SunOS リファレンスマニュアル 1M: システム管理コマンド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–1211–13
2007 年 7 月

原典: man pages section 1M : System Administration Commands

Part No: 816-5156-10

Revision A

071015@19028

目次

はじめに

SunOS オペレーティングシステムの初心者でも、熟練したユーザーでも、オンラインのマニュアルページを使ってシステムおよびその機能に関する情報を入手できます。すでに発見されているバ~グについて説明しています。可能な場合は対処法も示しています。マニュアルページは一般に参照マニュアルとして作られています。通常、マニュアルページには、リファレンスマニュアルの内容が含まれています。チュートリアルな要素は含んでいません。

概要

次に、マニュアルページの各セクションと、それらのセクションで示される情報の概要を示します。

- セクション 1 では、オペレーティングシステムで使用できるコマンドをアルファベット順に説明しています。
- セクション 1M では、主にシステムの保守と管理のために使用されるコマンドをアルファベット順に説明しています。
- セクション 2 では、すべてのシステムコールについて説明しています。ほとんどのシステムコールには 1 つ以上のエラー復帰があります。エラー状態は、ほかの場合には返されない戻り値によって示されます。
- セクション 3 では、さまざまなライブラリにある関数について説明しています。ただし、UNIX システムプリミティブを直接呼び出す関数については、セクション 2 で説明しています。
- セクション 4 では、各種ファイルの形式について説明しています。また、ファイル形式を宣言する C 構造体を適用できる場合には、そのつど説明しています。
- セクション 5 では、文字セットテーブルなど、ほかのセクションには該当しない情報を挙げています。
- セクション 6 では、使用できるゲームとデモについて説明しています。
- セクション 7 では、特定のハードウェア周辺装置やデバイスドライバを参照する、さまざまな特殊ファイルについて説明しています。STREAMS ソフトウェアドライバ、モジュール、またはシステムコールの STREAMS 汎用セットについても説明します。

- セクション 9 は、カーネル環境でデバイスドライバを記述するのに必要な参照情報を提供します。ここでは、次のふたつのデバイスドライバについて説明します。デバイスドライバインタフェース (DDI) とドライバ/カーネルインタフェース (DKI) がそれにあたります。
- セクション 9E では、開発者がデバイスドライバに組み込むことができる、DDI/DKI 両用、DDI 専用、および DKI 専用のエントリポイントルーチンについて説明しています。
- セクション 9F では、デバイスドライバで使用できるカーネル関数について説明しています。
- セクション 9S では、ドライバとカーネルの間で情報を共有するためにドライバで使用されるデータ構造について説明しています。

以下に、このマニュアルの項目を説明します。ほとんどのマニュアルページが下記の項目からなる共通の書式で書かれていますが、必要でない項目については省略されています。たとえば、記述すべきバグがコマンドにない場合などは、「使用上の留意点」という項目はありません。各マニュアルページの詳細は各セクションの `intro` を参照してください。マニュアルページの一般的な情報については `man`(1) を参照してください。

名前
: コマンドや関数の名称と概略が示されています。

形式
: この項には、コマンドまたは関数の構文が示されます。標準パスにコマンドやファイルが存在しない場合は、フルパス名が示されます。オプションと引数の順番は、アルファベット順です。特別な指定が必要な場合を除いて、1 文字の引数、引数のついたオプションの順に書かれています。

  この項では、次の特殊文字を使用します。

  [ ]
  : 括弧。このかっこに囲まれたオプションや引数は省略できます。角括弧がない場合、その引数は必須です。

  . . .
  : 省略符号。前の引数に変数を付けたり、引数を複数指定したりできることを意味します (例: ‘`filename.`..‘)。

  |
  : 区切り文字 (セパレータ)。この文字で分割されている引数のうち 1 つだけを指定できます。

  { }
  : 中括弧。このかっこに囲まれた複数のオプションや引数は省略できます。かっこ内を 1 組として扱います。

プロトコル
: この項が使われているのは、プロトコルが記述されているファイルを示すサブセクション 3R だけです。

機能説明
: コマンドの機能とその動作について説明します。つまり、コマンドの機能について簡単に説明します。オプションの説明や使用例

はここでは示されていません。対話形式のコマンド、サブコマンド、リクエスト、マクロ、関数などに関しては「使用法」で説明します。

IOCTL
: この項は、セクション 7 だけに含まれます。 `ioctl`(2) システムコールに適切なパラメータを提供するデバイスクラスのみが `ioctl` と呼ばれます。このデバイスクラスは、独自の見出しを生成します。特定のデバイスに関する `ioctl` は、(そのデバイスのマニュアルページに) アルファベット順に記述されています。デバイスの特定のクラスに関する `ioctl` は、 `mtio`(7I)のように `io` で終わる名前が付いているデバイスクラスのマニュアルページに記載されています。

オプション
: 各オプションがどのように実行されるかを説明しています。「形式」で示されている順に記述されています。オプションの引数はこの項目で説明され、必要な場合はデフォルト値を示します。

オペランド
: コマンドのオペランドを一覧表示し、各オペランドがコマンドの動作にどのように影響を及ぼすかを説明しています。

出力
: コマンドによって生成される出力 (標準出力、標準エラー、または出力ファイル) を説明しています。

戻り値
: 値を返す関数の場合、その値を示し、値が返される時の条件を説明しています。関数が 0 や -1 のような一定の値だけを返す場合は、値と説明の形で示されます。その他の場合は各関数の戻り値について簡単に説明しています。void として宣言された関数はこの項では扱いません。

エラー
: エラー発生時、ほとんどの関数はエラーコードとグローバル変数 `errno` に格納し、エラーの理由を示します。この項では、関数が生成しうるすべてのエラーコードの一覧 (アルファベット順) と、各エラーの発生条件を示します。 もし、ひとつ以上の条件が同じエラーの原因になる場合、各条件は、エラーコードの下に別々の項として記述されます。

使用法
: この項では、使用する際の手がかりとなる説明が示されています。特定の決まりや機能、詳しい説明の必要なコマンドなどが示されています。ここにあげる項は、組み込み型関数の説明で使用されます。

Commands
Modifiers
Variables
Expressions
Input Grammar

使用例	コマンドや関数の使用例または使用方法を説明しています。できるだけ実際に入力するコマンド行とスクリーンに表示される内容を例にしています。例の中には必ず `example%` のプロンプトが出てきます。スーパーユーザーの場合は `example#` のプロンプトになります。例では、その説明、変数置換の方法、戻り値が示されます。それらのほとんどが「形式」、「機能説明」、「オプション」、「使用法」の項からの実例となっています。
環境変数	この項には、コマンドまたは関数が影響を与えるすべての環境変数の一覧を示し、その影響について簡単に説明します。
終了ステータス	コマンドが呼び出しプログラムまたはシェルに返す値と、その状態を説明しています。通常、正常終了には 0 が返され、0 以外の値はそれぞれのエラー状態を示します。
ファイル	マニュアルページが参照するファイル、関連ファイル、およびコマンドが作成または必要とするファイルを示します。各ファイルについて簡単に説明しています。
属性	属性タイプとその対応する値を定義することにより、コマンド、ユーティリティ、およびデバイスドライバの特性を一覧しています。詳細については、`attributes(5)` のマニュアルページを参照してください。
関連項目	関連するマニュアルページ、当社のマニュアル、および一般の出版物が示されています。
診断	このセクションでは、エラー原因となった条件に関する簡単な説明と診断メッセージが示されています。
警告	作業に支障を与えるような現象について説明しています。これは診断メッセージの一覧ではありません。
注意事項	この項では、このマニュアルページの他のどの項にも記載されない追加情報を提供します。マニュアルページの内容とは直接関係のない事柄も参照用に扱っています。重要不可欠な情報はこの項では説明しません。
使用上の留意点	すでに発見されているバグについて説明しています。可能な場合は対処法も示しています。

参照

序章

名前

Intro – 管理コマンドおよびアプリケーションプログラムの序章

機能説明

本セクションでは、主にシステムの保守や管理に使用するコマンドを、アルファベット順に説明します。

コマンドが仮想ファイルシステムのアーキテクチャに合わせて再構成されているため、同じ名前で始まる複数のマニュアルページが存在します。たとえば、`mount` の名前に関しては、mount(1M)、`mount_cachefs`(1M)、`mount_hsfs`(1M)、`mount_nfs`(1M)、`mount_tmpfs`(1M)、mount_ufs(1M) のように 6 つのマニュアルページが存在します。このような場合、最初のマニュアルページにだけ、その総称コマンドの構文およびオプションが説明されています。つまり、これらのオプションは、すべてのファイルシステムのタイプに適用されるということです。以降のマニュアルページには、そのコマンドの機能のうちファイルシステムタイプに特有な部分が説明されています。このようなマニュアルページ名には、下線 (_) とそのコマンドに関係するファイルシステムのタイプ名が伴われています。管理者は、このようなファイルシステムに特有な部分を直接呼び出してはなりません。総称コマンドは、すべてのファイルシステムに共通なインタフェースを提供します。ですから、ファイルシステムタイプに特有なマニュアルページは、個々のコマンドを説明していると考えるべきではなく、コマンドのファイルシステムに特有な面を詳細に述べたものだと考えるべきです。

コマンドの構文

特に説明しないかぎり、本セクションで説明するコマンドは、次の構文に従って、オプションやその他の引数を受け付けます。

name [*option*(*s*)] [*cmdarg*(*s*)]

各表記の意味は次のとおりです。

name 実行可能ファイルの名前です。

option `-noargletter`(*s*) または

`-argletter`< >*optarg*。

< > は、空白 (オプション)。

noargletter 引数が必要でないオプション 1 文字を表します。

argletter 引数が必要なオプション 1 文字を表します。

optarg *argletter* に必要な引数 (文字列) です。

cmdarg パス名 (または他のコマンドの引数)。– だけを指定すると標準入力を表します。

属性

このセクションにリストされた属性については `attributes`(5) のマニュアルページを参照してください。

関連項目

`getopt`(1), `getopt`(3C), `attributes`(5)

診断　　終了時、すべてのコマンドは、正常に終了すると 0 を返します。ゼロでない値を返した場合、間違った引数を指定した、不良で受け入れることができないデータを指定した、その他、現在はうまく処理できないなどの障害を示します。このような値は、「終了コード」、「終了ステータス」、「リターンコード」などさまざまな呼ばれ方をします。そして、特別な使い方がある場合にかぎって説明されます。

注意事項　　すべてのコマンドにおいて標準の構文に準拠しているわけではありません。

参 照

システム管理コマンド

名前

accept, reject – 印刷要求の受付または拒否

形式

```
accept  destination...

reject  [-r reason] destination...
```

機能説明

accept コマンドは、指定した宛先への印刷要求を待ち行列に加えることを許可します。

reject コマンドは、指定した宛先への印刷要求を待ち行列に加えることができないようにします。

宛先が印刷要求を受け付けたか、または拒否したかを確認するには、lpstat -a を実行してください。

accept と request は印刷サーバー上で実行する必要があります。クライアントシステム上では意味をもちません。

オプション

reject コマンドには、次のオプションを指定できます。

-r *reason*　*destination* への印刷要求を抑止する理由を 文字列として記述します。*reason* に空白が含まれる場合は 引用符で囲んでください。*reason* は、プリンタの状況を調べる lpstat -a コマンドの出力中に表示されます。*reason* のデフォルト値は、既存の宛先に関しては unknown reason、システムに追加されたばかりで まだ印刷要求を受け付けたことのない宛先に関しては new printer となります。

オペランド

次のオペランドを指定できます。

destination　印刷要求を受け付ける、または拒否する宛先名。宛先にはプリンタ名やプリンタクラスを指定します (lpadmin(1M) 参照)。*destination* は名前を使用して指定します。名前の命名規約については printers.conf(4) のマニュアルページを参照してください。

終了ステータス

以下の終了ステータスが返されます。

0　正常終了

0 以外　エラーが発生した

ファイル

/var/spool/lp/*　LP 印刷待ち行列

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWpcu
CSI	対応済み (「注意事項」参照)

関連項目

enable(1), lp(1), lpstat(1), lpadmin(1M), lpsched(1M), printers.conf(4), attributes(5)

注意事項

accept と reject は印刷サーバーのスプール用システムの待ち行列に対してのみ有効です。したがって、クライアントシステムから出された要求については、印刷サーバーのスプール用システムが取り消したり受け取ったりするまでは、クライアントシステムの印刷待ち行列に入れられたままの状態になります。

accept は *destinations* 名を除いて CSI 対応が可能です。

名前

acct, acctdisk, acctdusg, accton, acctwtmp, closewtmp, utmp2wtmp – アカウンティングおよびいろいろなアカウントコマンドの概要

形式

```
/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg  [-u filename] [-p filename]

/usr/lib/acct/accton  [filename]

/usr/lib/acct/acctwtmp  reason filename

/usr/lib/acct/closewtmp

/usr/lib/acct/utmp2wtmp
```

機能説明

アカウンティングソフトウェアは、アカウンティングシステムを構築するためのツール群です (C 言語のプログラムおよびシェルプロシージャから構成される)。acctsh(1M) では、C 言語のプログラムの一番上に組むシェルプロシージャ群を説明します。

接続時間のアカウンティングは記録を /var/adm/wtmpx (utmpx(4) を参照) に書き込むプログラム群により管理されます。acctcon(1M) では、このファイルをセッションおよび課金記録に変換するプログラムを説明します。また、acctmerg(1M) では、この課金記録について集約します。

プロセスアカウンティングは、システムカーネルによって行われます。プロセスを終了すると、1 つのプロセスにつき 1 つのレコードが、あるファイル (通常は、/var/adm/pacct) に書き込まれます。acctprc(1M) では、課金のためにこのデータを集約します。コマンドの使用状況を集約するには、acctcms(1M) を使用します。acctcom(1) を用いて、現在のプロセスデータを調査することができます。

acctmerg (acct.h(3HEAD) の tacct フォーマットを参照) を使うと、プロセスのアカウンティングレコードおよび接続時間のアカウンティングレコード (あるいは acct.h(3HEAD) で説明する tacct フォーマットのアカウンティングレコードのいずれか) をアカウンティングレコードの合計にマージしたり、集約したりすることができます。どのアカウンティングレコードにも、あるいはすべてのアカウンティングレコードにでも prtacct (acctsh(1M) を参照) をフォーマットに使用します。

acctdisk は、ユーザー ID、ログイン名およびディスクブロック数がある行を読み取り、他のアカウンティングレコードとマージすることができる アカウンティングレコードの合計に、それらの情報を変換します。入力ファイルが壊れているか、または正しくフォーマットされていない場合は、acctdisk はエラーを返します。

acctdusg は、その標準入力を (通常は、find / -print から) 読み取り、ログインによるディスクの資源消費 (間接ブロックを含む) を計算します。

accton では、引数がない場合、プロセスアカウンティングをオフにします。*filename* を指定する場合は、そのファイルは、カーネルがプロセスのアカウンティングレコード (acct(2) および acct.h(3HEAD) を参照) を追加するための、既存のファイル名である必要があります。

acctwtmp は、utmpx(4) のレコードを *filename* へ書き込みます。レコードには、現在の時間および *reason* を説明する文字列が入っています。ACCOUNTING のレコードの型が割り当てられます (utmpx(4) を参照)。*reason* には、11 文字以下の文字列、数、$、または空白を指定してください。たとえば、以下は、リブートのプロシージャおよびシャットダウンのプロシージャでの使用例です。

```
acctwtmp "acctg on" /var/adm/wtmpx
acctwtmp "acctg off" /var/adm/wtmpx
```

現在ログオンしている各ユーザーについて、closewtmp は、偽の DEAD_PROCESS レコードを /var/adm/wtmpx ファイルに入れます。runacct (runacct(1M) を参照) がこの偽の DEAD_PROCESS を使用することにより、接続アカウンティングプロシージャは runacct を起動する前にログオンしたユーザーが使用した時間を追跡することができます。

現在ログオンしている各ユーザーについて、runacct は utmp2wtmp を使用して、runacct が作成したファイル /var/adm/wtmpx にエントリを作成します。この /var/adm/wtmpx のエントリを見れば、引き続いて起こる runacct の呼び出しで、現在ログインしているユーザーの接続時間がわかります。

オプション

以下のオプションを指定できます。

-u *filename*　誰にも課金されないファイル名からなるレコードを、*filename* に入れます (ディスクの課金を拒否しようとしたユーザーを見つけるための潜在的な情報源となる) 。

-p *filename*　-p は、パスワードファイル *filename* を指定します。パスワードファイルが /etc/passwd である場合は、このオプションは不要です。

環境

LC_* 変数 (LC_CTYPE、LC_MESSAGES、LC_TIME、LC_COLLATE、LC_NUMERIC、LC_MONETARY) (environ(5) 参照) のいずれも環境に設定されていなければ、それぞれ対応する ロケールのカテゴリにおける acct の動作は、環境変数 LANG によって決定されます。もし、LC_ALL が設定されていれば、その内容が LANG 変数やその他の LC_* 変数より優先されます。上記の変数が環境にまったく設定されていなければ、C ロケール (米国スタイル) が acct の動作を決定します。

LC_CTYPE　acct の文字の処理方法を決定します。LC_CTYPE に有効な値が設定されていると、acct は、そのロケール にあった文字を含むテキストやファイル名を表示および処理できます。acct は拡張 UNIX コード (EUC) も表示および処理できます。この場合、文字は 1 バイト幅、2 バイト幅、3 バイト幅のいずれも使用できます。また、acct は 1、 2、またはそれ以上のカラム幅の EUC 文字も処理することができます。C ロケールにおいては、ISO 8859-1 の文字だけが有効です。

LC_TIME　acct の日付および時間のフォーマットの処理方法を決定します。C ロケールにおいては、日付および時間の処理方法は米国ルールに従います。

ファイル

`/etc/passwd`	ログイン名からユーザー ID への変換に用いる
`/usr/lib/acct`	本マニュアルの 1M 章に含まれるアカウンティングコマンドが置かれる
`/var/adm/pacct`	現在のプロセスアカウンティングファイル
`/var/adm/wtmpx`	ユーザーアクセスまたは管理情報の履歴

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWaccu

関連項目

`acctcms`(1M), `acctcom`(1), `acctcon`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct.h`(3HEAD), `passwd`(4), `utmpx`(4), `attributes`(5), `environ`(5)

『『Solaris のシステム管理 (基本編)』』

名前

afbconfig, SUNWafb_config – AFB グラフィックスアクセラレータの設定

形式

```
/usr/sbin/afbconfig  [-dev device-filename] [-res video-mode
 [now | try]  [noconfirm | nocheck]] [-file machine | system]
 [-deflinear true | false] [-defoverlay true | false]
 [-overlayorder first | last] [-expvis enable | disable]
 [-sov enable | disable] [-maxwinds n] [-extovl enable | disable]
 [-g gamma-correction-value] [-gfile gamma-correction-file]
 [-propt] [-prconf] [-defaults]

/usr/sbin/afbconfig  [-propt] [-prconf]

/usr/sbin/afbconfig  [-help] [-res ?]
```

機能説明

afbconfig は、AFB グラフィックスアクセラレータおよび AFB 対応の X11 ウィンドウシステムのデフォルトの一部を設定します。

afbconfig の次の形式では、指定したオプションを OWconfig ファイルに保存します。

```
/usr/sbin/afbconfig  [-devdevice-filename] [-res video-mode
 [now | try]  [noconfirm | nocheck]] [-file machine | system]
 [-deflinear true | false] [-defoverlay true | false]
 [-overlayorderfirst | last] [-expvisenable | disable]
 [-sov enable | disable] [-maxwindsn] [-extovl enable | disable]
 [-ggamma-correction-value] [-gfilegamma-correction-file]
 [-propt] [-prconf] [-defaults]
```

これらのオプションは、次にウィンドウシステムをそのデバイスで実行するときに AFB デバイスを初期化するために使用されます。OWconfig ファイル内のオプションの更新は、異なるウィンドウセッションや再起動したシステムでも有効となります。

-prconf、-propt、-help、-res ? オプションだけを起動する次の形式では、OWconfig ファイルは更新されません。

```
/usr/sbin/afbconfig  [-propt] [-prconf]

/usr/sbin/afbconfig  [-help] [-res ?]
```

また、次の形式では、その他のオプションはすべて無視されます。

```
/usr/sbin/afbconfig  [-help] [-res ?]
```

オプションは、一度に 1 つの AFB デバイスに対してのみ指定することができます。複数の AFB デバイスに対してオプションを指定するには、afbconfig コマンドを複数回起動する必要があります。

afbconfig で指定できるのは、AFB 固有のオプションだけです。デフォルトの表示色数、デフォルトのビジュアルクラスなどを指定する通常のウィンドウシステムのオプションは、openwin コマンド行のデバイス修飾子で指定してください。

ユーザーは、更新する OWconfig ファイルを指定することもできます。デフォルトでは、/usr/openwin ディレクトリツリーにあるマシン固有のファイルが更新されます。別のファイルを指定するには、-file オプションを使用します。たとえば、/etc/openwinディレクトリツリーにあるシステム共通の OWconfig ファイルを代わりに更新することができます。

これらの標準 OWconfig ファイルのどちらもスーパーユーザーのみが書き込みを行えます。したがって、スーパーユーザーが所有するafbconfig プログラムは、setuid による root の権限で実行されます。

オプションのデフォルト値

afbconfig コマンド行で指定されていないオプションについては、対応する OWconfig ファイル中のオプションは更新されず、ファイル内の値がそのまま使用されます。ウィンドウシステムを実行する際に、afbconfig による AFB オプションの指定がまったくなかった場合は、デフォルト値が使用されます。オプションのデフォルトは次のとおりです。

-dev	/dev/fbs/afb0
-file	machine
-res	none
-deflinear	false
-defoverlay	false
-linearorder	last
-overlayorder	last
-expvis	enabled
-sov	enabled
-maxwids	32
-extovl	enabled
-g	2.22

-res オプションのデフォルト値 none とは、ウィンドウシステムが実行された場合に、画面解像度がそのデバイスに現在プログラムされている表示モードになることを意味しています。

これによって、PROM によってデバイスの解像度を指定しているユーザーとの共用性が保てます。(GX などの) 一部のデバイスでは、PROM が表示モードを指定する唯一の手段です。これは、デフォルトの AFB 表示モードは、最終的に PROM によって決まることを意味しています。

オプション

次のオプションがサポートされています。

`-defaults`
すべてのオプションの値をそれぞれのデフォルト値に戻します。

`-deflinear` true | false
AFB には、2 種類の画像表示形式があります。リニア画像と非リニア画像です。リニア画像はガンマ補正され、非リニア画像は補正されません。リニア画像版も非リニア画像版も、ともに持つ画像表示形式が 2 つあります。24 ビットトゥルーカラーと 8 ビットスタティックグレーです。

`true` のときは、デフォルトの画像表示形式として、デフォルトで選択されたオプション (特に、Xsun(1) の defdepth および defclass オプション。詳細は OpenWindows のマニュアルページを参照) を満たすリニア画像がセットされます。

false のとき、または他のデフォルトで選択されたオプションを満たすリニア画像がないときは、これらの他のオプションを満たす、非リニア画像がデフォルトとして選択されます。AFB にはリニアオーバーレイ画像表示形式がないため、`-defoverlay` オプションが存在するときには、このオプションを使用することはできません。

`-defoverlay` true | false
AFB が、残りの AFB 画像から切り離されたピクセルを持つ 8 ビット疑似カラー画像を提供します。これを、オーバーレイ画像といいます。この画像表示形式で作成されたウィンドウは、他の画像表示形式で作成されたウィンドウに影響を与えません。逆に、他の画像表示形式で作成されたウィンドウは、オーバーレイウィンドウに影響を与えます。

この画像を使用して作成されたウィンドウで使用することができる色の数は `-extovl` オプションの設定に依存します。`-extovl` オプションが有効になっている場合は、256 種類の不透明カラーの値による拡張オーバーレイを使用することができます(`-extovl` を参照)。`-extovl` が無効になっている場合は、拡張オーバーレイを使用することはできず、この画像には、不透明カラーの (256 `-maxwids`) の値が使用されます (`-maxwids`を参照)。

`-defoverlay` の値が `true` である場合には、オーバーレイ画像がデフォルト画像になります。`-defoverlay` の値が `false` の場合には、他のデフォルトで選択された `def`、`depth` および `defclass`オプションを満たすオーバーレイでない画像表示形式が、デフォルトの画像表示形式として選択されます。詳細は、OpenWindows のマニュアルページを参照してください。

`-defoverlay true` オプションが使用されるときは、常に `openwin` コマンド行で選択されたデフォルトの深さとクラスは 8 ビット疑似カラーである必要があります。それ以外の場合は、警告メッセージが出力され、`-defoverlay` オプションは `false` として扱われます。

`-deflinear` オプションが存在するときには、AFB にはリニアオーバーレイ画像表示形式がないため、このオプションは使用することができません。

-dev *device-filename*
AFB 特殊ファイルを指定します。デフォルトは/dev/fbs/afb0 です。

-expvis enable | disable
enable にすると、OpenGL Visual Expansion が起動されます。選択された画像表示形式グループ (8 ビット PseudoColor、24 ビット TrueColor など) が画像表示形式リストに見つかります。

-extovl enable | disable
enable にすると、拡張オーバーレイを使用することができます。このオーバーレイ画像には 256 種類の不透明カラーがあります。SOV 画像には 255 種類の不透明カラーと 1 種類の透明カラーがあります。

また、このオプションは、ハードウェアによる透明カラーを有効にするため、SOV 画像を使用するウィンドウで、より高い性能が得られます。

-file machine | system
更新する OWconfig ファイルを指定します。machine が指定された場合は、/etc/openwin ディレクトリツリーにあるマシン固有の OWconfig ファイルが更新されます。system が指定された場合は、/usr/openwin ディレクトリツリーにある共通の OWconfig ファイルが更新されます。指定されたファイルがない場合は、新たに生成されます。ほかのオプションを指定していない場合、このオプションは効果がありません。デフォルトは *machine* です。

-g *gamma-correction-value*
ガンマ補正の値を変えることができます。すべてのリニア画像ではガンマ補正を使用することができます。デフォルトでは、*gamma-correction-value*は 2.22 です。0 より小さい値は無効 (不正) です。ガンマ補正の値はリニア画像に適用され、リニア画像の有効ガンマ値は 1.0 になります。これは、XSolarisGetVisualGamma(3) によって返される値です。この機能については、XSolarisGetVisualGamma(3) を参照してください。

このオプションは、ウィンドウシステムが稼動しているときに使用することができます。ガンマ補正の値を変更すると、リニア画像を使用して表示されているすべてのウィンドウが影響を受けます。

-gfile *gamma-correction-file*
指定されたファイル (*gamma-correction-file*) からガンマ補正表を読み込みます。*gamma-correction-file*は、各行が R、G、B チャネルの値を持つように書式化されている必要があります。それらの値は、16 進数で指定し、値と値の間は 1 つ以上のスペースで区切ります。また、*gamma-correction-file*は、そのような 3 つの値の組が 256 種類定義されます。

*gamma-correction-file*の例を以下に示します。

```
0x00 0x00 0x00
0x01 0x01 0x01
0x02 0x02 0x02
```

```
...
...
0xff 0xff 0xff
```

このオプションを使用することによって、ウィンドウシステムが稼動しているときにガンマ補正表を読み込むことができます。新しいガンマ補正は、このリニア画像によって表示されているすべてのウィンドウに影響を与えます。ユーザーが指定した表によってガンマ補正を行う際は、ガンマ補正の値は定義されません。デフォルトでは、ウィンドウシステムはガンマ補正値として 2.22 を使用し、このガンマ補正値に対応してウィンドウシステムが作成したガンマ補正表を読み込みます。

`-help`
`afbconfig` コマンド行のオプションと機能の概要を一覧で表示します。

`-linearorder` first | last
`first` のときには、AFB スクリーン用の X11 スクリーン画像表示形式リスト上で、リニア画像が非リニア画像より前に表示されます。`last` のときには、非リニア画像は、リニア画像より前に表示されます。

`-maxwids` *n*
ウィンドウ ID s (WIDs) として使用するために予約される最大数の AFB X チャネルピクセル値を指定します。オーバーレイカラーマップのピクセル値の残りは、通常の X11 の未使用のカラーピクセルのために使用されます。確保された WIDs は、(XGL などの) 3 次元グラフィックスウィンドウ、MBX ウィンドウと、デフォルト以外の画像表示形式をもつウィンドウにより発生順に割り当てられます。X チャネルコードの 0 から(255- *n*) は、未使用のカラーピクセルです。(255 - *n* + 1) から 255 の X チャネルコードは、WID として使用するために予約されます。適切な値は、1、2、4、8、16、32、64です。

このオプションは `-extovl` が無効になっている場合のみ使用することができます。

`-overlayorder` first | last
`first` のときには、AFB スクリーン用の X11 スクリーン画像表示形式リスト上で、8 ビット疑似カラーオーバーレイ画像が、非オーバーレイ画像より前に表示されます。`last` のときには、非オーバーレイ画像は、オーバーレイ画像より前に表示されます。

`-propt`
`-file` オプションで指定された `OWconfig` ファイルに書かれた AFB オプションの値のうち、`-dev` オプションで指定されたデバイスに対するものすべてを表示します。`afbconfig` の呼び出しが終了した後に、`OWconfig` ファイルに書き込まれるオプションの値を表示します。

次に表示例を示します。

```
--- OpenWindows Configuration for /dev/fbs/afb0 ---
OWconfig: machine
Video Mode: 1280x1024x76
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
                 Overlay Visuals are last
OpenGL Visual Expansion: enabled
Server Overlay Visuals: enabled
Extended Overlay: enabled
Underlay WIDs: 64 (not configurable)
Overlay WIDs: 4 (not configurable)
Gamma Correction Value: 2.220
Gamma Correction Table: Available
```

-prconf
AFB ハードウェア構成を表示します。

次に表示例を示します。

```
--- Hardware Configuration for /dev/fbs/afb0 ---
Type: double-buffered AFB with Z-buffer
Board: rev 0 (Horizontal)
Number of Floats: 6
PROM Information: @(#)afb.fth x.xx xx/xx/xx
AFB ID: 0x101df06d
DAC: Brooktree 9070, version 1 (Pac2)
3DRAM: Mitsubishi 130a, version x
EDID Data: Available - EDID version 1 revision x
Monitor Sense ID: 4  (Sun 37x29cm RGB color monitor)
Monitor possible resolutions: 1024x768x77, 1024x800x84, 1
        1152x900x76, 1280x1024x67, 1280x1024x76, 960x680xx108s
Current resolution setting: 1280x1024x76
```

-sov enable | disable
enable にすると、ルートウィンドウのSERVER_OVERLAY_VISUALS 属性が有効になります。SOV 画像が転送され、それらの透過タイプ、値、階層は、この属性によって参照することができます。disable にすると、SERVER_OVERLAY_VISUALS 属性は定義されません。SOV 画像は転送されません。

-res *video-mode* [now | try [noconfirm | nocheck]]
指定した AFB デバイスに接続されているモニターを制御する際に使われる表示モードを指定します。

組み込まれている表示モードの形式は次のとおりです。*width*x*height*x*rate* *width* はピクセル単位のスクリーン幅、*height* はピクセル単位のスクリーンの高さ、*rate* は画面を垂直方向に再描画する周期です。

960x680x112s や 960x680x108s の s 接尾辞は、これらが立体表示モードであることを意味します。640x480x60i や 768x575x50i の i 接尾辞は、インタレース表示タイミングを有効にします。この接尾辞がない場合は、ノンインタレースタイミングが使用されます。

便宜上、-res にリフレッシュレートを指定する際、値の直前に x の代わりに @ を使用できます。たとえば、1280x1024@76 のように指定できます。AFB が対応している一部の表示モードには、モニターが対応していない場合があります。また、AFBがサポートする表示モードにも、モニターがサポートしていないものがあります。AFB デバイスとモニターの両方がサポートしている表示モードのリストは、-res ? オプション付きの afbconfig (形式の項に記された 3番目の形式) を実行することによって得ることができます。

AFB がサポートしている表示モードのリストを次に示します。

1024x768x60
1024x768x70
1024x768x75
1024x768x77
1024x800x84
1152x900x66
1152x900x76
1280x800x76
1280x1024x60
1280x1024x67
1280x1024x76
960x680x112s　(立体表示)
960x680x108s　(立体表示)
640x480x60
640x480x60i　(インタレース)
768x575x50i　(インタレース)

便宜上、AFB がサポートしている表示モードのいくつかには記号名が定義されています。*width*x*height*x*rate* の形式の代わりに、記号名を -res オプションの引数として指定することができます。記号名 none は、ウィンドウシステムを実行すると、画面の解像度は現在デバイスにプログラムされている表示モードになることを意味します。

AFB がサポートしている表示モードのリストを次に示します。

記号名　対応する表示モード
svga　1024x768x60
1152　1152x900x76

1280 1280x1024x76
stereo 960x680x112s
ntsc 640x480x60i
pal 768x575x50i
none (上記参照)

-res オプションには、表示モードの直後に次の追加引数を指定することができます。追加引数は、単独でも複数でも指定できます。

noconfirm -res オプションを指定した際に、システムが使用不可であっても、表示出力のない状態になる場合があります。このような状況は、特定のコードが読み込まれた際のモニターセンスコードにあいまいさがあった場合などに発生します。このような事態を避けるために afbconfig のデフォルトの動作では、この問題についての警告メッセージと、処理を継続するかどうかを確認するメッセージを表示します。noconfirm オプションを指定すると、afbconfig コマンドはこの確認をせずに、要求のあった表示モードにプログラムします。このオプションは、afbconfig がシェルスクリプトから実行されている場合に便利です。

nocheck このオプションを指定すると、モニターセンスコードに基づく通常のエラーチェックが行われません。ユーザーによって指定された表示モードは、現在接続されているモニターに適切かどうかにかかわらず受け付けられます。このオプションは、AFB デバイスに異なるモニターを接続する場合に便利です。このオプションの指定は、noconfirm の指定も兼ねます。

now OWconfig ファイルの表示モードを更新するとともに、AFB デバイスが指定した表示モードにただちにプログラムされます。この機能は、ウィンドウシステムを開始する前に表示モードを変更する際に便利です。

対象となるデバイスが稼働している間 (たとえば、ウィンドウシステムの稼働中) に、この引数をafbconfig に指定しないでください。予期しない結果になることもあります。now 引数を指定して afbconfig コマンドを実行する場合は、最初にウィンドウシステムを終了してください。now 引数がウィンドウシステムのセッション中に使用された場合、表示モードはただちに変更されますが、画面の幅や高さはそのセッションが終了して次のセッションに入るまで変更されません。さらに、立体表示モードではシステムが変更を認識しないことがあります。したがって、ウィンドウシステムの稼働中には絶対に now オプションを指定しないでください。

try このオプションを指定すると、指定した表示モードが試験的にプログラムされます。ユーザーは、指定した表示モードを使用する場合は、メッセージが表示されてから 10 秒以内に「y」を入力しま

す。表示されたモードを使用しない場合は、10 秒以内に任意の文字を入力します。「y」またはReturn キー以外の文字の入力は、すべて「使用しない」と判断され、以前の表示モードに戻されるため、OWconfig ファイル中の表示モードは書き換えられません (その他の指定されたオプションは有効となります)。Return キーの入力があった場合は、新しい表示モードを保持するかどうかを yes または no で確認するメッセージが表示されます。構成するデバイスが使用中である場合 (たとえば、ウィンドウシステムを実行している場合)、try サブオプションを afbconfig に指定してはなりません。予期せぬ結果になることがあります。try サブオプションを指定して afbconfig を実行する前には、ウィンドウシステムを停止しておく必要があります。

使用例

例 1 モニターの種類の変更

モニターの種類を、垂直周波数 76 Hz で解像度 1280x1024 に変更する例を以下に示します。

```
example% /usr/sbin/afbconfig -res 1280x1024x76
```

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWafbcf

関連項目

mmap(2), attributes(5)

名前

automount – 自動マウントポイントのインストール

形式

```
/usr/sbin/automount  [-t duration] [-v]
```

機能説明

automount ユーティリティーは、autofs マウントポイントをインストールし、automount マップと各マウントポイントを対応付けます。ローカルまたは分散された automount マップのどちらかに重要なエントリがあり、automountd(1M) デーモンがまだ起動していない場合、automount ユーティリティーは automountd(1M) デーモンを起動します。autofs ファイルシステムは、自身のファイルシステム内のディレクトリへのアクセスを監視し、automountd(1M) デーモンに通知します。このデーモンはマップを使用してファイルシステムを特定し、autofs ファイルシステム内の参照ポイントにそのファイルシステムをマウントします。マップは、/etc/auto_master マップのエントリを使用するか、または直接マップを使用することによって、autofs マウントに割り当てることができます。

一定時間内 (デフォルトでは 10 分間) にファイルシステムに対してアクセスがなければ、automountd デーモンはファイルシステムのマウントを解除します。

ファイル /etc/auto_master により、すべての autofs マウントポイントの位置が決定されます。デフォルトでは、このファイルには 3 つのエントリが含まれます。

```
# Master map for automounter
#
+auto_master
/net          -hosts     -nosuid
/home         auto_home
```

+auto_master エントリは、外部 NIS または NIS+ マスターマップへの参照です。マスターマップが存在する場合、+auto_master エントリの位置へ読み取られます。マスターファイルの残りのエントリは、autofs マウントが行われるディレクトリを指定し、さらにそのディレクトリに対応付けるオートマウントマップを指定します。各エントリの 3 番目のオプションフィールドには、任意のマウントオプションを指定できます。これらのオプションは、マウントオプションが明示的に指定されていない、マップ内のすべてのエントリに適用されます。automount コマンドは通常、引数を指定しないで実行します。automount コマンドは、/etc/auto_master 内のエントリと /etc/mnttab 内の autofs マウントのその時点のリストを比較し、autofs マウントを追加、削除、または変更することによって、/etc/auto_master の情報を /etc/mnttab ファイルに反映させます。automount コマンドはブート時に、マスターマップからすべての autofs マウントをインストールします。その後、マスターマップまたは直接マップの新規エントリを autofs マウントにインストールするか、削除されたエントリに関してのマウントを解除します。

Solaris Trusted Extensions でのオートマウント

Solaris Trusted Extensions で構成されたシステムの場合、マルチレベルのホームディレクトリアクセスが可能となるように、追加の処理が実行されます。現在のゾーンよりも下位のラベルを持つゾーンのリストが生成され、デフォルトの auto_home オートマウントマップがその時点で存在していない場合はそれらのマップが生成されます。これらのオートマウントマップの名前は、

auto_home_<*zonename*> となります。ここで、<*zonename*> は各ゾーンの下位ゾーンの名前です。続いて、そのような auto_home マップがマスターマップ内に明示的または暗黙的に含まれているかどうかにかかわらず、それらの各マップの autofs マウントが実行されます。ゾーンは、標準 auto_home マップの autofs マウントを行う代わりに、自身のゾーン名が末尾に追加された auto_home ファイルを使用します。各ゾーンの auto_home マップに一意の名前が割り当てられるのは、共通のネームサーバーを使用するすべてのゾーンがそれらのマップを保守および共有できるようにするためです。

デフォルトでは、各ゾーンのブート時に、下位ゾーンのホームディレクトリが読み取り専用として、/zone/<*zonename*>/export/home の下にマウントされます。デフォルトの auto_home_<*zonename*> オートマウントマップでは、/zone/<*zonename*>/home/<*username*> への lofs 再マウントのソースディレクトリとして、そのパスが指定されます。たとえば、上位ゾーンによって生成されたファイル auto_home_public 内に次の情報が含まれているとします。

+auto_home_public

* -fstype=lofs :/zone/public/export/home/&

ホームディレクトリが参照された際に、その名前が auto_home_public マップ内のほかのどのキーとも一致しなかった場合、それはこのループバックマウント指定に一致します。このループバック一致が発生し、かつその名前が、public ゾーン内にホームディレクトリを持たない有効なユーザーに対応していた場合、ユーザーに代わってディレクトリが自動的に作成されます。

オプション

サポートしているオプションは、次のとおりです。

-t *duration*
: ファイルシステムが使用されていないときに、マウントしたままにしておく時間の間隔を秒数で指定します。デフォルトは 10 分です。

-v
: 詳細表示 (Verbose) モードです。autofs のマウント、マウント解除、またはその他の関連情報を通知します。

使用法

マップエントリの形式

単純マップエントリ (マッピング) の形式は、次のとおりです。

key [*-mount-options*] *location* . . .

key は、直接マップで使用する場合はマウントするディレクトリのフルパス名であり、間接マップの場合はサブディレクトリからの相対名です。*mount-options* は、コンマで区切った mount オプションのリストです。*location* には、ファイルシステムがマウントされるディレクトリを指定します。単純 NFS マウントの場合は、mount_nfs(1M) で指定されているのと同じオプションが使用できます。*location* の形式は次のとおりです。

host: *pathname*

host は、マウントするファイルシステムが置かれているホスト名です。*pathname* は、マウントするディレクトリの絶対パス名です。

ほかのファイルシステム用のオプションについては、`mount_cachefs`(1M) など、ほかの `mount_*` のマニュアルページを参照してください。

複製されたファイルシステム

複製された NFS ファイルシステムには複数の *location* フィールドを指定できます。その場合、 `automount` とカーネルのそれぞれがその情報を使用して、可用性の向上を計ります。読み取り専用フラグがマップエントリに設定されている場合、`automountd` はカーネルが使用する、いくつかの条件によってソートされている位置リストを使用してマウントします。マウント時に利用できる位置だけがマウントされ、カーネルで利用可能になります。サーバーが応答しなかった場合、カーネルは代替サーバーに切り替えます。`automount` のソート順序は、次のサーバーの選択方法を決定するのに使用されます。読み取り専用フラグが設定されていない場合、`automount` は同じソート順序によって選択された、最善の位置をマウントします。新しいサーバーが選択されるのは、マウント解除が可能だった場合に限られ、再マウントが実行されます。同じローカルサブネット上のサーバーに最も高い優先順位が与えられ、その次にローカルネット上のサーバーに高い優先順位が与えられます。等距離のサーバー間では、重み付け係数 (後述) が使用されていないかぎり、応答時間によって順序が決まります。

NFS Version 2 プロトコルと NFS Version 3 プロトコルの両方を使用するサーバーの位置がリストに含まれている場合、`automount` はリストに指定されたサーバー位置の一部だけを選択し、すべてのエントリが同じプロトコルになるようにします。 NFS Version 3 プロトコルを使用するサーバーが選択されるのは、ローカルサブネット上にある、NFS Version 2 プロトコルを使用するサーバーが無視されない場合に限られます。詳細は 『『Solaris のシステム管理 (IP サービス)』』 を参照してください。

リストの各 *location* が同じ *pathname* を共有している場合、コンマで区切ったホスト名に対して、1 つの *location* を使用できます。

```
hostname,hostname . . . : pathname
```

括弧で囲んだ整数を重み付け係数としてサーバー名のうしろに指定することにより、サーバー要求に重み付けを設定できます。重み付けが設定されていないサーバーは、ゼロの値 (選択される可能性が最も高い) が与えられているものとみなされます。値が大きくなるにしたがって、選択される可能性が小さくなります。次に例を示します。

```
man -ro alpha,bravo,charlie(1),delta(4) : /usr/man
```

ホスト `alpha` および `bravo` に最も高い優先順位が与えられており、ホスト `delta` には最も低い優先順位が与えられています。

選択プロセスではサーバーの近接度が優先されます。上記の例で、サーバー `delta` はクライアントと同じネットワークセグメントにあり、ほかのサーバーは異なる

ネットワークセグメントにある場合、重み付けの値は無視され、delta が選択されます。重み付けが有効なのは、同じネットワーク近接度を持つサーバー間で選択が行われる場合だけです。自動マウンターは重み付けに関わらず、同じネットワークセグメント上のほかのサーバー経由でローカルホストをつねに選択します。

サーバーごとにエクスポートポイントが異なる場合にも、重み付けを適用できます。たとえば、次のように指定します。

```
man -ro alpha:/usr/man  bravo,charlie(1):/usr/share/man
     delta(3):/export/man
```

マッピングを複数の入力行にわたって記述する場合は、バックスラッシュ (\) で復帰改行をエスケープします。コメントは番号記号 (#) で始まり、その後の復帰改行で終了します。

マップキーの置換

アンパサンド (&) 記号は、使用されているエントリの key フィールドの値に展開されます。次のコマンドを見てください。

```
jane sparcserver : /home/&
```

この場合、& は jane に展開されます。

ワイルドカードキー

アスタリスク (*) 記号は、key フィールドとして与えられた場合、どのようなキーにも当てはまるエントリとして認識されます。このようなエントリは、それまで一致しなかったあらゆるキーと一致します。たとえば、/configの間接マップに次のエントリが指定されている場合、

```
*         & : /export/config/&
```

位置を次のように指定できるリモートファイルシステムは、/config にすべて自動マウントされます。

```
hostname : /export/config/hostname
```

変数の置換

automount マップ内でクライアント固有の変数を使用できます。たとえば、マップに $HOST が指定されている場合、automount は $HOST をクライアントのホスト名に対応する現在値に展開します。次の変数を使用できます。

ARCH	uname -m の出力に基づくアプリケーションアーキテクチャー	アーキテクチャー名。たとえば、 sun4u マシンでは sun4。
CPU	uname -p の出力	プロセッサタイプ たとえば、sparc
HOST	uname -n の出力	ホスト名 たとえば、biggles

OSNAME	uname -s の出力	OS 名 たとえば、SunOS
OSREL	uname -r の出力	OS リリース名 たとえば、5.3
OSVERS	uname -v の出力	OS バージョン たとえば、beta1.0
NATISA	isainfo -n の出力	システム固有の 命令セットアーキテクチャー たとえば、sparcv9

参照が接辞付き文字から保護される必要がある場合は、変数名を中括弧 ({ }) で囲みます。

多重マウント

多重マウントエントリの形式は、次のとおりです。

key [*-mount-options*] [[*mountpoint*] [*-mount-options*] *location* . . .] . . .

最初の /[*mountpoint*] は、最初のマウントでは省略可能ですが、以降のすべてのマウントでは必須です。省略可能な *mountpoint* は、key で指定されたディレクトリからの相対パス名とみなされます。最初の *mountpoint* を省略した場合、/ (ルート) の *mountpoint* が暗黙に使用されます。

/src の直接マップに次のエントリがあるとします。

```
beta     -ro\
  /           svr1,svr2:/export/src/beta  \
  /1.0        svr1,svr2:/export/src/beta/1.0  \
  /1.0/man    svr1,svr2:/export/src/beta/1.0/man
```

すべてのオフセットは、beta の下のサーバー上になければなりません。automount は svr1 または svr2 のどちらかから、つまり最も近くにあり先に応答した方のホストから、/src/beta、/src/beta/1.0、および /src/beta/1.0/man を必要に応じて自動的にマウントします。

その他のファイルシステムタイプ

オートマウントは、デフォルトのファイルシステムタイプとして NFS マウントを想定しています。ほかのファイルシステムタイプを指定する場合は、fstype マウントオプションを使用します。fstype オプションと組み合わせることにより、そのファイルシステムタイプ固有のほかのマウントオプションを使用できます。位置フィールドにはファイルシステムタイプ固有の情報を指定する必要があります。たとえば、CD ファイルシステムをマウントする場合のように、位置フィールドがスラッシュで始まる場合は、その前にコロンが必要です。

```
cdrom -fstype=hsfs,ro   : /dev/sr0
```

autofs マウントを実行するには、次のように指定します。

```
src   -fstype=autofs    auto_src
```

注: この手順は、Volume Manager を使用しない場合に限って使用してください。

マップのデフォルトとしてマップ全体に適用する場合は、CacheFS によるマウントが最も便利です。マスターマップの次のエントリで、キャッシュファイルシステムのホームディレクトリのマウントを指定します。この場合、キャッシュディレクトリのデフォルト位置は /cache です。

```
/home auto_home       -fstype=cachefs,backfstype=nfs
```

オプションの継承については、「注意事項」を参照してください。

間接マップ

間接マップを使用すると、コマンド行で指定した directory からマウントするサブディレクトリの、マッピングを指定できます。間接マップの各 key は、必要に応じてマウントする 1 つ以上のファイルシステムを参照する単純名です。

直接マップ

直接マップのエントリは、autofs マウントポイントと直接対応づけられます。各 key は autofs マウントポイントのフルパス名です。直接マップ全体が 1 つのディレクトリに対応付けられることはありません。

インクルードマップ

次の形式のエントリを使用すると、マップ内に別のマップの内容を含めることができます。

+*mapname*

mapname がスラッシュで始まる場合、*mapname* はローカルファイルのパス名とみなされます。それ以外の場合、マップの位置は次に示すように、/etc/nsswitch.conf 内のオートマウンタ用のエントリに基づくネームサービススイッチのポリシーによって決まります。

```
automount: files nis
```

ネームサービスが filesの場合、/etc 内のローカルファイルの名前が想定されます。検索対象のキーが取り込まれたマップにない場合、次のエントリで検索が続けられます。

特殊マップ

使用可能な特殊マップには 2 つの種類があります。-hosts および -null です。-hosts マップは /net ディレクトリと組み合わせて使用します。この場合、マップのキーは NFS サーバーのホスト名とみなされます。automountd デーモンは、エクスポートされたファイルシステムのサーバーリストに基づいて、マップエントリを動的に作成します。/net/hermes のディレクトリ参照では、hermes のルートからの相対位置にある、対応するディレクトリが参照されます。

-null マップは、指定されたディレクトリに対してすでに設定されているマップを取り消します。これは /etc/auto_master において、+auto_master によって与えられるエントリから継承されるエントリを取り消す場合に便利です。ただし、取り込まれたマップのエントリの前に -null エントリを挿入しないと、効果はありません。

実行可能マップ

ファイルパーミションの実行ビットが設定されているローカルマップは、オートマウンタによって実行され、引数として検索されるキーを与えられます。実行可能マップは、標準出力にオートマウンタのマップエントリの内容を返すようにします。エントリが特定できなかった場合、出力はありません。直接マップを実行可能にすることはできません。

構成と auto_master マップ

引数を指定しないで automount を起動した場合、automount はマスターマップで、autofs マウントポイントのリストおよび対応するマップを調べます。automount はまだマウントされていない autofs マウントがあればそれらをマウントし、マスターマップまたは直接マップから削除されている autofs マウントがあればそれらをマウント解除します。

マスターマップは auto_master とみなされ、その位置はネームサービススイッチのポリシーによって決まります。通常、ローカルファイル /etc/auto_master がマスターマップとして最初に検索されます。

ブラウズ機能

Solaris 2.6 リリースでは、間接マップをブラウズできます。これにより、マウントされているかどうかに関係なく、マウント可能なマウントポイントをすべて表示できます。任意の間接 autofs マップに -nobrowse オプションを追加すると、ブラウズ機能を無効にできます。たとえば、次のように指定します。

```
/net     -hosts      -nosuid,nobrowse
/home    auto_home
```

この場合、/net で表示されるのはマウントされている *hostnames* だけですが、/homeではマウント可能なすべてのマウントポイントが表示されます。autofs ファイルシステムのブラウズ機能は、-browse オプションによって有効になります。これはすべての間接マップにおけるデフォルトの設定です。

マウント制限マップ

あるマップに指定したオプションは、そのマップ内のすべてのエントリに対するデフォルトのオプションとして使用されます。そして、このようなオプションは、マップ内のエントリが独自のマウントオプションを指定するときに初めて無視されます。

しかし、マウントマップとそのサブマウント全体に、nosuid、nodevices、nosetuid、または noexec を強制的に適用することが望ましい場合もあります。これを行うには、拡張マウントオプション -restrict を使用します。

```
 /home     auto_home       -restrict,nosuid,hard
```

-restrict オプションを使用すると、制限的なオプション (つまり、nosuid、nodevices、nosetuid、および noexec) はすべて、-restrict オプション自身とともに、強制的に継承されます。上記例では、nosuid と restrict オプションは継承されますが、hard オプションは継承されません。また、restrictオプションは「実行可能なマップ」の実行を防ぎます。さらに、restrict オプションは、あるプログラムが自分のゾーン内で利用できるすべての特権を持たずに、一部の特権だけで確立した自動マウントに対しても強制されます。

終了ステータス　次の終了値が返されます。

0　正常終了。

1　エラーが発生しました。

ファイル

/etc/auto_master	マスターオートマウントマップ
/etc/auto_home	オートマウントされるホームディレクトリ用のマップ
/etc/default/autofs	automount と automountd のパラメータにデフォルトの値を提供します。autofs(4) を参照してください。
/etc/nsswitch.conf	ネームサービススイッチの構成ファイル。nsswitch.conf(4) を参照してください。

属性　次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目　isainfo(1), ls(1), svcs(1), uname(1), automountd(1M), mount(1M), mount_cachefs(1M), mount_nfs(1M), svcadm(1M), autofs(4), attributes(5), nfssec(5), smf(5)

『Solaris のシステム管理 (IP サービス)』

注意事項　autofs マウントポイント間に階層関係が存在してはなりません。automount では、autofs マウント内に別の autofs マウントポイントを作成できません。

直接マップエントリごとに新しい autofs マウントが作成されるので、直接マップはできるだけ短くしてください。

直接マップと間接マップのエントリはどちらも、いつでも変更できます。新しい情報は、automountd が次回そのマップエントリを使用してマウントを実行するときに使用されます。

マスターマップまたは直接マップに追加された新しいエントリは、automount コマンドを実行してそれらの新しいエントリを新しい autofs マウントポイントとしてインストールするまで使用できません。間接マップに追加された新しいエントリは、ただちに使用できます。

Solari 2.6 リリースでは、間接マップに対応づけられた autofs ディレクトリの表示 (ls(1) のマニュアルページを参照) には、潜在的にマウント可能なエントリがすべて含まれます。潜在的にマウント可能なエントリに対応づけられる属性は一時的なものです。実際のファイルシステム属性が表示されるのは、ファイルシステムのマウント後に限られます。

マスターマップの 3 番目のオプションフィールドを指定すると、デフォルトのマウントオプションをマップ全体に割り当てることができます。これらのオプションが適用されるのは、マウントオプションが指定されていないマップエントリだけです。マップエントリにオプションが指定されていると、デフォルトのオプションは無効になります。現時点では、オプションの連結は行われません。連結機能は、将来のリリースに備えて計画中です。

オートマウンタのデフォルトの NFS マウント操作の再試行回数は 0 です。つまり、マウント試行は 1 回だけで、再試行されません。これは、`mount_nfs`(1M) ユーティリティーのデフォルト (10000 回) と大きく異なる点なので、注意してください。

ネットワーク情報サービス (NIS) は従来、Sun イエローページ (YP) と呼ばれていました。これらの機能は同等です。

`automount` サービスはサービス管理機能 `smf`(5) により次のサービス識別子の下で管理されます。

```
svc:/system/filesystem/autofs:default
```

有効化、無効化、または再起動要求など、このサービスに関する管理操作は、svcadm(1M) を使用して実行できます。サービスの状態は `svcs`(1) コマンドを使用して照会できます。

名前

boot – システムカーネルまたはスタンドアロンプログラムの起動

形式

SPARC

```
boot [OBP names] [file] [-aV] [-D default-file]
     [boot-flags] [—] [client-program-args]
```

x86

```
kernel multiboot [file] [boot-args] [-B prop=val [,val]...]

i
```

機能説明

ブートストラップとは、スタンドアロンプログラムを読み込んで実行する処理のことです。ここでのブートストラップとは、起動可能なオペレーティングシステムを読み込んで実行することを意味します。通常、スタンドアロンプログラムはオペレーティングシステムカーネル (kernel(1M) のマニュアルページを参照) ですが、代わりに任意のスタンドアロンプログラムを起動することもできます。SPARC ベースのシステムでは、オペレーティングシステム以外に起動できるスタンドアロンプログラムの代表的な例として、マシンの診断モニターがあります。

スタンドアロンプログラムが動的にリンクされる実行可能プログラムとして識別された場合、boot は実行可能形式によって指定されたインタプリタ (リンカー / ローダー) を読み込み、そのインタプリタに制御を渡します。スタンドアロンプログラムが静的にリンクされている場合は、そのスタンドアロンプログラムに直接ジャンプします。

カーネルは読み込まれると、UNIX システムを起動して、必要なファイルシステム (vfstab(4) のマニュアルページを参照) をマウントし、/sbin/init を実行して、システムを /etc/inittab で指定されている initdefault 状態にします。inittab(4) を参照してください。

SPARC でのブートストラップ手続き

SPARC ベースのシステムでは、ほとんどのマシンでのブートストラップ手続きは次の基本フェーズからなります。

マシンの電源を投入すると、(PROM 内の) システムファームウェアが電源投入時自己診断テスト (POST) を実行します。この診断テストの形式と範囲は、システムに搭載されているファームウェアのバージョンによって異なります。

診断テストが正常に完了した後、ファームウェアが使用する不揮発性記憶領域に適切なフラグが設定されていれば、ファームウェアは自動起動を試みます。ファームウェアは、読み込むファイルの名前とそのファイルを読み込むデバイスを選択することができます。

これらのフラグと名前は、シェルから eeprom(1M) コマンドを使用するか、またはシステムの停止後に ok プロンプトで PROM コマンドを使用することにより、設定できます。

第 2 レベルのプログラムは、ufsboot (ディスクから起動する場合) または inetboot または wanboot (ネットワークから起動する場合) のどれかです。

ネットワーク起動

ネットワーク起動は 2 段階で行われます。まず、クライアントが、二次起動プログラムの読み込みに必要な IP アドレスとその他のすべてのパラメータを取得します。続いて、二次起動プログラムが、UNIX カーネルを読み込みます。

IP アドレスは、PROM で使用可能な機能と PROM の構成に応じて、RARP、DHCP、手動構成のいずれかの方法を使って取得できます。sun4u カーネルアーキテクチャのマシンには、DHCP 対応の PROM が搭載されています。

次に、2 種類のネットワーク起動方式を指定する boot コマンド構文を示します。

```
boot net:rarp
boot net:dhcp
```

次のコマンドを見てください。

```
boot net
```

このコマンドのように、rarp または dhcp 指示子を指定しないと、net が別名になっているネットワークインタフェースを介して、デフォルトのネットワーク起動方式が呼び出されます。

以降では、RARP/bootparams を使用するネットワーク起動のイベントシーケンスについて説明します。そのあとで、DHCP を用いたシーケンスについて説明します。

RARP/bootparams を使ってネットワークから起動する場合、PROM はまず、応答を受信するまで逆 ARP 要求をブロードキャストします。応答を受信すると、TFTP 要求をブロードキャストして inetboot の最初のブロックを取得します。続いて、その最初のブロック要求に一番先に応答したサーバーに後続の要求が送信されます。読み込みの完了後、inetboot も同様に、逆 ARP を使って IP アドレスを取得した後、bootparams RPC 呼び出し (bootparams(4) を参照) をブロードキャストすることで、構成情報とルートファイルシステムを検索します。続いて、inetboot は NFS 経由でカーネルを読み込んだ後、そのカーネルに制御を移します。

DHCP を使用してネットワークから起動する場合、PROM はハードウェアアドレスおよびカーネルアーキテクチャをブロードキャストし、IP アドレス、起動パラメータ、およびネットワーク構成情報を要求します。(可能性のある複数のサーバーの中から) DHCP サーバーが応答し選択されると、そのサーバーがクライアントに IP アドレスおよびクライアントの起動に必要な他のすべての情報を送信します。この情報を受信したクライアントの PROM は、読み込むファイルの名前を調べ、そのファイル名が HTTP URL であるかどうかにより、2 種類の動作のいずれかを実行します。ファイル名が HTTP URL でなかった場合、PROM は、inetboot をダウンロードし、それをメモリー内に読み込んで実行します。すると、inetboot はカーネルを呼び出し、さらにそのカーネルが必要なファイルを読み込んでから inetboot を解放します。その後、起動スクリプトが DHCP エージェント (dhcpagent(1M) を参照) を起動します。以降の DHCP アクティビティは、その DHCP エージェントによって実行されます。

読み込むファイルが HTTP URL の場合、PROM は、その参照ファイルを HTTP を使って読み込みます。クライアントが HMAC SHA-1 鍵を使って構成されていた場合、クライアントは読み込んだファイルの完全性を検査した後で、そのファイルを実行します。そのファイルは wanboot バイナリであるとみなされます。起動された wanboot は、処理を継続できるだけの情報が揃っているかどうかを判断します。必要な情報が不足していた場合は、適切なエラーを出力して処理を終了するか、コマンドインタプリタを起動して必要な構成情報の入力をユーザーに求めます。必要な情報を取得した wanboot は、HTTP 経由で起動ファイルシステムをメモリー内に読み込みます。クライアント上に暗号鍵がインストール済みである場合、wanboot は、起動ファイルシステムイメージと関連ハッシュを復号化した後、そのハッシュを検証します (暗号鍵は存在するがハッシュ鍵が存在しない、という状況はエラーになります)。起動ファイルシステムには、wanboot が正しい時刻の設定とルートファイルシステムの取得を実行するのに必要な各種構成データが含まれています。

wanboot は、起動ファイルシステムを調べることで、HTTP、HTTPS のいずれを使用すべきかを判断します。HTTP を使用すべきであることが判明した場合でかつ、クライアントが HMAC SHA-1 鍵を使って構成されていた場合、wanboot はルートファイルシステムの完全性検査を実行します。ルートファイルシステムのメモリー内への読み込み (および必要に応じて完全性検査の実行) が完了すると、wanboot はそこから UNIX を読み込みおよび実行します。なお、wanboot.conf(4) ファイルで boot_logger URL が指定されていた場合、wanboot は処理状況に関するログを定期的に記録します。

すべての PROM が URL を処理できるわけではありません。クライアントにその機能が備わっているかどうかを判断するには、list-security-keys OBP コマンド (monitor(1M) を参照) を使用します。

現在のところ、x86 プラットフォーム上では WANブートは利用できません。

wanboot コマンド行

クライアントプログラムが wanboot である場合、クライアントプログラムは、次の形式の client-program-args を受け付けます。

boot ... -o *opt1*[,*opt2*[,...]]

