります。

- 全システム構成変数
- システム標準構成変数

全システム構成変数

全システム構成変数には、システムのあらゆる部分を満たす最小値が入っています。次に示すリストに、**getconf** コマンドと一緒に使用する全システム構成変数の定義を示します。

項目	説明
_CS_PATH	コマンドの検索に使用する **PATH** 環境変数の値です。
ARG_MAX	環境データを含め、**exec** サブルーチンの 1 つの引数の最大バイト数です。
BC_BASE_MAX	**bc** コマンドで **obase** 変数に指定できる最大値です。
BC_DIM_MAX	**bc** コマンドで配列に指定できるエレメントの最大数です。
BC_SCALE_MAX	**bc** コマンドで **scale** 変数に指定できる最大値です。
BC_STRING_MAX	**bc** コマンドに指定できる文字列定数の最大長です。
CHARCLASS_NAME_MAX	文字クラス名の最大バイト数です。
CHAR_BIT	タイプ文字内のビット数です。
CHAR_MAX	タイプ文字の最大値です。
CHAR_MIN	タイプ文字の最小値です。
CHILD_MAX	各実ユーザー ID の同時プロセスの最大数です。
CLK_TCK	**time** サブルーチンから戻される 1 秒当たりのクロック・ティックです。
COLL_WEIGHTS_MAX	ロケール定義ファイルの **LC_COLLATE** ロケール・スタンザのエントリーに割り当てられた重みの最大数です。
CS_PATH	コマンドの検索に使用する **PATH** 環境変数の値です。
EXPR_NEST_MAX	**expr** コマンドによって、括弧内にネストできる式の最大数です。
INT_MAX	**int** 型の最大値です。
INT_MIN	**int** 型の最小値です。
LINE_MAX	ユーティリティーがテキスト・ファイル処理中と表示されたときの、コマンド入力行 (標準入力、またはその他のファイル) の最大長 (バイト単位) です。この長さには、後続の改行文字を入れる余地が含まれます。
LONG_BIT	**long int** 型のビット数です。
LONG_MAX	**long int** 型の最大値です。
LONG_MIN	**long int** 型の最小値です。
MB_LEN_MAX	サポートされるロケールの文字内の最大バイト数です。
NGROUPS_MAX	各プロセスの同時的な 2 次的グループ ID の最大数です。
NL_ARGMAX	**printf** および **scanf** サブルーチンの呼び出しに使える最大桁です。
NL_LANGMAX	LANG 名に使える最大バイト数です。
NL_MSGMAX	最大メッセージ数です。
NL_NMAX	N 対 1 の照合マッピングに使える最大バイト数です。
NL_SETMAX	最大セット数です。
NL_TEXTMAX	メッセージ文字列に使える最大バイト数です。

項目	説明
NZERO	デフォルトのプロセス優先順位です。
OPEN_MAX	1 つのプロセスが一度にオープンできる最大ファイル数です。
PATH	コマンドの検索に使用されるコロンで区切られたパス・プレフィックスのシーケンスです。
RE_DUP_MAX	**ed** コマンドと一緒に、 *m* パラメーターおよび *n* パラメーターなどの、インターバル表記パラメーターを使用するときの、正規表現の反復回数の許容最大数です。
SCHAR_MAX	**signed char** 型の最大値です。
SCHAR_MIN	**signed char** 型の最小値です。
SHRT_MAX	**short** 型の最大値です。
SHRT_MIN	**short** 型の最小値です。
SSIZE_MAX	**ssize_t** 型のオブジェクトの最大値です。
STREAM_MAX	1 つのプロセスが同時にオープンできるストリーム数です。
TMP_MAX	**tmpnam** サブルーチンにより生成される固有のパス名の最小数です。アプリケーションが **tmpnam** サブルーチンを確実に呼び出すことができる最大回数です。
TZNAME_MAX	タイムゾーンの名前に使用できる最大バイト数です (**TZ** 環境変数の長さではありません)。
UCHAR_MAX	**unsigned char** 型の最大値です。
UINT_MAX	**unsigned int** 型の最大値です。
ULONG_MAX	**unsigned long int** 型の最大値です。
USHRT_MAX	**unsigned short int** 型の最大値です。
WORD_BIT	ワードまたは **int** 型のビット数です。
KERNEL_BITMODE	カーネルのビット・モードで、32 ビットまたは 64 ビットです。
REAL_MEMORY	実メモリー・サイズです。
HARDWARE_BITMODE	マシン・ハードウェアのビット・モードで、32 ビットまたは 64 ビットです。
MP_CAPABLE	マシンの MP 機能です。

システム標準構成変数

システム標準構成変数には、特定のシステム標準に必要な最小 値が入っています。プレフィックス **_POSIX_**、**POSIX2_**、および **_XOPEN_** は、それぞれ POSIX 1003.1、POSIX 1003.2、および X/OPEN システム標準に必要なシステム特性の最小値が変数に入っていることを示します。システム標準は、特定のシステム標準をサポートするためのシステム全体に渡る最小値です。実際の構成値がこれらの標準を超える場合があります。 **getconf** コマンド用のシステム標準構成変数の定義は次のとおりです。

項目	説明
_POSIX_ARG_MAX	環境データを含め、**exec** サブルーチンの 1 つの引数の最大バイト数です。
_POSIX_CHILD_MAX	各実ユーザー ID の同時プロセスの最大数です。
_POSIX_JOB_CONTROL	システムがジョブ制御をサポートする場合は、値 1 です。
_POSIX_LINK_MAX	1 つのファイルへの最大リンク数です。
_POSIX_MAX_CANON	端末正規入力キュー内の最大バイト数です。
_POSIX_MAX_INPUT	端末入力キュー内の許容最大バイト数です。
_POSIX_NAME_MAX	ファイル名の最大バイト数です (最後の null を除く)。
_POSIX_NGROUPS_MAX	各プロセスの同時的な 2 次的グループ ID の最大数です。
_POSIX_OPEN_MAX	1 つのプロセスが一度にオープンできる最大ファイル数です。
_POSIX_PATH_MAX	パス名の最大バイト数です。
_POSIX_PIPE_BUF	パイプへの書き込み時にアトミックであることが保証される最大バイト数です。
_POSIX_SAVED_IDS	値 1 です。各プロセスのセット・ユーザー ID とセット・グループ ID が保存されています。
_POSIX_SSIZE_MAX	**ssize_t** 型のオブジェクト内に格納できる最大値です。
_POSIX_STREAM_MAX	1 つのプロセスが同時にオープンできるストリーム数です。
_POSIX_TIMESTAMP_RESOLUTION	ナノ秒単位でのすべてのファイル・タイム・スタンプのレゾリューション。
_POSIX_TZNAME_MAX	タイムゾーンの名前に使用できる最大バイト数です (**TZ** 環境変数の長さではありません)。
_POSIX_VERSION	オペレーティング・システムが準拠する POSIX 1 標準 (C 言語バインディング) のバージョンです。
_XOPEN_CRYPT	システムが X/Open Encryption Feature Group をサポートする場合は、値 1 です。

項目	説明
_XOPEN_ENH_I18N	システムが X/Open Enhanced Internationalisation Feature Group をサポートする場合は、値 1 です。
_XOPEN_SHM	システムが X/Open Shared Memory Feature Group をサポートする場合は、値 1 です。
_XOPEN_VERSION	オペレーティング・システムが準拠する X/Open Portability Guide のバージョンです。
_XOPEN_XCU_VERSION	オペレーティング・システムが準拠する X/Open Commands and Utilities 仕様のバージョンです。
_XOPEN_XPG2	システムが X/Open Portability Guide, Volume 2, (1987 年 1 月) の XVS System Calls and Libraries をサポートする場合の値は 1 で、それ以外の場合は未定義です。
_XOPEN_XPG3	システムが X/Open Specification (1992 年 2 月) System Interfaces and Headers, Issue 3 をサポートする場合の値は 1 で、それ以外の場合は未定義です。
_XOPEN_XPG4	システムが X/Open CAE Specification (1992 年 7 月) の System Interfaces and Headers, Issue 4 をサポートする場合の値は 1 で、それ以外の場合は未定義です。
POSIX2_BC_BASE_MAX	**bc** コマンドで **obase** 変数に指定できる最大値です。
POSIX2_BC_DIM_MAX	**bc** コマンドで配列に指定できるエレメントの最大数です。
POSIX2_BC_SCALE_MAX	**bc** コマンドで **scale** 変数に指定できる最大値です。
POSIX2_BC_STRING_MAX	**bc** コマンドに指定できる文字列定数の最大長です。
POSIX2_CHAR_TERM	システムが端末タイプを少なくとも 1 つはサポートする場合の値は 1、それ以外の場合の値は -1 です。
POSIX2_COLL_WEIGHTS_MAX	ロケール定義ファイル内の **LC_COLLATE** ロケール変数のエントリーに割り当て可能な重みの最大数です。
POSIX2_C_BIND	システムが POSIX 2 の C Language Binding Option をサポートする場合の値は 1 で、それ以外の場合は -1 です。
POSIX2_C_DEV	システムが POSIX 2 の C Language Development Utilities をサポートする場合の値は 1、それ以外の場合の値は -1 です。
POSIX2_C_VERSION	オペレーティング・システムが準拠する POSIX 2 標準 (C 言語バインディング) のバージョンです。
POSIX2_EXPR_NEST_MAX	**expr** コマンドによって、括弧内にネストできる式の最大数です。
POSIX2_FORT_DEV	システムが POSIX 2 の FORTRAN Development Utilities Option をサポートする場合の値は 1 で、それ以外の場合は -1 です。
POSIX2_FORT_RUN	システムが POSIX 2 の FORTRAN Runtime Utilities Option をサポートする場合の値は 1 で、それ以外の場合は -1 です。
POSIX2_LINE_MAX	コマンドがテキスト・ファイルを処理中として記述されたときの、コマンドの入力行 (標準入力または別ファイル) の最大長 (バイト単位) です。この長さには、後続の改行文字を入れる余地が含まれます。
POSIX2_LOCALEDEF	システムが **localedef** コマンドによるロケールの作成をサポートする場合の値は 1、それ以外の場合は未定義です。
POSIX2_RE_DUP_MAX	**ed** コマンドと一緒に、 *m* パラメーターおよび *n* パラメーターなどの、インターバル表記パラメーターを使用するときの、正規表現の反復回数の許容最大数です。
POSIX2_SW_DEV	システムが Software Development Utilities Option をサポートする場合の値は 1、それ以外の場合の値は -1 です。
POSIX2_UPE	システムが POSIX 2 の User Portability Utilities Option をサポートする場合の値は 1 で、それ以外の場合は -1 です。
POSIX2_VERSION	システムがサポートする最新バージョンの POSIX 2 標準の承認日です。日付は 6 桁の数で、最初の 4 桁は年、最後の 2 桁は月を示します。さまざまなバージョンの POSIX 2 標準が IEEE 標準化委員会によって定期的に承認され、承認日は各バージョンを区別するために使用されます。

システム・パス構成変数

PathConfiguration パラメーターは、システム内のパスとパス構造の情報が値に含まれているシステム・パス構成変数を指定します。これらの変数の定義は次のとおりです。

項目	説明
_POSIX_CHOWN_RESTRICTED	**chown** () サブルーチンの実行は、該当する特権を持つプロセスに対して、またファイルのグループ ID をプロセスの実効グループ ID またはその補足グループ ID のいずれかに変更することに対して限定されます。 *PathName* パラメーターがディレクトリーを参照する場合、戻される値は、既存のディレクトリー、またはディレクトリー内で作成可能なディレクトリー以外の、任意のファイルに適用されます。
_POSIX_NO_TRUNC	パス名が *NAME_MAX* 変数により指定された制限を超えると、エラーが生成されます。*PathName* パラメーターがディレクトリーを参照する場合、戻される値はそのディレクトリー内のファイル名に適用されます。
_POSIX_VDISABLE	**termios.h** ファイルに定義された端末特殊文字は、ここで指定する文字値によって使用不可にすることができます。
LINK_MAX	1 つのファイルへの最大リンク数です。*PathName* パラメーターがディレクトリーを参照する場合、戻される値はそのディレクトリーに適用されます。
MAX_CANON	端末正規入力待ち行列内の最大バイト数です。
MAX_INPUT	端末入力キュー内で使用できる空間の最大バイト数です。
NAME_MAX	ファイル名の最大バイト数です (最後の null を除く)。*PathName* パラメーターがディレクトリーを参照する場合、戻される値はそのディレクトリー内のファイル名に適用されます。
PATH_MAX	最後の null 文字を含めた、パス名の最大バイト数です。*PathName* パラメーターがディレクトリーを参照する場合、指定したディレクトリーが作業ディレクトリーであるときは、戻される値は相対パス名の最大長です。
PIPE_BUF	パイプへの書き込み時にアトミックであることが保証される最大バイト数です。*PathName* パラメーターが FIFO またはパイプを参照する場合、戻される値は参照先オブジェクトに適用されます。 *PathName* パラメーターがディレクトリーを参照する場合、戻される値は既存の FIFO、またはそのディレクトリー内で作成できる FIFO に適用されます。
DISK_PARTITION	ディスクの物理区画サイズです。 **注: DISK_PARTITION** パス構成変数を指定する場合は、情報を照会するディスクの完全なパスを *PathName* パラメーターで指定する必要があります。
DISK_SIZE	ディスク・サイズを MB で指定します。 **注: DISK_SIZE** パス構成変数を指定する場合は、情報を照会するディスクの完全なパスを *PathName* パラメーターで指定する必要があります。

デバイス変数

DeviceVariable パラメーターは、*DeviceName* パラメーターが **/dev/hdisk0** のようなデバイスのパスであることを示します。ディスクのパスを指定すると、**getconf** コマンドは、ディスクのデバイス名またはロケーションを表示します。

項目	説明
DISK_DEVNAME	デバイス名またはデバイスのロケーションです。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	指定された変数が有効で、現在の状態に関する情報が正常に書き込まれました。
>0	エラーが発生しました。

例

1. **ARG_MAX** 変数の値を表示するには、次のコマンドを入力します。

   ```
   getconf ARG_MAX
   ```

2. **/usr** ディレクトリーの **NAME_MAX** 変数の値を表示するには、次のコマンドを入力します。

   ```
   getconf NAME_MAX /usr
   ```

3. 次のシェル・コマンドのシーケンスは、未指定の戻り値をどのように処理するかを示しています。

   ```
   if value=$(getconf PATH_MAX /usr)
   then
        if [ "$value" = "undefined" ]
        then
                echo
                      The value of PATH_MAX in /usr is undefined.
        else
                echo
                      The value of PATH_MAX in /usr is $value.
        fi
   else
        echo Error in the getconf command.
   fi
   ```

4. コマンドを次のように入力したとします。

   ```
   getconf _XBS5_ILP32_OFF32
   ```

   この場合、-1¥n、または undefined¥n は標準出力に書き出されないので、コマンドを次のような書式にします。

   ```
   getconf -v XBS5_ILP32_OFF32 ...
   ```

   このコマンドは、拡張記述である **c89** に指定した、 XBS5_ILP32_OFF32 コンパイル環境に対応する、構成変数の値を判別します。

5. コマンドを次のように入力したとします。

   ```
   getconf _XBS5_ILP32_OFFBIG
   ```

   この場合、-1¥n、または undefined¥n は標準出力に書き出されないので、コマンドを次のような書式にします。

   ```
   getconf -v XBS5_ILP32_OFFBIG ...
   ```

   このコマンドは、拡張記述である **c89** に指定した、 XBS5_ILP32_OFFBIG コンパイル環境に対応する、構成変数の値を判別します。

6. コマンドを次のように入力したとします。

   ```
   getconf _XBS5_LP64_OFF64
   ```

   この場合、-1¥n、または undefined¥n は標準出力に書き出されないので、コマンドを次のような書式にします。

   ```
   getconf -v XBS5_LP64_OFF64 ...
   ```

   このコマンドは、拡張記述である c89 に指定した XBS5_LP64_OFF64 コンパイル環境に対応する構成変数の値を判別します。

7. コマンドを次のように入力したとします。

   ```
   getconf _XBS5_LPBIG_OFFBIG
   ```

   この場合、-1¥n、または undefined¥n は標準出力に書き出されないので、コマンドを次のような書式にします。

   ```
   getconf -v _XBS5_LPBIG_OFFBIG
   ```

   このコマンドは、拡張記述である c89 に指定した XBS5_LPBIG_OFFBIG コンパイル環境に対応する構成変数の値を判別します。
8. root ユーザーとして、ディスク hdisk0 のディスク・サイズを判別するには、次のコマンドを入力します。

   ```
   getconf DISK_SIZE /dev/hdisk0
   ```
9. 実メモリー・サイズを判別するには、次のコマンドを入力します。

   ```
   getconf REAL_MEMORY
   ```
10. マシン・ハードウェアが 32 ビットであるか 64 ビットであるかを判別するには、次のコマンドを入力します。

    ```
    getconf HARDWARE_BITMODE
    ```
11. カーネルが 32 ビットであるか 64 ビットであるかを判別するには、次のコマンドを入力します。

    ```
    getconf KERNEL_BITMODE
    ```
12. ディスク hdisk0 のデバイス名またはロケーションを判別するには、次のコマンドを入力します。

    ```
    getconf DISK_DEVNAME hdisk0
    ```

ファイル

項目	説明
/usr/bin/getconf	**getconf** コマンドが入っています。
/usr/include/limits.h	システム構成変数を定義します。
/usr/include/unistd.h	システム構成変数を定義します。

関連情報:

confstr コマンド

pathconf コマンド

sysconf コマンド

Commands コマンド

---

getdev コマンド

目的

指定した基準に一致するデバイスをリストします。

構文

getdev [**-a**] [**-e**] [*Criteria*] [*DeviceList*]

説明

指定した基準に一致するデバイスをリストします。基準は、式の形式で指定します。 **getdev** コマンドは、システム上のすべてのデバイスまたは指定したリストのデバイスを検査することができます。

フラグ

項目	説明
-a	すべての基準に一致するデバイスが、このコマンドによって生成されるリストに組み込まれることを指定します。基準が定義されていない場合には、**-a** フラグは無効になります。
-e	*devicelist* に指定されているデバイスを **getdev** コマンドにより生成されるリストから除外することを指定します。*-e* フラグを指定しないと、*devicelist* 内のデバイスのみが生成されます。このフラグは、デバイスが指定されていない場合には無視されます。

パラメーター

項目	説明
Criteria	デバイスが、生成されるリストに組み込まれる前に、一致していなければならない基準を定義します。*Criteria* は、式または式のリストとして指定することができます。この基準に一致するデバイスが、**getdev** により生成されるリストに組み込まれます。基準が指定されていない場合は、すべてのデバイスがリストに組み込まれます。 デバイスは、リスト内の基準のうち少なくとも 1 つは満たしている必要があります。ただし、 **-a** オプションを使用して、「論理 AND」操作を行うように指定することができます。このオプションを指定すると、リスト内のすべての基準に一致するデバイスのみが結果リストに組み込まれます。 *Criteria* パラメーターで指定される基準には、次の 4 つのタイプの式があります。 *Attribute* **=** *Value* 定義された *Attribute* を持っており、それが *Value* に等しいメンバーを持つデバイスをすべて取り出します。 *Attribute***!=***Value* 定義された *Attribute* を持っており、それが *Value* に等しくないメンバーを持つデバイスをすべて取り出します。 *Attribute***:*** 定義された *Attribute* を持っているメンバーを持つデバイスをすべて取り出します。 *Attribute***!:*** 定義された *Attribute* を持っていないメンバーを持つデバイスをすべて取り出します。 有効なデバイス属性は、次のとおりです。 **alias** デバイスが認識されている名前。 **desc** デバイスの説明。 **type** デバイスのタイプを示すトークン。**type** 属性の有効な値のセットは、次のコマンドを実行して取得できます。 **odmget PdDv | grep -w class | awk '{print $3}' | sed 's/"//g' | sort | uniq** **status** デバイスの現在の状態。可能な状態の値は、1.Defined (定義済み) 2.Available (使用可能) 3.Stopped (停止) 4.Diagnose (診断) です。**status** の値は大文字と小文字が区別されません。
DeviceList	*Criteria* を検査するデバイスがスペースで区切られたリストを指定します。

終了状況

0 コマンドは正常に完了しました。

> 1 故障が発生しました。

例

1. すべてのデバイスを表示するには、次のように入力します。

```
getdev
```

2. タイプが「logical_volume」であるデバイスをリストするには、次のように入力します。

```
getdev type=logical_volume
```

3. タイプが「logical_volume」でないデバイスをリストするには、次のように入力します。

   ```
   getdev type!=logical_volume
   ```

4. タイプが「logical_volume」であるか、または別名が「sys0」であるデバイスをリストするには、次のように入力します。

   ```
   getdev type=logical_volume alias=sys0
   ```

   出力は次のようになります。

   ```
   hd1
   hd2
   hd3
   hd4
   ...
   sys0
   ```

5. タイプが「logical_volume」で、別名が「lv01」であるデバイスをリストするには、次のように入力します。

   ```
   getdev -a type=logical_volume alias=lv01
   ```

6. **status** 属性が定義されているデバイスを表示するには、次のように入力します。

   ```
   getdev status:*
   ```

7. **desc** 属性が定義されていないデバイスを表示するには、次のように入力します。

   ```
   getdev desc!:*
   ```

ファイル

項目	説明
/usr/sbin/getdev	**getdev** コマンドが入っています。

関連資料:

『getdgrp コマンド』

---

getdgrp コマンド

目的

指定した基準に一致するデバイス・クラスをリストします。

構文

getdgrp [**-a**] [**-e**] [**-l**][*Criteria*] [*DeviceClassList*]

説明

指定した基準に一致するデバイスが含まれているデバイス・クラスをリストします。基準は、式の形式で指定します。

フラグ

項目	説明
-a	すべての基準に一致するデバイスが含まれているデバイス・クラスが、このコマンドによって生成されるレポートに組み込まれるように指示します。基準が定義されていない場合には、**-a** フラグは無効になります。
-e	パラメーター・リストに指定されているデバイス・クラスを、このコマンドによって生成されるレポートから除外するように指示します。デバイスが指定されていない場合には、 **-e** フラグは無効になります。
-l	デバイス・クラスに有効なデバイス・メンバーが含まれていない場合でも、 **-e** オプションおよび **dgroup** リストに該当するすべてのデバイス・クラスをリストするように指示します。このオプションは、コマンド・ラインに *Criteria* が指定されている場合には、無効になります。

パラメーター

項目	説明
Criteria	デバイスが属しているデバイス・クラスが、生成されるリストに組み込まれる前に、デバイスが一致していなければならない基準を定義します。 *Criteria* は、式または式のリストとして指定することができます。この基準に一致するデバイスが含まれているデバイス・クラスが、**getdgrp** により生成されるリストに組み込まれます。基準が指定されていない場合は、すべてのデバイス・クラスがリストに組み込まれます。 デバイスは、リスト内の基準のうち少なくとも 1 つは満たしている必要があります。ただし、 **-a** オプションを使用して、「論理 AND」操作を行うように指定することができます。このオプションを指定すると、リスト内のすべての基準に一致するデバイスが含まれているデバイス・クラスのみが結果リストに組み込まれます。 *Criteria* パラメーターで指定される基準には、次の 4 つのタイプの式があります。 *Attribute* **=** *Value* 定義された *Attribute* を持っており、それが *Value* に等しいメンバーが含まれているデバイス・クラスをすべて取り出します。 *Attribute***!=***Value* 定義された *Attribute* を持っており、それが *Value* に等しくないメンバーが含まれているデバイス・クラスをすべて取り出します。 *Attribute***:*** 定義された *Attribute* を持っているメンバーが含まれているデバイス・クラスをすべて取り出します。 *Attribute***!:*** 定義された *Attribute* を持っていないメンバーが含まれているデバイス・クラスをすべて取り出します。 有効なデバイス属性は、次のとおりです。 **alias** デバイスが認識されている名前。 **desc** デバイスの説明。 **type** デバイスのタイプを示すトークン。 **status** デバイスの現在の状態。可能な状態の値は、1.Defined (定義済み) 2.Available (使用可能) 3.Stopped (停止) 4.Diagnose (診断) です。**status** の値は大文字と小文字が区別されません。
DeviceClassList	カスタマイズ・デバイス構成データベースまたは事前定義デバイス構成データベース内のデバイス・クラス名を指定します。

終了状況

0 コマンドは正常に完了しました。

1 コマンド構文が間違っています。無効なオプションが使用されたか、または内部エラーが発生しました。

2 カスタマイズ・デバイス・オブジェクト・クラスまたは事前定義デバイス・オブジェクト・クラスを読み取りのためにオープンできませんでした。

例

1. すべてのデバイス・クラスを表示するには、次のように入力します。

   ```
   getdgrp
   ```

   出力は、次のようになります。

   ```
   adapter
   aio
   bus
   cdrom
   disk
   diskette
   gxme
   if
   keyboard
   lft
   logical_volume
   lvm
   memory
   mouse
   planar
   processor
   pty
   pwrmgt
   rcm
   sys
   tape
   tcpip
   tty
   ```

2. タイプが「logical_volume」であるデバイスが含まれているデバイス・クラスをリストするには、次のように入力します。

   ```
   getdgrp type=logical_volume
   ```

   出力は次のようになります。

   ```
   logical_volume
   ```

3. タイプが「logical_volume」であるか、または別名が「sys0」であるデバイスが含まれているデバイス・クラスをリストするには、次のように入力します。

   ```
   getdgrp type=logical_volume alias=sys0
   ```

   出力は次のようになります。

   ```
   logical_volume
   sys
   ```

4. status 属性が定義されているデバイスが含まれているデバイス・クラスをリストするには、次のように入力します。

   ```
   getdgrp status=defined
   ```

   出力は次のようになります。

   ```
   logical_volume
   posix_aio
   rcm
   ```

5. **status** 属性が定義されており、「processor」デバイス・クラスに属しているデバイスが含まれているデバイス・クラスを表示するには、次のように入力します。

   ```
   getdgrp status:* processor
   ```

   出力は次のようになります。

```
processor
```

6. **status** 属性が定義されていないデバイスが含まれているデバイス・クラスを表示するには、次のように入力します。

```
getdgrp status!:* processor
```

ファイル

項目	説明
/usr/sbin/getdgrp	**getdgrp** コマンドが入っています。

関連資料:

711 ページの『getdev コマンド』

---

getea コマンド

目的

名前付きの拡張属性をファイルから検索します。

構文

getea [**-n** *Name*] [**-l**] [**-e** *RegExp*] [**-s**] *FileName*

説明

getea コマンドは、名前付きの拡張属性をファイルから読み取ります。 **-n** *Name* パラメーターを指定すると、*Name* に一致する拡張属性だけが検索されます。

注**:** 命名の衝突を防ぐために、JFS2 では、システム定義の拡張属性用に 8 文字のプレフィックス (0xf8)SYSTEM(0xF8) が予約されています。ユーザー定義の拡張属性の命名には、このプレフィックスを使用しないでください。

-e *RegExp* パラメーターを指定すると、正規表現 *RegExp* に一致する拡張属性だけが検索されます。 **-n** または **-e** のフラグをどちらも指定しないと、すべての拡張属性が検索されます。

このコマンドは ACL を取得するためには使用されません。 ACL を取得するには **aclget** コマンドを使用します。

フラグ

項目	説明
-e *RegExp*	正規表現を指定して、それに一致するすべての拡張属性を検索するように指示します。値は文字フォーマットで表示されます。
-l	指し示しているファイルからではなく、シンボリック・リンクそのものから拡張属性を取得することを指定します。
-n *Name*	検索する特定の拡張属性の名前を指定します。値は文字フォーマットで表示されます。
-s	拡張属性の名前だけを表示し、値は表示しません。
FileName	拡張属性の読み取り元となるファイルを指定します。

終了状況

項目	説明
0	正常終了。
正の整数	エラーが発生しました。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. ファイル `design.html` のすべての名前付き拡張属性を検索するには、次のように入力します。

   ```
   getea design.html
   ```

2. `design.html` ファイルについて、名前付き拡張属性 `Approver` を検索するには、次のように入力します。

   ```
   getea -n Approver design.html
   ```

3. ファイル `design.html` について、すべての名前付き拡張属性の名前だけを検索するには、次のように入力します。

   ```
   getea -s design.html
   ```

4. シンボリック・リンク `design.html` の名前付き拡張属性をすべて検索するには、次のように入力します。

   ```
   getea -l design.html
   ```

位置

/usr/sbin

関連情報:

chfs コマンド

crfs コマンド

setea コマンド

---

getopt コマンド

目的

コマンド・ラインのフラグとパラメーターを解析します。

構文

getopt *Format Tokens*

説明

getopt コマンドは、期待されたフラグと引数を使用してトークンのリストを解析します。フラグは単一の ASCII 文字で、後ろに : (コロン) が付いている場合は、引数が付いているものと見なされます。各引数は、 1 つ以上のタブかスペースで区切ることができますが、必ずしもその必要はありません。引数に使うことはできますが、フラグ文字には使えません。

getopt コマンドは、すべてのトークンを読み取るとき、または特殊トークン — (二重ハイフン) を検出すると、処理を完了します。次に、**getopt** コマンドは処理したフラグ、 — (二重ハイフン)、および残りのトークンを出力します。

トークンがフラグと一致しなければ、**getopt** コマンドはメッセージを標準エラーに書き出します。

例

getopt コマンドをスケルトン・シェル・スクリプトで使用すると、次の例のようにオプションを解析することができます。

```
#!/usr/bin/bsh
# parse command line into arguments
set -- `getopt a:bc $*`
# check result of parsing
if [ $? != 0 ]
then
        exit 1
fi
while [ $1 != -- ]
do
        case $1 in
        -a)     # set up the -a flag
                AFLG=1
                AARG=$2
                shift;;
        -b)     # set up the -b flag
                BFLG=1;;
        -c)     # set up the -c flag
                CFLG=1;;
        esac
        shift   # next flag
done
shift   # skip --
# now do the work
.
.
.
```

注**:** C シェルで、**getopt** コマンドを実行する場合は、次のコマンドを使用してください。

```
set argv=`getopt OptionString $*`
```

次の各例で、**getopt** コマンドはフラグと引数を同じように処理します。

- `-a ARG -b -c`
- `-a ARG -bc`
- `-aARG -b -c`
- `-b -c -a ARG`

ファイル

項目	説明
/usr/bin/getopt	**getopt** コマンドが入っています。

関連情報:

bsh コマンド

csh コマンド

getopt コマンド

シェル・コマンド

---

getopts コマンド

目的

コマンド・ラインの引数を処理し、有効なオプションを検査します。

構文

getopts *OptionString Name* [*Argument ...*]

説明

getopts コマンドは、Korn/POSIX シェル組み込みコマンドの 1 つで、パラメーターのリストからオプションとオプション引数を検索します。オプションは、**+** (正符号) または **-** (負符号) で始まり、その後ろに 1 文字が続きます。 **+** または **-** で始まらないオプションは、*OptionString* を終了します。 **getopts** コマンドは、起動されるたびに、次のオプションの値を *Name* に設定し、処理対象の次の引数のインデックスをシェル変数 **OPTIND** に設定します。シェルが起動されるたびに、**OPTIND** は 1 に初期化されます。オプションが **+** で始まる場合には、**+** が *Name* の値の前に付加されます。