ここで、各オプションに指定できるアクションは、次のとおりです。

dhcp
: DHCP 経由で構成パラメータを取得するように wanboot に指示します。

prompt
: コマンドインタプリタを起動するように wanboot に指示します。

<*cmd*>
: 下記のインタプリタコマンドのいずれか。

...または、下記のインタプリタパラメータ名を使用した代入文。

wanboot コマンドインタプリタ

起動時に client-program-args として「-o prompt」を指定すると、wanboot コマンドインタプリタが起動されます。単一のコマンドまたは代入文、もしくは、コンマで区切られた一連のコマンドまたは代入文が入力可能です。利用可能な構成パラメータは、次のとおりです。

host-ip
クライアントの IP アドレス (ドット区切り 10 進表記)。

router-ip
デフォルトのルーターの IP アドレス (ドット区切り 10 進表記)。

subnet-mask
サブネットマスク (ドット区切り 10 進表記)。

client-id
DHCP クライアント識別子 (引用符付き ASCII 文字列または 16 進 ASCII)。

hostname
DHCP トランザクションで要求するホスト名 (ASCII)。

http-proxy
HTTP プロキシサーバー指定 (IPADDR[:PORT])。

利用可能な鍵名は、次のとおりです。

3des
トリプル DES 暗号鍵 (48 個の 16 進 ASCII 文字)。

aes
AES 暗号鍵 (32 個の 16 進 ASCII 文字)。

sha1
HMAC SHA-1 署名鍵 (40 個の 16 進 ASCII 文字)。

最後に、WANブート CGI の URL を参照する方法を示します。

bootserver
WANブート CGI の URL (OBP の file パラメータと同等)。

インタプリタが受け付けるコマンドは、次のとおりです。

help
利用可能なコマンドの簡単な説明文を出力します。

var=*val*
val を *var* に代入します。ここで、*var* は、構成パラメータ名、鍵名、bootserver のいずれかです。

var=
パラメータ *var* の設定を解除します。

`list`
すべてのパラメータとその設定値を一覧表示します (OBP 経由で取得された鍵の値は表示されません)。

`prompt`
未設定パラメータに対する値の入力を、ユーザーに求めます。各パラメータの名前と現在値 (もしあれば) が出力されます。ユーザーは、Return キーを押して現在値を受け入れることもできますし、新しい値を入力することもできます。

`go`
すべての値の入力を完了したら、インタプリタを終了し起動処理を継続します。

`exit`
起動インタプリタを終了し、OBP の `ok` プロンプトに戻ります。

上記の代入文やコマンドは、コマンド行の `-o` オプションの一部として、いくつでも指定できますが、`boot` コマンドの引数の長さには 128 バイトの OBP 制限があります。たとえば、「`-o list,go`」とすると、パラメータの現在値 (デフォルト値) が一覧表示された後、起動処理が継続します。

ディスクからの起動

ディスク (またはディスク同様のデバイス) から起動する場合、ブートストラップ処理は、一次起動および二次起動という概念的に異なる 2 種類のフェーズからなります。一次起動フェーズでは、PROM が、起動デバイスとして選択されたディスクパーティションのブロック 1 ～ 15 から主起動ブロックを読み込みます。

スタンドアロンプログラムのパス名が相対パス名の場合 (スラッシュで始まらない場合)、第 2 レベルの起動において、プラットフォームに依存する検索パスでスタンドアロンプログラムが検索されます。このパスには必ず、`/platform/`*platform-name* が含まれています。多くの SPARC プラットフォームでは、次にプラットフォーム固有のパスエントリ `/platform/`*hardware-class-name* が検索されます。`filesystem`(5) のマニュアルページを参照してください。絶対パス名の場合、`boot` は指定されたパスを使用します。その後、`boot` プログラムは該当するアドレスからスタンドアロンプログラムを読み込み、制御を渡します。

ファイル名がコマンド行で、または `boot-file` NVRAM 変数などで指定されていない場合、`boot` はシステムにインストールされているソフトウェア、ハードウェアおよびファームウェアの能力に基づいて、読み込むべきデフォルトのファイルを選択します。

カーネルへのパスには、空白を含めてはいけません。

OpenBoot PROM `boot` コマンドの動作

OpenBoot `boot` コマンドは、次の形式の引数をとります。

```
ok boot [device-specifier] [arguments]
```

デフォルトの `boot` コマンドに引数はありません。

```
ok boot
```

boot コマンド行で *device-specifier* を指定しない場合、OpenBoot は通常、*boot-device* 変数または *diag-device* NVRAM 変数を使用します。オプションの *arguments* をコマンド行で指定しないと、OpenBoot は通常、*boot-file* または *diag-file* NVRAM 変数をデフォルトの boot 引数として使用します。(システムが診断モードの場合、*boot-device* および *boot-file* の代わりに、*diag-device* と *diag-file* が使用されます)。

arguments には複数の文字列を指定できます。すべての *argument* 文字列は二次起動プログラムに渡され、OpenBoot では解釈されません。

boot コマンド行に *arguments* を指定した場合、*boot-file* または *diag-file* NVRAM 変数のどちらも使用されません。NVRAM 変数の内容がコマンド行の引数とマージされることもありません。たとえば、次のコマンドを見てください。

```
ok boot -s
```

このコマンドでは、*boot-file* および *diag-file* の設定値は無視され、文字列 -s は *arguments* として解釈されます。この場合、boot は *boot-file* または *diag-file* の内容を使用しません。

以前の PROM におけるコマンド例を示します。

```
ok boot net
```

このコマンドでは引数が指定されていないため、代わりに、*boot-file* または *diag-file* の設定値 (設定されている場合) が、boot に渡すデフォルトのファイル名および引数として使用されます。ほとんどの場合、システムタイプ、システムのハードウェアとファームウェア、およびルートファイルシステムにインストールされているソフトウェアに基づいて boot コマンドに適切なデフォルト値を選択させるのが、最善の方法です。*boot-file* または *diag-file* を変更すると、状況によっては予期せぬ結果を招くおそれがあります。

これは、大部分の OpenBoot 2.x および 3.x ベースのシステムにおける一般的な動作です。ただし、プラットフォームによっては動作が異なる可能性もあります。

次のコマンドを見てください。

ok **boot cdrom**

...このコマンドも通常は、引数を指定しません。したがって、*boot-file* に 64 ビットカーネルのファイル名が設定されている環境で、「boot cdrom」と入力してインストール CD または DVD を起動しようとしても、インストール媒体に 32 ビットカーネルしか含まれていなければ、その起動処理は失敗します。

使用される boot コマンドの形式によっては、*boot-file* または *diag-file* の内容が無視されるので、プロダクションシステムで *boot-file* に依存することは一般に推奨しません。

ローカル (CD または DVD 上) に存在する wanboot コピーから WANブートを実行するには、次のコマンドを使用する必要があります。

ok **boot cdrom -F wanboot - install**

最近の PROM に含まれるネットワーク起動サポートパッケージは、次の構文をサポートするように強化されており、引数を処理できるようになっています。

[*protocol*,] [*key=value*,]*

すべての引数は省略可能であり、指定する順序にも制限はありません。ただし、リストの末尾でない限り、引数の後にはコンマが必要です。ここで指定した引数は、すべてのデフォルト値よりも優先されるほか、DHCP を使って起動する場合には、それらの引数に対応する DHCP サーバーから提供される構成情報よりも優先されます。

上記の *protocol* には、使用するアドレス検索プロトコルを指定します。

key=value 属性ペアには、以下の構成パラメータを指定します。

`tftp-server`
TFTP サーバーの IP アドレス。

`file`
TFTP を使ってダウンロードするファイルの名前、または WANブートの URL。

`host-ip`
クライアントの IP アドレス (ドット区切り 10 進表記)。

`router-ip`
デフォルトのルーターの IP アドレス。

`subnet-mask`
サブネットマスク (ドット区切り 10 進表記)。

`client-id`
DHCP クライアント識別子。

`hostname`
DHCP トランザクションで使用するホスト名。

`http-proxy`
HTTP プロキシサーバー指定 (IPADDR[:PORT])。

`tftp-retries`
TFTP の最大リトライ回数。

`dhcp-retries`
DHCP の最大リトライ回数。

ネットワークブートサポートパッケージによって処理される一連の引数は、次のいずれかの方法で指定します。

- パッケージの `open` メソッドに渡される引数として

- NVRAM 変数 network-boot-arguments 内に指定する引数として

network-boot-arguments 内に指定された引数は、パッケージの open メソッドに引数が 1 つも渡されなかった場合にのみ処理されます。

引数の値

protocol には、使用するアドレス検索プロトコルを指定します。rarp、dhcp のいずれかを指定できます。

このドキュメントで定められた新しい構文とスタイルを使って他の構成パラメータが指定されていた場合、*protocol* パラメータが指定されていなければ、手動の構成を意味します。

他の構成パラメータが 1 つも指定されていない場合や、他の構成パラメータが現在サポートされている位置に基づくパラメータ構文を使って指定されていた場合、*protocol* パラメータを指定しなければ、ネットワークブートサポートパッケージは、プラットフォーム固有のデフォルトのアドレス検索プロトコルを使用することになります。

手動構成を行う場合、IP アドレス、ブートファイル名、ブートファイルイメージの提供元サーバーのアドレスの各情報を、クライアントに提供する必要があります。また、ネットワークの構成によっては、subnet-mask と router-ip も指定する必要があります。

protocol 引数が指定されなかった場合、ネットワークブートサポートパッケージは、プラットフォーム固有のデフォルトのアドレス検索プロトコルを使用します。

tftp-server は、TFTP を使用する場合、ダウンロードするファイルを提供する TFTP サーバーの IP アドレス (標準 IPv4 のドット区切り 10 進表記) です。

DHCP を使用する場合、その値は、DHCP 応答に指定された TFTP サーバーの値よりも優先されます。

サーバーが引数として指定された場合と DHCP 応答内に指定された場合、TFTP RRQ はサーバーにユニキャストされます。それ以外の場合、TFTP RRQ はブロードキャストされます。

file には、TFTP を使用する場合は TFTP サーバーから読み込むファイルを指定し、HTTP を使用する場合は URL を指定します。ファイル名が URL である場合、すなわち、ファイル名が http: で始まる場合 (大文字小文字の区別あり)、HTTP が使用されます。

RARP と TFTP を使用する場合、このドキュメントで前述したように、デフォルトのファイル名はクライアントの IP アドレスの ASCII 16 進表記になります。

DHCP を使用する場合、この引数は、DHCP 応答に指定されたブートファイル名よりも優先されます。

DHCP と TFTP を使用する場合、デフォルトのファイル名は、root ノードの `name` プロパティーから自動生成されます。その際、コンマ (,) はピリオド (.) に置き換えられます。

ファイル名をコマンド行から指定する場合、ファイル名にスラッシュ (/) を含めることはできません。

URL の形式については、RFC 2396 で説明されています。HTTP サーバーは、IP アドレス (標準 IPv4 のドット区切り 10 進表記) として指定する必要があります。ポート番号 (省略可能) は、10 進数として指定します。ポートを指定しなかった場合、ポート 80 (10 進) が使用されます。

指定された URL は「安全にエンコードされている」必要があります。というのも、パッケージは、指定された URL にエスケープエンコーディングを適用しないからです。コンマを含む URL は、引用符付きの文字列として指定する必要があります。それ以外の URL は、必ずしも引用符で囲む必要はありません。

`host-ip` には、クライアントすなわちブート対象システムの、IP アドレス (標準 IPv4 のドット区切り 10 進表記) を指定します。アドレス検索プロトコルとして RARP を使用している場合にこの引数を指定すると、RARP を使用する必要がなくなります。

DHCP を使用している場合に `host-ip` 引数を指定すると、クライアントは、RFC 2131 の「Externally Configured Network Address」で規定されている手順に従うようになります。

`router-ip` には、直接接続されたネットワーク上にあるルーターの IP アドレス (標準 IPv4 のドット区切り 10 進表記) を指定します。このルーターは、ネットワーク通信の最初の接続先として使用されます。この引数を指定した場合、そこで指定されたルーターは、DHCP 応答に指定された推奨ルーターよりも優先されます。

`subnet-mask` (標準 IPv4 のドット区切り 10 進表記で指定) は、クライアントが存在するネットワークのサブネットマスクです。サブネットマスクが (この引数または DHCP 応答を通じて) 指定されなかった場合、ブート対象クライアントに割り当てられたアドレスに対するネットワーククラス (Class A、B、C のいずれか) に適するデフォルトマスクが使用されます。

`client-id` には、クライアントの一意に決まる識別子を指定します。DHCP クライアント識別子は、この値に基づいて生成されます。クライアント識別子の指定方法には、次の 2 つがあります。

- 識別子の ASCII 16 進表現
- 引用符付きの文字列

したがって、「`client-id="openboot"`」と「`client-id=6f70656e626f6f74`」はどちらも、DHCP クライアント識別子 6F70656E626F6F74 を表わします。

コマンド行から指定される識別子には、スラッシュ (/) や空白を含めることはできません。

DHCP クライアント識別子の最大長は、32 バイト (ASCII 16 進形式を使用する場合は 32 バイトを表す 64 文字) です。後者の形式を使用する場合、識別子を構成する文字数は偶数でなければなりません。有効な文字は、0 〜 9、a 〜 f、A 〜 F です。

クライアントが正しく認識されるためには、クライアントが接続されているサブネット上で、クライアント識別子が一意に決まる必要があります。この要件を満足する識別子を選択するのは、システム管理者の役割です。

コマンド行で指定されたクライアント識別子は、任意の DHCP メカニズムを使って指定された識別子よりも優先されます。

`hostname` (文字列として指定) には、DHCP トランザクション内で使用されるホスト名を指定します。この名前は、ローカルドメイン名で修飾してもしなくてもかまいません。ホスト名の最大長は、255 文字です。

注 – `hostname` パラメータは、クライアントが、希望するホスト名を DHCP サーバーに提供することが必要なサービス環境で使用できます。クライアントが、希望するホスト名を DHCP サーバーに提供すると、DHCP サーバーは、そのホスト名とクライアントに割り当てられた IP アドレスを、DNS を使って登録します。

`http-proxy` は、次のようなホストの標準記法を使って指定します。

host [":"" *port*]

...ここで、*host* には IP アドレス (標準 IPv4 のドット区切り 10 進表記) を指定し、*port* (省略可能) には 10 進数を指定します。*port* を指定しなかった場合、ポート 8080 (10 進) が使用されます。

`tftp-retries` は、TFTP 処理が失敗したとみなされるまでの、最大リトライ回数 (10 進で指定) です。デフォルトのリトライ回数は、無限回です。

`dhcp-retries`は、DHCP 処理が失敗したとみなされるまでの、最大リトライ回数 (10 進で指定) です。デフォルトのリトライ回数は、無限回です。

x86 でのブートストラップ手続き

x86 システム上では、ブートストラップ手続きは、カーネルのロードと、カーネルの初期化という概念的に異なる 2 つのフェーズからなります。カーネルのロードは、システムボード上の BIOS ROM および周辺装置上の BIOS ROM 拡張機能を使用して、GRUB (GRand Unified Bootloader) 上で実装されます。BIOS はフロッピーディスクの物理的な先頭セクター、ハードディスク、DVD あるいは、CD を読み込むことにより、GRUB をロードします。ネットワークアダプター上の ROM でサポートされている場合は、BIOS はネットワークサーバーから `pxegrub` バイナリをダウンロードすることもできます。GRUB は配置されると、メニューにあるコマンドを実行してカーネルプログラムとデータを含んでいる事前に設定されたブートアーカイブをロードします。また、GRUB は multiboot と呼ばれる小さなプログラムもロードします。これは、カーネル側のマルチブート仕様を実装します。

GRUB がブートアーカイブをロードし終わって、`multiboot` プログラムに制御を引き渡したときに、カーネルの初期化が始まります。この時点で、GRUB は非アクティブとなり、ブートデバイスとの入出力はこれ以上発生しません。`multiboot` プ

ログラムは中核カーネルモジュールを集約し、オペレーティングシステムを開始し、ブートアーカイブから必要なモジュールをリンクし、実際のルートデバイス上のルートファイルシステムをマウントします。この時点で、カーネルは、ストレージに対する入出力を再開し、さらなるファイルシステムをマウントします(vfstab(4) を参照)。そして、多様なオペレーティングシステムのサービスを開始します (smf(5) を参照)。

オプション

SPARC

次の SPARC オプションを指定できます。

`-a`
boot プログラムはこのフラグを問い合わせと解釈し、スタンドアロンプログラムの名前を要求します。`-a` フラグはその後、スタンドアロンプログラムに渡されます。

`-D` *default-file*
default-file を明示的に指定します。一部のシステムでは、なにも指定されていない場合、`boot` は動的デフォルトファイルを選択します。このオプションを使用すると、*default-file* を明示的に指定できます。`kmdb`(1) をブートする場合に便利です。これは、`kmdb` がデフォルトで、`boot` プログラムによってエクスポートされたデフォルトファイルを読み込むためです。

`-V`
詳細なデバッグ情報を表示します。

boot-flags
boot プログラムはすべての *boot-flags* を `file` に渡します。これらの引数は、`boot` では解釈されません。デフォルトのスタンドアロンプログラムで使用できるオプションについては `kernel`(1M) および `kmdb`(1) のマニュアルページを参照してください。

client-program-args
`boot` プログラムはすべての *client-program-args* を *file* に渡します。これらの引数は、`boot` では解釈されません。

file
ブートするスタンドアロンプログラムの名前。`boot` コマンド行または *boot-file* NVRAM 変数でファイル名を明示的に指定しないと、`boot` は適切なデフォルトファイル名を選択します。

OBP names
OpenBoot PROM 指定を行います。たとえば、Desktop SPARC ベースのシステム上で `/sbus/esp@0,800000/sd@3,0:a` を指定した場合は、スロット 0 の esp ホストアダプタで、SCSI バス上の lun0 、ターゲット 3 の SCSI ディスク (sd) を意味します。

x86

次の x86 オプションを指定できます。

-B *prop=val...*
ひとつ以上の設定値ペアが multiboot プログラムに渡されます。複数の設定値ペアはコンマで区切られる必要があります。このオプションの使用方法は、次のコマンドと同様です。eeprom *prop=val*設定情報とその有効な値については、eeprom(1M) を参照してください。

boot-args
boot プログラムはすべての *boot-args* を file に渡します。これらの引数は、boot では解釈されません。カーネルで使用できるオプションについては、kernel(1M) および kmdb(1) のマニュアルページを参照してください。

file
ブートするスタンドアロンプログラムの名前。デフォルトでは、ルートパーティションから、/platform/i86pc/kernel/unix (CPU が 64 ビット対応の場合は /platform/i86pc/kernel/amd64/unix) をブートしますが、コマンド行に他のプログラムを指定することもできます。

multiboot
GRUB によってロードされるマルチブートプログラムの名前。デフォルトの名前は、/platform/i86pc/multiboot です。この名前は変更してはいけません。

x86 でのブートシーケンスの詳細

PC 互換マシンの電源を投入すると、BIOS ROM のシステムファームウェアが電源投入時自己診断テスト (POST) を実行し、周辺ボード上の ROM の BIOS 拡張機能を実行し、ソフトウェア割り込み INT 19h のブートストラップをブートします。INT 19h のハンドラは通常、標準の PC 互換起動処理を実行します。その場合、1 番目のフロッピーディスクドライブから先頭物理セクターを読み取ります。読み取れない場合は、1 番目のハードディスクから読み取ります。次にプロセッサは、メモリー内でこのセクターイメージの先頭バイトにジャンプします。

x86 での一次ブート

フロッピーディスクの先頭のセクターにはマスターブートブロックがあります(GRUB ステージ 1)。ステージ 1 は ステージ 2 をロードします。この時点で GRUB は準備が完全に整います。GRUB は、メニューファイル (/boot/grub/menu.lst) を読み込み、実行します。同様のシーケンスが DVD あるいは、CD ブートでも発生します。しかし、マスターブートブロックの位置と内容は El Torito 仕様で規定されています。El Torito 準拠のブートでも、strap.com が読み込まれ、それが次に boot.bin が読み込まれます。

ハードディスクの最初のセクターには、マスターブートブロックがあり、そのブロックにはマスターブートプログラムと FDISK テーブルが含まれています。この名前は、PC プログラムが管理している FDISK に由来しています。マスターブートは、FDISK テーブルの中のアクティブパーティションを探し、その先頭セクター (GRUB ステージ 1) をロードして、メモリー内の先頭バイトにジャンプします。ハードディスクからの標準の PC 互換起動処理はこれで完了します。GRUB ステージ 1 がマスターブートブロック上にインストールされると (installgrub(1M) の -m を参照) 、ステージ 2 が、アクティブパーティションかどうかには関わらず Solaris FDISK パーティションから直接ロードされます。

Solaris ソフトウェア用 x86 FDISK の パーティションは、1 シリンダのブートスライスから始まり、これは、GRUB の ステージ 1 を先頭セクターに含んでいます。そして、第 2 および第 3 のセクターは、標準 Solaris ディスクラベルおよびボリューム構成テーブル (VTOC) を含んでいます。そして、第 50 とその後続セクターは、GRUB の ステージ 2 を含んでいます。第 4 から 第 49 までのセクターには、Solaris の古いバージョンのブートブロックが含まれている可能性があります。これによって、同じ FDISK の中に、複数の Solaris リリースを同時に存在させることが可能になります。Solaris ソフトウェア用 FDISK パーティションがアクティブパーティションの場合、マスターブートプログラム (mboot) は、先頭セクターにあるパーティションブートプログラムをメモリーの中に読み込み、そこへジャンプします。次に GRUB ステージ 2 プログラムをメモリーに読み込み、そこへジャンプします。GRUB メニューが表示されると、ユーザーは異なるパーティション、異なるディスク、あるいは 可能であればネットワークからオペレーティングシステムをブートできるよう選択できます。

Intel の Preboot eXecution Environment (PXE) 規格が、ネットワークブートのためにサポートされています。PXE を使用してネットワークからのブートを行う場合には、システムあるいはネットワークアダプタの BIOS が DHCP を使用してブートサーバ上のネットワークブートストラッププログラム (pxegrub) を見つけ、Trivial File Transfer Protocol (TFTP) を使用してそのプログラムを読み込みます。BIOS が、メモリ上の先頭バイトにジャンプすることで、pxegrub を実行します。pxegrub は、メニューファイルをダウンロードし、ユーザーにその項目を表示します。

x86 カーネルの開始

カーネルの開始処理は、カーネルのロード処理に依存しません。カーネルが開始している間、コンソール入出力は、console プロパティーで指定されたデバイスで行われます。ルートデバイスは bootpath プロパティーで設定され、ルートファイルシステムの種類は、 fstype プロパティーで指定されます。これらのプロパティーは、Solaris インストールおよびアップグレード処理で /boot/solaris/bootenv.rc の中に設定されるべきです。そして、上記のように (eeprom(1M) を参照) -B で上書きされます。これらのプロパティーが存在していなければ、コンソールの入出力は、デフォルトで、画面とキーボードになります。ルートデバイスは、デフォルトで ramdisk、 ファイルシステムはデフォルトで ufs になります。

使用例

SPARC

例 1 対話モードでシングルユーザーとしてデフォルトのカーネルをブートする

対話モードでシングルユーザーとしてデフォルトのカーネルをブートするには、ok プロンプトで次のいずれかを入力します。

```
boot -as
```

```
boot disk3 -as
```

例 2　WANブートに対応したPROMによるネットワーク起動

さまざまなbootコマンド行呼び出し間の微妙な違いを示すために、ここでは network-boot-arguments が設定されており、以下のコマンドのように net に対してデバイス別名が設定されているものとします。

次のコマンドでは、デバイス別名内のデバイス引数が、デバイスドライバによって処理されます。ネットワークブートサポートパッケージは、network-boot-arguments 内の引数を処理します。

```
boot net
```

次のコマンドでは、デバイス引数なし、とみなされます。ネットワークブートサポートパッケージは、network-boot-arguments 内の引数を処理します。

```
boot net:
```

次のコマンドでは、デバイス引数なし、とみなされます。rarp は唯一のネットワークブートサポートパッケージ引数です。network-boot-arguments は無視されます。

```
boot net:rarp
```

次のコマンドでは、指定されたデバイス引数が処理されます。ネットワークブートサポートパッケージは、network-boot-arguments 内の引数を処理します。

```
boot net:speed=100,duplex=full
```

例 3　以前のPROMにおけるwanbootの使用

次のコマンドでは、DVDまたはCDからwanbootバイナリが読み込まれます。読み込まれたwanbootは、DHCPの実行時にコマンドインタプリタを起動し、鍵やその他の必要な構成情報をユーザーが入力できるようにします。

```
boot cdrom -F wanboot -o dhcp,prompt
```

x86　例 4　対話モードでシングルユーザーとしてデフォルトのカーネルをブートする

シングルユーザー状態の対話モードでデフォルトのカーネルをブートするには、コマンドラインでGRUBを次のように編集します。

```
kernel /platform/i86pc/multiboot kernel/unix -as
```

x86 (64 ビットのみ)　例 5　対話モードでシングルユーザーとしてデフォルトのカーネルをブートする

シングルユーザー状態の対話モードでデフォルトのカーネルをブートするには、コマンドラインでGRUBを次のように編集します。

```
kernel /platform/i86pc/multiboot kernel/amd64/unix -as
```

例 6　64 ビット x86 プラットフォーム上での 32 ビットおよび 64 ビットカーネル間の切り替え

`boot-file` `eeprom`(1M) 変数のデフォルト値は NULL 文字列です。これにより、使用するシステムのハードウェアに適切なカーネル (32 ビットまたは 64 ビット) を二次ブートプログラムで選択できます。どちらかのカーネルを選択するには、次の手順に従います。

32 ビットカーネルを指定するには、root 権限または同等の権限で次のように入力します。

```
# eeprom boot-file kernel/unix
```

次にリブートすると、システムは 32 ビットカーネルで実行されます。

あるいは、32 ビットカーネルを GRUB のメニューで次のように指定することができます。

```
kernel /platform/i86pc/multiboot kernel/unix
```

64 ビットカーネルを指定するには、root 権限または同等の権限で次のように入力します。

```
# eeprom boot-file kernel/amd64/unix
```

次にリブートすると、システムは 64 ビットカーネルで実行されます。

あるいは、64 ビットカーネルを GRUB のメニューで次のように指定することができます。

```
kernel /platform/i86pc/multiboot kernel/amd64/unix
```

二次ブートプログラムで使用するシステムのハードウェアに適切なカーネルを選択できるように、`boot-file` 変数をデフォルト値である NULL 文字列に戻すには次のように入力します。

```
# eeprom boot-file ""
```

非特権ユーザーとして、`boot-file` 変数の現在の値を確認するには次のように入力します。

```
% eeprom boot-file
```

そのコマンドの詳細については、`eeprom`(1M) のマニュアルページを参照してください。

ファイル

`/platform/`*platform-name*`/ufsboot`
　ディスク、DVD あるいは CD からブートするための 第 2 レベルのプログラム

`/etc/inittab`
　`initdefault` 状態が指定されているテーブル

`/sbin/init`
　`initdefault` 状態にシステムを移行するプログラム

64 ビット SPARC のみ

`/platform/`*platform-name*`/kernel/sparcv9/unix`
システムをブートするデフォルトのプログラム。

x86 のみ

`/boot`
ブートに関連するファイルが置かれているディレクトリ。

`/platform/i86pc/multiboot`
カーネル側のマルチブート仕様を実装するカーネルプログラム

`/platform/i86pc/kernel/unix`
システムをブートするデフォルトのプログラム。

64 ビット x86 のみ

`/platform/i86pc/kernel/amd64/unix`
システムをブートするデフォルトのプログラム。

関連項目

`kmdb`(1), `uname`(1), `eeprom`(1M), init(1M), `installboot`(1M), `kernel`(1M), `monitor`(1M), shutdown(1M), `uadmin`(2), `bootparams`(4), `inittab`(4), `vfstab`(4), `wanboot.conf`(4), `filesystem`(5)

RFC 903, A Reverse Address Resolution Protocol, `http://www.ietf.org/rfc/rfc903.txt`

RFC 2131, Dynamic Host Configuration Protocol, `http://www.ietf.org/rfc/rfc2131.txt`

RFC 2132, DHCP Options and BOOTP Vendor Extensions,`http://www.ietf.org/rfc/rfc2132.txt`

RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax, `http://www.ietf.org/rfc/rfc2396.txt`

『Solaris のシステム管理 (基本編)』

『Sun ハードウェアマニュアル』

『OpenBoot Command Reference Manual』

警告

`boot` ユーティリティは、ブート可能プログラムとして使用できるファイルかどうか判別できません。ブート不可能なファイルを起動するように要求された場合、`boot` ユーティリティはそのファイルを読み込み、そのファイルに制御を渡します。その場合の結果は予測できません。

注意事項

platform-name を調べるには、`uname`(1) の `-i` オプションを使用します。
hardware-class-name を調べるには、`uname`(1) の `-m` オプションを使用します。

現在のリリースの Solaris オペレーティングシステムは、UltraSPARC-I CPU を搭載するマシンをサポートしません。

名前

bootadm – GRUB 対応オペレーティングシステムのブート管理

形式

```
/sbin/bootadm  update-archive [-vn] [-R altroot]

/sbin/bootadm  list-archive [-vn] [-R altroot]

 x86 のみ

/sbin/bootadm  set-menu [-R altroot] key=value

/sbin/bootadm  list-menu [-R altroot]
```

機能説明

bootadm コマンドは、ブートアーカイブと GRUB (GRand Unified Bootloader) メニューを管理します。update–archive オプションを使えば、ブートアーカイブの更新を、障害予防策として、あるいは復旧手順の一環として、実施することができます。set-menu サブコマンドを使えば、GRUB メニュー内の自動起動タイムアウトやデフォルトブートエントリを切り替えることができます。

list-menu サブコマンドは、GRUB メニューの場所と現在のエントリ内容を表示します。GRUB メニューの場所は通常は /boot/grub/menu.lst ですが、使用されたインストール方法によっては、アクティブな GRUB メニューがほかの場所に格納されている可能性もあります。list-menu サブコマンドを使ってアクティブな GRUB メニューの場所を特定してください。たとえば、Live Upgrade を使ってインストールされたシステムの場合、現在のブート環境内に GRUB メニューが格納されていない可能性があります。list-menu オプションの典型的な出力については、「使用例」セクションを参照してください。

SPARC システムにはブートアーカイブという概念がありません。SPARC システム上の bootadm コマンドの目的は、x86 ディスクレスクライアントを管理することです。

サブコマンド

bootadm コマンドのサブコマンドを、次に示します。

update-archive
: 現在のブートアーカイブを必要に応じて更新します。SPARC と x86 の両方のプラットフォームに適用されます。

list-archive
: ブートアーカイブに含めるファイルとディレクトリを一覧表示します。SPARC と x86 の両方のプラットフォームに適用されます。

set-menu
: GRUB メニューを維持管理します。現在の GRUB メニューは、boot/grub/menu.lst にあります。ただし、これはルートからの相対パスです。これは変更される可能性があるため、この場所には依存しないでください。x86 プラットフォームにのみ適用されます。

list-menu
: アクティブな GRUB メニューの場所と現在の GRUB メニューエントリを一覧表示します。これには、自動起動タイムアウト、デフォルトエントリ番号、および各エントリのタイトルが含まれます。x86 プラットフォームにのみ適用されます。

オプション

bootadm コマンドには、次のオプションを指定できます。

-v
update-archive 処理の場合、無効なファイルが標準エラー出力に表示されます。

-n
update-archive 処理の場合、アーカイブ内容のチェックは行われますが、その更新は行われません。

-R *altroot*
別のルートパスに処理が適用されます。

注 – いかなる非大域ゾーンのルートファイルシステムも -R で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。zones(5) のマニュアルページを参照してください。

key=value
可能な値は次のとおりです。

default=*entrynum*
GRUB メニュー内の項目番号 (0、1、2 など)。タイムアウト時にブートするオペレーティングシステムを指定します。

timeout=*seconds*
デフォルト項目番号で指定されたオペレーティングシステムがブートされるまでの秒数。値が –1 の場合、自動起動は無効になります。

使用例

例 1 現在のブートアーカイブの更新

次のコマンドは、現在のブートアーカイブを更新します。

```
# bootadm update-archive
```

例 2 別のルート上にあるブートアーカイブの更新

次のコマンドは、別のルート上にあるブートアーカイブを更新します。

```
# bootadm update-archive -R /a
```

例 3 インストール済み OS インスタンスの一覧表示

次のコマンドは、GRUB メニューに含まれている、インストール済みのオペレーティングシステムインスタンスを一覧表示します。

```
# bootadm list-menu

default=0
timeout=10
(0) Solaris10
(1) Solaris10 Failsafe
```

例 3 インストール済み OS インスタンスの一覧表示 (続き)

```
(2) Linux
```

例 4 デフォルトブートエントリの切り替え

次のコマンドは、1 つ前の例で表示されたメニューを参照しています。ユーザーは、Linux (項目 2) を選択しています。

```
# bootadm set-menu default=2
```

例 5 GRUB メニューのエントリと場所の一覧表示

次のコマンドは、GRUB メニューのエントリと場所を一覧表示しています。

```
# bootadm list-menu
The location for the active GRUB menu is: /stubboot/boot/grub/menu.lst
default 0
timeout 10
0 Solaris10
1 Solaris10 failsafe
2 Linux
```

例 6 GRUB メニューの場所の表示

次のコマンドは、GRUB メニューの場所を表示しています。

```
# bootadm list-menu
The location for the active GRUB menu is: /dev/dsk/c0t1d0s0 (not mounted)
The filesystem type of the menu device is <ufs>
default 2
timeout 10
0 c0t1d0s3
1 c0t1d0s3 failsafe
2 Solaris10
3 Solaris10 failsafe
```

この例では、アクティブな GRUB メニューが、マウントされていないデバイス上に格納されています。この GRUB メニューにアクセスするには、このデバイスをマウントし、<マウントポイント>/boot/grub/menu.lst にある GRUB メニューにアクセスします。

終了ステータス 次の終了ステータスが返されます。

0 コマンドが正常に完了しました。

1 エラーが発生したため、コマンドが終了しました。

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	安定

関連項目

boot(1M), installgrub(1M), attributes(5)

GRUB ホームページ:

http://www.gnu.org/software/grub/

名前　catman – 参照マニュアル用のフォーマット整形したファイルの作成

形式

```
/usr/bin/catman  [-c] [-n] [-p] [-t] [-w] [-M directory]
 [-T macro-package] [sections]
```

機能説明

catman は nroff(1) または sgml(5) の入力ファイルから、オンラインマニュアルページのプレフォーマット・バージョンを生成します。プレフォーマットされたマニュアルページのディレクトリは、自己包括的にかつフォーマットされていないエントリから独立して作成されるので、一群のマシンの間でそれらのマニュアルページを (rdist(1) などによって) 容易に配布することができます。

catman は、MANPATH または -M で指定されたディレクトリに windex のデータベースファイルも作成します。windex のデータベースファイルは、キーワード、そのキーワードが指す参照マニュアルページ、参照マニュアルページに書かれたユーティリティーまたはインタフェースの目的を説明するテキスト行の 3 つのカラムで構成されているリストです。各キーワードは、「名前」 (NAME) の行の '–' (ダッシュ) の前にあるコンマで区切られた単語のリストから抽出します。キーワードが指す参照マニュアルページは、「名前」の行の最初の単語です。3 つ目のカラムの記述は、「名前」の行の – のあとに続くテキストから抽出します。「名前」の行は、.TH マクロによって作成されるページヘッダーのすぐあとに置く必要があります (必要とされる形式については、「注意事項」を参照)。

各マニュアルページを検査し、それに対応するプレフォーマット・バージョンが存在しないものあるいは、現状のマニュアルページよりも古いものについて、プレフォーマット・バージョンを再生成します。変更箇所があれば、catman は windex データベースも再作成します。

マニュアルページがシャドウページである場合、つまりその内容が書かれた別のマニュアルページをソースファイルにしている場合、対象となるプレフォーマットされたマニュアルページへのシンボリックリンクが cat*x* または fmt*x* ディレクトリ内に作成されます。

フォーマットされていない nroff のソースファイルにあるシャドウファイルは、最初の行に .so man*x*/yyy.*x* という形式の行があることで識別できます。

SGML のソースファイルにあるシャドウファイルは、文字列 SHADOW_PAGE があることで識別できます。シャドウファイルで宣言されたファイルの実体は、ソースとなるファイルを示しています。

オプション　サポートしているオプションは、次のとおりです。

-c　SGML のソースファイルから、フォーマットされていない nroff のソースファイルを適切な man サブディレクトリに作成します。このオプションは SGML ファイルと同じ名前で man ディレクトリにある既存のファイルをすべて上書きします。

-n	windex データベースの作成または再作成を行いません。このオプションを指定した場合は windex データベースが作成されないので、apropos、whatis、man -f、man -k コマンドを実行しても失敗します。
-p	実際の処理は実行せず、どのような処理が行われるかの表示だけを行います。
-t	cat サブディレクトリへの nroff を行う代わりに、該当する fmt サブディレクトリ内に troff 処理後のエントリを作成します。
-w	whatis(1) および man(1) の -f と -k オプションで使用する windex データベースの作成だけを行います。マニュアルページの再フォーマットは実行しません。
-M *directory*	引数に指定したディレクトリ (デフォルトは /usr/share/man) 中にあるマニュアルページを更新します。コンマは、マニュアルセクションの番号を示すために使用されているので、-M オプションの引数として指定するディレクトリ名には ',' (コンマ) が含まれないようにしてください。man(1) を参照してください。
-T *macro-package*	標準のマニュアルページマクロの代わりに、*macro-package* 引数で指定したマクロパッケージを使用します (デフォルトは man(5))。

オペランド

次のオペランドを指定できます。

sections 先頭文字が '-' でないパラメータは、catman によって処理される マニュアルセクションとみなします。各セクションは空白で区切ります。このオペランドが指定されると、指定するマニュアルセクションだけが処理されます。次に例を示します。

```
catman 1 2 3
```

上記のコマンドは、セクション 1、2、3 のマニュアルページだけを更新します。セクションが指定されない場合、 環境変数 MANPATH に指定された man ディレクトリにあるすべてのセクションが処理されます。

環境

TROFF	-t オプションが指定された場合に用いるフォーマッタの名前。この環境変数が設定されていない場合は、 troff(1) が用いられます。
MANPATH	catman と man(1) で処理されるディレクトリの、コロンで区切られたリスト。各ディレクトリは、コンマで区切られたセクションのリストの後に続いて指定できます。この環境変数に値を設定すると、デフォルトのディレクトリ検索パスである /usr/share/man と、デフォルトのセクション検索パスである man.cf を無効にします。また、-M フラグはデフォルトのディレクトリ検索パスを、-s フラグはデフォルトのセクション検索パスを、それぞれ無効にします。

ファイル

/usr/share/man	マニュアルページのデフォルトのディレクトリ
/usr/share/man/man*/*.*	nroff 入力ファイル (未処理)
/usr/share/man/sman*/*.*	SGML 入力ファイル (未処理)
/usr/share/man/cat*/*.*	プレフォーマットされた nroff 処理後のマニュアルページ
/usr/share/man/fmt*/*.*	プレフォーマットされた troff 処理後のマニュアルページ
/usr/share/man/windex	目次とキーワードのデータベース
/usr/lib/makewhatis	windex データベース生成用のコマンドスクリプト
/usr/share/lib/tmac/an	デフォルトのマクロパッケージ

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWdoc
CSI	対応済み

関連項目

apropos(1), man(1), nroff(1), rdist(1), rm(1), troff(1), whatis(1), attributes(5), man(5), sgml(5)

診断

man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory
: メッセージの冒頭に示されたファイルは、カッコ内のファイルが参照しようとしたが、存在しませんでした。

target of .so in man?/xxx.? must be relative to /usr/man
: catman では、ディレクトリ /usr/man からの相対パスで示したファイル名のみを参照できます。

opendir:man?: No such file or directory
: catman が通常検索するディレクトリのうちの 1 つが見つかりません (軽度の警告メッセージ)。

.: No such file or directory
: catman によって、 中身が空のディレクトリが見つかりました (軽度の警告メッセージ)。

警告

以前に catman を実行して cat* ディレクトリがすでにインストールされている場合に、オペレーティングシステムをアップグレードした場合は、catman を実行する前に cat* ディレクトリ構造全体を削除してください (rm(1) 参照)。

すべての man* ディレクトリがそろっていない場合には、whatis データベースを再構築するために catman を再度実行しないでください。catman は、 man* ディレクトリにもとづいてこの windex ファイルを構築します。

注意事項

catman には、正しい windex のインデックスファイルを生成するための要件があります。catman は、個々のマニュアルページのファイル中に、特定の形式を持つ 2 つのマクロ行 (ページの先頭の .TH 行と .SH NAME の行) を必要とします。

.TH マクロには、最低 3 つの引数 (ファイル名、セクション番号、日付) が必要です。.TH行は、.TH マクロで始まり、そのあとに、1 つの空白、マニュアルページのファイル名、1 つの空白、セクション番号、1 つの空白、日付が続きます。日付は、“day month year” (日本語の場合は “年月日”) として二重引用符で囲みます。このうちの month には 3 文字の省略形 (Jan、Feb、Mar、など) を指定します。

.TH .SH NAME マクロは、.TH 行のすぐあとに置く必要があります (これら 2 つの行の間には何も置かない)。「名前」の行では、フォントの変更はできません。.SH NAME のすぐ次の行には、マニュアルページのファイル名、コンマで区切られたシャドウファイル名 (もしあれば)、ダッシュ、簡単な概要を含む行が続きます。これらの要素は、すべて 1 つの行に収める必要があります (改行を入れないでください)。

.TH 行と .SH NAME の行の正しいコーディング例を次に示します。

```
.TH nismatch 1M "10 Apr 1998"
.SH NAME
nismatch, nisgrep \- utilities for searching NIS+ tables
```

名前

cfgadm – 構成の管理

形式

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o hardware_options] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_options] -h
     [ap_id | ap_type]
```

機能説明

`cfgadm` コマンドを使用して、動的な再構成が可能なハードウェア資源に対して構成の管理を行うことができます。これらの操作には、状態 (state) の表示 (`-l`)、検査の開始 (`-t`)、構成状態の変更の開始 (`-c`)、ハードウェア固有の機能の実行 (`-x`)、および構成管理のヘルプ情報の表示 (`-h`)、が含まれます。構成管理は、接続点 (*attachment point*) で実行されます。接続点は、Solaris の動作中にハードウェア資源の動的再構成を行うことにシステムソフトウェアが対応している場所です。

構成の管理では、マシン上に実際にあるハードウェア資源と、構成済みで Solaris が認識できるハードウェア資源が区別されます。構成管理機能の特性はハードウェアに依存し、ハードウェア固有のライブラリを呼び出すことで実行されます。

構成管理は、接続点で実行されます。接続点に設置されているハードウェア資源には、システムの稼働中に物理的な交換ができるものとできないものがありますが、構成管理インタフェースによって、動的に再構成することはできます。

接続点は、接続点の向こう側に位置するハードウェア資源とは別の2 つの固有の要素を定義します。接続点の 2 つの要素は、受容体(*receptacle*) と占有装置 (*occupant*) です。ハードウェア資源の物理的な取り付け、取り外しは接続点で行われ、その結果、受容体に占有装置が追加されたり削除されたりします。構成管理は、接続点での構成管理機能だけでなく、この物理的な着脱操作にも対応しています。

接続点には、状態 (state) と条件 (condition) の情報が関連付けられています。構成管理インタフェースを使用して、接続点の状態の変化を制御することができます。受容体は、`empty`、`disconnected`、`connected` の 3 つの状態のいずれかになります。また、占有装置は、`configured` と `unconfigured` のいずれかの状態になります。

受容体は、接続点に占有装置がない場合に、必ず受容体の通常の状態である `empty` になります。この状態には、稼働中のシステムの一部を一時的に停止することができる、ハードウェア固有の機能が関係します。受容体が占有装置をシステムの通常の使用から切り離すことができる場合に、その受容体は `disconnected` 状態になることもできます。この状態は、占有装置のハードウェア資源をシステムが完全に利用できるようにする前にそのハードウェアに対する検査を実行する場合や、占有装置の物理的な取り外しや再構成のための準備の 1 つの段階として、主に使用されま

す。disconnected 状態の受容体は、ハードウェアの許容範囲内で占有装置をシステムから分離しますが、検査や設定が必要な場合は使用を許可する場合があります。受容体は、占有装置に含まれるハードウェア資源の通常の使用が許可されている場合に、必ず connected 状態になります。connected 状態は、占有装置を含み、かつ構成管理操作が実行されていない受容体の通常の状態です。

unconfigured 状態の占有装置に含まれるハードウェア資源は、Solaris の通常のデータ構造では表現されないため、Solaris はそのハードウェアを使用できません。未構成の占有装置に対して実行できる操作は、構成管理操作に限られています。configured 状態の占有装置に含まれるハードウェア資源は、Solaris の通常のデータ構造で表現されるため、Solaris は、一部またはすべてのハードウェア資源を使用することができます。占有装置は、必ず configured 状態か unconfigured 状態になります。

接続点は、unknown、ok、failing、failed、unusable の 5 つの条件のいずれかになります。接続点は、電源投入検査と不揮発性記録保存の結果によって、システムをどの条件にも置く可能性があります。

configured 状態の占有装置を持つ接続点は、unknown、ok、failing、failed の条件のいずれかになります。failing または failed 条件にない接続点は、ハードウェア固有の回復可能なエラーがしきい値を超えると、操作中に failing 状態になる場合があります。また、failed 条件にない接続点は、回復不可能なエラーによって、操作中に failed 条件に変わる場合があります。

unconfigured 状態にある占有装置を持つ接続点は今までに挙げた条件のどれになる可能性もあります。unconfigured 状態にある占有装置を持つ接続点の条件は、マシン固有の時間しきい値が経過した後に、ok から unknown になる場合があります。検査機能を開始した場合は、検査の結果によって接続点の条件が ok、failing、failed のいずれかに変わります。検査機能を持たない接続点は、接続点を unknown 条件のままにする場合があります。検査が中断された場合は、接続点の条件は、以前の条件、unknown、failed に設定することができます。unknown、ok、failing、failed のいずれかの条件にある接続点には、再検査を行うことができます。

接続点は、さまざまな理由によって unusable 条件になります。理由としては、受容体に対する不適切な電力投入や冷却、占有装置が認識できない、対応していない、不適切に構成されている、などが挙げられます。unusable 条件にある接続点は、システムで使用することができません。通常、この条件は、接続点に対して物理的な対処がなされない限り変わりません。

また、接続点は、状態の変更処理が進行中である場合や、条件が再評価されている場合に、それを示す使用状態情報を保持します。

接続点は、システムデバイス階層構造の中での接続点のタイプと位置に関連するハードウェア固有の識別子 (*ap_ids*) に対応しています。*ap_id* は単一の接続点を特定するために、一意になっている必要があります。*ap_id* の仕様には、物理タイプと

論理タイプの 2種類が用意されています。物理 *ap_id* には、完全なパス名を指定します。論理 *ap_id* には、簡略表記法を使用し、ユーザーにとってより簡単な方法で接続点を指定します。

たとえば、システムのバックプレーンスロット番号 7 にある接続点の物理 *ap_id* は `/devices/central/fhc/sysctrl:slot7` となり、論理 *ap_id* は `system:slot7` になります。また、システムの第 2 PCI 入出力バス上にある 3 番目の受容体の論理 *ap_id* は `pci2:plug3` になります。

接続点も動的に作成されます。動的接続点には、そのシステムに設定されている基本接続点を基にして名前が付けられます。動的接続点の `ap_ids` は、2 つのコロン (`::`)、基本構成要素、および動的構成要素で構成されます。基本構成要素は、基本接続点 `ap_id` です。動的構成要素は、ハードウェア固有で、対応するハードウェア固有のライブラリによって生成されます。

たとえば、SCSI HBA を表現し、物理 `ap_id` が`/devices/sbus@1f,0/SUNW,fas@e,8800000:scsi` で、論理 `ap_id` が c0 である基本接続点を想定します。この SCSI HBA に接続されているディスクは、論理 `ap_id` が `c0::dsk/c0t0d0` である動的接続点によって表現されます。ここで、`c0` は基本構成要素で、`dsk/c0t0d0` はハードウェア固有の動的構成要素です。同様に、この動的接続点の物理 `ap_id` は `/devices/sbus@1f,0/SUNW,fas@e,8800000:scsi::dsk/c0t0d0` になります。

ap_type は `ap_id` の一部で、それ自身だけでは一意にならず、単一の接続点を特定することができません。*ap_type* は、論理 `ap_id` の一部を含み、コロン (`:`) 区切り記号を含まない部分文字列です。たとえば、`pci` の *ap_type* は、論理 `ap_id` が `pci` で始まる 接続点をすべて出力します。

ap_types は、できるだけ使用しないでください。`-s` オプションの新しい選択サブオプションを使用すれば、より汎用的な方法で柔軟に接続点を選択することができます。各オプションを参照してください。

`cfgadm` コマンドは、ハードウェア固有ライブラリに含まれるハードウェア固有の機能と主に対話するため、cfgadm コマンドの動作はハードウェアに依存します。

それぞれの構成管理操作では、サービスの中断が必要になる場合があります。要求された機能を完了するために、対話式で操作中のユーザーにとって目に見えるサービスの中断が必要になる場合は、機能の開始前に標準エラー出力に確認メッセージが表示され、標準入力による確認を促します。すべての質問に対する yes を意味する `-y` オプション、または no を意味する `-n` オプションを指定することによって、確認を省略することもできます。検査レベルなどのハードウェア固有のオプションは `-o` オプションを使用して、サブオプションとして指定することができます。

システム構成の状態を変更する操作は、システムログデーモンである syslogd(1M) によって監視されます。

このコマンドの引数は、`getopt`(3C) および `getsubopt`(3C) の構文規約に従います。

オプション　次のオプションを指定できます。

`-a`　`-l`オプションによって、動的接続点のリストも出力されるように指定します。

`-c`*function*　*ap_id* で指定された接続点の状態を *function* で指定された状態に変更します。

function には、`insert`、`remove`、`disconnect`、`connect`、`configure`、`unconfigure` のいずれかを指定することができます。これらの関数は、ハードウェア固有のライブラリルーチンを呼び出して接続点の状態を変更します。これらの関数の定義を以下に示します。

`insert`　占有装置を手動で追加する操作を実行したり、物理的な追加を実行するハードウェア機能を起動します。`insert` には、システムの一部を一時的に停止するハードウェア固有の副作用が伴う場合があります。このような場合、ハードウェア固有のライブラリは、対応する警告メッセージを生成し、ユーザーに対して、そのハードウェア固有の注意事項と手順を提供します。ハードウェアに起因するさまざまなエラーによってこの関数が失敗し、受容体の条件が `unusable` になる場合があります。

`remove`　占有装置を手動で削除する操作を実行したり、物理的な削除を実行するハードウェア機能を起動します。`remove` には、システムの一部を一時的に停止するハードウェア固有の副作用が伴う場合があります。このような場合、ハードウェア固有のライブラリは、対応する警告メッセージを生成し、ユーザーに対して、そのハードウェア固有の注意事項と手順を提供します。ハードウェアに起因するさまざまなエラーによってこの関数が失敗し、受容体の条件が unusable になる場合があります。

`disconnect`　ハードウェア固有の操作を実行して、受容体を disconnected 状態にします。disconnected 状態にすることによって、

占有装置に対して、受容体を介した通常の方法による操作が行えなくなります。

`connect` ハードウェア固有の操作を実行して、受容体を `connected` 状態にします。受容体が `connect` 状態にすることによって、占有装置に対して、受容体を介した通常の方法による操作が行えるようになります。

`configure` ハードウェア固有の操作を実行して、占有装置のハードウェア資源を Solaris が使用できるようにします。構成された占有装置はシステム構成の一部になり、`psradm`(1M)、mount(1M)、ifconfig(1m) などの Solarisデバイス操作メンテナンスコマンドによる操作の対象となります。

`unconfigure` ハードウェア固有の操作を実行して、占有装置のハードウェア資源をシステムから論理的に削除します。この関数を使用するには、占有装置が現在構成されていて、占有装置のハードウェアが Solaris によって使用されていない必要があります。

状態変更関数は、接続点の条件や、その他のハードウェア固有の問題によって失敗する場合があります。資源を追加するための状態変更関数 (`insert`、`connect`、`configure`) は、接続点が `ok` または `unknown` 条件にある場合に、ハードウェア固有のライブラリに渡されます。接続点がそれ以外の条件にある場合は、強制オプション (-f) を使用した場合に限って、資源を追加するための関数がハードウェア固有のライブラリに渡されます。システムからハードウェア資源を削除するための関数 (`remove`、`disconnect`、`unconfigure`) によるハードウェア固有のライブラリの呼び出しは、接続点の条件によって妨げられることはありません。接続点が `unknown` 条件にある場合に、関数は、ハードウェア固有のライブラリによって拒否される場合があります。

接続点の条件は、状態変更関数によって変更されないこともありますが、状態変更操作中のエラーによって接続点の条件が変わる場合があります。条件の書き換えと状

態の強制的な変更は、強制オプション (-f) を指定した場合だけ実行することができます。これらの処理は、強制オプションを指定しないと失敗します。強制オプションは、ハードウェア固有の安全性検査および完全性検査によって無効になる場合があります。

-f
: 指定された処理を強制的に実行します。これは主に、ハードウェア固有の安全機能を無効にするために使用します。状態の変更操作を強制することによって、ハードウェア固有の安全検査によって ok や unknown 状態にない占有装置のハードウェア資源を使用することができる場合があります。

-h [*ap_id* | *ap_type* . . .]
: ヘルプメッセージテキストを出力します。*ap_id* または*ap_type* を指定すると、この引数によって指定された接続点に関するハードウェア固有のライブラリのヘルプルーチンが呼び出されます。

-l [*ap_id* | *ap_type* . . .]
: 指定された接続点の状態や条件を一覧表示します。接続点を抽出するには、-s オプションと select サブオプションを使用します。いずれかの処理オプションを使用せずに cfgadm コマンドを起動するのは、引数を使用せずに -l を使用するのと同じです。表示画面の書式は -v および -s オプションによって制御されます。-a オプションが指定されているときは、接続点が動的に展開されます。

-n
: 対話型の確認を行わず、応答が *no* であるとみなします。-n と -y のいずれも指定しないと、標準エラー出力と標準入力によって対話型の確認が行われます。これらの標準的なチャネルのいずれも端末 (isatty(3C) によって判定されている) に対応していない場合は、-n オプションが想定されます。

-o*hardware_options*
: コマンドの主オプションに対してハードウェア固有のオプションを指定します。hardware_options の文字列の書式と内容は完全にハードウェア固有のものです。オプション文字列の *hardware_options* は getsubopt(3C) の構文規約に従います。

-s*listing_options*
: 一覧表示 (-l) コマンドに対して一覧表示オプションを指定します。*listing_options* は getsubopt(3C) の構文規約に従います。サブオプションを使用して、接続点の選択条件 (select=*select_string*)、適切な照合タイプ (match=*match_type*)、一覧表示する順序 (sort=*field_spec*)、

表示するデータ (cols=*field_spec* と cols2=*field_spec*)、列の区切り記号 (delim=*string*)、列の見出し行の抑制 (noheadings) を指定します。

select サブオプションを指定すると、指定された条件と一致する接続点だけが一覧表示されます。select サブオプションの構文は次のとおりです。