OptionString の文字の後ろに **:** (コロン) を付けると、そのオプションが引数を持つと想定されます。オプションにオプション引数が必要な場合、**getopts** コマンドでは、オプション引数を変数 **OPTARG** に設定します。

OptionString に含まれていないオプション文字が見つかった場合、あるいは見つかったオプションに必要なオプション引数が存在しない場合には、次の処理が行われます。

- *OptionString* が、 **:** (コロン) で始まらない場合
  - *Name* が不明のオプションを表す **?** (疑問符) 文字に設定されます。
  - **OPTARG** が設定解除されます。
  - 診断メッセージが標準エラーに書き出されます。

この状態は、引数が起動側アプリケーションに提供されたという形で検出されたエラーと考えられますが、**getopts** コマンドの処理上のエラーではありません。上記に述べたように、診断メッセージが書き込まれますが、終了状況はゼロとなります。

- *OptionString* が、 **:** (コロン) で始まる場合
  - *Name* が不明のオプションを表す **?** (疑問符) 文字または必須オプションの欠落を表す **:** (コロン) 文字に設定されます。
  - **OPTARG** は、見つかったオプションに設定されます。
  - 標準エラーには何も出力されません。

オプションが終了するのは、特殊なオプション **--** が指定されたとき、**-** または **+** で始まらない引数が検出されたとき、エラーが見つかったときのいずれかです。

オプションの終了時には、次のことが行われます。

- **getopts** コマンドは、ゼロより大きい戻り値を伴って終了します。
- **OPTARG** は、最初のオプション引数以外のインデックスに設定されます。この場合の最初の **--** 引数は、その前に非オプション引数が他にない場合は、オプション引数と見なされ、非オプション引数がまったくない場合は値 $#+1 と見なされます。
- *Name* が不明のオプションを表す **?** (疑問符) 文字に設定されます。

パラメーター

項目	説明
OptionString	**getopts** コマンドによって認識されるオプション文字の文字列を指定します。文字の後ろにコロンが続く場合には、そのオプションが引数を持つと想定されます。引数は、オプションから離して指定しなければなりません。オプションと引数は、ブランクで区切ります。 *OptionString* の先頭文字により、オプション文字が不明な場合、またはオプション引数が存在しない場合に、**getopts** コマンドによって行われる処理が決まります。 注**:** 疑問符とコロンの文字は、アプリケーションでオプションの文字として使用しないでください。英数字以外のその他の文字を使用すると、結果は保証されていません。
名前	**getopts** コマンドによって、見つけられたオプション文字に設定されます。
Argument ...	ホワイト・スペースで区切られた 1 つ以上の文字列で、**getopts** コマンドによって、有効なオプションであるかどうかが検査されます。 *Argument* を省略すると、定位置パラメーターが使用されます。定位置パラメーターについて詳しくは、Korn シェルの『Korn シェルまたは POSIX シェルにおけるパラメーター置換』のセクションを参照してください。 注**:** 一般的に、*Argument* を **getopts** コマンドの一部として指定することはありませんが、スクリプトのデバッグ時に役に立つ場合があります。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	*OptionString* によって指定されたオプション、または指定されなかったオプションが検出されました。
>0	オプションの終了、あるいはエラーが発生しました。

例

1. 次の **getopts** コマンドは、 a、b、および c が有効なオプションであり、オプション a と c に、引数があることを示します。

   ```
   getopts a:bc: OPT
   ```

2. 次の **getopts** コマンドでは、 a、b、 c が有効なオプションであること、オプション a と b が引数を持つこと、コマンド・ラインに未定義のオプションが存在するときには、**getopts** が、 OPT の値を **?** に設定することを指定しています。

   ```
   getopts :a:b:c OPT
   ```

3. 次のスクリプトは、構文解析を行い、引数を表示します。

   ```
   aflag=
   bflag=

   while getopts ab: name
   do
               case $name in
               a)     aflag=1;;
   ```

```
            b)      bflag=1
                            bval="$OPTARG";;
            ?)      printf "Usage: %s: [-a] [-b value] args¥n" $0
                            exit 2;;
          esac
done

if [ ! -z "$aflag" ]; then
          printf "Option -a specified¥n"
fi

if [ ! -z "$bflag" ]; then
          printf 'Option -b "%s" specified¥n' "$bval"
fi

shift $(($OPTIND -1))
printf "Remaining arguments are: %s¥n" "$*"
```

関連情報:

Korn シェルまたは POSIX シェルのコマンド

---

getrunmode コマンド

目的

システムが稼働しているモードを表示します。

構文

getrunmode

説明

getrunmode コマンドは、システムが稼働しているモードを表示します。稼働モードは、CONFIGURATION モードまたは OPERATIONAL モードです。

例

稼働モードを取得するには、次のように入力します。

```
getrunmode
```

ファイル

項目	説明
/usr/sbin/getrunmode	**getrunmode** コマンドが入っています。

関連情報:

setrunmode コマンド

Trusted AIX

---

getsecconf コマンド

目的

システム・セキュリティー・フラグを表示します。

構文

getsecconf { **-c** | **-o** }

説明

getsecconf コマンドは、システム・セキュリティー・フラグを表示します。オプションを指定しないで呼び出すと、**getsecconf** コマンドはシステムが稼働しているモードに関連したセキュリティー・フラグを表示します。

フラグ

項目	説明
-c	CONFIGURATION (構成) モードを指定します。
-o	OPERATIONAL (動作) モードを指定します。

終了状況

getsecconf コマンドは次の終了値を返します。

項目	説明
0	正常実行。
>0	エラーが発生しました。

例

1. CONFIGURATION (構成) モードでシステム・セキュリティー・フラグを表示するには、次のように入力します。

   ```
   getsecconf –c
   ```

2. OPERATIONAL (動作) モードでシステム・セキュリティー・フラグを表示するには、次のように入力します。

   ```
   getsecconf –o
   ```

ファイル

項目	説明
/usr/sbin/getsecconf	**getsecconf** コマンドが入っています。

関連情報:

setsecconf コマンド

Trusted AIX

---

getsyslab コマンド

目的

システムの最小ラベルと最大ラベルを表示します。

構文

getsyslab

説明

システムの最小と最大の機密ラベル (SL)、および最小と最大の保全性ラベル (TL) を表示するには、**getsyslab** コマンドを使用します。

セキュリティー

getsyslab コマンドは特権コマンドです。このコマンドを正常に実行するには、次の権限が必要です。

項目	説明
aix.mls.system.label.read	システム・ラベルをリストするために必要です。

アクセスされるファイルは以下のとおりです。

項目	説明
モード	ファイル
r	**/etc/security/enc/LabelEncodings**

例

システム・ラベルを表示するには、次のように入力します。

```
getsyslab
```

ファイル

項目	説明
/usr/sbin/getsyslab	**getsyslab** コマンドが入っています。
/etc/security/enc/LabelEncodings	システム・デフォルトのラベル・エンコード・ファイル。

関連情報:

setsyslab コマンド

Trusted AIX

---

gettable コマンド

目的

ホストからネットワーク情報センター (NIC) フォーマットのホスト・テーブルを入手します。

構文

/usr/sbin/gettable [**-v**] *Host* [*OutFile*]

説明

/usr/sbin/gettable コマンドは、*Host* パラメーターにより指定されたサーバーから NIC 標準ホスト・テーブルを入手するために使用します。テーブルは取り出すと、*OutFile* パラメーターで示したファイル内に置かれます。

gettable コマンドは、*Host* パラメーター用のサービス指定内で示されたポートへの伝送制御プロトコル (TCP) 接続をオープンします。次に、すべての名前が要求され、結果的に入手された情報が出力ファイル内に入ります。

gettable コマンドは、NIC 標準ファイル・フォーマットを、ネットワーク・ライブラリー・ルックアップ・ルーチンで使用されるファイル・フォーマットに変換する、 **htable** コマンドと一緒に使用するのが最善の使用方法です。

フラグ

項目	説明
-v	詳細なホスト・テーブルの代わりにバージョン番号のみを取り出して、出力を *OutFile* ファイルに入れます。デフォルトでは、**hosts.ver** ファイルに入ります。

パラメーター

項目	説明
Host	ホスト・テーブル情報を提供するサーバーを指定します。
OutFile	ホスト・テーブル情報を入れるファイルを指定します。 **gettable** コマンドで **-v** フラグを指定しなかった場合は、デフォルトのファイル名が **hosts.txt** になります。

関連資料:

817 ページの『htable コマンド』

関連情報:

伝送制御プロトコル (TCP)

TCP/IP プロトコル

---

gettrc コマンド

目的

トレース・ファイルの収集を管理します。

構文

gettrc [**-c**] [**-C** *dirname*] [**-m**] [**-M** *dirname*] [**-s**] [**-S** *filename*]

説明

gettrc コマンドは、**snap** コマンドと組み合せて使用するスクリプトです。このコマンドは、システム・トレース・ファイル、軽量メモリー・トレース (LMT) ファイル、およびコンポーネント・トレース (CT) ファイルの収集を管理します。

フラグ

フラグ	説明
-c	コンポーネント・トレース・ファイルを収集します。
-C *dirname*	*dirname* で指定されたディレクトリーからコンポーネント・トレース・ファイルを収集します。
-m	メモリー・トレース・ファイルを収集します。
-M *dirname*	*dirname* で指定されたディレクトリーから軽量メモリー・トレース・ファイルを収集します。
-s	システム・トレース・ファイルを収集します。
-S *filename*	*filename* で指定されたディレクトリーからシステム・トレース・ファイルを収集します。

終了状況

項目	説明
0	コマンドは正常に完了しました。
>0	エラーが発生しました。

例

1. **gettrc** を **snap** コマンドと組み合せて使用し、各種のトレース・ファイルを検索するには、次のように入力します。

   ```
   snap "gettrc -c -C dirname -m -M dirname -s -S filename"
   ```

   このコマンドでは、*dirname* で指定されたディレクトリーにリストされるファイルを含めて、システム・トレース・ファイル、LMT ファイル、および CT ファイルが戻されます。

位置

/usr/lib/ras/snapscripts/gettrc

ファイル

/usr/lib/ras/cpufmt

/etc/trcfmt

関連情報:

snap コマンド

---

getty コマンド

目的

ポートの特性を設定します。

構文

getty [[**-r** | **-u** | **-U**] [**-d**] [**-H** *HeraldString*] [**-M** *motdFile*] [**-N**]] *PortName*

説明

getty コマンドでは、端末装置の回線とポートが設定され、管理されます。 **getty** コマンドは、**init** コマンドによって実行されます。 **getty** コマンドは、Terminal State Manager プログラムにリンクされます。Terminal State Manager プログラムからは、端末装置制御機能とログイン機能が組み合わせて提供されます。

getty コマンドは、ホーム・ディレクトリーがない場合にログイン時にホーム・ディレクトリーを作成するように構成できます。**getty** コマンドにより **mkuser.sys** コマンドが呼び出されて、ホーム・ディレクトリーの作成とアカウントのカスタマイズが実行されます。この機能を有効にするには、**/etc/security/login.cfg** ファイルにある **usw** スタンザの **mkhomeatlogin** 属性を真に設定します。

注**: getty** コマンドは、コマンド・ライン上では入力されません。

Terminal State Manager プログラムを **getty** コマンドとして起動すると、次のような通常のポート管理機能が提供されます。

項目	説明
両方向使用	端末回線を使って接続を開始し、受け入れます。
回線速度	送受信のボー・レートを設定します。
パリティー	パリティーを偶数、奇数、またはなしに設定します。
遅延	復帰、タブ、改行、用紙送りに関する遅延を設定します。
文字セット・マッピング	大文字と小文字の区別、タブ、紙送り制御に関する文字セット・マッピングを設定します。
ロガー・プログラム	ユーザーがシステムへログインするときに使用するプログラムを指定します。この属性を設定すると、セキュア・アテンション・キー (SAK) の処理が使用不可になります。この属性を設定しない場合は、デフォルトの **/usr/sbin/login** が使用されます。 **logger** 属性は、オブジェクト・データ・マネージャー (ODM) データベースに入っています。
文字と行の消去	文字と行の消去に使用するキー・ストロークを設定します。
エコー・モード	エコーをローカルまたはリモートに設定します。

getty コマンドを起動すると、次のステップで処理が実行されます。

1. ODM データベース内の **owner** 属性、および **protection** 属性に従って、ポート保護が設定されます。これらの属性が指定されていない場合は、デフォルトの root および 622 が使用されます。
2. *PortName* パラメーターにより指定されたポートがオープンされます。ポート上でキャリア検出が使用可能な場合は、キャリアが存在するか、別のプロセスによってポートを持つキャリアが失われるまで、オープンは完了しません。
3. 指定されたポートがロックされることがあります。 **getty** コマンドに、 **-u** フラグ、または **-r** フラグを付けて実行すると、ポートのロックが試行されます。ポートが既にロックされている場合は、コマンドはポートが使用可能になるまで待機してから、終了します。 **-r** フラグを指定した場合、**getty** コマンドは、ポート上で 1 バイトのデータを受け取るまで待機してから、処理を続行します。
4. 指定したポートに関する設定情報に従って、端末属性が設定されます。システム設定によっては、この時点でセキュア・アテンション・キーの処理を使用可能にできます。
5. 指定したポートに herald メッセージが書き出されます。
6. 指定したポートからログイン名が読み取られます。フレーム・エラーまたは中断が発生すると、**getty** コマンドによって、次の設定済み端末属性グループを使って、ステップ 4 と 5 が再度実行されます。この使用頻度が最も高いのは、モデムのボー・レートを循環するときです。ただし、どの ODM フィールド (`logmodes` および `runmodes` を除く) も、ODM データベース内に値のリストをコンマで区切って入力すると循環できます。
7. `runmodes` パラメーターとログイン名に従って、端末モードがリセットされます。ログイン名が改行で終了していると、**getty** コマンドによって、復帰から改行へのマッピングがオンに設定されます。すべての英文字が大文字の場合、可能であれば小文字でログインするように促され、小文字から大文字へのマッピングがオンになります。
8. ロガー・パラメーターでプログラムを指定した場合は、そのプログラムが実行され、セキュア・アテンション・キーの処理が使用不可になります。それ以外の場合は、Terminal State Manager プログラムによって標準システム・ログインが実行されます。

   注**:** ユーザーのログイン時にセキュア・アテンション・キーのシーケンスを入力した場合には、ユーザーはトラステッド・シェルにログインされます (ただし、ポートが承認されており、ユーザーがトラステッド・パス上で許可されているシステム設定の場合に限ります)。

フラグ

項目	説明
-d	デバッグ情報を提示します。
-H *HeraldString*	ログイン名のプロンプトを出すためにポートに書き込む代替の herald メッセージを指定します。メッセージ文字列は 1 ワードにする必要があり、スペースを含めることはできません。この文字列は **/etc/security/login.cfg** ファイルで定義された herald メッセージより優先します。このオプションまたは **login.cfg** ファイルで文字列が何も指定されていないと、メッセージ・カタログからデフォルトの herald が使用されます。
-M *motdFile*	パスを day ファイルの代替メッセージに指定します。これが指定されていないと、値はデフォルトにより **/etc/motd** になります。
-N	**getty** に **/etc/utmp** ファイルにあるプロセス ID のチェックをバイパスさせます。その結果、最も低いログイン・シェル以外のプロセスは、**exec getty** を行うことができます。
-r	ポートを共用 (両方向使用) できるようにします。ロックに失敗すると、**getty** コマンドは、ロックが使用可能になるまで待機してから、終了します。ロックに成功すると、**getty** コマンドは、ポートをロックした後でポート上で 1 バイトのデータを待機します。
-u	ポートを共用 (両方向使用) できるようにします。ロックに失敗すると、**getty** コマンドは、ロックが使用可能になるまで待機してから、終了します。
-U	**-u** フラグと同じです。ただし、**getty** は、ロックが使用可能になるのを待ちません。これは、ロックとは関係なく、ポートを使用可能にします。

セキュリティー

アクセス制御: このプログラムは、トラステッド・コンピューティング・ベース内のプログラムとしてインストールされ、任意のユーザーによる実行が可能で、**setuid** で root に設定されていなければなりません。

例

tty0 へのログ記録を使用可能にするには、**/etc/inittab** ファイルに次の行を追加します。

```
tty0:2:respawn: /usr/sbin/getty /dev/tty0
```

このコマンドによって、ポート `/dev/tty0` が初期化され、ポートの特性が設定されます。

ファイル

項目	説明
/usr/sbin/getty	**getty** コマンドが入っています。
/etc/locks	通信デバイスの複数使用とリモート・システムの複数の呼び出しを防止するロック・ファイルが入っています。
/usr/sbin/login	**login** コマンド。
/etc/security/login.cfg	ポート・ログイン構成が入っています。
/etc/motd	ログインの後で表示される day のメッセージが入っています。
/usr/bin/setmaps	**setmaps** コマンド。
/etc/utmp	システムにログインしたユーザーについての情報が入っています。

関連情報:

login コマンド

shell コマンド

telinit または init

プログラマーのためのオブジェクト・データ・マネージャー (ODM) の概要

---

gmvgstat コマンド

gmvgstat コマンドのマニュアル・ページには、**gmvgstat** コマンドのための参照情報があります。

目的

GMVG 統計情報を表示します。

構文

```
gmvgstat [-h] | [-r] [-t] [-i Interval [-c Count] [-w]]
[gmvg_name . . .]
```

説明

gmvgstat コマンドは、以下を含む、1 つ以上の GMVG の状況情報を表示します。

- 物理ボリューム数
- リモート物理ボリューム数
- ボリューム (PV および RPV) の合計数
- 不整合ボリューム数
- 物理区画 (PP) の合計数
- 不整合 PP 数
- GMVG の同期化率 (%)

gmvgstat コマンドはオプションで、-i フラグおよび -c フラグを指定してモニター・モードで呼び出すことができます。

1 つ以上の GMVG 名をコマンド行で指定した場合、**gmvgstat** コマンドは、リストされた GMVG 名がいずれも有効で使用可能なオンライン GMVG であることを検査します。モニター・モードでは、ユーザー指定の GMVG リストが各ループ中に検査されます。

GMVG 名をコマンド行で指定しない場合、**gmvgstat** コマンドは、有効で使用可能なすべてのオンライン GMVG に関する情報を報告します。モニター・モードでは、報告される GMVG のリストが各ループ中に再生成されます。

フラグ

表 *6. gmvgstat* コマンドのフラグ

フラグ	説明
-h	コマンド構文およびヘルプを表示します。
-r	表示される GMVG に関連付けられたそれぞれの個別 RPV クライアントの情報を組み込みます。
-t	日時のあるヘッダーを表示します。
-i Interval	<Interval> 秒ごとに状況を自動的に再表示します。 <Interval> パラメーターの値は、1 以上 3600 以下の整数でなければなりません。 <Interval> パラメーターを指定しない場合、状況情報は一度のみ表示されます。 -i 間隔は、モニター・モードでの GMVG 統計情報の一連の各収集および表示間の時間 (秒) です。この間隔は、一連の各更新表示間の経過時間を正確に計測した値ではありません。 gmvgstat コマンドは、他のコマンドを呼び出すことで、表示する情報の一部を取得しますが、それらのコマンドが処理を完了するまでの時間は制御できません。 GMVG の数が増えるほど、gmvgtstat コマンドによる情報の収集には時間がかかり、モニター・モードでの一連の表示間隔は長くなります。場合によっては、内在するコマンドの完了に過度に長い時間がかかることがあり、その結果 gmvgstat コマンドは、表示間の -i 間隔よりもずっと長い時間がかかることになります。
-c Count	指示された間隔で情報を <Count> 回再表示します。 <Count> パラメーターの値は、1 以上 999999 以下の整数でなければなりません。 <Interval> パラメーターを指定し、<Count> パラメーターを指定しない場合は、無制限に再表示が行われます。
-w	各再表示の間に画面を消去します。

オペランド

表 7. オペランド・フィールド

フィールド	値
gmvg_name	情報を表示する 1 つ以上の GMVG の名前。 GMVG 名を指定しない場合は、有効で使用可能なすべてのオンライン GMVG に関する情報が表示されます。

終了状況

表 8. 終了状況

値	説明
0	エラーはありません。
>0	エラーが発生しました。

例

1. すべての GMVG の統計情報を表示するには、次のように入力します。

   ```
   gmvgstat
   ```

2. red_gmvg7 という名前の GMVG の統計情報を表示するには、次のように入力します。

   ```
   gmvgstat red_gmvg7
   ```

3. red_gmvg7 という名前の GMVG の統計情報を、そのボリューム・グループに関連付けられたすべての RPV に関する統計情報とともに表示するには、次のように入力します。

   ```
   gmvgstat -r red_gmvg7
   ```

4. GMVG red_gmvg7 の詳細情報を表示し、それを 10 秒ごとに自動的に再表示するには、次のように入力します。

   ```
   gmvgstat red_gmvg7 -i 10
   ```

5. GMVG red_gmvg7 の詳細情報を表示し、その情報を自動的に 10 秒ごとに 20 回再表示し、再表示間で画面を消去するには、次のように入力します。

   ```
   gmvgstat red_gmvg7 -i 10 -c 20 -w
   ```

ファイル

gmvgstat コマンドは /usr/sbin/gmvgstat にあります。

---

gprof コマンド

目的

コール・グラフのプロファイル・データを表示します。

構文

/usr/ccs/bin/gprof [**-b**] [**-c** [*filename*]] [**-e** *Name*] [**-E** *Name*] [**-f** *Name*] [**-g** *filename*] [**-i** *filename*] [**-p** *filename*] [**-F** *Name*] [**-L** *PathName*] [**-s**] [**-x** [*filename*]] [**-z**] [**a.out** [**gmon.out** ...]]

説明

gprof コマンドは、C、FORTRAN、または COBOL の各プログラムの実行プロファイルを生成します。呼び出されたルーチンの効果は、各呼び出し元のプロファイルに組み込まれます。 **gprof** コマンドは、プ

ログラムがどのようにプロセッサー・リソースを使用するかを決定するときに役立ちます。プログラム内のどの関数 (ルーチン) によってプロセッサーが使用されているかを確認するには、**gprof** コマンドを使ってプログラムのプロファイルを作成します。

プロファイル・データは、**-pg** オプションを使用している **cc** コマンドを使ってコンパイルしたプログラムによって作成されたコール・グラフ・プロファイル・ファイル (デフォルトでは **gmon.out**) から取り出されます。また、**-pg** オプションでは、プロファイル作成のためにコンパイルしたバージョンのライブラリー・ルーチンにリンクし、名前付きオブジェクト・ファイル (デフォルトでは **a.out**) 内の記号テーブルを読み取って、コール・グラフのプロファイル・ファイルと相関関係を持たせます。複数のプロファイル・ファイルを指定すると、**gprof** コマンド出力に、指定したプロファイル・ファイル内のプロファイル情報がすべて示されます。

-pg オプションを指定すると、コンパイラーによって、**mcount** サブルーチンの呼び出しが、プログラムの各再コンパイル関数ごとに生成されたオブジェクト・コードに挿入されます。プログラム実行中に親が子関数を呼び出すたびに、子は **mcount** サブルーチンを呼び出して、その親子の対の独立カウンターを増分させます。 **-pg** オプションを使用して再コンパイルしなかったプログラムには、**mcount** サブルーチンがないので、呼び出し元の記録は残りません。

注**:** C++ オブジェクト・ファイル名からのシンボルは、使用する前に変更されます。

GPROF 環境変数を使用して、プロファイル作成のために異なるオプションを設定できます。この環境変数の構文は、以下のように定義されています。

```
GPROF = profile:<profile-type>,scale:<scaling-factor>,file:<file-type>,filename:<filename>
```

それぞれの意味は次のとおりです。

- <profile-type> は、どのようなタイプのプロファイル作成が必要であるかを記述します。これは、process または thread のいずれかです。タイプ「process」はプロファイル作成の細分度がプロセス・レベルであることを指示し、「thread」はプロファイル作成の細分度がスレッド・レベルであることを指示します。
- <scaling-factor> は、コール・グラフのプロファイルに割り振る必要のあるメモリー量を記述します。デフォルトでは、プロセス・レベルのプロファイル作成では倍率 2、スレッド・レベルのプロファイル作成では倍率 8 です。この倍率を 2 にするとプロセス・サイズの半分のメモリーが各プロセスまたはスレッドに割り振られ、また、8 にするとプロセス・サイズの 1/8 のメモリーが各プロセスまたはスレッドに割り振られるという指示になります。このメモリーは、コール・グラフ情報を保管するためのバッファー域です。
- <file-type> は、必要とする **gmon.out** ファイルのタイプを記述します。 **multi** という値はプロセスごとに 1 **gmon.out** ファイルが必要であることを指定し、multithread という値はスレッドごとに 1 **gmon.out** ファイルが必要であることを指定します。アプリケーションが **-pg** オプションを指定してプロファイルが作成されていて、fork を行う場合は、**multi** の指定によって、親プロセスのために 1 つの **gmon.out** ファイルが、子プロセスのためにもう 1 つのファイルが生成されます。生成される **gmon.out** ファイルの命名規則は、以下のとおりです。
  - multi ファイル・タイプの場合: <prefix>-processname-pid.out
  - multithread ファイル・タイプの場合: <prefix>-processname-pid-Pthread<threadid>.out

  <prefix> はデフォルトでは **gmon** です。 GPROF 環境変数の *filename* パラメーターを使用して、独自のプレフィックスを定義できます。
- <filename> は、生成される **gmon.out** ファイルに使用する必要のあるプレフィックスを記述します。デフォルトでは、プレフィックスは **gmon** です。

注: `profile:thread` によって、フォーマット **gmon.out** ファイルが生成されます。このファイルは AIX 5.3 **gprof** コマンドによってのみ読むことができます。旧フォーマットの **gmon.out** ファイルが必要で、しかも、**profile:thread** を指定したい場合は、**file:multithread** を指定する必要があります。これによって、旧フォーマットの **gmon.out** ファイルがスレッドごとに生成されます。したがって、アプリケーションに 2 つのスレッドがあれば、命名規則を使用して 2 つの **gmon.out** ファイル (1 スレッドにつき 1 ファイル) が生成されます。AIX 5.2 またはそれ以前で **-pg** フラグを指定してアプリケーションをコンパイルし、それを AIX 5.3 で実行する場合、スレッド・レベルでのプロファイル作成は使用可能になりません。スレッド・レベルでのプロファイル作成を使用可能にするには、アプリケーションを **-pg** フラグ指定で AIX 5.3 以降でコンパイルする必要があります。

gprof コマンドは、次の 3 つの項目を生成します。

1. 第 1 に、**prof** コマンドによって提供されるものと同様の、フラット・プロファイルが生成されます。このリストには、プログラム内の各関数の合計実行時間とコール・カウントが、降順にソートされて表示されます。次に、時間がコール・グラフの縁に沿って伝搬されます。サイクルが見つかると、サイクルの時間を共用するためにサイクルへの呼び出しが行われます。
2. 第 2 のリストには、コール・グラフの子孫の時間も含めて、表す時間に従ってソートされた関数が示されます。各関数項目の下には、その (直接の) コール・グラフの子と、時間がこの関数にどのように伝搬されるかが示されます。関数の上の同様の表示には、関数の時間とその下位の時間がどのように (直接の) コール・グラフの親に伝搬されるかが示されます。
3. サイクルも表示されます。サイクルについては、サイクル全体に関するエントリー、サイクルのメンバーのリスト、時間への影響、サイクルのコール・カウントに関するリストが表示されます。

注: **gprof** コマンドへの入力にスレッド・レベルのプロファイル作成データ (**gmon.out** ファイルのフォーマットによる) が含まれる場合は、**gprof** コマンドによって上記の 3 つの項目がスレッドごとに作成されます。最初が累積報告、次にスレッドごとの報告 (スレッド ID の昇順にソートされている) が続きます。

grpof コマンドを使用して、リモート・マシン上のプログラムの実行プロファイルを分析することもできます。その分析を行うには、コール・グラフ・プロファイル・ファイル (デフォルトでは **gmon.out**) 上で **-c** オプション指定の **gprof** コマンドを実行して、後でリモート・マシン上で処理できるファイル (デフォルトでは **gprof.remote**) を生成します。**gmon.out** 以外のコール・グラフ・プロファイル・ファイルを使用する場合は、コール・グラフ・プロファイル・ファイル名を、**-c** *Filename* および実行可能ファイル名の後に指定する必要があります。**GPROF** 環境変数の **file**属性が **multi** に設定されている場合は、*Filename* を指定する必要があります。これにより、複数の **gmon.out** ファイルが作成されます。この場合、実行プログラムの fork 時に PID ごとに 1 つずつ **gmon.out** ファイルが作成されます。リモート・マシン上で **-x** オプションを使用して、プロファイル報告書を生成するための **gprof.remote** (デフォルトの場合) ファイルを処理できます。

fork および **exec** サブルーチンを使ったプロファイル処理

プログラムが複数の並行処理に対して、 **fork**、または **exec** サブルーチンを実行する場合は、 **gprof** コマンドを使ったプロファイル処理が問題となります。プロファイル処理は、各プロセスの環境の属性なので、新しいプロセスを fork するプロセスのプロファイル処理をすると、その子のプロファイルも処理されます。ただし、両方のプロセスによって、親プロセスを実行するディレクトリーに **gmon.out** ファイルが書き込まれるので、一方が上書きされます。複数のプロセスのプロファイル処理をするときには、**tprof** コマンドの使用をお勧めします。**file:multi** を使用して、親プロセスの **gmon.out** ファイルが削除されるのを防ぐことができます。**file:multi** は AIX の命名規則を使用して **gmon.out** ファイルを生成するので、子プロセスの **gmon.out** ファイルは親と同じ名前を持ちません。したがって、上書きが避けられます。

ソース・コードなしのプロファイル処理

プログラムのソースがない場合は、再コンパイルせずに **gprof** コマンドを使ってプロファイル処理します。しかし、プログラム・モジュールを適切なコンパイラー・コマンド (例えば、C の場合は **cc**) に再リンクできなければなりません。再コンパイルしないと、コール・カウントを取得できませんが、コール・カウントがなくてもフラット・プロファイルは役に立ちます。追加の利点として、プログラムの実行速度は通常とほとんど変わりません。次の例では、プロファイルの方法について説明します。

```
cc -c dhry.c         # Create dhry.o without call counting code.
cc -pg dhry.o -L/lib -L/usr/lib -o dhryfast
                     # Re-link (and avoid -pg libraries).
dhryfast             # Create gmon.out without call counts.
gprof >dhryfast.out  # You get an error message about no call counts
                     #  -- ignore it.
```

コール・カウントなしで実行した結果は、ある種の高速に実行された関数が (呼び出されたのに) リストには現れないということを示しています。しかし、これが **gprof** コマンドの通常の結果です。**gprof** コマンドは、最低でも 1 回呼び出された関数、または最低でも 1 クロック・ティック間登録された関数だけを表示します。ところが、高速に実行された関数は、実際に実行されても、ほとんどの場合クロック・ティックをまったく受け取っていません。コール・カウントは中断されているので、このような小さな関数はリストには現れません。 (ランタイム・ルーチンのコール・カウントを取得するには、**cc -pg** コマンド・ラインで **-L** オプションを省略します)。

使用する実メモリーの量を少なくする

gprof コマンドを使用してプロファイルを作成すると、プログラムによって過度のページングが行われる可能性があります。これは、**-pg** オプションにより、プログラム・テキストのサイズの半分に相当する固定された実メモリー・バッファー・スペースが専用化されるためです。ページングが過度に行われても、プロファイル作成によって生成されるデータに影響はありません。これは、プロファイル作成されるプログラムでは、プロセッサーを使用するときだけティックが生成され、入出力の待ち状態のときにティックが生成されないためです。過度のページングによる時間遅延を受け入れることができない場合は、**tprof** コマンドの使用をお勧めします。

フラグ

項目	説明
-b	プロファイル内の各フィールドの記述が出力されるのを抑止します。
-c *Filename*	プロファイル作成情報のリモート処理に必要な情報を含むファイルを作成します。**-c** フラグを他のフラグと組み合わせて使用してはなりません。
-E *Name*	**-e** フラグと同様に、ルーチン *Name* とその子孫ルーチンに関するグラフ・プロファイル・エントリーの出力を抑止しますが、時間の合計とパーセントの計算から、ルーチン *Name* とその子孫ルーチンによって費やされた時間を除外します。(デフォルトは、**-E** *MonitorCount* **-E** *MonitorCleanup* です。)
-e *Name*	ルーチン *Name* とそのすべての子孫ルーチンに関するグラフ・プロファイル・エントリーの出力を (ほかの祖先が抑止されていない場合のみ) 抑止します。**-e** フラグは複数指定できます。ルーチンは、**-e** フラグごとに 1 つしか指定できません。
-F *Name*	**-f** フラグと同様に、ルーチン *Name* とその子孫ルーチンについてのみグラフ・プロファイル・エントリーを出力しますが、合計時間とパーセントの計算には出力されたルーチンの時間だけを使います。 **-F** フラグは複数指定できます。ルーチンは、**-F** フラグごとに 1 つしか指定できません。**-F** フラグを指定すると、**-E** フラグは指定変更します。
-f*Name*	指定したルーチン *Name* とその子孫ルーチンについてのみグラフ・プロファイル・エントリーを出力します。 **-f** フラグは複数指定できます。ルーチンは、**-f** フラグごとに 1 つしか指定できません。
-g *Filename*	コール・グラフ情報を指定の出力 *filename* に書き込みます。 **-p** フラグを使用していない場合は、これによってプロファイル情報も抑止されます。
-i *Ffilename*	ルーチン・インデックス・テーブルを指定の出力 *filename* に書き込みます。このフラグを使用しない場合は、インデックス・テーブルは、標準出力の終わり、または **-p** と **-g** のフラグで指定されたファイル名の最後のいずれかに書き込まれます。
-L *PathName*	共用オブジェクトの位置を決定するときに使用する代替パス名を使用します。

項目	説明
-p *Filename*	指定された出力ファイル名にフラット・プロファイル情報を書き込みます。**-g** フラグが使用されていなければ、これによってコール・グラフ情報も抑止されます。
-s	指定したすべてのプロファイル・ファイル内のすべてのプロファイル情報がまとめられたプロファイル・ファイル **gmon.sum** を生成します。この要約プロファイル・ファイルを指定して **gprof** コマンドを (**-s** フラグを使用して) 実行すると、**a.out** ファイルの複数の実行にわたってプロファイル・データを累積できます。
-x *Filename*	プロファイル報告書を生成するために *Filename* (**-c** オプションを使用して作成されたファイル) から情報を取り出します。*Filename* を指定しない場合は、**gprof** コマンドはデフォルトの **gprof.remote** ファイルを検索します。
-z	使用量がゼロのルーチンを表示します (コール・カウントと累積時間で示されます)。

例

1. プロファイル作成済みの出力を取得するには、次のコマンドを入力します。

   ```
   gprof
   ```

2. 以前に実行し、移動された可能性のあるコマンドから、プロファイル出力を取得するには、次のコマンドを入力します。

   ```
   gprof -L/home/score/lib runfile runfile.gmon
   ```

   この例では、**runfile.gmon** ファイルをサンプル・データとして、また **runfile** ファイルをローカル記号として使用し、ロード可能なオブジェクトを **/u/score/lib** ファイル内で探します。

3. サンプル・プログラム **dhry.c** のプロファイルを作成するには、次のようにします。

   a. 次のように、アプリケーション・プログラムを **cc -pg** コマンドで再コンパイルします。

      ```
      cc -pg dhry.c -o dhry # Re-compile to produce gprof output.
      ```

   b. 再コンパイルしたプログラムを実行します。gmon.out というファイルが現在の作業ディレクトリーに作成されます (プログラム実行可能ファイルが置かれているディレクトリーではありません)。

      ```
      dhry    # Execute program to generate ./gmon.out file.
      ```

   c. **gprof** コマンドを、gmon.out ファイルが入っているディレクトリーで実行し、コール・グラフ・レポートとフラット・プロファイル・レポートを生成します。

      ```
      gprof >gprof.out     # Name the report whatever you like
      vi gprof.out         # Read flat profile first.
      ```

   d. 細分度がスレッド・レベルのプロファイル作成を生成し、GPROF 環境変数を以下のようにエクスポートし、アプリケーションを実行するには、次のコマンドを入力します。

      ```
      export GPROF=profile:thread
      dhry   # Execute program to generate ./gmon.out file which has thread level granularity
      ```

   e. プロセスごとの **gmon.out** ファイルを、プレフィックス mygmon で生成するには、次のコマンドを入力します。

      ```
      export GPROF=file:multi,filename:mygom
      dhry  # Execute program to generate ./gmon-dhry-2468.out
      ```

   f. スレッドごとの **gmon.out** ファイルを、倍率 10、tgmon というプレフィックスを付けたファイル名で生成するには、次のコマンドを入力します。

      ```
      export GPROF=profile:thread,file:multithread,scale:10,filename:tgmon
      dhry # Execute program to generate ./tgmon-dhry-2468-Pthread215.out
      ```

   g. gmon-dhry-2468.out からフラット・プロファイル・レポートだけを見るためには、次のコマンドを入力します。

      ```
      gprof -p fprofile.out ./dhry ./gmon-dhry-2468.out
      ```

h. gmon-dhry-2468.out からコール・グラフ・プロファイル・レポートだけを見るためには、次のコマンドを入力します。

```
gprof -g callgraph.out ./dhry ./gmon-dhry-2468.out
```

4. **gprof** コマンドのリモート処理機能を使用する場合:
   a. 次のように、アプリケーション・プログラムを **cc -pg** コマンドで再コンパイルします。

```
cc -pg thread.c -o thread  -lpthread
```

   b. 次のように、スレッド・レベルのプロファイル細分度を使用可能にし、**gmon.out** に別の名前を使用します。

```
export GPROF=profile:thread,filename:mygmon
```

   c. 再コンパイルしたプログラムを実行します。**mygmon.out** というファイルが現在の作業ディレクトリー (プログラム実行可能ファイルが置かれているディレクトリーではありません) に作成されます。

```
thread    # Execute program to generate mygmon.out file.
```

   d. 次のように **-c** フラグを使用して、**my.remote** ファイルを生成します。このファイルは、リモート・マシンで使用して処理できます。

```
gprof -c my.remote thread mygmon.out
```

   e. リモート・マシンでは、次のように **-x** フラグを使用して、**my.remote** ファイルから情報を抽出します。

```
gprof -x my.remote
```

以上の **gprof** コマンドの説明では、ほとんどの例で C プログラム **dhry.c** を使用しています。しかし、C コンパイラーの **cc** を適切なコンパイラー名に置き換え、関数 という言葉をサブルーチン という言葉に置き換えれば、FORTRAN または COBOL の各モジュールについても同様に実行できます。例えば、次のコマンドは、 matrix.f という FORTRAN プログラムのプロファイルを、どのように作成するかを示します。

```
xlf -pg matrix.f -o matrix # FORTRAN compile of matrix.f program
matrix                     # Execute with gprof profiling,
                           #   generating gmon.out file
gprof > matrix.out         # Generate profile reports in
                           #   matrix.out from gmon.out
vi matrix.out              # Read flat profile first.
```

ファイル

項目	説明
a.out	名前リストとテキスト・スペース
gmon.out	動的コール・グラフとプロファイル
gmon.sum	要約された動的コール・グラフとプロファイル
gprof.remote	リモート・プロファイル作成用のファイル
/usr/ucb/gprof	**gprof** コマンドが入っています。
/usr/ccs/bin/gprof	**gprof** コマンドが入っています。

関連情報:

prof コマンド

exit コマンド

コマンドとサブルーチンのモニターおよびチューニング

サブルーチンの概要

grap コマンド

目的

pic コマンドで処理するグラフをタイプセットします。

構文

grap [**-l**] [**-T** *Name*] [**—**] [*File ...*]

説明

grap コマンドは、grap 言語入力ファイルを処理し、**pic** コマンドに対する入力を生成します。grap 言語とは、グラフのタイプ・セット用の言語です。典型的なコマンド・ラインは次のとおりです。

```
grap File | pic | troff | Typesetter
```

グラフは、**.G1** と **.G2 troff** のコマンド要求で囲みます。これらの要求で囲まれたデータは、自動的に指定されるティック・マークを使ってスケーリングされ、プロットされます。コマンドにはフレーム変更コマンド、ラベル追加コマンド、デフォルトの目盛りを指定変更するコマンド、プロット・スタイル変更コマンド、座標範囲と変形を定義するコマンド、ファイルからのデータを指定するコマンドがあります。さらに、**grap** コマンドは、**pic** コマンドと同じループ、条件付き処理、マクロ処理を提供します。

Grap 言語ファイルには grap プログラムが定義されています。grap プログラムは次の構文で書かれています。

```
.G1
grap Statement
grap Statement
grap Statement
.G2
```

パラメーター

項目	説明
File	grap コマンドが処理して、**pic** コマンドへ入力する **grap** 言語ファイル (grap プログラム) を指定します。

grap ステートメントの要約

以下に、grap プログラムの作成時に使用できる grap ステートメントをまとめます。

項目	説明
frame	グラフを囲むフレームを定義します。構文は次のとおりです。 `frame [ht Expression] [wid Expression] [[Side] LineDescription]`

属性は次のように定義されます。

- *Side*: `top , bot , left , right`
- *LineDescription*: `solid, invis, dotted [Expression], dashed [Expression]`

高さのデフォルト値は 2 インチ、幅のデフォルト値は 3 インチ、側面のデフォルト値は実線です。`side` を省略した場合は、*linedesc* がフレーム全体に適用されます。

項目	説明
label	グラフの指定した側にラベルを付けます。構文は次のとおりです。 `label Side StringList ... Shift`

属性は次のように定義されます。

- *Shift*: `left, right, up, down` *expression*
- *StringList*: str ... `rjust, ljust, above, below [size (+)Expression] ...`
- *String*: `"..."`

項目	説明
coord	指定変更するシステムを定義します。構文は次のとおりです。 `coord [Name] [x Expression,Expression] [y Expression,Expression] [[log x] [log y] [log log]]`
ticks	フレームの一方の側にティック・マークを付けます。構文は次のとおりです。 `ticks side [[in] [out] [Expression]] [Shift] [TickLocations]`

属性は次のように定義されます。

- *Shift*: `left, right, up, down Expression`
- *TickLocations*: at [*Name*] *Expression* [*String*], *Expression* [*String*], ... from [*Name*] *Expression* to *Expression* [by [*Operation*] *Expression*] *String*

ティック・マークが指定されていない場合は、自動的に付加されます。`ticks off` を指定した場合は、ティック・マークが自動的に付加されるのを抑止します。

項目	説明
grid	指定した側面に沿って (つまり、垂直に) 格子線を作成します。構文は次のとおりです。 `grid Side [LineDescription] [Shift] [TickLocations]`

格子にはティック・マークと同様の方法でラベルが付けられます。

項目	説明
plot	ポイントにテキストを配置します。構文は次のとおりです。 `StartList at Point plot Expression [Start] at Point`

属性は次のように定義されます。

- *StringList*: str ... `rjust, ljust, above, below[size+)Expression] ...`
- *Point*: `[Name] Expression Expression`

項目	説明
line	ある点から他の点に直線または矢印を引きます。構文は次のとおりです。 `{line | arrow} from Point to Point [LineDescription]`

属性 linedesc は次のように定義されます。

- *Point*: [*Name*] *Expression Expression*
- *LineDescription*: `solid, invis, dotted[Expression],dashedExpression]`

項目	説明
circle	円を描きます。構文は次のとおりです。 `circle at Point [radius Expression]`

半径の単位はインチで、デフォルトのサイズは small です。

項目	説明
draw	行のシーケンスを定義します。構文は次のとおりです。 `draw [Name] at Point[LineDescription]`
next	シーケンスを続行します。構文は次のとおりです。 `next [Name] at Point [LineDescription]`
new	新しいシーケンスを開始します。構文は次のとおりです。 `new [Name] at Point [LineDescription]`
numberlist	任意の数値群から直線を作成します。数値は点 *x*, *y*1, *y*2 などのように扱われます。つまり、*x* の値を持つ一点にプロットされます。構文は次のとおりです。 `number x, y1, y2 ...`
for	ループを作成します。構文は次のとおりです。 `for Variable {from | =} Expression to Expression ¥` `[by [arithmetic or multiplicative operator] Expression] do X Anything X` X は文字列中にはない 1 つの文字です。 X が左中括弧 { の場合は、文字列の中に対になる中括弧が含まれてその後ろに右中括弧 } が続く場合もあります。Variable が第 1 の Expression から第 2 の Expression までの値を継続して読み取るとき、テキスト Anything は反復されます。
if	条件評価を作成します。構文は次のとおりです。 `if Expression then X Anything X [else X Anything X]`
define	優先割り込みコントローラー (PIC) と同じマクロ・プロセッサーを提供します。構文は次のとおりです。 `define MacroName X Anything X`
copy	現行ファイルの内容を含め、ファイルをコピーします。構文は次のとおりです。 `copy Filename`
copy-thru	次のように、マクロを介してファイルをコピーします。 `copy Filename thru MacroName` 各数値または引用される文字列は、引数として扱われます。コピーは、ファイルの終わりか次の .G2 まで続行されます。untilString 節を任意選択すると、最初のフィールドが String になっている行がきたとき、コピーは中止されます。 次のステートメントでは、マクロを介してその後ろの行がコピーされます。 `copy thru MacroName` どの場合にも、次のように名前でなくインラインでマクロを指定できます。 `copy thru x MacroBody x`
sh	テキストを UNIX シェルに渡します。構文は次のとおりです。 `sh x Anything x` 変数 Anything はマクロを探してスキャンされます。**pid** マクロは組み込み式です。プロセス識別番号からなる文字列であり、固有のファイル名を生成するために使用できます。
pic	テキストを **pic** に渡し、pic を除去します。変数とマクロは評価されません。(数字ではなく) ピリオドで始まる行は、**troff** コマンドであるという想定のもとに、文字どおりの内容が渡されます。
graph	*Picname* という名前の新しいグラフを定義し、すべての座標系をリセットします。構文は次のとおりです。 `graph Picname [pic-text]` graph プログラムで **graph** コマンドを使用する場合、.G1 の後ろのステートメントは **graph** コマンドにしなくてはなりません。次の例にあるように、pic-text を使用して、以前のグラフの Frames を参照することにより、前のグラフを基準にしてこのグラフの位置を決めることができます。 `graph First` `...` `graph Second with .Frames.w at First.Frame.e + [0.1,0]` pic-text 内のマクロと式は評価されません。Picnames は、**pic** 構文に従って大文字で始まらなければなりません。

項目	説明
印刷	grap が入力を処理しているときに、stderr に書き出します。このステートメントはデバッグに役立ちます。構文は次のとおりです。 `print [Expression | String]`

grap 言語規則

次の規則が適用されます。

- コメントの先頭には、 # (ポンド記号) が付きます。コメントは行の終わりで自動的に終了します。
- 2 行以上に渡るステートメントでは、改行した行頭すべてに ¥ (円記号) を付けなければなりません。
- 1 行に複数のステートメントがある場合は、セミコロンで区切らなければなりません。
- grap 言語はブランク行を無視します。
- bullet、plus、box、 star、dot、times、htick、vtick、square、および delta は、事前定義済み文字列です。
- grap で使用できる組み込み関数には、 log (基数 10)、exp (基数 10)、 int、sin、 cos、atan2、sqrt、min、 max、および rand が定義されています。

フラグ

項目	説明
-l	**grap** コマンドがマクロ定義の **/usr/lib/dwb/grap.defines** ライブラリー・ファイルを探すのを停止させます。
-T*Name*	*Name* 変数の値を **grap** コマンドの出力デバイスとして指定します。デフォルト値は、**-Tibm3816** です。
- -	(二重ダッシュ) フラグの終わりを示します。

ファイル

項目	説明
/usr/lib/dwb/grap.defines	標準プロット文字の定義が入っています。

関連情報:

pic コマンド

---

greek コマンド

目的

テレタイプ・モデル 37 ワークステーションからの英語出力を他のワークステーション用の出力に変換します。

構文

greek [**-T** *Name*]

説明

greek コマンドは、テレタイプ・モデル 37 文字セットを、反転や 1/2 行移動を含めて、他のワークステーションで表示できるように変換します。また、可能であれば重ね打ちによって特殊文字をシミュレートします。 **greek** コマンドは、標準入力を読み取って標準出力に書き出します。

フラグ

項目	説明
-T*Name*	指定されたワークステーション名を使用します。**-T** フラグを省略すると、**greek** コマンドは **$TERM** 環境変数で指定されたワークステーションを使用します。 *Name* 変数の値は、次のいずれかです。
300	DASI 300
300-12	12 ピッチの DASI 300
300s	DASI 300s
300s-12	12 ピッチの DASI 300s
450	DASI 450
450-12	12 ピッチの DASI 450
2621	Hewlett-Packard 2621、2640、2645
2640	Hewlett-Packard 2621、2640、2645
2645	Hewlett-Packard 2621、2640、2645
4014	Tektronix 4014
hp	Hewlett-Packard 2621、2640、2645
tek	Tektronix 4014

環境変数

項目	説明
$TERM	ワークステーション名を指定します。

関連資料:

418 ページの『eqn コマンド』

803 ページの『hp コマンド』

関連情報:

mm コマンド

troff コマンド

---

grep コマンド

目的

ファイルからパターンを検索します。

構文

grep [**-E** | **-F**] [**-i**] [**-h**] [**-H**] [**-L**] [**-r** | **-R**] [**-s**][**-u**] [**-v**] [**-w**] [**-x**] [**-y**] [[[**-b**] [**-n**]] | [**-c** | **-l** | **-q**]] [**-p** [*Separator*]] { [**-e** *PatternList* ...] [**-f** *PatternFile* ...] | *PatternList* ... } [*File* ...]

説明

grep コマンドは、*Pattern* パラメーターで指定されたパターンを検索し、一致する各行を標準出力に書き出します。パターンは、**ed** コマンドまたは **egrep** コマンドのスタイルの正規表現に限定されます。 **grep** コマンドは、コンパクトな決定論的でないアルゴリズムを使います。

File パラメーター内で複数の名前を指定すると、**grep** コマンドは一致した行が入っているファイルの名前を表示します。シェルに対して特別な意味を持つ文字 ($、*、[、|、^、(、)、¥) は、 *Pattern* パラメーター内では引用符で囲まなければなりません。 *Pattern* パラメーターが単純な文字列でない場合、通常はパターン全体を単一引用符で囲まなければなりません。 `[a-z]` などの式において、- (負符号) cml は現在の

照合シーケンスに従った範囲を意味します。照合シーケンスは文字範囲内で使用する等価クラスを定義できます。ファイルを指定しなければ、**grep** は標準入力を想定します。

注**:**

1. 予測できない結果を生ずるので、**grep** コマンドをスペシャル・ファイル上で実行するのは避けてください。入力行には、NULL 文字を使用しないでください。
2. 入力ファイルの終わりには、改行文字を付けてください。
3. 改行文字は、正規表現で一致させることはできません。
4. 複数のフラグを同時に指定できますが、フラグの中には他のフラグを指定変更してしまうものがあります。例えば、**-l** オプションは、他のすべてのフラグよりも優先されます。 **-E** フラグと **-F** フラグを指定すると、最後に指定したフラグが優先されます。

フラグ

項目	説明
-b	各行の前に、その行が見つかったブロック番号を付けます。このフラグを使用すると、ディスク・ブロック番号をコンテキストで見つけるときに便利です。 **-b** フラグは、標準入力またはパイプからの入力と一緒には使用できません。
-c	一致した行の数のみを表示します。
-E	拡張正規表現 (ERE) として指定されたものとして、指定された各パターンを扱います。 ERE の NULL 値は各行に一致します。 **注:** 注: **-E** フラグの付いた、 **grep** コマンドは、エラー・メッセージと使用方法メッセージ、および **-s** フラグの機能が異なるほかは、 **egrep** コマンドと同じ機能が実行されます。
-e *PatternList*	1 つ以上の検索パターンを指定します。このフラグは単純なパターンと同じように機能しますが、パターンが - (負) で始まるときに有効です。パターンは改行文字で区切らなければなりません。NULL パターンは、2 つの隣接する改行文字か、改行文字が後ろに続く引用符 ("¥n) で指定できます。 **-E** フラグも **-F** フラグも指定しなければ、各パターンは基本正規表現 (BRE) のように扱われます。複数の **-e** および **-f** フラグは、**grep** によって受け入れられます。行のマッチング時には、すべての指定されたパターンが使用されますが、評価の順序は指定されません。
-F	指定した各パターンを、正規表現ではなく文字列として扱います。NULL 文字列は各行に一致します。 **注: -F** フラグを指定した、 **grep** コマンドは、 **fgrep** コマンドと同じ機能を実行しますが、エラー・メッセージと使用方法メッセージおよび、 **-s** フラグの機能が異なります。
-f *PatternFile*	検索パターンの入ったファイルを指定します。各パターンは改行文字で区切らなければなりません。また、空の行は NULL パターンと見なされます。 **-E** フラグも **-F** フラグも指定しなければ、各パターンは基本正規表現 (BRE) のように扱われます。
-h	マッチング行を含むファイル名がその行に付加されることを防ぎます。複数のファイルを指定したときに、ファイル名を抑止します。
-H	**-r** または **-R** オプションが指定されていて、タイプがディレクトリーのファイルを参照するシンボリック・リンクがコマンド・ラインに指定されている場合は、**grep** はそのシンボリック・リンクで参照されるディレクトリーのファイルと、ファイル階層でその下にあるすべてのファイルを検索します。
-i	比較するときには、大文字と小文字の区別を無視します。

項目	説明
-l	一致する行を持つファイルの名前のみを (一度) 表示します。各ファイル名は、改行文字で区切られます。標準入力を検索した場合は、パス名 (StandardInput) が戻されます。**-l** フラグを **-c** および **-n** フラグと併用すると、**-l** フラグのみを指定した場合のように動作します。
-L	**-r** または **-R** オプションが指定されていて、タイプがディレクトリーのファイルを参照するシンボリック・リンクがコマンド・ラインに指定されているかまたはファイル階層の横断中に検出された場合は、**grep** は、そのシンボリック・リンクが参照しているディレクトリーのファイルおよびファイル階層でその下にあるすべてのファイルを検索します。 **-H** と **-L** の両方が指定されている場合は、コマンド・ラインの最後に指定されているオプションが有効になります。
-n	各行の前にファイル内の相対行番号を付けます。各ファイルは line 1 から始まり、行カウンターはファイルごとにリセットされます。
-p[*Separator*]	一致する行を含むパラグラフ全体を表示します。パラグラフは、*Separator* パラメーターで指定された、パラグラフ・セパレーターで区切られます。パラグラフ・セパレーターは検索パターンと同じ書式のパターンです。パラグラフ・セパレーターを含んでいる行は、セパレーターとしてのみ使用され、出力には含まれません。デフォルトのパラグラフ・セパレーターはブランク行となります。
-q	行の一致に関係なく、標準出力への書き出しをすべて抑止します。入力行を選択した場合は、状況 0 を戻して終了します。 **-q** フラグを **-c**、**-l**、**-n** フラグと併用すると、**-q** フラグのみを指定した場合のように動作します。
-r	ディレクトリーを反復して検索します。デフォルトにより、続いてそのディレクトリーへリンクします。
-R	ディレクトリーを反復して検索します。デフォルトでは、そのディレクトリーへのリンクは続きません。
-s	存在しないファイルまたは読み取れないファイルに関して通常出力されるエラー・メッセージを抑止します。他のエラー・メッセージは抑止されません。
-u	出力をバッファーしません。
-v	指定したパターンに一致しない行をすべて表示します。
-w	ワードの検索を行います。
-x	指定したパターンに正確に一致し、余分の文字がない行を表示します。
-y	比較を行うときに、大文字と小文字の区別を無視します。
PatternList	検索中に使用される 1 つ以上のパターンを指定します。パターンは **-e** フラグで指定した場合と同様に扱われます。
File	パターンを検索するファイル名を指定します。*File* 変数が与えられない場合は、標準入力を使用します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	一致が見つかりました。
1	一致が見つかりませんでした。
>1	構文エラーが見つかったか、(一致が見つかったとしても) ファイルにアクセスできませんでした。

例

1. パターン・マッチング文字 *、^、?、[、]、¥(、¥)、¥{、¥} のいずれかが入っているパターンを使う場合は、次のように入力します。

   ```
   grep  "^[a-zA-Z]"  pgm.s
   ```

   これにより、pgm.s 内から英文字で始まるすべての行が表示されます。
2. パターンに一致しないすべての行を表示する場合は、次のように入力します。

   ```
   grep   -v  "^#" pgm.s
   ```

   これにより、pgm.s 内で、最初の文字が # (ポンド記号) でないすべての行が表示されます。

3. abc 文字列か、 xyz 文字列に一致するfile1 ファイル内のすべての行を表示するには、次のように入力します。

   ```
   grep   -E  "abc|xyz"  file1
   ```

4. test2 ファイル内の、 $ (ドル記号) を検索するには、次のように入力します。

   ```
   grep ¥¥$ test2
   ```

   シェルが **grep** コマンドに、 ¥$ (1 つの円記号とドル記号) を渡すようにするには、 ¥¥ (2 つの円記号) 文字が必要です。 ¥ (1 つの円記号) は、後ろに続く文字 (この例では $) が正規表現の文字ではなく、通常の文字であることを **grep** コマンドに伝えます。円記号などのエスケープ文字の使用を避けるには、**fgrep** コマンドを使用します。

5. **/tmp** の中を反復して検索しながら IBM という語を持つファイルを検出するが、ディレクトリーを指すリンクの中の反復検索は行わないのであれば、次のように入力します。

   ```
   grep –R IBM /tmp
   ```

   または

   ```
   grep –r -H IBM /tmp
   ```

6. **/tmp** の中を反復して検索しながら IBM という語を持つファイルを検出するが、リンクの中も反復検索するには、次のように入力します。

   ```
   grep –r IBM /tmp
   ```

   または

   ```
   grep -R -L IBM /tmp
   ```

ファイル

項目	説明
/usr/bin/grep	**grep** コマンドが入っています。

関連資料:

358 ページの『egrep コマンド』

545 ページの『fgrep コマンド』

関連情報:

sed コマンド

入出力リダイレクト

---

groups コマンド

目的

グループのメンバーシップを表示します。

構文

groups [*User...*]

説明

デフォルトでは、**groups** コマンドは、現行プロセスのグループ・メンバーシップ情報を標準出力に書き出します。コマンド・パラメーターとして複数のユーザーが指定されている場合には、データベースから各ユーザーのグループ・メンバーシップが表示されます。

groups コマンドは、指定されているユーザーがユーザー・データベースに見つからない場合には、警告メッセージを出した後、パラメーター・リスト上の次のユーザーについての操作を継続します。

セキュリティー

アクセス制御: このプログラムを、トラステッド・コンピューティング・ベースに通常のユーザー・プログラムとしてインストールしてください。

例

パラメーター・リストにリストされているユーザーのグループ・メンバーシップを表示するには、次のように入力します。

```
$ groups sys root lp adm
sys : sys
root : system bin sys security cron audit lp
lp : lp printq
adm : adm
```

ファイル

項目	説明
/usr/bin/groups	**groups** コマンドが入っています。
/usr/ucb/groups	**groups** コマンドへのシンボリック・リンク。
/etc/group	グループ・ファイル。グループ ID が入っています。
/etc/ogroup	前バージョンのグループ・ファイル。
/etc/passwd	パスワード・ファイル。ユーザー ID が入っています。
/etc/opasswd	前バージョンのパスワード・ファイル。

関連資料:

725 ページの『getty コマンド』

関連情報:

login コマンド

setgroups コマンド

---

grpck コマンド

目的

グループ定義が正しいかどうかを確認します。このドキュメントには、AIX **grpck** コマンドと System V **grpck** コマンドの両方についての説明が記載されています。

構文

grpck { **-n** | **-p** | **-t** | **-y** } { **ALL** | *Group* ... }

説明

grpck コマンドは、すべてのグループまたは *Group* パラメーターで指定されたグループに関する定義を検査して、ユーザー・データベース・ファイル内のグループ定義の正しさを検証します。複数のグループを指定する場合は、各グループをスペースで区切る必要があります。

注**:** このコマンドは、メッセージを `stderr` に書き込みます。

フラグを指定して、システムが間違った属性を修正するべきかどうかを指示する必要があります。次の属性が検査されます。

項目	説明
name	グループ名の固有性と構成を検査します。グループ名は、8 バイト以内の固有の文字列でなければなりません。これを + (正符号)、: (コロン)、- (負符号)、~ (ティルド) で始めることはできません。文字列は、コロン (:) を含んでいたり、`ALL` キーワードや `default` キーワードにしたりすることはできません。システム修正はできません。
groupID	グループ ID の固有性と構成を検査します。 ID は null であってはならず、10 進数だけで構成する必要があります。システム修正はできません。
ユーザー	グループ・データベース・ファイルに指定されているユーザーの存在を調べます。エラーを修正するようにシステムに指示すると、ユーザー・データベース・ファイル内で見つからないすべてのユーザーは削除されます。
adms	グループ・データベース・ファイルにグループ管理者として指定されているユーザーの存在を調べます。エラーを修正するようにシステムに指示すると、ユーザー・データベース・ファイル内で見つからないすべての管理者は削除されます。
admin	`/etc/security/group` ファイル内の各グループの admin 属性が有効であるかどうかを調べます。システム修正は利用できません。

一般に、トラステッド・システムのインストール検査の一部として、**sysck** コマンドによって **grpck** コマンドが呼び出されます。また、このコマンドを入力できるのは、root ユーザーまたはセキュリティー・グループのメンバーです。

grpck コマンドは、データベース管理セキュリティー・ファイル (`/etc/passwd.nm.idx`、`/etc/passwd.id.idx`、`/etc/security/passwd.idx`、`/etc/security/lastlog.idx` の各ファイル) が最新の状態であるかどうか、対応するシステム・セキュリティー・ファイルよりも新しい状態であるかどうかを調べます。`/etc/security/lastlog.idx` が `/etc/security/lastlog` よりも新しくなくても、問題はありません。データベース管理セキュリティー・ファイルが最新の状態でない場合は、root ユーザーが **mkpasswd** コマンドを実行するように指示する警告メッセージが表示されます。