```
cfgadm -s select=attr1(value1):attr2(value2)...
```

attr は、ap_id、class、type のいずれかです。ap_id は論理 *ap_id* フィールド、class は接続点のクラス、type はタイプフィールドです。*value1*、*value2* などは、照合する値です。照合タイプを指定するには、次のように match サブオプションを使用します。

```
cfgadm -s match=match_type,select=attr1(value1)...
```

match_type は、exact または partial から選択します。デフォルト値は exact です。

select サブオプションの引数は、シェルから保護するために引用符で囲みます。

field_spec には、1 つの *data-field*、または次のようにコロン (:) で区切った複数の *data-field* を指定します。*data-field*:*data-field*:*data-field*。*data-field* は、ap_id、physid、r_state、o_state、condition、type、busy、status_time、status_time_p、class、info のいずれかです。ap_id フィールドの出力は接続点の論理名で、physidフィールドは物理名です。r_state フィールドは、empty、disconnected、connected のいずれかになります。o_stateフィールドは、configured と unconfigured のいずれかになります。busy フィールドは、接続点が使用中の場合に y になり、使用中でない場合に n となります。type フィールドと info フィールドはハードウェア固有のフィールドです。status_time フィールドには、r_state、o_state、または接続点の条件が最後に変更された時刻が表示されます。status_time_p フィールドは、構文解析が可能な status_time フィールドです。接続点にクラスが関連付けられている場合は、class フィールドにクラス名が表示されます。接続点にクラスが関連付けられていない場合は、class フィールドに none が表示されます。

field_spec 内のフィールドの順序は重要です。sort サブオプションでは、最初に与えられたフィールドが主ソートキーになります。cols および cols2 サブオプションでは、指定した順序でフィールドが出力されます。*data-field* に対するソートの順序は、sort サブオプションに対する *field_sec* 内の *data-field* 名の前にマイナス (–) を付けることによって逆になります。sort のデフォルトの値は ap_id です。cols および cols2 のデフォルトの値は -v オプションが指定されているかどうかによって異なります。-v が指定されていない場合は、cols は ap_id:r_state:o_state:condition になり、cols2 は設定されません。-v が指定されている場合は、cols は ap_id:r_state:o_state:condition:info になり、cols2 は status_time:type:activity:physid: になります。delim のデフォルトの値は、単一の空白文字です。delim の値には、任意の長さの文字列を指定することもできます。区切り記号にはコンマ (,) を含めることはできません (getsubopt(3C) を参照)。これらの一覧表示オプションは、構文解析が可能な出力を生成するために使用することができます。「注意事項」 を参照してください。

-t

1 つまたは複数の接続点の検査を実行します。この検査機能は、接続点の条件を再評価するために使用します。*hardware_options* の中で検査レベル指示子を指定しないと、重度の障害を検出する最も早い検査方法が使用されます。

より包括的な検査は個々のハードウェアに依存するため、*hardware_options* を使用して選択します。

検査の結果は、指定された占有装置の条件を ok (障害が発見されなかった場合)、failing (回復可能な障害が発見された場合)、failed (回復不可能な障害が発見された場合) のいずれかに更新するために使用されます。

検査が中断された場合は、接続点の条件は、以前の値に戻すか、ok (障害が発見されなかった場合)、failing (回復可能な障害が発見された場合)、failed (回復不可能な障害が発見された場合) のいずれかに設定されます。接続点は、エラーがなく、検査が正常に完了した場合のみ ok に設定されます。

-v

詳細表示モードで実行します。-c、-t、-x オプションを指定した場合に、各試行操作の結果を表示するメッセージを出力します。-h オプションを指定した場合は、詳細

なヘルプ情報が表示されます。-l オプションを指定した場合は、各接続点に関するすべての情報を出力します。

-x*hardware_function*　ハードウェア固有の機能を実行します。受容体や占有装置の状態は、専用ハードウェア固有の機能を使用して変更します。接続点の状態は、ハードウェア固有の機能の動作中に検出されたエラーの結果として変化する場合があります。*hardware_function* 文字列の書式と内容は完全にハードウェア固有のものです。オプション文字列の *hardware_function* は getsubopt(3C) の構文規約に従います。

-y　対話型の確認を行わず、応答が yes であるとみなします。

使用法

このコマンドを使用するために必要な特権は、ハードウェアに依存します。一般的に、デフォルトのシステム設定では、一覧表示オプション以外のすべての機能はスーパーユーザーのみが使用することができます。

使用例

例 1　デバイスツリー内の接続点の一覧表示

以下の例は、動的接続点以外のすべての接続点を一覧表示します。

```
example# cfgadm

  Ap_Id         Type        Receptacle      Occupant       Cond
system:slot0    cpu/mem     connected       configured     ok
system:slot1    sbus-upa    connected       configured     ok
system:slot2    cpu/mem     connected       configured     ok
system:slot3    unknown     connected       unconfigured   unknown
system:slot4    dual-sbus   connected       configured     failing
system:slot5    cpu/mem     connected       configured     ok
system:slot6    unknown     disconnected    unconfigured   unusable
system:slot7    unknown     empty           unconfigured   ok
c0              scsi-bus    connected       configured     unknown
c1              scsi-bus    connected       configured     unknown
```

例 2　構成することができるすべてのハードウェアの情報の一覧表示

以下の例は、現在構成することができるすべてのハードウェアの情報を一覧表示します。動的接続点で表現されるハードウェアも表示されます。

```
example# cfgadm -al

  Ap_Id           Type         Receptacle      Occupant        Cond
system:slot0      cpu/mem      connected       configured      ok
system:slot1      sbus-upa     connected       configured      ok
```

例 2 構成することができるすべてのハードウェアの情報の一覧表示 (続き)

```
system:slot2       cpu/mem      connected       configured      ok
system:slot3       unknown      connected       unconfigured    unknown
system:slot4       dual-sbus    connected       configured      failing
system:slot5       cpu/mem      connected       configured      ok
system:slot6       unknown      disconnected    unconfigured    unusable
system:slot7       unknown      empty           unconfigured    ok
c0                 scsi-bus     connected       configured      unknown
c0::dsk/c0t14d0    disk         connected       configured      unknown
c0::dsk/c0t11d0    disk         connected       configured      unknown
c0::dsk/c0t8d0     disk         connected       configured      unknown
c0::rmt/0          tape         connected       configured      unknown
c1                 scsi-bus     connected       configured      unknown
```

例 3 接続点の属性に基づいて選択された情報の一覧表示

以下の例は、接続点のうち、scsi で始まるクラス、c で始まる ap_id、および scsi で始まる type フィールドを持つものをすべて一覧表示します。-s オプションの引数は、シェルから保護するために引用符で囲みます。

```
example# cfgadm -s "match=partial,select=class(scsi):ap_id(c):type(scsi)"

Ap_Id         Type          Receptacle      Occupant          Cond
 c0          scsi-bus      connected       configured        unknown
 c1          scsi-bus      connected       configured        unknown
```

例 4 現在構成することができるハードウェアの情報の詳細表示

以下の例は、*ap-type* system の現在構成することができるハードウェアの情報を詳細表示モードで示します。

```
example# cfgadm -v -l system
Ap_Id                      Receptacle Occupant   Condition Information
When         Type      Busy     Phys_Id
system:slot1               connected  configured ok
Apr  4 23:50 sbus-upa  n        /devices/central/fhc/sysctrl:slot1
system:slot3               connected  configured ok        non-detachable
Apr 17 11:20 cpu/mem   n        /devices/central/fhc/sysctrl:slot3
system:slot5               connected  configured ok
Apr  4 23:50 cpu/mem   n        /devices/central/fhc/sysctrl:slot5
system:slot7               connected  configured ok
Apr  4 23:50 dual-sbus n        /devices/central/fhc/sysctrl:slot7
```

When 列は status_time フィールドの内容を表示します。

例 5　ハードウェア固有の拡張検査を使用した 2 つの占有装置の検査

以下の例は、ハードウェア固有の拡張検査を使用して 2 つの占有装置を検査します。

```
example# cfgadm -v -o extended -t system:slot3 system:slot5
Testing attachment point system:slot3 ...  ok
Testing attachment point system:slot5 ...  ok
```

例 6　強制オプションを使用した占有装置の構成

以下の例は、強制オプションを使用して、failing 状態の占有装置をシステムに構成します。

```
example# cfgadm -f -c configure system:slot3
```

例 7　システムからの占有装置の構成解除

以下の例は、システムから占有装置を構成解除します。

```
example# cfgadm -c unconfigure system:slot4
```

例 8　接続点の占有装置の構成

以下の例は、占有装置を構成します。

```
example# cfgadm -c configure c0::dsk/c0t0d0
```

環境

cfgadm の実行に影響を与える次の環境変数についての詳細は、environ(5) を参照してください。LC_TIME、LC_MESSAGES、NLSPATH、および TZ。

LC_MESSAGES　cfgadm が見出し行とエラーメッセージを表示する方法を特定します。出力されるデータは、この環境変数の影響を受けません。

LC_TIME　cfgadm が状態の変更時間 (status_time) を表示する方法 (人による判読が可能な形式) を特定します。

TZ　状態の変更時間を変換する際に使用される時間帯を特定します。これは人による判読が可能な (status_time) と構文解析が可能な (status_time_p) 形式の両方に当てはまります。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

1　エラーが発生した。

2　指定された宛先が構成管理に対応していない。

3　使用方法上のエラー。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

cfgadm_fp(1M), cfgadm_ib(1M), cfgadm_pci(1M),cfgadm_sbd(1M), cfgadm_scsi(1M), cfgadm_usb(1M), ifconfig(1m), mount(1M), prtdiag(1M), psradm(1M), syslogd(1M), config_admin(3CFGADM), getopt(3C), getsubopt(3C), isatty(3C), attributes(5), environ(5)

診断

診断メッセージは標準エラー出力に表示されます。オプションや使用方法のエラー以外に、次の診断メッセージがこのユーティリティによって表示されます。

cfgadm: Configuration administration not supported on*ap_id*

cfgadm: No library found for *ap_id*

cfgadm: *ap_id*is ambiguous

cfgadm: *operation*: Insufficient privileges

cfgadm: Attachment point is busy, try again

cfgadm: No attachment points with specified attributes found

cfgadm: System is busy, try again

cfgadm: *operation*: Operation requires a service interruption

cfgadm: *operation*: Data error: *error_text*

cfgadm: *operation*: Hardware specific failure: *error_text*

エラーメッセージの詳細については config_admin(3CFGADM) を参照してください。

注意事項

ハードウェア資源は、ハードウェア固有の方法で未構成プールに入ります。これは、システムの初期設定や構成解除操作の結果としてなど、さまざまな状況で発生します。unconfigured 状態にある占有装置は、システムによる特定の介入が発生するまで、システムが使用することはできません。このような干渉は、オペレーターが起動したコマンドや、自動構成機構によって発生します。

cfgadm コマンドの一覧表示オプションは、シェルスクリプトの中などで、別のコマンドに対する構文解析可能な入力として使用することができます。構文解析可能な出力の場合は、必要なフィールドを選択するときに、-s オプションを使用する必要があります。-s オプションは、列の見出しを抑制するときにも使用できます。次のフィールドは、常に構文解析が可能な出力を生成します。ap_id、physid、r_state、o_state、condition、busy、status_time_p、class、および type。解析可能な出力には、フィールドの値に空白文字が含まれることはありません。

以下はシェルスクリプトの一部分で、タイプが CPU の正常な未構成の占有装置で最初のものを検出します。

```
found=
cfgadm -l -s "noheadings,cols=ap_id:r_state:condition:type" | \
while read ap_id r_state cond type
do
     if [ "$r_state" = unconfigured -a "$cond" = ok -a "$type" = CPU ]
     then
          if [ -z "$found" ]
          then
               found=$ap_id
          fi
     fi
done
if [ -n "$found" ]
then
         echo "Found CPU $found"
fi
```

構文解析が可能な時間フィールド (status_time_p) の形式は、*YYYYMMDDhhmmss* で、文字列の比較を行うのに都合がよい書式で、年、月、日、時間、分、秒を表示します。

システム構成管理で使用できるものの詳細については、ハードウェア固有のマニュアルを参照してください。

名前

cfgadm_ac – EXX00 メモリーシステムの管理

形式

```
/usr/sbin/cfgadm  [-c configure] [-f] [-o disable-at-boot
 |  enable-at-boot ] ac#:bank# ...

/usr/sbin/cfgadm  [-c unconfigure] [-o disable-at-bootp
 |  enable-at-boot  ] ac#:bank# ...

/usr/sbin/cfgadm  [-v] [-o quick | normal | extended,
 [max_errors=#] ] -t ac#:bank#...

/usr/sbin/cfgadm  -x relocate-test ac#:bank# ...

/usr/sbin/cfgadm  [-l] -o disable-at-boot |  enable-at-boot
 ac#:bank# ...
```

機能説明

ac ハードウェア固有ライブラリ `/usr/platform/sun4u/lib/cfgadm/cfgadm_ac.so.1` は、 cfgadm_sysctrl(1m) を使用する CPUU/メモリーボードの Dynamic Reconfiguration (DR: 動的再構成）の一部として、`E6X00`、`E5X00`、`E4X00`、`E3X00` システムのメモリーバンクの構成と構成解除に関連する機能を提供します。

メモリーバンクは、デバイスツリー上の接続点として表されます。 CPU/メモリーボードそれぞれに2つの接続点が用意されます。1つがボード上の各バンク (バンク `0` とバンク `1`) 用になります。 バンクが空いている場合は、受容体の状態は empty (空き) になります。バンクが使用されている場合は、受容体の状態は connected(接続) になります。 受容体の状態が disconnected (接続解除) になることはありません。接続されているメモリーバンクの占有状態は、構成することも構成解除することもできます。 占有状態が構成されている場合は、メモリーは Solaris によって使用されています。構成解除されている場合は、使用されていません。

オプション

コマンドオプションの詳細については、`cfgadm`(1M) を参照してください。

以下のオプションがサポートされています。

-c configure | unconfigure

占有状態を変更します。引数が `configure` の場合は、メモリーが初期化され、Solaris のメモリープールに追加します。引数が `unconfigure` の場合は、Solaris によって使用されているメモリーを切り離します。CPU/メモリーボードをシステムから取り外すには、2 つのメモリーバンクを構成解除する必要があります。

ボード上のメモリーが `disabled-at-boot` (`info`フィールド参照) に指定されていて、`-f` (強制) オプションやブート許可フラグ (`-o enable-at-boot`) がいずれも指定されていない場合は、`cfgadm` コマンドは、構成操作を許可しません。構成操作には、初期化する必要のあるメモリーのサイズに応じて、少し時間がかかります。

システムに十分な使用可能メモリーがない (VM viability エラー)、または構成解除されるべきバンクに切り離せないメモリーがある (non-relocatable pages エラー) 場合、cfgadm コマンドは、構成解除操作を許可しません。 再配置不可のページの現状は、info フィールドの一覧にある permanent によって表されます。 Solaris が使用しているメモリの取り外しを行うと、システム負荷や補助記憶装置のページングの大きさによって、かなり時間がかかります。構成解除操作は、いつでも中止することができ、シグナルによるコマンドの中断によって、メモリーは完全に構成されている状態に戻ります。もし、タイムアウトまでの期間内に削除できるメモリーがない場合は、構成解除操作は、自動的に取り消されます。デフォルトのタイムアウト期間は 60 秒ですが、-o timeout=# オプションで変更することもできます。この数値を 0 にすると、タイムアウトは不許可になります。

`-f`

強制オプション。このオプションは、非揮発性の変数 disabled-memory-list で、disabled at boot (ブート不可) に指定された、構成されているメモリーバンクのブロックを無効にしたい場合に使用してください。詳細は、『特記事項: *Sun Enterprise 6x00/5x00/4x00/3x00* システム』を参照してください。

`-l`

リストオプション。このオプションについての説明は、cfgadm(1M)のマニュアルページを参照してください。

type フィールドは、常に *memory* です。

info フィールドには、以下に示すような空のメモリーバンクについての情報が保持されます。

```
slot# empty
```

slot# は、CPU/メモリーボードが挿入されているシステムスロットを示しています。たとえば、これがスロット 11 である場合、cfgadm が関連付けられたボードを操作するために使用する接続点は、 sysctrl0:slot11 になります。*info* フィールドには、以下に示すような接続されているバンクについての情報が保持されます。

```
slot# sizeMB|sizeGB [(sizeMB|sizeGB を使用)] base 0x###
      [interleaved #-way] [disabled at boot] [permanent]
```

バンクのサイズは、MB か GB か適切な単位で与えられます。メモリーがまだ使い切られていない場合、使用サイズが表示されます。 物理ベースアドレスは、16 進数で与えられます。メモリーバンクが他のバンクによってインタリーブされている場合は、インタリーブファクター(因子) が出力されます。ボード上のメモリーが、非揮発性の変数 `disabled-memory-list` を使ってブート不可になっている場合は、そのことが表示されます。バンクに切り離せないメモリーがあると、固定 (permanent) として出力されます。

`-o` disable-at-boot | enable-at-boot

このオプションは、非揮発性の disabled-memory-list変数を変更することを許可します。これらのオプションは、コマンド要求がない場合は、`-c` オプションの発行や、明白または暗示的なリスティングオプション `-l` と結合して、使用することができます。使用不可メモリーの一覧にあるボード上に構成されているメモリーブロックを無効にするには、 `configure` コマンドを `-o enable-at-boot` オプションで使用してください。

`-o` extended | normal | quick

テストレベルを指定するには、`-t` オプションを使用してください。

テストレベル `normal` では、各メモリーセルに `0` と `1` のいずれも格納できることを確認して、さらにすべてのセルが個別にアドレス可能であるかどうか検査します。 テストレベル `quick` では、すべてのメモリーに `0` と `1` を書き込むテストだけを行うため、アドレス線の障害を発 見することはできません。`extended` テストでは、近接したセル同士の干渉の問題をテストするパターンを使用します。デフォルトのテストレベルは、`normal` です。 `-t` オプションを参照してください。

`-o` max_errors=#

`-t` オプションと共に使用して、許可するエラーの最大数を指定します。このオプションを指定しなかった場合は、デフォルト値の 32 が適用されます。

`-o` timeout=#	構成解除コマンドと共に使用して、自動取り消しによるタイムアウトを設定します。デフォルト値は `60` で、単位は秒です。数値が `0` の場合は、タイムアウトしません。
`-t`	構成されていないメモリーのバンクをテストします。`-o quick|normal|extended` オプションでテストレベルを指定してください。 メモリーバンクテストが実行できた場合は、`cfgadm` コマンドはステータス `0` (成功) で終了します。テストの結果は、接続点のためという条件で利用可能です。
`-v`	詳細表示オプションです。`-t` オプションと組み合わせて使用することにより、テストの進行状況と結果を詳細に表示できます。
`-x` relocate-test	指定されたメモリーバンクで使用されているメモリーの全ページに対し、構成解除コマンドで行われるように、再配置を試みます。この操作の成功は、バンクが構成解除できるかどうかを保証しません。この操作の失敗は、構成解除できなかったことを意味します。このオプションの用途は、テストのみに限定されます。

オペランド

以下のオペランドがサポートされています。

ac#:*bank#*

メモリーバンクの接続点は、アドレスコントローラー(*ac#*) ドライバ (ac) のインスタンスによって作成されます。 ac ドライバの 1 つのインスタンスは、各システムボードによって作成されますが、CPU/メモリーボードに関連付けられたインスタンスだけは、バンク 0 とバンク 1 の 2 つの接続点を作成します。

このフォームは、`cfgadm`(1M) によって与えられた論理 `ap_id` 指定と一致します。これに対応する物理 `ap_id` のリストは、ファイルの項目にあります。

ac ドライバインスタンスの番号付けは、対応するボードのスロット番号とは関係はありません。完全な物理接続点の識別子には、`fhc@` に続く、スロット番号の 2 倍を 16 進数で表したスロット番号が入ります。

ファイル

/devices/fhc@*,f8800000/ac@0,1000000:bank?	接続点
/usr/platform/sun4u/lib/cfgadm/cfgadm_ac.so.1	ハードウェア固有ライブラリファイル

属性

以下の属性についての説明は、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWkvm.u

関連項目

cfgadm(1m), cfgadm_sysctrl(1m), `config_admin`(3CFGADM), `attributes`(5)

『日本語*Solaris7 Sun Enterprise 6X00, 5X00, 4X00, 3X00* システム *Dynamic Reconfiguration* ユーザーマニュアル』

『特記事項: *Sun Enterprise 6x00/5x00/4x00/3x00* システム』

注意事項

EXX00 システムの CPU/メモリーボードの Dynamic Reconfigura-tion に関する詳細は、『日本語*Solaris7 Sun Enterprise 6x00, 5x00, 4x00, 3x00* システム *Dynamic Reconfiguration* ユーザーマニュアル』を参照してください。

名前　cfgadm_sysctrl – EXX00 システムボードの管理

形式

```
/usr/sbin/cfgadm -c function [-f]
     [-o disable-at-boot | enable-at-boot] [-n | -y] sysctrl0:slot# ...

/usr/sbin/cfgadm -x quiesce-test sysctrl0:slot#

/usr/sbin/cfgadm -x insert-test | remove-test sysctrl0:slot# ...

/usr/sbin/cfgadm -x set-condition-test=# sysctrl0:slot# ...

/usr/sbin/cfgadm [-l]
     -o disable-at-boot | enable-at-boot sysctrl0:slot# ...
```

機能説明

sysctrl ハードウェア固有ライブラリ、/usr/platform/sun4u/lib/cfgadm/sysctrl.so.1 により、E6X00、E5X00、E4X00、E3X00 システム上で、動的再構成を用いてシステムボードの構成や構成解除ができるようになります。これによりシステムが稼働中であっても再起動せずに、I/O ボード、および CPU ボードを、Solaris 用に設定されたシステムのスロットに挿入できるようになります。また同様に、いずれのタイプのボードも稼働中のシステムから切断および取り外しが可能になり、再起動の必要はなくなります。

システムスロットはデバイスツリー上で「接続点 (attachment point)」として表現され、システムシャーシ内の各スロットには、この接続点が 1 つづつ設定されます。スロットにボードが設置されていない場合は、受容体の状態は empty と認識されます。ボードの電源が切断されシステムから取り外せる状態の場合は、受容体の状態は disconnected (接続解除) と認識されます。ボードに電源が投入されシステムバスに接続されている状態であれば、受容体の状態は connected (接続) と認識されます。

受容体の状態が empty の場合、その受容体の占有装置は unconfigured の状態にあると認識されます。受容体の状態が connected の場合は、その受容体の占有装置は configured または unconfigured のいずれかの状態を取ります。

Solaris でボード上のデバイスを利用できるのは、この状態が configured になっている場合です。unconfigured になっている場合、ボード上のデバイスは利用できません。

ボードを挿入すると、受容体の状態は empty から disconnected に変わります。ボードを取り外すと、受容体の状態は disconnected から empty に変わります。connected になっているボードを取り外してしまうと、オペレーティングシステムのクラッシュを起こし、システムに修復不可能な障害を与える場合があります。

オプション

オプションについての詳細は、マニュアルページの cfgadm(1m) の項目を参照してください。

次のオプションを指定できます。

-c *function*　状態の変更に使用します。*function* の部分に connect、disconnect、configure、unconfigure のいずれかを指定します。

configure	占有装置の状態を `configure` に変更します。 受容体の状態が `disconnected` になっている場合は、`configure` 機能はまず受容体への接続を試みます。`connect` 機能の一部として作成される OBP デバイスツリーを調べてSolaris デバイスノードを作成し、要求のあったデバイスを接続します。CPU/メモリーボードの場合は、電源が切断状態にある CPUのリストに、この CPU を加えます。この情報は `psrinfo`(1M) または `psradm`(1M) コマンドを用いて見ることができます。CPU/メモリーボードにはメモリー接続点が 2 つ作成されます。新規に設置したボード上の I/Oデバイスを使えるように設定するには、mount(1M) および ifconfig(1m) コマンドを利用します。新たなプロセッサをオンラインに設定するには、`psradm -n` コマンドを利用します。メモリーバンクのテストおよび設定には cfgadm_ac(1M) を利用します。
connect	受容体の状態を `connected` に変更します。 受容体の状態の変更を要求すると、バス信号の接続中はシステムバスを凍結し、その間にボードをテストします。休止処理によってバスは凍結します。この休止処理の間は、すべてのプロセス活動は停止し、すべてのドライバの動作が中断します。この休止処理およびそれに後続する復元処理にはかなりの時間がかかります。またすべてのドライバがこの機能をサポートして

いるわけではありません。このため、-x quiesce-test というオプションが用意されています。これを使用してシステムバスを凍結させている間に、接続されているボードをファームウェアからテストできます。I/O ボードの場合はこの処理には短時間しか費やしませんが、CPU/メモリーボードの場合は CPU 外部キャッシュのテストが原因で非常に長時間かかります。ここではメモリーのテストはしません。休止処理を開始する前には、実行の確認を促してきます。オプションに -y または -n を使用すると、この確認作業を省略できます。disabled-at-bootに指定されているボードは、接続処理を拒否します。ただし、指定された処理を強制実行する -f フラグを利用、または -o enable-at-boot を使って指定を無効化した場合は、接続処理が実行されます。-l の解説を参照してください。

disconnect　受容体の状態を disconnected に変更します。

占有装置の状態が configure になっている場合は、まず占有装置を構成解除します。この disconnect では休止処理を実行しないので、短時間で終了します。ボードの電源は切断され、取り外しが可能な状態になります。

unconfigure　占有装置の状態を unconfigureed に変更します。

この処理の間、ボード上のデバイス はSolaris からは見えない状態になります。I/O ボード上の

I/O デバイスは、 Solarisのデバイスツリーから削除されます。デバイスが使用中の場合、この unconfigure の処理は停止し、使用中であることを報告してきます。この場合はデバイスの使用を停止して、その後で unconfigure 処理を再試行する必要があります。CPU/メモリーボードの場合は、ボードの unconfigure 処理を実行する前に、メモリーの状態を unconfigure に設定しておかなければいけません。ボード上の CPU をオフラインにしてから、電源を切断して Solaris CPU リストから削除します。接続中のプロセスが存在する CPU をオフラインにすることはできません。CPU のオフライン化についての詳細は、マニュアルページの `psradm`(1M), `psrinfo`(1M), `pbind`(1M), `p_online`(2) の各項目を参照してください。

`-f`

強制的に、ボードに接続されているブロックの非揮発性変数 `disabled-board-list` を、`disabled-at-boot` に指定します。『特記事項: Sun Enterprise 6x00、5x00、4x00、3x00 システム』のマニュアルも参照してください。

`-l`

リストを表示させるオプションです。これはマニュアルページの `cfgadm`(1M) で解説しているものと同様の機能です。

type フィールドには `cpu/mem`、`mem`、`dual-sbus`、`sbus-upa`、`dual-pci`、`soc+sbus`、`soc+upa`、`disk`、`unknown` のいずれかを指定できます。

ハードウェア固有情報のフィールドは、次のように設定されています。 `[disabled at boot] [non-detachable] [100 MHz capable]`

ボードのタイプが `sbus-upa` または `soc+upa` の場合は、まず最初に次のような追加情報が表示さ

れます。[single buffered ffb|double buffered ffb|no ffb installed] ボードのタイプが disk の場合は、次のような追加情報が最初に表示されます。{target: # | no disk} {target: # | no disk}

-o disable-at-boot | enable-at-boot

非揮発性変数 disabled-board-list の変更に使用します。この -o オプションを使う時は、-c *function* または -l オプションと併用します。

disabled-at-boot に設定されているボードに関連するブロックを起動時に有効となるようにするには、-o enable-at-boot と -c connect を併用します。

-x insert-test | remove-test

テストを実行します。

ここで remove-test を使うと、ボードを物理的に取り外さなくても、テストシーケンスを自動処理している間、指定したスロットのドライバ状態は disconnected から empty に変更されます。

また、insert-test の方を使った場合は、remove-testコマンドで empty に指定したスロットのドライバ状態が isconnected に変更され、ボードはスロットに挿入されているものとして認識されます。

-x quiesce-test sysctrl0:slot1

テストを実行します。

このテストの実行により、ボードの接続処理に必要となる quiesce 処理が実行できる状態になります。現在のソフトウェアおよびハードウェアの構成を保ったままで、システムを確実に休止できるようになります。休止処理できないデバイスやプロセスが存在した場合は、エラーメッセージでその名前を表示します。このコマンドは有効なボード接続点に対してであればどれにでも使用可能ですが、いかなるシステムも必ず slot1 を 1 つ所有しているので、上に示した形式での使用を推奨しておきます。

-x set-condition-test=#

テストを実行します。

状態変更コマンドのポリシーロジックをテストできるように、システムボードの接続点の条件

を設定します。新しい設定の内容は、1 から 4 の数字で表現されます。各数字の意味は、以下のようになります。

```
0       unknown (現在の条件が不明)
1       ok (障害なし)
2       failing (回復可能な障害がある)
3       failed (回復不可能な障害がある)
4       unusable (接続点が empty)
```

オペランド

次のオペランドを指定できます。

sysctrl0:slot#
: `sysctrl` ドライバのインスタンス `0` (`sysctrl0`) が、`EXX00` システム上のボードに接続点を発行します。接続点の名前には、`slot0` から `slot15` までの番号を割り当てます。`#` の部分には `0` から `15` までの数字を入れます。この値がスロット番号を示すことになります。この形式は、cfgadm(1m) を用いた論理的な `ap_id` の指定と一致します。これに対応する物理 `ap_id` のリストは、ファイルの項目にあります。

ファイル

/usr/platform/sun4u/lib/cfgadm/sysctrl.so.1
: ハードウェア固有ライブラリ

/devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot*
: 接続点

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWkvm.u

関連項目

cfgadm(1m), cfgadm_ac(1M), ifconfig(1m), mount(1M), `pbind`(1M), `psradm`(1M), `psrinfo`(1M), `config_admin`(3CFGADM), `attributes`(5)

『Sun Enterprise 6x00、 5x00、 4x00、 3x00 システム Dynamic Reconfiguration ユーザーマニュアル』

『特記事項: Sun Enterprise 6x00、 5x00、 4x00、 3x00 システム』

注意事項

EXX00 システムの CPU/メモリーボードの動的再構成について、更に詳しい情報が必要な場合は、『Sun Enterprise 6x00、 5x00、 4x00、 3x00 システム Dynamic Reconfiguration ユーザーマニュアル』を参照してください。

名前　coreadm – コアファイルの管理

形式

```
coreadm  [-g pattern] [-G content] [-i pattern] [-I content]
 [-d option]... [-e option]...

coreadm  [-p pattern] [-P content] [pid]...

coreadm  -u
```

機能説明

coreadm コマンドを使用して、異常終了するプロセスにより生成されるコアファイルの名前と場所を指定できます。core(4) を参照してください。

sys_admin 特権を持つユーザーだけが、「形式」の最初の形式を実行できます。この形式では、大域的なコアファイル名のパターンおよび init(1M) プロセスのコアファイル名のパターンを含む、システム全体のコアファイルオプションを構成します。すべての設定は、ブート時に設定される coreadm の構成ファイル /etc/coreadm.conf に保存されます。init(1M) を参照してください。

非特権ユーザーは、「形式」の 2 番目の形式を実行できます。この形式では、オペレーティングシステムがプロセスごとのコアファイルを生成するときに使用する、ファイル名のパターンとコアファイルのコンテンツを指定します。

sys_admin 特権を持つユーザーのみが「形式」の 3 番目の形式を実行できます。この形式では、/etc/coreadm.conf の内容に基づいて、システム全体のすべてのコアファイルオプションを更新します。通常、このオプションは、リブート時に svc:/system/coreadm:default を起動するときに使用されます。

コアファイル名のパターンは、% から始まる文字で指定される組み込み変数を含んだ、通常のファイルシステムのパス名です。この変数は、オペレーティングシステムがコアファイルを生成するときに有効な値から展開されます。使用可能な組み込み変数は次のとおりです。

%d　実行ファイルのディレクトリ名。最大文字数は MAXPATHLEN

%f　実行ファイルの名前。最大文字数は MAXCOMLEN

%g　実効グループ ID

%m　マシン名 (uname -m)

%n　システムノード名 (uname -n)

%p　プロセス ID

%t　time(2) の 10 進数の値

%u　実効ユーザー ID

%z　プロセスが実行されているゾーン名 (zonename)

%%　リテラル %

たとえば、コマンドが foo でプロセス ID が 1234 の場合、コアファイル名のパターン /var/core/core.%f.%p は /var/core/core.foo.1234 になります。

コアファイルコンテンツの記述は、プロセスのバイナリイメージを識別する一連のトークンを使用して指定されます。

anon　メインスレッドスタックではないスレッドスタックを含む、匿名プライベートマッピング

ctf　ロードされたオブジェクトファイルの CTF タイプ情報セクション

data　書き込み可能プライベートファイルマッピング

dism　DISM マッピング

heap　プロセスヒープ

ism　ISM マッピング

rodata　読み取り専用プライベートファイルマッピング

shanon　匿名共有マッピング

shfile　ファイルによってバックアップされる共有マッピング

shm　System V 共有メモリー

stack　プロセススタック

symtab　ロードされたオブジェクトファイルのシンボルテーブルセクション

text　読み取り可能および実行可能なプライベートファイルマッピング

また、トークン all を使用すると、コアファイルにプロセスのバイナリイメージのすべての部分を含めるよう指定できます。トークン none を使用すると、マッピングをまったく含めないよう指定できます。default トークンを使用すると、システムのデフォルトコンテンツ (stack+heap+shm+ism+dism+text+data+rodata+anon+shanon+ctf) を含めるよう指定できます。/proc ファイルシステムデータ構造は、マッピングコンテンツに関係なく、常にコアファイル内に存在します。

+ および - を使用して、トークンを連結できます。たとえば、コアファイルコンテンツ default-ism は、詳細共有メモリーマッピングを除くマッピングのデフォルトセットを使用してコアファイルを生成します。

引数なしの coreadm コマンドは、現在のシステム構成を報告します。たとえば、次のようになります。

```
$ coreadm
    global core file pattern: /var/core/core.%f.%p
    global core file content: all
      init core file pattern: core
```

```
      init core file content: default
           global core dumps: enabled
      per-process core dumps: enabled
     global setid core dumps: enabled
per-process setid core dumps: disabled
    global core dump logging: disabled
```

プロセス ID のリストのみを指定した coreadm コマンドは、各プロセスにおけるプロセスごとのコアファイル名のパターンを報告します。たとえば、次のようになります。

```
$ coreadm 278 5678
  278:   core.%f.%p default
  5678:  /home/george/cores/%f.%p.%t all-ism
```

プロセスの所有者または proc_owner 特権を持つユーザーのみがこの方法でプロセスを調べることができます。

プロセスがコアをダンプしている時、次のように最大で 3 つのコアファイルを生成できます。1 つはプロセスごとの場所、1 つはシステム全体の大域的な場所、さらに、プロセスが局所 (非大域) ゾーンで実行されていた場合は、1 つをそのプロセスが実行されていたゾーンの大域的な場所に生成できます。各コアファイルは、対応する場所に有効なオプションに従って生成されます。

生成時に、大域コアファイルはモード 600 で作成され、スーパーユーザーによって所有されます。非特権ユーザーはそのようなファイルを調べることができません。

通常のプロセスごとのコアファイルは、プロセスの資格に基づいてモード 600 で作成されます。プロセスの所有者はそのようなファイルを調べることができます。

setuid か setgid であるプロセス、またはこのプロセスの最後の exec(2) 以降に setuid か setgid だったことのあるプロセスには、コアのダンプに関連するセキュリティーの問題が存在します。同様に、最初にスーパーユーザー特権を持っていて、setuid(2) によってそれらの特権を失ったプロセスにも、コアのダンプに関連するセキュリティーの問題が存在します。どちらのタイプのプロセスにも、そのアドレス空間の中に、現在の非特権プロセス所有者がアクセスできるべきではない機密情報が含まれている可能性があります。setid コアファイルが有効な場合、それらはモード 600 で作成され、スーパーユーザーによって所有されます。

オプション　サポートしているオプションは、以下のとおりです。

-d *option...*　指定したコアファイルオプションを使用不可にします。指定可能なオプションについては、-e *option* を参照してください。

複数の -e および -d オプションをコマンド行で指定できます。sys_admin 特権を持つユーザーのみがこのオプションを使用できます。

-e *option*... 指定したコアファイルオプションを使用可能にします。*option* には次のいずれかを指定します。

global	大域コアパターンを使用するコアダンプを許可します。
global-setid	大域コアパターンを使用する set-id コアダンプを許可します。
log	大域コアファイルの生成が試行されるときに `syslog`(3C) メッセージを生成します。
process	プロセスごとのコアパターンを使用するコアダンプを許可します。
proc-setid	プロセスごとのコアパターンを使用する set-id コアダンプを許可します。

複数の `-e` および `-d` オプションをコマンド行で指定できます。`sys_admin` 特権を持つユーザーのみがこのオプションを使用できます。

-g *pattern* 大域コアファイル名のパターンを *pattern* に設定します。パターンは `/` から始める必要があり、「機能説明」で説明されている特殊な `%` 変数のいずれも含めることができます。

`sys_admin` 特権を持つユーザーのみがこのオプションを使用できます。

-G *content* 大域コアファイルコンテンツを *content* に設定します。コンテンツは、「機能説明」で説明されているトークンを使用して指定する必要があります。

`sys_admin` 特権を持つユーザーのみがこのオプションを使用できます。

-i *pattern* デフォルトのプロセスごとのコアファイル名を *pattern* に設定します。プロセスごとのパターンがまだデフォルトに設定されているプロセスは、これによってプロセスごとのパターンが変更されます。プロセスごとのパターンが設定されているプロセス、またはプロセスごとのパターンが設定されているプロセスの子孫のプロセス (`-p` オプションを使用) は、影響を受けません。このデフォルトは、リブート後も持続します。

`sys_admin` または `proc_owner` 特権を持つユーザーのみがこのオプションを使用できます。

-I *content* デフォルトのプロセスごとのコアファイルコンテンツを *content* に設定します。プロセスごとのコンテンツがまだデフォルトに設定されて

いるプロセスは、これによってプロセスごとのコンテンツが変更されます。プロセスごとのコンテンツが設定されているプロセス、またはプロセスごとのコンテンツが設定されているプロセスの子孫のプロセス (-P オプションを使用) は、影響を受けません。このデフォルトは、リブート後も持続します。

sys_admin または proc_owner 特権を持つユーザーのみがこのオプションを使用できます。

-p *pattern* 指定した各プロセス ID の、プロセスごとのコアファイル名のパターンを *pattern* に設定します。パターンには、「機能説明」で説明されている特殊な % 変数を含めることができます。/ から始める必要はありません。パターンが / から始まっていない場合は、プロセスがコアファイルを生成したときのカレントディレクトリから相対的に判断されます。

非特権ユーザーは、そのユーザーが所有するプロセスにのみ -p オプションを適用できます。proc_owner 特権を持つユーザーは、任意のプロセスにこのオプションを適用できます。プロセスごとのコアファイル名のパターンは、影響を受けるプロセスの将来の子プロセスに継承されます。fork(2) を参照してください。

プロセス ID が指定されていない場合、-p オプションは、親プロセス (通常、coreadm を実行したシェル) についてプロセスごとのコアファイル名のパターンを *pattern* に設定します。

-P *content* 指定した各プロセス ID のプロセスごとのコアファイルコンテンツを *content* に設定します。コンテンツは、「機能説明」で説明されているトークンを使用して指定する必要があります。

非特権ユーザーは、そのユーザーが所有するプロセスにのみ -p オプションを適用できます。proc_owner 特権を持つユーザーは、任意のプロセスにこのオプションを適用できます。プロセスごとのコアファイル名のパターンは、影響を受けるプロセスの将来の子プロセスに継承されます。fork(2) を参照してください。

プロセス ID が指定されていない場合、 -P オプションは、親プロセス (通常、coreadm を実行したシェル) についてプロセスごとのファイルコンテンツを *content* に設定します。

-u 構成ファイル /etc/coreadm.conf のコンテンツからシステム全体のコアファイルオプションを更新します。 構成ファイルが見つからない場合、または構成ファイルに無効な値が含まれている場合は、デフォルト値が代入されます。更新に続いて、構成ファイルがシステムコアファイル構成に再同期されます。

sys_admin 特権を持つユーザーのみがこのオプションを使用できます。

オペランド　次のオペランドがサポートされています。

pid　プロセス ID

使用例

例 1　コアファイル名パターンの設定

ユーザーの $HOME/.profile または $HOME/.login から実行した場合、次のコマンドはログインセッション中に実行されるすべてのプロセスのコアファイル名のパターンを設定します。

```
example$  coreadm -p core.%f.%p
```

プロセス ID が省略されているため、現在実行されているシェルでプロセスごとのコアファイル名パターンが設定され、それがすべての子プロセスに継承されます。

例 2　サブディレクトリへのユーザーのファイルのダンプ

次のコマンドでは、ユーザーのすべてのコアダンプを、システムノード名によって判別される、ホームディレクトリの corefiles サブディレクトリにダンプします。このコマンドは、多くの異なるマシンを使用し、共有ホームディレクトリを持つユーザーに役立ちます。

```
example$  coreadm -p $HOME/corefiles/%n.%f.%p 1234
```

例 3　大域コアファイルリポジトリの選別

次のコマンドでは、実行可能ファイルが /usr/bin または /usr/sbin から実行された場合にのみ大域リポジトリにコアファイルを生成するようにシステムを設定します。

```
example# mkdir -p /var/cores/usr/bin
example# mkdir -p /var/cores/usr/sbin
example# coreadm -G all -g /var/cores/%d/%f.%p.%n
```

ファイル　/etc/coreadm.conf

終了ステータス　次の終了ステータスが返されます。

0　正常終了。

1　システムコアファイル構成の取得または変更中に致命的エラーが発生しました。

2　無効なコマンド行オプションが指定されました。

属性　次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

gcore(1), svcs(1), init(1M), svcadm(1M), exec(2), fork(2), setuid(2), time(2), syslog(3C), core(4), attributes(5), smf(5)

注意事項

局所 (非大域) ゾーンでは、大域設定はそのゾーン内で実行中のプロセスに適用されます。また、大域ゾーンはすべてのゾーンで実行されるプロセスに適用されます。

大域設定とは、システムまたはゾーン全体に適用される設定のことで、必ずしも大域ゾーンで有効になる設定を意味するものではありません。

coreadm サービスはサービス管理機能 smf(5) により次のサービス識別子の下で管理されます。

```
svc:/system/coreadm:default
```

有効化、無効化、または再起動要求など、このサービスに関する管理操作は、svcadm(1M) を使用して実行できます。サービスの状態は svcs(1) コマンドを使用して照会できます。

名前　cvcd – 仮想コンソールデーモン

形式　`/platform/`*platform_name*`/cvcd`
`[-a` *auth*`] [-e` *encr*`] [-u` *esp_auth*`]`

機能説明　仮想コンソールデーモン `cvcd` は、いくつかのプラットフォーム上で提供されているネットワークコンソールをサポートするサーバープロセスです。`cvcd` デーモンはリモートホストからのネットワークコンソール接続を受け付けます (一度に 1 つのホストだけ)。コンソール入力はこの接続から読み取られ、`cvcredir`(7D) 経由で `cvc`(7D) に転送されます。

同様に、コンソール出力は、`cvcredir`(7D) から読み取られ、そのネットワークコンソール接続経由で転送されます。`cvcd` が終了すると、特定の内部ハードウェアインタフェース経由でコンソールトラフィックが自動的に経路指定し直されます。

`cvcd` デーモンは通常、システムブート時に起動されます。各ドメインがサポートする `cvcd` プロセスは、一度に 1 つだけです。

注意 – Sun Enterprise 10000 ドメイン上の `cvcd` は、構成ファイル (`/etc/ssphostname`) に基づいて、ネットワークコンソール接続が許可されたホストの名前を判定します。リモートコンソールホストの名前が変更された場合には、この構成ファイルを編集し、その変更内容を反映させる必要があります。

`cvcd` デーモンは、次に説明する各オプションを通じてソケット単位の IP セキュリティーアーキテクチャー (IPsec) をサポートします。`ipsec`(7P) を参照してください。

オプション　`cvcd` デーモンがサポートするオプションは、次のとおりです。

`-a` *auth*　IPsec 認証ヘッダー (AH) アルゴリズムを制御します。*auth* に指定できるのは、`none`、`md5`、`sha1` のいずれかです。

`-e` *encr*　IPsec ESP (Encapsulating Security Payload) 暗号化アルゴリズムを制御します。*encr* に指定できるのは、`none`、`des`、`3des` のいずれかです。

`-u` *esp_auth*　IPsec ESP (Encapsulating Security Payload) 認証アルゴリズムを制御します。*esp_auth* に指定できるのは、`none`、`md5`、`sha1` のいずれかです。

オペランド　次のオペランドがサポートされています。

platform_name　パッケージ化やコードで使用される正式な Sun プラットフォーム名。たとえば、Sun Fire 15K サーバーの場合、*platform_name* は `SUNW,Sun-Fire-15000` になります。

使用例　例 1　IPSec オプションの設定

次のコマンドは、IPsec 認証ヘッダーアルゴリズムの値を `md5` に設定します。このコマンドの結果として、`cvcd` は HMAC-MD5 認証アルゴリズムを使用するようになります。

例 1　IPSec オプションの設定　　*(続き)*

```
# svccfg -s svc:/system/cvc setprop cvc/ah_auth = "md5"
# svccfg -s svc:/system/cvc setprop cvc/esp_encr = "none"
# svccfg -s svc:/system/cvc setprop cvc/esp_auth = "none"
# svcadm refresh svc:/system/cvc
```

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
アーキテクチャー	Sun Enterprise 10000 サーバー、Sun Fire High-End システム
使用条件	SUNWcvc.u

関連項目

svcs(1), svcadm(1M), svccfg(1M), services(4), attributes(5), smf(5), cvc(7D), cvcredir(7D), ipsec(7P)

『Sun Enterprise 10000 SSP リファレンスマニュアル』

『System Management Services (SMS) リファレンスマニュアル』

注意事項

cvcd サービスは、サービス管理機能 smf(5) によって、次の障害管理リソース識別子 (FMRI) の下で管理されます。

```
svc:/system/cvc
```

有効化、無効化、再起動要求など、このサービスに対する管理アクションを実行するには、svcadm(1M)、svccfg(1M) のいずれかを使用します。サービスの状態は svcs(1) コマンドを使用して照会できます。

名前　dd – ファイルの変換とコピー

形式　`/usr/bin/dd [operand=value]...`

機能説明　dd は、指定した入力ファイルに可能な変換を行なって、指定した出力へコピーします。デフォルトでは、標準入力および標準出力が使用されます。raw 入出力装置の特性を利用するために入出力のブロックサイズで指定することが可能です。サイズは、バイト単位で指定し、数字の後に k、b、または w を付加することができ、それぞれ 1024、512、または 2 の倍数として指定します。また、数字を x で区切ることにより乗算を表すことができます。

dd は、指定された入力ブロックサイズを用いて、入力データを 1 ブロックずつ読み込みます。その後、実際に渡されたデータブロックを処理します。そのサイズは、指定されたブロックサイズより小さい場合があります。dd は指定された変換処理をブロックに対して行い、結果のデータを、指定された出力ブロックサイズに従ってブロック単位で書き出します。

cbs は、ascii、asciib、unblock、ebcdic、ebcdicb、ibm、ibmb、または block 変換が指定されている場合にかぎり使用されます。最初の 2 つの指定では、cbs 文字は変換バッファにコピーされ、任意の指定文字のマッピングが行われます。また後続の空白文字は切り捨てられ、行を送信する前に復帰改行が追加されます。残りの 3 つの指定では、復帰改行までの文字が変換バッファに読み込まれ、サイズ cbs の出力レコードを構成するために空白文字が追加されます。ASCII ファイルは復帰改行文字を含むものとします。cbs が指定されていないかまたは 0 であると、ascii、asciib、ebcdic、ebcdicb、ibm、ibmb の各オプションは入力ファイルのブロック構造を変更せずに文字セットを変換します。unblock および block の各オプションは単純なファイルコピーを行います。

終了後に、dd は全体および部分的な入出力ブロック数を報告します。

オペランド　以下のオペランドが指定できます。

`if=`*file*　入力パス名。デフォルトは標準入力です。

`of=`*file*　出力パス名。デフォルトは標準出力です。seek=*expr* 変換が指定されていないとき、conv=notrunc も指定されていなければ、コピー処理の実行前に出力ファイルは切り捨てられます。seek=expr が指定され、conv=notrunc が指定されていなければ、コピー処理の結果としては dd がシークする出力ファイル中のブロックは保持されますが、出力ファイルのその他の部分は保持されません。シークするサイズと入力ファイルのサイズの合計が出力ファイルの元のサイズより小さい場合、コピー処理により出力ファイルは小さくなります。

`ibs=`*n*　入力ブロックサイズを *n* バイトとします。デフォルト値は 512 です。

obs=*n*
: 出力ブロックサイズを *n* バイトとします。デフォルト値は 512 です。

bs=*n*
: 入力ブロックサイズと出力ブロックサイズをともに *n* バイトとします。この指定は ibs= と obs= 指定よりも優先されます。sync、noerror、notrunc 以外の変換が 1 つも指定されない場合、各入力ブロックは複数の短いブロックを 1 つにまとめる処理は行われず、それぞれ単独のブロックとして出力側にコピーされます。

cbs=*n*
: block と unblock 用の変換ブロックサイズを *n* バイトに指定します。デフォルト値は 0 です。cbs= を指定しないかまたは 0 を指定した場合、block や unblock を使うと結果は予測できません。

  このオプションは ASCII または EBCDIC 変換を指定した場合にだけ有効です。ascii と asciib オペランドを指定した場合、後続の空白文字を消去する前に文字が ASCII に変換されるという点を除き、入力処理は unblock オペランド指定時と同じです。ebcdic、ebcdicb、ibm、または ibmb オペランドを指定した場合、後方に空白文字を追加した後で文字が EBCDIC または IBM EBCDICに変換されるという点を除き、入力処理は block オペランド指定時と同じです。

files=*n*
: 終了する前に、*n* 個の入力ファイルをコピーして連結します (入力が磁気テープまたは同様な装置の場合だけ有効です)。

skip=*n*
: コピーを開始する前に、指定された入力ブロックサイズを用いて *n* 個の入力ブロックを読み飛ばします。シーク可能なファイルに対しては、システムはそれらのブロックを読み込むか、あるいはシークを行います。シーク不可能なファイルに対しては、ブロックを読み込んで、そのデータを捨てます。

iseek=*n*
: コピーを行う前に、入力ファイルの先頭から *n* 個のブロックをシークします (skip の動作が遅いディスクファイルに適しています)。

oseek=*n*
: コピーを行う前に、出力ファイルの先頭から *n* 個のブロックをシークします。

seek=*n*
: コピーを行う前に、出力ファイルの先頭から *n* 個のブロックをスキップします (指定された出力ブロックサイズを使用)。シーク不可能なファイルに対しては、既存のブロックを読み込み、現在のファイルの終わり位置から指定されたオフセット位置までの間に空白があれば、その空白

を NULL バイトで埋めます。シーク可能なファイルに対しては、指定されたオフセット位置までをシークするか、またはシーク不可能なファイルの場合と同様にブロックを読み込みます。

`count=`*n*　　*n* 個の入力ブロックだけをコピーします。

`conv=`*value*[`,`*value*...]　　1 つ以上の *value* をコンマで区切って記述します。各 *value* は以下のいずれかです。

`ascii`　　EBCDIC を ASCII に変換します。

`asciib`　　BSD 互換の文字変換を使用して、EBCDIC を ASCII に変換します。

`ebcdic`　　ASCII を EBCDIC に変換します。復帰改行のない固定長の ASCII レコードを変換する場合は、前もって `dd conv=unblock` でパイプラインを整えてください。

`ebcdicb`　　BSD 互換の文字変換を使用して、ASCII を EBCDIC に変換します。復帰改行のない固定長の ASCII レコードを変換する場合は、前もって `dd conv=unblock` でパイプラインを整えてください。

`ibm`　　ASCII から EBCDIC への変換とわずかに異なるマップを使用します。復帰改行のない固定長の ASCII レコードを変換する場合は、前もって `dd conv=unblock` でパイプラインを整えてください。

`ibmb`　　BSD 互換の文字変換を使用してASCII から EBCDIC への変換とわずかに異なるマップを使用します。復帰改行のない固定長の ASCII レコードを変換する場合は、前もって `dd conv=unblock` でパイプラインを整えてください。

`ascii`(または `asciib`)、`ebcdic`(または `ebcdicb`)、`ibm`(または `ibmb`) のうち 2 つ以上を同時に指定することはできません。

`block`　　入力データを、入力ブロック境界に関係なく、復帰改行文字もしくは EOF で終わる可変長レコードの集まりと見なします。各レコードは、変換ブロックサイズで指定した長さの固定長レコードに変換されます。入力行に復帰改行文字

があれば削除されます。変換ブロックサイズより短い行に関しては、ブロックを埋めるために空白文字が付加されます。変換ブロックサイズより長い行に関しては、ブロックサイズを満たす最大長になるように文字が捨てられます。切り捨てられた行の総数が報告されます。

`unblock` 固定長のレコードを可変長に変換します。変換ブロックサイズと等しい長さ分のバイト (それより短いデータしか残っていなければそのすべて) を読み込み、後続の空白文字を削除して、復帰改行文字を付加します。

`block` と `unblock` を同時に指定することはできません。

`lcase` `LC_CTYPE` カテゴリ中のキーワードの `tolower` で指定された大文字を、対応する小文字にマップします。マッピングを指定されなかった文字は、この変換によって変更されることはありません。

`ucase` `LC_CTYPE` カテゴリ中のキーワードの `toupper` で指定された小文字を、対応する大文字にマップします。マッピングを指定されなかった文字は、この変換によって変更されることはありません。

`lcase` と `ucase` を同時に指定することはできません。

`swab` 入力データを 2 バイトずつの対になっているものとし、各々の対についてバイトの値を交換します。入力レコードの長さが奇数バイトの場合には、最終バイトは無視されます。

`noerror` 入力エラーが発生しても処理を停止しません。入力エラーが起こると、標準エラー出力に診断メッセージと、入力および出力ブロック数が出力されます。ブロック数の形式は、正常に終了したときに出力されるものと同じです。`sync` 変換が指定されたときは、入力データのうち得られなかったバイトは NULL バイトに置き換えられて通常どおりに処理されます。`sync` が指定されなければ、入力ブロックは出力上には現れません。

`notrunc` 出力ファイルを切り捨てません。今回の `dd` 呼び出しで明示的に書き出されなかったブロックも出力ファイル中に保持します (前述の `of=`*file* オペランドの説明を参照)。

`sync` 入力ブロックに NULL バイトを付加して、サイズが `ibs=` で指定した値に等しくなるようにします。なお `block` または `unblock` も指定されている場合には、NULL バイトの代わりに空白文字を付加します。

`conv=` 以外のオペランドが複数回指定されたときは、最後に記述された `operand=`*value* が有効となります。

`bs=`、`cbs=`、`ibs=`、`obs=` に関しては、バイト単位でサイズを指定する式をアプリケーション側で提供しなければなりません。式 *expr* は以下のいずれかです。

1. 正の 10 進数。
2. 正の 10 進数の後に 1024 倍を示す `k` を付加したもの。
3. 正の 10 進数の後に 512 倍を示す `b` を付加したもの。
4. 2 つ以上の正の 10 進数 (`k` や `b` を付加してもしなくてもよい) を文字 `x` で区切ったもの。その 2 つの値を乗算した結果を表す。

オペランドはすべて、入力データを読み込む前に処理されます。

使用法

ファイルが 2G バイト (2^{31} バイト) 以上ある場合の `dd` の動作については、`largefile`(5) を参照してください。

使用例

例 1 テープドライブ 0 から 1 へのコピー

次のコマンドは、標準的なデバイス命名規約を用いて、テープドライブ 0 から 1 へコピーします。

```
example% dd if=/dev/rmt/0h  of=/dev/rmt/1h
```

例 2 標準入力の先頭の 10 バイトの削除

次のコマンドは、標準入力の先頭の 10 バイトを削除します。

```
example% dd ibs=10  skip=1
```

例 3 テープを ASCII ファイルに読み込む

次のコマンドは、テープブロックごとに 10 個の 80 バイト EBCDIC カードイメージにブロック化した EBCDIC テープを ASCII ファイル `x` に読み込みます。

```
example% dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

例 4　テープの書き込みに conv=sync を使用

次のコマンドは、テープに書き込む場合に conv=sync を使用する例です。

```
example% tar cvf - . | compress | dd obs=1024k of=/dev/rmt/0 conv=sync
```

環境

dd の実行に影響を与える環境変数 LC_CTYPE、LC_MESSAGES、NLSPATH についての詳細は、environ(5) を参照してください。

終了ステータス

以下の終了ステータスが返されます。

0　　ファイルは正常にコピーされた

>0　　エラーが発生した

noerror 変換が指定されていないときに入力エラーが発生すると、部分的に生成された出力ブロックがあればそれを出力ファイルに書き出し、診断メッセージを出力し、コピー処理を中止します。その他のエラーを検出した場合には、診断メッセージを出力し、コピー処理を中止します。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

cp(1), sed(1), tr(1), attributes(5), environ(5), largefile(5)

診断

f+p records in(out)　　読み取られた (書き込まれた) 完全 (f) および部分 (p) ブロック数

注意事項

ブロックサイズが異なるファイルシステム間でファイルのコピーを行う場合には、dd を使用しないでください。

ファイルをコピーするのにブロック型デバイスを使用すると、最終ブロックをブロック境界にあわせるために余分な NULL バイトがファイルに追加されます。

dd が ibs=X および obs=Y オペランドを用いてパイプから読み取ると、その出力はつねにサイズ Y の固まりでブロック化されます。bs=Z が使用されると、出力ブロックはその時点でパイプから読み取ることができる大きさになります。

dd を使用してテープデバイスにファイルをコピーする場合、ファイルのサイズはデバイスのセクタサイズ (たとえば 512K) の倍数でなければなりません。任意のサイズのファイルをテープデバイスにコピーする場合は、tar(1) または cpio(1) を使用してください。

SIGINT が発生した場合、dd はステータス情報を標準エラー出力に書き出して処理を終了します。その他のシグナル発生時には、標準的な動作を行います。

名前　　df – 使用可能なディスクブロックおよび使用可能なファイル数の表示

形式

```
/usr/bin/df [-F FSType] [-abeghklntVvZ]
     [-o FSType-specific_options]
     [block_device | directory | file | resource ...]

/usr/xpg4/bin/df [-F FSType] [-abeghklnPtVZ]
     [-o FSType-specific_options]
     [block_device | directory | file | resource ...]
```