フラグ

項目	説明
-n	エラーを修正せずに報告します。
-p	エラーを報告せずに修正します。
-t	エラーを報告して、修正すべきかどうかを尋ねます。
-y	エラーを修正して報告します。

セキュリティー

アクセス制御: このコマンドは、root ユーザーとセキュリティー・グループのメンバーに、実行 (x) アクセス権を与えます。root ユーザーに対する **setuid** コマンドは、**trusted computing base** 属性を備えている必要があります。

アクセスされるファイルは次のとおりです。

モード	ファイル
r	/etc/passwd
r	/etc/security/user
rw	/etc/security/group
rw	/etc/group

監査イベント:

イベント	情報
GROUP_User	user, groups, attribute | error, status
GROUP_Adms	user, groups, attribute | error, status

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. すべてのグループ・メンバーと管理者がユーザー・データベース内に存在することを検査し、すべてのエラーを報告する (ただし、修正はしない) ためには、次のコマンドを入力します。

   ```
   grpck  -n ALL
   ```

2. すべてのグループ・メンバーと管理者がユーザー・データベース内に存在することを検査し、すべてのエラーを修正する (ただし、報告はしない) ためには、次のコマンドを入力します。

   ```
   grpck  -p ALL
   ```

3. `install` グループに関して定義された、グループ名とグループ ID の一意性を調べるには、次のコマンドを入力します。

   ```
   grpck  -n install
   ```

   または、

   ```
   grpck  -t install
   ```

   または、

   ```
   grpck  -y install
   ```

   grpck コマンドでは、グループ名と ID は訂正されません。したがって、**-n**、**-t**、**-y** の各フラグによってグループ名とグループ ID に関する問題は報告されますが、訂正は行われません。

ファイル

項目	説明
/usr/sbin/grpck	**grpck** コマンドが入っています。
/etc/passwd	ユーザーの基本属性が入っています。
/etc/security/user	ユーザーの拡張属性が入っています。
/etc/group	グループの基本属性が入っています。
/etc/security/group	グループの拡張属性が入っています。

System V grpck コマンド

構文

/usr/sysv/bin/grpck

説明

/usr/sysv/bin/grpck コマンドは、すべてのグループに関する定義を検査して、ユーザー・データベース・ファイル内のグループ定義の正しさを検証します。この **/usr/sysv/bin/grpck** コマンドは、/usr/sbin/ 内の既存の **grpck** コマンドの System V バージョンです。このコマンドは、**-n** フラグと **ALL** オプション付きで **/usr/sbin/grpck** コマンドを呼び出します。

終了状況

0 正常終了。

>0 エラーが発生しました。

例

1. すべてのグループ・メンバーと管理者がユーザー・データベース内に存在することを検査し、すべてのエラーを報告する (ただし、修正はしない) ためには、次のコマンドを入力します。

   ```
   /usr/sysv/bin/grpck
   ```

ファイル

/usr/sysv/bin/grpck
grpck コマンドの System V バージョンが入っています。

関連情報:

pwdck コマンド

sysck コマンド

usrck コマンド

セキュリティー

---

grpsvcsctrl コマンド

目的

グループ・サービス・サブシステムを始動します。

構文

grpsvcsctrl { **-a** | **-s** | **-k** | **-d** | **-c** | **-u** | **-t** | **-o** | **-h** }

説明

grpsvcsctrl コマンドは、グループ・サービス・サブシステムを開始します。この制御スクリプトは、グループ・サービスに必要なサブシステムの操作を制御します。これらのサブシステムは、サブシステム・リソース・コントローラー (SRC) の制御下にあり、 **grpsvcs** と呼ばれるサブシステム・グループに属しています。各サブシステムには、デーモンが関連付けられています。操作上の視点から、グループ・サービス・サブシステム・グループは次のように編成されています。

サブシステム
グループ・サービス

サブシステム・グループ
grpsvcs

SRC サブシステム

grpsvcs — **hagsd** デーモンと関連しています。ノード上のサブシステム名は **grpsvcs** です。各ノード上の **grpsvcs** サブシステムは、そのノードが属しているクラスターと関連しています。

デーモン

hagsd — グループ・サービス機能の大部分を提供します。

grpsvcsctrl スクリプトは、通常はコマンド・ラインから実行されません。このスクリプトは、通常、クラスターのインストール時に startup コマンドによって呼び出されます。

grpsvcsctrl スクリプトは、グループ・サービス・サブシステムの操作に関するさまざまな制御を提供します。

- サブシステムの追加、開始、停止、削除およびクリーンアップ
- トレースのオンおよびオフ

これらの機能を実行する前に、スクリプトは、現行のクラスター名を入手します。

サブシステムの追加**: -a** フラグを指定すると、制御スクリプトは **mkssys** コマンドを使用して、SRC にグループ・サービス・サブシステムを追加します。制御スクリプトは次のように動作します。

1. **grpsvcs** サブシステムが停止していることを確認します。
2. このクラスターの **grpsvcs** サブシステムのポート番号をグローバル・オブジェクト・データ・マネージャー (ODM) から入手し、そのポート番号が **/etc/services** ファイルに設定されていることを確認します。ポート番号の有効範囲は 10000 から 10100 まで (両方の値が含まれる) です。
3. **/etc/services** ファイルに入っているサービス名は **grpsvcs.***cluster_name* です。
4. SRC から **grpsvcs** サブシステムを除去します (それが万一存在している場合に備えて)。
5. **grpsvcs** サブシステムを SRC に追加します。クラスター名は、**mkssys** コマンドのデーモン・パラメーターに従って構成されます。

サブシステムの始動**: -s** フラグを指定すると、制御スクリプトは **startsrc** コマンドを使用して、グループ・サービス・サブシステム **grpsvcs** を始動します。

サブシステムの停止**: -k** フラグを指定すると、制御スクリプトは **stopsrc** コマンドを使用して、グループ・サービス・サブシステム **grpsvcs** を停止します。

サブシステムの削除**: -d** フラグを指定すると、制御スクリプトは **rmssys** コマンドを使用して、SRC からグループ・サービス・サブシステムを除去します。制御スクリプトは次のように動作します。

1. **grpsvcs** サブシステムが停止していることを確認します。
2. **rmssys** コマンドを使用して、SRC から **grpsvcs** サブシステムを除去します。
3. ポート番号を **/etc/services** ファイルから除去します。

サブシステムのクリーンアップ**: -c** フラグを指定すると、制御スクリプトは、すべてのシステム区画のグループ・サービス・サブシステムを停止して、SRC から除去します。制御スクリプトは次のように動作します。

1. **stopsrc -g grpsvcs** コマンドを使用して、すべての区画におけるサブシステム・グループ内のサブシステムのすべてのインスタンスを停止します。
2. **rmssys** コマンドを使用して、すべての区画におけるサブシステム・グループ内のサブシステムのすべてのインスタンスを SRC から除去します。

トレースのオン**: -t** フラグを指定すると、制御スクリプトは、**traceson** コマンドを使用して、**hagsd** デーモンに対するトレースをオンにします。

トレースのオフ**: -o** フラグを指定すると、制御スクリプトは、**tracesoff** コマンドを使用して、**hagsd** デーモンに対するトレースをオフ (デフォルト・レベルに戻す) にします。

ロギング**:** グループ・サービス・デーモンは、実行中に、**/var/ha/log** ディレクトリーにあるログ・ファイルにエントリーを書き込むことにより、稼働状況に関する情報を提供します。

各デーモンは、ログ・サイズを事前に設定された行数に制限しています。デフォルトは 5000 行です。この制限に達すると、デーモンは現行ログ・ファイルの名前に文字列 **.bak** を追加して、新規ログを開始します。**.bak** バージョンが既に存在している場合は、現行ログの名前を変更する前に、既存のファイルを除去します。

フラグ

-a サブシステムを追加します。

-s サブシステムを始動します。

-k サブシステムを停止します。

-d サブシステムを削除します。

-c サブシステムをクリーンアップします (つまり、すべてのシステム区画のサブシステムを削除します)。

-u すべての区画からグループ・サービス・サブシステムを除去します。

-t サブシステムのトレースをオンにします。

-o サブシステムのトレースをオフにします。

-h スクリプトの使用方法に関するステートメントを標準出力に書き込みます。

セキュリティー

有効なユーザー ID である **root** を使用して実行する必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 エラーが発生したことを示します。

制限

このスクリプトは、HACMP 環境でのみ有効です。

標準出力

-h フラグが指定されている場合は、このコマンドの使用状況ステートメントが標準出力に書き込まれます。

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

1. グループ・サービス・サブシステムを SRC に追加するには、次のように入力します。
   ```
   grpsvcsctrl -a
   ```
2. グループ・サービス・サブシステムを開始するには、次のように入力します。
   ```
   grpsvcsctrl -s
   ```
3. グループ・サービス・サブシステムを停止するには、次のように入力します。
   ```
   grpsvcsctrl -k
   ```
4. グループ・サービス・サブシステムを SRC から削除するには、次のように入力します。
   ```
   grpsvcsctrl -d
   ```
5. グループ・サービス・サブシステムをクリーンアップするには、次のように入力します。
   ```
   grpsvcsctrl -c
   ```
6. グループ・サービス・デーモン **hagsd** のトレースをオンにするには、次のように入力します。
   ```
   grpsvcsctrl -t
   ```
7. グループ・サービス・デーモン **hagsd** のトレースをオフにするには、次のように入力します。
   ```
   grpsvcsctrl -o
   ```

位置

/opt/rsct/bin/grpsvcsctrl
: **grpsvcsctrl** スクリプトが入っています。

ファイル

/var/ha/log/grpsvcs_nodenum_instnum.cluster_name_
: ノード上の **hagsd** デーモンのログが入っています。

ファイル名の変数は次のとおりです。

nodenum
: デーモンが稼働しているノード番号。

instnum
: デーモンのインスタンス番号。

cluster_name
: デーモンが稼働しているクラスターの名前。

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連情報:

lssrc コマンド

mkssys コマンド

startsrc コマンド

stopsrc コマンド

gssd デーモン

目的

GSS 操作のためのカーネル要求を処理します。

構文

/usr/sbin/gssd

説明

Kerberos 5 などのいくつかの NFS セキュリティー・メソッドが、General Security Services (GSS) と呼ばれる、より一般的なメカニズムのもとに提供されています。 AIX では、GSS サービスは、IBM Network Authentication Service (NAS) ファイルセット内のライブラリーによって提供されます。 NAS は拡張パックで出荷されます。 **gssd** デーモンによって、これらの GSS サービスが NFS サーバー・カーネル・コードに使用可能になります。 **gssd** デーモンを実行しない場合は、Kerberos 5 などの GSS セキュリティー・メソッドを使用して NFS を介してファイルにアクセスしようとしても、失敗します。 **gssd** デーモンは、RPC プログラム番号 400234 を使用して登録します。

gssd デーモンは、以下のシステム・リソース・コントローラー (SRC) コマンドによって始動され、停止されます。

```
startsrc -s gssd
stopsrc -s gssd
```

ファイル

項目	説明
/etc/nfs/hostkey	以下のフォーマットで、**keytab** ファイル場所とホスト・プリンシパルを指定します。 `path to keytab file` `host principal`
/etc/nfs/princmap	以下のフォーマットでホスト・プリンシパルへのマッピングを指定します。 `principal1 alias1 alias2 alias3` `principal2 alias1` 別名は、IP アドレスまたはホスト名とすることができます。プリンシパルは、Kerberos が保持しているホスト・キーと一致する必要があります。

h

以下の AIX コマンドは文字 h で始まります。

ha.vsd コマンド

目的

リカバリー可能仮想共用ディスク・サブシステムの **rvsd** デーモンのアクティビティーを照会および管理します。

構文

ha.vsd

{**adapter_recovery [on** | **off]** | **debug [off]** | **mksrc** | **query** | **quorum** *n* | **qsrc** | **refresh [noquorum]** | **reset** | **reset_quorum** | **rmsrc** | **start** | **stop** | **trace [off]**}

説明

このコマンドを使用して、リカバリー可能仮想共用ディスク・サブシステムの情報の表示、クォーラムに必要なノードの数の変更、およびサブシステムの状況の変更を行います。

フラグ

-a すべての仮想共用ディスクを指定します。

-v *vsd_name_list*
コンマで区切られた、1 つ以上の仮想共用ディスク名を指定します。

-n *node_list*
コンマで区切られた、1 つ以上のノード番号を指定します。

パラメーター

adapter_recovery [on | off]
通信アダプター・リカバリーを使用可能または使用不可にします。デフォルトは **on** です。

このオペランドを有効にするためには、リカバリー可能仮想共用ディスク・サブシステムを再始動する必要があります。

debug [off]
debug を指定して、リカバリー可能仮想共用ディスク・サブシステムの標準出力および標準エラーをコンソールに転送し、リカバリー可能仮想共用ディスク・サブシステムがエラーで終了する場合は、再作成されないようにします。(現行のコンソールを判別するためには **lscons** コマンドを使用できます。)

このオペランドを有効にするためには、リカバリー可能仮想共用ディスク・サブシステムを再始動する必要があります。

デバッグが開始され、リカバリー可能仮想共用ディスク・サブシステムが再始動すると、 **ha.vsd trace** が発行されてトレースが開始されます。

このオペランドは、IBM サービス技術員の説明に基づいて使用してください。

注**:** ノードがブートされているときのデフォルトは、標準出力および標準エラーをコンソールに経路指定する、となっています。デバッグが終了すると、標準出力および標準エラーは **/dev/null** に経路指定され、その後のトレース・メッセージはすべて失われます。**ha.vsd qsrc** を発行することで、デバッグが開始されているかを判別することができます。デバッグが開始されていると、戻り値は次のようになります。

```
action = "2"
```

mksrc リカバリー可能仮想共用ディスク・サブシステムを作成するには **mkssys** を使用します。

照会 リカバリー可能仮想共用ディスク・サブシステムの現状を詳細に表示します。

quorum *n*

クォーラムの値を設定します。このクォーラムは仮想共用ディスクが活動化される前にグループを結合しなければならないノードの総数です。通常クォーラムは、RSCT ピア・ドメイン内で仮想共用ディスク・ノードとして定義されたノードの過半数として定義されますが、このコマンドはその定義を上書きすることができます。

このコマンドを使用するときは、リカバリー可能仮想共用ディスク・サブシステムが活動状態でなければなりません。これは永続的な変更ではありません。

qsrc リカバリー可能仮想共用ディスク・デーモンの System Resource Controller (SRC) 構成を表示します。

refresh [noquorum]

refresh コマンドを使用して、すべての実行中のリカバリー可能仮想共用ディスク・サブシステムに対してリフレッシュ・プロトコルを非同期的に開始します。**noquorum** を指定していない限り、クォーラムはリフレッシュが起こる前にリセットされます。**ha.vsd query** を使用して完了したかを検査してください。以下の項目はデバイス・ドライバーでリフレッシュされたものです。

1. 追加または削除されたノード
2. 追加または削除された仮想共用ディスク
3. 仮想共用ディスクに対して変更された size_in_MB 属性

reset リカバリー可能仮想共用ディスク・サブシステムを停止および再始動します。

reset_quorum

デフォルトのクォーラムをリセットします。

rmsrc リカバリー可能仮想共用ディスク・サブシステムを除去するには、**rmssys** を使用します。

start リカバリー可能仮想共用ディスク・サブシステムを始動します。

stop リカバリー可能仮想共用ディスク・サブシステムを停止します。

trace [off]

リカバリー可能仮想共用ディスク・サブシステムのトレースを要求または停止します。このコマンド発行時は、リカバリー可能仮想共用ディスク・サブシステムが活動状態でなければなりません。

このオペランドが意味があるのは、**debug** オペランドを使用して、コンソールへ標準出力および標準エラーを送信し、リカバリー可能仮想共用ディスク・サブシステムが再始動した後です。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

終了状況

0 コマンドが正常終了したことを示します。

nonzero
エラーが発生したことを示します。

制限

このコマンドはピア・ドメイン内のオンラインのノードから発行する必要があります。オンラインのピア・ドメインに移動するには **startrpdomain** コマンドを使用します。既存のピア・ドメイン内で、オンラインの特定ノードに移動するには **startrpnode** コマンドを使用します。RSCT ピア・ドメインの作成と管理の詳細については、「*RSCT* 管理ガイド」を参照してください。

例

1. リカバリー可能仮想共用ディスク・サブシステムを停止し、再始動するには、次のように入力します。

   ```
   ha.vsd reset
   ```

   システムは次のメッセージを戻します。

   ```
   Waiting for the rvsd subsystem to exit.
   rvsd subsystem exited successfully.
   Starting rvsd subsystem.
   rvsd subsystem started PID=xxx.
   ```

2. クォーラムを RSCT ピア・ドメインの 5 つのノードへ変更するには、次のように入力します。

   ```
   ha.vsd quorum 5
   ```

   システムは次のメッセージを戻します。

   ```
   Quorum has been changed from 8 to 5.
   ```

3. rvsd サブシステムを照会するには次のように入力します。

   ```
   ha.vsd query
   ```

   システムは次のようなメッセージを表示します。

   ```
   Subsystem         Group            PID     Status
         rvsd              rvsd             18320   active
         rvsd(vsd): quorum= 9/4, active=1, state=idle, isolation=member,
                    NoNodes=10, lastProtocol=nodes_failing,
                    adapter_recovery=on, adapter_status=up,
                    RefreshProtocol has never been issued from this node,
                    Running function level 4.1.0.0.
   ```

   それぞれの意味は次のとおりです。

   quorum
   クォーラムは仮想共用ディスクが活動化される前にグループを結合しなければならないノードまたはサーバー・ノードの総数です。上記のシステム出力で、**クォーラム 9/4** は、ノードの総数 (9) およびサーバー・ノードの数 (4) を示しています。

   active 結合されているグループの活動化状況を示します。

   0: グループは活動状態にありません (クォーラムは条件が満たされていません)。

   1: グループはアクティブであり共用ディスクは活動状態にあります。

   state 実行中の現行プロトコルを示します。

   isolation
   グループ・メンバーシップ状況を示します。

   isolated:
   グループの「結合化」は計画されていません。

proposed:
グループの「結合化」が計画されています。

member:
グループのメンバー (プロバイダー) です。

NoNodes
グループを結合したノードの数を示します。

lastProtocol
グループ全体に渡って実行された最後のプロトコルを示します。

adapter_recovery
通信アダプター・リカバリー・サポートを次のように示します。

on: アダプター・リカバリーは使用可能です。

off: アダプター・リカバリーは使用不可です。

adapter_status
通信アダプター状況を次のように示します。

up: アダプターは作動しています。

down:
アダプターは停止しています。

unknown:
アダプター状況は認識できません。

RefreshProtocol ...
リフレッシュ・プロトコルがこのノードから発行されているかを示します。発行されていれば、成功またはエラーの日付および時刻が表示されます。

Running function level
サブシステムが実行されている機能レベルが、バージョン、リリース、変更、修正レベル形式 (vrmf) で示されます。(サブシステムの機能レベルを低いままで共存させていると、機能レベルを削減して実行する場合に制限が生じることがあります。)

位置

/opt/rsct/vsd/bin/ha.vsd

関連資料:

755 ページの『ha_vsd コマンド』

---

ha_star コマンド

目的

高可用性イベントを処理します。

構文

ha_star [-C]

説明

ha_star コマンドは、汎用性のある高可用性処理コマンドです。ファームウェアが CPU の障害の前触れを報告すると、オペレーティング・システムは、 **/etc/rc.ha_star** を使用して、このコマンドを自動的に起動します。

フラグを指定しないで **ha_star** を起動する場合は、新規のイベントしか処理されません。新規のイベントが見つからないと、 **ha_star** は終了します。

ha_star は、実行中はすべての新規イベントを処理します。 **ha_star** が既にあるイベントを処理している最中に到着したイベントも、処理の対象になります。 **ha_star** のインスタンスは、一度に 1 つしか実行できません。 2 番目のインスタンスを起動すると、**ha_star** は終了します。

オペレーティング・システムは、高可用性イベントが報告されると、 **ha_star** を起動します。イベント処理は、失敗したり、取り消されたりする (例えばシグナルによって) 場合があります。打ち切られたり取り消されたりしたイベントは、カーネル内のメモリーに保存されます。異常終了の原因が取り除かれれば、イベント処理を再試行することができます。このときは、システム管理者は手動で **ha_star** を起動します。

ha_star コマンドは、エラーまたは障害のエラー・ログ・エントリーを生成します。

イベント型による記述

ha_star コマンドは、オペレーティング・システムによって起動され、プロセッサー障害イベントの前触れが検出されると、CPU の割り当てを解除します。スレッドの中には、割り当てが解除される CPU にバウンドしたままのものがあるので、この割り当て解除は失敗する場合もあります。場合によっては、システム管理者が、割り当て解除が失敗する原因となった状態を改善することができます。例えば、最後の論理 CPU にバウンドされているスレッドを使用するアプリケーションを識別して、停止することができるかもしれません。

-C フラグは、再開される高可用性イベントが CPU の割り当て解除イベントであることを示しています。

フラグ

項目	説明
-C	再開されるイベントが CPU の割り当て解除となるように指定します。

ファイル

項目	説明
/usr/sbin/ha_star	**ha_star** コマンドが入っています。

関連情報:

動的なプロセッサーの割り当て解除

使用可能化コマンド

---

ha_vsd コマンド

目的

リカバリー可能仮想共用ディスク・サブシステムを始動および再始動します。これには仮想共用ディスクの構成およびリカバリー可能サブシステムの活動化が含まれます。

構文

ha_vsd [reset]

説明

このコマンドを使用して、リカバリー可能仮想共用ディスクのソフトウェア・インストール後に始動します。あるいは、**reset** オプション指定でこのプログラムを停止および再始動します。

フラグ

-a すべての仮想共用ディスクを指定します。

-v *vsd_name_list*
コンマで区切られた、1 つ以上の仮想共用ディスク名を指定します。

-n *node_list*
コンマで区切られた、1 つ以上のノード番号を指定します。

パラメーター

reset リカバリー可能仮想共用ディスク・サブシステムを停止および再始動します。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 エラーが発生したことを示します。

制限

このコマンドはピア・ドメイン内のオンラインのノードから発行する必要があります。オンラインのピア・ドメインに移動するには **startrpdomain** コマンドを使用します。既存のピア・ドメイン内で、オンラインの特定ノードに移動するには **startrpnode** コマンドを使用します。RSCT ピア・ドメインの作成と管理の詳細については、「*RSCT* 管理ガイド」を参照してください。

例

リカバリー可能仮想共用ディスク・サブシステムを停止し、再始動するには、次のように入力します。

```
ha_vsd reset
```

位置

/opt/rsct/vsd/bin/ha_vsd

関連資料:

751 ページの『ha.vsd コマンド』

haemd デーモン

目的

リソース・モニターが更新するリソース変数のインスタンスを監視し、イベントを生成してそれをクライアント・プログラムに報告します。

構文

haemd

説明

haemd (イベント・マネージャー) デーモンは、リソース・モニターが更新するリソース変数のインスタンスを監視し、イベントを生成してそれをクライアント・プログラムに報告します。

haemd デーモンの 1 つのインスタンスがクラスターの各ノードで実行されます。 **haemd** デーモンは、システム・リソース・コントローラー (SRC) の制御下にあります。

このデーモンは SRC の制御下にあるため、コマンド・ラインから直接始動することはできません。これは通常、**emsvcsctrl** コマンドによって始動されます。このデーモンを直接、始動または停止しなければならないときは、**emsvcsctrl** コマンドを使用します。

SRC が **haemd** デーモンを作成するとき、実際に始動されるプログラムは **haemd_HACMP** です。**haemd_HACMP** プログラムは、そのデーモンに必要な情報を収集した後、**haemd** プログラムを実行します。言い換えれば、**haemd_HACMP** プログラムは、SRC によって作成されたプロセスの中で、**haemd** プログラムと置き換えられるということです。

イベント・マネージャー・デーモンの詳細については、**emsvcsctrl** コマンドを参照してください。

実装上の固有な条件

このデーモンは、AIX 用の Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

位置

/opt/rsct/bin/haemd
　　haemd デーモンの位置

関連資料:

379 ページの『emsvcsctrl コマンド』

『haemd_HACMP コマンド』

haemd_HACMP コマンド

目的

イベント・マネージャー・デーモンの始動プログラムです。

構文

haemd_HACMP [**-d** *trace_arg*]

説明

haemd_HACMP プログラムは **haemd** デーモンの始動プログラムです。 **emsvcsctrl** コマンドでイベント管理サブシステムを SRC (システム・リソース・コントローラー) 内に構成時に、**haemd_HACMP** が、始動するプログラムとして指定されます。

このプログラムは SRC によってのみ始動できます。イベント管理サブシステムを始動するには **emsvcsctrl** コマンドを使用します。

フラグ

-d *trace_arg*
: このフラグは、IBM サポートから指導があった場合にのみ使用します。指定できるトレース引数は、**reg** と **dinsts** を除いて **haemtrcon** コマンドと同じです。このフラグを使用するには、**-a** フラグ指定の **chssys** コマンドを使用して、SRC 内の **emsvcs** サブシステム定義を変更する必要があります。次に、デーモンを停止してから再始動する必要があります。

制限

このコマンドは、HACMP 環境でのみ有効です。

実装上の固有な条件

このスクリプトは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

位置

/opt/rsct/bin/haemd_HACMP
: **haemd_HACMP** プログラムの位置

関連資料:

379 ページの『emsvcsctrl コマンド』
757 ページの『haemd デーモン』
765 ページの『haemtrcon コマンド』

---

haemqvar コマンド

目的

リソース変数を照会します。

構文

haemqvar [**-H** *domain* | **-S** *domain*] [**-c** | **-d** | **-i**] [**-f** *file*] [**-h**] [*class var rsrcID* ["]]

説明

haemqvar コマンドはイベント管理サブシステムにリソース変数についての情報を照会します。デフォルトでは、コマンドは現行 SP ドメイン (すなわち SP_NAME 環境変数で定義された現行 SP システム区画) 内のすべてのリソース変数の定義を標準出力に書き込みます。 SP_NAME が設定されていなければ、デフォルトのシステム区画が使用されます。 -S フラグを使用して別の SP ドメイン (システム区画) を指定できます。 HACMP ドメイン内の変数を照会するには、-H フラグを使用します。 SP ドメインの場合、ド

メイン・フラグ引数はシステム区画名です。 HACMP ドメインの場合、ドメイン・フラグ引数は HACMP クラスター名です。 -H フラグが指定されていれば、コマンドは HACMP/ES クラスター内のノードの 1 つで実行する必要があります。

それぞれのリソース変数定義について次の情報が報告されます。

- 変数名
- 値の型
- データ型
- SBS フォーマット (データ型が SBS (構造化バイト・ストリング) の場合)
- 初期値
- クラス
- ロケーター
- 変数説明
- リソース ID とその説明
- デフォルト式 (定義されている場合) とその説明

このコマンドのデフォルトの動作では大量の出力を生成する可能性があるため、標準出力をファイルにリダイレクトしてください。

-d フラグが指定されていれば、リソース変数名と簡略説明が 1 行に 1 変数名と説明という形で標準出力に書き込まれます。

-c フラグが指定されていれば、すべてのリソース変数インスタンスの現在値が 1 行に 1 変数という形で標準出力に書き込まれます。出力行には、リソース変数インスタンスの位置 (ノード番号)、リソース変数名、インスタンスのリソース ID、およびリソース変数インスタンス値が入ります。リソース変数が SBS (構造化バイト・ストリング) データ型であれば、各 SBS フィールドの値が報告されます。

-i フラグは、変数インスタンス値が現在値ではなく最後に認識された値であることを除いて、-c フラグと同じ情報を報告します。 -i フラグによって、存在するリソース変数インスタンスを判別できます。

-c フラグと -i フラグのどちらも、リソース変数インスタンスの情報を取得している間にエラーが発生すると、出力行には、エラー・メッセージ、シンボリック・エラー・コード、エラーの発生場所 (判別できた場合)、リソース変数名、およびリソース ID が入ります。

特定のリソース変数についての情報を取得する場合は、class、var、および rsrcID オペランドを指定します。これらのオペランドは複数回 (繰り返して) 使用できるため、複数のリソース変数を指定できます。さらに var と rsrcID オペランドはワイルドカードによる指定が許されます。これにより多数のリソース変数に一致させることができます。 null 文字列のオペランドまたはアスタリスクは引用符で囲まなければならないことに注意してください。

class が null 文字列でなければ、var および rsrcID 引数でさらに限定された上で、指定されたクラス内のすべての変数が照会の対象となります。 class が null 文字列であれば、var および rsrcID 引数でさらに限定された上で、すべてのクラスの変数が照会の対象となります。 var 引数をワイルドカードを使用して指定する場合、次のいずれかになります。

1. 変数名を null 文字列として指定する
2. 名前をあるコンポーネントの後ろで切り捨てる

リソース変数名が 1 番目のようにワイルドカード指定されると、 class および rsrcID 引数でさらに限定された上で、すべてのリソース変数が照会の対象となります。リソース変数名が 2 番目のようにワイルドカード指定されると、 class および rsrcID 引数でさらに限定された上で、高位の (左端の) コンポーネントが var 引数と一致するすべてのリソース変数が照会の対象となります。

class および var 引数に指定された変数の中で rsrcID 引数と一致するすべてのリソース変数インスタンス (-c または -i フラグのどちらも指定されていない場合はリソース変数の定義) が照会の対象となります。

-c または -i フラグのどちらも指定されていない場合、rsrcID 引数はリソース ID エレメント名をセミコロンで区切ったリストです。 -c または -i フラグが指定されている場合、rsrcID 引数は名前と値のペアをセミコロンで区切ったリストです。名前と値のペアは、リソース ID のエレメント名、続いて等号、その後ろにリソース ID エレメント値で構成します。エレメント値は単一の値、値の範囲、コンマで区切った単一の値のリスト、またはコンマで区切った範囲のリストで指定します。範囲は a-b の形式で、リソース ID の整数型のエレメントのみ指定できます (型の情報は変数定義から取得できます)。リソース ID にブランクを入れることはできません。

リソース ID のエレメントはエレメントの値をアスタリスクにしてワイルドカードによる指定ができます。そのエレメントが含まれるように定義された変数のみ、また、rsrcID 引数に指定されたエレメントのみ照会の対象となります。リソース ID のエレメントに、名前と値のペア (または定義の照会であれば名前のみ) ではなく、アスタリスクが入っている場合、少なくとも残りの指定されたエレメントが含まれるように定義されたすべての変数が照会の対象となります。リソース ID がアスタリスクのみで構成されていれば、リソース ID 全体がワイルドカード指定されたことになります。 class および var 引数でさらに限定された上で、すべてのリソース変数のすべてのインスタンスが照会の対象となります。

rsrcID 引数は、中にセミコロンまたはアスタリスクがある場合は全体を引用符で囲む必要があることに注意してください。

class、var、および rsrcID オペランドはコマンド引数として指定する代わりに、1 行に 1 セットのオペランドの形式でファイルに入れることもできます。 -f フラグを使用してコマンドに渡すファイル名を指定します。 -f フラグが使用されると、コマンドに指定されたオペランドはすべて無視されます。ファイル内で、null 文字列は 2 つの隣接する二重引用符で指定します。全体がワイルドカード指定されたリソース ID は、単独のアスタリスク (*) または二重引用符で囲まれたアスタリスク ("*") です。引数は各行でブランク・スペースまたはタブで区切る必要があります。

rsrcID 引数にワイルドカードを使用した例を次に示します。これらの例では class および var 引数は null 文字列と想定しています。 class または var 引数のどちらか、あるいは両方が null 文字列でない場合は、その指定によって照会の対象が限定されます。最初の 3 つの例では、すべての変数をリソース ID で示し、エレメント NodeNum、VG、LV を含むように定義されており、これらのエレメントのみが一致しています。