機能説明

df コマンドは、マウントされているもしくはマウントされていないファイルシステムが使用しているディスク容量を表示します。 さらに、使用中の容量、使用可能な容量、ファイルシステムの全容量のうちどの程度が使用されたかを表示します。対象のファイルシステムとしては、デバイス、または特定のファイルシステム上のファイルあるいはディレクトリを指定します。

オペランドもオプションも指定せずに df を実行すると、全ファイルシステムに関する情報が出力されます。

df はすべての *FSTypes* (ファイルシステムタイプ) をサポートしません。

自動マウンタがまだマウントしていないネットワーク上のマウントポイントで df を実行する場合、ファイルシステムサイズの情報がゼロとして出力されます。自動マウンタがファイルシステムをマウントすると、サイズの情報は正確に出力されるようになります。

オプション　　次のオプションは、/usr/bin/df と /usr/xpg4/bin/df で指定できます。

-a　　/etc/mnttab のエントリ (mnttab(4) を参照) に ignore オプションセットを含むファイルシステムについても情報を出力します。

-b　　使用可能な容量の合計を、K バイト単位で報告します。

-e　　使用可能なファイル数だけを報告します。

-F *FSType*　　df コマンドが動作する *FSType* を指定します。-F オプションはマウントされていないファイルシステムを使用するためのオプションです。*FSType* は、このオプションで指定するか、あるいは /etc/vfstab 内の記述 (vfstab(4) を参照) から特定できるようにしておく必要があります。後者の場合、具体的には *directory*、*block_device*、または *resource* 引数の指定値とテーブル中のエントリの値を一致させるか、あるいは /etc/default/fs を参照して決定されます。詳細は default_fs(4) を参照してください。

`-g` `statvfs`(2) 構造体全体を報告します。このオプションはマウントされているファイルシステムに対してのみ有効です。`-o` オプションと同時に指定することはできません。このオプションは、`-b`、`-e`、`-k`、`-n`、`-P`、`-t` の各オプション (指定されている場合) よりも優先されます。

`-h` `-k` と似ていますが、サイズ情報がより読みやすい形式で表示されます。1 つのファイルシステムにつき 1 行の情報が出力されます。情報の内容は、ファイルシステム名、そのファイルシステムに割り当てられている容量の合計、既存のファイルに割り当てられている容量の合計、特権を持たないユーザーが新たなファイルを生成する場合に使用できる容量の合計、そのファイルシステム上の全ファイルに現在割り当てられている通常使用可能な容量の割合 (パーセント単位)です。すべてのサイズを縮小して、読みやすい形式で出力します。たとえば、`14K`、`234M`、`2.7G`、`3.0T` などのようになります。縮尺は、`1024` を除数として行われます。

このオプションは `-b`、`-e`、`-g`、`-k`、`-n`、`-t`、および `-V` オプションを無効にします。このオプションはマウントされているファイルシステム上でのみ有効であり、`-o` オプションと一緒には使用できません。

`-k` 割り当てられているディスク容量を K バイト単位で出力します。1 つのファイルシステムにつき 1 行の情報が出力されます。情報の内容は、ファイルシステム名、そのファイルシステムに割り当てられている容量の合計、既存のファイルに割り当てられている容量の合計、特権を持たないユーザーが新たなファイルを生成する場合に使用できる容量の合計、そのファイルシステム上の全ファイルに現在割り当てられている通常使用可能な容量の割合 (パーセント単位)です。このオプションは、`-b`、`-e`、`-n`、`-t` の各オプション (指定されている場合) よりも優先されます。

`-l` ローカルファイルシステムについての情報だけを報告します。このオプションはマウントされているファイルシステムに対してのみ有効です。`-o` オプションと同時に指定することはできません。

`-n` *FSType* の名前だけを報告します。オペランド指定を省略すると、このオプションはマウントされているファイルシステムタイプの一覧を出力します。このオ

プションはマウントされているファイルシステムに対してのみ有効です。-o オプションと同時に指定することはできません。

-o *FSType-specific_options*　*FSType* 固有のオプションを指定します。オプションとオプションとの間はコンマだけで区切り、空白は入れないでください。詳細については *FSType* コマンド用のマニュアルページを参照してください。

-t　合計値を含む完全なリストを出力します。このオプションは、-b、-e、-n の各オプション (指定されている場合) よりも優先されます。

-V　指定されたコマンド行の全内容のエコーだけを行い、コマンド自体は実行しません。コマンド行の内容としては、ユーザーが指定したオプションやオペランドに加え、/etc/mnttab、/etc/vfstab、/etc/default/fs の各ファイルから得られた情報が付加されます。このオプションは、コマンド行を確認および検証するときに使用します。

-Z　すべての可視ゾーン内にあるマウントを表示します。デフォルトでは、df は、現在のゾーン内にあるマウントだけを表示します。このオプションは、非大域ゾーンでは効果がありません。

/usr/bin/df　次のオプションは、/usr/bin/df でのみ指定できます。

-v　-k オプションと似ていますが、サイズ情報がそれぞれのファイルシステムを構成する最小ブロックサイズの倍数で表示されます。

1 つのファイルシステムにつき 1 行の情報が出力されます。1 行の情報の内容は次のとおりです。

- ファイルシステムのマウントポイント
- ファイルシステム名
- そのファイルシステムに割り当てられているブロックの合計数
- 既存のファイルに割り当てられているブロック数
- 特権を持たないユーザーが新たなファイルを生成する場合に使用できるブロック数
- ファイルに使用されているブロックの割合 (パーセント単位)

/usr/xpg4/bin/df　次のオプションは、/usr/xpg4/bin/df でのみ指定できます。

-P　-k オプションと同じですが、出力の単位は 512 バイトとなります。

オペランド　df は、次の優先度に従ってオペランドを解釈します。*block_device*、*directory*、*file* です。次のオペランドを指定できます。

block_device ブロック型特殊デバイス (たとえば dev/dsk/c1d0s7/) を指定します。対応するファイルシステムはマウントされている必要はありません。

directory 有効なディレクトリ名を指定します。df は *directory* を含むファイルシステムについて報告します。

file 有効なファイル名を指定します。df は *file* があるファイルシステムについて報告します。

resource NFS リソース名を指定します。

使用法

ファイルが 2 ギガバイト (2^{31} バイト) 以上ある場合の df の動作については、largefile(5) を参照してください。

使用例

例 1 df コマンドを実行する

以下は、df コマンドとその出力の例です。

```
example% /usr/bin/df

/                  (/dev/dsk/c0t0d0s0 ):  287530 blocks    92028 files
/system/contract   (ctfs              ):       0 blocks 2147483572 files
/system/object     (objfs             ):       0 blocks 2147483511 files
/usr               (/dev/dsk/c0t0d0s6 ): 1020214 blocks   268550 files
/proc              (/proc             ):       0 blocks      878 files
/dev/fd            (fd                ):       0 blocks        0 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/var/run           (swap              ):  396016 blocks     9375 files
/tmp               (swap              ):  396016 blocks     9375 files
/opt               (/dev/dsk/c0t0d0s5 ):  381552 blocks    96649 files
/export/home       (/dev/dsk/c0t0d0s7 ):  434364 blocks   108220 files
```

各列は、左から、マウントポイント、デバイス (または、df -k による「ファイルシステム」)、空きブロック、および空きファイルを示します。contract ファイルシステムの場合、マウントポイントは /system/contract、ファイルシステムは ctfs (SMF が使用)、空きブロックは 0、そして、空きファイルは 2147483582 (INTMAX-1) です。object ファイルシステムの場合、マウントポイントは /system/object、ファイルシステムは objfs (objfs(7FS) を参照)、空きブロックは 0、そして、空きファイルは 2147483511 です。

例 2 ファイルシステム /usr に関する情報を出力する

次の例は、ファイルシステム /usr に関する情報を出力します。

```
example% /usr/xpg4/bin/df -P /usr
```

例 3　ファイルシステム /usr に関する情報を出力する (/usr/src が /usr ファイルシステムの一部である場合)

/usr/src が /usr ファイルシステムの一部である場合、 次の例も上記の例と同じ結果を出力します。

```
example% /usr/xpg4/bin/df -P /usr/src
```

例 4　すべての ufs ファイルシステムの i ノード使用率を表示する

次の例は、すべての ufs ファイルシステムにおける i ノードの使用率を表示します。

```
example%/usr/bin/df -F ufs -o i
```

環境

SYSV3　この環境変数はデフォルトの df の動作を無効にして、INTERACIVE UNIX システムと SCO UNIX のインストールスクリプトとの互換性を提供するために使用します。SYSV3 は互換性だけを目的とした環境変数なので、新しいスクリプトでは使用しないでください。

設定した場合、通常「ファイル」を表示するヘッダーは、すべて「ノード」を表示するようになります。df の実行に影響を与える次の環境変数についての詳細は、environ(5) を参照してください。LANG、LC_ALL、LC_CTYPE、LC_MESSAGES、および NLSPATH。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生した。

ファイル

/dev/dsk/*	ディスクデバイス
/etc/default/fs	ローカルファイルシステムタイプのデフォルト値。デフォルト値は、/etc/default/fs 内で次のように設定されています。たとえば、次のように指定します。FSType が指定されていない場合に LOCAL がコマンドに対するデフォルトパーティションの場合は、LOCAL=ufs と指定します。
/etc/mnttab	マウントテーブル
/etc/vfstab	各ファイルシステム用のデフォルトパラメータ

属性

次の属性については attributes(5) のマニュアルページを参照してください。

/usr/bin/df

属性タイプ	属性値
使用条件	SUNWcsu

/usr/xpg4/bin/df

属性タイプ	属性値
使用条件	SUNWxcu4
インタフェースの安定性	標準

関連項目

find(1), df_ufs(1M), mount(1M), statvfs(2), default_fs(4), mnttab(4), vfstab(4), attributes(5), environ(5), largefile(5), standards(5), objfs(7FS)

注意事項

ファイルシステムで UFS ログが有効になっている場合、ログで使用されるディスク容量は df の報告に反映されます。ログは、 ファイルシステムの空きブロックから割り当てられ、ファイルシステム 1G バイト当り 約 1M バイトから最大約 64M バイトのサイズになります。

名前

dtrace – DTrace 動的トレースコンパイラおよびトレースユーティリティー

形式

```
dtrace  [-32 |  -64] [-aACeFGHhlqSvVwZ] [-b bufsz] [-c cmd]
 [-D name [=value]] [-I path] [-L path] [-o output] [-s script]
 [-U name] [-x arg [=val]] [-Xa |  c |  s |  t] [-p pid]
 [-P provider [[predicate] action]] [-m [provider:] module [[predicate] action]]
 [-f [[provider:] module:] function [[predicate] action]]
 [-n [[[provider:] module:] function:] name [[predicate] action]]
 [-i probe-id [[predicate] action]]
```

機能説明

DTrace は、Solaris オペレーティングシステム用の総合的な動的トレースフレームワークです。DTrace が提供する強力なインフラストラクチャーを使えば、管理者、開発者、およびサービス担当者は、オペレーティングシステムやユーザープログラムの動作に関するさまざまな質問に対して、簡潔に答えることができるようになります。

『Solaris 動的トレースガイド』では、DTrace を使ってシステム動作を監視、デバッグ、およびチューニングする方法について説明しています。バンドルされている DTrace の監視ツール、計測プロバイダ、D プログラミング言語など、DTrace の各種機能の詳細については、このマニュアルを参照してください。

`dtrace` コマンドは、DTrace 機能が提供する基本サービスに対する汎用インタフェースを提供します。それらを次に示します。

- DTrace によって現在公開されている一連のプローブやプロバイダを一覧表示するためのオプション
- 任意のプローブ記述指定子 (provider、module、function、name) を使ってプローブを直接有効にするためのオプション
- D コンパイラを実行し、1 つ以上の D プログラムファイルまたはコマンド行に直接記述されたプログラムをコンパイルするためのオプション
- 匿名トレースプログラムを生成するためのオプション
- プログラム安定性レポートを生成するためのオプション
- DTrace のトレース動作やバッファリング動作を変更したり、追加の D コンパイラ機能を有効化したりするためのオプション

`dtrace` を使えば、D スクリプトを作成できます。それには、`#!` 宣言内にこの `dtrace` を指定してインタプリタファイルを作成します。また、`dtrace` を使えば、実際に `-e` オプションを使用してトレースを有効化することなく D プログラムのコンパイルを試行し、そのプロパティーを決定することもできます。各オプションを参照してください。`dtrace` ユーティリティーを使ってこれらのタスクを実行する方法に関する詳しい例については、『Solaris 動的トレースガイド』を参照してください。

オプション

`-P`、`-m`、`-f`、`-n`、および `-i` オプションが受け入れる引数には、1 つの D 言語 *predicate* をスラッシュ `//` で囲んだものと、一連の D 言語 *action* 文を中括弧 `{}` で囲んだものを、オプションで含めることができます。コマンド行から指定した D プロ

グラムのコードは、適切に引用符で囲む必要があります。そうしないと、メタ文字がシェルによって解釈されてしまう可能性があります。

サポートしているオプションは、次のとおりです。

-32 | -64
D コンパイラは、オペレーティングシステムカーネルのネイティブデータモデルを使ってプログラムを生成します。isainfo -b コマンドを使えば、現在のオペレーティングシステムのデータモデルを確認できます。-32 オプションが指定された場合、dtrace は、32 ビットデータモデルを使って D プログラムをコンパイルするよう、D コンパイラに指示します。-64 オプションが指定された場合、dtrace は、64 ビットデータモデルを使って D プログラムをコンパイルするよう、D コンパイラに指示します。これらのオプションは通常の場合、必要ありません。なぜなら、dtrace は、ネイティブデータモデルをデフォルトとして選択するからです。データモデルは整数型のサイズおよびほかの言語のプロパティーに影響を与えます。いずれかのデータモデル用にコンパイルされた D プログラムは、32 ビットカーネル上でも、64 ビットカーネル上でも実行できます。また、-G オプションによって生成される ELF ファイルの形式が ELF32、ELF64 のどちらになるかも、この -32 オプションおよび -64 オプションによって決まります。

-a
匿名トレース状態を要求し、トレースされたデータを表示します。この -a オプションを -e オプションと組み合わせれば、匿名トレース状態を消費したあと、別のデータを待ち続ける代わりにすみやかに終了するように、dtrace に指示することができます。匿名トレースの詳細については、『Solaris 動的トレースガイド』を参照してください。

-A
匿名トレース用の driver.conf(4) 指令を生成します。このオプションは、指定されたプローブが匿名トレース用に有効化されてから終了するまでの一連の dtrace(7D) 構成ファイル指令を作成します。dtrace はデフォルトで、ファイル /kernel/drv/dtrace.conf に指令を格納しようとします。-o オプションを使って別の出力ファイルを指定すれば、この動作を変更できます。

-b *bufsz*
主トレースバッファーサイズ (*bufsz*) を設定します。このトレースバッファーサイズには、サイズサフィックス k、m、g、t のいずれかを含めることができます。dtrace は、バッファー領域の割り当てに失敗すると、バッファーサイズを減らそうとするか、終了します。どちらになるかは、bufresize プロパティーの設定によります。

-c *cmd*
指定されたコマンド *cmd* を実行し、その実行が完了したら終了します。コマンド行に -c オプションが複数指定されている場合、dtrace は、すべてのコマンドの終了後に終了しますが、それぞれの子プロセスが終了するたびにその終了ステータスを報告します。最初のコマンドのプロセス ID は $target マクロ変数を

介して、コマンド行から指定されたか -s オプションを使って指定されたすべての D プログラムから利用可能になります。マクロ変数の詳細については、『Solaris 動的トレースガイド』を参照してください。

-C
D プログラムをコンパイルする前に、それらのプログラムに対して C プリプロセッサ cpp(1) を実行します。-D、-U、-I、-H の各オプションを使えば、C プリプロセッサにオプションを渡すことができます。-X オプションを使えば、C 標準への準拠レベルを選択できます。C プリプロセッサの呼び出し時に D コンパイラによって定義される一連のトークンについては、-X を参照してください。

-D *name* [=*value*]
-C オプションで有効化された cpp(1) を呼び出す際に、*name* を定義します。等号 (=) と追加の *value* を指定した場合、名前に対応する値が代入されます。このオプションは、cpp が呼び出されるたびに、それに -D オプションを渡します。

-e
すべての要求をコンパイルし、匿名トレース状態を消費し終わったあと (-a オプション)、プローブを有効化する前に終了します。このオプションを -a オプションと組み合わせれば、匿名トレースデータを出力してから終了させることができます。このオプションを D コンパイラのオプションと組み合わせることもできます。この組み合わせを使用した場合、プログラムがコンパイルされるかどうかの検証は行われますが、それらのプログラムが実際に実行されたり対応する計測機能が有効化されたりすることはありません。

-f [[*provider*:]*module*:]*function*[[*predicate*]*action*]]
トレースまたは一覧表示 (-l オプション) の対象となる関数名を指定します。対応する引数には、プローブ記述形式 *provider:module:function*、*module:function*、*function* のいずれかを含めることができます。指定されなかったプローブ記述フィールドは空のままとなり、そのフィールドの値に関係なくすべてのプローブに一致します。*function* 以外の修飾子が記述内に指定されなかった場合、その対応する *function* を持つすべてのプローブに一致します。-f の引数の後にはオプションで、D プローブ節を付加することもできます。-f オプションは、コマンド行で一度に複数指定できます。

-F
関数の開始 (entry) と終了 (return) を識別することにより、トレース出力をひとつにまとめます。関数開始プローブのレポートはインデントされ、-> の後ろに出力されます。関数終了プローブのレポートはインデント解除され、<- の後ろに出力されます。システムコール開始プローブのレポートはインデントされ、=> の後ろに出力されます。システムコール終了プローブのレポートはインデント解除され、<= の後ろに出力されます。

-G
埋め込まれた DTrace プログラムを含む ELF ファイルを生成します。プログラム内に指定された DTrace プローブは、ELF オブジェクトの内側に保存されます。このオブジェクトは再配置可能であり、別のプログラムにリンクできます。-o

オプションが存在している場合、ELF ファイルはこのオペランドの引数として指定されたパス名を使って保存されます。-o オプションが存在しておらず、かつ *filename*.d という名前のファイルに DTrace プログラムが格納されている場合、ELF ファイルは、*filename*.o という名前で保存されます。それ以外の場合、ELF ファイルは名前 d.out として保存されます。

-H
-C オプションで有効化された cpp(1) を呼び出す際に、インクルードされたファイルのパス名を出力します。このオプションは、cpp が呼び出されるたびに、それに -H オプションを渡します。その結果、一連のパス名が 1 行に 1 つずつ、標準エラー出力に出力されます。

-h
指定したプロバイダ定義内のプローブに対応するマクロを格納するヘッダーファイルを生成します。このオプションは、あとで -G オプションとともに使用するほかのソースファイルによってインクルードされるヘッダーファイルを生成する場合に使用してください。-o オプションが存在している場合、ヘッダーファイルはこのオプションの引数として指定されたパス名を使って保存されます。-o オプションが存在せず、かつ DTrace プログラムが *filename*.d というファイルに格納されている場合、ヘッダーファイルは名前 *filename*.h として保存されます。

-i *probe-id*[[*predicate*] *action*]
トレースまたは一覧表示 (-l オプション) の対象となるプローブ ID (*probe-id*) を指定します。プローブ ID を指定する際には、dtrace -l で表示される 10 進整数を使用します。-i の引数の後にはオプションで、D プローブ節を付加することもできます。-i オプションは一度に複数指定できます。

-I *path*
-C オプションで有効化された cpp(1) を呼び出す際に、指定されたディレクトリ *path* を #include ファイルの検索パスに追加します。このオプションは、cpp が呼び出されるたびに、それに -I オプションを渡します。指定された *path* は、検索パス内のデフォルトディレクトリリストの前に挿入されます。

-L *path*
指定されたディレクトリ *path* を、DTrace ライブラリの検索パスに追加します。DTrace ライブラリは、D プログラムを記述する際に使用可能な共通定義を格納する目的で使用されます。指定された *path* は、デフォルトライブラリ検索パスの後に追加されます。

-l
プローブを有効化しないで一覧表示します。-l オプションが指定された場合、dtrace は、-P、-m、-f、-n、-i、-s の各オプションを使って指定された記述に一致するプローブのレポートを生成します。これらのオプションが 1 つも指定されなかった場合、このオプションはすべてのプローブを一覧表示します。

-m [[*provider:*] *module:* [[*predicate*] *action*]]
トレースまたは一覧表示 (-l オプション) の対象となるモジュール名を指定します。対応する引数には、プローブ記述形式 *provider:module*、*module* のいずれかを含めることができます。指定されなかったプローブ記述フィールドは空のままとなり、そのフィールドの値に関係なくすべてのプローブに一致します。*module* 以外の修飾子が記述内に指定されなかった場合、その対応する *module* を持つすべてのプローブに一致します。-m の引数の後にはオプションで、D プローブ節を付加することもできます。-m オプションは、コマンド行で一度に複数指定できます。

-n [[[*provider:*] *module:*] *function:*] *name* [[*predicate*] *action*]
トレースまたは一覧表示 (-l オプション) の対象となるプローブ名を指定します。対応する引数には、プローブ記述形式 *provider:module:function:name*、*module:function:name*、*function:name*、*name* のいずれかを含めることができます。指定されなかったプローブ記述フィールドは空のままとなり、そのフィールドの値に関係なくすべてのプローブに一致します。*name* 以外の修飾子が記述内に指定されなかった場合、その対応する *name* を持つすべてのプローブに一致します。-n の引数の後にはオプションで、D プローブ節を付加することもできます。-n オプションは、コマンド行で一度に複数指定できます。

-o *output*
-A、-G、および -l オプションまたはトレースデータ自体に対する *output* ファイルを指定します。-A オプションが存在していて -o が存在していない場合、デフォルトの出力ファイルは /kernel/drv/dtrace.conf になります。-G オプションが存在していて -s オプションの引数が *filename*.d の形式であり、かつ -o が存在していない場合、デフォルトの出力ファイルは *filename*.o になります。それ以外の場合、デフォルトの出力ファイルは d.out になります。

-p *pid*
指定されたプロセス ID *pid* を獲得し、そのシンボルテーブルをキャッシュし、その実行が完了したら終了します。コマンド行に -p オプションが複数指定されている場合、dtrace は、すべてのコマンドの終了後に終了しますが、それぞれの子プロセスが終了するたびにその終了ステータスを報告します。最初のプロセス ID は $target マクロ変数を介して、コマンド行から指定されたか -s オプションを使って指定されたすべての D プログラムから利用可能になります。マクロ変数の詳細については、『Solaris 動的トレースガイド』を参照してください。

-P *provider* [[*predicate*] *action*]
トレースまたは一覧表示 (-l オプション) の対象となるプロバイダ名を指定します。残りのプローブ記述フィールド module、function、および name は空のままとなり、そのフィールドの値に関係なくすべてのプローブに一致します。-P の引数の後にはオプションで、D プローブ節を付加することもできます。-P オプションは、コマンド行で一度に複数指定できます。

-q
非出力モードを設定します。dtrace は、指定されたオプションや D プログラムの条件に一致するプローブ数などのメッセージを抑制し、列ヘッダーの CPU ID

や プローブ ID を出力しなくなるほか、出力に改行を挿入しなくなります。`trace()` や `printf()` といった D プログラム文によってトレースおよび書式設定されたデータのみが、標準出力に表示されます。

`-s`
指定された D プログラムのソースファイルをコンパイルします。`-e` オプションが存在している場合、プログラムのコンパイルは行われますが、計測機能は有効化されません。`-l` オプションが存在している場合、プログラムのコンパイルと一致するプローブの一覧表示は行われますが、計測機能は有効化されません。`-e`、`-l`、`-G`、`-A` のいずれも存在しない場合には、D プログラムによって指定された計測機能が有効化され、トレースが開始されます。

`-S`
D コンパイラの中間コードを表示します。D コンパイラは、個々の D プログラムごとに生成された中間コードに関するレポートを生成し、標準エラー出力に出力します。

`-U` *name*
`-C` オプションで有効化された cpp(1) を呼び出す際に、指定された *name* を未定義にします。このオプションは、`cpp` が呼び出されるたびに、それに `-U` オプションを渡します。

`-v`
冗長モードを設定します。`-v` オプションが指定された場合、`dtrace` は、プログラム安定性レポートを生成します。このレポートでは、指定された D プログラムの最小インタフェースの安定性レベルと依存性レベルが示されます。DTrace の安定性レベルの詳細については、『Solaris 動的トレースガイド』を参照してください。

`-V`
`dtrace` がサポートする D プログラミングインタフェースのバージョンのうち、もっとも高いバージョンを報告します。このバージョン情報が標準出力に出力されたあと、`dtrace` コマンドは終了します。DTrace のバージョン管理機能の詳細については、『Solaris 動的トレースガイド』を参照してください。

`-w`
`-s`、`-P`、`-m`、`-f`、`-n`、`-i` のいずれかのオプションで指定された D プログラム内で、破壊アクションを使用できるようにします。`-w` オプションが指定されなかった場合、`dtrace` は、破壊アクションを含む D プログラムのコンパイルや有効化を許可しません。

`-x` *arg* [=*val*]
DTrace の実行時オプションまたは D コンパイラオプションを有効化または変更します。オプションの一覧については、『Solaris 動的トレースガイド』を参照してください。ブール型のオプションを有効化するには、その名前を指定します。値を持つオプションを設定するには、オプションの名前と値を等号 (=) で分離します。

`-X a | c | s | t`

`-C` オプションで有効化された `cpp`(1) を呼び出す際に選択すべき ISO C 標準への準拠レベルを指定します。`-X` オプションの引数は、`__STDC__` マクロの値やその存在有無に影響を与えますが、具体的にどのような影響を与えるかは、この引数の文字の値によります。

`-X` オプションでサポートされる引数は、次のとおりです。

`a` デフォルト。ISO C + K&R 互換性拡張。ISO C が必要とするセマンティック変更を含みます。`-X` が指定されなかった場合、これがデフォルトのモードになります。`cpp` 呼び出し時に `-Xa` オプションが指定された場合、定義済みマクロ `__STDC__` の値は 0 になります。

`c` 準拠。ISO C に厳密に準拠。K&R C 互換性拡張は含みません。`cpp` 呼び出し時に `-Xc` オプションが指定された場合、定義済みマクロ `__STDC__` の値は 1 になります。

`s` K&R C のみ。`cpp` 呼び出し時に `-Xs` オプションが指定された場合、マクロ `__STDC__` は未定義になります。

`t` 移行。ISO C + K&R C 互換性拡張。ISO C が必要とするセマンティック変更は含みません。`cpp` 呼び出し時に `-Xt` オプションが指定された場合、定義済みマクロ `__STDC__` の値は 0 になります。

`-X` オプションは D コンパイラが C プリプロセッサを呼び出す方法にのみ影響を与えるため、`-Xa` オプションと `-Xt` オプションは、D から見れば等価です。それでもその両方が提供されているのは、C 構築環境の設定を再利用しやすくするためです。

次の追加 C プリプロセッサ定義は、`-X` モードの内容にかかわらず、すべてのモードで常に指定され、有効になります。

- `__sun`
- `__unix`
- `__SVR4`
- `__sparc` (SPARC システム上でのみ)
- `__sparcv9` (SPARC システム上で 64 ビットプログラムをコンパイルする場合のみ)
- `__i386` (x86 システム上で 32 ビットプログラムをコンパイルする場合のみ)
- `__amd64` (x86 システム上で 64 ビットプログラムをコンパイルする場合のみ)
- `__`*‘uname -s‘*`_`*‘uname -r‘* (例: `__SunOS_5_10`)
- `__SUNW_D=1`
- `__SUNW_D_VERSION=0x`*MMmmmuuu*

ここで、*MM* は 16 進のメジャーリリース値、*mmm* は 16 進のマイナーリリース値、*uuu* は 16 進のマイクロリリース値です。DTrace のバージョン管理の詳細については、『Solaris 動的トレースガイド』を参照してください。

`-Z`
一致するプローブが 1 つも見つからないようなプローブ記述を許可します。`-Z` オプションが指定されない場合、D プログラムファイル内で指定されたプローブ記述 (`-s` オプション) またはコマンド行で指定されたプローブ記述 (`-P`、`-m`、`-f`、`-n`、または `-i` オプション) の中で、既知のどのプローブにも一致しないような記述を含むプローブ記述が見つかると、`dtrace` はエラーを報告し、処理を終了します。

オペランド

`dtrace` コマンド行に 0 個以上の追加引数を指定し、一連のマクロ変数 (`$1` や `$2` など) を定義することができます。これらの追加引数は、`-s` オプションを使って指定された D プログラム内またはコマンド行から指定された D プログラム内で使用できます。マクロ変数の使用法の詳細については、『Solaris 動的トレースガイド』を参照してください。

終了ステータス

次の終了値が返されます。

`0` 正常終了。

D プログラム要求の場合、終了ステータス `0` は、プログラムのコンパイル、プローブの有効化、匿名状態の取得、のいずれかが成功したことを示します。指定されたトレース要求でエラーや欠落が発生した場合でも、`dtrace` は `0` を返します。

`1` エラーが発生しました。

D プログラム要求の場合、終了ステータス `1` は、プログラムのコンパイルが失敗したか、指定された要求に応じられなかったことを示します。

`2` コマンド行に無効なオプションまたは引数が指定された。

属性

次の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWdtrc
インタフェースの安定性	以下を参照。

コマンド行の構文は開発中 (Evolving) です。人間が読める形式の出力は不安定 (Unstable) です。

関連項目

`cpp`(1), `isainfo`(1), `libdtrace`(3LIB), `driver.conf`(4), `attributes`(5), `dtrace`(7D)

『Solaris 動的トレースガイド』

名前　　fdisk – 固定ディスクパーティションテーブルの作成または変更

形式　　
```
fdisk [-o offset] [-s size] [-P fill_patt] [-S geom_file]
     [-w | -r | -d | -n | -I | -B | -t | -T | -g | -G | -R | -E]
     [--F fdisk_file] [ [-v] -W {fdisk_file | –}]
     [-h] [-b masterboot]
     [-A id : act : bhead : bsect : bcyl : ehead : esect :
         ecyl : rsect : numsect]
     [-D id : act : bhead: bsect : bcyl : ehead: esect :
         ecyl : rsect : numsect] rdevice
```

機能説明　　このコマンドは、次のように使用します。

- x86 システム上の `fdisk` パーティションテーブルを作成および変更する
- SPARC または x86 システム上のリムーバブルメディアで `fdisk` パーティションテーブルを作成および変更する
- x86 システムのみ、固定ディスクの最初のセクターに格納されるマスターブートレコードをインストールする

このテーブルは、異なるオペレーティングシステムごとに予約されているディスク領域を識別するために、または第 2 段階のブートストラップのあるパーティション (アクティブな Solaris パーティション) を識別するために、第 1 段階のブートストラップ (またはファームウェア) により使用されます。*rdevice* 引数は、固定ディスクに対応する raw デバイスを指定する場合に使用しなくてはなりません。たとえば、`/dev/rdsk/c0t0d0p0` のように指定します。

このプログラムは、3 種類の動作モードで実行します。最初の動作モードは対話モードです。対話モードでは、ディスク上に存在するパーティションテーブルが表示され、さらにユーザーがテーブルを変更できるように、メニューが表示されます。メニュー、問い合わせ、警告、およびエラーメッセージは、特に説明がなくても理解できるものです。

対話モードでは、ディスク上にパーティションテーブルがない場合、ユーザーはデフォルトのパーティション分割を行うか、または初期テーブル値を指定するかの、どちらかを選択できます。デフォルトのパーティション分割では、ディスク全体が Solaris システムに割り当てられ、Solaris システムパーティションがアクティブになります。いずれの場合も、初期テーブルが作成された時点で、`fdisk` はパーティションテーブルとともに、第 1 段階のブートストラップ (x86 のみ) コードも書き込みます。

2 番目の動作モードは、エントリの追加、削除、または `fdisk` テーブル全体の置き換えを自動的に行う場合に使用します。このモードでは、コマンド行に指定したエントリを追加または削除できます。また、ファイルから `fdisk` テーブル全体を読み込んで、元のテーブルと置き換えることもできます。`fdisk` はこのファイルを作成する目的でも使用できます。コマンド行オプションを使用すると、任意の `fdisk` テーブルが、ディスク全体を Solaris システムに割り当てるデフォルトの `fdisk` テーブルに置き換えられます。

3 番目の動作モードは、ディスクの診断時に使用します。このモードでは、ディスクのあるセクションをユーザーが指定したパターンで埋めることができ、ディスクのモードセクションを読み書きすることもできます。

メニューオプション

fdisk プログラムの対話モードで使用できるメニューオプションは、次のとおりです。

`Create a partition`
このオプションを使用すると、新しいパーティションを作成できます。パーティションの最大数は 4 つです。パーティションのタイプ (SOLARIS、MS-DOS、UNIX、またはその他) が尋ねられます。次に、パーティションのサイズをディスクの割合で指定するように求められます。この時点で c と入力すると、開始シリンダ番号およびシリンダ内のパーティションのサイズが尋ねられます。c を入力しなかった場合は、パーティションが収まるように、fdisk プログラムが開始シリンダ番号を決定します。いずれの場合でも、パーティションが既存のパーティションとオーバーラップする場合、または収まりきらない場合は、メッセージが表示されて最初のメニューに戻ります。

`Change Active (Boot from) partition`
このオプションを使用すると、第 1 段階のブートストラップが第 2 段階のブートストラップを検索するパーティション (アクティブパーティションと呼ばれる) を指定できます。

`Delete a partition`
このオプションを使用すると、作成済みのパーティションを削除できます。そのパーティション内のすべてのデータが破壊されるので注意してください。

`Change between Solaris and Solaris2 Partition IDs`
このオプションを使用すると、ユーザーは、現在と以前の fdisk オペレーティングシステムパーティション ID を切り替えることができます。これにより、そのディスクパーティション内のデータが影響を受けることはありません。このオプションは、以前のソフトウェアとの互換性を維持するために提供されています。

この時点で、次に示すオプションを使用してパーティションテーブルの設定を変更することも、あるいは、テーブルを変更せずにセッションを中止することもできます。

`Exit` このオプションを使用すると、fdisk によってこのセッションで作成された新しいテーブルが固定ディスクに書き込まれ、fdisk は終了します。

`Cancel` パーティションテーブルを変更せずに終了します。

オプション

fdisk で使用できるオプションは、次のとおりです。

`-A` *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
引数で指定されたパーティションを追加します (形式については以下の -F オプションを参照)。fdisk テーブルが変更された場合、このオプションを使用すると、Solaris パーティション上の VTOC がゼロで埋められます。

`-b` *master_boot*
マスターブートプログラムとしてファイル *master_boot* を指定します。デフォルトのマスターブートプログラムは `/usr/lib/fs/ufs/mboot` です。

`-B`
デフォルトでディスク全体を 1 つの Solaris パーティションに割り当てます。

`-d`
詳細デバッグモードを有効にします。`fdisk` を使用するたびに、その状態を標準エラー出力に送ります。このオプションの出力を `-F` オプションで使用しないでください。

`-D` *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
引数で指定されたパーティションを削除します (形式については以下の `-F` オプションを参照)。引数が完全に一致しないと、エントリが削除されないことに注意してください。このオプションを使用して `fdisk` テーブルを変更すると、Solaris パーティションの VTOC がゼロで埋められます。

`-E`
ディスク全体を使用する EFI パーティションを作成します。

`-F` *fdisk_file*
ファイル *fdisk_file* を使用してテーブルを初期化します。`fdisk` テーブルが変更された場合、このオプションを使用すると、Solaris パーティション上の VTOC がゼロで埋められます。

fdisk_file には指定行を 4 行まで含めることができます。各行は復帰改行文字 (`\n`) で区切ります。行の先頭文字がアスタリスク (*) の場合、その行はコメント行とみなされます。各行は位置に依存するエントリからなり、エントリは空白またはコロンで区切ります。形式は次のとおりです。

id act bhead bsect bcyl ehead esect ecyl rsect numsect

エントリの値は次のとおりです。

id パーティションのタイプ。指定可能な数値は、`fdisk.h` に記述されています。

act アクティブパーティションプラグ。`0` は非アクティブ、`128` はアクティブを意味します。

bhead パーティションが開始するヘッド。`0` に設定されている場合、`fdisk` は他の情報に基づいて適切な値を設定します。

bsect パーティションが開始するセクター。`0` に設定されている場合、`fdisk` は他の情報に基づいて適切な値を設定します。

bcyl パーティションが開始するシリンダ。`0` に設定されている場合、`fdisk` は他の情報に基づいて適切な値を設定します。

ehead パーティションが終了するヘッド。0 に設定されている場合、fdisk は他の情報に基づいて適切な値を設定します。

esect パーティションが終了するセクター。0 に設定されている場合、fdisk は他の情報に基づいて適切な値を設定します。

ecyl パーティションが終了するシリンダ。0 に設定されている場合、fdisk は他の情報に基づいて適切な値を設定します。

rsect パーティションが開始するディスクの先頭からの相対的な位置にあるセクター。この指定は必須です。この値は、fdisk が他のフィールドを設定するのに使用できます。

numsect セクター数で表したこのディスクパーティションのサイズ。この指定は必須です。この値は、fdisk が他のフィールドを設定するのに使用できます。

-g
ディスクのラベルジオメトリを取得して、標準出力に出力します (形式については -S オプションを参照)。

-G
ディスクの物理ジオメトリを取得して、標準出力に出力します (形式については -S オプションを参照)。

-h
詳細メッセージを表示します。メッセージには全オプションのリストとともに、各オプションの説明が示されます。

-I
デバイスチェックを省略します。デバイスを使用せずに、ディスクに送られる内容のファイルイメージを生成する場合に使用します。このオプションは -S と組み合わせて使用する必要があります (上記を参照)。

-n
他のオプションで明示的に指定されている場合を除き、fdisk テーブルを更新しません。他のオプションを指定しない場合、-n オプションは、マスターブートレコードだけをディスクに書き込みます。また、-n オプションを指定した場合、fdisk は対話モードで起動しないことにも注意してください。

-o *offset*
ディスクの先頭からのブロックオフセット。このオプションは -P、-r、および -w とともに使用します。このオプションを指定しない場合、ゼロが想定されます。

-P *fill_patt*
パターン *fill_patt* でディスクを埋めます。*fill_patt* は 10 進数または 16 進数にすることができ、一定のロングパターンを表す数として使用します。*fill_patt* が # の場合、パターンは各ブロックで # になります。パターンはロングワードとして各ブロックに格納されて各ブロックを埋めます (-o および -s を参照)。

`-r`
ディスクから読み込んで標準出力に書き込みます。操作の開始ポイントおよびサイズを指定するオプション `-o` と `-s` を参照してください。

`-R`
読み取り専用ディスクとして扱います。このオプションはテスト用です。

`-s` *size*
操作を実行するブロック数 (`-o`を参照)

`-S` *geom_file*
geom_file の内容をラベルジオメトリとして設定します。*geom_file* には、行ごとに 1 つの指定を記述します。各行は復帰改行文字 (`\n`) で区切ります。行の先頭文字がアスタリスク (*) の場合、その行はコメント行とみなされます。各行は位置に依存するエントリからなり、空白またはコロンで区切ります。形式は次のとおりです。

pcyl ncyl acyl bcyl nheads nsectors sectsiz

エントリの値は次のとおりです。

pcyl ドライブの物理シリンダ数

ncyl ドライブの使用可能シリンダ数

acyl ドライブの代替シリンダ数

bcyl ドライブのオフセットシリンダ数 (ゼロにすべき)

nheads このドライブのヘッド数

nsectors トラックあたりのセクター数

sectsiz セクターサイズ (バイト単位)

`-t`
パーティションテーブル境界を越えないように、無効なスライステーブルエントリを調整します。

`-T`
パーティションテーブル境界にまたがる無効なスライステーブルエントリを削除します。

`-v`
HBA (仮想) ジオメトリディメンションを出力します。このオプションは `-W` フラグと組み合わせて使用する必要があります。このオプションは、仮想ジオメトリをサポートするプラットフォームで使用できます。(x86 のみ)。

`-w`
ディスクに書き込み、標準入力から読み込みます。操作の開始ポイントおよびサイズを指定するオプション `-o` と `-s` を参照してください。

`-W -`
ディスクテーブルを標準出力に書き込みます。

`-W` *fdisk_file*
ディスクテーブルから `fdisk` ファイル *fdisk_file* を作成します。このオプションは、`-F` オプションと組み合わせて使用できます。

ファイル

`/dev/rdsk/c0t0d0p0`	固定ディスクに対応づけられる raw デバイス
`/usr/lib/fs/ufs/mboot`	デフォルトのマスターブートプログラム

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
アーキテクチャ	x86 と SPARC
使用条件	SUNWcsu

関連項目

`uname`(1), fmthard(1M), `prtvtoc`(1M), `attributes`(5)

診断

ほとんどのメッセージは、読めば理解できるものです。プログラムの開始直後に次のメッセージが表示される場合があります。

`Fdisk: cannot open <device>`
このメッセージは、デバイス名引数が無効であることを示します。

`Fdisk: unable to get device parameters for device <device>`
このメッセージは、固定ディスクの構成に問題があるか、または固定ディスクドライバでエラーが発生したことを示します。

`Fdisk: error reading partition table`
このメッセージは、固定ディスクから最初の読み込み時にエラーが発生したことを示します。固定ディスクのコントローラまたはドライバに問題があるか、固定ディスクの構成に問題がある可能性があります。

`Fdisk: error writing boot record`
このメッセージは、固定ディスクに新しいパーティションテーブルを書き込むときに、エラーが発生したことを示します。固定ディスクコントローラ、ディスク自体、ドライバ、または固定ディスクの構成に問題がある可能性があります。

名前

ffbconfig, SUNWffb_config – FFB グラフィックスアクセラレータの設定

形式

```
/usr/sbin/ffbconfig [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file | machine | system]
     [-deflinear | true | false]
     [-defoverlay | true | false]
     [-linearorder | first | last]
     [-overlayorder | first | last]
     [-expvis | enable | disable]
     [-sov | enable | disable] [-maxwids n]
     [-extovl | enable | disable]
     [-g gamma-correction-value]
     [-gfile gamma-correction-file] [-propt] [-prconf]
     [-defaults]

/usr/sbin/ffbconfig [-propt ] [-prconf]

/usr/sbin/ffbconfig [-help] [-res ?]
```

機能説明

`ffbconfig`は、FFB グラフィックスアクセラレータおよび FFB 対応の X11 ウィンドウシステムのデフォルトの一部を設定します。

`ffbconfig`の 1 番目の形式では、指定したオプションを OWconfigファイルに保存します。これらのオプションは、次にウィンドウシステムをそのデバイスで実行するときに FFB デバイスを初期化するために使用されます。OWconfig ファイル内のオプションの更新は、異なるウィンドウセッションや再起動したシステムでも有効となります。

`-prconf`、`-propt`、`-help`、`-res ?` オプションだけを起動する 2番目と 3 番目の形式では、OWconfig ファイルは更新されません。また、3 番目の形式では、その他のオプションはすべて無視されます。

オプションは、一度に 1 つの FFB デバイスに対してのみ指定することができます。複数の FFB デバイスに対してオプションを指定するには、`ffbconfig` を複数回起動する必要があります。

`ffbconfig` で指定できるのは、FFB 固有のオプションだけです。デフォルトの表示色数、デフォルトの画像表示形式クラスなどを指定する通常のウィンドウシステムのオプションは、`openwin` コマンド行のデバイス修飾子で指定してください。詳細については、『OpenWindows Desktop Reference Manual』を参照してください。

ユーザーは、更新する OWconfig ファイルを指定することもできます。デフォルトでは、`/etc/openwin` ディレクトリツリーにあるマシン固有のファイルが更新されます。別のファイルを指定するには、`-file` オプションを使用します。たとえば、`/usr/openwin` ディレクトリツリーにあるシステム共通の OWconfig ファイルを代わりに更新することができます。

これらの標準 OWconfig ファイルのどちらもスーパーユーザーのみが書き込みを行えます。したがって、スーパーユーザーが所有する `ffbconfig` プログラムは、setuid による root の権限で実行されます。

オプション

`-dev` *device-filename*
FFB 特殊ファイルを指定します。デフォルトは`/dev/fbs/ffb0` です。

`-file machine|system`
更新する `OWconfig` ファイルを指定します。`machine` が指定された場合は、`/etc/openwin` ディレクトリツリーにあるマシン固有の `OWconfig` ファイルが更新されます。`system` が指定された場合は、`/usr/openwin` ディレクトリツリーにある共通の `OWconfig` ファイルが更新されます。指定されたファイルがない場合は、新たに生成されます。ほかのオプションを指定していない場合、このオプションは効果がありません。デフォルトは *machine* です。

`-res` *video-mode* [`now` | `try` [`noconfirm` | `nocheck`]]
指定した FFB デバイスに接続されているモニターを制御する際に使われる表示モードを指定します。

表示モードの形式は *width*`x`*height*`x`*rate* で、*width* はピクセル単位の画面幅、*height* はピクセル単位の画面の高さ、*rate* は画面を垂直方向に再描画する周期です。

`960x680x112s` や `960x680x108s` の `s` 接尾辞は、これらが立体表示モードであることを意味します。`640x480x60i` や `768x575x50i` の `i` 接尾辞は、インタレース表示タイミングを有効にします。この接尾辞がない場合は、ノンインタレースタイミングが使用されます。

`-res` (「形式」に記されている 3 番目の形式) にリフレッシュレートを指定する際は、値の直前に `x` の代わりに `@` を使用することができます。たとえば、`1280x1024@76` のように指定することができます。

一部の表示モードは、FFB の一部のバージョンのみが対応しています。また、FFB が対応している表示モードには、モニターが対応していないものもあります。FFB デバイスとモニターの両方が対応している表示モードのリストは、`-res ?` オプション付きの `ffbconfig` を実行することによって得ることができます。

FFB が対応している表示モードのリストを以下に示します。

記号名	説明
`1024x768x60`	
`1024x768x70`	
`1024x768x75`	
`1024x768x77`	

記号名	説明
1024x800x84	
1152x900x66	
1152x900x76	
1280x800x76	
1280x1024x60	
1280x1024x67	
1280x1024x76	
960x680x112s	(立体表示)
960x680x108s	(立体表示)
640x480x60	
640x480x60i	(インタレース)
768x575x50i	(インタレース)
1440x900x76	(高解像度)
1600x1000x66	(高解像度)
1600x1000x76i	(高解像度)
1600x1280x76	(高解像度)
1920x1080x72	(高解像度)
1920x1200x70	(高解像度)

記号名

便宜上、表示モードのいくつかには記号名が定義されています。*width*x*height*x*rate* の形式の代わりに、記号名を -res の引数として指定することができます。記号名 none は、ウィンドウシステムを実行すると、画面の解像度は現在デバイスにプログラムされている表示モードになることを意味します。

記号名	対応する表示モード
svga	1024x768x60
1152	1152x900x76
1280	1280x1024x76

記号名	対応する表示モード
stereo	960x680x112s
ntsc	640x480x60i
pal	768x575x50i
none	(デバイスでプログラムされている表示モード)

-res オプションには、表示モードの直後に次の追加引数を指定することができます。追加引数は、単独でも複数でも指定することができます。

now
OWconfig ファイルの表示モードを更新するとともに、FFB デバイスが指定した表示モードにただちにプログラムされます。この機能は、ウィンドウシステムを開始する前に表示モードを変更する際に便利です。

対象となるデバイスが稼働している間 (たとえば、ウィンドウシステムの稼働中) に、この追加オプションを ffbconfig に指定することはお勧めしません。予期しない結果になることもあります。now オプションを指定して ffbconfig コマンドを実行する場合は、最初にウィンドウシステムを終了してください。now オプションがウィンドウシステムのセッション中に使用された場合、表示モードはただちに変更されますが、画面の幅や高さはそのセッションが終了して次のセッションに入るまで変更されません。さらに、立体表示モードではシステムが変更を認識しないことがあります。したがって、ウィンドウシステムの稼働中には絶対に now オプションを指定しないでください。

noconfirm
確認と警告メッセージを省略し、要求された表示モードにプログラムします。

-res オプションを指定した際に、システムが使用できない状態になり、表示出力がなくなる場合があります。このような状況は、特定のコードが読み込まれた際のモニターセンスコードにあいまいさがあった場合などに発生します。このような事態を避けるために ffbconfig のデフォルトの動作では、この問題についての警告メッセージと処理を継続するかどうかを確認するメッセージを表示します。このオプションは、ffbconfig がシェルスクリプトから実行されている場合に便利です。

nocheck
モニターセンスコードに基づく通常のエラーチェックが行われません。ユーザーによって指定された表示モードは、現在接続されているモニターに適切かどうかにかかわらず受け付けられます。このオプションは、FFB デバイスに異なるモニターを接続する場合に便利です。注 : このオプションを指定すると、noconfirm も指定されます。

try
指定した表示モードに試験的にプログラムされます。ユーザーは、指定した表示モードを使用する場合は、メッセージが表示されてから 10 秒以内に y と入力します。表示されたモードを使用しない場合は、10 秒以内に任意の文字を入力します。y または RETURN キー以外の文字の入力は、すべて「使用しない」とみなされ、以前の表示モードに戻され、OWconfig ファイル中の表示モードは書き換えられません。その他の指定されたオプションは有効となります。RETURN キーの入力があった場合は、新しい表示モードを保持するかどうかを yes または no で確認するメッセージが表示されます。

構成済みデバイスを使用中に (たとえば、ウィンドウシステムを実行している場合)、try サブオプションを ffbconfig に指定してはなりません。予期せぬ結果になることがあります。try サブオプションを指定して ffbconfig を実行する前には、ウィンドウシステムを停止しておく必要があります。

-deflinear true | false
FFB には、2 種類の画像表示形式があります。リニア画像と非リニア画像です。リニア画像はガンマ補正され、非リニア画像は補正されません。リニア画像版も非リニア画像版も、ともに持つ画像表示形式が 2 つあります。24 ビット TrueColor と 8 ビット StaticGray です。

-deflinear true を指定すると、デフォルトの画像表示形式として、デフォルトで選択されたオプションを満たすリニア画像を設定します。具体的には、デフォルトの画像表示形式の選択オプションは、Xsun(1) の defdepth および defclassオプションによって設定されたものです。詳細については、『OpenWindows Desktop Reference Manual』を参照してください。

-deflinear false を指定すると、または他のデフォルトで選択されたオプションを満たすリニア画像がない場合は、これらの他のオプションを満たす非リニア画像がデフォルトとして選択されます。

FFB にはリニアオーバーレイ画像表示形式がないため、-defoverlay オプションが存在する場合は、このオプションを使用することはできません。

-defoverlay true | false
FFB が、残りの FFB 画像から切り離されたピクセルを持つ 8ビット疑似カラー画像を提供します。これを、オーバーレイ画像といいます。この画像表示形式で作成されたウィンドウは、他の画像表示形式で作成されたウィンドウに影響を与えません。逆に、他の画像表示形式で作成されたウィンドウは、オーバーレイウィンドウに影響を与えます。この画像表示形式では、256 種類の不透明カラーの値による拡張オーバーレイを使用することができます。-maxwids を参照してください。

-defoverlay に true を指定すると、オーバーレイ画像がデフォルト画像になります。-defoverlay に false を指定すると、他のデフォルトで選択された defdepth および defclass オプションを満たすオーバーレイでない画像表示形式が、デ

フォルトの画像表示形式として選択されます。詳細については、『OpenWindows Desktop Reference Manual』を参照してください。

`-defoverlay true` を使用する場合は、openwin コマンド行で選択されたデフォルトの深さとクラスは常に 8 ビット疑似カラーである必要があります。それ以外の場合は、警告メッセージが出力され、`-defoverlay` オプションは false として扱われます。`-deflinear` オプションが存在する場合、FFB にはリニアオーバーレイ画像表示形式がないため、このオプションは使用することができません。

`-linearorder first | last`
`first` を指定すると、FFB 画面用の X11 画面画像表示形式リスト上で、リニア画像が非リニア画像より前に表示されます。`last` を指定すると、非リニア画像は、リニア画像より前に表示されます。

`-overlayorder first | last`
`first` を指定すると、FFB 画面用の X11 画面画像表示形式リスト上で、8 ビット疑似カラーオーバーレイ画像が、非オーバーレイ画像より前に表示されます。`last` を指定すると、非オーバーレイ画像は、オーバーレイ画像より前に表示されます。

`-expvis enable | disable`
enable を指定すると、OpenGL Visual Expansion が起動されます。選択された画像表示形式グループ (8 ビット PseudoColor、24 ビット TrueColor など) は、画面画像表示形式リストで見つけることができます。

`-sov enable | disable`
`enable` を指定すると、ルートウィンドウの `SERVER_OVERLAY_VISUALS` 属性が有効になります。SOV 画像が転送され、それらの透過タイプ、値、階層は、この属性によって参照することができます。`disable` を指定すると、`SERVER_OVERLAY_VISUALS` 属性は定義されません。SOV 画像は転送されません。

`-maxwids` *n*
ウィンドウ ID (WID) として使用するために予約される最大数の FFB チャネルピクセル値を指定します。オーバーレイカラーマップのピクセル値の残りは、通常の X11 の未使用のカラーピクセルのために使用されます。確保された WID は、(XGL などの) 3 次元グラフィックスウィンドウ、MBX ウィンドウと、デフォルト以外の画像表示形式をもつウィンドウにより発生順に割り当てられます。X チャネルコードの `0` から (255-*n*) は、未使用のカラーピクセルです。(255-*n*+1) から 255 の X チャネルコードは、WID として使用するために予約されます。FFB と FFB2 の 有効な値は、`1`、`2`、`4`、`8`、`16`、`32` です。FFB2+ の有効な値は、`1`、`2`、`4`、`8`、`16`、`32`、`64` です。

`-extovl enable | disable`
このオプションは FFB2+ のみで使用することができます。enable を指定すると、拡張オーバーレイを使用することができます。このオーバーレイ画像には 256 種類の不透明カラーがあります。SOV 画像には 255 種類の不透明カラーと 1 種類の

透明カラーがあります。このオプションは、ハードウェアによる透明カラーを有効にするため、SOV 画像を使用するウィンドウで、より高い性能が得られます。

-g *gamma-correction value*
このオプションは FFB2+ のみで使用することができます。このオプションによって、ガンマ補正の値を変えることができます。すべてのリニア画像ではガンマ補正を使用することができます。デフォルトでは、ガンマ補正の値は 2.22 です。0より小さい値は無効 (不正) です。ガンマ補正の値はリニア画像に適用され、リニア画像の有効ガンマ値は 1.0 になります。これは、`XSolarisGetVisualGamma`(3) によって返される値です。この機能については、`XSolarisGetVisualGamma`(3) を参照してください。

このオプションは、ウィンドウシステムが稼動しているときに使用することができます。ガンマ補正の値を変更すると、リニア画像を使用して表示されているすべてのウィンドウが影響を受けます。

-gfile *gamma-correction file*
このオプションは FFB2+ のみで使用することができます。このオプションは、指定されたファイルからガンマ補正表を読み込みます。このファイルは、各行が R、G、B チャネルの値を持つように書式化されている必要があります。それらの値は、16 進数で指定し、値と値の間は 1 つ以上の空白文字で区切ります。このファイルでは、3 つの値の組が 256 種類定義されます。このファイルの例を以下に示します。

```
0x00 0x00 0x00
0x01 0x01 0x01
0x02 0x02 0x02
...
...
0xff 0xff 0xff
```

このオプションを使用することによって、ウィンドウシステムが稼動しているときにガンマ補正表を読み込むことができます。新しいガンマ補正は、このリニア画像によって表示されているすべてのウィンドウに影響を与えます。ユーザーが指定した表によってガンマ補正を行う際は、ガンマ補正の値は定義されません。デフォルトでは、ウィンドウシステムはガンマ補正値として 2.22 を使用し、このガンマ補正値に対応してウィンドウシステムが作成したガンマ補正表を読み込みます。

-defaults
すべてのオプションの値をそれぞれのデフォルト値に戻します。

-propt
`-file` オプションで指定された OWconfig ファイルに書かれた FFB オプションの値のうち、`-dev` オプションで指定されたデバイスに対するものすべてを表示します。`ffbconfig` の呼び出しが終了した後に、OWconfig ファイルに書き込まれるオプションの値を表示します。次に表示例を示します。

```
--- OpenWindows Configuration for /dev/fbs/ffb0 ---
OWconfig: machine
Video Mode: NONE
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
                 Overlay Visuals are last
OpenGL Visuals: disabled
SOV: disabled
Allocated WIDs: 32
```

`-prconf`
FFB のハードウェア構成を表示します。次に表示例を示します。

```
--- Hardware Configuration for /dev/fbs/ffb0 ---
Type: double-buffered FFB2 with Z-buffer
Board: rev x
PROM Information: @(#)ffb2.fth x.x xx/xx/xx
FBC: version x
DAC: Brooktree 9068, version x
3DRAM: Mitsubishi 1309, version x
EDID Data: Available - EDID version 1 revision x
Monitor Sense ID: 4  (Sun 37x29cm RGB color monitor)
Monitor possible resolutions: 1024x768x60, 1024x768x70,
      1024x768x75, 1152x900x66, 1152x900x76,
      1280x1024x67, 1280x1024x76,
      960x680x112s, 640x480x60
Current resolution setting: 1280x1024x76
```

`-help`
`ffbconfig` コマンド行のオプションと機能の概要を一覧で表示します。

デフォルト設定

`ffbconfig` コマンド行で指定されていないオプションについては、対応する OWconfig ファイル中のオプションは更新されず、ファイル内の値がそのまま使用されます。

ウィンドウシステムを実行する際に、`ffbconfig` による FFB オプションの指定がまったくなかった場合は、デフォルト値が使用されます。オプションのデフォルト値を以下に示します。

オプション	デフォルト値
`-dev`	`/dev/fbs/ffb0`
`-file`	`machine`
`-res`	`none`
`-deflinear`	`false`

オプション	デフォルト値
-defoverlay	false
-linearorder	last
-overlayorder	last
-expvis	enabled
-sov	enabled
-maxwids	32

-res オプションのデフォルト値 none とは、ウィンドウシステムが実行された場合に、画面解像度がそのデバイスに現在プログラムされている表示モードになることを意味しています。

これによって、PROM によってデバイスの解像度を指定しているユーザーとの共用性が保てます。(GX などの)一部のデバイスでは、PROM が表示モードを指定する唯一の手段です。これは、デフォルトの FFB 表示モードは、最終的に PROM によって決まることを意味しています。

使用例

例 1　モニターの種類の変更

モニターの種類を、垂直周波数 76 Hz で解像度 1280 × 1024 に変更する例を以下に示します。

```
example% /usr/sbin/ffbconfig -res 1280x1024x76
```

ファイル

/dev/fbs/ffb0　　デバイス特殊ファイル

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWffbcf

関連項目

mmap(2), attributes(5), fbio(7I), ffb(7D)

『OpenWindows Desktop Reference Manual』

名前

fmadm – fault management configuration tool

形式

`fmadm [-q] [`*subcommand* `[`*arguments*`]]`

機能説明

The `fmadm` utility can be used by administrators and service personnel to view and modify system configuration parameters maintained by the Solaris Fault Manager, fmd(1M). `fmd` receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

`fmadm` can be used to:

- view the set of diagnosis engines and agents that are currently participating in fault management,
- view the list of system components that have been diagnosed as faulty, and
- perform administrative tasks related to these entities.

The Fault Manager attempts to automate as many activities as possible, so use of `fmadm` is typically not required. When the Fault Manager needs help from a human administrator, service repair technician, or Sun, it produces a message indicating its needs. It also refers you to a knowledge article on Sun's web site, http://www.sun.com/msg/. The web site might ask you to use `fmadm` or one of the other fault management utilities to gather more information or perform additional tasks. The documentation for fmd(1M), fmdump(1M), and fmstat(1M) describe more about tools to observe fault management activities.

The `fmadm` utility requires the user to possess the `SYS_CONFIG` privilege. Refer to the 『Solaris のシステム管理 (セキュリティサービス)』 for more information about how to configure Solaris privileges. The `fmadm load` subcommand requires that the user possess all privileges.

SUBCOMMANDS

`fmadm` accepts the following subcommands. Some of the subcommands require additional options and operands:

`fmadm config`

Display the configuration of the Fault Manager itself, including the module name, version, and description of each component module. Fault Manager modules provide services such as automated diagnosis, self-healing, and messaging for hardware and software present on the system.

`fmadm faulty [-ai]`

Display the list of resources that the Fault Manager currently believes to be faulty. Faulty resources are determined by the set of modules that are performing automated diagnosis activities. The Fault Management Resource Identifier (FMRI), resource state, and Universal Unique Identifier (UUID) of the diagnosis are listed for each resource. An FMRI is a string that acts as the formal

name for a particular resource for which Solaris can perform automated fault management activities.

The Fault Manager associates the following states with every resource for which telemetry information has been received:

`ok` The resource is present and in use and has no known problems so far as the Fault Manager is concerned.

`unknown` The resource is not present or not usable but has no known problems. This might indicate the resource has been disabled or deconfigured by an administrator. Consult appropriate management tools for more information.

`degraded` The resource is present and usable, but one or more problems have been diagnosed in the resource by the Fault Manager.

`faulted` The resource is present but is not usable because one or more problems have been diagnosed by the Fault Manager. The resource has been disabled to prevent further damage to the system.

The UUID shown in the output for degraded and faulted resources uniquely identifies the Fault Manager diagnosis that discovered the problem. You can obtain additional details about the diagnosis using `fmdump -v -u` *uuid*. The `fmdump` output includes a message identifier that can be used to learn more about the problem impact and resolution procedures on Sun's web site, http://www.sun.com/msg/. By default, the `fmadm` faulty command only lists output for resources that are currently present and faulty. If you specify the `-a` option, all resource information cached by the Fault Manager is listed. The listing includes information for resources that might no longer be present in the system. If you specify the `-i` option, the persistent cache identifier for each resource in the Fault Manager is shown instead of the most recent state and UUID.

`fmadm flush` *fmri*	Flush the information cached by the Fault Manager for the specified resource, named by its FMRI. This subcommand should only be used when indicated by a documented Sun repair procedure. Typically, the use of this command is not necessary as the Fault Manager keeps its cache up-to-date automatically. If a faulty resource is flushed from the cache, administrators might need to apply additional commands to enable the specified resource.
`fmadm load` *path*	Load the specified Fault Manager module. *path* must be an absolute path and must refer to a module present in one of the defined directories for modules. Typically, the use of this command is not necessary as the Fault Manager loads modules automatically when Solaris initially boots or as needed.
`fmadm unload` *module*	Unload the specified Fault Manager module. Specify *module* using the basename listed in the `fmadm config` output. Typically, the use of this command is not necessary as the Fault Manager loads and unloads modules automatically based on the system configuration
`fmadm repair` *fmri* | *uuid*	Update the Fault Manager's resource cache to indicate that no problems are present in one or more resources that have been diagnosed to be faulty. If an *fmri* is specified, the state of the specified resource is updated. If a *uuid* is specified, the state of all resources associated with the corresponding diagnosis are updated. If the resource is currently believed to be faulted, it is set to the `unknown` state. If the resource is currently believed to be degraded, it is set to the `ok` state. Administrators might need to apply additional commands to re-enable a previously faulted resource. The `fmadm repair` subcommand should only be used at the direction of a documented Sun repair procedure. The use of this command is typically not necessary as the Fault Manager updates its resource cache automatically.
`fmadm reset [-s` *serd*`]` *module*	Reset the specified Fault Manager module or module subcomponent. If the `-s` option is present, the specified Soft Error Rate Discrimination (SERD) engine is reset within the module. If the `-s` option is not present, the entire module is reset and all persistent state associated with the module is deleted. The `fmadm reset`

subcommand should only be used at the direction of a documented Sun repair procedure. The use of this command is typically not necessary as the Fault Manager manages its modules automatically.

`fmadm rotate errlog | fltlog` Schedule a rotation of the specified fault manager log file. The log files are automatically rotated by an entry in the `logadm`(1M) configuration file that uses this subcommand. See `logadm`(1M) for more information on how to change the default log rotation options.

オプション

The following options are supported:

`-q` Set quiet mode. `fmadm` does not produce messages indicating the result of successful operations to standard output.

オペランド

The following operands are supported:

cmd The name of a subcommand listed in `SUBCOMMANDS`.

args One or more options or arguments appropriate for the selected *subcommand*, as described in `SUBCOMMANDS`.

終了ステータス

The following exit values are returned:

`0` Successful completion.

`1` An error occurred. Errors include a failure to communicate with `fmd` or insufficient privileges to perform the requested operation.

`2` Invalid command-line options were specified.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	See below.

The command-line options are Evolving. The human-readable output is Unstable.

関連項目

fmd(1M), fmdump(1M), fmstat(1M), `logadm`(1M), syslogd(1M), `attributes`(5)

『Solaris のシステム管理 (セキュリティサービス)』

http://www.sun.com/msg/

名前 fmd – fault manager daemon

形式 `/usr/lib/fm/fmd/fmd [-V] [-f file] [-o opt=val] [-R dir]`

機能説明 `fmd` is a daemon that runs in the background on each Solaris system. `fmd` receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components. When appropriate, the fault manager also sends a message to the syslogd(1M) service to notify an administrator that a problem has been detected. The message directs administrators to a knowledge article on Sun's web site, http://www.sun.com/msg/, which explains more about the problem impact and appropriate responses.

Each problem diagnosed by the fault manager is assigned a Universal Unique Identifier (UUID). The UUID uniquely identifes this particular problem across any set of systems. The fmdump(1M) utility can be used to view the list of problems diagnosed by the fault manager, along with their UUIDs and knowledge article message identifiers. The fmadm(1M) utility can be used to view the resources on the system believed to be faulty. The fmstat(1M) utility can be used to report statistics kept by the fault manager. The fault manager is started automatically when Solaris boots, so it is not necessary to use the `fmd` command directly. Sun's web site explains more about what capabilities are currently available for the fault manager on Solaris.

オプション The following options are supported

`-f` *file*	Read the specified configuration *file* prior to searching for any of the default fault manager configuration files.
`-o` *opt=value*	Set the specified fault manager option to the specified value. Fault manager options are currently a Private interface; see `attributes`(5) for information about Private interfaces.
`-R` *dir*	Use the specified root directory for all pathnames evaluated by the fault manager, instead of the default root (`/`).
`-V`	Print the fault manager's version to stdout and exit.

終了ステータス The following exit values are returned:

`0`	Successful completion
`1`	An error occurred which prevented the fault manager from initializing, such as failure to open the telemetry transport.
`2`	Invalid command-line options were specified.

ファイル

`/etc/fm/fmd`	Fault manager configuration directory
`/usr/lib/fm/fmd`	Fault manager library directory
`/var/fm/fmd`	Fault manager log directory

属性

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	Evolving

関連項目

svcs(1), fmadm(1M), fmdump(1M), fmstat(1M), syslogd(1M), attributes(5), smf(5)

http://www.sun.com/msg/

注意事項

The Fault Manager is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/fmd:default
```

The service's status can be queried using the svcs(1) command. Administrators should not disable the Fault Manager service.

名前

fmdump – fault management log viewer

形式

```
fmdump [-efvV] [-c class] [-R dir] [-t time] [-T time]
     [-u uid] [file]
```

機能説明

The `fmdump` utility can be used to display the contents of any of the log files associated with the Solaris Fault Manager, fmd(1M). The Fault Manager runs in the background on each Solaris system. It receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

The Fault Manager maintains two sets of log files for use by administrators and service personnel:

error log — A log which records error telemetry, the symptoms of problems detected by the system.

fault log — A log which records fault diagnosis information, the problems believed to explain these symptoms.

By default, `fmdump` displays the contents of the fault log, which records the result of each diagnosis made by the fault manager or one of its component modules.

An example of a default `fmdump` display follows:

```
# fmdump
TIME                 UUID                                 SUNW-MSG-ID
Dec 28 13:01:27.3919 bf36f0ea-9e47-42b5-fc6f-c0d979c4c8f4 FMD-8000-11
Dec 28 13:01:49.3765 3a186292-3402-40ff-b5ae-810601be337d FMD-8000-11
Dec 28 13:02:59.4448 58107381-1985-48a4-b56f-91d8a617ad83 FMD-8000-OW
...
```

Each problem recorded in the fault log is identified by:

- The time of its diagnosis
- A Universal Unique Identifier (UUID) that can be used to uniquely identify this particular problem across any set of systems
- A message identifier that can be used to access a corresponding knowledge article located at Sun's web site, http://www.sun.com/msg/

If a problem requires action by a human administrator or service technician or affects system behavior, the Fault Manager also issues a human-readable message to syslogd(1M). This message provides a summary of the problem and a reference to the knowledge article on the Sun web site, http://www.sun.com/msg/.

You can use the `-v` and `-V` options to expand the display from a single-line summary to increased levels of detail for each event recorded in the log. The `-c`, `-t`, `-T`, and `-u` options can be used to filter the output by selecting only those events that match the specified *class*, range of times, or *uuid*.

If more than one filter option is present on the command-line, the options combine to display only those events that are selected by the logical `AND` of the options. If more than one instance of the same filter option is present on the command-line, the like options combine to display any events selected by the logical `OR` of the options. For example, the command:

```
# fmdump -u uuid1 -u uuid2 -t 02Dec03
```

selects events whose attributes are (`uuid1` `OR` `uuid2`) `AND` (time on or after 02Dec03).

オプション

The following options are supported:

`-c` *class*
: Select events that match the specified class. The class argument can use the glob pattern matching syntax described in `sh`(1). The class represents a hierarchical classification string indicating the type of telemetry event. More information about Sun's telemetry protocol is available at Sun's web site, http://www.sun.com/msg/.

`-e`
: Display events from the fault management error log instead of the fault log. This option is shorthand for specifying the pathname of the error log file.

  The error log file contains Private telemetry information used by Sun's automated diagnosis software. This information is recorded to facilitate post-mortem analysis of problems and event replay, and should not be parsed or relied upon for the development of scripts and other tools. See `attributes`(5) for information about Sun's rules for Private interfaces.

`-f`
: Follow the growth of the log file by waiting for additional data. `fmdump` enters an infinite loop where it will sleep for a second, attempt to read and format new data from the log file, and then go back to sleep. This loop can be terminated at any time by sending an interrupt (`Control-C`).

`-R` *dir*
: Use the specified root directory for the log files accessed by `fmdump`, instead of the default root (`/`).

`-t` *time*
: Select events that occurred at or after the specified time. The time can be specified using any of the following forms:

mm/*dd*/*yy* *hh*:*mm*:*ss*	Month, day, year, hour in 24-hour format, minute, and second. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

mm/*dd*/*yy hh*:*mm*	Month, day, year, hour in 24-hour format, and minute. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.
mm/*dd*/*yy*	12:00:00AM on the specified month, day, and year.
ddMonyy hh:*mm*:*ss*	Day, month name, year, hour in 24-hour format, minute, and second. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.
ddMonyy hh:*mm*	Day, month name, year, hour in 24-hour format, and minute. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.
Mon dd hh:*mm*:*ss*	Month, day, hour in 24-hour format, minute, and second of the current year.
yyyy-*mm*-*dd* [T *hh*:*mm*[:*ss*]]	Year, month, day, and optional hour in 24-hour format, minute, and second. The second, or hour, minute, and second, can be optionally omitted.
ddMonyy	12:00:00AM on the specified day, month name, and year.
hh:*mm*:*ss*	Hour in 24-hour format, minute, and second of the current day.
hh:*mm*	Hour in 24-hour format and minute of the current day.
*T*ns | *T*nsec	*T* nanoseconds ago where T is an integer value specified in base 10.
*T*us | *T*usec	*T* microseconds ago where T is an integer value specified in base 10.
*T*ms | *T*msec	T milliseconds ago where T is an integer value specified in base 10.

T`s` | *T*`sec`	T seconds ago where *T* is an integer value specified in base 10.
T`m` | *T*`min`	*T* minutes ago where *T* is an integer value specified in base 10.
T`h` | *T*`hour`	*T* hours ago where *T* is an integer value specified in base 10.
T`d` | *T*`day`	*T* days ago where *T* is an integer value specified in base 10.

You can append a decimal fraction of the form `.`*n* to any `-t` option argument to indicate a fractional number of seconds beyond the specified time.

`-T` *time* Select events that occurred at or before the specified time. *time* can be specified using any of the time formats described for the `-t` option.

`-u` *uuid* Select fault diagnosis events that exactly match the specified *uuid*. Each diagnosis is associated with a Universal Unique Identifier (UUID) for identification purposes. The `-u` option can be combined with other options such as `-v` to show all of the details associated with a particular diagnosis.

If the `-e` option and `-u` option are both present, the error events that are cross-referenced by the specified diagnosis are displayed.

`-v` Display verbose event detail. The event display is enlarged to show additional common members of the selected events.

`-V` Display very verbose event detail. The event display is enlarged to show every member of the name-value pair list associated with each event. In addition, for fault logs, the event display includes a list of cross-references to the corresponding errors that were associated with the diagnosis.

オペランド

The following operands are supported:

file Specifies an alternate log file to display instead of the system fault log. The `fmdump` utility determines the type of the specified log automatically and produces appropriate output for the selected log.

終了ステータス

The following exit values are returned:

`0` Successful completion. All records in the log file were examined successfully.

`1` A fatal error occurred. This prevented any log file data from being examined, such as failure to open the specified file.

`2` Invalid command-line options were specified.

3 The log file was opened successfully, but one or more log file records were not displayed, either due to an I/O error or because the records themselves were malformed. `fmdump` issues a warning message for each record that could not be displayed, and then continues on and attempts to display other records.

ファイル

`/var/fm/fmd` Fault management log directory

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	See below.

The command-line options are Evolving. The human-readable error log output is Private. The human-readable fault log output is Evolving.

関連項目

`sh`(1), fmadm(1M), fmd(1M), fmstat(1M), syslogd(1M), `libexacct`(3LIB), `attributes`(5)

『Solaris のシステム管理 (セキュリティサービス)』

http://www.sun.com/msg/

注意事項

Fault logs contain references to records stored in error logs that can be displayed using `fmdump -V` to understand the errors that were used in the diagnosis of a particular fault. These links are preserved if an error log is renamed as part of log rotation. They can be broken by removing an error log file, or by moving it to another filesystem directory. `fmdump` can not display error information for such broken links. It continues to display any and all information present in the fault log.

名前

fmstat – report fault management module statistics

形式

```
fmstat  [-astTz] [-m module] [interval [count]]
```

機能説明

The `fmstat` utility can be used by administrators and service personnel to report statistics associated with the Solaris Fault Manager, fmd(1M) and its associated set of modules. The Fault Manager runs in the background on each Solaris system. It receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

You can use `fmstat` to view statistics for diagnosis engines and agents that are currently participating in fault management. The documentation for fmd(1M), fmadm(1M), and fmdump(1M) describes more about tools to observe fault management activities.

If the `-m` option is present or the `-t` option is present, `fmstat` reports any statistics kept by the specified fault management module. The module list can be obtained using `fmadm config`.

If the `-m` option is not present, `fmstat` reports the following statistics for each of its client modules:

`module`	The name of the fault management module, as reported by `fmadm config`.
`ev_recv`	The number of telemetry events received by the module.
`ev_acpt`	The number of events accepted by the module as relevant to a diagnosis.
`wait`	The average number of telemetry events waiting to be examined by the module.
`svc_t`	The average service time for telemetry events received by the module, in milliseconds.
`%w`	The percentage of time that there were telemetry events waiting to be examined by the module.
`%b`	The percentage of time that the module was busy processing telemetry events.
`open`	The number of active cases (open problem investigations) owned by the module.
`solve`	The total number of cases solved by this module since it was loaded.
`memsz`	The amount of dynamic memory currently allocated by this module.
`bufsz`	The amount of persistent buffer space currently allocated by this module.

The `fmstat` utility requires the user to posses the `SYS_CONFIG` privilege. Refer to the 『Solaris のシステム管理 (セキュリティサービス)』 for more information about how to configure Solaris privileges.

オプション

The following options are supported:

`-a` Print all statistics for a module, including those kept on its behalf by `fmd`. If the `-a` option is not present, only those statistics kept by the module are reported. If the `-a` option is used without the `-m` *module*, a set of global statistics associated with `fmd` are displayed.

`-m` *module* Print a report on the statistics associated with the specified fault management module, instead of the default statistics report. Modules can publish an arbitrary set of statistics to help Sun service the fault management software itself. The module statistics constitute a Private interface. See `attributes`(5) for information on Sun's rules for Private interfaces. Scripts should not be written that depend upon the values of fault management module statistics as they can change without notice.

`-s` Print a report on Soft Error Rate Discrimination (SERD) engines associated with the module instead of the default module statistics report. A SERD engine is a construct used by fault management software to determine if a statistical threshold measured as *N* events in some time *T* has been exceeded. The `-s` option can only be used in combination with the `-m` option.

`-t` Print a report on the statistics associated with each fault management event transport. Each fault management module can provide the implementation of one or more event transports.

`-T` Print a table of the authority information associated with each fault management event transport. If the `-m` option is present, only transports associated with the specified module are displayed.

`-z` Omit statistics with a zero value from the report associated with the specified fault management module. The `-z` option can only be used in combination with the `-m` option.

オペランド

The following operands are supported:

count Print only count reports, and then exit.

interval Print a new report every *interval* seconds.

If no *interval* and no *count* are specified, a single report is printed and `fmstat` exits. If an *interval* is specified but no *count* is specified, `fmstat` prints reports every *interval* seconds indefinitely until the command is interrupted.

終了ステータス

The following exit values are returned:

`0` Successful completion.

`1` A fatal error occurred. A fatal error could be the failure to communicate with fmd(1M). It could also be that insufficient privileges were available to perform the requested operation.

2 Invalid command-line options were specified.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	See below.

The command-line options are Evolving. The human-readable default report is Unstable. The human-readable module report is Private.

関連項目

fmadm(1M), fmd(1M), fmdump(1M), `attributes`(5)

『Solaris のシステム管理 (セキュリティサービス)』

名前

fmthard – populate label on hard disks

形式

SPARC

```
fmthard -d data | -n volume_name | -s datafile [-i] /dev/rdsk/c?
     [t?] d?s2
```

x86

```
fmthard -d data | -n volume_name | -s datafile [-i] /dev/rdsk/c?
     [t?] d?s2
```

機能説明

The `fmthard` command updates the VTOC (Volume Table of Contents) on hard disks and, on x86 systems, adds boot information to the Solaris `fdisk` partition. One or more of the options `-s` *datafile*, `-d` *data*, or `-n` *volume_name* must be used to request modifications to the disk label. To print disk label contents, see `prtvtoc`(1M). The `/dev/rdsk/c?`[`t?`]`d?s2` file must be the character special file of the device where the new label is to be installed. On x86 systems, fdisk(1M) must be run on the drive before `fmthard`.

If you are using an x86 system, note that the term "partition" in this page refers to *slices* within the x86 `fdisk` partition on x86 machines. Do not confuse the partitions created by `fmthard` with the partitions created by `fdisk`.

オプション

The following options are supported:

`-d` *data*
: The *data* argument of this option is a string representing the information for a particular partition in the current VTOC. The string must be of the format *part:tag:flag:start:size* where *part* is the partition number, *tag* is the ID TAG of the partition, *flag* is the set of permission flags, *start* is the starting sector number of the partition, and *size* is the number of sectors in the partition. See the description of the *datafile* below for more information on these fields.

`-i`
: This option allows the command to create the desired VTOC table, but prints the information to standard output instead of modifying the VTOC on the disk.

`-n` *volume_name*
: This option is used to give the disk a *volume_name* up to 8 characters long.

`-s` *datafile*
: This option is used to populate the VTOC according to a *datafile* created by the user. If the *datafile* is "–", `fmthard` reads from standard input. The *datafile* format is described below. This option causes all of the disk partition timestamp fields to be set to zero.

  Every VTOC generated by `fmthard` will also have partition 2, by convention, that corresponds to the whole disk. If the input in *datafile* does not specify an entry for partition 2, a default partition 2 entry will be created automatically in VTOC with the tag `V_BACKUP` and size equal to the full size of the disk.

The *datafile* contains one specification line for each partition, starting with partition 0. Each line is delimited by a new-line character (`\n`). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, separated by "white space" and having the following format:

partition tag flag starting_sector size_in_sectors

where the entries have the following values:

partition — The partition number. Currently, for Solaris SPARC, a disk can have up to 8 partitions, `0`–`7`. Even though the *partition* field has 4 bits, only 3 bits are currently used. For x86, all 4 bits are used to allow slices `0`–`15`. Each Solaris `fdisk` partition can have up to 16 slices.

tag — The partition tag: a decimal number. The following are reserved codes: `0` (`V_UNASSIGNED`), `1` (`V_BOOT`), `2` (`V_ROOT`), `3` (`V_SWAP`), `4` (`V_USR`), `5` (`V_BACKUP`), `6` (`V_STAND`), `7` (`V_VAR`), and `8` (`V_HOME`).

flag — The flag allows a partition to be flagged as unmountable or read only, the masks being: `V_UNMNT` `0x01`, and `V_RONLY` `0x10`. For mountable partitions use `0x00`.

starting_sector — The sector number (decimal) on which the partition starts.

size_in_sectors — The number (decimal) of sectors occupied by the partition.

You can save the output of a `prtvtoc` command to a file, edit the file, and use it as the *datafile* argument to the `-s` option.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

関連項目

`uname`(1), format(1M), `prtvtoc`(1M), `attributes`(5)

x86 Only — fdisk(1M), installgrub(1M)

注意事項

Special care should be exercised when overwriting an existing VTOC, as incorrect entries could result in current data being inaccessible. As a precaution, save the old VTOC.

For disks under one terabyte, `fmthard` cannot write a VTOC on an unlabeled disk. Use format(1M) for this purpose.

名前

format – ディスクのパーティション分割および保守のためのユーティリティ

形式

```
format  [-f command-file] [-l log-file] [-x data-file]
 [-d disk-name] [-t disk-type] [-p partition-name] [-s]
 [-m] [-M] [-e] [disk-list]
```

機能説明

format を使用すると、システムディスクのフォーマット、ラベル付け、修復、および分析が行えます。従来のディスク保守用プログラムとは異なり、format は SunOS 環境で実行します。システムの稼働中はシステムディスクへの操作が制限されるので、format はメモリー常駐型のシステム環境でも使用することができます。しかし、大部分のアプリケーションでは、SunOS 環境で format を実行する方が簡単です。

-x オプションが指定されると、format は最初に *data-file* で定義されているディスクリストを使用します。次に format は、FORMAT_PATH 環境変数、コロンで区切られたファイル名またはディレクトリ (あるいはその両方) のリストを調べます。ディレクトリの場合、format はそのディレクトリで format.dat というファイルを検索します。ファイル名は絶対パス名でなければならず、そのまま使用されます。format は指定された各ファイルのディスクおよびパーティション定義をすべて作業用セットに追加します。複数の同じ定義は無視され、そのことが通知されることはありません。FORMAT_PATH が設定されていない場合のデフォルトのパスは /etc/format.dat です。

disk-list は、c?t?d? または /dev/rdsk/c?t?d?s? の形式のディスクリストです。後者の形式では、シェルのワイルドカードを指定できます。たとえば、/dev/rdsk/c2* を指定すると、format はコントローラ c2 だけに接続されている全ドライブを操作対象とします。*disk-list* が指定されていない場合、format は、操作対象となり得る、システム上に存在するすべてのディスクを一覧表示します。

着脱式媒体装置がリストに含まれるのは、ユーザーがエキスパートモード (-e オプション) で format を実行した場合だけです。この機能は、下位互換性を確保するために用意されています。書き替え可能な着脱式媒体装置には rmformat(1) を使用します。

オプション

次のオプションを指定できます。

-d *disk-name*
: format プログラム開始時にカレントにするディスクを指定します。ディスクは論理名 (-d c0t1d0など) で指定します。ディスクリストにディスクを 1 つだけ指定しても、同じ結果が得られます。

-e
: SCSI エキスパートメニューを有効にします。ただし、このオプションは、不用意に使用しないでください。

-f *command-file*
: 標準入力ではなく、*command-file* からコマンド入力を受け付けます。ファイルにはキーボードから入力した場合とまったく同じコマンドが指定されていなければなりません。このオプションが指定されている場合、format は continue? プロンプト

を出力しません。*command-file* 内に y(es) または n(o) の応答を指定する必要はありません。非対話モードの場合、`format` はディスク選択番号の入力を求めることはありません。`format` を起動するときに、`-d` *disk-name* オプションを使用して現在の作業用ディスクを指定するか、または *command-file* に `disk` およびディスク選択番号を指定する必要があります。

`-l` *log-file*
: 指定された *log-file* に `format` セッションのトランスクリプトを記録します。これには標準入力、標準出力、および標準エラー出力が含まれます。

`-m`
: 拡張メッセージを有効にします。エラー発生時に詳細情報が得られます。

`-M`
: 拡張メッセージおよび診断メッセージを有効にします。フォーマット中に、SCSI デバイスのモードページの状態についてさまざまな情報が得られます。

`-p` *partition-name*
: プログラムの開始時にカレントにするディスクのパーティションテーブルを指定します。このテーブルは、データファイルで定義されている名前を使用して指定します。このオプションを使用できるのは、ディスクがカレントになっており、さらにディスクタイプが指定されているか、ディスクタイプがディスクラベルから取得できる場合に限られます。

`-s`
: サイレント。あらゆる標準出力を抑制します。エラーメッセージは引き続き表示されます。このオプションは通常、`-f` オプションと組み合わせて使用します。

`-t` *disk-type*
: プログラムの開始時にカレントにするディスクのタイプを指定します。ディスクタイプは、データファイルで定義されている名前を使用して指定します。このオプションを使用できるのは、上記と同様、ディスクがカレントになっている場合に限られます。

`-x` *data-file*
: *data-file* に指定されているディスクのリストを使用します。

使用法

オプションを指定しないで format を起動した場合、あるいは、`-e`、`-l`、`-m`、`-M`、または `-s` オプションを指定して format を起動した場合、使用できるディスクを示した番号付きリストが表示され、リスト内の番号でディスクを指定するように求められます。マシンに 10 台以上のディスクが搭載されている場合は、スペースキーを押して、次の 1 画面分のディスクを表示します。

ディスクが現在の画面に表示されていなくても、リスト内の番号でディスクを指定できます。たとえば、画面にディスク 11 〜 12 が表示されている場合に 25 を入力すると、リスト内の 25 番目のディスクを指定できます。現在の画面に表示されてい

ないディスクの番号を入力した場合は、選択の確認を求めるプロンプトが表示されます。表示リストに含まれている番号を入力した場合は、そのまま選択が受け付けられ、プロンプトは出力されません。

ディスクを指定すると、メインメニューが表示されます。このメニューでは次の作業を実行できます。

analyze　読み取り、書き込み、および比較テストを実行します。

backup　バックアップラベルを検索します。

cache　書き込みキャッシュおよび読み取りキャッシュを有効または無効にします。またはその状態を照会します。このメニュー項目が表示されるのは、-e オプションを指定して format を起動した場合だけです。このオプションを使用できるのは、SCSI デバイスに限られます。

current　デバイス名、ディスクジオメトリ、およびディスク装置のパス名を表示します。

defect　欠陥領域リストを検索して出力します。このオプションを使用できるのは、SCSI デバイスに限られます。IDE ディスクの場合は、自動欠陥領域管理機能が実行されます。`defect` オプションを IDE ディスクで使用すると、次のメッセージが出力されます。

```
Controller does not support defect management
or disk supports automatic defect management
```

disk　以降の操作で使用するディスク (カレントディスク) を選択します。

fdisk　fdisk(1M) プログラムを実行し、Solaris ソフトウェア (x86 ベースのシステムのみ) 用の `fdisk` パーティションを作成します。

format　カレントディスクをフォーマットして検証します。このオプションを使用できる、SCSI デバイスに限られます。IDE ディスクは製造元により、あらかじめフォーマットされています。IDE ディスクに対して `format` オプションを使用すると、次のメッセージが出力されます。

```
Cannot format this drive. Please use your
manufacturer-supplied formatting utility.
```

inquiry　カレントドライブの製造元、製品名、およびリビジョンレベルを表示します。

label　カレントディスクに新しいラベルを書き込みます。

partition　スライスを作成または変更します。

quit　フォーマットメニューを終了します。

repair　ディスク上の特定のブロックを修復します。

save　新しいディスクおよびスライス情報を保存します。

type　ディスクタイプを選択 (定義) します。

verify　ラベルを読み取って表示します。シリンダ数、代替シリンダ数、ヘッド数、セクター数、パーティションテーブルなどの情報を出力します。

volname　8 文字の新しいボリューム名を用いて、ディスクにラベルを付けます。

環境

FORMAT_PATH　ディスクとパーティション定義用のコロンで区切られたファイル名またはディレクトリ (あるいはその両方)。ディレクトリを指定した場合、format はそのディレクトリでファイル format.dat を検索します。

ファイル

/etc/format.dat　デフォルトのデータファイル

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

fmthard(1M), prtvtoc(1M), rmformat(1), format.dat(4), attributes(5), sd(7D)

『Solaris のシステム管理 (基本編)』』のディスク管理に関する章を参照してください。

x86のみ

fdisk(1M)

警告

format 機能を選択して Maxtor 207MB ディスクをフォーマットすると、次のメッセージが出力されます。

```
Mode sense page(4) reports rpm value as 0, adjusting it to 3600
```

これはドライバのバグであり、他社の旧式のドライブでも発生する可能性があります。上記メッセージはエラーではありません。ドライブは正常に動作します。

シリンダ 0 にはパーティションテーブル (ディスクラベル) がありますが、raw ディスクパーティションで使用すると、他社のソフトウェアによって上書きされることがあります。

format は容量が 1 TB を超えるディスクまたは LUN をサポートするために、EFI に準拠したディスクラベルの記載をサポートします。しかし、ファイルシステムやボリュームマネージャなど、多くのソフトウェアコンポーネントは容量がまだ 1 TB もしくはそれ以下に制限されているので、注意が必要です。詳細については、『Solaris のシステム管理 (基本編)』を参照してください。

注意事項

format にはヘルプ機能が用意されており、入力を求められたときにいつでも使用できます。要求されている情報についてヘルプが必要な場合は、単に疑問符 (?) を入力します。求められている情報についての簡単な説明が出力されます。メニュープロンプトに ? を入力すると、使用できるコマンドの一覧が表示されます。

SCSI ディスクの場合、Primary および Grown 両方の欠陥領域リストでフォーマットがデフォルトで実行されます。フォーマットを実行する前に、欠陥領域メニューで Primary リストだけを抽出しておくと、Primary リストに対するフォーマットだけが実行されます。

キャッシュの状態を変更できるのは、SCSI デバイスだけです。また、すべての SCSI デバイスがキャッシュ状態の変更または保存をサポートしているわけではありません。

名前　fsck – ファイルシステムの検査および修復

形式

```
fsck  [-F FSType] [-m] [-V] [special]...

fsck  [-F FSType] [-n |  N |  y |  Y] [-V] [-o FSType-specific-options]
 [special]...
```

機能説明

fsck はファイルシステムの状態を検査し、不整合を対話形式で修復します。ファイルシステムに不整合がある場合、デフォルトでは、修復処理を行う前に、ユーザーからの yes または no の応答を待機します。ユーザーに書き込み権が与えられていない場合、fsck はデフォルトで no の応答に対する動作をします。修復処理によっては、データが失われることがあります。データが失われる量と重要度は、診断出力から判断できます。

FSType-specific-options は、(空白を入れずに) コンマで区切ったオプションのリストまたはキーワード / 属性のペアのリストとして指定します。これらは、*FSType-* 固有のコマンドモジュールによって解釈されます。

special は、ファイルシステムが配置されている文字型特殊デバイスを指定します。/dev/rdsk/c1t0d0s7 などがその例です。注 : ブロック型特殊デバイスではなく、文字型特殊デバイスを使用してください。fsck は、ブロック型特殊デバイスがマウントされている場合には、動作しません。

special デバイスを指定しなかった場合、fsck は /etc/vfstab に指定されているファイルシステムを検査します。/etc/vfstab の中で、fsckdev フィールドに文字型特殊デバイスエントリがあり、fsckpass フィールドにゼロ以外の数値エントリのあるエントリが検査されます。-F *FSType* を指定すると、指示されたタイプのファイルシステムタイプだけが検査されます。

special が指定されていても、 -F が指定されていない場合は、/etc/vfstab の対応するエントリを探すことによって、ファイルシステムタイプが判別されます。対応するエントリがない場合は、/etc/default/fs に指定されているデフォルトのローカルファイルシステムタイプが使用されます。

ファイルシステムタイプが並列検査に対応している場合 (たとえば ufs)、検査可能ないくつかのファイルシステムを並列に検査できます。詳細は、ファイルシステム固有のマニュアルページ (fsck_ufs(1M) など) を参照してください。

オプション　次の汎用オプションを指定できます。

-F *FSType*　操作するファイルシステムのタイプを指定します。

-m　検査だけで修復は行いません。このオプションを指定すると、ファイルシステムがマウントできる状態かどうかが検査され、該当する終了ステータスが返されます。ファイルシステムがマウントできる状態であれば、fsck は次のメッセージを表示します。

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

`-n | -N` `fsck` からのすべての問い合わせに対して `no` の応答をするものとみなします。ファイルシステムを書き込みでオープンしません。

`-V` コマンド行を展開して表示しますが、コマンドは実行しません。このオプションは、コマンド行の検査および検証するのに使用します。

`-y | Y` `fsck` からのすべての問い合わせに対して `yes` の応答をするとみなします。

`-o` *specific-options* *specific-options* には、次のサブオプションの組み合わせを (空白は入れずに) コンマで区切って指定します。

`b=`*n* ファイルシステムのスーパーブロックとしてブロック *n* を使用します。ブロック 32 はつねに、代替スーパーブロックの 1 つです。`-Nv` オプションを指定して newfs(1M) を実行すると、他のスーパーブロックの位置を調べることができます。

`c` 古い (静的テーブル) 形式のファイルシステムを新しい (動的テーブル) 形式に変換します。ファイルシステムが新しい形式のときは、ファイルシステム構成が古い形式をサポートできる場合、古い形式に変換します。対話モードの場合、変換の向きが表示され、変換を実行するかどうかが尋ねられます。否定応答を返すと、そのファイルシステムに対してそれ以上の操作は実行されません。非対話モードの場合は、変換の向きが表示され、可能であれば、ユーザーとの対話なしで実行されます。すべてのファイルシステムを一括変換する場合は、非対話モードでの変換が便利です。ファイルシステムのタイプは `fstyp`(1M) の出力の先頭行からわかります。注 : `c` オプションを使用することはほとんどありません。4.1 より前のリリースとの互換性を確保する場合に限って指定してください。今後のリリースにこのオプションが含まれるかどうかは保証されません。

`f` スーパーブロックのクリーンフラグの状態に関係なく、ファイルシステムを強制的に検査します。

`p` 非対話 (preen) モードでファイルシステムを検査して修復します。ユーザーとの対話を必要とする問題が検出された場合には、ただちに終了します。並列ファイルシステム検査を有効にする場合、このオプションは必須です。

`w` 書き込み可能なファイルシステムだけを検査します。

終了ステータス

`0` ファイルシステムに問題はなく検査不要

`1` 誤ったパラメータが指定された

`32` ファイルシステムがマウント解除されており、検査が必要 (`fsck -m` オプションの場合のみ)

`33` ファイルシステムはマウント済み

`34` デバイスの状態が取得できない

`36` 修正不可能なエラーが検出された - 通常どおりに終了

`37` 処理中にシグナルが捕捉された

`39` 修正不可能なエラーが検出された - 即時終了

`40` ルートの場合 `0` と同じ

使用法

2G バイト (2^{31} バイト) 以上のファイルに対する `fsck` の動作については、`largefile`(5) のマニュアルページを参照してください。

ファイル

`/etc/default/fs` デフォルトのローカルファイルシステムタイプ。デフォルト値は `/etc/default/fs` 内の次のフラグに設定できる。例: `LOCAL=ufs`。

`LOCAL` FSType が指定されていない場合はコマンドのデフォルトファイルタイプ

`/etc/vfstab` 各ファイルシステム用のデフォルトのパラメータリスト

属性

次の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

`clri`(1M), `fsck_cachefs`(1M), `fsck_ufs`(1M), `fsdb_ufs`(1M), `fsirand`(1M), `fstyp`(1M), mkfs(1M), `mkfs_ufs`(1M), mountall(1M), newfs(1M), reboot(1m), `vfstab`(4), `attributes`(5), `largefile`(5)

警告

オペレーティングシステムはファイルシステムデータをバッファリングします。マウントされているファイルシステムに `fsck` を実行すると、オペレーティングシステムのバッファー内容がディスク上のものよりも古くなります。そのため、`fsck` を使用する場合は、ファイルシステムをマウント解除してください。それができない場合は、`fsck` の実行後に、システムを停止させ、ただちに再起動してください。それでも不十分な場合がしばしばあります。ファイルシステム上で `fsck` を実行した結果、ファイルシステムが変更された場合には、パニックが発生することがあります。

注意事項

このコマンドは、すべての *FSTypes* で使用できるわけではありません。

次のブロック型デバイスを使用するように選択した場合、2G バイトを超えるファイルシステム上で fsck を実行すると失敗します。

```
fsck /dev/dsk/c?t?d?s?
```

代わりに、次の raw (文字型特殊) デバイスを使用します。

```
fsck /dev/rdsk/c?t?d?s?
```

Solaris 9 以降、fsck はディスク上の拡張属性データを管理します (拡張ファイル属性については、fsattr(5) のマニュアルページを参照)。 拡張属性が設定されたファイルシステムを、属性を認識しないバージョンの Solaris (Solaris 9 より前のバージョン) にマウントすることはできますが、属性にアクセスすることはできません。fsck はファイルからそれらの属性を取り除き、lost+found に移します。属性が取り除かれたファイルシステムは、属性を認識するバージョンの Solaris 上でも完全に安定動作しますが、破壊しているファイルシステムとみなされてしまいます。この場合、属性を認識するバージョンの fsck を実行し、ファイルシステムを安定させてから、属性を認識する環境でそのファイルシステムを使用してください。

名前

fuser – ファイルまたはファイル構造を使用しているプロセスの特定

形式

```
/usr/sbin/fuser  [-c |  -d |  -f] [-nu] [-k |  -s sig]
 files [ [- ] [-c |  -d |  -f] [-nu] [-k |  -s sig] files]
  ...
```

機能説明

fuser は、引数として指定された *files* を使用しているプロセスのプロセス ID を表示します。

各プロセス ID には文字修飾詞が続きます。文字修飾詞は、プロセスがファイルをどのように使用しているかを示すもので、以下があります。

c　カレントディレクトリとしてファイルを使用している

m　mmap(2) によってマップされたファイルを使用している。詳細は mmap(2) のマニュアルページを参照

n　そのファイルに対して、非ブロッキング強制ロックを保持している

o　オープンファイルとしてファイルを使用している

r　ルートディレクトリとしてファイルを使用している

t　テキストファイルとしてファイルを使用している

y　制御端末としてファイルを使用している

ファイルシステムがマウントされているブロック型特殊デバイスについては、そのデバイス上のファイルを使用しているすべてのプロセスが表示されます。その他のタイプのファイル (テキストファイル、実行可能ファイル、ディレクトリ、デバイスなど) については、そのファイルを使用しているプロセスだけが報告されます。

fuser は、すべてのタイプのデバイスに対して、デバイスを開いている既知のカーネル消費者も表示します。カーネル消費者は、次のうちの 1 つの形式で表示されます。

[*module_name*]
[*module_name*,dev_path=*path*]
[*module_name*,dev=(*major*,*minor*)]
[*module_name*,dev=(*major*,*minor*),dev_path=*path*]

複数のファイルグループを指定する場合は、ファイルグループごとにオプションを指定できます。現在使用中のオプションを取り消す場合は、単独のダッシュを使用します。

プロセス ID は標準出力に単一行として出力されます。プロセス ID は空白で区切られ、1 つの復帰改行 (NEWLINE) で終了します。その他の出力はすべて、標準エラー出力に書き込まれます。

fuser はだれでも実行できますが、他のユーザーのプロセスを終了させることができるのは、スーパーユーザーだけです。

オプション

次のオプションを指定できます。

`-c` ファイルシステムのマウントポイントであるファイル、およびマウントされているファイルシステム内のすべてのファイルについて報告します。

`-d` 指定したマイナーノードと同じデバイスノードに関連するすべてのマイナーノードについて、デバイス使用状況を報告します。このオプションは、マウントされているファイルシステム内にあるファイルについてのファイル使用状況は報告しません。

`-f` マウントされているファイルシステム内のファイルではなく、指定したファイルについて報告します。

`-k` 各プロセスに `SIGKILL` シグナルを送ります。このオプションは各プロセスに kill コマンドを発行するので、強制終了メッセージがすぐに表示されないことがあります (`kill`(2) のマニュアルページを参照)。カーネルファイル消費者にはシグナルは送られません。

`-n` ファイルに対して非ブロッキング強制ロックを保持しているプロセスだけを表示します。

`-s` *sig* 各プロセスにシグナルを送ります。*sig* オプション引数には、`<signal.h>` ヘッダーで定義されている記号名の 1 つを指定するか、または 10 進整数のシグナル番号を指定します。*sig* が記号名で、`SIG` 接頭辞を付けずに、大文字小文字の区別なしで認識されます。`-k` オプションは `-s KILL` または `-s 9` と同じです。カーネルファイル消費者にはシグナルは送られません。

`-u` プロセス ID の後に、括弧で囲んだユーザーのログイン名を表示します。

使用例

例 1 マウントポイントおよびファイルについて報告する

次の例は、マウントポイントおよびマウントされているファイルシステム内のファイルについて報告します。

```
fuser -c /export/foo
```

例 2 マウントポイントおよびファイルについて報告するときの出力を制限する

次の例は、マウントポイントおよびマウントされているファイルシステム内のファイルについて報告し、その出力を非ブロッキング強制ロックを保持しているプロセスに限定します。

```
fuser -cn /export/foo
```

例 3　非ブロッキング強制ロックを保持しているプロセスへ SIGTERM を送信する

次のコマンドは、ファイル /export/foo/my_file に対して非ブロッキング強制ロックを保持しているすべてのプロセスに、SIGTERM を送ります。

```
fuser -fn -s term /export/foo/my_file
```

環境

fuser の実行に影響を与える環境変数、LANG、 LC_ALL、LC_CTYPE、 LC_MESSAGES、および NLSPATH については、environ(5) のマニュアルページを参照してください。

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

ps(1), mount(1M), kill(2), mmap(2), signal(3C), attributes(5), environ(5)

注意事項

fuser はシステムイメージのスナップショットで動作するため、fuser の実行中にファイルを使用し始めたプロセスを見逃してしまうことがあります。また、ファイルを使用していると報告されたプロセスが、fuser の実行中にファイルの使用を中止していることもあります。このような理由から、-k オプションの使用は勧められません。

名前

growfs – UFS ファイルシステムを非破壊的に拡張する

形式

```
/usr/sbin/growfs  [-M mount-point] [newfs-options]
 [raw-device]
```

機能説明

growfs は、ファイルシステムのスライスのサイズまで、マウントされているまたはマウントされていない UNIX ファイルシステム (UFS) を非破壊的に拡張します。

通常、ディスク領域は、最初メタデバイスにスライスを追加し、次に growfs コマンドを実行することで拡張されます。ミラーに領域を追加する場合は、各サブミラーを拡張してからファイルシステムを拡張することになります。

growfs は拡張時に、マウントされたファイルシステムを「書き込みロック」(lockfs(1M) を参照) します。ファイルシステムが書き込みロックされる時間は、ファイルシステムを段階的に拡張することによって、短くすることができます。たとえば、ファイルシステムを 1G バイトから 2G バイトに拡張する場合は、16M バイト単位で拡張することができます。この場合、-s オプションで、各段階ごとに新しいファイルシステムの合計サイズを指定してください。-s の引数は、セクター数で指定し、シリンダサイズの倍数にしてください。 注 : シリンダサイズとして 2 未満の値が指定された場合 には、ファイルシステムは拡張できません。ファイルシステムを拡張させるときに指定できるオプションの詳細については、newfs(1M) マニュアルページを参照してください。

growfs はファイルシステムの拡張時に mkfs と同じ情報を表示します。

growfs が異常終了した場合は、ファイルシステムをマウント解除し、fsck コマンドを実行して失われた空き領域を復元するか、または growfs コマンドをもう一度実行します。

オプション

次のどのオプションを実行する場合でも、スーパーユーザーになる必要があります。

-M mount-point
: 拡張されるファイルシステムは *mount-point* にマウントされています。ファイルシステムのロック (lockfs) が使用されます。

newfs-options
: これらのオプションについては、newfs のマニュアルページを参照してください。

raw-device
: /dev/mr/rdsk または /dev/rdsk にそれぞれ存在する、raw メタデバイスまたは raw 特殊デバイスの名前を指定します。これには、ファイルシステムを拡張させるディスクスライスが含まれます。

使用例

例 1　/export ファイルシステムでの非メタデバイススライスの拡張

以下は、/export ファイルシステムで非メタデバイススライスを拡張する例です。この例では、既存のスライス /dev/dsk/c1t0d0s3 がメタデバイスに変換されるので、追加のスライスを連結できます。

```
# metainit -f d8 2 1 c1t0d0s3 1 c2t0d0s3
# umount /export
```

例 2　/export を新しいメタデバイスに対応付ける

/etc/vfstab ファイルを編集して、/export のエントリを新しく定義されたメタデバイス d8 に変更します。

```
# mount /export
# growfs -M /export /dev/md/rdsk/d8
```

この例は、-f オプション付きで metainit コマンドを実行することで開始し、新しい連結メタデバイス d8 を強制的に作成します。d8 は、既存のスライス /dev/dsk/c1t0d0s3 と新しいスライス /dev/dsk/c2t0d0s3 で構成されています。次に、/export 上のファイルシステムのマウントを解除する必要があります。/etc/vfstab ファイルを編集し、/export のエントリを、スライス名ではなく、新しく定義したメタデバイス名に変更します。ファイルシステムを再度マウントした後、growfs コマンドをファイルシステムを拡張するために実行します。 ファイルシステムは、growfs が完了するとメタデバイス全体に広がります。-M オプションによって growfs コマンドはマウントされたファイルシステムを拡張します。拡張の間は、growfs がファイ ルシステムのロックを解除するまで /export への書き込みアクセスは一時停止されます。読み取りアクセスはその影響を受けませんが、ロック中はアクセス時刻は記録されません。

例 3　/export ファイルシステムの動的拡張

以下の例は、前述の例の一部を使用しています。ここでは、メタデバイス d8 にマウントされた /export ファイルシステムは動的に拡張されます。

```
# metattach d8 c0t1d0s2
# growfs -M /export /dev/md/rdsk/d8
```

この例は、metattach コマンドを使用することで開始し、新しいスライス /dev/dsk/c0t1d0s2 を既存のメタデバイス d8 の最後に動的に連結させます。次に、growfs コマンドは、マウントポイント /export を raw メタデバイス /dev/md/rdsk/d8 上に拡張します。ファイルシステムは、growfs が完了するとメタデバイス全体に広がります。拡張の間は、growfs がファイルシステムのロックを解除するまで /export への書き込みアクセスは一時停止されます。読み取りアクセスはその影響を受けませんが、ロック中はアクセス時刻は記録されません。

例 4　マウントされたファイルシステムの既存のミラーへの拡張

以下の例は、マウントされたファイルシステム /files を既存のミラー d80 に拡張するものです。このミラーには 2 つのサブミラー d9 と d10 が含まれています。

```
# metattach d9 c0t2d0s5
# metattach d10 c0t3d0s5
# growfs -M /files /dev/md/rdsk/d80
```

例 4　マウントされたファイルシステムの既存のミラーへの拡張　(続き)

この例では、metattach コマンドは各サブミラーに新しいスライスを動的に連結します。metattach コマンドは各サブミラーに対して実行する必要があります。ミラーは最後のサブミラーが動的に連結されると、自動的に拡張されます。ミラーは最小のサブミラーのサイズまで拡張されます。次に、growfs コマンドでファイルシステムを拡張します。growfs コマンドはマウント先 /files のraw メタデバイス /dev/md/rdsk/d80 を拡張します。ファイルシステムは growfs コマンドが完了するとミラー全体に広がります。拡張の間は、growfs がファイルシステムのロックを解除するまでファイルシステムへの書き込みアクセスは一時停止されます。読み取りアクセスはその影響を受けませんが、ロック中はアクセス時刻は記録されません。

終了ステータス

以下の終了ステータスが返されます。

0　正常終了

>0　エラーが発生した

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

fsck(1m), lockfs(1M), mkfs(1M), metattach(1M), newfs(1M), attributes(5)

『Solaris ボリュームマネージャの管理』

制限事項

(マウントされているされていないを問わず) UFS ファイルシステムだけが growfs コマンドを使用して拡張できます。ファイルシステムは一度拡張されると、そのサイズを小さくすることはできません。 以下の場合には、ファイルシステムを拡張することはできません。acct が起動されており、アカウンティングファイルがターゲットデバイス上にある場合。C2 セキュリティが有効で、ロギングファイルがターゲットのファイルシステム上にある場合。ターゲットのファイルシステム上に、ローカルの swap ファイルがある場合。ファイルシステムがルート (/)、/usr、swap のいずれかである場合。

名前

halt, poweroff – プロセッサの停止

形式

```
/usr/sbin/halt  [-dlnqy]

/usr/sbin/poweroff  [-dlnqy]
```

機能説明

`halt` および `poweroff` ユーティリティは、保留されている情報をディスクに書き出してから、プロセッサを停止させます。`poweroff` ユーティリティは、可能であれば、マシンの電源を自動的に切断します。

`halt` および `poweroff` ユーティリティは通常、システム停止をシステムログデーモン syslogd(1M) に送信し、ログインアカウントファイル `/var/adm/wtmpx` に停止情報を記録します。ただし、`-n` または `-q` オプションが指定されている場合、これらのアクションは禁止されます。

オプション

次のオプションを指定できます。

`-d` 再起動の前にシステムクラッシュダンプを強制的に実行します。システムクラッシュダンプの設定については、`dumpadm`(1M) のマニュアルページを参照してください。

`-l` `halt` を実行したユーザーに関するメッセージを、システムログデーモン syslogd(1M) に送信しないようにします。

`-n` 停止前の `sync`(1M) を行いません。

`-q` 即時停止。通常の停止手続きを実行しません。

`-y` ダイヤルアップ端末からでも、システムを停止します。

ファイル

`/var/adm/wtmpx` ユーザーアクセスと管理情報の履歴

属性

次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

`dumpadm`(1M), init(1M), reboot(1m), shutdown(1M), `sync`(1M), syslogd(1M), `inittab`(4), `attributes`(5), `smf`(5)

注意事項

`halt` ユーティリティおよび `poweroff` ユーティリティは、`smf`(5) サービスをクリーンな状態で停止しません。`/etc/rc`*num*`.d` のスクリプトを実行するか、`inittab`(4) の停止アクションを実行します。システムサービスを完全に停止させるためには、shutdown(1M) を使用するか、または init(1M) を使用して Solaris システムを再起動してください。

名前　　ifconfig – ネットワークインタフェースのパラメータの構成

形式

```
/sbin/ifconfig interface [address_family] [address
 [/prefix_length] [dest_address]] [addif address
 [/prefix_length]] [removeif address [/prefix_length]]
 [arp | -arp] [auth_algs authentication algorithm]
 [encr_algs encryption algorithm] [encr_auth_algs authentication algorithm]
 [auto-revarp] [broadcast address] [deprecated
 | -deprecated] [preferred | -preferred] [destination
 dest_address] [ether [address]] [ [failover]
 | [-failover]] [group [ [name] | ""]] [index {if_index}]
 [metric n] [modlist] [modinsert mod_name@pos] [modremove mod_name@pos]
 [mtu n] [netmask mask] [plumb] [unplumb] [private
 | -private] [nud | -nud] [set [address] [/netmask]]
 [ [standby] | [-standby]] [subnet subnet_address]
 [tdst tunnel_dest_address] [token address/prefix_length]
 [tsrc tunnel_src_address] [trailers | -trailers]
 [up] [down] [usesrc [name | none]] [xmit | -xmit]
 [encaplimit n | -encaplimit] [thoplimit n] [router
 | -router] [zone zonename | -zone]

/usr/sbin/ifconfig interface [address_family] [address
 [/prefix_length] [dest_address]] [addif address
 [/prefix_length]] [removeif address [/prefix_length]]
 [arp | -arp] [auth_algs authentication algorithm]
 [encr_algs encryption algorithm] [encr_auth_algs authentication algorithm]
 [auto-revarp] [broadcast address] [deprecated
 | -deprecated] [preferred | -preferred] [destination
 dest_address] [ether [address]] [ [failover]
 | [-failover]] [group [ [name] | ""]] [index {if_index}]
 [metric n] [modlist] [modinsert mod_name@pos] [modremove mod_name@pos]
 [mtu n] [netmask mask] [plumb] [unplumb] [private
 | -private] [nud | -nud] [set [address] [/netmask]]
 [ [standby] | [-standby]] [subnet subnet_address]
 [tdst tunnel_dest_address] [token address/prefix_length]
 [tsrc tunnel_src_address] [trailers | -trailers]
 [up] [down] [usesrc [name | none]] [xmit | -xmit]
 [encaplimit n | -encaplimit] [thoplimit n] [router
 | -router] [zone zonename | -zone]

/sbin/ifconfig interface {auto-dhcp | dhcp} [primary]
 [wait seconds] drop | extend | inform | ping
 | release | start | status

/usr/sbin/ifconfig interface {auto-dhcp | dhcp}
 [primary] [wait seconds] drop | extend | inform
 | ping | release | start | status
```

機能説明　　コマンド ifconfig は、ネットワークインタフェースにアドレスを割り当てたり、ネットワークインタフェースのパラメータを構成したりするために使用されます。ブート時には、この ifconfig コマンドを使ってマシン上に存在する各インタ

フェースのネットワークアドレスを定義する必要があります。また、その後も、このコマンドを使って特定のインタフェースのアドレスやその他の動作パラメータを定義し直すことができます。オプションが 1 つも指定されなかった場合、`ifconfig` は、ネットワークインタフェースの現在の構成を表示します。アドレスファミリが指定された場合、`ifconfig` は、そのアドレスファミリに固有の詳細のみを報告します。ネットワークインタフェースの構成を変更できるのは、特権ユーザーだけです。中括弧 (`{ }`) で囲まれたオプションは、そのうちのどれか 1 つのオプションを指定することを示します。

DHCP 構成

このコマンドの 3 番目と 4 番目の形式は、インタフェースの動的ホスト構成プロトコル (「DHCP」) 構成を制御するために使用されます。DHCP は、アドレスファミリが `inet` のインタフェースでのみ使用できます。このモードでは、DHCP クライアントデーモンである `dhcpagent`(1M) の動作を制御する目的で、`ifconfig` が使用されます。あるインタフェースがいったん `start` オペランド経由で DHCP の制御下に置かれたら、通常の操作では、そのインタフェースのアドレスや特性を `ifconfig` を使って変更するべきではありません。DHCP 配下の特定のインタフェースのアドレスが変更されると、`dhcpagent` はそのインタフェースを制御対象外にします。

オプション

サポートしているオプションは、以下のとおりです。

`-addif` *address*
: 指定された物理インタフェース上に、次の未使用の論理インタフェースを作成します。物理インタフェースがマルチパスグループに属している場合、その同じグループ内の異なる物理インタフェースにその論理インタフェースが追加されることがあります。

`-arp`
: ネットワークレベルのアドレスとリンクレベルのアドレスとの間のマッピングを行う際に、アドレス解決プロトコル (「ARP」) の使用を有効にします (デフォルト)。これは現時点では、IPv4 アドレスと MAC アドレス間のマッピング用として実装されています。

`-arp`
: ARP の使用を無効にします。

`-auth_algs` *authentication algorithm*
: 特定のトンネルに対し、指定された認証アルゴリズムを使って IPsec AH を有効にします。このアルゴリズムは、数字、アルゴリズム名のどちらで指定してもかまいません。どのアルゴリズムでもかまわないことを示す *any* も使用できます。すべての IPsec トンネルプロパティーは、同一コマンド行に指定する必要がありま

す。トンネルのセキュリティーを無効にするには、`auth_alg` に *none* を指定します。

`-auto-dhcp`

このインタフェースのアドレスを、DHCP を使って自動的に取得します。このオプションには、`dhcp` という名前の、完全に等価な別名があります。

primary

このインタフェースを `primary` として定義します。このインタフェースは、クライアント全体の構成データを配信するための優先インタフェースとして定義されます。主インタフェースになれるインタフェースは、一度に 1 つだけです。その後、別のインタフェースが主インタフェースとして選択された場合、以前の主インタフェースはそれに置き換えられます。クライアントワークステーションのブート完了後に特定のインタフェースを主インタフェースとして指定することは、あまり意味がありません。これは、多くのアプリケーションはすでに起動されており、以前の主インタフェースから読み取ったデータに基づいて構成されているからです。

wait *seconds*

`ifconfig` コマンドは、処理が完了するか、指定された時間が経過するまで待機します。実際の待機時間はどちらか早いほうになります。待機時間が指定されず、かつ処理内

容がすぐに完了できない性質のものであった場合、`ifconfig` は 30 秒間、要求された処理が完了するのを待ちます。シンボリック値 `forever` も、文字どおりの意味で使用できます。

drop
: 指定されたインタフェースを DHCP の制御対象外にします。さらに、IP アドレスを 0 に設定し、そのインタフェースを「down」としてマークします。

extend
: インタフェースの IPv4 アドレスのリースを延長しようとします。これは必須ではありません。リースが期限切れになる前に、エージェントによって自動的にリースが延長されるからです。

inform
: DHCP からネットワーク構成パラメータを取得します。その際、IP アドレスのリースは取得しません。これは、DHCP 以外の機構を使って IP アドレスを取得する場合に役立ちます。

ping
: 指定されたインタフェースが DHCP の制御下にあるかどうか、つまり、そのインタフェースが DHCP エージェントによって管理されており、かつ正しく動作しているかをチェックします。終了状態 `0` は、成功したことを意味します。指定さ

れたインタフェースが複数のインタフェースを表している場合にこのサブコマンドを使用しても、意味がありません。

release	インタフェースの IPv4 アドレスを解放し、そのインタフェースを「down」としてマークします。
start	インタフェース上で DHCP を開始します。
status	インタフェースの DHCP 構成状態を表示します。

`-auto-revarp`

逆アドレス解決プロトコル (「RARP」) を使ってこのインタフェースのアドレスを自動的に取得します。IPoIB (IP over InfiniBand) など、RARP をサポートしていないインタフェースでは、この処理は失敗します。

`-broadcast` *address*

IPv4 専用。ネットワークへのブロードキャストを示すために使用するアドレスを指定します。デフォルトのブロードキャストアドレスは、ホスト部分がすべて 1 であるようなアドレスです。ブロードキャストの値として「+」(プラス記号) を指定した場合、ブロードキャストアドレスは、(新しくなった可能性のある) アドレスとネットマスクに適したデフォルトにリセットされます。`ifconfig` の引数は左から右に解釈されます。したがって、