1. この例では 1 つのインスタンスのみ一致します。

   ```
   NodeNum=5;VG=rootvg;LV=hd4
   ```

2. この例では、それぞれのノードから 1 つのインスタンスが一致します。

   ```
   NodeNum=*;VG=rootvg;LV=hd4
   ```

3. この例では、NodeNum、VG、LV が含まれるように定義されたリソース ID のリソース変数のすべてのインスタンスが一致します。

   ```
   NodeNum=*;VG=*;LV=*
   ```

4. この例では、リソース ID にエレメント NodeNum のみ含まれるように定義されたすべての変数が一致します。一致するインスタンスはノード 9 に関連付けられているものです。

```
NodeNum=9
```

5. この例では、前の例と同じ変数のセットが一致しますが、各変数のすべてのインスタンスが一致します。

```
NodeNum=*
```

6. この例では、リソース ID がエレメント NodeNum と VG を含むように定義されたすべての変数と、ゼロまたは 1 個以上の他のエレメントも一致します。一致するインスタンスはノード 9 に関連付けられているものです。

```
NodeNum=9;VG=*;*
```

7. この例では、リソース ID にエレメント NodeNum を含むように定義されたすべての変数と、ゼロまたは 1 個以上の他のエレメントも一致します。変数のすべてインスタンスが一致します。

```
NodeNum=*;*
```

照会するリソース変数の指定に柔軟性が与えられている場合、どのリソース変数インスタンスまたはリソース変数定義も一致しないこともあります。一致するものがなければ、適切なエラー情報が上述の形式または次に示す形式で報告されます。

class、var、または rsrcID 引数の指定でエラーがあれば、出力行にはエラー・メッセージ、シンボリック・エラー・コードと指定されたクラス名、リソース変数名、およびリソース ID が含まれます。

フラグ

-H *domain*
: *domain* で指定された HACMP ドメイン内でリソース変数を照会します。

-S *domain*
: *domain* で指定された SP ドメイン内でリソース変数を照会します。

-c リソース変数の現行値を照会します。

-d リソース変数定義を照会し、簡略形式の出力を生成します。

-i リソース変数のインスタンスを照会します。

-f *file* *file* に指定されたリソース変数を照会します。

-h 使用方法に関する記述を表示します。

パラメーター

class リソース変数クラスの名前または null 文字列を指定します。

var リソース変数の名前または null 文字列を指定します。

rsrcID リソース ID またはアスタリスクを指定します。

セキュリティー

このコマンドを実行するには、ユーザーは root 権限と SDR への書き込みアクセス権限を持っている必要があります。

コントロール・ワークステーション上で実行する必要があります。このコマンドを実行する前に、SP_NAME 環境変数を適切なシステム区画名に設定しておく必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 エラーが発生したことを示します。これと一緒に、エラーの原因を示す 1 つ以上のエラー・メッセージが出力されます。

制限

このコマンドは PSSP 環境のみで有効です。

標準出力

コマンドが正常終了すると、次の通知メッセージが書き込まれます。

```
Reading Event Management data for partition syspar_name

CDB=new_EMCDB_file_name Version=EMCDB_version_string
```

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

1. 現行クラスター内のすべてのリソース変数の定義を取得し、出力をファイルに書き込むには、次のように入力します。

   ```
   haemqvar -H HAcluster > vardefs.out
   ```

2. クラスター名が HAcluster の HACMP クラスター内にあり、リソース ID に VG というエレメントが含まれているすべてのリソース変数の簡略形式のリストを取得するには、次のように入力します。

   ```
   haemqvar -H HAcluster -d "" "" "VG;*"
   ```

3. リソース ID に VG と NodeNum というエレメントのみ含まれているリソース変数を取得するには、次のように入力します。

   ```
   haemqvar -H HAcluster -d "" "" "VG;NodeNum"
   ```

位置

/opt/rsct/bin/haemqvar
haemqvar コマンドの位置

ファイル

/opt/rsct/install/config/haemloadlist
イベント管理サブシステムのデフォルトの構成データが入っています。

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

---

haemtrcoff コマンド

目的

イベント・マネージャー・デーモンのトレースをオフにします。

構文

haemtrcoff -s *subsys_name* **-a** *trace_list*

説明

haemtrcoff コマンドを使用して、イベント・マネージャー・デーモンの指定されたアクティビティーのトレースをオフにします。トレース出力はシステム区画のイベント管理トレース・ログに置かれます。

フラグ

-s *subsys_name*
: イベント管理サブシステムの名前を指定します。ノード上ではこれは emsvcs です。この引数は指定しなければなりません。

-a *trace_list*
: トレース引数のリストを指定します。各引数はトレースをオフにするアクティビティーのタイプを指定します。少なくとも 1 つの引数を指定する必要があります。複数の引数を指定するときは、各引数をコンマで区切る必要があります。リストにブランクを入れてはなりません。

パラメーター

次のトレース引数を指定できます。

init イベント・マネージャー・デーモンの初期化のトレースを停止します。

config
: 構成ファイルからの情報のダンプを停止します。

insts デーモンが処理するリソース変数インスタンスのトレースを停止します。

rmctrl リソース・モニター制御のトレースを停止します。

cci クライアント通信 (内部) インターフェースのトレースを停止します。

emp イベント・マネージャー・プロトコルのトレースを停止します。

obsv リソース変数監視のトレースを停止します。

evgn イベント生成と通知のトレースを停止します。

reg イベント登録と登録解除のトレースを停止します。

pci ピア通信 (内部) インターフェースのトレースを停止します。

msgs デーモンが受け取るメッセージおよびデーモンが発行するメッセージのトレースをすべて停止します。

照会 デーモンが処理する照会のトレースを停止します。

gsi グループ・サービス (内部) インターフェースのトレースを停止します。

eval 式の評価のトレースを停止します。

rdi 高信頼性デーモン (内部) インターフェースのトレースを停止します。

sched 内部スケジューラーのトレースを停止します。

shm 共用メモリー管理アクティビティーのトレースを停止します。

all すべてのアクティビティーのトレースを停止します。

all_but_msgs
: メッセージ以外のすべてのアクティビティーのトレースを停止します。メッセージ・アクティビティーは msgs 引数によって定義されます。

セキュリティー

このコマンドを実行するには、ユーザーは root 権限と SDR への書き込みアクセス権限を持っている必要があります。

コントロール・ワークステーション上で実行する必要があります。このコマンドを実行する前に、SP_NAME 環境変数を適切なシステム区画名に設定しておく必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 エラーが発生したことを示します。これと一緒に、エラーの原因を示す 1 つ以上のエラー・メッセージが出力されます。

制限

通常の操作ではこのコマンドは使用しないでください。このコマンドは、IBM サポートから指導があった場合にのみ使用します。これはデバッグのための情報を提供し、イベント管理サブシステムまたはシステム区画内で実行されているもののパフォーマンスを低下させる可能性があります。

標準出力

コマンドが正常終了すると、次の通知メッセージが書き込まれます。

```
Reading Event Management data for partition syspar_name

CDB=new_EMCDB_file_name Version=EMCDB_version_string
```

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

1. クラスター・ノードの 1 つに関してイベント管理サブシステムのすべてトレースをオフにするには、そのノードにログインして次のように入力します。

   ```
   haemtrcoff -s emsvcs -a all
   ```

2. あるクラスター・ノードに関してイベント管理サブシステムの初期化と構成のすべてのトレースをオフにするには、そのノードにログインして次のように入力します。

   ```
   haemtrcoff -s emsvcs -a init,config
   ```

位置

/opt/rsct/bin/haemtrcoff
haemtrcoff コマンドの位置

ファイル

/var/ha/log/em.trace.cluster_name
クラスター名が *cluster_name* のクラスターに関する、**haemd** デーモンのトレース・ログが入っています。

/var/ha/log/em.msgtrace.cluster_name
クラスター名が *cluster_name* のクラスターに関する、イベント・マネージャー・デーモンからのメッセージ・トレース出力が入っています。

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

関連資料:

『haemtrcon コマンド』

757 ページの『haemd デーモン』

379 ページの『emsvcsctrl コマンド』

---

haemtrcon コマンド

目的

イベント・マネージャー・デーモンのトレースをオンにします。

構文

haemtrcon -s *subsys_name* **-a** *trace_list*

説明

haemtrcon コマンドを使用して、イベント・マネージャー・デーモンの指定されたアクティビティーのトレースをオンにします。トレース出力はシステム区画のイベント管理トレース・ログに置かれます。**regs**、**dinsts**、**iolists**、および **olists** パラメーターは、使用されると一回限りのトレースを実行します。指定された情報はトレース・ログに入れられますが、それ以上のトレースは行われません。

フラグ

-s *cluster_name*
: イベント管理サブシステムの名前を指定します。ノード上では *cluster_name* は **emsvcs** です。このフラグとパラメーターは指定しなければなりません。

-a *trace_list*
: トレース・パラメーターのリストを指定します。各パラメーターはトレースをオンにするアクティビティーのタイプを指定します。少なくとも 1 つのパラメーターを指定する必要があります。複数のパラメーターを指定するときは、各パラメーターをコンマで区切る必要があります。リストにブランクを入れてはなりません。

パラメーター

次のトレース・パラメーターを指定できます。

init イベント・マネージャー・デーモンの初期化をトレースします。

config
: 構成ファイルからの情報をダンプします。

insts デーモンが処理するリソース変数インスタンスをトレースします。

rmctrl リソース・モニター制御をトレースします。

cci クライアント通信 (内部) インターフェースをトレースします。

emp イベント・マネージャー・プロトコルをトレースします。

obsv リソース変数監視をトレースします。

evgn イベント生成と通知をトレースします。

reg イベント登録と登録解除をトレースします。

pci ピア通信 (内部) インターフェースをトレースします。

msgs デーモンが受け取るメッセージおよびデーモンが発行するメッセージをすべてトレースします。

照会 デーモンが処理する照会をトレースします。

gsi グループ・サービス (内部) インターフェースをトレースします。

eval 式の評価をトレースします。

rdi 高信頼性デーモン (内部) インターフェースをトレースします。

sched 内部スケジューラーをトレースします。

shm 共用メモリー管理アクティビティーをトレースします。

all すべてのアクティビティーをトレースします。

all_but_msgs
メッセージ以外のすべてのアクティビティーのトレースを停止します。メッセージ・アクティビティーは msgs 引数によって定義されます。

regs 現在登録されているイベントをトレースします。

dinsts デーモンが認識しているすべてのリソース変数インスタンスをトレースします。

iolists 即時監視リストをトレースします。

olists 監視リストをトレースします。

制限

通常の操作ではこのコマンドは使用しないでください。このコマンドは、IBM サポートから指導があった場合にのみ使用します。これはデバッグのための情報を提供し、イベント管理サブシステムまたはシステム区画内で実行されているもののパフォーマンスを低下させる可能性があります。

実装上の固有な条件

このコマンドは、Reliable Scalable Cluster Technology (RSCT) ファイルセットの一部です。

例

1. クラスター・ノードの 1 つに関してイベント管理サブシステムのすべてのトレースをオンにするには、そのノードにログインして次のように入力します。

   ```
   haemtrcon -s emsvcs -a all
   ```

2. あるクラスター・ノードに関してイベント管理サブシステムの初期化と構成のすべてのトレースをオンにするには、そのノードにログインして次のように入力します。

   ```
   haemtrcon -s emsvcs -a init,config
   ```

位置

/opt/rsct/bin/haemtrcon
haemtrcon コマンドの位置

関連資料:

762 ページの『haemtrcoff コマンド』

757 ページの『haemd デーモン』

379 ページの『emsvcsctrl コマンド』

haemunlkrm コマンド

目的

リソース・モニターのアンロックと始動を行います。

構文

haemunlkrm -s *subsys_name* **-a** *resmon_name*

説明

イベント管理デーモンがリソース・モニターの始動を 2 時間の間に 3 回失敗するか、 2 時間の間に *n* 回リソース・モニターのインスタンスに正常に接続できた場合、リソース・モニターは「ロックされた状態」であり、それ以上始動またはそのインスタンスへの接続は試行されません。 HACMP/ES クラスター内では *n* は **3** です。失敗の原因が判別されて問題が訂正された後、**haemunlkrm** コマンドを使用してリソース・モニターをアンロックし、リソース・モニターの始動またはリソース・モニター・インスタンスへの接続を試行できます。

イベント・マネージャー・デーモンのステータス (**lssrc** コマンドによって表示される) は、リソース・モニターがロックされているかどうかを示します。

フラグ

-s *subsys_name*
: イベント管理サブシステムの名前を指定します。ノード上では *subsys_name* は **emsvcs** です。このフラグとパラメーターは指定しなければなりません。

-a *resmon_name*
: アンロックおよび始動するリソース・モニターの名前を指定します。

パラメーター

次のトレース・パラメーターを指定できます。

init イベント・マネージャー・デーモンの初期化をトレースします。

config
: 構成ファイルからの情報をダンプします。

insts デーモンが処理するリソース変数インスタンスをトレースします。

rmctrl リソース・モニター制御をトレースします。

cci クライアント通信 (内部) インターフェースをトレースします。

emp イベント・マネージャー・プロトコルをトレースします。

obsv リソース変数監視をトレースします。

evgn イベント生成と通知をトレースします。

reg イベント登録と登録解除をトレースします。

pci ピア通信 (内部) インターフェースをトレースします。

msgs デーモンが受け取るメッセージおよびデーモンが発行するメッセージをすべてトレースします。

照会　デーモンが処理する照会をトレースします。

gsi　グループ・サービス (内部) インターフェースをトレースします。

eval　式の評価をトレースします。

rdi　高信頼性デーモン (内部) インターフェースをトレースします。

sched　内部スケジューラーをトレースします。

shm　共用メモリー管理アクティビティーをトレースします。

all　すべてのアクティビティーをトレースします。

all_but_msgs
メッセージ以外のすべてのアクティビティーのトレースを停止します。メッセージ・アクティビティーは msgs 引数によって定義されます。

regs　現在登録されているイベントをトレースします。

dinsts　デーモンが認識しているすべてのリソース変数インスタンスをトレースします。

iolists　即時監視リストをトレースします。

olists　監視リストをトレースします。

セキュリティー

このコマンドを実行するには、ユーザーは root 権限と SDR への書き込みアクセス権限を持っている必要があります。

コントロール・ワークステーション上で実行する必要があります。このコマンドを実行する前に、SP_NAME 環境変数を適切なシステム区画名に設定しておく必要があります。

終了状況

0　コマンドが正常終了したことを示します。

1　エラーが発生したことを示します。これと一緒に、エラーの原因を示す 1 つ以上のエラー・メッセージが出力されます。

制限

通常の操作ではこのコマンドは使用しないでください。このコマンドは、IBM サポートから指導があった場合にのみ使用します。これはデバッグのための情報を提供し、イベント管理サブシステムまたはシステム区画内で実行されているもののパフォーマンスを低下させる可能性があります。

標準出力

コマンドが正常終了すると、次の通知メッセージが書き込まれます。

```
Reading Event Management data for partition syspar_name

CDB=new_EMCDB_file_name Version=EMCDB_version_string
```

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

1. この例はノード上のリソース・モニターをアンロックしようとするものです。

lssrc コマンドの出力がプログラム・リソース・モニター **IBM.PSSP.harmpd** がロックされていることを示した場合、リソース・モニターを始動できないようにしている状態を訂正して、次のように入力します。

```
haemunlkrm -s emsvcs -a IBM.PSSP.harmpd
```

位置

/opt/rsct/bin/haemunlkrm
haemunlkrm コマンドの位置

ファイル

/var/ha/log/em.trace.*cluster_name*
クラスター名が *cluster_name* のクラスターに関する、haemd デーモンのトレース・ログが入っています。

/var/ha/log/em.msgtrace.cluster_name
クラスター名が *cluster_name* のクラスターに関する、イベント・マネージャー・デーモンからのメッセージ・トレース出力が入っています。

関連資料:

762 ページの『haemtrcoff コマンド』
757 ページの『haemd デーモン』
379 ページの『emsvcsctrl コマンド』

---

hagsd デーモン

目的

リソース・モニターが更新するリソース変数のインスタンスを監視し、イベントを生成してそれをクライアント・プログラムに報告します。

構文

hagsd **[-a]** **[-s]** **[-k]** **[-d]** **[-c]** **[-u]** **[-t]** **[-o]** **[-r]** **[-h]** *daemon_name*

説明

hagsd デーモンはグループ・サービス・サブシステムの一部で、クラスターのノード上で実行されているアプリケーションの状態の調整および変更のモニターについての汎用機能を提供します。このデーモンはサブシステムの大部分のサービスを提供します。 *daemon_name* は、デーモンがログ・ファイルに付ける名前を指定します。この名前によって AIX エラー・ログ内のメッセージが識別されます。

hagsd デーモンの 1 つのインスタンスが各クラスター・ノード上で実行されます。 **hagsd** デーモンは SRC (システム・リソース・コントローラー) の制御下にあります。

このデーモンは SRC の制御下にあるため、コマンド・ラインから直接始動しない方がよいものです。これは通常 **grpsvcsctrl** コマンドによって呼び出されます。そのコマンドはクラスター始動プロセスによって呼び出されます。このデーモンを直接、始動または停止しなければならないときは、**startsrc** または **stopsrc** コマンドを使用します。

フラグ

-a サブシステムを追加します。

-s サブシステムを始動します。

-k サブシステムを停止します。

-d サブシステムを削除します。

-c サブシステムをクリーンアップします。すなわち、サブシステムをすべてのシステム区画から削除します。

-u サブシステムをすべてのシステム区画から構成解除します。

-t サブシステムのトレースをオンにします。

-o サブシステムのトレースをオフにします。

-r サブシステムをリフレッシュします。

-h 使用方法に関する情報を表示します。

パラメーター

daemon_name
デーモンがログ・ファイルに付ける名前を指定します。この名前によって AIX エラー・ログ内のメッセージが識別されます。

セキュリティー

このスクリプトを実行するには、**root** 権限を持っている必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 エラーが発生したことを示します。

制限

このコマンドは PSSP 環境のみで有効です。

標準出力

-h フラグが指定されている場合は、このコマンドの使用状況ステートメントが標準出力に書き込まれます。

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

1. グループ・サービス・サブシステムを現行システム区画内の SRC に追加するには、SP_NAME 環境変数を適切なシステム区画名に設定し、次のように入力します。
   ```
   hagsctrl -a
   ```
2. 現行システム区画内のグループ・サービス・サブシステムを始動するには、SP_NAME 環境変数を適切なシステム区画名に設定し、次のように入力します。
   ```
   hagsctrl -s
   ```
3. 現行システム区画内のグループ・サービス・サブシステムを停止するには、SP_NAME 環境変数を適切なシステム区画名に設定し、次のように入力します。

```
hagsctrl -k
```

4. グループ・サービス・サブシステムを現行システム区画内の SRC から削除するには、SP_NAME 環境変数を適切なシステム区画名に設定し、次のように入力します。

   ```
   hagsctrl -d
   ```

5. すべてのシステム区画上のグループ・サービス・サブシステムをクリーンアップするには、次のように入力します。

   ```
   hagsctrl -c
   ```

6. コントロール・ワークステーション上で、グループ・サービス・サブシステムをすべてのシステム区画から構成解除するには、次のように入力します。

   ```
   hagsctrl -u
   ```

7. 現行システム区画内のグループ・サービス・デーモンのトレースをオンにするには、SP_NAME 環境変数を適切なシステム区画名に設定し、次のように入力します。

   ```
   hagsctrl -t
   ```

8. 現行システム区画内のグループ・サービス・デーモンのトレースをオフにするには、SP_NAME 環境変数を適切なシステム区画名に設定し、次のように入力します。

   ```
   hagsctrl -o
   ```

位置

/opt/rsct/bin/hagsd
: **hagsd** デーモンが入っています。

ファイル

/var/ha/log/hags_nodenum_instnum**.** *syspar_name*
: ノード上の **hagsd** デーモンのログが入っています。

/var/ha/log/hags.*syspar_name_nodenum_instnum***.***syspar_name*
: コントロール・ワークステーション上の各 **hagsd** デーモンのログが入っています。

ファイル名には次の変数が含まれています。

- *nodenum* はデーモンが実行されているノードのノード番号
- *instnum* はデーモンのインスタンス番号
- *syspar_name* はデーモンが実行されているシステム区画の名前

関連資料:

746 ページの『grpsvcsctrl コマンド』

---

hagsns コマンド

目的

グループ・サービス・ネーム・サーバーの情報を取得します。

構文

hagsns [**-h** *host*] [**-c**] **-g** *group_name*

hagsns [**-h** *host*] [**-c**] **-s** *subsystem_name*

hagsns [**-h** *host*] [**-c**] **-p** *subsystem_pid*

説明

hagsns コマンドを使用してグループ・サービス・ネームサーバーのステータスを照会します。

フラグ

-c 出力を「English_only」に強制します。**-c** フラグが指定されていない場合は、出力にデーモンのロケールが使用されます。

-g *group_name*
状況を入手するサブシステムのグループを指定します。サブシステム・オブジェクト・クラスに *group_name* 変数が入っていなければ、このコマンドは失敗します。

-h *host*
ネームサーバーのステータスを取得するホストを指定します。

-p *subsystem_pid*
ネームサーバーのステータスを取得する *subsystem_pid* の特定のインスタンスを指定します。

-s *subsystem_name*
状況を入手するサブシステムを指定します。 *subsystem_name* 変数には実際のサブシステム名か、サブシステムの同義名を指定します。サブシステム・オブジェクト・クラスに *subsystem_name* 変数が入っていなければ、このコマンドは失敗します。

パラメーター

daemon_name
デーモンがログ・ファイルに付ける名前を指定します。この名前によって AIX エラー・ログ内のメッセージが識別されます。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

終了状況

0 コマンドが正常終了したことを示します。

ゼロ以外の値
エラーが発生したことを示します。

制限

このコマンドは PSSP 環境のみで有効です。

標準出力

-h フラグが指定されている場合は、このコマンドの使用状況ステートメントが標準出力に書き込まれます。

標準エラー

このコマンドは、必要な場合はエラー・メッセージを標準エラーに書き込みます。

例

グループ・サービス・サブシステムからドメイン情報を取得するには、次のように入力します。

```
hagsns -c -s cthags
```

または

```
hagsns -s cthags
```

出力は次のようになります。

```
HA GS NameServer Status
NodeID=1.16, pid=14460, domainID=6.14, NS established,CodeLevel=GSLevel(DRL=8)
NS state=kCertain, protocolInProgress=kNoProtocol,outstandingBroadcast=KNoBcast
Process started on Jun 19 18:34:20, (10d 20:19:22) ago, HB connection took (19:14:9).
Initial NS certainty on Jun 20 13:48:45, (10d 1:4:57) ago, taking (0:0:15).
Our current epoch of Jun 23 13:05:19 started on (7d 1:48:23), ago.
Number of UP nodes: 12
List of UP nodes: 0 1 5 6 7 8 9 11 17 19 23 26
```

この例で、**domainID=6.14** は、ノード 6 がネームサーバー (NS) ノードであることを意味します。ドメイン ID はノード番号と具体化番号から成ります。具体化番号は整数で、グループ・サービス・デーモンが始動するたびに値 1 が加算されます。 **NS established** はネームサーバーが確立されたことを意味します。

位置

/opt/rsct/bin/hagsns
: **hagsns** コマンドが入っています。

ファイル

/var/ha/log/hags_nodenum_instnum**.** *syspar_name*
: ノード上の **hagsd** デーモンのログが入っています。

/var/ha/log/hags.*syspar_name_nodenum_instnum***.***syspar_name*
: コントロール・ワークステーション上の各 **hagsd** デーモンのログが入っています。

ファイル名には次の変数が含まれています。

- *nodenum*。デーモンが稼働中のノード番号です。
- *instnum*。デーモンのインスタンス番号です。
- *syspar_name* はデーモンが実行されているシステム区画の名前

関連資料:

『hagsvote コマンド』

関連情報:

lssrc コマンド

nlssrc コマンド

---

hagsvote コマンド

目的

グループ・サービス・グループの vote 情報を取得します。

構文

hagsvote [**-h** *host*] [**-l**] [**-a** *argument*] [**-c**] **-g** *group_name*

hagsvote [**-h** *host*] [**-l**] [**-a** *argument*] [**-c**] **-s** *subsystem_name*

hagsvote [**-h** *host*] [**-l**] [**-a** *argument*] [**-c**] **-p** *subsystem_pid*

説明

hagsvote コマンドを使用してグループ・サービスの voting プロトコルのステータスを照会します。

フラグ

-a グループ・サービスのグループ名を指定します。このグループ名は **-g** フラグのものと異なります。こちらの場合、このグループは、プロトコルを結合するためのクライアントの最初の呼び出しによって作成されたものです。

-c グループ・サービス voting 情報の標準出力を要求します。出力はインストールされた言語ロケールにかかわらず、英語で表示されます。 **-c** が指定されていなければ、出力にデーモンのロケールが使用されます。

-g *group_name*
状況を入手するサブシステムのグループを指定します。サブシステム・オブジェクト・クラスに *group_name* 変数が入っていなければ、このコマンドは失敗します。

-h *host*
ステータスを取得するホスト名を指定します。

-l 「長」形式の明細出力を要求します。

-p *subsystem_pid*
vote を取得する *subsystem_pid* の特定のインスタンスを指定します。

-s *subsystem_name*
vote を行うサブシステムを指定します。*subsystem_name* 変数には実際のサブシステム名か、サブシステムの同義名を指定します。サブシステム・オブジェクト・クラスに *subsystem_name* 変数が入っていなければ、このコマンドは失敗します。

パラメーター

daemon_name
デーモンがログ・ファイルに付ける名前を指定します。この名前によって AIX エラー・ログ内のメッセージが識別されます。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

終了状況

0 コマンドが正常終了したことを示します。

ゼロ以外の値
エラーが発生したことを示します。

制限

このコマンドは PSSP 環境のみで有効です。

標準出力

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

標準エラー

このコマンドは、必要な場合はエラー・メッセージを標準エラーに書き込みます。

例

1. グループ **theSourceGroup** の voting プロトコルのステータス情報を長形式で表示するには、次のように入力します。

   ```
   hagsvote -ls cthags -a theSourceGroup (locale-dependent)
   ```

   出力は次のようになります。

   ```
   Number of groups: 4
   Group name [theSourceGroup] GL node [26] voting data:
   GL in phase [1] of n-phase protocol of type [Join].
   Local voting data:
   Number of providers: 1
   Number of providers not yet voted: 1 (vote not submitted).
   Given vote: [No vote value] Default vote: [No vote value]
   ProviderID Voted? Failed? Conditional?
   [101/26] No No Yes
   Global voting data:
   Number providers not yet voted: 1
   Given vote: [No vote value] Default vote: [No vote value]
   Nodes that have voted: []
   Nodes that have not voted: [26]
   ```

   出力の 1 行目はグループの合計数が 4 であることを示しています。2 行目はグループ名とグループ・リーダー・ノード (ここでは 26) です。その後の行に次の voting データが続きます。

   - グループ・リーダーは n- フェーズ・プロトコルのフェーズ 1 にあります。
   - プロトコルは Join プロトコルです。
   - ローカル・ノードについて、1 つのプロバイダーがあり、まだ vote (断定) していないプロバイダーの数は 1 です。
   - デフォルトの vote 値は指定されておらず、また vote 値も指定されていません。
   - 「ProviderID Voted? Failed? Conditional?」の行の下の「[101/26] No No Yes」は、プロバイダー ID が 101/26、まだ vote されていない、失敗していない、vote を待機している (そのような条件付である) ということを意味しています。

   その後で、出力はグローバルの voting ステータスを示しています。

   - まだ vote していないプロバイダー数は 1 です。
   - vote 値はまだ指定されておらず、デフォルトの vote 値はありません。
   - vote したノードはありません。
   - vote していないノードは、ノード 26 です。
2. 次の例では、ノード 26 がグループ・リーダー・ノードであることを除いて、出力の各行の意味は最初の例と同じです。

   ```
   hagsvote -ls cthags -a theSourceGroup -c (canonical form)
   ```

出力は次のようになります。

```
Number of groups: 4
Group Name: theSourceGroup
GL Node: 26 (I am GL)
Current phase number of an n-phase protocol: 1
Protocol name: [Join]
Local voting data:
Number of local providers: 1
Number of local providers not yet voted: 1 (vote not submitted)
Given vote: [No vote value] Default vote: [No vote value]Global voting data:
Number of nodes in group: 1
Number of global providers not yet voted: 1
Given vote: [No vote value] Default vote: [No vote value]
Nodes that have voted: []
Nodes that have not voted: [26]
```

位置

/opt/rsct/bin/hagsvote
: **hagsvote** コマンドが入っています。

ファイル

/var/ha/log/hags_nodenum_instnum_**.** _syspar_name_
: ノード上の **hagsd** デーモンのログが入っています。

/var/ha/log/hags._syspar_name_nodenum_instnum_**.**_syspar_name_
: コントロール・ワークステーション上の各 **hagsd** デーモンのログが入っています。

ファイル名には次の変数が含まれています。

- *nodenum* はデーモンが実行されているノードのノード番号
- *instnum* はデーモンのインスタンス番号
- *syspar_name* はデーモンが実行されているシステム区画の名前

関連資料:

771 ページの『hagsns コマンド』

関連情報:

lssrc コマンド

nlssrc コマンド

---

halt または fasthalt コマンド

目的

プロセッサーを停止します。

構文

{ **halt** | **fasthalt** } [**-l**] [**-n**] [**-p**] [**-q**] [**-y**]

説明

halt コマンドは、データをディスクに書き込んでからプロセッサーを停止します。マシンは再始動しません。このコマンドは、root ユーザーしか実行できません。他のユーザーがシステムにログインしている場合は、このコマンドを使わないでください。他のユーザーがログインしていなければ、**halt** コマンドが使

えます。マシンを直ちに再始動しない場合に、**halt** コマンドを使用してください。「 `....Halt completed....`」というメッセージが表示されたら、電源をオフにすることができます。

halt コマンドは、**syslogd** コマンドを使ってシャットダウンをログに記録し、シャットダウンのレコードをログイン・アカウンティング・ファイル `/var/adm/wtmp` 内に入れます。また、システムにより、システムがシャットダウンされたことを示すエントリーがエラー・ログに書き込まれます。

fasthalt コマンドは、**halt** コマンドを呼び出してシステムを停止します。 **fasthalt** コマンドは BSD と互換性があります。

フラグ

項目　説明

-l　アカウンティング・ファイル内に停止をログ記録しません。**-l** フラグは、アカウンティング・ファイルの更新を抑止しません。**-n** および **-q** フラグは、**-l** フラグを暗黙指定します。

-n　停止する前に **sync** を防止します。

-p　パワーダウンせずにシステムを停止します。

注: -p フラグは、永久停止を必要としないフラグと組み合わせて使用された場合、効果がありません。他のオペランドが遅延パワーオンと再始動を要求した場合でも、電源はオフになったままです。

-q　即時に停止させます。

注:

- **-q** フラグを指定して **halt** コマンドを実行しても、**sync** が出されないため、システムは即時に停止します。
- ワークロード・パーティション (WPAR) 内で **-q** フラグを指定して **halt** コマンドを実行すると、その **halt** コマンドは WPAR を停止させて、`D` (定義済み) 状態にすることができます。WPAR は完全には停止せず、WPAR を `T` (移行) 状態にする場合があります。これは、ファイル・システムをアンマウントする間に生じるタイムアウト状態または遅延のためです。

-y　システムをダイヤル呼び出し操作から停止します。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. 停止をアカウンティング・ファイル内にログ記録せずにシステムを停止させるには、次のコマンドを入力します。

   ```
   halt -l
   ```

2. システムを即時に停止させるには、次のコマンドを入力します。

   ```
   halt -q
   ```

3. システムをダイヤルアップ操作から停止させるには、次のコマンドを入力します。

   ```
   halt -y
   ```

ファイル

項目	説明
/etc/rc	システム始動スクリプトを指定します。
/var/adm/wtmp	ログイン・アカウンティング・ファイルを指定します。

関連情報:

fastboot コマンド

shutdown コマンド

sync コマンド

syslogd コマンド

---

hangman コマンド

目的

ハングマン (言葉当てゲーム) を開始します。

構文

hangman [*File*]

説明

hangman コマンドは、標準辞書から最低 7 文字のワードを 1 つ選択します。*File* パラメーターは、代替辞書を指定します。ユーザーは、そのワードに含まれる文字を一度に 1 つずつ推測してワードを当てます。間違えても 7 回まで答えられます。

ハングマンを開始すると、以下のように表示されます。

```
guesses: word: ....... errors: 0/7
guess:
```

guesses には、ユーザーが推測した文字が表示されます。ユーザーが推測した文字はすべて guesses の後ろに表示されます。 word: .... には、当てるワードの文字数が表示されます。上記の例では、7 つの .(ピリオド) があるので、このワードの文字数は 7 文字となります。推測した文字が当たっていると、該当する . がその正解の文字に置き換わります。errors: 0/7 は、不正解の回数を表示します。guess: プロンプトの後ろに推測した文字を入力します。次に例を示します。

```
guesses: word: .......... errors: 0/7
guess: q
guesses: q word: .......... errors: 1/7
guess: a
guesses: aq word: .a....a... errors: 1/7
guess: b
guesses: abq word: .a....a... errors 2/7
guess: j
guesses: abjq word: .a....a... errors: 3/7
guess: s
guesses: abjqs word: .a....a..s errors: 3/7
guess: z
guesses: abjqsz word: .a....a..s errors: 4/7
guess: y
guesses: abjqsyz word: .a....a..s errors: 5/7
guess: k
guesses: abjkqsyz word: .a....a..s errors: 6/7
guess: x
the answer was calculates, you blew it
```

ゲームを終了するには、割り込み (Ctrl-C) またはファイルの終わり (Ctrl-D) キー・シーケンスを押します。

ファイル

項目	説明
/usr/games	システム・ゲームのロケーション。

関連情報:

arithmetic コマンド

number コマンド

quiz コマンド

turnon コマンド

---

hash コマンド

目的

コマンド・パス名を記憶または報告します。

構文

パス名リストにコマンドのパスを追加する

hash [*Command* ...]