```
example% ifconfig -a netmask + broadcast +
```

と

```
example% ifconfig -a broadcast + netmask +
```

では、インタフェースのブロードキャストアドレスに異なる値が割り当てられる可能性があります。

`-deprecated`	論理インタフェースを非推奨としてマークします。非推奨のインタフェースに関連付けられたアドレスが送信パケットの発信元アドレスとして使用されるのは、そのインタフェース上で利用可能なアドレスがほかに存在しない場合と、アプリケーションがそのアドレスに明示的にバインドした場合だけです。状態表示では、`DEPRECATED` がフラグの一部として表示されます。`ifconfig` でサポートされるフラグについては、「インタフェースフラグ」を参照してください。
`-deprecated`	論理インタフェースを「非推奨でない」としてマークします。そのようなインタフェースに関連付けられたアドレスは、送信パケットの発信元アドレスとして使用できます。
`-preferred`	論理インタフェースを優先としてマークします。このオプションは IPv6 アドレスでのみ有効です。優先論理インタフェースに割り当てられたアドレスは、システム上で構成されたほかのどのアドレスよりも、発信元アドレスとして優先されます。ただし、そのアドレスが着信先アドレスから見て不適切なスコープを持つ場合はその限りではありません。優先アドレスは、それらがどの物理インタフェースに割り当てられているかにかかわらず、常に発信元アドレスとして使用されます。たとえば、ループバックインタフェース上で優先発信元アドレスを構成し、そのアドレスの到達可能性を、経路指定プロトコルを使って通知できます。
`-preferred`	論理インタフェースを非優先としてマークします。
`-destination` *dest_address*	ポイントツーポイントインタフェースの着信先アドレスを設定します。
`-dhcp`	このオプションは、オプション `auto-dhcp` の別名です。

`-down`

論理インタフェースを「down」としてマークします。(つまり、`IFF_UP` ビットをオフにします)。特定の論理インタフェースが「down」としてマークされると、システムは、そのインタフェースに割り当てられたアドレスを送信パケットの発信元アドレスとして使用しようとしなくなるほか、そのアドレス宛の受信パケットをこのホスト宛のものとして認識しなくなります。さらに、ある物理インタフェース上のすべての論理インタフェースが「down」になった場合、その物理インタフェースそのものが無効になります。

特定の論理インタフェースが停止すると、`route`(1M) コマンドの `-ifp` オプションまたは `route`(7P) ソケットの `RTA_IFP` を使ってそのインタフェースを出力として指定した経路がすべて、転送テーブルから削除されます。`RTF_STATIC` とマークされた経路はインタフェース復旧時にテーブルに戻されますが、`RTF_STATIC` とマークされていない経路は単に削除されます。

特定のゲートウェイアドレスに到達するために使用可能な論理インタフェースのすべてが停止した場合 (直前の段落で説明したインタフェースオプションなしで指定された場合)、その影響を受けるゲートウェイ経路は、`RTF_BLACKHOLE` フラグが設定されている場合と同様に処理されます。一致するすべてのパケットは、そのゲートウェイに到達できずに破棄されます。

`-encaplimit` *n*

インタフェースのトンネルカプセル化制限を n に設定します。このオプションを使用できるのは、IPv6 内 IPv4 トンネルと IPv6 内 IPv6 トンネルに対してのみです。トンネルカプセル化制限は、特定のパケットが任意のトンネルを出る前にさ

らにいくつのトンネルに入れるか (つまりトンネルのネストレベル) を制御します。

`-encaplimit`

トンネルカプセル化制限の生成を無効にします。このオプションを使用できるのは、IPv6 内 IPv4 トンネルと IPv6 内 IPv6 トンネルに対してのみです。

`-encr_auth_algs` *authentication algorithm*

特定のトンネルに対し、指定された認証アルゴリズムを使って IPsec ESP を有効にします。これは、数字、アルゴリズム名のどちらで指定してもかまいません。どのアルゴリズムでもかまわないことを示す `any` や `none` も使用できます。ESP 暗号化アルゴリズムは指定されたがその認証アルゴリズムが指定されなかった場合、ESP 認証アルゴリズムのデフォルト値は `any` になります。

`-encr_algs` *encryption algorithm*

特定のトンネルに対し、指定された暗号化アルゴリズムを使って IPsec ESP を有効にします。これは、数字、アルゴリズム名のどちらで指定してもかまいません。すべての IPsec トンネルプロパティーは同一コマンド行に指定する必要があるので、注意してください。トンネルセキュリティーを無効にするには、`encr_alg` の値として `none` を指定します。ESP 認証アルゴリズムは指定されたがその暗号化アルゴリズムが指定されなかった場合、ESP 暗号化アルゴリズムのデフォルト値は `null` になります。

`-ether` [*address*]

アドレスが指定されなかった場合、ユーザーがスーパーユーザーであるか、あるいは対象デバイスをオープンできるだけの十分な特権を備えていれば、現在の Ethernet アドレス情報を表示します。

それ以外の場合、ユーザーがスーパーユーザーであるか十分な特権を備えていれば、インタフェースの Ethernet アドレスを *address* に設定します。このアドレスは Ethernet アドレスであり、*x:x:x:x:x:x* として表現されます。ここで、*x* は 0 か

	ら FF までの 16 進数です。同様に、IPoIB (IP over InfiniBand) インタフェースの場合、このアドレスは、`0` から `FF` までの 16 進数がコロンで区切られた 20 バイトの文字列になります。 Ethernet インタフェースカードの中には、固有のアドレスを持つものもあります (すべてではない)。固有のアドレスを持たないカードを使用する場合には、IEEE 802.3 仕様の 3.2.3(4) セクションを参照し、ローカル管理アドレス空間の定義を確認してください。マルチパスグループの使用は、固有のアドレスを持つカードの場合だけに限定するべきです (「マルチパスグループ」を参照)。
`-failover`	論理インタフェースを非フェイルオーバーインタフェースとしてマークします。非フェイルオーバー論理インタフェースに割り当てられたアドレスに対しては、そのインタフェースで障害が発生してもフェイルオーバーが実行されません。状態表示では、`NOFAILOVER` がフラグの一部として表示されます。
`-failover`	論理インタフェースをフェイルオーバーインタフェースとしてマークします。そのようなインタフェースに割り当てられたアドレスに対しては、インタフェースで障害が発生した際にフェイルオーバーが実行されます。状態表示では、`NOFAILOVER` がフラグの一部として表示されません。
`-group` [*name* |`""`]	*name* で指定されたマルチパスグループ内に論理インタフェースを挿入します。グループからインタフェースを削除するには、NULL 文字列 `""` を使用します。ID 0 の論理インタフェース上で呼び出された場合、その状態表示にグループ名が含まれます。
`-index` *n*	インタフェースのインタフェースインデックスを変更します。*n* の値は、ほかのインタフェースで使用されていないイ

ンタフェースインデックス (*if_index*) でなければいけません。*if_index* は 0 でない正数であり、システム上のネットワークインタフェースを一意に識別します。

`-metric` *n*

インタフェースの経路指定メトリックを *n* に設定します。値が指定されなかった場合のデフォルトは、`0` になります。経路指定メトリックは、経路指定プロトコルによって使用されます。メトリックが高いほど、その経路は好まれません。メトリックは、着信先となるネットワークまたはホストへの加算ホップとしてカウントされます。

`-modinsert` *mod_name@pos*

mod_name という名前のモジュールを、デバイスのストリーム内の位置 *pos* に挿入します。この位置は、ストリームの先頭からの相対位置です。位置 `0` は、ストリームの先頭のすぐ下を意味します。

`modlist` オプションの例に基づいて次のコマンドを使用すると、`ipqos` という名前のモジュールが、`ip` モジュールの下、`firewall` モジュールの上に挿入されます。

```
example% ifconfig eri0 modinsert ipqos@2
```

このあと、デバイスのストリーム内のすべてのモジュールを一覧表示した結果を、次に示します。

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 ipqos
3 firewall
4 eri
```

`-modlist`

デバイスのストリーム内のすべてのモジュールを一覧表示します。

次の例では、デバイスのストリーム内のすべてのモジュールを一覧表示しています。

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 firewall
3 eri
```

-modremove *mod_name@pos*

mod_name という名前のモジュールを、デバイスのストリーム内の位置 *pos* から削除します。この位置は、ストリームの先頭からの相対位置です。

modinsert オプションの例に基づいて次のコマンドを使用すると、ipqos モジュール挿入後のストリームから firewall モジュールが削除されます。

```
example% ifconfig eri0 modremove firewall@3
```

このあと、デバイスのストリーム内のすべてのモジュールを一覧表示した結果を、次に示します。

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 ipqos
3 eri
```

ip や tun などのコア IP スタックモジュールは削除できません。

-mtu *n*

インタフェースの最大転送単位を *n* に設定します。多くのネットワークタイプでは、mtu には上限があります。たとえば、Ethernet の場合は 1500 です。このオプションを使用すると、対象のインタフェース上に FIXEDMTU フラグが設定されます。

-netmask *mask*

IPv4 専用。ネットワークのサブネットワークへのサブ分割用としてアドレスの何ビットを確保するかを指定します。マスクにはローカルアドレスのネットワーク部とサブネット部が含まれます。これは、アドレスのホストフィールドから取得されます。マスクでは、32 ビットアドレス内でネットワーク部とサブネット部用として使用するべきビット位置に 1

が、ホスト部に対するビット位置に 0 が、それぞれ含まれています。マスクには少なくとも標準ネットワーク部を含めてください。また、サブネットフィールドはネットワーク部に隣接するようにしてください。マスクの指定方法には次の 4 つがあります。

1. 0x で始まる単一の 16 進数の使用
2. ドット表記アドレスの使用
3. 「+」(プラス記号) アドレスの使用
4. ネットワークデータベース `networks`(4) 内に収められた擬似ホスト名/擬似ネットワーク名の使用

ネットマスク値として「+」(プラス記号) が指定された場合、`netmasks`(4) データベース内でマスクの検索が行われます。この検索では、データベース内で一致する最長のネットマスクを見つけるために、インタフェースの IPv4 アドレスをキーとして検索を開始し、アドレスのより多くの下位ビットを順次マスキングしていきます。この反復的な検索手法により、1 つのネットワーク番号内でさまざまな長さのサブネットマスクが使用されている場合に `netmasks`(4) データベースをネットマスク指定用として使用できることが保証されます。

擬似ホスト名 / 擬似ネットワーク名がネットマスク値として指定された場合、`hosts` または `networks` データベース内でネットマスクデータが検索されます。まず、`gethostbyname`(3NSL) を使って名前の検索が行われます。名前がそこに見つからなかった場合には、`getnetbyname`(3SOCKET) で名前の検索が行われます。これらのインタフェースは通常、実際の値を取得するのにどのデータストア (複数可) を使用するべきかを、`nsswitch.conf`(4) を使って判定します。

	inet と inet6 のどちらでも、*mask* が表現しているのと同じ情報を、*address* パラメータに付随する *prefix_length* として指定することができます。
-nud	特定のポイントツーポイントインタフェース上で近傍到達不可能性検出機構を有効にします。
-nud	特定のポイントツーポイントインタフェース上で近傍到達不可能性検出機構を無効にします。
-plumb	物理インタフェース名に関連付けられたデバイスをオープンし、IP がデバイスを使用する際に必要となるストリームを設定します。このコマンドを論理インタフェース名とともに使用した場合、その名前を持つ論理インタフェースが作成されます。インタフェースの plumb は、IPv4 用と IPv6 用とで別々に行う必要があります。ifconfig コマンドが IPv4、IPv6 のどちらに適用されるかは、*address_family* パラメータによって決まります。 plumb を行う前のインタフェースは、ifconfig -a コマンドの出力には現れません。
-private	指定された論理インタフェースは通知するべきでないことを、in.routed 経路指定デーモンに伝えます。
-private	通知されないインタフェースを表します。
-removeif *address*	指定された物理インタフェース上の論理インタフェースのうち、指定された *address* に一致するものを削除します。そのインタフェースが特定のマルチパスグループに属している場合、そのグループ内でそのアドレスを保持している物理インタフェースから、その論理インタフェースが削除されます。

`-router`	インタフェース上の IP 転送を有効にします。有効にした場合、そのインタフェースは `ROUTER` としてマークされ、そのインタフェースへの、およびそのインタフェースからの、IP パケットの転送が行えるようになります。
`-router`	インタフェース上の IP 転送を無効にします。そのインタフェースへの、およびそのインタフェースからの、IP パケットの転送は行われません。
`-set`	特定の論理インタフェースの *address* または *prefix_length*、あるいはその両方を設定します。
`-standby`	物理インタフェースを待機インタフェースに指定します。あるマルチパスグループに属するインタフェースを `STANDBY` としてマークした場合、グループ内のほかのインタフェースで障害が発生してネットワークアクセスがこの待機インタフェースにフェイルオーバーされないかぎり、パケット送信用として選択されることはありません。 状態表示では「`STANDBY, INACTIVE`」と表示されますが、これは、そのインタフェースが待機インタフェースであり、かつアクティブでないことを意味します。`IFF_INACTIVE` がクリアされるのは、同じマルチパスグループに属するほかのインタフェースからこのインタフェースへのフェイルオーバーが発生した場合です。いったんフェイルバックが発生すると、状態表示は `INACTIVE` に戻ります。
`-standby`	このインタフェースの待機をオフにします。
`-subnet`	インタフェースのサブネット *address* を設定します。
`-tdst` *tunnel_dest_address*	トンネルの着信先アドレスを設定します。このアドレスをトンネルの `dest_address` と同じ値にするべきではあ

りません。そのようなトンネル経由でシステムを離れるパケットは存在しないからです。

`-thoplimit` *n*

トンネルインタフェースのホップ制限を設定します。IPv4 内 IPv6 および IPv4 内 IPv4 トンネルの場合、このホップ制限値は IPv4 ヘッダー内の TTL として使用されます。IPv6 内 IPv6 および IPv6 内 IPv4 トンネルの場合、このホップ制限値は IPv6 ヘッダー内のホップ制限として使用されます。

`-token` *address*/*prefix_length*

アドレス自動構成で使用される、インタフェースの IPv6 トークンを設定します。

```
example% ifconfig eri0 inet6 token ::1/64
```

`-trailers`

このフラグは以前、一部のリンクレベルで `inet` パケットの非標準カプセル化を引き起こしていました。このリリースに付属するドライバは、このフラグを使用しません。これは互換性のために提供されていますが、無視されます。

`-trailers`

「trailer」リンクレベルカプセル化の使用を無効にします。

`-tsrc` *tunnel_src_address*

トンネルの発信元アドレスを設定します。これは、外側のカプセル化している IP ヘッダーの発信元アドレスです。これは、`ifconfig` ですでに構成済みの別のインタフェースのアドレスである必要があります。

`-unplumb`

この物理インタフェース名に関連付けられたデバイスと、IP がデバイスを使用できるように `ifconfig` によって設定されたすべてのストリームをクローズします。論理インタフェース名とともに使用された場合、その論理インタフェースがシステムから削除されます。このコマンドの実行後、`ifconfig -a` の出力にそのデバイス名が表示されなくなります。

`-up`

論理インタフェースを「up」としてマークします。論理インタフェースに最初の

アドレスを割り当てる際には、これが自動的に起こります。`ifconfig down` のあとに `up` オプションを使用すると、インタフェースが有効になり、ハードウェアが再初期化されます。

`-usesrc [ *name* | none ]`

特定の物理インタフェースを発信元アドレス選択用として指定します。キーワード `none` を使用した場合、それまでの選択がすべてクリアされます。

アプリケーションが `bind`(3SOCKET) を使って 0 以外の発信元アドレスを選択しなかった場合、送信インタフェースとアドレス選択規則 (`ipaddrsel`(1M) を参照) に基づいて、システムが適切な発信元アドレスを選択します。

`usesrc` が指定され、そこで指定されたインタフェースが転送テーブル内で出力用として選択されていた場合、システムは発信元アドレスの選択時に、まず、その指定された物理インタフェースとそれに関連付けられた論理インタフェースを調べます。転送テーブル内に使用可能なアドレスが 1 つも見つからなかった場合には、通常の選択規則が適用されます。たとえば、次のように入力したとします。

```
# ifconfig eri0 usesrc vni0
```

...ここで、`vni0` にアドレス 10.0.0.1 が割り当てられているとすると、システムは、`eri0` 経由で送信されるローカル接続からのすべてのパケットに対し、10.0.0.1 を発信元アドレスとして優先的に使用します。その他の例については、「使用例」セクションを参照してください。

どのような物理インタフェースでも (ループバックでさえも) 指定できますが、仮想 IP インタフェース (`vni`(7d) を参照) を指定することもできます。仮想 IP インタフェースは、どの物理ハードウェアにも関連付けられていないため、ハードウェア障害の影響を一切受けませ

ん。1 つの仮想インタフェース上にホストされた発信元アドレスを、任意の数の物理インタフェースで使用するよう指定できます。これにより、経路指定に基づくマルチパス化の構成が単純化されます。物理インタフェースの 1 つで障害が発生した場合、残りの正常に機能している物理インタフェースのいずれかを介して通信が継続されます。このシナリオは、仮想インタフェース上にホストされたアドレスの到達可能性が、経路指定プロトコルを使用するなど、何らかの方法を使って通知されていることを前提にしています。

`ifconfig` の `preferred` オプションは、すべてのインタフェースに適用されるため、`usesrc` オプションよりも粗粒度であると言えます。それは、`usesrc` や `setsrc` (route サブコマンド) によって、その順番で上書きされます。

`usesrc` オプションは、`ifconfig` の IP マルチパス化オプションである `group` や `standby` と、互いに排他の関係にあります。つまり、あるインタフェースがすでに特定の IP マルチパスグループの一部になっているか、あるいは `standby` インタフェースとして指定されている場合、そのインタフェースを `usesrc` オプションで指定することはできません。その逆も同様です。IP マルチパス化の詳細については、`in.mpathd`(1M) と『System Administration Guide: IP Services』を参照してください。

`-xmit`

特定の論理インタフェースのパケット送信機能を有効にします。これは、論理インタフェースが「up」状態にある場合のデフォルト動作です。

`-xmit`

特定のインタフェース上でパケット送信を無効にします。そのインタフェースは、パケットの受信は引き続き行います。

-zone *zonename*	論理インタフェースをゾーン *zonename* 内に配置します。指定されたゾーンは、カーネル内でアクティブになっており、準備完了か実行中の状態になっていなければいけません。ゾーンが停止または再起動すると、インタフェースは unplumb されます。
-zone	IP インタフェースを大域ゾーン内に配置します。これはデフォルトです。

オペランド　ここでは、*interface* オペランドとそれに影響を与えるアドレスパラメータについて説明します。

interface　文字列。次のいずれかの形式になります。

- *name physical-unit*。例: `eri0`、`ce1`
- *name physical-unit*:*logical-unit*。例: `eri0:1`
- `ip.tun`*N* または `ip6.tun`*N*。トンネル用

ダッシュ (-) で始まるインタフェース名は、インタフェース群を表すオプションがいくつか組み合わさったものとして解釈されます。そのような場合、`-a` は必ずオプションに含まれている必要がありますが、それ以外の次の追加オプションは、どれでも任意の順序で追加できます。これらのインタフェース名のいずれかが指定された場合、それ以降のコマンドは、条件に一致するすべてのインタフェースに対して適用されます。

`-a`	指定されたアドレスファミリのすべてのインタフェースに対して、コマンドを適用します。コマンド行からも `/etc/default/inet_type` 経由でもアドレスファミリが指定されなかった場合、すべてのアドレスファミリが選択されます。
`-d`	システム内のすべての「down」インタフェースに対して、コマンドを適用します。
`-D`	DHCP (動的ホスト構成プロトコル) の制御下にないすべてのインタフェースに対して、コマンドを適用します。
`-u`	システム内のすべての「up」インタフェースに対して、コマンドを適用します。
`-Z`	ユーザーのゾーン内のすべてのインタフェースに対して、コマンドを適用します。

-4 すべての IPv4 インタフェースに対して、コマンドを適用します。

-6 すべての IPv6 インタフェースに対して、コマンドを適用します。

address_family アドレスファミリを指定するには、*address_family* パラメータを使用します。`ifconfig` コマンドが現時点でサポートしているファミリは、次のとおりです。`inet` と `inet6`。アドレスファミリが指定されなかった場合のデフォルトは、`inet` です。

`ifconfig` は、インタフェースの情報を表示する際に、`/etc/default/inet_type` ファイル内の `DEFAULT_IP` 設定に従います。`DEFAULT_IP` が `IP_VERSION4` に設定されていた場合、`ifconfig` は、IPv6 インタフェースに関する情報を省略します。ただし、ユーザーがアドレスファミリ (`inet`、 `inet6` のいずれか) を、`ifconfig` コマンド行で明示的に指定した場合、そのコマンド行のほうが `DEFAULT_IP` 設定よりも優先されます。

address IPv4 ファミリ (`inet`) の場合、*address* は、ホスト名データベース (`hosts`(4) を参照) 内またはネットワーク情報サービス (NIS) のマップ `hosts` 内に存在しているホスト名、インターネット標準の「ドット表記」で表現された IPv4 アドレス、のいずれかになります。

IPv6 ファミリ (`inet6`) の場合、*address* は、ホスト名データベース (`ipnodes`(4) を参照) 内またはネットワーク情報サービス (NIS) のマップ ipnode 内に存在するホスト名、インターネット標準のコロン区切り 16 進形式で表現された IPv6 アドレス、のいずれかになります。後者は *x:x:x:x:x:x:x:x* として表現されます。ここで、*x* は `0` から `FFFF` までの 16 進数です。

prefix_length IPv4 ファミリと IPv6 ファミリ (`inet` と `inet6`) の場合、*prefix_length* は、0 からアドレス内のビット数までの数値です。アドレス内のビット数は、`inet` の場合は 32、`inet6` の場合は 128 です。*prefix_length* は、ネットマスク内の先頭のセットビットの数を表します。

dest_address *address* パラメータのほかに *dest_address* パラメータが指定された場合、そのアドレスは、ポイントツーポイントリンクの他端に位置する対応するインタフェースのアドレスを表します。

tunnel_dest_address 構成中のトンネル以外の特定のインタフェースから到達可能、または到達可能と予想されるアドレス。これによって、

トンネルパケットの送信先をトンネルに指示します。このアドレスは、構成中のインタフェース着信先アドレスと同じであってはいけません。

tunnel_src_address `ifconfig` を使って「up」として構成された構成済みインタフェースに割り当てられるアドレス。

インタフェースフラグ

`ifconfig` コマンドがサポートするインタフェースフラグは、次のとおりです。この文脈では、「アドレス」という用語は `eri0:0` などの論理インタフェースを表します。一方、「インタフェース」は `eri0` などの物理インタフェースを表します。

`ADDRCONF` このアドレスは、ステートレス `addrconf` からのものです。ステートレス機構を使えば、ホストは、ルーターが通知する情報とローカルで利用可能な情報とを組み合わせて固有のアドレスを生成できます。ルーターは、そのリンクに関連付けられたサブネットを識別するプレフィックスを通知します。一方、ホストは、サブネット内のインタフェースを一意に識別する「インタフェース識別子」を生成します。ルーターからの情報が存在しない場合、ホストはリンクローカルアドレスを生成できます。このフラグは IPv6 に固有です。

`ANYCAST` `anycast` アドレスを示します。`anycast` アドレスは、あるタイプのサービスを提供する特定のシステムグループの最近傍メンバーを識別します。`anycast` アドレスは、特定のシステムグループに割り当てられます。パケットは、`anycast` アドレスで識別される最近傍グループメンバーに配信されます。グループのすべてのメンバーに配信されるのではありません。このフラグは IPv6 に固有です。

`BROADCAST` この `broadcast` アドレスは有効です。このフラグと `POINTTOPOINT` は互いに排他です。

`CoS` このインタフェースは、何らかの形式の CoS (Class of Service) マーキングをサポートしています。一例として、VLAN インタフェース上でサポートされる 802.1D ユーザー優先順位マーキングなどが挙げられます。

`DEPRECATED` このアドレスは非推奨です。このアドレスが送信パケットの発信元アドレスとして使用されるのは、このインタフェース上で利用可能なアドレスがほかに存在しない場合と、アプリケーションがこのアドレスに明示的にバインドされている場合だけです。IPv6 非推奨アドレスは最終的に、未使用時に削除されます。これに対し、IPv4 非推奨アドレスはしばしば、IP ネットワークマルチパス化 IPv4 テストアドレスとともに使用されます。そうしたテストアドレスは、`NOFAILOVER` フラグの設定によって識別できます。さらに、`DEPRECATED` フラグは、IPv6 における再採番用標準機構の一部となっています。

DHCP	このアドレスを管理するために DHCP が使用されています。
FAILED	このインタフェースで障害が発生しました。このインタフェース上で新しいアドレスを作成することはできません。このインタフェースが IP ネットワークマルチパスグループに属している場合、可能であれば、そのグループ内の別のインタフェースへのフェイルオーバーが発生します
FIXEDMTU	-mtu オプションで MTU が設定されました。このフラグは読み取り専用です。このフラグが設定されたインタフェースは固定の MTU 値を持ちますが、その値は、ドライバがリンク MTU 変更を IP に通知する際に発生する可能性のある動的 MTU 変更の影響を受けません。
INACTIVE	待機インタフェース上でのみ設定されます。このフラグは、インタフェースへのフェイルオーバーが発生していないことを示します。このインタフェース上で新しいアドレスを作成することはできません。インタフェースへのフェイルオーバーが発生すると、このフラグはクリアされます。
LOOPBACK	これがループバックインタフェースであることを示します。
MIP	モバイル IP がこのインタフェースを制御することを示します。
MULTI_BCAST	このインタフェースではブロードキャストアドレスがマルチキャスト用として使用されることを示します。
MULTICAST	このインタフェースはマルチキャストをサポートします。IP は、ハードウェアブロードキャストをサポートするすべてのインタフェースとポイントツーポイントリンクインタフェースは、マルチキャストをサポートするものと仮定します。
NOARP	ブロードキャストアドレスを持たないデバイスのすべてのインタフェースに対応した、このインタフェース用のアドレス解決プロトコル (ARP) が存在しません。このフラグは IPv4 に固有です。
NOFAILOVER	このアドレスは、インタフェースでの障害発生時にフェイルオーバーしません。IP ネットワークマルチパス化テストアドレスは、nofailover としてマークする必要があります。
NOLOCAL	このインタフェースはアドレスを持ちません。持つのはオンリンクサブネットだけです。
NONUD	このインタフェースでは、NUD が無効になっています。各ノードは NUD (近傍到達不可能性検出) を使って、自身が能動的にパケットを送信する近傍ノードの到達状態を追跡し、到達不可能な近傍ノードが検出された際に回復処理を実行します。このフラグは IPv6 に固有です。

NORTEXCH　このインタフェースは経路指定情報を交換しません。RIP-2 の場合、経路指定パケットがこのインタフェース経由で送信されることはありません。さらに、このインタフェースを経由したと思われるメッセージには、応答が返されません。このインタフェースのサブネットまたはアドレスが、ほかのインタフェース経由でのほかのルーター宛の通知に含められることはありません。

NOXMIT　このアドレスがパケットを送信しないことを示します。また、RIP-2 はこのアドレスを通知しません。

OFFLINE　インタフェースがオフラインになっていることを示します。このインタフェース上で新しいアドレスを作成することはできません。IP ネットワークマルチパスグループ内のインタフェースは、動的再構成による削除や置換が実行される前にオフラインになります。

POINTOPOINT　このアドレスがポイントツーポイントリンクであることを示します。このフラグと BROADCAST は互いに排他です。

PREFERRED　このアドレスは優先 IPv6 発信元アドレスです。このアドレスは、すべての IPv6 着信先との IPv6 通信の発信元アドレスとして使用されます。ただし、システム上の別のアドレスがより適切なスコープを持つ場合は除外します。DEPRECATED フラグは PREFERRED フラグよりも優先されます。

PRIVATE　このアドレスが通知されないことを示します。RIP-2 の場合、このインタフェースは通知を送信する際に使用されます。ただし、サブネットやこのアドレスが、ほかのルーター宛の通知に含められることはありません。

ROUTER　このインタフェースへの、およびこのインタフェースからの、IP パケットの転送を行えることを示します。

RUNNING　インタフェースが必要とするリソースが確保されていることを示します。一部のインタフェースでは、これは、リンクが稼働していることも示します。

STANDBY　これが障害発生時に使用するべき待機インタフェースであることを示します。待機インタフェースとして指定するのは、IP ネットワークマルチパスグループに属するインタフェースに限定するべきです。このインタフェースが IP ネットワークマルチパスグループに属する場合、このインタフェースがパケット送信用として選択されるのは、そのグループ内のほかのインタフェースからのフェイルオーバーが発生した場合だけです。

TEMPORARY　これが RFC 3041 に規定された一時 IPv6 アドレスであることを示します。

UNNUMBERED　このフラグが設定されるのは、このリンクのローカル IP アドレスが、システム内のほかのリンクのローカルアドレスに一致した場合です。

UP　このインタフェースが「up」状態にあること、つまり、このインタフェースのすべての経路指定エントリとその他の情報の設定が完了していることを示します。

XRESOLV　インタフェースが IPv6 外部リゾルバを使用することを示します。

論理インタフェース

Solaris の TCP/IP では、1 つの物理ネットワークインタフェースに複数の論理インタフェースを関連付けることができます。これにより、ネットワークインタフェースを 1 つしか持たないようなマシンにも、複数の IP アドレスを割り当てることが可能になります。物理ネットワークインタフェースが *driver-name physical-unit-number* という形式の名前を持つのに対し、論理インタフェースは *driver-name physical-unit-number*:*logical-unit-number* という形式の名前を持ちます。特定の物理インタフェースを構成してシステムに組み込むには、`plumb` コマンドを使用します。たとえば、次のように指定します。

```
example% ifconfig eri0 plumb
```

物理インタフェースの「plumb」が完了すると、その物理インタフェースに関連付けられた論理インタフェースを構成できるようになります。それには、`-plumb` または `-addif` オプションを指定して再度 `ifconfig` コマンドを実行します。

```
example% ifconfig eri0:1 plumb
```

これは、物理インタフェース `eri0` に関連付けられた特定の論理インタフェースを割り当てます。次のコマンドを見てください。

```
example% ifconfig eri0 addif 192.168.200.1/24 up
```

これは、`eri0` 物理インタフェース上で利用可能な次の論理ユニット番号を割り当てるとともに、*address* と *prefix_length* も割り当てます。

論理インタフェースは、関連付けられた物理インタフェースとは異なるパラメータ (*address*、*prefix_length* など) を使って構成することができます。また、同じ物理インタフェースに関連付けられた複数の論理インタフェースにそれぞれ異なるパラメータを与えることもできます。各論理インタフェースは、既存の「up」状態の物理インタフェースに関連付ける必要があります。したがって、たとえば、論理インタフェース `eri0:1` を構成できるのは、物理インタフェース `eri0` を plumb し終わったあとです。

論理インタフェースを削除するには、`-unplumb` または `-removeif` オプションを使用します。次に例を示します。

```
example% ifconfig eri0:1 down unplumb
```

これは、論理インタフェース `eri0:1` を削除します。

マルチパスグループ

同じ IP ブロードキャストドメインを共有する物理インタフェースは、group キーワードを使って特定のマルチパスグループ内に集めることができます。同じマルチパスグループに割り当てられたインタフェースは互いに同等とみなされ、それらのインタフェース全体に対して、送信トラフィックが IP 着信先単位で分散されます。さらに、マルチパスグループ内の個々のインタフェースで障害が発生していないかの監視が行われます。そして、障害が発生したインタフェースに関連付けられたアドレスは、正常に機能しているグループ内のほかのインタフェースに自動的に転送されます。

IP マルチパス化の詳細については、in.mpathd(1M) と『Solaris のシステム管理 (IP サービス)』を参照してください。IP 着信先単位の情報については、netstat(1M) を参照してください。

Ipv6 インタフェースの構成

ifconfig による IPv6 物理インタフェースの plumb と「up」構成が完了すると、そのインタフェースには、IPv6 リンクローカルアドレスが自動的に割り当てられます。このアドレスの最後の 64 ビットは、そのインタフェースの MAC アドレスから計算されます。

```
example% ifconfig eri0 inet6 plumb up
```

次の例は、リンクローカルアドレスのプレフィックスが fe80::/10 であることを示しています。

```
example% ifconfig eri0 inet6
ce0: flags=2000841<UP,RUNNING,MULTICAST,IPv6>
          mtu 1500 index 2
        inet6 fe80::a00:20ff:fe8e:f3ad/10
```

リンクローカルアドレスは、ローカルサブネット上での通信にのみ使用され、ほかのサブネットからは見えません。

プレフィックスを通知しているリンク上に通知 IPv6 ルーターが存在する場合、新しく plumb された IPv6 インタフェースは、1 つ以上の論理インタフェースをそのプレフィックス通知に基づいて自動構成します。たとえば、プレフィックス通知が 2001:0db8:3c4d:0:55::/64 であった場合、自動構成されたインタフェースは次のようになります。

```
eri0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6>
          mtu 1500 index 2
        inet6 2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64
```

プレフィックス通知がリンク上に存在しない場合でも、グローバルアドレスを手動で割り当てることができます。次に例を示します。

```
example% ifconfig eri0 inet6 addif \
2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64 up
```

インタフェース eri0 のブート時のデフォルトを構成するには、/etc/hostname6.eri0 ファイル内に次のエントリを格納します。

```
addif  2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64 up
```

IPv6/IPv4 トンネルの構成

IPv4 トンネル経由の IPv6 インタフェースは、IPv4 パケット内にカプセル化された IPv6 パケットを送受信できます。両端にトンネルを作成します。その際、2 つのトンネルが互いをポイントするようにします。IPv4 トンネル経由の IPv6 では、トンネル発信元とトンネル着信先の IPv4 アドレスと IPv6 アドレスが必要となります。Solaris 8 は、自動トンネルと構成済みトンネルの両方をサポートします。自動トンネルでは、IPv4 に準拠した IPv6 アドレスが使用されます。次に、自動トンネル構成の場合を示します。

```
example% ifconfig ip.atun0 inet6 plumb
example% ifconfig ip.atun0 inet6 tsrc IPv4-address \
   ::IPv4 address/96 up
```

ここで、IPv4–address はトンネルのトラフィックが通過するインタフェースの IPv4 アドレス、IPv4-address、::<IPv4–address> はそれに対応する IPv4 互換 IPv6アドレスです。

次に、構成済みトンネルの例を示します。

```
example% ifconfig ip.tun0 inet6 plumb tsrc my-ipv4-address \
   tdst peer-ipv4-address up
```

これにより、my-ipv4-address と peer-ipv4-address との間に、対応するリンクローカルアドレスを持つ構成済みトンネルが作成されます。グローバルアドレスまたはサイトローカルアドレスを持つトンネルの場合、次の形式を使って論理トンネルインタフェースを構成する必要があります。

```
example% ifconfig ip.tun0 inet6 addif my-v6-address peer-v6-address up
```

次に例を示します。

```
example% ifconfig ip.tun0 inet6 plumb tsrc 109.146.85.57 \
   tdst 109.146.85.212 up
example% ifconfig ip.tun0 inet6 addif 2::45 2::46 up
```

「up」状態で構成済みの IPv6 インタフェースをすべて表示するには、次のようにします。

```
example% ifconfig -au6
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6>
          mtu 1480 index 3
        inet tunnel src 109.146.85.57   tunnel dst 109.146.85.212
        tunnel hop limit 60
        inet6 fe80::6d92:5539/10 --> fe80::6d92:55d4
ip.tun0:1: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6>
```

```
        mtu 1480 index 3
      inet6 2::45/128 --> 2::46
```

IPv4/IPv6 トンネルの構成

IPv6 トンネル経由の IPv4 インタフェースは、IPv6 パケット内にカプセル化された IPv4 パケットを送受信できます。両端にトンネルを作成します。その際、2 つのトンネルが互いをポイントするようにします。IPv6 トンネル経由の IPv4 では、トンネル発信元とトンネル着信先の IPv6 アドレスと IPv4 アドレスが必要となります。次に、自動トンネル構成の場合を示します。

```
example% ifconfig ip6.tun0 inet plumb tsrc my-ipv6-address \
   tdst peer-ipv6-address my-ipv4-address \
   peer-ipv4-address up
```

これにより、*my-ipv6-address* と *peer-ipv6-address* との間に、*my-ipv4-address* と *peer-ipv4-address* をポイントツーポイントインタフェースの端点として持つ構成済みトンネルが作成されます。次に例を示します。

```
example% ifconfig ip6.tun0 inet plumb tsrc fe80::1 tdst fe80::2 \
10.0.0.208 10.0.0.210 up
```

「up」状態で構成済みの IPv4 インタフェースをすべて表示するには、次のようにします。

```
example% ifconfig -au4
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
     inet 127.0.0.1 netmask ff000000
eri0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 \
index 2
     inet 172.17.128.208 netmask ffffff00 broadcast 172.17.128.255
ip6.tun0: flags=10008d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST,IPv4> mtu \
     1460 index 3
     inet6 tunnel src fe80::1 tunnel dst fe80::2
     tunnel hop limit 60 tunnel encapsulation limit 4
     inet 10.0.0.208 --> 10.0.0.210 netmask ff000000
```

使用例

例 1　`ifconfig` コマンドの使用

ワークステーションが Ethernet に接続されていない場合、そのネットワークインタフェース (たとえば `eri0` など) を次のようにして「down」としてマークしてください。

```
example% ifconfig eri0 down
```

例 2　アドレス指定情報の出力

各インタフェースのアドレス指定情報を出力するには、次のコマンドを使用します。

例 2 アドレス指定情報の出力　　(続き)

```
example% ifconfig -a
```

例 3 ブロードキャストアドレスのリセット

ネットマスクが正しく設定された状態で、各インタフェースのブロードキャストアドレスをリセットするには、次のコマンドを使用します。

```
example% ifconfig -a broadcast +
```

例 4 Ethernet アドレスの変更

インタフェース ce0 の Ethernet アドレスを変更するには、次のコマンドを使用します。

```
example% ifconfig ce0 ether aa:1:2:3:4:5
```

例 5 IP 内 IP トンネルの構成

特定の IP 内 IP トンネルを構成するには、まず、次のコマンドを使ってそのトンネルを plumb します。

```
example% ifconfig ip.tun0 plumb
```

続いて、それをポイントツーポイントインタフェースとして構成します。それには、次のようにトンネル発信元とトンネル着信先を指定します。

```
example% ifconfig ip.tun0 myaddr mydestaddr tsrc another_myaddr \
          tdst a_dest_addr up
```

トンネルセキュリティーのプロパティーは、次のように 1 回の ifconfig 呼び出しで構成する必要があります。

```
example% ifconfig ip.tun0 encr_auth_algs md5 encr_algs 3des
```

例 6 アルゴリズムの優先順を指定せずにサービスを要求する

アルゴリズムの優先順を指定しないでサービスを要求するには、次のように any を指定します。

```
example% ifconfig ip.tun0 encr_auth_algs any encr_algs any
```

例 7　すべてのセキュリティーの無効化

すべてのセキュリティーを無効にするには、次のように、すべてのセキュリティーサービスのアルゴリズム値として none を指定します。

```
example% ifconfig ip.tun0 auth_algs none
```

または

```
example% ifconfig ip.tun0 encr_algs none
```

例 8　6to4 トンネルの構成

6to4 トンネルを構成するには、次のコマンドを使用します。

```
example% ifconfig ip.6to4tun0 inet6 plumb
example% ifconfig ip.6to4tun0 inet6 tsrc IPv4-address 6to4-address/64 up
```

IPv4-address は、カプセル化を行うインタフェースのアドレスを表します。*6to4-address* は、ローカル IPv6 アドレスの次の形式のアドレスを表します。2002:*IPv4-address*:*SUBNET-ID*:*HOSTID*。

システム管理者が使用しているアドレス指定計画で、SUBNET-ID や HOSTID の値が別の目的で予約されている可能性を考慮して、潜在的な衝突を回避するために、この長形式を使用することをお勧めします。

インタフェースの plumb 完了後に、次のようにして 6to4 トンネルを構成できます。

```
example% ifconfig ip.6to4tun0 inet6 tsrc IPv4-address up
```

この短形式はアドレスを設定します。次の規則が使用されます。

```
2002:IPv4-address::1
```

SUBNET-ID が 0、HOSTID が 1 になっています。

例 9　インタフェース上の IP 転送の構成

単一インタフェース上の IP 転送を有効にするには、次のコマンドを使用します。

```
example% ifconfig eri0 router
```

単一インタフェース上の IP 転送を無効にするには、次のコマンドを使用します。

```
example% ifconfig eri0 -router
```

例 10　仮想インタフェースを使用した発信元アドレス選択の構成

次のコマンドは発信元アドレス選択を構成することで、発信元アドレスが関連付けられずにローカルで生成されて qfe2 から出力されるすべてのパケットに対し、vni0 上にホストされた発信元アドレスが優先的に使用されるようにします。

```
example% ifconfig qfe2 usesrc vni0
```

qfe2 および vni0 インタフェースに対する ifconfig -a の出力は、次のようになります。

```
qfe2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4> mtu
  1500 index 4
  usesrc vni0
  inet 1.2.3.4 netmask ffffff00 broadcast 1.2.3.255
  ether 0:3:ba:17:4b:e1
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
  mtu 0 index 5
  srcof qfe2
  inet 3.4.5.6 netmask ffffffff
```

上記の ifconfig 出力に含まれる usesrc と srcof キーワードに注目してください。これらのキーワードは、物理インタフェース単位のパラメータであるにもかかわらず、物理インタフェースの論理インスタンス上にも現れます。構成元のインタフェースに対する ifconfig には、srcof キーワードは現れません。この情報は、usesrc が設定された一連のインタフェースから自動的に決定されます。

none キーワードを使用した次のコマンドは、先の ifconfig usersrc コマンドによる効果を取り消します。

```
example% ifconfig qfe2 usesrc none
```

このコマンドの実行後、ifconfig -a の出力は次のようになります。

```
qfe2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4> mtu
  1500 index 4
  inet 1.2.3.4 netmask ffffff00 broadcast 1.2.3.255
  ether 0:3:ba:17:4b:e1
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
  mtu 0 index 5
  inet 3.4.5.6 netmask ffffffff
```

上記の出力には usesrc と srcof キーワードは含まれていません。

例 11　IPv6 アドレスの発信元アドレス選択の構成

次のコマンドは、vni0 上にホストされた発信元アドレスを選択することで、IPv6 アドレスの発信元アドレス選択を構成します。

例 11　IPv6 アドレスの発信元アドレス選択の構成　　(続き)

```
example% ifconfig qfe1 inet6 usesrc vni0
```

このコマンドの実行後、ifconfig -a の出力は次のようになります。

```
qfe1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
  usesrc vni0
  inet6 fe80::203:baff:fe17:4be0/10
  ether 0:3:ba:17:4b:e0
vni0: flags=2002210041<UP,RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 fe80::203:baff:fe17:4444/128
vni0:1: flags=2002210040<RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 fec0::203:baff:fe17:4444/128
vni0:2: flags=2002210040<RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 2000::203:baff:fe17:4444/128
```

qfe1 から出力されるパケットの着信先のスコープに応じて、適切なスコープを持つ発信元アドレスが vni0 とその別名から選択されます。

例 12　ゾーンでの発信元アドレス選択の使用

次の例は、Solaris の zones(5) 機能における usesrc 機能の使用方法を示したものです。大域ゾーン内で次の各コマンドを呼び出したとします。

```
example% ifconfig hme0 usesrc vni0
example% ifconfig eri0 usesrc vni0
example% ifconfig qfe0 usesrc vni0
```

前述のコマンドの実行後、その仮想インタフェースに対する ifconfig -a の出力は次のようになります。

```
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
   mtu 0 index 23
   srcof hme0 eri0 qfe0
   inet 10.0.0.1 netmask ffffffff
vni0:1:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test1
   srcof hme0 eri0 qfe0
   inet 10.0.0.2 netmask ffffffff
```

例 12 ゾーンでの発信元アドレス選択の使用 (続き)

```
vni0:2:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test2
   srcof hme0 eri0 qfe0
   inet 10.0.0.3 netmask ffffffff
vni0:3:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test3
   srcof hme0 eri0 qfe0
   inet 10.0.0.4 netmask ffffffff
```

仮想インタフェースの別名が、ゾーン (test1、test2、および test3) ごとに 1 つずつ存在しています。同じゾーン内の仮想インタフェース別名に含まれる発信元アドレスが選択されます。これらの仮想インタフェース別名は、次のように zonecfg(1M) を使って作成されたものです。

```
example% zonecfg -z test1
zonecfg:test1> add net
zonecfg:test1:net> set physical=vni0
zonecfg:test1:net> set address=10.0.0.2
```

test2 ゾーンと test3 ゾーンのインタフェースとアドレスも同じ方法で作成されます。

ファイル

/etc/netmasks	ネットマスクのデータ。
/etc/default/inet_type	デフォルトのインターネットプロトコルタイプ。

属性

以下の属性については、attributes(5) を参照してください。

/usr/sbin

属性タイプ	属性値
使用条件	SUNWcsu
オプション modlist、modinsert、および modremove に対するインタフェースの安定性	開発中

/sbin

属性タイプ	属性値
使用条件	SUNWcsr
オプション modlist、modinsert、および modremove に対するインタフェースの安定性	開発中

関連項目

dhcpinfo(1), dhcpagent(1M), in.mpathd(1M), in.routed(1M), ndd(1M), netstat(1M), zoneadm(1M), ethers(3SOCKET), gethostbyname(3NSL), getnetbyname(3SOCKET), hosts(4), inet_type(4), netmasks(4), networks(4), nsswitch.conf(4), attributes(5), privileges(5), zones(5), arp(7P), ipsecah(7P), ipsecesp(7P), tun(7M)

『Solaris のシステム管理 (IP サービス)』

診断

ifconfig は、次のことを示すメッセージを送信します。

- 指定されたインタフェースが存在しないかどうか
- 要求されたアドレスが未知かどうか
- 非特権ユーザーが特定のインタフェースの構成を変更しようとしているかどうか

注意事項

ホスト名を選択する際に、名前 broadcast、down、private、trailers、up やその他のオプション名を選択しないでください。これらの名前のいずれかをホスト名として選択した場合、診断が非常に難しい特異な問題が発生する可能性があります。

名前

inetadm – inetd が制御するサービスを監視または構成する

形式

```
inetadm
inetadm  -?
inetadm  -p
inetadm  -l {FMRI | pattern}
inetadm  -e {FMRI | pattern}
inetadm  -d {FMRI | pattern}
inetadm  -m {FMRI | pattern}... {name=value}...
inetadm  -M {name=value}...
```

機能説明

inetadm ユーティリティーは、inetd が管理する SMF サービスに次の機能を提供します。

- インストールされているすべてのサービスの一覧を表示します。
- サービスのプロパティーと値の一覧を表示します。
- サービスを有効に、または無効にすることができます。
- サービスのプロパティー値を変更したり、inetd によって提供されるデフォルトのサービスプロパティー値を変更したりできます。

SMF サービスについては、smf(5) を参照してください。

引数を指定しない場合、inetadm は inetd(1M) によって制御されるすべてのサービス (現在の実行状態や有効であるかどうかなどの属性を含む) を表示します。

オプション

オプションが 1 つ以上の FMRI オペランド (FMRI の説明については、smf(5) を参照) をとるときに、オペランドが (サービスインスタンスではなく) サービスを指定し、そのサービスが単一のインスタンスしか持たない場合、inetadm はそのインスタンス上で動作します。

サービス名が指定されていてサービス名に複数のインスタンスが含まれている場合、またはパターンが指定されていてパターンが複数のインスタンスに一致する場合は、警告メッセージが表示され、そのオペランドは無視されます。

name=value パラメータをとるオプションに使用できる名前と許可されている値の説明は、inetd(1M) のマニュアルページに記載されています。

サポートしているオプションは、以下のとおりです。

`-?` 使用方法に関するメッセージを表示します。

`-p` inetd によって提供されるすべてのデフォルトの inet サービスプロパティー値を、

name=value の組の形式で一覧表示します。値がブール型の場合は、TRUE または FALSE で表示されます。

`-l` {*FMRI* | *pattern*}... 指定されたサービスインスタンスのすべてのプロパティーを、 *name=value* の組で一覧表示します。また、`inetd` によって提供されたデフォルト値からプロパティー値が継承されている場合、その *name=value* の組には「default」というトークンも表示されます。プロパティーの継承は、プロパティーにサービスインスタンスのデフォルトが指定されていない場合に発生します。

`-e` {*FMRI* | *pattern*}... 指定されたサービスインスタンスを有効にします。

`-d` {*FMRI* | *pattern*}... 指定されたサービスインスタンスを無効にします。

`-m` {*FMRI* | *pattern*}...{*name=value*}... 識別されたサービスインスタンスの指定されたプロパティーの値を変更します。プロパティーは、*name=value* の組を空白で区切って指定します。インスタンス固有の値を削除し、プロパティーのデフォルト値を使うように指定するには、たとえば、`name=` のように、値なしでプロパティーを指定します。

`-M` {*name=value*}... 指定された `inetd` のデフォルトのプロパティー値を変更します。プロパティーは、*name=value* の組を空白で区切って指定します。

使用例

例 1 サービスのプロパティーの表示

次のコマンドは `spray` サービスのプロパティーを表示します。

```
# inetadm -l network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=1
         wait=TRUE
         exec="/usr/lib/netsvc/spray/rpc.sprayd"
         user="root"
```

例 1 サービスのプロパティーの表示 (続き)

```
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=FALSE
default  tcp_wrappers=FALSE
```

例 2 デフォルトプロパティーの表示

次のコマンドは、デフォルトプロパティーを表示します。

```
# inetadm -p
NAME=VALUE
bind_addr=""
bind_fail_max=-1
bind_fail_interval=-1
max_con_rate=-1
max_copies=-1
con_rate_offline=-1
failrate_cnt=40
failrate_interval=60
inherit_env=TRUE
tcp_trace=FALSE
tcp_wrappers=FALSE
```

例 3 サービスのプロパティー値の変更

次のコマンドは、spray サービスの rpc_high_version を 3 に、 tcp_trace を TRUE に変更します。

```
# inetadm -m network/rpc/spray:default \
         rpc_high_version=3 tcp_trace=TRUE
# inetadm -l network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=3
```

例 3 サービスのプロパティー値の変更 (続き)

```
        wait=TRUE
        exec="/usr/lib/netsvc/spray/rpc.sprayd"
        user="root"
default bind_addr=""
default bind_fail_max=-1
default bind_fail_interval=-1
default max_con_rate=-1
default max_copies=-1
default con_rate_offline=-1
default failrate_cnt=40
default failrate_interval=60
default inherit_env=TRUE
        tcp_trace=TRUE
default tcp_wrappers=FALSE
```

終了ステータス

次の終了ステータスが返されます。

0 操作は正常に完了しました。

1 致命的なエラーが発生しました。詳細情報は同時に表示されるエラーメッセージに記述されます。

2 無効な引数 (あいまいなサービス FMRI またはパターンなど) が指定されました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	開発中

関連項目

inetd(1M), svccfg(1M), attributes(5), smf(5)

名前

inetconv – inetd.conf エントリを smf サービスマニフェストに変換し、それらを smf リポジトリ内にインポートする

形式

```
inetconv  -?

inetconv  [-f] [- n] [-i srcfile] [-o destdir]

inetconv  -e [-n] [-i srcfile]
```

機能説明

inetconv ユーティリティーは、inetd.conf(4) のレコードを含むファイルを smf(5) サービスマニフェストに変換したあと、それらのマニフェストを smf リポジトリ内にインポートします。inetd.conf ファイルの変換処理がいったん完了すると、inetadm(1m) ユーティリティーを使用しないと各 inet サービスの特性を変更できなくなります。

入力ファイル内の 1 つのサービス行と、それに対して生成されるマニフェストとの間には、1 対 1 の対応関係が成り立ちます。マニフェストはデフォルトで、次のテンプレートに従って命名されます。

<*svcname*>-<*proto*>.xml

<*svcname*> トークンはサービスの名前で、<*proto*> トークンはサービスのプロトコルで、それぞれ置き換えられます。ソース行のサービス名やプロトコルに含まれるスラッシュ (/) 文字はすべて、下線 (_) で置き換えられます。

各サービス行は、変換先となるサービスのプロパティーとして記録されます。

変換処理中に、不正なサービス行や inetd に対する内部的なサービス行が検出された場合には、マニフェストは生成されず、そのサービス行はスキップされます。

入力ファイルは変換処理の影響を受けず、元の状態に保たれます。

オプション

サポートしているオプションは、次のとおりです。

-? 使用方法に関するメッセージを表示します。

-e 入力ファイル内に記述されている smf サービスを有効にします。

-f このオプションが指定された場合、生成予定のサービスマニフェストと同じ名前のマニフェストが出力ディレクトリ内で見つかった場合に、inetconv はそのマニフェストを上書きします。それ以外の場合、エラーメッセージが生成され、そのサービスの変換は実行されません。

-i *srcfile* 別の入力ファイル *srcfile* を指定できるようにします。このオプションが指定されなかった場合、inetd.conf(4) ファイルが入力として使用されます。

-n 変換処理中に生成されたマニフェストの自動インポートを無効にします。その後、生成済みマニフェストを smf(5) リポジトリ内にインポートする必要が生じた場合、svccfg(1M) ユーティリティーを使えばそれを行うことができます。

-e オプションが指定された場合、-n オプションは単に、有効化される smf サービスを表示します。

-o 生成されたマニフェストに対する別の出力ディレクトリ *destdir* を指定できるようにします。このオプションが指定されなかった場合、RPC サービスのマニフェストは /var/svc/manifest/network/rpc 内に、それ以外のサービスのマニフェストは /var/svc/manifest/network 内に、それぞれ格納されます。

使用例

例 1 inetd.conf からの smf マニフェストの生成

次のコマンドは、inetd.conf(4) から smf(5) マニフェストを生成し、それらを /var/tmp 内に格納します。その際、同名の既存マニフェストはすべて上書きします。続いて、それらのマニフェストを smf リポジトリ内にインポートします。

```
# inetconv -f -o /var/tmp
100232/10 -> /var/tmp/100232_10-rpc_udp.xml
Importing 100232_10-rpc_udp.xml ...Done
telnet -> /var/tmp/telnet-tcp6.xml
Importing telnet-tcp6.xml ...Done
```

例 2 別の入力ファイルからのマニフェストの生成

次のコマンドは、異なる入力ファイルを指定しています。また、結果のマニフェストを smf リポジトリ内にロードしません。

```
# inetconv -n -i /export/test/inet.svcs -o /var/tmp
100232/10 -> /var/tmp/100232_10-rpc_udp.xml
telnet -> /var/tmp/telnet-tcp6.xml
```

終了ステータス

次の終了ステータスが返されます。

0 処理が正常終了しました (エラーなし)。

1 無効なオプションが指定されました。

2 不正なサービス行が 1 つ以上存在します。それらに対するマニフェストは生成されませんでした。

3 1 つ以上の生成済みマニフェストのインポート中にエラーが発生しました。

4 システムエラーが発生しました。

ファイル

/var/svc/manifest/network/{rpc}/<*svcname*>-<*proto*>.xml
デフォルトの出力マニフェストファイル名

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	開発中

関連項目 inetadm(1m), inetd(1M), svccfg(1M), `inetd.conf`(4), `attributes`(5), `smf`(5)

名前

inetd – Solaris Management Facility delegated restarter for inet services

形式

```
inetd  [configuration-file] start |  stop |  refresh

 svc:/network/inetd:default
```

機能説明

`inetd` is the delegated restarter for internet services for the Service Management Facility (SMF). Its basic responsibilities are to manage service states in response to administrative requests, system failures, and service failures; and, when appropriate, to listen for network requests for services.

Services are no longer managed by editing the inetd configuration file, `inetd.conf`(4). Instead, you use inetconv(1m) to convert the configuration file content into SMF format services, then manage these services using inetadm(1m) and svcadm(1M). Once a service has been converted by `inetconv`, any changes to the legacy data in the `inetd` config file will not become effective. However, `inetd` does alert the administrator when it notices change in the configuration file. See the start description under the “inetd Methods” section for further information.

Also note that the current `inetd` cannot be run from outside the SMF. This means it cannot be run from the command line, as was supported by the previous `inetd`. If you attempt to do this, a message is sent to stderr displaying mappings between the options supported by the previous `inetd` to the SMF version of `inetd`.

`inetd` listens for connections on behalf of all services that are in either the `online` or `degraded` state. A service enters one of these states when the service is enabled by the user and `inetd` manages to listen on its behalf. A listen attempt can fail if another server (whether standalone or a third-party internet service) is already listening on the same port. When this occurs, `inetd` logs this condition and continues trying to bind to the port at configured intervals a configured number of times. See the property `bind_fail_max` under “Service Properties,” below, for more details.