パス名リストを消去する

hash -r

説明

hash コマンドは、パス名をパス名リストに追加するかリストの内容を削除することにより、現行シェルがコマンドのパス名を記憶する方法を変更します。

パラメーターまたはフラグを指定しないと、**hash** コマンドはパス名リストの内容を標準出力に報告します。このレポートには、以前に **hash** コマンドを起動した際に見つかった現行シェル環境内のコマンドのパス名が含まれます。また、起動されたコマンドと通常のコマンド検索プロセスで見つかったコマンドも表示されます。

注**: hash** コマンドでは、シェル組み込みコマンドは報告されません。

-r フラグを使用すると、コマンド・パス名リストの内容をクリアすることができます。また、**PATH** 環境変数の値をリセットしても、リストからパス名をクリアすることができます。最も単純なフォーマットでクリアする場合は、次のように入力します。

```
PATH="$PATH"
```

Command パラメーターを使用すると、**hash** コマンドは指定したコマンドのパス名を検索し、このパスをリストに追加します。コマンドの指定には、/ (スラッシュ) を使用しないでください。

hash コマンドは現行シェル環境に影響を与えるので、Korn シェル (POSIX シェル) の正規組み込みコマンドとして用意されています。以下の例のように、**hash** コマンドを別のコマンドの実行環境内で呼び出すと、呼び出し元の環境のコマンド検索プロセスには影響しません。

```
nohup hash -r
find . -type f | xargs hash
```

hash コマンドの使用方法は、**alias -t** コマンドの使用方法と同じです。

フラグ

項目	説明
-r	パス名リストの内容をクリアします。

パラメーター

項目	説明
コマンド	パス名リストに追加する *Command* を指定します。

終了状況

次の終了値が戻されます。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. **wc** コマンドのパス名を検索して、パス名リストに追加するには、次のように入力します。

   ```
   hash wc
   ```

2. パス名リストの内容をクリアするには、次のように入力します。

   ```
   hash -r
   ```

ファイル

項目	説明
/usr/bin/ksh	Korn シェルの **hash** 組み込みコマンドが入っています。
/usr/bin/hash	**hash** コマンドが入っています。

関連情報:

alias コマンド

bsh コマンド

ksh コマンド

---

hatsoptions コマンド

目的

ノード上またはコントロール・ワークステーション上のトポロジー・サービス・オプションを制御します。

構文

hatsoptions [-s] [-d]

説明

このコマンドを実行する前に、環境変数 HB_SERVER_SOCKET を、トポロジー・サービス・サブシステムが使用する UNIX ドメイン・ソケットのロケーションに設定しておく必要があります。次のステートメントを使用できます。

```
export HB_SERVER_SOCKET=/var/ha/soc/hats/server_socket.partition name
```

代わりに、変数 HA_SYSPAR_NAME に区画名を設定することもできます。

このコマンドが成功するためには、トポロジー・サービス・デーモンが実行されている必要があります。

hatsoptions を使用して、トポロジー・サービス内の多数のオプションを制御できます。オプション **-s** はトポロジー・サービス・デーモンに、遅延しているメッセージをリジェクトするよう指示します。これは、メッセージがネットワーク内あるいは送信側または受信側のノードでしばしば遅延する、大規模なシステム構成で使用できます。このオプションは、時刻機構 (TOD) がすべてのノードとコントロール・ワークステーションにわたって同期化されている場合にのみ使用します。そうでなければ、送信側の時刻機構 (TOD) が受信側のものより遅れている場合に、該当しないものが破棄される可能性があります。

オプション **-d** はトポロジー・サービス・デーモンに、遅延しているメッセージをリジェクトしないよう指示します。これはデフォルトです。

フラグ

-s トポロジー・サービス・デーモンに、遅延しているメッセージをリジェクトするよう指示します。

-d トポロジー・サービス・デーモンに、遅延しているメッセージをリジェクトしないよう指示します (これがデフォルトです)。

セキュリティー

このコマンドを実行するには、**root** 権限を持っている必要があります。

終了状況

0 コマンドが正常終了したことを示します。

1 コマンドが正常に終了しなかったことを示します。

環境変数

HB_SERVER_SOCKET

この環境変数は、このコマンドを実行する前に設定してください。トポロジー・サービス・クライアントがトポロジー・サービス・デーモンとの接続に使用する UNIX ドメイン・ソケットのロケーションを設定する必要があります。この環境変数は、**/var/ha/soc/hats/server_socket.***partition name* に設定する必要があります。

HA_SYSPAR_NAME

HB_SERVER_SOCKET を設定しなかった場合は、HA_SYSPAR_NAME をパーティション名に設定する必要があります。

制限

このコマンドは、ピア・ドメインでのみ有効です。

標準出力

-h フラグが指定されている場合は、このコマンドの使用状況ステートメントが標準出力に書き込まれます。すべての冗長メッセージは標準出力に書き込まれます。

標準エラー

このコマンドは、エラー・メッセージを (必要に応じて) 標準エラーへ書き込みます。

例

ローカル・ノード上のトポロジー・サービス・デーモンに、明らかに遅延しているメッセージの廃棄を開始するように指示するには、次のように入力します。

```
export HA_SYSPAR_NAME=partition1

/opt/rsct/bin/hatsoptions -s
```

位置

/opt/rsct/bin/hatsoptions
hatsoptions コマンドが入っています。

ファイル

/var/ha/soc/hats/server_socket.*partition name*

関連情報:

lssrc コマンド

startsrc コマンド

stopsrc コマンド

---

head コマンド

目的

ファイルの最初の数行を表示します。

構文

head [**-***Count* | **-c** *Number* | **-n** *Number*] [*File* ...]

説明

head コマンドは、指定した各ファイルまたは標準入力の指定した行の数またはバイトを標準出力に書き出します。 **head** コマンドにフラグを指定しないと、最初の 10 行がデフォルト設定により表示されます。 *File* パラメーターは、入力ファイル名を指定します。入力ファイルはテキスト・ファイルでなければなりません。複数のファイルが指定されると、それぞれのファイルの開始は以下のように見えます。

```
==> filename <==
```

1 組のショート・ファイルを、それぞれを識別しながら表示するには、次のように入力します。

```
example% head -9999 filename1 filename2...
```

フラグ

項目	説明
-*Count*	指定した各ファイルの先頭からの表示行数を指定します。*Count* 変数は正の 10 進整数である必要があります。このフラグは **-n** *Number* フラグと同じように機能しますが、移植性に留意する場合は使用しないでください。
-c *Number*	表示バイト数を指定します。*Number* 変数は正の 10 進整数でなくてはなりません。
-n *Number*	指定した各ファイルの先頭からの表示行数を指定します。*number* 変数は正の 10 進整数でなくてはなりません。このフラグは **-***Count* フラグと同じように機能します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

Test ファイルの最初の 5 行を表示するには、次のように入力します。

```
head -5 Test
```

または

```
head -n 5 Test
```

関連情報:

tail コマンド

Files コマンド

入出力リダイレクト

---

help コマンド

目的

新規ユーザー向けの情報を提供します。

構文

help

説明

help コマンドは、新規ユーザー向けの情報を 1 ページ表示します。次のトピックに関する情報が使用可能です。

- ファイルの連結または表示
- 対話フォーマットによる行の編集
- メールの送受信

- システム・メッセージの読み取り
- パスワード・ファイル情報の変更
- システムの現在のユーザーの識別
- システム上の他のユーザーへのメッセージの送信
- ディレクトリーの内容の表示
- ソース・コード制御システムに関する情報の表示
- 端末モードの設定

例

ヘルプ情報を入手するには、コマンド・ラインに `help` と入力します。

関連情報:

ls コマンド

mail コマンド

sccshelp コマンド

who コマンド

---

hfistat コマンド

目的

ホスト・ファブリック・インターフェースのパフォーマンス統計情報を表示します。

構文

hfistat [-O options] [interval [count]]

hfistat [-h]

説明

hfistat コマンドは、ホスト・ファブリック・インターフェースに関するパフォーマンス統計情報を表示します。

次の情報は、出力テーブルの列見出しの説明です。

タイプ	列名と説明
HFI	**HFI** ホスト・ファブリック・インターフェース ID (0、1、...) は、統計情報が表示されるインターフェースを識別します。 **packets - sent** 送信されたパケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、クラスター・ネットワークに送信されたパケット数のことです。 **packets - imm_send** 即時送信パケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、即時送信される必要があるパケット数のことです。 **packets - receive** 受信されたパケットの総数 (56 ビット・カウンター) とは、受信元のウィンドウに関係なく、クラスター・ネットワークから正常に受信されたパケット数のことです。**packets - receive** カウンターがインクリメントされるのは、いずれかのウィンドウのパケット受信カウンターがインクリメントされる時点です。 **sent_packets - fullRDMA** 送信されたフル RDMA パケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、送信されたフル RDMA パケット数のことです。 **sent_packets - halfRDMA** 送信されたハーフ RDMA パケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、送信されたハーフ RDMA パケット数のことです。**sent_packets - halfRDMA** カウンターは、通知に対してはインクリメントしません。
	sent_packets - smallRDMA 送信されたスモール RDMA パケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、送信されたスモール RDMA パケット数のことです。**sent_packets - smallRDMA** カウンターは、通知に対してはインクリメントしません。 **sent_packets - ip** 送信された IP パケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、送信された IP パケット数のことです。 **sent_packets - cau** 送信された CAU パケットの総数 (56 ビット・カウンター) とは、送信元のウィンドウに関係なく、送信された CAU パケット数のことです。 **sent_packets - gups** 送信された GUPS パケットの総数 (56 ビット・カウンター) は、送信元のウィンドウに関係なく、送信された GUPS パケット数を指定します。 **dropped_packets – sending** 送信から除去されたパケットの総数 (56 ビット・カウンター) は、ウィンドウに関係なく、除去され、パケット送信用の FIFO によって送信されなかった、パケット数を指定します。 **dropped_packets – receiving** 受信から除去されたパケットの総数 (56 ビット・カウンター) は、受信元のウィンドウに関係なく、除去され、受信されなかった、ISR からのパケット数を指定します。 **xlat – wait** アドレス xlat 待機カウント (56 ビット・カウンター) は、失敗した変換数であり、その変換が保留状態であるものを指定します。**xlat – wait** レジスターは、このレジスターに何か書き込むとリセットされます。

タイプ	列名と説明
ISR	**HFI** ホスト・ファブリック・インターフェース ID (0、1、...) は、統計情報が表示されるインターフェースを識別します。 **cycBlocked – sending** 待機中のフリット (flit) をリンク上で送信できない場合、送信からブロックされたサイクルの (64 ビットの) カウンターは、2 - 3 GHz チップ・サイクルごとにインクリメントします。 **flits - sent** 送信されたフリット (flit) の数 (64 ビット・カウンター) は、フリット・ヘッダーが、対応する ISR インターフェースを通過するたびにインクリメントすることを除けば、ブロックされたサイクルのカウンターと似ています。 **flits - dropped** 除去されたフリット (flit) の数 (40 ビット・カウンター) とは、次のような特定のイベントが起きるたびに除去され、カウントされたフリット数のことです。 1. ポートのリンク状況ビットがオフである。 2. ISR ID が無効である。 **link – retries** リンク・レベル再試行 (24 ビット・カウンター) カウンターは、エラーのためにリンク再生バッファーからフリット (flit) が除去され、再びリンク経由で送信されるたびに、インクリメントします。
NMMU	**dyn_prot_cache – hits** ネスト・メモリー管理ユニットの動的保護キャッシュ・ヒット数。 **dyn_prot_cache – misses** ネスト・メモリー管理ユニットの動的保護キャッシュ・ミス数。 **ATLB – hits** ネスト・メモリー管理ユニットのアドレス変換バッファー・ヒット数。 **ATLB – misses** ネスト・メモリー管理ユニットのアドレス変換バッファー・ミス数。
CAU	**cycles – waiting** クレジット処理されるのを待機しているサイクル (無索引のカウンター)。

タイプ	列名と説明
ウィンドウ・ベース	**HFI** ホスト・ファブリック・インターフェース ID (0、1、...) は、統計情報が表示されるインターフェースを識別します。 **Win** ウィンドウ番号 (0, 1, 2 ...) は、統計情報が表示されるウィンドウを識別します。 **packet_indicated – send** パケット起因の送信カウント (56 ビット・カウンター) は、パケット (ヘッダー・セットにパケット起因のカウント・ビットがある) が送信されるたびにインクリメントします。 **packet_indicated – receive** パケット起因の受信カウント (56 ビット・カウンター) は、パケット (ヘッダー・セットにパケット起因のカウント・ビットがある) が受信されるたびにインクリメントしますが、このインクリメントはパケットがメモリーに書き込まれる前に行われます。 **packets – sent** 送信されたパケット数 (56 ビット・カウンター) は、クラスター・ネットワークに正常に送信されたパケットの数であり、パケットが折り返されるクラスター・ネットワークにパケットを送信することによって、同じ HFI に送信されたパケット数を含みます。 **packets – received** 受信されたパケット数 (56 ビット・カウンター) は、クラスター・ネットワークから正常に受信されたパケット数であり、同じ HFI から受信されたパケット数 (クラスター・ネットワークによって折り返されたパケット) を含みます。 **packet_dropped – sending** 送信から除去されたパケット数 (40 ビット・カウンター) とは、除去され、送信されなかった、送信 FIFO からのパケット数のことです。 **packet_dropped – receiving** 受信から除去されたパケット数 (40 ビット・カウンター) とは、除去され、受信されなかった、ISR からのパケット数のことです。 **immediate – send_pkts** 即時送信パケット数 (56 ビット・カウンター)。

フラグ

項目	説明
オプション	レポートの内容と表示方法を指定します。**Options** パラメーターは、**-O** フラグと一緒に使用してください。 `-O option1=value1,option2=value2,option3="value3 value4 value5"` オプションのリストはコンマで区切る必要があります。また、値のリストは引用符 (" ") で囲み、スペースで区切る必要があります。 サポートされているオプションとその値は次のとおりです。 • *type = [window nonwindow hfi isr nmmu cau all]* **type** オプションは、**hfistat** コマンドが、指定されたタイプのレジスター値のみを表示することを指定します。 `Default value: hfi` type オプションには、次の値を指定できます。 **window** ウィンドウ・ベースのパフォーマンス統計情報を表示します。 **nonwindow** 非ウィンドウ・ベースのパフォーマンス統計情報を表示します。 **hfi** HFI のパフォーマンス・カウンター値を表示します。 **isr** 統合スイッチ・ルーター (ISR) のレジスター値を表示します。 **nmmu** ネスト・メモリー管理ユニット (NMMU) のレジスター値を表示します。 **cau** Collectives Acceleration Unit (CAU) のレジスター値を表示します。 **all** すべてのレジスター値を表示します。 • *display = [raw | delta]* display オプションは、レジスター値のダンプに使用されます。 *Default value: none* 次の値を display オプションと一緒に指定できます。 **raw** 収集されたレジスター値を未加工状態でダンプします。 **delta** レジスターのデルタ値をダンプします。 **注:** 1. display オプションが指定されない場合、**hfistat** ツールは定様式出力を表示します。 2. raw および delta のオプション値は、同時に指定することはできません。 • *hfi = [0 1 ...]* **hfi** オプションは、レジスター値が報告されるホスト・ファブリック・インターフェースのリストを指定します。 *Default value: All available HFIs in the system* **注:** 1. 次のいずれかの方法で、ホスト・ファブリック・インターフェースの範囲を指定できます。 `hfistat –O hfi="0 1 2 3"` `hfistat –O hfi=0-3` 2. リストの最後を空にすると、最後の使用可能なホスト・ファブリック・インターフェースを示すことができます。次の **hfi** オプションの例は、1 から最後の使用可能なホスト・ファブリック・インターフェースまでの範囲を示しています。 `hfistat –O hfi=1-`

項目	説明
	• *window = [0 1 2...]* **window** オプションは、レジスター値が報告される HFI ウィンドウ番号のリストを指定します。 *Default value: All available HFI windows for the specified HFIs* **注:** 1. 次のいずれかの方法で、 HFI ウィンドウ番号の範囲を指定できます。 `hfistat –O window="210 211 212 213 214 215 216"` `hfistat –O window=210-216` 2. リストの最後を空にすると、最後の使用可能な HFI ウィンドウ番号を示すことができます。次の例は、0 から最後の使用可能な HFI ウィンドウまでの範囲を示す **window** オプションを指定しています。 `hfistat –O window=0-` • *output = <filename>* **output** オプションは、stdout の代わりに使用される出力ファイルを指定します。 *Default value: None* **注:** ファイル名は必須です。
interval	**hfistat** コマンドが統計情報を収集して印刷する間隔を、秒単位で指定します。**interval** パラメーターが指定されないと、**hfistat** コマンドは 2 秒間隔で実行します。
Count	**hfistat** コマンドが統計情報を収集して印刷する反復回数を指定します。**Count** パラメーターは、interval オプションと一緒に使用してください。**Count** パラメーターと **interval** パラメーターの両方が指定されないと、**hfistat** コマンドは 10 回実行します。**interval** パラメーターが指定されて、**Count** パラメーターが指定されないと、**hfistat** コマンドは無限に実行します。

例

1. 1. 10 回の反復に対して 2 秒の間隔を使用することによって、すべての使用可能な HFI の HFI ベースのパフォーマンス統計情報を表示するには、次のコマンドを実行します。

   ```
   # hfistat
   ```

2. 2. すべてのウィンドウのウィンドウ・ベースのパフォーマンス統計情報を含む、使用可能な HFI のすべてのパフォーマンス統計情報を定様式出力で表示するには、次のコマンドを実行します。

   ```
   # hfistat –O type=all 2 5
   ```

3. 3. より少ないウィンドウ (0-15) のウィンドウ・ベースのパフォーマンス統計情報のみを定様式出力で表示するには、次のコマンドを実行します。

   ```
   # hfistat –O type=window,window=0-15
   ```

4. 4. HFI-1 の CAU レジスター値を、10 個のサンプル (デフォルト) と 2 秒の間隔 (デフォルト) の定様式出力で表示するには、次のコマンドを実行します。

   ```
   # hfistat –O type=cau,hfi=1
   ```

5. 5. CAU とネスト・メモリー管理ユニット (NMMU) の両方のレジスター値を定様式出力で表示するには、次のコマンドを実行します。

   ```
   # hfistat –O type="cau nmmu"
   ```

6. 6. 2 秒の間隔と 5 回の反復を指定して、ウィンドウ 0 のウィンドウ・ベースのパフォーマンス統計情報を含む、すべての HFI のレジスター値を未加工状態でダンプするには、次のコマンドを実行します。

   ```
   # hfistat –O display=raw,type=all,window=0 2 5
   ```

7. 7. 2 秒の間隔と 5 回の反復を指定して、ISR パフォーマンス・カウンターのみのデルタ値をダンプするには、次のコマンドを実行します。

```
# hfistat –O display=delta,type=isr 2 5
```

ファイル

項目	説明
/usr/bin/hfistat	**hfistat** コマンドが含まれています。

関連情報:

perfstat_hfistat コマンド

Perfstat API

---

hmcauth コマンド

目的

hmcauth コマンドは、ハードウェア管理コンソール (HMC) に対する認証を行う場合、および HMC サービスをAIX Live Update操作に使用するためのトークンを入手する場合に使用します。また、トークンの無効化にも使用できます。

構文

HMC に対する認証を行い、トークンを入手するには、次の構文を使用します。

```
hmcauth [ -u user_name ] [ -p password ] [ -a hmc ] [ -P port ]
```

以前に生成したトークンの無効化および削除を行うには、次の構文を使用します。

```
hmcauth -r [ -a hmc ] [ -u user_name ]
```

既知のすべての HMC 認証トークンをリストするには、次の構文を使用します。

```
hmcauth -l
```

コマンドの使用法ステートメントを表示するには、次の構文を使用します。

```
hmcauth -h
```

説明

hmcauth コマンドは、すべてのオブジェクトへアクセスでき、HMC の適切な管理権限がある場合に使用することができます。**hmcauth** コマンドは、Live Update のオペレーションを実行するために AIX 区画の管理者が使用できるトークンを生成します。コマンドが正常に実行されると、トークンはカーネルに格納され、**geninstall** インターフェースが Live Update のオペレーションを実行できるようになります。

このコマンドを使用するには、以下のタスクを実行する権限が必要です。

- 管理対象区画の電源をオンにする。
- 管理対象区画をシャットダウンする。
- 管理対象区画を削除する (自動モードのみ)。
- 現行のプロファイルに基づき、管理対象区画を作成する (自動モードのみ)。
- 管理対象区画のブート・デバイスを設定する。
- 仮想イーサネット・アダプターを管理する。

hmcclientliveupdate HMC ロールは、Live Update操作に必要な特権をすべて持っています。このロールのある HMC 上でユーザーが識別される場合、認証は、hscroot ユーザーではなく、このユーザーに対して行うことができます。

hmcauth は、フラグを指定せずに使用することもできます。フラグを指定しない場合、**hmcauth** コマンドは、必要なすべての情報 (*user_name*、*hmc*、*password* など) の入力を求めるプロンプトを出します。

注**:** LPAR が再始動された場合、HMC 認証トークンは保存されません。したがって、Live Update操作を試みる前に、再度 HMC に対する認証を行う必要があります。

パラメーター

項目	説明
user_name	HMC のユーザー名を指定する、最大 64 文字の文字列。
password	パスワードを指定する、最大 64 文字の文字列。
hmc	認証する対象の HMC のホスト名または IP アドレスを指定する、最大 64 文字の文字列。
port	HMC に接続するポート番号を指定する最大 16 文字の文字列。

フラグ

項目	説明
-a *hmc*	認証する対象の HMC のホスト名または IP アドレスを指定します。 *hmc* 変数を指定しない場合、コマンドはその入力を求めるプロンプトを出します。
-h	コマンドの使用方法のステートメントを標準出力に書き込みます。
-r	HMC が生成するトークンを削除します。
-P *port*	HMC への接続に使用されるポート番号を指定します。 **-P** フラグはオプションです。したがって、ポート番号を指定しない場合、このポート番号はデフォルト値の 12443 に指定されます。HMC は常にポート 12443 を使用しますが、プロキシー・セットアップが使用されている場合は、**-P** オプションを使用して 12443 以外のポートをプロキシーで使用できるようにします。
-p *password*	認証のパスワードを指定します。コマンド・ラインでパスワードを指定しないと、パスワードの入力を求めるプロンプトが出されます。
-u *user_name*	認証に使用する HMC ユーザー名を指定します。すべてのオブジェクトにアクセスできること、および HMC での適切なタスク権限が必要です。

例

1. apollo という HMC に対して認証を行うには、次のコマンドを入力します。

   ```
   # hmcauth -a apollo -u hscroot -p T2x6z42p
   ```

2. パスワード・プロンプトを使用して、IP 5.5.55.121 の HMC に対して認証を行うには、次のコマンドを入力します。

   ```
   # hmcauth -a 5.5.55.121 -u hscroot
   Enter password for hscroot:
   ```

3. IP 5.5.55.121 の HMC に対して以前に行った認証を無効化するには、次のコマンドを入力します。

   ```
   # hmcauth -r -a 5.5.55.121
   ```

4. HMC ポート 12443 にアクセスできないファイアウォールを持つ、apollo という HMC に対して認証を行うには、再バインドされたプロキシー・ノードを、別のオープン・ポートを使用するようにセットアップできます。 proxy1 というプロキシー・ノード上のポート 14111 を持つ SSH クライアントを使用して論理区画 mylpar から認証を行うには、次のコマンドを入力します。

```
(0) root @ proxy1: /
# ssh -R localhost:14111:apollo:12443 root@mylpar

(0) root @ mylpar: /
# hmcauth -a localhost -u hscroot -P 14111
Enter HMC password:
```

lvupdate.data ファイルの hmc スタンザで management_console 属性として localhost を指定して、Live Update 操作を開始できます。

---

host コマンド

目的

ホスト名をインターネット・プロトコル (IP) アドレスに解決するか、IP アドレスをホスト名に解決します。

構文

host [**-n** [**-a**] [**-c** *Class*] [**-d**] [**-r**] [**-t** *Type*] [**-v**] [**-w**]] *Hostname* | *Address* [*Server*]

hostnew [**-a**] [**-c** *Class*] [**-d**] [**-r**] [**-t** *Type*] [**-v**] [**-w**] *Hostname* | *Address* [*Server*]

説明

/usr/bin/host コマンドは、*HostName* パラメーターを指定すると、ホスト・マシンの IP アドレスを戻し、*Address* パラメーターを指定するとホスト名を戻します。ネーム・レゾリューション・サービスの構成に応じて、 **host** コマンドは、*HostName* パラメーターに関連付けられているすべての別名も表示する場合があります。ネーム・レゾリューション・サービスの例には、 **local**、**nis**、および **bind** が含まれます。

ローカル・ホストがドメイン名プロトコルを使用している場合は、ローカルの **/etc/hosts** ファイルを検索する前に、ローカルまたはリモートのネームサーバー・データベースに対して照会が行われます。

フラグ

項目	説明
-a	-v -t * を使用するのと同じ機能です。
-c *Class*	インターネット以外のデータを検索するときに調べるクラスを指定します。有効なクラスは、次のとおりです。 **IN** インターネット・クラス **CHAOS** カオス・クラス **HESIOD** MIT Althena Hesiod クラス **ANY** ワイルドカード (上記のいずれか)
-d	デバッグ・モードをオンにします。
-n	/usr/bin/hostnew コマンドを出すのと同じです。 **hostnew** コマンドは、**bind** レゾリューション・サービスを実行します。
-r	再帰的処理を使用不可にします。

項目	説明
-t *Type*	照会するレコードのタイプを指定します。有効なタイプは、次のとおりです。 **A** ホストの IP アドレス **CNAME** 標準の別名 **HINFO** ホスト・プロセッサーとオペレーティング・システム・タイプ **KEY** セキュリティー・キー・レコード **MINFO** メールボックスまたはメール・リストの情報 **MX** メール交換機能 **NS** 指定したゾーンのネームサーバー **PTR** 照会が IP アドレスの場合はホスト名、それ以外の場合はその他の情報を指し示すポインター **SIG** 署名レコード **SOA** ドメインの「権限の開始」情報 **TXT** テキスト情報 **UINFO** ユーザー情報 **WKS** サポートされる定義済みサービス
-v	詳細モード。
-w	DNS サーバーからの応答を永続的に待ちます。

パラメーター

項目	説明
Address	ホスト名の解決に使うホスト・マシンの IP アドレスを指定します。*Address* パラメーターは、小数点フォーマットによる有効な IP アドレスでなければなりません。
HostName	IP アドレスの解決に使用するホスト・マシン名を指定します。*HostName* パラメーターには、固有のホスト名または予約済みホスト名 (*nameserver*、*printserver*、*timeserver* など、それらの名前が存在する場合) のいずれかが使用できます。
Server	照会するネームサーバーを指定します。

例

1. mephisto という名前のホスト・マシンのアドレスを表示するには、次のコマンドを入力します。

   ```
   host mephisto
   ```

   出力は、次の情報のようになります。

   ```
   mephisto is 192.100.13.5, Aliases: engr, sarah
   ```

2. アドレスが 192.100.13.1 のホストを表示するには、次のコマンドを入力します。

   ```
   host 192.100.13.1
   ```

   出力は、次の情報のようになります。

   ```
   mercutio is 192.100.13.1
   ```

3. ドメイン名が test.ibm.com の MX レコードを表示するには、次のように入力します。

   ```
   host -n -t mx test.ibm.com
   ```

   または

   ```
   hostnew -t mx test.ibm.com
   ```

出力は、次の情報のようになります。

```
test.ibm.com mail is handled (pri=10) by test1.tt.ibm.com
test.ibm.com mail is handled (pri=10) by test2.aix.ibm.com
```

ファイル

項目	説明
/etc/hosts	ローカル・ネットワーク上のホストのインターネット・プロトコル (IP) 名とアドレスが入っています。

関連資料:

801 ページの『hostname コマンド』

関連情報:

named コマンド

通信およびネットワークのセクション

---

host9 コマンド

目的

DNS 検索を実行します。

構文

host9 [**-aCdlrsTwv**] [**-c** *class*] [**-N** *ndots*] [**-R** *number*] [**-t** *type*] [**-W** *wait*] [**-m** *flag*] [**-4**] [**-6**] *name* [*server*]

説明

host9 コマンドは、DNS 検索を実行するための簡易ユーティリティーです。このコマンドを使用すると、名前を IP アドレス (またはその逆) に変換することができます。**host9** コマンドに引数やオプションを指定しない場合は、コマンド・ラインの引数とオプションの要約が出力されます。

フラグ

項目	説明
-a	**-v -t *** のフラグを使用する場合と同じです。
-c *class*	指定されたクラスの DNS 照会を行うように指示します。このフラグを使用すると、Hesiod クラスまたは Chaosnet クラスのリソース・レコードを検索することができます。デフォルトのクラスは IN (インターネット) です。
-C	ゾーンに対してリストされたすべての権限ネームサーバーからのゾーン名の SOA レコードを表示します。そのゾーンに対して検出された NS レコードにより、ネームサーバーのリストが定義されます。
-d	詳細出力を生成します。このフラグは **-v** フラグと同じように機能します。
-l	リスト・モードを指定します。これにより、**host9** コマンドがゾーン名に対してゾーン転送を実行します。NS、PTR およびアドレス・レコード (A/AAAA) を出力するゾーンを転送します。**-l** フラグを **-a** フラグと一緒に使用すると、**host9** コマンドはすべてのレコードを出力します。
-m *flag*	メモリー使用デバッグ・フラグのレコード、使用、およびトレースを設定します。
-N *ndots*	名前内に表示されるドットの数を設定します (絶対として考えられるため)。デフォルト値は、**/etc/resolv.conf** ファイル内の ndots ステートメントを使用して定義された値、または ndots ステートメントが存在しない場合は 1 です。より少ないドット数を持つ名前は相対名として解釈され、検索内にリストされたドメイン内または **/etc/resolv.conf** ファイル内のドメイン指示内で検索されます。

項目	説明
-r	**host9** コマンドで、通常は他のネームサーバーへの参照である非再帰照会を行い、これらの照会への応答を受信することで、ネームサーバーの動作を模倣できるようにします。
-R *number*	検索に対する UDP の再試行回数を変更します。*number* 値は、**host9** コマンドが応答のない照会を繰り返す回数を示します。デフォルトの再試行回数は 1 です。この数値が負またはゼロの場合は、再試行回数がデフォルトの 1 になります。
-s	SERVFAIL 応答で応答するサーバーがある場合に、照会を次のネームサーバーに送信しないように **host9** コマンドに知らせます。
-t *type*	照会タイプを選択します。このタイプには、CNAME、NS、SOA などの認識されている照会タイプを指定できます。照会タイプが指定されていない場合は、**host9** コマンドにより適切な照会タイプが自動的に選択されます。デフォルトでは A レコードが検索されますが、**-C** フラグを指定した場合は SOA レコードに対して照会が行われ、名前がドット 10 進数の IPv4 アドレスまたはコロンで区切られた IPv6 アドレスである場合は **host9** コマンドにより PTR レコードに対して照会が行われます。IXFR の照会タイプを選択した場合は、等号の後に開始シリアル番号を追加して開始シリアルを指定することができます (例: **-t** IXFR=12345678)。
-T	ネームサーバーの照会時に TCP 接続を使用します。ゾーン転送 (AXFR) 要求など、TCP を必要とする照会に対して TCP が自動的に選択されます。
-v	詳細出力を生成します。このフラグは **-d** フラグと同じように機能します。
-w	応答を永続的に待機します。応答を待機する時間は、ハードウェアの整数の最大値による秒数に設定されます。
-W *wait*	*wait* 秒間待機します。 *wait* 値が 1 より小さい場合は、待機間隔が 1 秒に設定されます。
-4	**host9** コマンドで IPv4 照会のトランスポートのみを使用するようにします。
-6	**host9** コマンドで IPv6 照会のトランスポートのみを使用するようにします。
name	検索するドメイン名を指定します。これにはドット 10 進数の IPv4 アドレスまたはコロンで区切られた IPv6 アドレスを指定することもできます。この場合、**host9** コマンドはそのアドレスに対して逆検索を実行します。
サーバー	オプションの引数 (**host9** コマンドが、**/etc/resolv.conf** ファイル内にリストされたサーバーの代わりに、照会するネームサーバーの名前または IP アドレス) を指定します。

IDN サポート

host9 コマンドが国際化ドメイン名 (IDN) サポートを使用してビルドされている場合は、非 ASCII のドメイン名を受け入れて表示することができます。 **host9** コマンドは、ドメイン名の文字エンコードを適切に変換してから、DNS サーバーに要求を送信したり、サーバーからの応答を表示します。何らかの理由で IDN サポートをオフにする場合は、IDN DISABLE 環境変数を定義します。**host9** コマンドの実行時にこの変数を設定すると、IDN サポートは使用不可になります。

ファイル

項目	説明
/etc/resolv.conf	

例

1. mephisto という名前のホスト・マシンのアドレスを表示するには、次のコマンドを入力します。

   ```
   host9 mephisto
   ```

   このコマンドは、次のような情報を表示します。

   ```
   mephisto is 192.100.13.5, Aliases: engr, sarah
   ```

2. アドレスが 192.100.13.1 のホスト・マシンを表示するには、次のコマンドを入力します。

   ```
   host9 192.100.13.1
   ```

   このコマンドは、次のような情報を表示します。

```
mercutio is 192.100.13.1
```

3. ドメイン名が test.ibm.com の MX レコードを表示するには、次のコマンドを入力します。

```
host9 -n -t mx test.ibm.com
```

このコマンドは、次のような情報を表示します。

```
test.ibm.com mail is handled (pri=10) by test1.tt.ibm.com
test.ibm.com mail is handled (pri=10) by test2.aix.ibm.com
```

関連情報:

named9 コマンド

nsupdate9 コマンド

rndc-confgen コマンド

rndc.conf コマンド

---

hostent コマンド

目的

システム構成データベース内のアドレス・マッピング・エントリーを直接操作します。

構文

アドレスからホスト名へのマッピングを追加する

hostent -a *IPAddress* **-h "***HostName...***"**

アドレスからホスト名へのマッピングを削除する

hostent -d *IPAddress*

アドレスからホスト名へのマッピングをすべて削除する

hostent -X

アドレスからホスト名へのマッピングを変更する

hostent -c *IPAddress* **-h "***HostName...***"[-i** *NewIPAddress* **]**

アドレスまたはホスト名をコロン・フォーマットで表示する

hostent -s { *IPAddress* | **"***HostName***"** } **[-Z]**

アドレスからホスト名へのマッピングをコロン・フォーマットですべて表示する

hostent -S [-Z]

説明

hostent 低レベル・コマンドは、システム構成データベース内でアドレス・マッピング・エントリーを追加、削除、または変更します。データベース内のエントリーは、インターネット・プロトコル (IP) アドレス (ローカルまたはリモート) を等価のホスト名にマッピングするときに使います。

hostent コマンドは、**/etc/hosts** ファイル内のアドレスからホスト名へのマッピング・エントリーを 1 つまたはすべて表示できます。所定のローカルまたはリモート・ホストのインターネット・プロトコル (IP) アドレスは、1 つ以上のホスト名に対応付けることができます。 IP アドレスを表すには、小数点付き 10 進の構文を使います。ホスト名は最大長が 255 文字の文字列として表され、ブランク文字は使いません。各エントリーは 1 行に設定しなければなりません。複数の *HostNames* (または別名) を指定できます。

注**:** 有効なホスト名またはホスト名の別名には、英文字が少なくとも 1 文字入っていなければなりません。 x で始まり、その後ろに 16 進の数字 (0 から f) が続くホスト名または別名を指定する場合は、ホスト名または別名に 16 進数で表現できない文字を 1 つ以上指定する必要があります。システムは、ホスト名または別名のうち 16 進数でない文字が少なくとも 1 文字ある場合を除いて、後ろに 16 進数が付いた先行 x をアドレスの基本 16 進表現として解釈します。したがって、xdeer は有効なホスト名ですが xdee は無効です。

System Management Interface Tool (SMIT) の **smit hostent** 高速パスを使用して、このコマンドを実行できます。

フラグ

注**: -a**、 **-d**、**-c**、および **-s** フラグは併用できません。

項目	説明
-a *IPAddress*	データベース内のインターネット (IP) プロトコル・アドレスに関して、IP アドレスからホスト名へのマッピング・エントリーを追加します。ホスト名を **-h** フラグで指定してください。
-c *IPAddress*	*IPAddress* 変数により指定されたアドレスに対応するデータベース内で、アドレスからホスト名へのマッピング・エントリーを変更します。変更したホスト名を **-h** フラグで指定してください。現在の IP アドレスを新しいアドレス (*IPAddress*) に変更する場合は、**-i** フラグを使用してください。
-d *IPAddress*	*IPAddress* 変数により指定されたアドレスに対応するデータベース内で、IP アドレスからホスト名へのマッピング・エントリーを削除します。
-h "*HostName...***"**	ホスト名のリストを指定します。リスト内のエントリーはブランクで区切ります。**-h"***HostName...***"** フラグは、 **-a** フラグと一緒に指定してください。 **-c** フラグには、**-h"***HostName...***"** フラグを付ける必要があります。
-i *NewIPAddress*	新しい IP アドレスを指定します。このフラグは、既存の IP アドレスを、 *NewIPAddress* 変数で置換する場合、 **-c** フラグで必要となります。
-S	データベース内のすべてのエントリーを表示します。
-s"*HostName***"**	**"***HostName***"** 変数により指定されたホスト名と一致する、IP アドレスからホスト名へのマッピング・エントリーを表示します。
-s *IPAddress*	*IPAddress* 変数により指定されたエントリーと一致する、IP アドレスからホスト名へのマッピング・エントリーを表示します。
-X	データベース内で IP アドレスからホスト名へのマッピング・エントリーをすべて削除します。
-Z	コロン・フォーマットで照会のレポートを生成します。このフラグは、**hostent** コマンドを SMIT ユーザビリティー・インターフェースから開始する場合に使用します。

注**: hostent** コマンドは、.08、.008、.09、.009 の各アドレスを認識できます。先行ゼロが付いたアドレスは 8 進として解釈され、8 進数には 8 または 9 が使えません。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

例

1. エントリーをデータベースに追加し、アドレスを一連のホスト名に関連付けるには、次のフォーマットでコマンドを入力します。

   ```
   hostent  -a 192.100.201.7  -h "alpha bravo charlie"
   ```

   例 1 で、IP アドレス 192.100.201.7 が、同義語が bravo および charlie である、 1 次ホスト名 alpha を持つホストのアドレスとして指定されています。
2. データベース内でホスト名と一致するエントリーを表示するには、次のフォーマットでコマンドを入力します。

   ```
   hostent  -s alpha
   ```

   例 2 では、表示されるエントリーはホスト名 alpha と一致します。
3. エントリーの IP アドレスを新しい IP アドレスに変更するには、次のフォーマットでコマンドを入力します。

   ```
   hostent  -c 192.100.201.7  -i 192.100.201.8
   ```

   例 3 では、古い IP アドレスが 192.100.201.7 で、新アドレスは 192.100.201.8 です。

ファイル

項目	説明
/etc/hosts	ネットワークのホスト名とアドレスが入っています。

関連資料:

801 ページの『hostname コマンド』

関連情報:

TCP/IP のネーム・レゾリューション

---

hostid コマンド

目的

現行ローカル・ホストの ID を設定または表示します。

構文

/usr/sbin/hostid [*HexNumber* | *InternetAddress* | *HostName*]

説明

/usr/sbin/hostid コマンドは、現行ローカル・ホストの ID (固有のホスト名または数値による引数) を 16 進数として表示します。この数値は、すべてのホスト間で固有であり、一般に *InternetAddress* または *HostName* パラメーターにより指定されたホストのアドレスに設定されると予想されます。 root ユーザーは、*HexNumber*、*InternetAddress*、または *HostName* パラメーターに 16 進数を指定して、**hostid** コマンドを設定できます。ホスト ID は、**/etc/rc.net** ファイルによってホスト名に設定されます。

パラメーター

項目	説明
HexNumber	現行ローカル・ホストを表す固有の 16 進数。
InternetAddress	現行ローカル・ホストを表すインターネット・アドレス。
HostName	固有のホストにマッピングするシンボル名。

例

1. **hostid** コマンドを使って、ローカル・ホストの ID をローカル・インターネット・アドレスに設定するには、次のフォーマットでコマンドを入力します。

   ```
   hostid   192.9.200.3
   0xc009c803
   ```

   hostid コマンドは、インターネット・アドレス `192.9.200.3` を 16 進数表記 `0xc009c803` に変換した後、ローカル・ホスト (ネットワークに接続されているワークステーション) をこのアドレスに設定します。
2. ローカル・ホストの ID を表示するには、次のように入力します。

   ```
   hostid
   0xc009c803
   ```

   hostid コマンドは、ホストの ID を 16 進数として表示します。

関連資料:

801 ページの『hostname コマンド』

関連情報:

gethostid コマンド

sethostid コマンド

TCP/IP アドレッシング

---

hostmibd デーモン

目的

hostmibd dpi2 サブエージェント・デーモンをバックグラウンド・プロセスとして始動します。

構文

hostmibd [**-f** *File*] [**-d** [*Level*]] [**-h** *Hostname*] [**-c** *Community*]

説明

hostmibd コマンドは **hostmibd** dpi2 サブエージェントを始動します。このコマンドは、root 権限を持つユーザーまたはシステム・グループのメンバーのみが発行できます。

hostmibd デーモンは RFC 1592 で定義されている標準 Simple Network Management Protocol Distributed Protocl Interface Version 2.0 を使用してコンパイルします。これは dpiPortForTCP.0 (1.3.6.1.4.1.2.2.1.1.1.0) を介して dpi2 エージェントと通信するときに、dpi2 サブエージェントとして動作します。dpiPortForTCP.0 (1.3.6.1.4.1.2.2.1.1.1.0) は RFC 1592 Section 3.1 に定義されています。

MIB (管理情報ベース) は RFC 1155 に定義されています。 **hostmibd** が管理する特定の MIB 変数は RFC 2790 に定義されています。 **hostmibd** が管理する実際の MIB 変数は次の 6 つのサブツリーです。

- hrSystem (1.3.6.1.2.1.25.1)

- hrStorage (1.3.6.1.2.1.25.2)
- hrDevice (1.3.6.1.2.1.25.3)
- hrSWRun (1.3.6.1.2.1.25.4)
- hrSWRunPerf (1.3.6.1.2.1.25.5)
- hrSWInstalled (1.3.6.1.2.1.25.6)

hostmibd デーモンは、通常、**/etc/rc.tcpip** シェル・スクリプトが呼び出されたときのシステム起動の間に実行されます。

hostmibd デーモンは SRC (システム・リソース・コントローラー) を使用して制御されます。コマンド・ラインに **hostmibd** を入力することはお勧めできません。

次の SRC コマンドを使用して **hostmibd** デーモンを操作します。

startsrc
サブシステム、サブシステム・グループ、またはサブサーバーを始動します。

stopsrc
サブシステム、サブシステムのグループ、またはサブサーバーを停止します。

refresh
サブシステムまたはサブシステム・グループが該当する構成ファイルを再読み取りします。

lssrc サブシステム、サブシステム・グループ、またはサブサーバーの状況を取得します。 **lssrc** コマンドを詳細形式のステータスを要求して発行したユーザーが root ユーザーではない場合、コミュニティー名情報は表示されません。

フラグ

項目	説明
-c *Community*	指定されたコミュニティー名を使用します。**-c** フラグが指定されなければ、デフォルトのコミュニティー名は「public」です。
-d *Level*	トレース/デバッグのレベルを指定します。レベルは次のとおりです。 • 0 = 最低レベル • 8 = DPI レベル 1 • 16 = DPI レベル 2 • 32 = 内部レベル 1 • 64 = 内部レベル 2 • 128 = 内部レベル 3 複数トレース・レベルの番号を付加します。 **-d** フラグが指定されていて *Level* が指定されていない場合、デフォルトのレベルは 56 です。 **-d** フラグが指定されていなければ、デフォルトのレベルは 0 です。
-f *File*	デフォルト以外の構成ファイルを指定します。**-f** フラグが指定されていなければ、デフォルトの構成ファイルは **/etc/hostmibd.conf** です。このファイル・フォーマットの情報は **/etc/hostmibd.conf** ファイルを参照してください。
-h *Host*	要求を指定されたホストに送信します。*Host* 値には、IPv4 アドレス、IPv6 アドレス、またはホスト名のいずれかを指定できます。**-h** フラグが指定されていない場合、デフォルトの宛先ホストは 'loopback' (127.0.0.1) です。

例

1. **hostmibd** デーモンを始動するには、次のようなコマンドを入力します。

```
startsrc -s hostmibd -a "-f /tmp/hostmibd.conf"
```

このコマンドは **hostmibd** デーモンを始動し、構成ファイルを **/tmp/hostmibd.conf** から読み取ります。

2. **hostmibd** デーモンを停止するには、通常次のように入力します。

   ```
   stopsrc -s hostmibd
   ```

   このコマンドは **hostmibd** デーモンを停止します。**-s** フラグは、後に続くサブシステムを停止させることを指定します。

3. **hostmibd** から簡略状況を取得するには、次のように入力します。

   ```
   lssrc -s hostmibd
   ```

   このコマンドは、デーモン名、デーモンのプロセス ID、デーモンの状態 (アクティブまたは非アクティブ) を戻します。

4. **hostmibd** デーモンから詳細形式のステータスを取得するには、次のように入力します。

   ```
   lssrc -ls hostmibd
   ```

   ユーザーが root ユーザーであれば、この詳細形式のステータス・レポートによって構成パラメーターが **/etc/hostmibd.conf** にリストされます。

ファイル

項目	説明
/etc/hostmibd.conf	**hostmibd** コマンドの構成パラメーターを定義します。
/etc/mib.defs	SNMP エージェントとマネージャーが認識して処理する必要のある管理情報ベース (MIB) 変数を定義します。

関連情報:

snmpdv3 コマンド

snmpmibd コマンド

---

hostname コマンド

目的

現行ホスト・システムの名前を設定または表示します。

構文

/usr/bin/hostname [*HostName*] [**-s**]

説明

/usr/bin/hostname コマンドは、現行ホスト・システムの名前を表示します。ホスト名を設定できるのは root ユーザー権限を持つユーザーのみです。**mkdev** コマンドおよび **chdev** コマンドもホスト名を永続設定します。初めて TCP/IP インターフェースを定義するときは、**mkdev** コマンドを使用してください。

System Management Interface Tool (SMIT) の **smit mkhostname** 高速パスを使用して、このコマンドを実行できます。

フラグ

項目	説明
-s	表示される名前からドメイン情報を切り出します。

パラメーター

項目	説明
HostName	ホストの 1 次名を設定します。

注**:** *HostName* パラメーターを使用するには、 root ユーザー権限が必要です。

セキュリティー

RBAC ユーザーおよび **Trusted AIX** ユーザーへの注意**:** このコマンドは特権命令を実行できます。 特権命令を実行できるのは特権ユーザーのみです。 権限および特権についての詳細情報は、「セキュリティー」の『特権コマンド・データベース』を参照してください。 このコマンドに関連した特権および権限のリストについては、**lssecattr** コマンドまたは **getcmdattr** サブコマンドの項を参照してください。

関連情報:

chdev コマンド

mkdev コマンド

gethostname コマンド

TCP/IP のネーム・レゾリューション

---

hosts2ldif コマンド

目的

hosts ファイルから、LDAP データ交換形式 (LDIF) ファイルを作成します。

構文

hosts2ldif [**-i** *InputFile*] [**-o** *OutputFile*] [**-s** *SearchBase*]

説明

/usr/sbin/hosts2ldif コマンドは、 **/etc/hosts** ファイル、あるいは **/etc/hosts** に似た別のファイルから、LDAP データ交換形式を作成します。 **/etc/hosts** ファイルをフラグを指定せずに使用して、 baseDN として `cn=hosts` を使用する、 **/tmp/hosts.ldif** LDIF ファイルを作成します。

このコマンドによって作成された LDIF ファイルは SecureWay ディレクトリー・スキーマに準拠しており、**ldap** メカニズムのセットアップに使用されます。 **ldap** メカニズムはサポートされていますが、**ldap** メカニズムではなく、**nis_ldap** メカニズムの使用をお勧めします。

フラグ

項目	説明
-i *InputFile*	入力に使用する hosts ファイルを指定します。
-o *OutputFile*	出力に使用する LDIF ファイルを指定します。
-s *SearchBase*	LDAP サーバー上で、ホスト・テーブルの baseDN を指定します。

例

1. **/etc/hosts** ファイルから **/home/ldifhosts** を作成するには、次のように入力します。

   ```
   hosts2ldif -o /home/ldifhosts
   ```

2. **/home/hosts.bak** ファイルから、 **/tmp/hosts.ldif** を作成するには、次のように入力します。

   ```
   hosts2ldif -i /home/hosts.bak
   ```

3. cn=hosttab を baseDN として使用して、 **/etc/hosts** ファイルから **/home/ldifhosts** を作成するには、次のように入力します。

   ```
   hosts2ldif -o /home/ldifhosts -s cn=hosttab
   ```

ファイル

項目	説明
/etc/hosts	ローカル・ネットワーク上のホストのインターネット・プロトコル (IP) 名とアドレスが入っています。

関連情報:

TCP/IP のネーム・レゾリューション

---

hp コマンド

目的

HP2640 および HP2621 シリーズの端末装置用の特殊機能を処理します。

構文

hp [**-e**] [**-m** ...]

説明

hp コマンドは、標準入力 (通常の場合、**nroff** コマンドからの出力) を読み取り、標準出力 (通常の場合、Hewlett-Packard 2640 および 2621 シリーズの端末ディスプレイ) に書き込みます。

端末装置がディスプレイ拡張機能を備えている場合は、添え字と肩文字を表示できます。数学記号機能を使用すると、2 つの例外を除いて、ギリシャ文字やその他の特殊文字を表示できます。 **hp** コマンドは、右矢印で論理演算子 NOT と似たものを表示します。また、hp コマンドは積分記号の上半分しか表示しません。

重ね打ち文字とは、後ろにバックスペースや他の文字が続く文字です。重ね打ち文字またはバックスペースの後ろの文字が下線文字であれば、重ね打ち文字は端末装置の拡張機能に応じてアンダーラインが付いた状態または逆転表示の状態で表示されます。

注**:** 制御文字のシーケンス (逆行送りとバックスペース) によっては、画面にテキストが表示されなくなる場合があります。**tbl** コマンドで生成された縦線が使用されているテーブルでは、縦線の下部を含むテキスト行が脱落することがあります。最初に **col** を使用して入力をパイプ接続し、次に **hp** コマンドでパイプ接続することでこのトラブルを回避できる場合があります。

フラグ

項目	説明
-e	重ね打ち文字をアンダーライン付きで、肩文字を半輝度で、添え字をアンダーライン付きの半輝度で表示します。それ以外の場合は、重ね打ち文字、添え字、および肩文字はすべて逆転表示 (明るい背景に濃い文字) で表示されます。ディスプレイにディスプレイ拡張機能が付いている場合にのみ、このフラグを使用してください。
-m	テキスト内に連続するブランク行が何行あっても、1 行だけブランク行を生成します。

関連資料:

418 ページの『eqn コマンド』

738 ページの『greek コマンド』

関連情報:

col コマンド

nroff コマンド

tbl コマンド

---

hplj コマンド

目的

HP LaserJet シリーズのプリンター用に **troff** コマンド出力を後処理します。

構文

hplj [**-F** *Directory*] [**-quietly**] [**-landscape**] [*File* ...]

説明

hplj コマンドは、Hewlett-Packard LaserJet シリーズのプリンターに出力できるように、**troff** コマンドの出力を処理します。

オプションとして 1 つ以上のファイルを指定すると、**hplj** コマンドはそのファイルを処理します。ファイルを指定しなければ、標準入力を解釈するフィルターとして機能します。 *File* パラメーターは、HP LaserJet シリーズ・プリンターに出力するときに **hplj** コマンドが処理するファイルを指定します。

注**:** プリンターに K カートリッジまたは Text-Equations カートリッジが装着されていれば、**hplj** コマンドはそれを使用することができます。(Text-Equations カートリッジ、HP 部品番号 C2053A #C07 は、K カートリッジに優先して使用されます)。また、デフォルト・フォント・ファイルはどちらかのカートリッジが装着されているものと想定します。 K カートリッジがない場合は、ダウンロードされたビットマップ・フォントを使用してください。そのためには、HP プリンターのフォント・ディレクトリー (**/usr/lib/font/devhplj**) 内の **no_cart** シェル・スクリプトを実行してください。

実際はマウントされていないのに、どちらかのカートリッジがマウントされているとフォント・ファイルが想定すると、出力が不正確になることがあります。また、K カートリッジか Text-Equations カートリッジの他にも別のカートリッジやソフト・フォントが装着されている場合も、出力が不正確になることがあります。

hplj コマンドの機能は、**/usr/lib/font/devhplj** ファイル内の **.out** で終わるファイルによって左右されます。これらのファイルが正しく設定されていなければ、このコマンドは正しい出力を生成しません。詳細については、**troff** フォント・ファイル・フォーマットのセクションを参照してください。

フラグ

項目	説明
-F*Directory*	指定されたディレクトリーをフォント・ファイルの検索場所として識別します。デフォルトでは、**hplj** コマンドはフォント・ファイルを **/usr/lib/font/devhplj** ディレクトリー内で探します。
-quietly	致命的でないエラー・メッセージをすべて抑止します。

項目	説明
-landscape	指定したファイルを横長形式 (ランドスケープ) で出力します。横長ページの向きは、通常の読みかたで、ページの幅がその長さよりも大きくなるように設定されます。デフォルトでは、**hplj** コマンドを縦長方向に出力します。 注**:** 横長形式は、Hewlett-Packard Jet II プリンター上のメモ Courier フォントにしか使用できません。したがって、**troff** 文書は Courier フォントでフォーマットする必要があります。そのためには、**troff** 入力ファイルの先頭に次の行を挿入します。 `.fp 1 C` `.fp 2 C` `.fp 3 CB` Courier フォントはフォント位置 #1 と #2 にロードされ、Courier ボールドは位置 #3 にロードされます。

例

1. **lp** コマンドを使って、 `foo` という **troff** ファイルを、 `hp` というプリンターに出力するには、次のように入力します。

   ```
   troff -mm -Thplj foo | hplj | lp -dhp -o -dp
   ```

2. **qprt** コマンドを使って、 `boo` という **troff** ファイルを、 `hp` というプリンターに出力するには、次のように入力します。

   ```
   troff -mm -Thplj boo | hplj | qprt -dp -Php
   ```

   注**:** 上記 2 つの例では、**-dp** フラグは、プリンターのデータをパススルー (未変更) モードでプリンターに送ります。

ファイル

項目	説明
/usr/lib/font/devhpl/*.out	フォント・ファイルが入っています。

関連情報:

troff コマンド

troff フォント

---

hpmcount コマンド

目的

アプリケーションのパフォーマンスを測定します。

構文

hpmcount [**-a**] [**-b** *time_base*] [**-d**] [**-D** *metrics*] [**-g** *event_groups*] [**-H**] [**-k**] [**-m** *metrics_groups*] [**-o** *file*] [**-s** *set*] [**-x**] *command*

hpmcount [-h]

説明

hpmcount コマンドは、壁時計の実行時刻、ハードウェアのパフォーマンス・カウンター情報、導き出されたハードウェア・メトリック、および *command* で名前を指定したアプリケーションに対するリソース使用状況統計情報 (**getrusage()** システム・コールにより得られたもの) を提供します。

モニターされるイベント・タイプおよび関連するハードウェア・パフォーマンス・カウンターを指定するには次の方法があります。すなわち、**-s** オプションを設定する、**HPM_EVENT_SET** 環境変数の中にイベント・グループ名、セット番号、またはコンマで区切られたセット番号のリストを指定する、あるいは **libHPM_events** 入力ファイル (**HPM_EVENT_SET** よりも優先) の中にカウンターとイベントのペア (POWER3 / PowerPC 604 RISC マイクロプロセッサー) またはイベント・グループ名 (POWER4 およびそれ以降) を指定します。各セットはカウント・モードで修飾できます。イベントのグループ番号または名前を指定するには、**-g** オプションを設定するか、または **HPM_EVENT_GROUP** 環境変数でイベント・グループのコンマ区切りリストを指定します。同じ方法で、各イベント・グループをカウント・モードで修飾できます。

有効なイベント・セット番号は、1 から上限値 (プロセッサー・タイプにより異なる) までの範囲です。プロセッサー・タイプは **pmlist** コマンドを使用してリストすることができます。セット番号の代わりに、コンマで区切られたイベント・セットのリストを指定することができます。その場合は、カウンター多重方式が選択されます。イベント・セットをすべて選択するには、番号の値を `0` に設定します。

派生メトリックのコンマ区切りリストを指定するには、**-D** オプションを設定します。それぞれの派生メトリックは、カウント・モードで修飾できます。

派生メトリック・グループのリストを指定するには、**-m** オプションを設定するか、または **HPM_PMD_GROUP** 環境変数で派生メトリック・グループのコンマ区切りリストを指定します。これにより、指定されたグループに関連する派生メトリックをすべて選択できます。各メトリック・グループは、カウント・モードで修飾できます。

-k および **-H** オプションを指定することにより、カウントの中にシステムとハイパーバイザー (ハイパーバイザー・モードをサポートしているプロセッサーの場合) の活動を含めることができます。

多重方式でカウントする場合、その結果を使用するには正規化する必要があります。 データの正規化に使用されるデフォルトの基準は、時間基準です。**-b** オプションを使用すると、データの正規化に PURR 時間または SPURR 時間 (プロセッサーでサポートされている場合) を使用できます。データの正規化のための基準は、**HPM_NORMALIZE** 環境変数を使用しても定義できます。

結果は **-x** オプションを使用して XML フォーマットで出力できます。

フラグ

項目	説明
-a	POE 実行に関するカウンターを集計します。
-b *time_base*	データの正規化のための基準を選択します。使用可能な基準は次のとおりです。 **time** 時間基準 **purr** PURR 時間 (使用可能な場合) **spurr** SPURR 時間 (使用可能な場合) デフォルト値は **time** です。
-d	カウンター多重方式のために詳細なセット・カウントを追加します。

項目	説明
-D *metrics*	評価する派生メトリックのリストを選択します。それぞれの派生メトリックは、次のようにカウント・モードで修飾できます。 `metric:counting_modes` (使用可能なカウント・モードについては、**-m** オプションを参照してください。)
-g *event_groups*	イベントの事前定義グループまたはイベントのグループ名または番号のコンマ区切りリストをリストします。コンマで区切られたグループのリストを使用する場合は、カウンター多重方式が選択されます。各イベント・グループは、次のようにカウント・モードで修飾できます。 `event_group:counting_modes` (使用可能なカウント・モードについては、**-m** オプションを参照してください。)
-H	そのプロセスのためにハイパーバイザーの活動状況を追加します。
-h	ヘルプ・メッセージを表示します。
-k	そのプロセスのためにシステム稼働状況を追加します。
-m *metrics_groups*	評価する派生メトリック・グループのリストを選択します。派生メトリック・グループは、特定の派生メトリック・グループに属さないすべての派生メトリックを参照します。各メトリック・グループは、次のようにカウント・モードで修飾できます。 `metric_group_name:counting_modes` 使用可能なカウント・モードは次のとおりです。 **u** ユーザー・モード **k** カーネル・モード **h** ハイパーバイザー・モード **r** runlatch モード **n** 非割り込みモード
-o *file*	出力ファイル名。
-s *set*	イベントの事前定義セットまたはセットのコンマ区切りリストをリストします (1 から *N*、すべて選択する場合は `0`。 **pmlist** コマンドを参照してください)。コンマで区切られたセットのリストを使用する場合は、カウンター多重方式が選択されます。各セットは、次のようにカウント・モードで修飾できます。 `event_set:counting_modes` (使用可能なカウント・モードについては、**-m** オプションを参照してください。)
-x	結果を XML フォーマットで表示します。

パラメーター

項目	説明
command	パフォーマンス測定対象の実行されるプログラムを指定します。

環境変数

以下の環境変数は、**hpmcount** コマンドの実行に直接、影響を与えます (その他に **MP_*** 環境変数がありますが、これは並列的に実行されるプログラムに影響を及ぼします)。

項目	説明
HPM_EVENT_SET	イベント・セットの 1 つを選択します。この値には、POWER3 システムでは 1 から 6 までの整数、PowerPC 604 RISC マイクロプロセッサー システムでは 1 から 4 までの整数、POWER4 およびそれ以降のシステムでは 1 からプロセッサー依存の上限値までを指定することができます。POWER4 およびそれ以降のシステムでは、イベント・グループ名を選択するためにも、この環境変数が使用されます。セット番号の代わりに、コンマで区切られたイベント・セットのリストを指定することができます。その場合は、カウンター多重方式が選択されます。各イベント・セットは、次のようにカウント・モードで修飾できます。 `event_set_number:counting_modes` **-g** または **-s** オプションが、この変数よりも優先されます。**HPM_EVENT_GROUP** 環境変数が、この変数よりも優先されます。
HPM_EVENT_GROUP	イベント・グループを選択します。イベント・グループのコンマ区切りリストを指定できます。その場合は、カウンター多重方式が選択されます。各イベント・グループは、次のようにカウント・モードで修飾できます。 `event_group_number:counting_modes` **-g** または **-s** オプションが、この変数よりも優先されます。**HPM_EVENT_GROUP** 環境変数が、**HPM_EVENT_SET** 変数よりも優先されます。
HPM_NORMALIZE	データの正規化に使用する基準を示します。**-b** オプションが、この変数よりも優先されます。
HPM_PMD_GROUP	派生メトリック・グループのコンマ区切りリストを指定します。各メトリック・グループは、カウント・モードで修飾できます。**-m** オプションが、この変数よりも優先されます。
HPM_PMD_METRIC	派生メトリックのコンマ区切りリストを指定します。それぞれの派生メトリックは、カウント・モードで修飾できます。**-D** オプションが、この変数よりも優先されます。
HPM_DIV_WEIGHT	POWER4 システム上で重みを付けたフリップの計算に使用される重み (1 よりも大きな整数) を提供します。
MP_CHILD	総計カウントの指定時に並列環境内で使用して、出力結果ファイル名 (*myID*) を補足、結果の照合を同期化、および詳細/デバッグ診断メッセージの一層の厳密識別を行います。
MP_PROCS	プログラム・タスクの数。
HPM_AGGREGATE_OUTPUT	POE アプリケーションに関するカウントを集計します (コマンド・ライン引数 **-a** を強制実行します)。このフラグを使用して、全タスクに対して単一ファイルのパフォーマンス・ファイルを生成します。この環境変数は POE または Load Leveller と連動するだけですが、このシステム上に並列ファイルシステム (GPFS など) が使用可能になっている必要があります。
HPM_LOG_DIR	このフラグを設定すると、**hpmcount** は、指定されたディレクトリーの中に、パフォーマンス・データが入った **hpm_log.***id* ファイルを書き込みます。この書き込みは、通常の出力とは別です。
MP_PARTITION	POE アプリケーションでは、*id* は **MP_PARTITION** で指定した POE ID です。POE アプリケーション以外の場合、これは *pid* です。内部ロックとデータ・ファイルの名前も指定されます。
HPM_MX_DURATION	カウンター多重方式でカウントする場合、このフラグは各タイム・スライスの所要時間を指定します。この時間はミリ秒で表され、10 ミリ秒から 30 秒の範囲内でなければなりません。このフラグを設定しない場合、タイム・スライスの所要時間に使用されるデフォルト値は 100 ミリ秒です。

さらに、ユーザー提供の以下の環境変数では、導出されたメトリック計算に対してメモリー、キャッシュ、および TLB ミス待ち時間の推定値が指定されます。以下の環境変数は、ファイル **HPM_flags.env** に最終的に指定される同一推定値に優先しません (HPM_flags.env が存在する場合)。

- **HPM_MEM_LATENCY**
- **HPM_L3_LATENCY**
- **HPM_L35_LATENCY**

- **HPM_AVG_L3_LATENCY**
- **HPM_AVG_L2_LATENCY**
- **HPM_L2_LATENCY**
- **HPM_L25_LATENCY**
- **HPM_L275_LATENCY**
- **HPM_L1_LATENCY**
- **HPM_TLB_LATENCY**

終了状況

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. **ls** コマンドを実行し、ハードウェア・カウンターからセット 5 のイベントに関する情報を書き込むには、次のように入力します。

   ```
   hpmcount -s 5 ls
   ```

2. **ls** コマンドを実行し、カウンター多重方式を使用してハードウェア・カウンターからセット 5、2、および 9 のイベントに関する情報を書き込むには、次のように入力します。

   ```
   hpmcount -s 5,2,9 ls
   ```

3. **ls** コマンドを実行し、kernel+user+hypervisor モードとユーザー・モードでそれぞれカウントされるデフォルトと cpi_breakdown メトリック・グループに関連する派生メトリックを報告するには、次のように入力します。

   ```
   hpmcount -m default:kuh,cpi_breakdown:u ls
   ```

実装上の固有な条件

hpmcount コマンドでは、PMAPI スレッド・レベル API を使用します。

hpmcount *command* パラメーターは、オプションをもつアプリケーション名に対してコマンド・ラインとして解析されることはありません。解析するには、コマンド・ラインを含むシェル・スクリプトを作成しなくてはなりません。

位置

/usr/bin/perf/pmapi/hpmcount

標準入力

使用されません。

標準出力

パフォーマンス・モニター結果が **stdout** に書き出されます。ただし、コマンド・ラインに **-o** *file* オプションが指定されている場合を除きます。

標準エラー

診断メッセージの場合にのみ使用されます。

ファイル

以下の入力ファイルが、存在すれば、使用されます。

項目	説明
libHPM_events	ユーザー提供のイベント・セット・ファイル。このファイルが、**-s** オプションを使用して指定されたコマンド・ラインに優先することはありません。 POWER3/PowerPC 604 RISC マイクロプロセッサー のカウンターとイベントのペアの形式は *counternumber eventname* です。次に例を示します。 `0 PM_LD_MISS_L2HIT` `1 PM_TAG_BURSTRD_L2MISS` `2 PM_TAG_ST_MISS_L2` `3 PM_FPU0_DENORM` `4 PM_LSU_IDLE` `5 PM_LQ_FULL` `6 PM_FPU_FMA` `7 PM_FPU_IDLE` コンマで区切られたイベントのリストも指定することができます。これを指定すると、カウンター多重方式がオンになります。 `0 PM_CYC,PM_FPU_FIN,PM_IC_MISS` `1 PM_LD_CMPL,PM_INST_CMPL,PM_DC_MISS` `2 PM_INST_CMPL,PM_FPU_WT,PM_INST_CMPL` `3 PM_LD_MISS_DC_XU,PM_CYC,PM_CYC` POWER4 イベント・グループ名の場合の形式は *event_group_name* です。次に例を示します。 `pm_hpmcount1` コンマで区切られたイベントのリストも指定することができます。これを指定すると、カウンター多重方式がオンになります。 `pm_hpmcount1,pm_hpmcount2,pm_basic`
HPM_flags.env	導出されたメトリック計算に使用される、環境変数と値のペアを含むファイル。次に例を示します。 `HPM_L2_LATENCY 12` `HPM_EVENT_SET  5`
./.hpm_lockfile_mp_partition_	ロック・ファイル。このファイルは、**hpmcount** コマンドの内部使用のために予約されます。
./.hpm_datafile_mp_partition_	累積結果ファイル。このファイルは、**hpmcount** コマンドの内部使用のために予約されます。

以下の出力ファイルが使用されます。

項目	説明
file_myID.pid_	**hpmcount** 出力結果に対して **-o** オプションで指定されたファイル。ここで、*myID* は、デフォルト値の `0000` を使用して **MP_CHILD** 環境変数から取られます。
HPM_LOG_DIR/hpm_log.MP_PARTITION または **HPM_LOG_DIR/hpm_log.**_pid_	POE 実行時に集計カウンター用に指定されるログ・ファイル。
./.hpm_lockfile_mp_partition_	ロック・ファイル。このファイルは、**hpmcount** コマンドの内部使用のために予約されます。
./.hpm_datafile_mp_partition_	累積結果ファイル。このファイルは、**hpmcount** コマンドの内部使用のために予約されます。

関連情報:

pmlist コマンド

getrusage コマンド

hpmstat コマンド

目的

システム全体のハードウェア・パフォーマンス・カウンター情報を提供します。

構文

hpmstat [**-b** *time_base*] [**-d**] [**-D** *metrics*] [**-g** *event_groups*] [**-H**] [**-k**] [**-m** *metrics_groups*] [**-o** *file*] [**-r**] [**-s** *set*] [**-T**] [**-U**] [**-u**] [**-x**] [**-@ ALL** | *WparName*] *interval count*

hpmstat [-h]

説明

hpmstat コマンドは、壁時計の実行時刻、ハードウェアのパフォーマンス・カウンター情報、および導き出されたハードウェア・メトリックを提供します。このコマンドを使用可能なのは、root 権限を持つユーザーのみです。

hpmstat は、コマンド・ライン・オプションなしで指定すると、各イベントのデフォルト設定 1 の場合に 1 秒ごとに、ユーザー、カーネル、およびハイパーバイザー (ハイパーバイザー・モードをサポートするプロセッサーの場合) 活動の 1 秒間のデフォルト 1 の繰り返しをカウントします。続いて、このコマンドは、ロウ・カウンター値および導出メトリックを標準出力に書き出します。デフォルトでは、**runlatch** は使用不可にされて、それによってアイドル・サイクル状態での実行時にカウント可能です。

-U オプションを指定時、*interval* はマイクロ秒単位であり、反復 *count* は無限大であり、さらに導出メトリックは計算されず、標準出力には書き出されません。カウンター多重方式が指定されている場合は、このオプションは無視されます。

-T オプション指定時は、出力情報の前にタイム・スタンプ (秒とマイクロ秒) が付けられ、秒数単位ではなくタイム・スタンプとしてタイミング情報が書き込まれます。

モニター対象のイベント・タイプおよびそれに関連するハードウェア・パフォーマンス・カウンターを指定するには、**-s** オプションを設定するか、**HPM_EVENT_SET** 環境変数でイベント・グループ名またはセット番号を指定します。あるいは、**libHPM_events** 入力ファイルの中にカウンターとイベントのペア (POWER3 / PowerPC 604 RISC マイクロプロセッサー) またはイベント・グループ名 (POWER4 およびそれ以降) を指定します (この指定は **HPM_EVENT_SET** よりも優先します)。各セットはカウント・モードで修飾できます。イベントのグループ番号または名前を指定するには、**-g** オプションを設定するか、または **HPM_EVENT_GROUP** 環境変数でイベント・グループのコンマ区切りリストを指定します。同じ方法で、各イベント・グループをカウント・モードで修飾できます。

セット番号の代わりに、コンマで区切られたイベント・セットのリストを指定することができます。その場合は、カウンター多重方式が選択されます。イベント・セットをすべて選択するには、セット番号の値を 0 に設定します。

有効なイベント・セット番号は、1 から上限値 (プロセッサー・タイプにより異なる) までの範囲です。プロセッサー・タイプは **pmlist** コマンドを使用してリストすることができます。

派生メトリックのコンマ区切りリストを指定するには、**-D** オプションを設定します。それぞれの派生メトリックは、カウント・モードで修飾できます。

派生メトリック・グループのリストを指定するには、**-m** オプションを設定するか、または **HPM_PMD_GROUP** 環境変数で派生メトリック・グループのコンマ区切りリストを指定します。これにより、指定されたグループに関連する派生メトリックをすべて選択できます。各メトリック・グループは、カウント・モードで修飾できます。

多重方式でカウントする場合、その結果を使用するには正規化する必要があります。 データの正規化に使用されるデフォルトの基準は、時間基準です。**-b** オプションを使用すると、データの正規化に PURR 時間または SPURR 時間 (プロセッサーでサポートされている場合) を使用できます。データの正規化のための基準は、**HPM_NORMALIZE** 環境変数を使用しても定義できます。

グローバル・ワークロード・パーティション (WPAR) から **hpmstat** コマンドを実行すると、**-@** *WparName* オプションを使用して特定の WPAR をモニターすることができます。 **-@ ALL** オプションを使用すると、システム内のすべてのアクティブな WPAR をモニターし、WPAR ごとのデータを取得することができます。

結果は **-x** オプションを使用して XML フォーマットで出力できます。

フラグ

項目	説明
-@ ALL | *WparName*	活動状況を測定するターゲット WPAR を選択します。**ALL** 値は、**hpmstat** コマンドがシステム内のすべてのアクティブな WPAR を測定し、各 WPAR の活動状況を報告することを意味します。このオプションは、グローバル WPAR から **hpmstat** コマンドを実行した場合のみ使用できます。それ以外の場合は、無視されます。
-b *time_base*	データの正規化のための基準を選択します。使用可能な基準は次のとおりです。 **time** 時間基準 **purr** PURR 時間 (使用可能な場合) **spurr** SPURR 時間 (使用可能な場合) デフォルト値は **time** です。
-d	カウンター多重方式のために詳細なセット・カウントを追加します。
-D *metrics*	評価する派生メトリックのリストを選択します。それぞれの派生メトリックは、次のようにカウント・モードで修飾できます。 `metric:counting_modes` (使用可能なカウント・モードについては、**-m** オプションを参照してください。)
-g *event_groups*	イベントの事前定義グループまたはイベントのグループ名または番号のコンマ区切りリストをリストします。コンマで区切られたグループのリストを使用する場合は、カウンター多重方式が選択されます。各イベント・グループは、次のようにカウント・モードで修飾できます。 `event_group:counting_modes` (使用可能なカウント・モードについては、**-m** オプションを参照してください。)
-H	ハイパーバイザー活動状況のみをカウントします。
-h	ヘルプ・メッセージを表示します。
-k	システム稼働状況のみをカウントします。

項目	説明
-m *metrics_groups*	評価する派生メトリック・グループのリストを選択します。デフォルトの派生メトリック・グループは、特定の派生メトリック・グループに属さないすべての派生メトリックを参照します。各メトリック・グループは、次のようにカウント・モードで修飾できます。 `metric_group_name:counting_modes` 使用可能なカウント・モードは次のとおりです。 **u** ユーザー・モード **k** カーネル・モード **h** ハイパーバイザー・モード **r** runlatch モード **n** 非割り込みモード
-o *file*	出力ファイル名。
-r	**runlatch** を使用可能にし、アイドル・サイクル状態で実行時にカウントを使用不可にします。
-s *set*	イベントの事前定義セットまたはセットのコンマ区切りリストをリストします (1 から *N*、すべて選択する場合は `0`。 **pmlist** コマンドを参照してください)。コンマで区切られたセットのリストを使用する場合は、カウンター多重方式が選択されます。各セットは、次のようにカウント・モードで修飾できます。 `event_set:counting_modes` (使用可能なカウント・モードについては、**-m** オプションを参照してください。)
-T	秒数ではなくタイム・スタンプを書き込みます。
-U	カウント時間間隔をマイクロ秒単位で書き込みます。カウンター多重方式が指定されている場合は、このオプションは無視されます。
-u	ユーザー活動状況のみをカウントします。
-x	結果を VPA XML フォーマットで表示します。

パラメーター

項目	説明
interval	デフォルト値 1 を指定して、秒単位またはマイクロ秒単位のカウント時間間隔を表示します。
count	カウントする反復数を表示します。デフォルトは、秒単位のインターバルを持った 1 であり、オプション **-U** 指定時は無限大です。

環境変数

以下の環境変数は、**hpmstat** コマンドの実行に直接影響します (その他に **MP_*** 環境変数がありますが、これは並列で実行するプログラムの稼働に影響します)。

項目	説明
HPM_EVENT_SET	イベント・セットの 1 つを選択します。この値には、POWER3 システムでは 1 から 6 までの整数、PowerPC 604 RISC マイクロプロセッサー システムでは 1 から 4 までの整数、POWER4 およびそれ以降のシステムでは 1 からプロセッサー依存の上限値までを指定することができます。POWER4 およびそれ以降のシステムでは、イベント・グループ名を選択するためにも、この環境変数が使用されます。各イベント・セットは、次のようにカウント・モードで修飾できます。 `event_set_number:counting_modes` **-g** または **-s** オプションが、この変数よりも優先されます。**HPM_EVENT_GROUP** 環境変数が、この変数よりも優先されます。
HPM_EVENT_GROUP	イベント・グループを選択します。イベント・グループのコンマ区切りリストを指定できます。その場合は、カウンター多重方式が選択されます。各イベント・グループは、次のようにカウント・モードで修飾できます。 `event_group_number:counting_modes` **-g** または **-s** オプションが、この変数よりも優先されます。**HPM_EVENT_GROUP** 環境変数が、**HPM_EVENT_SET** 変数よりも優先されます。
HPM_NORMALIZE	データの正規化に使用する基準を示します。**-b** オプションが、この変数よりも優先されます。
HPM_PMD_GROUP	派生メトリック・グループのコンマ区切りリストを指定します。各メトリック・グループは、カウント・モードで修飾できます。**-m** オプションが、この変数よりも優先されます。
HPM_PMD_METRIC	派生メトリックのコンマ区切りリストを指定します。それぞれの派生メトリックは、カウント・モードで修飾できます。**-D** オプションが、この変数よりも優先されます。
HPM_DIV_WEIGHT	POWER4 システム上で重みを付けたフリップの計算に使用される重み (1 よりも大きな整数) を提供します。
HPM_MX_DURATION	カウンター多重方式でカウントする場合、このフラグは各タイム・スライスの所要時間を指定します。この時間はミリ秒で表され、10 ミリ秒から 30 秒の範囲内でなければなりません。このフラグを設定しない場合、タイム・スライスの所要時間に使用されるデフォルト値は 100 ミリ秒です。

さらに、ユーザー提供の以下の環境変数では、導出されたメトリック計算に対してメモリー、キャッシュ、および TLB ミス待ち時間の推定値が指定されます。以下の環境変数は、ファイル **HPM_flags.env** に最終的に指定される同一推定値に優先しません (HPM_flags.env が存在する場合)。

- **HPM_MEM_LATENCY**
- **HPM_L3_LATENCY**
- **HPM_L35_LATENCY**
- **HPM_AVG_L3_LATENCY**
- **HPM_AVG_L2_LATENCY**
- **HPM_L2_LATENCY**
- **HPM_L25_LATENCY**
- **HPM_L275_LATENCY**
- **HPM_L1_LATENCY**
- **HPM_TLB_LATENCY**

終了状況

項目	説明
0	正常終了。
>0	エラーが発生しました。

例

1. 1 秒のインターバルの間に行われるシステム、ユーザー、およびハイパーバイザーの活動情報 (ハードウェア・カウンターのセット 2 のイベントに関係する情報) を書き込むには、次のコマンドを入力します。

   ```
   hpmstat -s 2
   ```

2. 5 秒のインターバルの間に行われる wpar1 WPAR に対するグループ 0 のイベントに関するユーザーの活動情報と、グループ 1 のイベントに関するシステムの活動情報を書き込むには、次のコマンドを入力します。

   ```
   hpmstat -@ wpar1 -g 0:u,1:k 5
   ```

位置

/usr/bin/perf/pmapi/hpmstat

標準入力

使用されません。

標準出力

パフォーマンス・モニター結果が **stdout** に書き出されます。ただし、コマンド・ラインに **-o** *file* オプションが指定されている場合を除きます。

標準エラー

診断メッセージの場合にのみ使用されます。

ファイル

以下の入力ファイルが、存在すれば、使用されます。

項目	説明
libHPM_events	ユーザー提供のイベント・セット・ファイル。このファイルが、**-s** オプションを使用して指定されたコマンド・ラインに優先することはありません。 POWER3/PowerPC 604 RISC マイクロプロセッサー のカウンターとイベントのペアの形式は *counternumber eventname* です。次に例を示します。 `0 PM_LD_MISS_L2HIT` `1 PM_TAG_BURSTRD_L2MISS` `2 PM_TAG_ST_MISS_L2` `3 PM_FPU0_DENORM` `4 PM_LSU_IDLE` `5 PM_LQ_FULL` `6 PM_FPU_FMA` `7 PM_FPU_IDLE` POWER4 イベント・グループ名の場合の形式は *event_group_name* です。次に例を示します。 `pm_hpmcount1`

項目	説明
HPM_flags.env	導出されたメトリック計算に使用される、環境変数と値のペアを含むファイル。次に例を示します。 `HPM_L2_LATENCY 12` `HPM_EVENT_SET  5`

以下の出力ファイルが使用されます。

項目	説明
file	**-o** オプションを使用して **hpmstat** 出力結果に指定されたファイル。

関連資料:

805 ページの『hpmcount コマンド』

関連情報:

pmlist コマンド

pm_initialize コマンド

Performance Monitor API Programming

---

hps_dump コマンド

目的

Network Terminal Accelerator (NTX) のアダプター・メモリーの内容をホスト・ファイルにダンプします。

構文

hps_dump [**-f** *Name*] [**-d** *Device*]

説明

hps_dump コマンドは、ローダー・インターフェースを使用して、すべてのメモリーをアダプター・ボードからファイルへアップロードします。それにより、後の分析やデバッグ用のシステムのスナップショットが生成されます。ファイルの最初の 1024 バイトには、以下の項目が含まれます。

項目	説明
80	バージョンを含む識別文字列。
80	ホスト・システムからのメモリー・ダンプの日時。
80	コメント。
268	ホスト・アダプターからのログ・テーブル。
32	システム・アドレス・テーブル。
8	メモリー・ダンプの開始アドレスと終了アドレスの範囲。
476	1024 バイトの合計埋め込みバイト。

フラグ

項目	説明
-f*Name*	メモリー・ダンプの名前を指定します。デフォルトのファイル名 **./hpscore** を指定変更するには、このオプションを使用します。
-d *Device*	アダプターのロウ・デバイスのファイル名を指定します。デフォルトのデバイス名 **/dev/rhp0** を指定変更するには、このオプションを使用します。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

監査イベント: 適用外。

例

1. デフォルト・アダプターのメモリー・ダンプを現行ディレクトリー内のファイル **hpscore** に取得するには、次のように入力します。

   ```
   hps_dump
   ```

2. デフォルト・アダプターのメモリー・ダンプを、デフォルト・アダプターの現行ディレクトリー内のファイル hpsdebug に取得するには、次のように入力します。

   ```
   hps_dump -f hpsdebug
   ```

3. アダプター **/dev/rhp1** のメモリーを、デフォルトのアダプターの現行ディレクトリー内のファイル **hpsdebug** にメモリー・ダンプするには、次のように入力します。

   ```
   hps_dump -f hpsdebug -d /dev/rhp1
   ```

ファイル

項目	説明
/usr/bin/hps_dump	**hps_dump** コマンドが入っています。
/dev/rhp0	デフォルトの NTX ロウ・デバイスのファイル名。

関連情報:

/dev/rhp コマンド

---

htable コマンド

目的

ホスト・ファイルをネットワーク・ライブラリー・ルーチンが使うフォーマットに変換します。

構文

/usr/sbin/htable [**-c** *connected-nets*] [**-l** *local-nets*] *input-file*

注: コンマの前後にはスペースを入れないでください。

説明

htable コマンドは、RFC 810 内で指定された構文のホスト・ファイルを、ネットワーク・ライブラリー・ルーチンが使うフォーマットに変換します。この変換によって、**/etc/hosts** ファイル、 **/etc/networks** ファイル、および **/etc/gateways** ファイルの 3 つのファイルが作成されます。

gethostbyname サブルーチンは、 **named** デーモンを使用しない場合、**hosts** ファイルを使用してホスト名をアドレスにマッピングします。**getnetent** サブルーチンは、**networks** ファイルを使用してネットワーク名を番号にマッピングします。

gateways ファイルは、受動インターネット・ゲートウェイを識別する際に **routed** デーモンが使用します。

現行ディレクトリー内にローカルの **hosts** ファイル、**networks** ファイル、または **gateways** ファイル (それぞれ、**localhosts**、**localnetworks**、または **localgateways**) がある場合、そのファイルの内容がプレフィックスとして出力ファイルに追加されます。 **htable** プログラムは、これらのファイルのうち **gateways** ファイルのみを解釈します。内容にプレフィックスが追加されるので、サイトで通常はマスター・データベース内に存在しないローカル・エントリーの保守が可能になります。

フラグ

項目	説明
-c *connected-nets*	ネットワーク経路指定デーモンで **gateways** ファイルを使用する場合、ホストが直接接続しているネットワークのリストを指定します。ネットワークをコンマで分離し、ネットワーク名または標準インターネット・ドット表記法を使用します (例えば、`-c arpanet,128.32,LocalEthernet`)。**htable** コマンドは、指定されたネットワークの 1 つに直接接続されるゲートウェイ、または接続されたネットワーク上の別のゲートウェイから到達できるゲートウェイのみを定義します。
-l *local-nets*	**htable** コマンドでローカルと見なされるネットワークのリストを指定します。 **localhosts** ファイルからのみ、ローカル・ネットワーク上のホストに関する情報を取り出します。ネットワークをコンマで区切り、ネットワーク名または標準インターネット・ドット表記法を使います (例えば、`-l 128.32,local-ether-net`)。**localhosts** ファイルが入力ファイル (コマンド・ラインで指定したファイル) 内でエントリーを指定変更できるように、メイン・データベースからのローカル・ホストに関するエントリーは省略されます。

ファイル

項目	説明
/*CurrentDirectory***/localgateways**	ローカル・ゲートウェイ情報が入っています。
/*CurrentDirectory***/localhosts**	ローカル・ホスト名情報が入っています。
/*CurrentDirectory***/localnetworks**	ローカル・ネットワーク情報が入っています。

関連資料:

723 ページの『gettable コマンド』

関連情報:

routed コマンド

ネットワーク・ファイル

TCP/IP の経路指定ゲートウェイ

hty_load コマンド

目的

Network Terminal Accelerator (NTX) のアダプター構成を表示するかダウンロードします。

構文

hty_load [**-d** *Device*] [**-f** *ConfigFileName*]

説明

hty_load コマンドは、アダプター構成を表示またはダウンロードします。このコマンドをフラグを指定せずに実行すると、システムは **/dev/rhp0** デバイス・ファイルの現行アダプター構成を表示します。 *Device* パラメーターを指定した場合は、**hty_load** コマンドにより構成ファイルが tty ドライバーにロードされます。 tty ドライバーは、このファイルを使用してホスト表示サービス (HPS) とアダプターの両方を設定します。

通常、**hty_load** コマンドは **/etc/rc.ntx** ファイルから起動されます。

構成ファイル

hty_load コマンドは、1 つの構成ファイルを使用してアダプターを設定します。各エントリーはそれぞれ 1 行で表示されます。エントリーとエントリーは改行文字で区切られます。 1 つのエントリー内のフィールドは、タブまたはスペース文字で区切られます。構成ファイルのエントリーには、以下のフィールドが含まれます。

MinorNumber Cluster NumberOfPorts

各フィールドには次の値が入ります。

項目	説明
MinorNumber	ボードのマイナー・デバイス番号を指定します。
Cluster	このフィールドは常に 1 です。

項目	説明
NumberOfPorts	hty デバイスの番号を指定します。デバイスの番号は、ユーザーが使用するアダプターのモデルにより異なります。 2MB ボードで使用できる番号は 1 から 256、8MB ボードで使用できる番号は 1 から 2048 です。

構成ファイルはコメントもサポートします。コメント行は # (ポンド記号) で始まります。コメント文字の右方にあるすべての情報は無視されます。コメント行は改行文字で終わります。

フラグ

項目	説明
-d *Device*	アダプターのロウ・デバイスのファイル名を指定します。デフォルトのデバイス名 **/dev/rhp0** を指定変更するには、このオプションを使用します。
-f *ConfigFileName*	ドライバーの構成ファイル名を指定します。デフォルトの構成ファイルは **/etc/hty_config** ファイルです。

終了状況

このコマンドは次の終了値を戻します。

項目	説明
0	正常終了。
>0	エラーが発生しました。

セキュリティー

アクセス制御: このコマンドを実行するには、root ユーザー権限が必要です。

監査イベント: 該当なし

例

システム構成をロードしてデフォルトのドライバー構成を使用するには、次のように入力します。

```
hty_load -d /dev/rhp0
```

ファイル

項目	説明
/usr/bin/hty_load	**hty_load** コマンドが入っています。
/etc/rc.ntx	**hty_load** コマンドを起動します。
/etc/hty_config	デフォルトの NTX ドライバー構成ファイル名。
/dev/rhp0	デフォルトの NTX ロウ・デバイスのファイル名。

関連情報:

/dev/rhp コマンド

---

hyphen コマンド

目的

ハイフンが付いたワードを検索します。

構文

hyphen [*File* ...]

説明

hyphen コマンドは、1 つ以上の英語ファイルを読み取り、ハイフン付きワードで終るすべての行を見つけて、それらのワードを標準出力に書き出します。 *File* パラメーターは、**hyphen** コマンドにより読み取られる英語ファイルを指定します。デフォルトは標準入力です。ファイルを指定しない場合、または - (ハイフン) を最終ファイル名として指定した場合、**hyphen** コマンドは標準入力を読み取ります。 **hyphen** コマンドはフィルターとして使えます。

注: **hyphen** コマンドは、イタリック体またはアンダーラインの付いたハイフン付きワードを処理できません。 **hyphen** コマンドは、不要な出力をすることがあります。

例

ファイル上でテキスト・フォーマット化プログラムが実行したハイフン付けを検査するには、次のように入力します。

```
mm [Flag...] [File...] | hyphen
```

関連情報:

mm コマンド

troff コマンド

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年).

このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie などの各種テクノロジーの使用について詳しくは、『IBM オンラインでのプライバシー・ステートメントのハイライト』(http://www.ibm.com/privacy/jp/ja/)、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』というタイトルのセクション、および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Windows は、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

［ハ行］

［マ行］

［ヤ行］

［ラ行］

A

D

E

F

G

H

I

L

M

N

P

R

S

T

U

W

［特殊文字］

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21