The configuration of all `inetd`'s managed SMF services is read when it is started. It is reread when `inetd` is refreshed, which occurs in response to an SMF request, or when it receives a SIGHUP signal. See the `refresh` description under “inetd Methods” for the behavior on configuration refresh.

You can use the inetadm(1m) or svccfg(1M) utilities to make configuration changes to Internet services within the SMF repository. `inetadm` has the advantage over `svccfg` in that it provides an Internet/RPC service context.

Service States

As part of its service management duties, `inetd` implements a state machine for each of its managed services. The states in this machine are made up of the `smf`(5) set of states. The semantics of these states are as follows:

`uninitialized`
  `inetd` has yet to process this service.

`online`
The service is handling new network requests and might have existing connections active.

`degraded`
The service has entered this state because it was able to listen and process requests for some, but not all, of the protocols specified for the service, having exhausted its listen retries. Existing network connections might be active.

`offline`
Connections might be active, but no new requests are being handled. This is a transient state. A service might be `offline` for any of the following reasons:

- The service's dependencies are unmet. When its dependencies become met the service's state will be re-evaluated.
- The service has exceeded its configured connection rate limit, `max_con_rate`. The service's state is re-evaluated when its connection offline timer, `con_rate_offline`, expires.
- The service has reached its allowed number of active connections, `max_copies`. The service's state is re-evaluated when the number of active connections drops below `max_copies`.
- `inetd` failed to listen on behalf of the service on all its protocols. As mentioned above, `inetd` retries up to a configured maximum number of times, at configured intervals.The service's state is re-evaluated when either a listen attempt is successful or the retry limit is reached.

`disabled`
The service has been turned off by an administrator, is not accepting new connections, and has none active. Administrator intervention is required to exit this state.

`maintenance`
A service is in this state because it is either malfunctioning and needs adminstrator attention or because an administrator has requested it.

Events constituting malfunctioning include: `inetd`'s inability to listen on behalf on any of the service's protocols before exceeding the service's bind retry limit, non-start methods returning with non-success return values, and the service exceeding its failure rate.

You request the maintenance state to perform maintenance on the service, such as applying a patch. No new requests are handled in this state, but existing connections might be active. Administrator intervention is required to exit this state.

Use inetadm(1m) to obtain the current state of a managed service.

Service Methods

As part of certain state transitions `inetd` will execute, if supplied, one of a set of methods provided by the service. The set of supported methods are:

`inetd_start`
Executed to handle a request for an `online` or `degraded` service. Since there is no separate state to distinguish a service with active connections, this method is not executed as part of a state transition.

`inetd_offline`
Executed when a service is taken from the `online` or `degraded` state to the `offline` state. For a `wait`-type service that at the time of execution is performing its own listening, this method should result in it ceasing listening. This method will be executed before the `disable` method in the case an `online`/`degraded` service is disabled. This method is required to be implemented for a `wait`-type service.

`inetd_online`
Executed when a service transitions from the `offline` state to the `online` state. This method allows a service author to carry out some preparation prior to a service starting to handle requests.

`inetd_disable`
Executed when a service transitions from the `offline` state to the `disabled` state. It should result in any active connections for a service being terminated.

`inetd_refresh`
Executed when both of the following conditions are met:

- `inetd` is refreshed, by means of the framework or a SIGHUP, or a request comes in to refresh the service, and
- the service is currently in the `online` state and there are no configuration changes that would result in the service needing to be taken `offline` and brought back again.

The only compulsory method is the `inetd_start` method. In the absence of any of the others, `inetd` runs no method but behaves as if one was run successfully.

Service Properties

Configuration for SMF–managed services is stored in the SMF repository. The configuration is made up of the basic configuration of a service, the configuration for each of the service's methods, and the default configuration applicable to all `inetd`-managed services.

For details on viewing and modifying the configuration of a service and the defaults, refer to inetadm(1m).

The basic configuration of a service is stored in a property group named `inetd` in the service. The properties comprising the basic configuration are as follows:

`bind_addr`
The address of the network interface to which the service should be bound. An empty string value causes the service to accept connections on any network interface.

`bind_fail_interval`
The time interval in seconds between a failed bind attempt and a retry. The values `0` and `-1` specify that no retries are attempted and the first failure is handled the same as exceeding `bind_fail_max`.

`bind_fail_max`
The maximum number of times `inetd` retries binding to a service's associated port before giving up. The value `-1` specifies that no retry limit is imposed. If none of the service's protocols were bound to before any imposed limit is reached, the service goes to the `maintenance` state; otherwise, if not all of the protocols were bound to, the service goes to the `degraded` state.

`con_rate_offline`
The time in seconds a service will remain offline if it exceeds its configured maximum connection rate, `max_con_rate`. The values `0` and `-1` specify that connection rate limiting is disabled.

`endpoint_type`
The type of the socket used by the service or the value `tli` to signify a TLI-based service. Valid socket type values are: `stream`, `dgram`, `raw`, `seqpacket`.

`failrate_cnt`
The count portion of the service's failure rate limit. The failure rate limit applies to `wait`-type services and is reached when *count* instances of the service are started within a given time. Exceeding the rate results in the service being transitioned to the `maintenance` state. This is different from the behavior of the previous `inetd`, which continued to retry every 10 minutes, indefinitely. The `failrate_cnt` check accounts for badly behaving servers that fail before consuming the service request and which would otherwise be continually restarted, taxing system resources. Failure rate is equivalent to the `-r` option of the previous `inetd`. The values `0` and `-1` specify that this feature is disabled.

`failrate_interval`
The time portion in seconds of the service's failure rate. The values `0` and `-1` specify that the failure rate limit feature is disabled.

`inherit_env`
If true, pass `inetd`'s environment on to the service's start method. Regardless of this setting, `inetd` will set the variables `SMF_FMRI`, `SMF_METHOD`, and `SMF_RESTARTER` in the start method's environment, as well as any environment variables set in the method context. These variables are described in `smf_method`(5).

`isrpc`
If true, this is an RPC service.

`max_con_rate`
The maximum allowed connection rate, in connections per second, for a `nowait`-type service. The values `0` and `-1` specify that that connection rate limiting is disabled.

`max_copies`
The maximum number of copies of a `nowait` service that can run concurrently. The values `0` and `-1` specify that copies limiting is disabled.

`name`
Can be set to one of the following values:

- a service name understood by `getservbyname`(3SOCKET);
- if `isrpc` is set to `true`, a service name understood by `getrpcbyname`(3NSL);
- if `isrpc` is set to `true`, a valid RPC program number.

`proto`
In the case of socket-based services, this is a list of protocols supported by the service. Valid protocols are: `tcp`, `tcp6`, `tcp6only`, `udp`, `udp6`, and `udp6only`. In the case of TLI services, this is a list of netids recognized by `getnetconfigent`(3NSL) supported by the service, plus the values `tcp6only` and `udp6only`. RPC/TLI services also support nettypes in this list, and `inetd` first tries to interpret the list member as a nettype for these service types. The values `tcp6only` and `udp6only` are new to `inetd`; these values request that `inetd` listen only for and pass on true `IPv6` requests (not IPv4 mapped ones). See "Configuring Protocols for Sockets-Based Services," below.

`rpc_low_version`
Lowest supported RPC version. Required when `isrpc` is set to `true`.

`rpc_high_version`
Highest supported RPC version. Required when `isrpc` is set to `true`.

`tcp_trace`
If true, and this is a `nowait`-type service, `inetd` logs the client's IP address and TCP port number, along with the name of the service, for each incoming connection, using the `syslog`(3C) facility. `inetd` uses the `syslog` facility `code` daemon and `notice` priority level. See `syslog.conf`(4) for a description of `syslog` codes and severity levels. This logging is separate from the logging done by the TCP wrappers facility.

`tcp_trace` is equivalent to the previous `inetd`'s `-t` option (and the `/etc/default/inetd` property `ENABLE_CONNECTION_LOGGING`).

`tcp_wrappers`
If `true`, enable TCP wrappers access control. This applies only to services with `endpoint_type` set to `streams` and `wait` set to `false`. The `syslog` facility `code` daemon is used to log allowed connections (using the `notice` severity level) and denied traffic (using the `warning` severity level). See `syslog.conf`(4) for a description of `syslog` codes and severity levels. The stability level of the TCP wrappers facility and its configuration files is External. As the TCP wrappers facility is not controlled by Sun, intra-release incompatibilities are not uncommon. See `attributes`(5).

For more information about configuring TCP wrappers, you can refer to the `tcpd(1M)` and `hosts_access(4)` man pages, which are delivered as part of the Solaris operating system at `/usr/sfw/man`. These pages are not part of the standard Solaris man pages, available at `/usr/man`.

`tcp_wrappers` is equivalent to the previous inetd's `/etc/default/inetd` property `ENABLE_TCPWRAPPERS`.

`wait`
If `true` this is a `wait`-type service, otherwise it is a `nowait`-type service. A `wait`-type service has the following characteristics:

- Its `inetd_start` method will take over listening duties on the service's bound endpoint when it is executed.
- `inetd` will wait for it to exit after it is executed before it resumes listening duties.

Datagram servers must be configured as being of type `wait`, as they are always invoked with the original datagram endpoint that will participate in delivering the service bound to the specified service. They do not have separate "listening" and "accepting" sockets. Connection-oriented services, such as TCP stream services can be designed to be either of type `wait` or `nowait`.

A number of the basic properties are optional for a service. In their absence, their values are taken from the set of default values present in the `defaults` property group in the `inetd` service. These properties, with their seed values, are listed below. Note that these values are configurable through inetadm(1m).

```
bind_fail_interval  -1
bind_fail_max       -1
con_rate_offline    -1
failrate_count      40
failrate_time       60
inherit_env         true
max_con_rate        -1
max_copies          -1
tcp_trace           false
tcp_wrappers        false
```

Each method specified for a service will have its configuration stored in the SMF repository, within a property group of the same name as the method. The set of properties allowable for these methods includes those specified for the services managed by svc.startd(1M). (See svc.startd(1M) for further details.) Additionally, for the `inetd_start` method, you can set the `arg0` property.

The `arg0` property allows external wrapper programs to be used with `inetd` services. Specifically, it allows the first argument, `argv[0]`, of the service's start method to be something other than the path of the server program.

In the case where you want to use an external wrapper program and pass arguments to the service's daemon, the arguments should be incorporated as arguments to the wrapper program in the exec property. For example:

```
exec='/path/to/wrapper/prog service_daemon_args'
arg0='/path/to/service/daemon'
```

In addition to the special method tokens mentioned in smf_method(5), inetd also supports the :kill_process token for wait-type services. This results in behavior identical to that if the :kill token were supplied, except that the kill signal is sent only to the parent process of the wait-type service's start method, not to all members of its encompassing process contract (see process(4)).

Configuring Protocols for Sockets-Based Services

When configuring inetd for a sockets-based service, you have the choice, depending on what is supported by the service, of the alternatives described under the proto property, above. The following are guidelines for which proto values to use:

- For a service that supports only IPv4: tcp and udp
- For a service that supports only IPv6: tcp6only and udp6only
- For a service that supports both IPv4 and IPv6:
  - Obsolete and not recommended: tcp6 and udp6
  - Recommended: use two separate entries that differ only in the proto field. One entry has tcp and the other has tcp6only, or udp plus udp6only.

See EXAMPLES for an example of a configuration of a service that supports both IPv4 and IPv6.

inetd Methods

inetd provides the methods listed below for consumption by the master restarter, svc.startd(1M).

start
: Causes inetd to start providing service. This results in inetd beginning to handle smf requests for its managed services and network requests for those services that are in either the online or degraded state.

  In addition, inetd also checks if the inetd.conf(4)–format configuration file it is monitoring has changed since the last inetconv(1m) conversion was carried out. If it has, then a message telling the administrator to re-run inetconv to effect the changes made is logged in syslog.

stop
: Causes inetd to stop providing service. At this point, inetd transitions each of its services that are not in either the maintenance or disabled states to the offline state, running any appropriate methods in the process.

`refresh`
Results in a refresh being performed for each of its managed services and the `inetd.conf`(4) format configuration file being checked for change, as in the `start` method. When a service is refreshed, its behavior depends on its current state:

- if it is in the `maintenance` or `disabled` states, no action is performed because the configuration will be read and consumed when the service leaves the state;
- if it is in the `offline` state, the configuration will be read and any changes consumed immediately;
- if it is in the `online` or `degraded` state and the configuration has changed such that a re-binding is necessary to conform to it, then the service will be transitioned to the `offline` state and back again, using the new configuration for the bind;
- if it is in the `online` state and a re-binding is not necessary, then the `inetd_refresh` method of the service, if provided, will be run to allow `online` `wait`–type services to consume any other changes.

オプション

No options are supported.

オペランド

configuration-file
Specifies an alternate location for the legacy service file (`inetd.conf`(4)).

`start`|`stop`|`refresh`
Specifies which of `inetd`'s methods should be run.

使用例

例 1 Configuring a Service that Supports Both IPv4 and IPv6

The following commands illustrate the existence of services that support both IPv4 and IPv6 and assign `proto` properties to those services.

```
example# svcs -a | grep mysvc
online         15:48:29 svc:/network/mysvc:dgram4
online         15:48:29 svc:/network/mysvc:dgram6
online         15:51:47 svc:/network/mysvc:stream4
online         15:52:10 svc:/network/mysvc:stream6

# inetadm -M network/rpc/mysvc:dgram4 proto=udp
# inetadm -M network/rpc/mysvc:dgram6 proto=udp6only
# inetadm -M network/rpc/mysvc:stream4 proto=tcp
# inetadm -M network/rpc/mysvc:stream6 proto=tcp6only
```

See `svcs`(1) and inetadm(1m) for descriptions of those commands.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

関連項目

fmd(1M), inetadm(1m), inetconv(1m), svcadm(1M), svccfg(1M), svcs(1), svc.startd(1M), syslog(3C), getnetconfigent(3NSL), getrpcbyname(3NSL), getservbyname(3SOCKET), inetd.conf(4), process(4), syslog.conf(4), attributes(5), smf(5), smf_method(5)

注意事項

The inetd daemon performs the same function as, but is implemented significantly differently from, the daemon of the same name in Solaris 9 and prior Solaris operating system releases. In the current Solaris release, inetd is part of the Solaris Management Facility (see smf(5)) and will run only within that facility.

The /etc/default/inetd file has been deprecated. The functionality represented by the properties ENABLE_CONNECTION_LOGGING and ENABLE_TCP_WRAPPERS are now available as the tcp_trace and tcp_wrappers properties, respectively. These properties are described above, under “Service Properties”.

名前

init, telinit – プロセス制御の初期化

形式

```
/sbin/init  [0123456abcQqSs]

/etc/telinit  [0123456abcQqSs]
```

機能説明

init はデフォルトの原始ユーザープロセスです。(ブート中にカーネルに渡されるオプションによっては、別の原始ユーザープロセスが呼び出されることもあります。kernel(1M) を参照してください)。init はサービス管理機能の中心のコンポーネントである、svc.configd(1M) および svc.startd(1M) を初期化し、初期化に失敗した場合はこれらのコンポーネントを再起動します。下位互換性を保つため、init は /etc/inittab に従って汎用プロセスの起動および再起動も行います (下記参照)。

下記の実行レベルおよびシステムブートの説明は、互換性を保つ目的でのみ記述されており、それ以外の場合はサービス管理機能 smf(5) によって旧式とされています。

init の障害

init がシステムのシャットダウン以外の何らかの理由で終了した場合、プロセス ID 1 で再起動されます。

定義されている実行レベル

任意の時点で、システムは 8 つの実行レベルのいずれか 1 つにあります。実行レベルはソフトウェア構成で、そこでは選択されたプロセスグループだけが存在します。各実行レベルに対する、init によって生成されるプロセスは、/etc/inittab で定義されています。init は 8 つの実行レベル、つまり 0–6 および S または s (S と s は同じ) のいずれか 1 つを取ることができます。実行レベルは、特権ユーザーが /sbin/init を実行すると変更されます。

init とシステムの起動

システムが起動されると、init が呼び出され、次のイベントが発生します。最初に init は、/etc/default/init を読み込み、環境変数を設定します。通常はここで、TZ (タイムゾーン) およびロケール関連の環境、すなわち LANG、LC_CTYPE などが設定されます(このページの最後にある「ファイル」の項を参照)。次に init は、/etc/inittab を調べ、initdefault エントリを探します(inittab(4) のマニュアルページを参照)。initdefault のエントリが、

存在する場合　オプションまたはマイルストーンのプロパティが svc.startd(1M) に指定されていない場合のみ、init は通常、このエントリで指定された実行レベルを開始時の最初の実行レベルとして使用します。

存在しない場合　サービス管理機能 smf(5) は、svc.startd(1M) で指定された構成を調べ、オプションまたはマイルストーンのプロパティで指定されたマイルストーンを入力します。

/etc/inittab 内の initdefault エントリは、次の実行レベルに対応します。

S または s　init はシングルユーザー状態に移行します。この状態では、システムコンソールデバイス (/dev/console) が読み書き用にオープンされ、コマンド /sbin/su (su(1M) のマニュアルページを参照) が呼び出され

ます。init または telinit のどちらか一方を使用して、システムの実行レベルを変更します。(ファイルの終わり (EOF) を使用して) シェルを終了させたが、/etc/inittab が存在しない場合、init は再びシングルユーザー状態に戻すだけです。

0-6 init は対応する実行レベルに移行します。実行レベル 0、5、および 6 はシステム停止用に予約されています。実行レベル 2、3、および 4 は、マルチユーザー操作用に使用できます。

電源投入後に、init が最初のシングルユーザー状態以外の実行レベルに移行する場合、init はまず、/etc/inittab を走査して boot および bootwait エントリを探します (inittab(4) のマニュアルページを参照)。移行する実行レベルがエントリの実行レベルと一致している場合は、/etc/inittab の他のプロセスが実行される前に、これらのエントリが実行されます。このようにすると、ファイルシステムのマウントなど、オペレーティングシステムの特殊な初期化を先に終えてから、ユーザーにシステムを使用させることができます。次に init は、/etc/inittab を走査し、その実行レベルで処理すべき他のすべてのエントリを実行します。

/etc/inittab の各プロセスを生成するために、init は各エントリを読み込み、再生成が必要なエントリごとに、子プロセスを生成します。/etc/inittab で指定された全プロセスを生成すると、init は、次の状態のために待機します。子孫プロセスの1 つが消滅する、powerfail シグナルを受けとる、システムの実行レベルの変更を要求する他の init または telinit プロセスから送られるシグナルを受けとる。これらの状態のいずれか 1 つが発生すると、init は /etc/inittab を再検査します。

inittab に関するその他の事項

/etc/inittab にはいつでも新しいエントリを追加できますが、init は上記の 3 つの状態のいずれかが発生するまで待機し続けるので、/etc/inittab を再検証しません。この状況を回避するには、init Q または init q コマンドを使用して、init が /etc/inittab をただちに再検査するようにします。

起動時に init が呼び出されたとき、また、システムがシングルユーザー状態から別の実行状態に移行するたびに、init はコンソールの ioctl(2) 状態をファイル /etc/ioctl.syscon に格納されているモードに設定します。init はシングルユーザー状態が移行するたびに、このファイルに書き込みを行います。

実行レベルの変更

実行レベルの変更が要求されると、init は目標の実行レベルで定義されていないすべてのプロセスに警告シグナル (SIGTERM) を送ります。init は 5 秒間待機したのち、終了シグナル (SIGKILL) を送信することによって、これらのプロセスを強制終了させます。さらに、init は、実行レベルが変更中であることを svc.startd(1M) に通知します。すると、svc.startd(1M) は、その実行レベル変更に対応するマイルストーンが依存している一連のサービスに、システムを制限します。

init によって生成されたプロセスが終了したことを通知するシグナルを受信すると、init は、その事実と原因を /var/adm/utmpx と /var/adm/wtmpx (存在していれば) に記録します (who(1) を参照)。生成されたプロセスの履歴は /var/adm/wtmpx に記録されます。

init は powerfail シグナル (SIGPWR) を受信した場合、/etc/inittab を走査し、タイプが powerfail と powerwait の特殊なエントリを探します。これらのエントリが呼び出されてから (実行レベルが許可する場合)、その後の処理が実行されます。このように、init はオペレーティングシステムの停止中にさまざまなクリーンアップおよび記録機能を実行します。

/etc/defaults/init の環境変数

タイムゾーンや文字の書式といった環境変数のデフォルト値を /etc/default/init で設定できます。環境変数の一覧については、「ファイル」の項を参照してください。

telinit

/sbin/init にリンクされている telinit は、init のアクションを指示する場合に使用します。telinit は、1 文字の引数を取り、適切なアクションを実行するように、init にシグナルを送ります。

セキュリティー

init は pam(3PAM) を使用してセッションを管理します。/etc/pam.conf に記述されている PAM 構成ポリシーにより、init に使用されるセッション管理モジュールが指定されます。次に、pam.conf ファイルの抜粋を示します。UNIX セッション管理モジュールを使用するinit のエントリが指定されています。

```
init    session    required     pam_unix_session.so.1
```

init サービスに対応するエントリがない場合、「other」のサービスのエントリが使用されます。

オプション

0
: ファームウェアモードに移行します。

1
: システムをシステム管理者モードにします。すべてのローカルファイルシステムがマウントされます。いくつかの重要なコアプロセスだけが実行を続けます。このモードはオプションのユーティリティーパッケージをインストールする場合など、管理作業を行うためのものです。すべてのファイルにアクセスできます。ユーザーはシステムにログインできません。

  この要求は、smf(5) がシステムのマイルストーンを次の値に制限するための要求に対応しています。svc:/milestone/single-user:default。

2
: システムをマルチユーザーモードにします。すべてのマルチユーザー環境用の端末プロセスおよびデーモンが生成されます。一般に、この状態を、マルチユーザー状態といいます。

  この要求は、smf(5) がシステムのマイルストーンを次の値に制限するための要求に対応しています。svc:/milestone/multi-user:default。

3	ネットワークを介してローカル資源を使用できるようにすることで、マルチユーザーモードを拡張します。 この要求は、smf(5) がシステムのマイルストーンを次の値に制限するための要求に対応しています。svc:/milestone/multi-user-server:default。
4	代替のマルチユーザー環境構成として定義できます。システム運用上は必要なく、通常は使用されません。
5	電源を切っても安全なようにマシンを停止します。可能であれば電源を自動的に切断します。
6	オペレーティングシステムを停止したあと、/etc/inittab の initdefault エントリに定義されている状態でリブートします。
a、b、c	/etc/inittab エントリに a、b、または c という実行レベルが設定されているときだけ処理します。これらは擬似状態であり、特定のコマンドを実行するように定義できますが、現在の実行レベルが変更されるわけではありません。
Q、q	/etc/inittab を再検査します。
S、s	シングルユーザーモードに移行します。適切な形式の /etc/inittab ファイルがなくてもかまわない唯一の実行レベルです。このファイルが存在しない場合、デフォルトで init が入ることができる唯一の正当な実行レベルはシングルユーザーモードだけです。シングルユーザーモードでは、基本システム動作に必要なファイルシステムがマウントされます。システムがシングルユーザーモードに移行すると、これらのファイルシステムは (リモートファイルサーバーが提供している場合でも) マウントされたままになります。他のローカルファイルシステムもマウントされたままになります。シングルユーザーモードへの切り替え時に、init または init.d によって開始された、マルチユーザーモードでしか実行してはならないプロセスはすべて強制終了されます。さらに、utmpx エントリが設定されているプロセスもすべて強制終了されます。この最後の状態では、SAC が起動したすべてのポートモニターが確実に強制終了され、これらのポートモニターによって起動されたサービスも、ttymon login ログインサービスを含めてすべて強制終了されます。 この要求は、smf(5) がシステムのマイルストーンを次の値に制限するための要求に対応しています。svc:/milestone/single-user:default。

ファイル

/dev/console	システムコンソール装置
/etc/default/init	環境変数とそのデフォルト値を指定します。たとえば、タイムゾーン変数 TZ の場合、TZ=US/Pacific のように指定できます。変数は次のとおりです。

TZ	タイムゾーン情報 (ctime(3C) のマニュアルページを参照) またはタイムゾーン情報ファイル /usr/share/lib/zoneinfo の名前のどちらか一方を指定します。 この設定を変更する前に、TIMEZONE(4) のマニュアルページを参照してください。
CMASK	init が使用し、すべてのプロセスが init プロセスから継承するマスク (umask(1) のマニュアルページを参照)。設定されていない場合、init はカーネルから継承したマスクを使用します。CMASK の設定に関係なく、init は必ず、022 で umask の適用を試みてからファイルを作成します。
LC_CTYPE	文字の種類の情報
LC_MESSAGES	翻訳メッセージ
LC_MONETARY	通貨の書式情報
LC_NUMERIC	数値の書式情報
LC_TIME	時刻の書式情報
LC_ALL	設定されている場合、他のすべての LC_* 環境変数でこの値が使用されます。
LANG	LC_ALL が設定されてなく、かつ特定の LC_* も設定されていない場合は、これら環境変数に LANG の値が使用されます。

/etc/initpipe	内部通信用の名前付きパイプ
/etc/inittab	init によるプロセスディスパッチ制御
/etc/ioctl.syscon	シングルユーザー状態に移行したときに init によって保存された、コンソールの ioctl (入出力制御) 状態
/var/adm/utmpx	ユーザーアクセスおよび管理情報
/var/adm/wtmpx	ユーザーアクセスおよび管理情報の履歴
/var/run/init.state	障害から回復するのに必要な init 状態

属性 属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

login(1), sh(1), stty(1), who(1), shutdown(1M), su(1M), ttymon(1M), ioctl(2), kill(2), ctime(3C), pam(3PAM), inittab(4), pam.conf(4), utmpx(4), attributes(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), termio(7I)

診断

2 分間に 10 回以上、/etc/inittab 内の 1 つのエントリが再起動されていることを検出すると、init はそのエントリのコマンド文字列に誤りがあるとみなして、システムコンソール上にエラーメッセージを表示します。さらに 5 分経過するか、あるいはユーザーが生成した init または telinit からシグナルを受信するまで、init はそのエントリを再起動することを拒否します。このようにすることで、inittab ファイルに入力ミスがあった場合や /etc/inittabで参照されているプログラムが削除された場合でも、init がシステム資源を使い果たすのを防止できます。

注意事項

init および telinit を実行できるのは、特権ユーザーだけです。

/etc/inittabで、S または s 状態をむやみに使用してはなりません。このファイルを変更するときに注意すべきことは、この状態を initdefault 以外の行に追加しないことです。

/etc/inittab の initdefault エントリでデフォルトの状態が指定されていない場合は、状態 6 になります。その結果、システムはファームウェアへのアクセスと再起動を繰り返すループに陥ります。

システムの起動時に utmpx ファイルを作成できない場合、システムは /etc/inittab の initdefault エントリで指定されている状態に関係なく、“s” の状態で起動します。/var ファイルシステムにアクセスできない場合には、この状況が発生することがあります。

システムが S (または s) 状態に移行するときは、/etc/nologin ファイル (nologin(4) のマニュアルページを参照) が作成されます。その後、実行レベル 2 に移行すると、/etc/rc2.d ディレクトリにあるスクリプトによってこのファイルが削除されます。

init は内部通信に名前付きパイプ /etc/initpipe を使用します。

pam-unix モジュールは、将来のリリースではサポートされなくなる可能性があります。同様の機能は、pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5)、および pam_unix_session(5) で提供されています。

名前　installer – Solaris Web Start インストールユーティリティ

形式

```
installer  [-locales list] [-nodisplay] [-noconsole]
 [-debug]
```

機能説明

`installer` ユーティリティは、Solaris Web Start のウィザードを起動します。ユーザーは、表示される一連のウィザードの指示に沿ってインストールを行います。`installer` ユーティリティは、Solaris に同梱されている別ソフトウェアの CD のトップディレクトリに含まれています。

`installer` が含まれている CD にデスクトップのファイルマネージャからアクセスしている場合は、`installer` のアイコンをダブルクリックすることによって、ウィザードを起動することができます。スーパーユーザーになっていない場合は、スーパーユーザーのパスワードを入力するように指示メッセージが表示されます。

`installer` ユーティリティは、他の UNIX スクリプトから実行することもできます。スクリプトから実行する場合は通常、`installer` ユーティリティを `-nodisplay` オプション付きで実行するようにします。対話形式でないスクリプトを使用する場合は、`-nodisplay` オプションを追加します。

オプション　次のオプションを指定できます。

`-locales` *list*　インストールする製品のロケールを選択します。*list* に指定されたロケールがインストールメディアに提供されていれば、そのロケールの製品がインストールされます。ロケールは、`-locales` オプションの後にカンマで区切って指定します。たとえば次のように指定すると、

```
installer -locales fr,de,it
```

フランス語 (`fr`)、ドイツ語 (`de`)、イタリア語 (`it`) ロケールの製品がインストールされます。

`-nodisplay`　GUI (グラフィカルユーザーインタフェース) を使用せずにインストールを行います。`-locales` オプションでロケールを指定していない場合は、デフォルトのロケール (英語) の製品をインストールします。

`-noconsole`　対話型テキストコンソールデバイスを使用せずにインストールを行います。対話型でない UNIX スクリプトで使用する場合に、`-nodisplay` オプションと組み合わせて使用すると便利です。

`-debug`　インストール中の処理状況を示す情報を出力します。このオプションは、おもにインストール処理を診断したいときに使用します。

ファイル　`/var/sadm/install/logs`　インストールログファイルの保存ディレクトリ

関連項目　prodreg(1M)

名前　installgrub – ディスクパーティションまたはフロッピーへの GRUB のインストール

形式　`/sbin/installgrub [-fm] stage1 stage2 raw-device`

機能説明　`installgrub` コマンドは x86 専用のプログラムです。GRUB は GRand Unified Bootloader を表します。

`installgrub` は、GRUB のステージ 1 ファイルとステージ 2 ファイルを、特定のディスクパーティションのブート領域にインストールします。`-m` オプションが指定された場合、`installgrub` は、ステージ 1 ファイルをディスクのマスターブートセクターにインストールします。

オプション　`installgrub` コマンドで使用できるオプションは、次のとおりです。

`-f`　マスターブートセクターの上書き時に対話を抑制します。

`-m`　GRUB の *stage1* を、マスターブートセクター上に対話的にインストールします。

オペランド　`installgrub` コマンドで使用できるオペランドは、次のとおりです。

stage1　GRUB のステージ 1 ファイルの名前。

stage2　GRUB のステージ 2 ファイルの名前。

raw-device　GRUB コードのインストール先となるデバイスの名前。これは、読み書き可能な文字デバイスである必要があります。ディスクデバイスの場合、GRUB メニューファイルの格納先となるスライスを指定します。(Solaris の場合、これはルートスライスになる)。フロッピーディスクの場合、これは `/dev/rdiskette` になります。

使用例

例 1　ハードディスクスライスへの GRUB のインストール

次のコマンドは、ルートスライスが `c0d0s0` であるようなシステム上に、GRUB をインストールします。

```
example# /sbin/installgrub /boot/grub/stage1 \
         /boot/grub/stage2 /dev/rdsk/c0d0s0
```

例 2　フロッピーへの GRUB のインストール

次のコマンドは、フォーマット済みのフロッピー上に GRUB をインストールします。

```
example# mount -F pcfs /dev/diskette /mnt
# mkdir -p /mnt/boot/grub
# cp /boot/grub/* /mnt/boot/grub
# umount /mnt
# cd /boot/grub
```

例 2　フロッピーへの GRUB のインストール　(続き)

```
# /sbin/installgrub stage1 stage2 /dev/rdiskette
```

ファイル

`/boot/grub`　GRUB ファイルが存在しているディレクトリ。

属性

属性についての詳細は、マニュアルページの `attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	開発中

関連項目

boot(1M), fdisk(1M), fmthard(1M), `kernel`(1M), `attributes`(5)

警告

マスターブートセクター上に GRUB をインストールする (`-m` オプション) と、マシン上に現在インストールされているブートマネージャーが上書きされます。どの `fdisk` パーティションがアクティブになっているかにかかわらず、システムは常に Solaris パーティション内の GRUB をブートするようになります。

名前

install_scripts, add_install_client, add_to_install_server, rm_install_client, setup_install_server, check – Solaris ソフトウェアをインストールするためのスクリプト

形式

media-mnt-pt/Solaris_*XX*/Tools/add_install_client
[-i *IP_address*]
[-e *Ethernet_address*] [-s *server_name* : *path*]
[-c *server_name* : *path*]
[-n [*server*] : *name_service* [(*netmask*]]
[-p *server_name* : *path*]
[-t *install_boot_image_path*] *host_name platform_group*

media-mnt-pt/Solaris_*XX*/Tools/add_install_client -d
[-s *server_name*:*path*]
[-c *server_name*:*path*] [-p *server_name*:*path*]
[-t *install_boot_image_path*] [-f *boot_file_name*]
platform_name platform_group

media-mnt-pt/Solaris_*XX*/Tools/add_install_client -d
[-s *server_name*:*path*]
[-c *server_name*:*path*] [-p *server_name*:*path*]
[-t *install_boot_image_path*] [-f *boot_file_name*] -e *Ethernet_address*
[-b *property=value*] *platform_group*

media-mnt-pt/Solaris_*XX*/Tools/add_to_install_server [-s]
[-p *product_image_path*] *install_server_path*

media-mnt-pt/Solaris_*XX*/Tools/jumpstart_sample/check
[-p *install_dir_path*]
[-r *rulesfile*]

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client *host_name*

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client *platform_name*

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client -e *Ethernet_address*

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client -f *boot_file_name*

media-mnt-pt/Solaris_*XX*/Tools/setup_install_server [-b]
[-t *install_boot_image_path*] [-w *wanboot_image_path*] *install_dir_path*

機能説明

これらのコマンドは、Solaris Software CD あるいは DVD のスライス `0` に置かれています。(ここで言う CD や DVD という用語は「インストール媒体」を示します)。Solaris インストール媒体をローカルディスクにコピーしている場合、`media_mnt_pt` はコピーした Solaris インストール媒体へのパスです。これらのコマンドは、さまざまなインストール作業に使用できます。

Solaris_*XX* の *XX* は、使用している Solaris リリースのバージョン番号です。

`add_install_client` コマンドには 3 種類の形式があります。「形式」の項を参照してください。

ネットワークを使用してインストールするクライアントを追加する場合は、次の形式の `add_install_client` を使用します (これらのコマンドは `bootparams`(4) ファイルを更新します)。`add_install_client` コマンドは、インストールサーバーの Solaris インストールイメージ (マウントした Solaris インストール媒体、またはハードディスクにコピーしている Solaris インストール媒体) あるいはブートサーバーの起動ディレクトリ (ブートサーバーが必要な場合) から実行する必要があります。Solaris インストールイメージまたはブート専用ディレクトリの Solaris リリースは、クライアントにインストールする予定の Solaris リリースと同じでなければなりません。

media-mnt-pt`/Solaris_`*XX*`/Tools/add_install_client`
`[-i` *IP_address*`]`
`[-e` *Ethernet_address*`] [-s` *server_name* `:` *path*`]`
`[-c` *server_name* `:` *path*`]`
`[-n [`*server* `] :` *name_service* `[(` *netmask*`]]`
`[-p` *server_name* `:` *path*`] [-t` *install_boot_image_path*`]`
host_name platform_group

プラットフォームグループ内のあるプラットフォームのインスタンスをインストールサーバーに追加する場合は、次の形式の `add_install_client` コマンドを使用します。このグループは DHCP を使用して起動および構成されます。スクリプトは、必要な構成作業をサーバー上で実行し、ユーザーがそのグループのために DHCP サーバーに追加しなければならないデータを出力します。

media-mnt-pt`/Solaris_`*XX*`/Tools/add_install_client -d`
`[-s` *server*`:`*path*`]`
`[-c` *server*`:`*path*`] [-p` *server*`:`*path*`]`
`[-t` *install boot image path*`]`
`[-t` *install_boot_image_path*`] [-f` *boot file name*`]`
platform_name platform_group

インストールサーバーに単一のクライアントを追加する場合は、次の形式の `add_install_client` コマンドを使用します。このクライアントは DHCP を使用して起動および構成されます。スクリプトは、必要な構成作業をサーバー上で実行し、ユーザーがそのクライアントのために DHCP サーバーに追加しなければならないデータを出力します。すでに使用しているものにも、上記のように `-f` フラグを追加する必要があります。`-f` を使用すると、ユーザーは所定のクライアントに使用する起動ファイルの名前を指定できます。

media-mnt-pt`/Solaris_ `*XX*`/Tools/add_install_client -d`
`[-s` *server_name*`:`*path*`]`
`[-c` *server_name*`:`*path*`] [-p` *server_name*`:`*path*`]`
`[-t` *install_boot_image_path*`] [-f` *boot_file_name*`]`
`-e` *Ethernet_address platform_group*
`[-b` *property*`=`*value*`]` *platform_group*

x86 アーキテクチャの Pre-boot eXecution Environment (PXE) クライアントを登録する場合は、必ず `-d` オプションを使用してください。x86 PXE クライアントは構成に DHCP を使用します。

他の Solaris インストール媒体とネットワークインストールサーバー上の既存のイメージをマージする場合は、add_to_install_server を使用します。マージ可能なインストール媒体 (1 枚目を除く各 OS CD または Language CD) には専用の add_to_install_server スクリプトが含まれています。配布されたインストール媒体に含まれているもの以外の add_to_install_server スクリプトを使用しないでください。

rules ファイル (カスタム JumpStart インストールを使用する場合に限り必要) のルールを検証する場合は、check を使用します。

ネットワークインストール用のクライアントを削除する場合は、rm_install_client を使用します (このコマンドは bootparams(4) ファイルを更新します)。

Solaris インストール媒体を (インストールサーバーを設定するために) ディスクにコピーする、(WANboot インストールサーバーを設定するために) WANboot ミニルートイメージを構築する、または Solaris インストール媒体のブートソフトウェアだけを (ブートサーバーを設定するために) ディスクにコピーする場合は、setup_install_server を使用します。ネットワーク経由でクライアントをインストールするには、インストールサーバーが必要です。インストールサーバーとインストールするクライアントが異なるサブネットに属している場合、ネットワークインストールを行うには、ブートサーバーも必要です (ブートサーバーはクライアントのサブネットに配置する必要があります)。

オプション

add_install_client には、次のオプションを指定できます。

-b *property=value*
: ブートサーバーの TFTP ディレクトリ (デフォルトは /tftpboot) にあるクライアント特有の menu.lst ファイルにプロパティ値を設定します。クライアントに特有のブートプロパティを設定する場合に、このオプションを使用します。

  このオプションは x86 クライアントにのみ使用できます。このオプションを使用するときには、必ず -d オプションと -e を同時に指定してください。

-c *server_name*:*path*
: このオプションが必要なのは、カスタム JumpStart インストール用の JumpStart ディレクトリを指定する場合だけです。*server_name* は、JumpStart ディレクトリが置かれているサーバーのホスト名です。*path* は JumpStart ディレクトリの絶対パスです。

-d
: DHCP クライアントを指定します。

-e *Ethernet_address*
: インストールするシステムの Ethernet アドレスを指定します。

-f
: インストールするクライアントの *boot_file_name* を指定します。

-i *IP_address*
インストールするクライアントの IP アドレスを指定します。

-n [*server*]: *name_service*[(*netmask*)]
このオプションでは、システム構成時に使用するネームサービスを指定します。`bootparams`(4) ファイル内の `ns` キーワードを設定します。

name_service
有効なエントリは `nis`、`nisplus`、および `none` です。

netmask
数字を 4 つずつピリオドで区切って並べたもので、IP アドレスのどの部分がネットワーク部分で、どれがホスト部分かを指定します。

server
サーバーの名前または指定したネームサービスの IP アドレス。指定したサーバーが異なるサブネット上にある場合は、クライアントからサーバーにアクセスするために、*netmask* が必要なことがあります。

-p *server_name*: *path*
このオプションは、ユーザーによって定義された `sysidcfg` ファイルを含む NFS または ZFS 共有ディレクトリを指定するために使用します。クライアントはブートすると、システムとネットワークの識別情報を取得するために、このディレクトリ内で `sysidcfg` という名前と明確に一致するファイルを読み取ろうとします。*server_name* は有効なホスト名または IP アドレスです。*path* は、`sysidcfg` ファイルを含むファイルサーバー上のディレクトリの絶対パスです。

-s *server_name*:*path*
このオプションが必要なのは、ブートサーバーから `add_install_client` を使用する場合だけです。サーバー名およびこのインストールで使用する Solaris インストールイメージの絶対パスを指定します。*path* はマウントされている Solaris インストール媒体のパスまたは Solaris インストール媒体のコピーが置かれているディレクトリのパスです。

-t
代替ミニルートを指定できるようにします。

`add_to_install_server` コマンドには、次のオプションを指定できます。

-p
コピーするインストール媒体 (補助的な製品が収められている) の位置を指定します。

-s
ユーザーが、インストールする必要がある製品だけをリストから選択できるようにします。

`check` コマンドには、次のオプションを指定できます。

`-p` *install_dir_path*
使用中のシステムの `check` スクリプトではなく、指定された Solaris インストールイメージの `check` スクリプトを使用することによって、ルールファイルを検査します。`install_dir_path` は、ローカルディスク上またはマウントした Solaris インストール媒体上の Solaris インストールイメージのパスです。

旧バージョンの Solaris が稼働しているシステムでは、このオプションを使用して、最新バージョンの `check` を実行できます。

`-r` *rulesfile*
`rules` 以外の名前のルールファイルを指定します。このオプションを使用すると、ルールの妥当性を検証してから、ルールファイルに組み込むことができます。`check` はルールが有効であるかどうかを報告するだけで、カスタム JumpStart インストールに必要な `rules.ok` ファイルは作成しません。

`rm_install_client` コマンドには、次のオプションを指定できます。

`-e` *Ethernet_address*
削除されるシステムの Ethernet アドレスを指定します。

`-f`
削除されるクライアントの *boot_file_name* を指定します。

`setup_install_server` コマンドには、次のオプションを指定できます。

`-b`
サーバーの設定を、ブートサーバー専用にします。

`-t`
代替ミニルートを指定できるようにします。

`-w`
WANboot ミニルートイメージを構築します。

オペランド

`add_install_client` コマンドには、次のオペランドを指定できます。

host_name
インストールするクライアントの名前です。

platform_group
特定のソフトウェアを配布するために、ベンダーが定義したハードウェアプラットフォームグループ。以下に有効なプラットフォームグループの例を示します。

システム	プラットフォームグループ
x86	`i86pc`

システム	プラットフォームグループ
Sun Fire 4800	sun4u

システムのプラットフォームグループを調べるには、(-m オプションを指定して) uname(1) コマンドを使用します。

platform_name
システムのプラットフォーム名を調べるには、-i オプションを指定して uname(1) コマンドを使用します。

次に、uname command コマンドを使用して Ultra 10 のシステムプラットフォーム名を調べる例を示します。

```
uname -i
```

システムは、次のように応答します。

```
SUNW,Ultra-5_10
```

システムのプラットフォーム名は SUNW,Ultra-5_10 です。

次のコマンドは、Ultra 10 用の add_install_client を呼び出します。

```
add_install_client -d SUNW,Ultra-5_10 sun4u
```

IA32 プラットフォームでは、プラットフォーム名はつねに SUNW.i86pc です。

次のコマンドは、IA32 プラットフォーム用の add_install_client を呼び出します。

```
add_install_client -d SUNW.i86pc i86pc
```

install_boot_image_path
-t オプションで指定した代替ミニルートのパス名です。

rm_install_client コマンドには、次のオペランドを指定できます。

host_name
削除するクライアントの名前です。

platform_name
削除されるクライアントのプラットフォーム名です。上記オペランドの説明を参照してください。

Ethernet_address
削除されるクライアントの Ethernet アドレスです。

boot_file_name
削除されるブートファイルの名前です。

setup_install_server コマンドには、次のオペランドを指定できます。

install_dir_path
Solaris ソフトウェアのコピー先となるディレクトリの絶対パスです。このディレクトリは空でなければなりません。

wanboot_image_path
WANboot ミニルートイメージを含むファイルが作成されるディレクトリの絶対パスです。

install_boot_image_path
-t オプションで指定した代替ミニルートのパス名です。

使用例

例 1 add_install_client の使用

次の add_install_client コマンドは、マウント済み Solaris インストール媒体からネットワークインストール用のクライアントを追加します。

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./add_install_client system_2/sun4u
```

例 2 add_install_client の使用

次の add_install_client コマンドは、ネットワークインストール用のクライアントをインストールサーバーに追加します。カスタム JumpStart インストールを実行するためのルールファイルおよびプロファイルファイルが置かれている JumpStart のサーバーとディレクトリのパスを -c オプションで指定します。また、Solaris インストール媒体は /export/install ディレクトリにコピーされています。

```
example# cd /export/install/Solaris_10/Tools
example# /add_install_client
          -c install_server:/jumpstart system_1 i86pc\
example# ./add_install_client -c install_server:/jumpstart\
          system_2 i86pc
```

例 3 add_install_client の使用

次の add_install_client コマンドは、次の起動ファイルを使用する特定の sun4u プラットフォームマシン (8:0:20:99:88:77) に対するサポートを追加します。起動ファイルは sun4u.solaris10 です。

```
example# add_install_client -d -f sun4u.solaris10\
          -e 8:0:20:99:88:77 sun4u
```

例 4 add_install_client の使用

次の add_install_client コマンドは、PXE 規格を使用してネットワークから起動する x86 クライアントを追加します。

```
example# add_install_client -d -s svrname:/mnt/export/root\
          SUNW.i86pc i86p
```

例 5 add_to_install_server の使用

次の add_to_install_server コマンドは、インストール媒体上のすべての製品ディレクトリにあるパッケージを既存のインストールサーバーにコピーします。

```
example# cd /cdrom/cdrom0/s0
example# ./add_to_install_server /export/Solaris_10
```

例 6 check の使用

次の check コマンドは、カスタム JumpStart インストールに使用するルールファイルの構文を検査します。

```
example# cd jumpstart_dir_path
example# ./check -p /cdrom/cdrom0/s0
```

例 7 rm_install_client の使用

次の rm_install_client コマンドは、ネットワークインストール用クライアントを削除します。

```
example# cd /export/install/Solaris_10/Tools
example# ./rm_install_client holmes
example# ./rm_install_client watson
```

例 8 setup_install_server の使用

次の setup_install_server コマンドは、マウント済み Solaris インストール媒体をローカルディスク上にある /export/install というディレクトリにコピーします。

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server /export/install
```

例 9 setup_install_server の使用

次の setup_install_server コマンドは、マウント済みの Solaris インストール媒体のブートソフトウェアを、サブネット用ブートサーバーになるシステムの /boot_dir というディレクトリにコピーします。

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server -b /boot_dir
```

例 10 setup_install_server の使用

setup_install_server はデフォルトで、マウント済み Solaris 配布ディスク上の Solaris ../Tools/Boot にあるインストールブートディレクトリを検索します。

例 10 setup_install_server の使用 (続き)

以前に ./setup_install_server -b /boot_dir コマンドを使用して、ネットワークブートサーバー上に作成したディレクトリと別のディレクトリが必要な場合は、-t オプションを使用できます。

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server -t /boot_dir /export/install
```

例 11 setup_install_server と WANboot オプションの使用

次の setup_install_server コマンドは WANboot ミニルートファイルシステムのイメージを作成し、それを /wanboot_dir/miniroot に保存します。

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server -w /wanboot_dir /export/install
```

例 12 x86: ネットワークインストール時に使用するシリアルコンソールの指定 (インストール媒体)

次の例は、x86 インストールクライアントをインストールサーバーに追加して、インストール時に使用するシリアルコンソールを指定する方法を示します。この例では、インストールクライアントを次の方法で設定します。

- -d オプションは、クライアントが DHCP を使用してインストールパラメータを設定することを示します。
- -e オプションは、Ethernet アドレスが 00:07:e9:04:4a:bf であるクライアントでのみインストールが行われることを示します。
- 1 つめと 2 つめの -b オプションは、インストールプログラムが入力デバイスおよび出力デバイスとしてシリアルポート ttya を使用することを示します。

```
install server# cd /export/boot/Solaris_10/Tools
install server# ./add_install_client -d -e "00:07:e9:04:4a:bf" \
     -b "input-device=ttya" -b "output-device=ttya" \
     i86pc
```

-b オプションで使用できるブートプロパティの変数とその値について詳細は、eeprom(1M) を参照してください。

例 13 ネットワークインストール時に使用するブートデバイスの指定 (インストール媒体)

次の例は、x86 インストールクライアントをインストールサーバーに追加して、インストール時に使用するブートデバイスを指定する方法を示します。インストールクライアントの設定時にブートデバイスを指定すると、インストール時、Device Configuration Assistant (デバイス構成用補助) はこの情報の入力プロンプトを表示しません。

例 13 ネットワークインストール時に使用するブートデバイスの指定 (インストール媒体) (続き)

この例では、インストールクライアントを次の方法で設定します。

- -d オプションは、クライアントが DHCP を使用してインストールパラメータを設定することを示します。
- -e オプションは、Ethernet アドレスが 00:07:e9:04:4a:bf であるクライアントでのみインストールが行われることを示します。
- 1 つめと 2 つめの -b オプションは、インストールプログラムが入力デバイスおよび出力デバイスとしてシリアルポート ttya を使用することを示します。
- 3 つめの -b オプションは、インストールプログラムがインストール時に特定のブートデバイスを使用することを示します。
- ブートデバイスのパスは、使用しているハードウェアによって異なります。
- i86pc というプラットフォーム名は、クライアントが x86 ベースのシステムであることを示します。

```
install server# cd /export/boot/Solaris_10/Tools
install server# ./add_install_client -d -e "00:07:e9:04:4a:bf" \
     -b "input-device=ttya" -b "output-device=ttya" \
     -b "bootpath=/pci@0,0/pci108e,16a8@8" i86pc
```

-b オプションで使用できるブートプロパティの変数とその値について詳細は、eeprom(1M) を参照してください。

終了ステータス

次の終了ステータスが返されます。

0
正常終了。

1
エラーが発生した。

属性

次の属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	Solaris CD および (または) DVD (インストール媒体)

関連項目

uname(1), eeprom(1M), bootparams(4), attributes(5)

『Solaris 10 インストールガイド (基本編)』

名前

kdmconfig – OpenWindows および国際化対応のキーボード、ディスプレイ、およびマウスオプションの設定または設定解除

形式

```
kdmconfig

kdmconfig [-fv] [-s hostname] -c | -t | -u | -d filename
```

機能説明

kdmconfig プログラムは、Xsun ウィンドウシステム環境のみに適用されます。このプログラムは、Solaris x86 のデフォルト環境ではなくなっています。Xsun を使用する場合は、コマンド行で kdmconfig を実行し、Xsun を選択してから残りの設定手順を実行する必要があります。

kdmconfig プログラムは、Solaris ソフトウェアが稼動する x86 システム上で、クライアントマシンに関連するキーボード、ディスプレイ、およびマウス情報を、/etc/openwin/server/etc/OWconfig ファイルに設定または設定解除します。また、kdmconfig は、サーバーマシン上の bootparams(4) データベースの *display*、*pointer*、および *keyboard* エントリ、または sysidcfg(4) ファイルの monitor、*keyboard*、*display*、および *pointer* キーワードを設定する場合にも使用できます。kdmconfig は、root または root と同等の権限でのみ実行できます。デバイス選択が完了すると、kdmconfig はユーザーに設定をテストするように要求します。このテストはウィンドウシステムを実行することによって行います。

オプション

次のオプションを指定できます。

-c
: 構成モードでプログラムを実行します。このモードは、OWconfig ファイルを作成または更新する場合に使用します。このオプションを指定して呼び出された kdmconfig はまず、bootparams(4) データベース内の関連する構成情報を探します。また、-s オプションが同時に使用されていないかぎり、デバイスプローブから戻された情報も考慮に入れます。クライアントが利用できる bootparams(4) データベースは、サーバーマシンが bootparamd(1M) デーモンを実行している、クライアントと同じサブネットにあるサーバー上のすべての /etc/bootparams ファイルです。sysidconfig(1M) によって呼び出される kdmconfig には、-c オプションが指定されます。

-d *filename*
: sysidcfg(4) ファイルを設定します。このオプションを指定した場合、-c オプションと同じ画面が表示されますが、指定した情報は sysidcfg(4) キーワード (monitor、*keyboard*、*display* 、および *pointer*) として保存されます。そのため、sysidcfg(4) ファイルを使用して、システムデバイス情報を事前に設定しておくと、インストール時の kdmconfig を省略できます。

  filename は作成される sysidcfg(4) ファイルです。パスを指定しない場合、kdmconfig が実行されるディレクトリに作成されます。指定したディレクトリに *filename* がすでにある場合、既存ファイルにキーワードが付加されます。

`-f`
強制的に画面モードにします。このオプションを指定して呼び出すと、ネットワークプローブは実行されません。クライアントの構成環境をデバッグする場合に便利です。`-s` オプションは `-f` オプションを使用することを意味し、サーバーの設定時にネットワークプローブが省略されます。

`-s` *hostname*
当該マシン上で、指定されたクライアント用の `bootparams`(4) データベースを設定します。このオプションを使用すると、クライアント側で実行した場合と同じ画面が表示されますが、情報は `/etc/bootparams` ファイルに書き込まれます。また、`-s` オプションは `-f` オプションを使用することを意味します。つまり、このオプションを指定すると、プログラムは必ずユーザーに画面を表示します。このオプションは `nsswitch.conf`(4) ファイルを再構成し、ローカルサーバー上で `bootparams`(4) データベースを探します。このオプションを使用できるのは、スーパーユーザーだけです。

`-t`
テストモードでプログラムを実行します。このモードでは、`kdmconfig` はデバイスプローブ情報を使用して、キーボード、ディスプレイ、およびマウスに関する、最新かつ完全な情報が `OWconfig` ファイルに含まれているかどうかを調べます。情報が正確な場合、`kdmconfig` はそのまま終了します。情報に問題がある場合、`kdmconfig` はスーパーユーザーパスワードを要求し、(オプションを指定しないで実行された場合と同様に) 通常の編集セッションに進みます。

`-u`
システムの設定を解除し、「初期状態」に戻します。この状態では、`/etc/openwin/server/etc/OWconfig` ファイルからデバイス構成エントリが削除されているため、デフォルトのキーボード、マウス、およびディスプレイが選択されます。これはディスプレイサーバーにとって、不適切な構成である可能性があります。

`-v`
詳細表示モードを有効にします。`kdmconfig` は通常、どのような出力も生成しません。このオプションは `kdmconfig` が実行したさまざまなアクションを標準エラー出力に記録するので、デバッグ時に役立ちます。

オプションなし　オプションを指定しないで実行すると、現在の構成を編集する目的で `kdmconfig` を使用できます。`kdmconfig` は `OWconfig` ファイルの情報、`bootparams`(4) ファイルから得た情報、およびデバイスプローブから得た情報を使用します。それ以外は、`-c` オプションを使用した場合と同様です。

ファイル

`/etc/openwin/server/etc/OWconfig`	OpenWindows の構成ファイル
`/etc/bootparams`	ディスクレスクライアントが起動時に使用するクライアントリスト
`/etc/nsswitch.conf`	ネームサービスの構成リスト

x86 のみ

`/dev/openprom`	インストールされているデバイスおよび属性

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
アーキテクチャ	x86
使用条件	SUNWos86r

関連項目

bootparamd(1M), sys-unconfig(1m), sysidconfig(1M), bootparams(4), nsswitch.conf(4), sysidcfg(4), attributes(5)

Xorg(1) および xorg.conf(4) のマニュアルページも参照してください。一部の Solaris システムでは、このマニュアルページは /usr/X11/man にあります。これらのマニュアルページは、SunOS マニュアルページコレクションには含まれません。

名前

lockfs – change or report file system locks

形式

```
/usr/sbin/lockfs [-adefhnuw] [-c string] [file-system]...
```

機能説明

`lockfs` is used to change and report the status of file system locks. `lockfs` reports the lock status and unlocks the file systems that were improperly left locked.

Using `lockfs` to lock a file system is discouraged because this requires extensive knowledge of SunOS internals to be used effectively and correctly.

When invoked with no arguments, `lockfs` lists the UFS file systems that are locked. If *file-system* is not specified, and `-a` is specified, `lockfs` is run on all mounted, UFS type file systems.

オプション

The options are mutually exclusive: `wndheuf`. If you do specify more than one of these options on a `lockfs` command line, the utility does not protest and invokes only the last option specified. In particular, you cannot specify a flush (`-f`) and a lock (for example, `-w`) on the same command line. However, all locking operations implicitly perform a flush, so the `-f` is superfluous when specifying a lock.

You must be super-user to use any of the following options, with the exception of `-a`, `-f` and `-v`.

The following options are supported.

`-a` Apply command to all mounted, UFS type file systems. *file-system* is ignored when `-a` is specified.

`-c` *string* Accept a string that is passed as the comment field. The `-c` only takes affect when the lock is being set using the `-d`, `-h`, `-n`, `-u`, or `-w` options.

`-d` Delete-lock (`dlock`) the specified *file-system*. dlock suspends access that could remove directory entries.

`-e` Error-lock (`elock`) the specified *file-system*. elock blocks all local access to the locked file system and returns `EWOULDBLOCK` on all remote access. File systems are elocked by UFS on detection of internal inconsistency. They may only be unlocked after successful repair by `fsck`, which is usually done automatically (see mount_ufs(1M)). elocked file systems can be unmounted.

`-f` Force a synchronous flush of all data that is dirty at the time `fsflush` is run to its backing store for the named file system (or for all file systems.)

It is a more reliable method than using `sync`(1M) because it does not return until all possible data has been pushed. In the case of UFS filesystems with logging enabled, the log is also rolled before returning. Additional data can be modified by the time `fsflush` exits, so using one of the locking options is more likely to be of general use.

`-h` Hard-lock (`hlock`) the specified *file-system*. hlock returns an error on every access to the locked file system, and cannot be unlocked. hlocked file systems can be unmounted.

`-n` Name-lock (`nlock`) the specified *file-system*. nlock suspends accesses that could change or remove existing directories entries.

`-u` Unlock (`ulock`) the specified *file-system*. ulock awakens suspended accesses.

`-v` Enable verbose output.

`-w` Write-lock (`wlock`) the specified *file-system*. wlock suspends writes that would modify the file system. Access times are not kept while a file system is write-locked.

オペランド

The following operands are supported.

file-system A list of path names separated by white spaces.

使用法

See `largefile`(5) for the description of the behavior of `lockfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

使用例

例 1 Using `lockfs -a`

In the following examples, *filesystem* is the pathname of the mounted-on directory (mount point). `Locktype` is one of "`write`," "*name*," "*delete*," "*hard*," or "*unlock*". When enclosed in parenthesis, the lock is being set. `Comment` is a string set by the process that last issued a lock command.

The following example shows the `lockfs` output when only the `-a` option is specified.

```
example# /usr/sbin/lockfs -a
```

Filesystem	Locktype	Comment
/	unlock	
/var	unlock	

example#

例 2 Using `lockfs -w`

The following example shows the `lockfs` output when the `-w` option is used to write lock the `/var` file system and the comment string is set using the `-c` option. The `-a` option is then specified on a separate command line.

例 2 Using `lockfs -w` (続き)

```
example#  /usr/sbin/lockfs -w -c "lockfs: write lock example" /var
example#  /usr/sbin/lockfs -a
```

Filesystem	Locktype	Comment
/	unlock	
/var	write	lockfs: write lock example

example#

例 3 Using `lockfs -u`

The following example shows the `lockfs` output when the `-u` option is used to unlock the `/var` file system and the comment string is set using the `-c` option.

```
example#  /usr/sbin/lockfs -uc "lockfs: unlock example" /var
example#  /usr/sbin/lockfs /var
```

Filesystem	Locktype	Comment
/var	unlock	lockfs: unlock example

example#

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

関連項目

`kill`(1), mount_ufs(1M), `sync`(1M), `attributes`(5), `largefile`(5), `ufs`(7FS),

『Solaris のシステム管理 (基本編)』

診断

file system`: Not owner`
You must be root to use this command.

file system `:Deadlock condition detected/avoided`
A file is enabled for accounting or swapping, on *file system*.

file system`: Device busy`
Another process is setting the lock on *file system*.

名前　lofiadm – lofi を使用してブロックデバイスとして使用可能なファイルを管理する

形式

```
/usr/sbin/lofiadm  -a file [device]

/usr/sbin/lofiadm  -d  file | device

/usr/sbin/lofiadm  [file | device]
```

機能説明

lofiadm は、lofi(7D) (ループバックファイルドライバ) を管理します。lofi(7D) は、ファイルをブロックデバイスに関連付けることを可能にします。関連付けられたファイルは、ブロックデバイスからアクセスできます。ブロックデバイスはファイルシステムのマウント、検査、または修復のために通常のシステムユーティリティーと一緒に使用できるため、これは、ファイルにファイルシステムのイメージ (フロッピーや CD-ROM のイメージなど) が含まれている場合に役立ちます。fsck(1m) および mount(1M) を参照してください。

lofiadm を使用して、ファイルをループバックデバイスとして追加したり、そのような関連付けを削除したり、現在の関連付けに関する情報を表示したりします。

オプション　サポートしているオプションは、以下のとおりです。

-a *file* [*device*]　*file* をブロックデバイスとして追加します。

device が指定されていない場合、使用可能なデバイスが 1 つ選択されます。

device が指定されている場合、lofiadm はそのデバイスの *file* への割り当てを試みます。*device* は使用可能である必要があり、そうでない場合は lofiadm は失敗します。デバイスを指定する機能は、関連付けの特定のセットを再設定するスクリプトで使用するために提供されています。

-d *file* | *device*　関連付けられたブロックデバイスがビジー状態ではない場合は、*file* または *device* 名で指定された関連付けを削除し、ブロックデバイスの割り当てを解除します。

オペランド　次のオペランドがサポートされています。

file　*file* に関連付けられているブロックデバイスを表示します。

device　ブロックデバイス *device* に関連付けられているファイル名を表示します。

引数が指定されていない場合、現在の関連付けのリストを一覧表示します。ファイル名は有効な絶対パス名である必要があります。

ファイルが追加されると、root による読み取りまたは書き込みのためにファイルが開かれます。すべての制限が適用されます (NFS 上のルートアクセスの制限など)。ファイルは、関連付けが削除されるまで開かれたままになります。ブロックデバイスが使用されるまでファイルは実際にアクセスされないため、ブロックデバイスが読み取り専用で開かれた場合

はファイルに書き込まれません。

使用例

例 1　既存の CD-ROM イメージのマウント

CD を作成する前に、Solaris がイメージを認識することを確認する必要があります。lofi を使用すればイメージをマウントでき、これが機能するかどうかを確認できます。

この例では、インターネットからダウンロードした Red Hat 6.0 CD の既存の CD-ROM イメージ (sparc.iso) をマウントします。これはインターネットから入手した mkisofs ユーティリティーで作成されました。

次のように lofiadm を使用して、これにブロックデバイスを接続します。

```
# lofiadm -a /home/mike_s/RH6.0/sparc.iso
/dev/lofi/1
```

lofiadm はデバイスを選択し、デバイス名を標準出力に表示します。次のコマンドを実行して、lofiadm を再度実行できます。

```
# lofiadm
Block Device     File
/dev/lofi/1      /home/mike_s/RH6.0/sparc.iso
```

または、次のコマンドを実行して一方の名前を指定し、もう一方の名前を要求できます。

```
# lofiadm /dev/lofi/1
/home/mike_s/RH6.0/sparc.iso
```

mount コマンドを使用してイメージをマウントします。

```
# mount -F hsfs -o ro /dev/lofi/1 /mnt
```

Solaris がイメージを認識することを確認します。

```
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1           512418  512418       0   100%    /mnt
# ls /mnt
./            RedHat/       doc/          ls-lR         rr_moved/
../           TRANS.TBL     dosutils/     ls-lR.gz      sbin@
.buildlog     bin@          etc@          misc/         tmp/
COPYING       boot/         images/       mnt/          usr@
README        boot.cat*     kernels/      modules/
RPM-PGP-KEY   dev@          lib@          proc/
```

Solaris は CD-ROM イメージをマウントし、ファイル名を認識できます。イメージが正しく作成されたため、確信をもって CD-ROM を作成できます。

例 1　既存の CD-ROM イメージのマウント　　(続き)

最終ステップとして、イメージのマウント解除および切り離しを行います。

```
# umount /mnt
# lofiadm -d /dev/lofi/1
# lofiadm
Block Device             File
```

例 2　フロッピーイメージのマウント

これは例 1 と似ています。

フロッピーディスクに必要なファイルが含まれているが、使用するマシンにフロッピードライブがない場合、`lofi` を使用してフロッピーイメージを含むファイルをマウントすると便利です。`dd` コマンドを使用してイメージをフロッピーにコピーするのに時間をかけたくない場合にも便利です。

これは、x86 プラットフォーム上で Solaris 用の MDB フロッピーを取得する例です。

```
# lofiadm -a /export/s28/MDB_s28x_wos/latest/boot.3
/dev/lofi/1
# mount -F pcfs /dev/lofi/1 /mnt
# ls /mnt
./            COMMENT.BAT*  RC.D/         SOLARIS.MAP*
../           IDENT*        REPLACE.BAT*  X/
APPEND.BAT*   MAKEDIR.BAT*  SOLARIS/
# umount /mnt
# lofiadm -d /export/s28/MDB_s28x_wos/latest/boot.3
```

例 3　ファイル上の UFS ファイルシステムの作成

UFS ファイルシステムをファイル上に作成すると、特にテストスイートに新しいファイルシステムが必要な場合に便利です。テストスイートのためだけにディスクをパーティションに再分割するのは手間がかかりますが、その必要はありません。`lofi` を使用するとファイルに `newfs` を実行できます

ファイルを作成します。

```
# mkfile 35m /export/home/test
```

作成したファイルをブロックデバイスに接続します。`newfs` に必要な文字デバイスも取得するため、`newfs` は次のようになります。

```
# lofiadm -a /export/home/test
/dev/lofi/1
```

例 3　ファイル上の UFS ファイルシステムの作成　　(続き)

```
# newfs /dev/rlofi/1
newfs: construct a new file system /dev/rlofi/1: (y/n)? y
/dev/rlofi/1:   71638 sectors in 119 cylinders of 1 tracks, 602 sectors
        35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
```

ufs はファイル全体を使用できない場合があります。ファイルシステムをマウントおよび使用します。

```
# mount /dev/lofi/1 /mnt
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1            33455       9   30101     1%    /mnt
# ls /mnt
./           ../          lost+found/
# umount /mnt
# lofiadm -d /dev/lofi/1
```

例 4　UNIX ファイル上の PC (FAT) ファイルシステムの作成

次に示す一連のコマンドは、FAT ファイルシステムを UNIX ファイル上に作成します。ファイルは lofiadm によって作成されたブロックデバイスに関連付けられています。

```
# mkfile 10M /export/test/testfs
# lofiadm -a /export/test testfs
/dev/lofi/1
(次のコマンドでは「lofi」ではなく「rlofi」を使用します)。
# mkfs -F pcfs -o nofdisk,size=20480 /dev/rlofi/1
Construct a new FAT file system on /dev/rlofi/1: (y/n)? y
# mount -F pcfs /dev/lofi/1 /mnt
# cd /mnt
# df -k .
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1            10142       0   10142     0%    /mnt
```

環境

lofiadm の実行に影響を与える次の環境変数の詳細については、environ(5) を参照してください。LC_CTYPE、LC_MESSAGES および NLSPATH。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

fsck(1m), mount(1M), mount_ufs(1M), newfs(1M), attributes(5), lofi(7D), lofs(7FS)

注意事項

マウントされたファイルシステムを持つディスクデバイスに直接アクセスしないのと同様に、lofi ファイルドライバを使用する場合を除き、ブロックデバイスに関連付けられているファイルにアクセスしないようにしてください。また、そのようなアクセスを防止するために、適切なアクセス権をファイルに設定することもお勧めします。

関連付けはリブート後に持続されません。必要な場合は、スクリプトを使用して関連付けを再設定できます。

lofiadm の機能およびこの機能を使用できるユーザーは、/dev/lofictl のアクセス権によって制御されます。読み取りアクセス権は、すべての関連付けの表示などのクエリー操作を可能にします。書き込みアクセス権は、関連付けの追加などの状態変更操作の実行に必要です。出荷時の /dev/lofictl は、root によって所有され、グループ sys に所属し、モード 0644 に設定されているため、すべてのユーザーがクエリー操作を実行できますが、変更操作はルートのみに可能です。管理者は、ユーザーに書き込みアクセスを与えて、関連付けの追加または削除を許可することができますが、これはセキュリティーホールになる可能性が非常に高いため、信頼できるグループにのみ与えるようにしてください。

ファイルシステムイメージをマウントする場合は、適切なマウントオプションを使用するように注意してください。特に、出所が不明な UFS イメージには、nosuid マウントオプションが適切な場合があります。また、UFS に logging や forcedirectio を使用する場合のように、いくつかのオプションは役に立たなかったり適切でなかったりすることがあります。互換性を保つために、raw デバイスもブロックデバイスと一緒にエクスポートされます。 たとえば、newfs(1M) にはこれが必要です。

lofiadm (引数なし) の出力は、将来のリリースで変更される可能性があります。

名前

lpadmin – configure the LP print service

形式

```
lpadmin  -p printer {options}

lpadmin  -x dest

lpadmin  -d [dest]

lpadmin  -S print-wheel -T [-A alert-type] [-W minutes]
 [-Q requests]
```

機能説明

`lpadmin` configures the LP print service by defining printers and devices. It is used to add and change printers, to remove printers from service, to set or change the system default destination, to define alerts for printer faults, and to mount print wheels.

オプション

The `lpadmin` command has options for:

- Adding or changing a printer
- Removing a printer destination
- Setting or changing the system default destination
- Setting an alert for a print wheel

The options for each of the above categories are specified in the following subsections.

Several options support the use of lists. A list might contain, for example, user names, printers, printer forms, or content types. A list of multiple items can have the form of either comma-separated names or have the entire list enclosed by double quotes with a space between each name. For example, both lists below are acceptable:

```
one,two,three
"one two three"
```

Adding or Changing a Printer

The first form of the `lpadmin` command (`lpadmin -p` *printer* {*options*}) configures a new printer or changes the configuration of an existing printer. It also starts the print scheduler.

When creating a new printer, one of three options (`-v`, `-U`, or `-s`) must be supplied. In addition, only one of the following can be supplied: `-e`, `-i`, or `-m`; if none of these three options is supplied, the model standard is used. The `-h` and `-l` options are mutually exclusive. Printer and class names must be no longer than 14 characters and must consist entirely of the characters `A-Z`, `a-z`, `0-9`, dash (`-`) and underscore (_). If `-s` is specified, the following options are invalid: `-A`, `-e`, `-F`, `-h`, `-i`, `-l`, `-M`, `-m`, `-o`, `-U`, `-v`, and `-W`.

The following options can appear in any order.

`-A` *alert-type* [`-W` *minutes*]
: The `-A` option is used to define an alert that informs the administrator when a printer fault is detected, and periodically thereafter, until the printer fault is cleared by the administrator. The *alert-types* are:

  `mail`
  : Send the alert message using mail (see `mail`(1)) to the administrator.

`write`
Write the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.

`quiet`
Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the fault has been cleared and printing resumes, messages will again be sent when another fault occurs with the printer.

`showfault`
Attempt to execute a fault handler on each system that has a print job in the queue. The fault handler is `/etc/lp/alerts/printer`. It is invoked with three parameters: *printer_name*, `date`, *file_name*. The *file_name* is the name of a file containing the fault message.

`none`
Do not send messages; any existing alert definition for the printer will be removed. No alert will be sent when the printer faults until a different alert-type (except `quiet`) is used.

shell-command
Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blank spaces embedded in the command, enclose the command in quotes. Notice that the `mail` and `write` values for this option are equivalent to the values `mail` *user-name* and `write` *user-name* respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the `su` command to change to another user ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.

`list`
Display the type of the alert for the printer fault. No change is made to the alert.

When a fault occurs, the printing subsystem displays a message indicating that printing for a specified printer has stopped and the reason for the stoppage. The message also indicates that printing will restart in a few minutes and that you can enter an `enable` command if you want to restart sooner than that.

Following a fault that occurs in the middle of a print job, the job is reprinted from the beginning. An exception to this occurs when you enter a command, such as the one shown below, that changes the page list to be printed.

```
% lp -irequest-id -P ...
```

For a given print request, the presence of multiple reasons for failure indicate multiple attempts at printing.

The LP print service can detect printer faults only through an adequate fast filter and only when the standard interface program or a suitable customized interface program is used. Furthermore, the level of recovery after a fault depends on the capabilities of the filter.

If, instead of a single printer, the keyword `all` is displayed in an alert, the alert applies to all printers.

If the `-W` option is not used to arrange fault alerting for *printer*, the default procedure is to mail one message to the administrator of *printer* per fault. This is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than zero, an alert will be sent at intervals specified by *minutes*.

`-c` *class*
Insert *printer* into the specified *class*. *class* will be created if it does not already exist. This option requires the `-U` *dial-info* or `-v` *device* options.

`-D` *comment*
Save this *comment* for display whenever a user asks for a full description of *printer* (see `lpstat`(1)). The LP print service does not interpret this comment.

`-e` *printer*
Copy the interface program of an existing *printer* to be the interface program for *printer*. (Options `-i` and `-m` must not be specified with this option.)

`-f allow:`*form-list*
`-f deny:`*form-list*
Allow or deny the forms in *form-list* to be printed on *printer*. By default no forms are allowed on a new printer.

For each printer, the LP print service keeps two lists of forms: an "allow-list" of forms that can be used with the printer, and a "deny-list" of forms that cannot be used with the printer. With the `-f allow` option, the forms listed are added to the allow-list and removed from the deny-list. With the `-f deny` option, the forms listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the forms in the list can be used on the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the forms in the deny-list cannot be used with the printer. All forms can be excluded from a printer by specifying `-f deny:all`. All forms can be used on a printer (provided the printer can handle all the characteristics of each form) by specifying `-f allow:all`.

The LP print service uses this information as a set of guidelines for determining where a form can be mounted. Administrators, however, are not restricted from mounting a form on any printer. If mounting a form on a particular printer is in disagreement with the information in the allow-list or deny-list, the administrator is warned but the mount is accepted. Nonetheless, if a user attempts to issue a print or change request for a form and printer combination that is in disagreement with the information, the request is accepted

only if the form is currently mounted on the printer. If the form is later unmounted before the request can print, the request is canceled and the user is notified by mail.

If the administrator tries to specify a form as acceptable for use on a printer that does not have the capabilities needed by the form, the command is rejected.

Notice the other use of `-f`, with the `-M` option, below.

The `-T` option must be invoked first with `lpadmin` to identify the printer type before the `-f` option can be used.

`-F` *fault-recovery*
This option specifies the recovery to be used for any print request that is stopped because of a printer fault, according to the value of *fault-recovery*:

`continue`	Continue printing on the top of the page where printing stopped. This requires a filter to wait for the fault to clear before automatically continuing.
`beginning`	Start printing the request again from the beginning.
`wait`	Disable printing on *printer* and wait for the administrator or a user to enable printing again. During the wait, the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. (See the `-i` option of the `lp` command.) If no change request is made before printing is enabled, printing resumes at the top of the page where stopped, if the filter allows; otherwise, the request is printed from the beginning.

`-h`
Indicate that the device associated with the printer is hardwired. If neither of the mutually exclusive options, `-h` and `-l`, is specified, `-h` is assumed.

`-i` *interface*
Establish a new interface program for *printer*. *interface* is the pathname of the new program. (The `-e` and `-m` options must not be specified with this option.)

`-I` *content-type-list*
Allow *printer* to handle print requests with the content types listed in a *content-type-list*.

The type `simple` is recognized as the default content type for files in the UNIX system. A `simple` type of file is a data stream containing only printable ASCII characters and the following control characters:

Control Char	Octal Value	Meaning
BACKSPACE	10	Move back one char, except at beginning of line
TAB	11	Move to next tab stop
LINEFEED (newline)	12	Move to beginning of next line
FORMFEED	14	Move to beginning of next page
RETURN	15	Move to beginning of current line

To prevent the print service from considering `simple` a valid type for the printer, specify either an explicit value (such as the printer type) in the *content-type-list*, or an empty list. If you do want `simple` included along with other types, you must include `simple` in the *content-type-list*.

In addition to content types defined by the print administrator, the type `PostScript` is recognized and supported by the Solaris print subsystem. This includes filters to support `PostScript` as the printer content type.

The type `any` is recognized as a special content type for files. When declared as the input type for a printer, it signals the print sub-system not to do any filtering on the file before sending it to the printer.

Except for `simple` and `any`, each *content-type* name is determined by the administrator. If the printer type is specified by the `-T` option, then the printer type is implicitly considered to be also a valid content type.

`-l`
Indicate that the device associated with *printer* is a login terminal. The LP scheduler (`lpsched`) disables all login terminals automatically each time it is started. (The `-h` option must not be specified with this option.)

`-m` *model*
Select *model* interface program, provided with the LP print service, for the printer. (Options `-e` and `-i` must not be specified with this option.)

`-M -f` *form-name* [`-a` [`-o filebreak`]] [`-t` *tray-number*]]
Mount the form *form-name* on *printer*. Print requests that need the pre-printed form *form-name* will be printed on *printer*. If more than one printer has the form mounted and

the user has specified `any` (with the `-d` option of the `lp` command) as the printer destination, then the print request will be printed on the one printer that also meets the other needs of the request.

The page length and width, and character and line pitches needed by the form are compared with those allowed for the printer, by checking the capabilities in the `terminfo` database for the type of printer. If the form requires attributes that are not available with the printer, the administrator is warned but the mount is accepted. If the form lists a print wheel as mandatory, but the print wheel mounted on the printer is different, the administrator is also warned but the mount is accepted.

If the `-a` option is given, an alignment pattern is printed, preceded by the same initialization of the physical printer that precedes a normal print request, with one exception: no banner page is printed. Printing is assumed to start at the top of the first page of the form. After the pattern is printed, the administrator can adjust the mounted form in the printer and press return for another alignment pattern (no initialization this time), and can continue printing as many alignment patterns as desired. The administrator can quit the printing of alignment patterns by typing `q`.

If the `-o filebreak` option is given, a formfeed is inserted between each copy of the alignment pattern. By default, the alignment pattern is assumed to correctly fill a form, so no formfeed is added.

If the `-t` *tray-number* option is specified, printer tray *tray-number* will used.

A form is "unmounted" either by mounting a new form in its place or by using the `-f none` option. By default, a new printer has no form mounted.

Notice the other use of `-f` without the `-M` option above.

`-M -S` *print-wheel*
Mount the *print-wheel* on *printer*. Print requests that need the *print-wheel* will be printed on *printer*. If more than one printer has *print-wheel* mounted and the user has specified `any` (with the `-d` option of the `lp` command) as the printer destination, then the print request will be printed on the one printer that also meets the other needs of the request.

If the *print-wheel* is not listed as acceptable for the printer, the administrator is warned but the mount is accepted. If the printer does not take print wheels, the command is rejected.

A print wheel is "unmounted" either by mounting a new print wheel in its place or by using the option `-S none`. By default, a new printer has no print wheel mounted.

Notice the other uses of the `-S` option without the `-M` option described below.

`-n` *ppdfilename*
Specify a PPD file for creating and modifying printer queues. *ppdfilename* is the full path and file name to the PPD file. Used in conjunction with the `-p`, `-d`, `-x`, or `-S` options.

-o *option*
The -o option defines default printer configuration values given to an interface program. The default can be explicitly overwritten for individual requests by the user (see lp(1)), or taken from a preprinted form description (see lpforms(1M) and lp(1)).

There are several options which are predefined by the system. In addition, any number of key-value pairs can be defined. See the section "Predefined Options Used with the -o Option", below.

-P *paper-name*
Specify a paper type list that the printer supports.

-r *class*
Remove *printer* from the specified *class*. If *printer* is the last member of *class*, then *class* will be removed.

-S *list*
Allow either the print wheels or aliases for character sets named in *list* to be used on the printer.

If the printer is a type that takes print wheels, then *list* is a comma or space separated list of print wheel names. These will be the only print wheels considered mountable on the printer. (You can always force a different print wheel to be mounted.) Until the option is used to specify a list, no print wheels will be considered mountable on the printer, and print requests that ask for a particular print wheel with this printer will be rejected.

If the printer is a type that has selectable character sets, then *list* is a list of character set name "mappings" or aliases. Each "mapping" is of the form *known-name=alias* The *known-name* is a character set number preceded by cs (such as cs3 for character set three) or a character set name from the terminfo database entry csnm. See terminfo(4). If this option is not used to specify a list, only the names already known from the terminfo database or numbers with a prefix of cs will be acceptable for the printer. If *list* is the word none, any existing print wheel lists or character set aliases will be removed.

Notice the other uses of the -S with the -M option described above.

The -T option must be invoked first with lpadmin to identify the printer type before the -S option can be used.

-s *system-name*[!*printer-name*]
Make a remote printer (one that must be accessed through another system) accessible to users on your system. *system-name* is the name of the remote system on which the remote printer is located it. *printer-name* is the name used on the remote system for that printer. For example, if you want to access *printer1* on *system1* and you want it called *printer2* on your system:

-p *printer2* -s *system1*!*printer1*

`-T` *printer-type-list*
Identify the printer as being of one or more *printer-type*s. Each *printer-type* is used to extract data from the `terminfo` database; this information is used to initialize the printer before printing each user's request. Some filters might also use a *printer-type* to convert content for the printer. If this option is not used, the default *printer-type* will be `unknown`; no information will be extracted from `terminfo` so each user request will be printed without first initializing the printer. Also, this option must be used if the following are to work: `-o cpi`, `-o lpi`, `-o width`, and `-o length` options of the `lpadmin` and `lp` commands, and the `-S` and `-f` options of the `lpadmin` command.

If the *printer-type-list* contains more than one type, then the *content-type-list* of the `-I` option must either be specified as `simple`, as empty (`-I ""`), or not specified at all.

`-t` *number-of-trays*
Specify the number of trays when creating the printer.

`-u allow:`*login-ID-list*
`-u deny:`*login-ID-list*
Allow or deny the users in *login-ID-list* access to the printer. By default all users are allowed on a new printer. The *login-ID-list* argument can include any or all of the following constructs:

login-ID	a user on any system
system-name`!`*login-ID*	a user on system *system-name*
system-name`!all`	all users on system *system-name*
`all!`*login-ID*	a user on all systems
`all`	all users on all systems

For each printer, the LP print service keeps two lists of users: an "allow-list" of people allowed to use the printer, and a "deny-list" of people denied access to the printer. With the `-u allow` option, the users listed are added to the allow-list and removed from the deny-list. With the `-u deny` option, the users listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the users in the list can use the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list cannot use the printer. All users can be denied access to the printer by specifying `-u deny:all`. All users can use the printer by specifying `-u allow:all`.

The `-U` option allows your print service to access a remote printer. (It does not enable your print service to access a remote printer service.) Specifically, `-U` assigns the "dialing" information *dial-info* to the printer. *dial-info* is used with the `dial` routine to call the printer. Any network connection supported by the Basic Networking Utilities will work. *dial-info* can be either a phone number for a modem connection, or a system name for other kinds of connections. Or, if `-U direct` is given, no dialing will take place, because the

name `direct` is reserved for a printer that is directly connected. If a system name is given, it is used to search for connection details from the file `/etc/uucp/Systems` or related files. The Basic Networking Utilities are required to support this option. By default, `-U direct` is assumed.

`-v` *device*
Associate a *device* with *printer*. *device* is the path name of a file that is writable by `lp`. Notice that the same *device* can be associated with more than one printer.

Removing a Printer Destination

The `-x` *dest* option removes the destination *dest* (a printer or a class), from the LP print service. If *dest* is a printer and is the only member of a class, then the class will be deleted, too. If *dest* is `all`, all printers and classes are removed. If there are no remaining local printers and the scheduler is still running, the scheduler is shut down.

No other *options* are allowed with `-x`.

Setting/Changing the System Default Destination

The `-d` [*dest*] option makes *dest* (an existing printer or class) the new system default destination. If *dest* is not supplied, then there is no system default destination. No other *options* are allowed with `-d`.

Setting an Alert for a Print Wheel

`-S` *print-wheel* [`-A` *alert-type*] [`-W` *minutes*] [`-Q` *requests*] `-T`
The `-S` *print-wheel* option is used with the `-A` *alert-type* option to define an alert to mount the print wheel when there are jobs queued for it. If this command is not used to arrange alerting for a print wheel, no alert will be sent for the print wheel. Notice the other use of `-A`, with the `-p` option, above.

The *alert-types* are:

`mail`	Send the alert message using the `mail` command to the administrator.
`write`	Write the message, using the `write` command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.
`quiet`	Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the *print-wheel* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests reaches the threshold specified by the `-Q` option.
`none`	Do not send messages until the `-A` option is given again with a different *alert-type* (other than `quiet`).
shell-command	Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blanks embedded in the command, enclose the command in quotes. Notice that the `mail` and `write` values for this option are equivalent to the values `mail` *user-name* and `write` *user-name* respectively, where

user-name is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the `su` command to change to another user ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.

`list` Display the type of the alert for the print wheel on standard output. No change is made to the alert.

The message sent appears as follows:

```
The print wheel print-wheel needs to be mounted
on the printer(s):
printer(integer1requests) integer2 print requests
await this print wheel.
```

The printers listed are those that the administrator had earlier specified were candidates for this print wheel. The number *integer1* listed next to each printer is the number of requests eligible for the printer. The number *integer2* shown after the printer list is the total number of requests awaiting the print wheel. It will be less than the sum of the other numbers if some requests can be handled by more than one printer.

If the *print-wheel* is `all`, the alerting defined in this command applies to all print wheels already defined to have an alert.

If the `-W` option is not given, the default procedure is that only one message will be sent per need to mount the print wheel. Not specifying the `-W` option is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than zero, an alert will be sent at intervals specified by *minutes*.

If the `-Q` option is also given, the alert will be sent when a certain number (specified by the argument *requests*) of print requests that need the print wheel are waiting. If the `-Q` option is not given, or *requests* is 1 or `any` (which are both the default), a message is sent as soon as anyone submits a print request for the print wheel when it is not mounted.

Predefined Options Used with the `-o` Option

A number of options, described below, are predefined for use with `-o`. These options are used for adjusting printer capabilities, adjusting printer port characteristics, configuring network printers, and controlling the use of banner. The `-o` also supports an arbitrary *keyword=value* format, which is referred to below as an undefined option.

Adjusting Printer Capabilities

The `length`, `width`, `cpi`, and `lpi` parameters can be used in conjunction with the `-o` option to adjust printer capabilities. The format of the parameters and their values is as follows:

```
length=scaled-decimal-number
width=scaled-decimal-number
cpi=scaled-decimal-number
lpi=scaled-decimal-number
```

The term *scaled-decimal-number* refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a "trailing" letter attached to the number. Three types of *scaled-decimal-numbers* can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing `c`); numbers that show sizes in inches (marked with a trailing `i`); and numbers that show sizes in units appropriate to use (without a trailing letter), that is, lines, characters, lines per inch, or characters per inch.

The option values must agree with the capabilities of the type of physical printer, as defined in the terminfo database for the printer type. If they do not, the command is rejected.

The defaults are defined in the `terminfo` entry for the specified printer type. The defaults can be reset by:

```
lpadmin -p printername -o length=
lpadmin -p printername -o width=
lpadmin -p printername -o cpi=
lpadmin -p printername -o lpi=
```

Adjusting Printer Port Characteristics

You use the `stty` keyword in conjunction with the o option to adjust printer port characteristics. The general form of the `stty` portion of the command is:

```
stty="'stty-option-list'"
```

The *stty-option-list* is not checked for allowed values, but is passed directly to the `stty` program by the standard interface program. Any error messages produced by `stty` when a request is processed (by the standard interface program) are mailed to the user submitting the request.

The default for `stty` is:

```
stty="'9600 cs8 -cstopb -parenb ixon
     -ixany opost -olcuc onlcr
     -ocrnl -onocr
     -onlret -ofill nl0 cr0 tab0 bs0 vt0 ff0'"
```

The default can be reset by:

```
lpadmin -p printername -o stty=
```

Configuring Network Printers

The `dest`, `protocol`, `bsdctrl`, and `timeout` parameters are used in conjunction with the `-o` option to configure network printers. The format of these keywords and their assigned values is as follows:

```
dest=string protocol=string bsdctrl=string \
     timeout=non-negative-integer-seconds
```

These four options are provided to support network printing. Each option is passed directly to the interface program; any checking for allowed values is done there.

The value of `dest` is the name of the destination for the network printer; the semantics for value `dest` are dependent on the printer and the configuration. There is no default.

The value of option `protocol` sets the over-the-wire protocol to the printer. The default for option `protocol` is `bsd`. The value of option `bsdctrl` sets the print order of control and data files (BSD protocol only); the default for this option is `control file first`. The value of option `timeout` sets the seed value for backoff time when the printer is busy. The default value for the `timeout` option is `10` seconds. The defaults can be reset by:

```
lpadmin -p printername -o protocol=
lpadmin -p printername -o bsdctrl=
lpadmin -p printername -o timeout=
```

Controlling the Use of the Banner Page

Use the following commands to control the use of the banner page:

```
lpadmin -p printer -o nobanner
lpadmin -p printer -o banner
lpadmin -p printer -o banner=always
lpadmin -p printer -o banner=never
lpadmin -p printer -o banner=optional
```

The first and fifth commands (`-o nobanner` and `-o banner=optional`) are equivalent. The default is to print the banner page, unless a user specifies `-o nobanner` on an `lp` command line.

The second and third commands (`-o banner` and `-o banner=always`) are equivalent. Both cause a banner page to be printed always, even if a user specifies `lp -o nobanner`. The root user can override this command.

The fourth command (`-o banner=never`) causes a banner page never to be printed, even if a user specifies `lp -o banner`. The root user can override this command.

Undefined Options

The `-o` option supports the use of arbitrary, user-defined options with the following format:

key=*value*

Each *key*=*value* is passed directly to the interface program. Any checking for allowed values is done in the interface program.

Any default values for a given *key*=*value* option are defined in the interface program. If a default is provided, it can be reset by typing the key without any value:

```
lpadmin -p printername -o key=
```

使用例

In the following examples, *prtr* can be any name up to 14 characters and can be the same name as the `ping`(1M) name.

例 1 Configuring an HP Postscript Printer with a Jet Direct Network Interface

The following example configures an HP postscript printer with a jet direct network interface:

例 1 Configuring an HP Postscript Printer with a Jet Direct Network Interface *(続き)*

```
example# lpadmin -p prtr -v /dev/null -m netstandard \
          -o dest=ping_name_of_prtr:9100 -o protocol=tcp -T PS -I \
           postscript
example# enable prtr
example# accept prtr
```

例 2 Configuring a Standard Postscript Network Printer

The following example configures a standard postscript network printer:

```
example# lpadmin -p prtr -v /dev/null -m netstandard \
         -o dest=ping_name_of_prtr -T PS -I postscript
example# enable prtr
example# accept prtr
```

終了ステータス

The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

ファイル

`/var/spool/lp/*`

`/etc/lp`

`/etc/lp/alerts/printer` fault handler for `lpadmin`.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

関連項目

`enable`(1), `lp`(1), `lpstat`(1), `mail`(1), `stty`(1), accept(1M), `lpforms`(1M), lpsched(1M), `lpsystem`(1M), `ping`(1M), `dial`(3NSL), `terminfo`(4), `attributes`(5)

『Solaris のシステム管理 (基本編)』

名前　lpmove – 印刷要求の移動

形式

```
lpmove [request-ID] destination
lpmove destination1 destination2
```

機能説明

lpmove コマンドは、 lp(1) または lpr(1B) によって待ち行列に入れられた印刷要求を宛先間で移動します。

第 1 の形式では、指定の印刷要求 (*request–ID*) を、指定の宛先 (*destination*) に移します。

第 2 の形式では、ある宛先 (*destination1*) から他の宛先 (*destination2*) にすべての印刷要求を移します。 この形式では lpmove は、新たに発生する *destination1* 宛の印刷要求も受け付けなくなります。

lpmove が個々の要求または待ち行列全体を移動できるのは、ローカルプリンタ間またはリモートプリンタ間のどちらか一方であり、ローカルプリンタとリモートプリンタ間では移動できません。また、移動できる要求はまだサーバーに転送されていない要求だけです。

要求を移動する際、lpmove は、印刷要求の移動先の受け入れ状態をチェックしません (accept(1M) を参照)。なお、オプション (たとえば、内容タイプ、特定の用紙の要求) つきの要求に関しては、新たな宛先がそのオプションを扱えなければ、lpmove は移動を行いません。

オペランド　以下のオペランドを指定できます。

request–ID　移動対象になる特定の印刷要求。lpstat が示す印刷要求に対応する識別子を *request–ID* に指定します。 lpstat(1) のマニュアルページを参照してください。

destination　lpmove が指定された印刷要求を移動するプリンタ名またはプリンタのクラス名 (lpadmin(1M) を参照)。名前、POSIX スタイル名 (*server*:*destination*)

destination1　lpmove がすべての印刷要求を移動する移動元の名前。名前、POSIX スタイル名 (*server*:*destination*)

destination2　lpmove がすべての印刷要求を移動する移動先の名前。名前、POSIX スタイル名 (*server*:*destination*)

名前や FNS 名の命名規約については printers.conf(4) を、POSIX については standards(5) を参照してください。

終了ステータス　以下の終了ステータスが返されます。

0　正常終了

non-zero　エラーが発生した

ファイル　　`/var/spool/print/*`　　LP 印刷待ち行列

属性　　次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWpcu

関連項目　　`lp`(1), `lpr`(1B), `lpstat`(1), accept(1M), lpadmin(1M), lpsched(1M), `printers.conf`(4), `attributes`(5), `standards`(5)

名前　lpsched – LP 印刷サービスの起動

形式　`lpsched [-f num_filters] [-n num_notifiers] [-p fd_limit] [-r reserved_fds]`

機能説明　`lpsched` コマンドは LP 印刷サービスを起動または再起動します。

`lpshut` コマンドは LP 印刷サービスを停止します。 `lpsched` でプリンタを再起動すれば、`lpshut` で印刷を中止した要求に関しては (始めから) 再印刷できます (lpshut(1M) を参照)。

svcadm(1M) を使用して LP 印刷サービスを起動および停止することを推奨します。詳細は「注意事項」を参照してください。

オプション　以下のオプションを指定できます。

`-f` *num_filters*　印刷サーバー上で実行できる、並行するスローフィルタ数を指定します。何も指定しない場合は、デフォルトとして `1` が使用されます。サーバーの構成によっては値を `1` にすると、サーバーにジョブの待ち行列が存在していても、プリンタがアイドル状態のままになります。

`-n` *num_notifiers*　印刷サーバー上で実行できる、並行する通知プロセス数を指定します。何も指定しない場合は、デフォルトとして `1` が使用されます。

`-p` *fd_limit*　`lpsched` プロセスで使用するファイル記述子のリソース制限値を指定します。 何も指定しない場合は、デフォルトとして `4096` が使用されます。非常に大型でアクティブな印刷サーバー上では、この値を増やす必要はありません。

`-r` *reserved_fds*　大量ロード下で、スケジューラが内部通信に予約する ファイル記述子の数を指定します。何も指定しない場合は、デフォルトとして `2` が使用されます。高速ロード下での問題を障害追跡する場合に、 その解析を指示しない限りは、この値を修正する必要はありません。

終了ステータス　以下の終了ステータスが返されます。

`0`　正常終了

`0` 以外　エラーが発生した

ファイル　`/var/spool/lp/*`　LP 印刷待ち行列

属性　次の属性については `attributes(5)` のマニュアルページを参照してください。

属性タイプ	属性値

使用条件	SUNWpsu

関連項目

lp(1), svcs(1), lpstat(1), lpadmin(1M), lpmove(1M), lpshut(1M), svcadm(1M), attributes(5), smf(5)

『Solaris のシステム管理 (基本編)』

注意事項

lpsched サービスはサービス管理機能 smf(5) により次のサービス識別子の下で管理されます。

```
svc:/application/print/server
```

有効化、無効化、再起動要求など、このサービスに関する管理操作は、svcadm(1M) を使用して実行できます。サービスの状態は svcs(1) コマンドを使用して照会できます。

名前　lpshut – LP 印刷サービスの停止

形式　`lpshut`

機能説明　`lpshut` コマンドは LP 印刷サービスを停止します。

`lpshut` が実行されると、プリンタはただちに印刷を中止します。プリンタを起動または再起動するには、lpsched(1M) を使用します。

終了ステータス　以下の終了ステータスが返されます。

`0`　正常終了

`0` 以外　エラーが発生した

ファイル　`/var/spool/lp/*`　LP 印刷待ち行列

属性　次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWpsu

関連項目　`lp`(1), `lpstat`(1), lpadmin(1M), lpmove(1M), lpsched(1M), `attributes`(5)

『Solaris のシステム管理 (基本編)』

名前　lu – Live Upgrade 機能の FMLI ベースインタフェース

形式　`/usr/sbin/lu`

機能説明　現在では、Sun は `lu` コマンドの使用を推奨していません。`lu` コマンドは、キャラクタユーザーインタフェース (CUI) を表示します。CUI の基礎的なコマンドシーケンス (通常は `lucreate`、`luupgrade`、および `luactivate`) の使い方は、単純です。後述の「関連項目」の項に、Solaris Live Upgrade の完全なコマンドセットが記載されています。

`lu` プログラムは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

`lu` プログラムは、FMLI (Forms and Menu Language Interpreter) ベースのユーザーインタフェースです (FMLI の説明は、`fmli`(1) のマニュアルページを参照してください)。lu プログラムを使用すると、ブート環境 (BE) の作成やアップグレード、また、BE に関する管理タスクを行うことができます。ただし、`lu` プログラムでできるのは、Live Upgrade コマンド行ユーティリティの一部の機能です。

`lu` コマンドを使用する場合は、下記のことを知っておく必要があります。

- `lu` は推奨されないインタフェースです。将来はほかのコマンドに置き換えられるので、重要な機能については使用しないでください。
- 新しいすべての Live Upgrade 機能は、現在、Live Upgrade コマンド行ユーティリティに実装されています。`lu` コマンドに新しい機能が追加される予定はありません。
- `lu` コマンドは国際化されていません。将来のリリースでも国際化される予定はありません。

`lu` は学習または実験のためだけに使用してください。製品用に使用したり、または Live Upgrade の全機能を使用したりする場合は、Live Upgrade コマンド行ユーティリティを使用してください。

`lu` コマンドを実行するには、root 権限が必要です。

`lu` コマンドには引数はありません。`lu` を呼び出すと、次のオプションが表示されます。

Activate　ブート環境をアクティブにします。このオプションを使用すると、システムは次回のリブート時に、指定した BE から起動されます。このオプションは、luactivate(1m) コマンド行ユーティリティと同等です。

Cancel　コピージョブを取り消します。Live Upgrade を使用すると、コピーやアップグレード、フラッシュ機能 (下記の説明を参照) をスケジュールし、後で実行することができます。取り消し機能により、スケジュールしたジョブを取り消すことができます。このオプションは、lucancel(1M) コマンド行ユーティリティと同等です。

Compare　BE の内容を比較します。2 つの BE の詳細な比較情報が得られます。このオプションは、lucompare(1M) コマンド行ユーティリティと同等です。

Copy　コピーを開始またはスケジュールします。この機能は、ある BE の内容を他の BE にコピーします。このオプションは、lumake(1M) コマンド行ユーティリティと同等です。Live Upgrade 操作は、同時に 1 つしかスケジュールできません。

Create　ブート環境を作成します。このオプションは、lucreate(1M) コマンド行ユーティリティの一部の機能を実行します。

Current　現在のブート環境の名前を表示します。このオプションは、lucurr(1M) コマンド行ユーティリティと同等です。

Delete　ブート環境を削除します。このオプションは、ludelete(1m) コマンド行ユーティリティと同等です。

List　ブート環境のファイルシステムを一覧表示します。このオプションは、lufslist(1m) コマンド行ユーティリティと同等です。

Rename　ブート環境の名前を変更します。このオプションは、lurename(1M) コマンド行ユーティリティと同等です。

Status　すべてのブート環境の状態を一覧表示します。このオプションは、lustatus(1M) コマンド行ユーティリティと同等です。

Upgrade　ブート環境をアップグレードします。または、非アクティブな BE 上の OS をアップグレードします。このオプションを使用すると、新しいオペレーティングシステムにアップグレードしたり、指定した BE に新しいパッケージやパッチをインストールすることができます。このオプションは、luupgrade(1M) コマンド行ユーティリティの一部の機能を実行します。CD を 2 枚以上使用してアップグレードを行う場合は、`-i` オプションを使用して `luupgrade` を実行する必要があります。

Flash　ブート環境をフラッシュします。このオプションを使用すると、フラッシュアーカイブから BE 上にオペレーティングシステムをインストールできます。luupgrade(1M) でも同じ操作が行えます。

Help　ヘルプ情報を表示します。この他にも、多数のオプション用のヘルプ画面を利用できます。

Exit　`lu` を終了します。

ナビゲーション　`lu` の画面を切り替えるには、矢印キーとファンクションキー (通常は、Sun デスクトップシステムのキーボード上の F2 から F9 キー) を使用します。使用できるキーの機能が `lu` 画面の下部に表示されます。Ctrl-F と数字キーを組み合わせると、ファンクションキーの代わりに使用できます。たとえば、F2 キーの代わりに、Ctrl-F と数字キー 2 を押します。

特定のオプション画面で Esc キーを押すと、そのオプションのヘルプ画面が表示されます。

表示の問題

tip 回線などを介してリモートから FMLI インタフェースを使用する場合は、TERM 環境変数を VT220 に設定しなければならない場合があります。CDE 環境で FMLI インタフェースを使用する場合は、TERM 変数の値として、xterm ではなく dtterm を使用します。

lu コマンドは、シングルバイト環境でのみ使用できます。

共通の機能

上記のほとんどのオプションでは、次の機能が利用できます。これらの機能は、画面下部に表示されるファンクションキーを使って実行できます。

Choice　入力するフィールドがあるときに使用できます。Choice ファンクションキーを押すと、ポップアップ画面に選択肢が表示されます。たとえば、BE のコピーやアップグレードを伴うオプションでは、使用可能な BE のリストが表示されます。そのあと、矢印キーとファンクションキーを使ってポップアップから必要な BE を選択できます。この機能には、無効な値を選択することがないという利点があります。たとえば、BE のコピーやアップグレードを伴うオプションで、コピーやアップグレード操作に利用できない BE を選択することはありません。なぜなら、アップグレード中の BE は、選択できないようになっているからです。

Cancel　操作を取り消します。

Save　操作を続けます。

その他の機能

上記の「Create」オプションでは、次の機能を使用できます。

Split　ファイルシステムを分割します。たとえば、/ ファイルシステムを /、/usr、および /var に分割できます。ファイルシステムを分割する場合には、分割した個々のファイルシステムをマウントするディスクスライスが必要です。使用できるディスクスライスがない場合、lu は format(1M) ユーティリティを呼び出すので、partition オプションを使って新しいディスクスライスを作成します。

Merge　1 つまたは複数のファイルシステムを、それぞれの親ファイルシステムと結合します。たとえば、/、/usr、および /var という別々のファイルシステムが存在するソース BE をターゲット BE 上で / の下にこれらのファイルシステムをマージできます。

ファイル

/etc/lutab　システム上にある BE のリスト

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

『Solaris インストールガイド』

警告

`lu` コマンドは推奨されないインタフェースです。「機能説明」を参照してください。

名前　　luactivate – ブート環境のアクティブ化

形式　　`/usr/sbin/luactivate [-l` *error_log*`] [-o` *outfile*`] [-s]`
`[`*BE_name*`] [-X]`

機能説明　　`luactivate` コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

引数を指定せずに `luactivate` コマンドを実行すると、システムを次回リブート時にアクティブになるブート環境 (BE) の名前が表示されます。引数として BE を指定すると、その BE がアクティブになります。

`luactivate` は、BE のルートパーティションをブート可能にすることによって、その BE をアクティブにします。x86 マシンでは、`luactivate` の実行後に、いくつかの手順を実行しなければならないことがあります。その場合、`luactivate` は必要となる手順を表示します。

BE をアクティブにするためには、BE が次の条件を満たしていなければなりません。

- lustatus(1M) で表示される BE の状態が complete でなければならない
- BE が現在の BE でない場合、その BE 上に lumount(1M) や mount(1M) でマウントされたパーティションがあってはならない
- アクティブにしたい BE が lucompare(1M) 操作の対象であってはならない

`luactivate` は、指定された BE をアクティブにした後に、次回のリブート時に問題が発生した場合にとるべきフォールバック手順を表示します。それらの手順を記録しておき、必要であればその手順に従います。

注 – 新しい BE をブートする前に、`luactivate` を実行して、その BE がアクティブであることを指定する必要があります。`luactivate` は、以下に説明するように、BE が正しく動作されていることを保証する作業を数多く実行します。このコマンドを実行するまで、BE でブート可能にならない場合があります。

`luactivate` コマンドは、次の作業を実行します。

- 新たに作成した BE を初めてブートすると、Live Upgrade ソフトウェアは、その BE と最後にアクティブだった BE の同期をとります。(アクティブだった BE は、新たに作成した BE のソースであるとは限りません)。ここでの「同期をとる」とは、特定のシステムファイルおよびディレクトリを、最後にアクティブだった BE から、ブートする BE にコピーすることを指します (`synclist`(4) を参照)。最初のブート以後は、`-s` オプションを指定しない限り、同期はとられません (詳細は `-s` オプションの説明を参照)。
- 同期をとったファイル間に衝突が検出された場合、`luactivate` は警告を発しますが、衝突のあったファイルの同期をとることはしません。衝突があった場合でも、アクティブ化が成功することがあります。衝突は、ある BE を別のオペ

レーティングシステム用にアップグレードしたり、BE 上のシステムファイル (/etc/passwd など) を変更した場合に発生する可能性があります。

- luactivate は、更新の問題が発生したかどうかをチェックします。たとえば、オペレーティングシステムが正しく動作するために必要なパッケージが失われているかどうかなどです。このパッケージ確認は、大域ゾーンおよび BE 内部のすべての非大域ゾーンに対して行われます。このコマンドは、警告を発したり、BE が完全でない場合は、アクティブ化を拒否したりすることもあります。
- luactivate は、ブートストラッププログラムを更新する必要があるかどうかを判断して、必要であれば、ブートストラッププログラムを更新します。オペレーティングシステムのリリース間でブートストラッププログラムが変更されている場合、更新されていないブートストラッププログラムは、更新されている BE を「ブートできない」と判断する可能性があります。installboot(1M) を参照してください。
- luactivate は、Solaris x86ディスク上の root パーティション ID を変更することによって、複数の BE が単一のディスク上に存在できるようにします。この構成では、luactivate を実行しない場合、BE のブートは失敗します。fmthard(1M) と dkio(7I) を参照してください。

luactivate コマンドを実行するには、root 権限が必要です。

オプション

luactivate コマンドには、次のオプションを指定できます。

-l *error_log*
: エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

-o *outfile*
: すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

-s
: 指定した BE に対する次のブートが、その BE に対する最初のブートでない場合でも、同期をとります (「機能説明」の項を参照)。このオプションは慎重に使用してください。アクティブだった BE で、このコマンドの実行者自身が管理または関知しない変更が行われている可能性があります。

  -s を使用する場合、アクティブだった BE にインストールされている Solaris より古いバージョンの Solaris でブートするときには、特に注意が必要です。たとえば、アクティブだった BE には Solaris 9 が含まれていて、これを Solaris 2.6 を含む BE でブートする場合を考えてみましょう。-s オプションを使用して強制的に同期をとると、Solaris 2.6 を含む BE のファイルは、Solaris 9 と互換性はあるが、Solaris 2.6 上では動作しない可能性のあるファイルと同期がとられることになります。

-X
: XML 出力を有効にします。XML の特性は DTD (/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

オペランド　*BE_name*　アクティブにする BE の名前を指定します。

終了ステータス　次の終了ステータスが返されます。

`0`　正常終了。

`>0`　エラーが発生した。

ファイル

`/etc/lutab`	システム上にある BE のリスト
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade の DTD (`-X` オプションを参照)

属性　次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目　lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5), `zones`(5)

名前

lucancel – スケジュールされた Live Upgrade 機能でのコピー/作成操作の取り消し

形式

```
/usr/sbin/lucancel [-l error_log] [-o outfile] [-X]
```

機能説明

`lucancel` コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

`lucancel` コマンドは、FMLI ベースのインタフェースである lu(1M) を使ってスケジュールされた、ブート環境 (BE) の作成またはアップグレードを取り消します。あるいは、lumake(1M) を使ってスケジュールされた BE の再生成を取り消します。Sun は `lu` コマンドの使用を推奨していません。

`lucancel` は、アクティブな (つまり、作成中や再生成中の) ジョブは取り消しません。

`lucancel` コマンドを実行するには、root 権限が必要です。

オプション

`lucancel` コマンドには、次のオプションを指定できます。

`-l` *error_log*　エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-o` *outfile*　すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

`-X`　XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

終了ステータス

次の終了ステータスが返されます。

`0`　正常終了。

`>0`　エラーが発生した。

ファイル

`/etc/lutab`　システム上にある BE のリスト

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

名前　lucompare – 2 つのブート環境の比較

形式

```
/usr/sbin/lucompare [-i infile | -t] [-o outfile] BE_name
     [-X]

/usr/sbin/lucompare [-C file [-o outfile]] [-X]
```

機能説明

lucompare コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、live_upgrade(5) のマニュアルページを参照してください。

lucompare コマンドは、現在のブート環境 (BE) の内容と他の BE の内容を比較します。lucompare に -C オプションを指定すると、ファイルの統計情報が比較されます。これによって、指定した時刻 (BE の作成時刻など) 以降に BE 上のどのファイルが変更されたかを知ることができます。指定する BE は非アクティブで、その状態が complete でなければなりません。BE の状態は、lustatus(1M) コマンドで知ることができます。また BE では、スケジュールされたコピージョブがあってはなりません (これも lustatus(1M) で検査できる)。指定する BE には、lumount(1M) や mount(1M) でマウントされたパーティションがあってはなりません。

lucompare は、指定された BE に定義されているファイルシステムごとに、そのすべてのファイルを、現在の BE にある同じパス名を持つファイルと比較します。アクティブな BE にはあるが、指定した BE にはないファイルと、その逆の場合のファイルが報告されます。比較するファイルのリストを指定するオプションも利用できます。

現在の BE とターゲット BE の完全な比較を行う代わりに -C オプションを指定すると、指定した BE のファイルと、任意のファイルに記録されているファイルリストが比較されます。BE が作成されたときに、lucreate(1M) は /etc/lu/compare の下に :<*BE_name*> というファイルを作成します。-C オプションを指定すると、指定した BE のファイルと /etc/lu/compare にあるこのスナップショットを比較できます。あるいは、-o オプションですでに作成してあるファイルを比較することもできます。BE と、/etc/lu/compare にあるそれ自身のスナップショットを比較することにより、BE の作成時以降に変更されたファイルを判別できます。

デフォルトでは、lucompare の出力は標準出力に書き込まれます。-C オプションを指定するとき、-o オプションで出力ファイルを指定する必要があります。lucompare の出力には、アクセス権、所有者、グループ、またはチェックサムが異なるファイルのリストと、その相違理由が含まれます。出力形式は次のとおりです。

```
> active BE
< BE_name
reason
> file_name:owner:group:number_of_links:mode:type:  size
or  major_minor number:checksum
< file_name:owner:group:number_of_links:mode:type:  size
or  major_minor number:checksum
```

上記の各フィールドは、ファイルの stat(2) 構造体から得られたものです。

type フィールドは次のいずれかです。

SYMLINK　シンボリックリンク

FIFO　FIFO ファイル

CHRSPC　文字型特殊ファイル

BLKSPC　ブロック型特殊ファイル

DIR　ディレクトリ

REGFIL　通常ファイル

UNKNOW　未知のファイルタイプ

lucompare は、指定した BE 上のファイルとアクティブな BE 上の対応するファイルが上記のすべてのフィールドで一致する場合にのみ、チェックサムを計算します。そして、チェックサムが一致しない場合、lucompare は比較したそれぞれのファイルのエントリにそのチェックサムを追加します。

lucompare コマンドを実行するには、root 権限が必要です。

オプション

lucompare コマンドには、次のオプションを指定できます。

-C *file*　BE のファイル統計情報と、*file* に記録されている統計情報を比較します。*file* は、BE の作成時に作成されたスナップショット (/etc/lu/compare/:<*BE_name*>) か、-o オプションですでに作成されているファイルです。このオプションを使用するときには、-o オプションを指定する必要があります。

-i *infile*　*infile* にリストされているファイルを比較します。これらのファイルは、絶対パス名で指定しなければなりません。ファイル内のエントリがディレクトリの場合には、比較はそのディレクトリにについて再帰的に行なわれます。-t オプションを同時に指定することはできません。

-o *outfile*　相違点を *outfile* に出力します。-C オプションを使用する場合は、このオプションも指定する必要があります。

-t　非バイナリファイルだけを比較します。この処理は、ツリー上の各ファイルに対して file(1) コマンドを実行し、テキストファイルだけを比較することによって行なわれます。-i オプションを同時に指定することはできません。

-X　XML 出力を有効にします。XML の特性は DTD (/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

オペランド

BE_name アクティブな BE の比較対象となる BE の名前を指定します。他の Live Upgrade 操作に使われている BE や、lumount(1M) または mount(1M) でパーティションがすでにマウントされている BE を指定することはできません。

使用例

例 1 作成時と現時点の相違点を検査する

次のコマンドは、BE `s8u5` の作成時と現時点における相違点を一覧表示します。

```
# lucompare -C /etc/lu/compare/:s8u5 -o /var/tmp/compare.out s8u5
```

`/etc/lu/compare/:s8u5` は、BE の作成時に `lucreate` が作成したファイルです。相違点の一覧は、`/var/tmp/compare.out` に書き込まれます。

終了ステータス

次の終了ステータスが返されます。

`0` 正常終了。

`>0` エラーが発生した。

ファイル

`/etc/lutab`	システム上にある BE のリスト
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade の DTD (`-X` オプションを参照)

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

注意事項

`lucompare` コマンドが BE 間の相違点を修正することはありません。

名前

lucreate – 新しいブート環境の作成

形式

```
/usr/sbin/lucreate  [-A BE_description] [-c BE_name]
 [-C ( boot_device | - )] -n BE_name [-f exclude_list_file]
 [-I] [-l error_log] [-o outfile] [-s ( - | source_BE_name )]
 [ [-M slice_list_file [-M]...] [-m mount_point:device
 [,volume]:fs_options [-m...]]] [-x exclude [-x]...]
 [-X] [-y include [-y]...] [-Y include_list_file] [-z filter_list]
```

機能説明

lucreate コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明と関連用語については、live_upgrade(5) のマニュアルページを参照してください。

lucreate コマンドではコマンド行オプションを指定することにより、次の操作が行えます。

- 現在のブート環境 (BE) から新しい BE を作成する
- 現在の BE 以外の BE から新しい BE を作成する
- BE 上のファイルシステムを新しい BE 上で分割または結合する。たとえば、/var と /opt を /の下に結合したり、これらのディレクトリを分割して異なるディスクスライスの下にマウントしたりすることができます。
- BE 上に、ファイルシステムを作成するが、ファイルは作成しないでおく。

これらの操作は、lucreate コマンド行オプションだけを使って行うこともできますが、-m および -M オプション (後述の説明を参照) を省略して FMLI ベースのインタフェースを自動的に呼び出し、curses ベースの画面を使って Live Upgrade 管理を行うこともできます。FMLI ベースのインタフェースは、lucreate でサポートされる Live Upgrade 機能をすべてサポートするわけではないことに注意してください。また、Sun は FMLI ベースのインタフェースの開発を継続していくことを保証しているわけではありません。

BE の作成には、BE のすべてのマウントポイントに対してディスクまたはデバイススライスの選択も含まれます。スライスには物理ディスク、または Solaris ボリュームマネージャのボリュームのような論理デバイスを選択できます。BE のマウントポイントは、FMLI ベースの構成画面で SPLIT または MERGE 機能を使って変更することもできます。

BE が適切に作成されると、lustatus(1M) を使って BE の状態を表示したり、lufslist(1m) を使って BE のファイルシステムを表示したりすることができます。さらに、luupgrade(1M) を使ってその BE の OS をアップグレードしたり、luactivate(1m) を使って BE をアクティブにしたりすることができます。BE をアクティブにすると、システムは次回のブート時にその BE からブートします。

注 – 新しい BE をブートする前に、luactivate を実行して、その BE がアクティブであることを指定する必要があります。luactivate は、BE が正しく動作しているこ

とを保証する作業を数多く実行します。このコマンドを実行するまで、BE でブート可能にならない場合があります。このコマンドで実行できる操作の一覧は、luactivate(1m) を参照してください。

lucreate コマンドは、OS を含むファイルシステム (/、./usr、/var、/opt など) と、OS を含まないファイルシステム (/export、/home などのユーザー定義ファイルシステム) を区別します。OS を含むファイルシステムを、ソース BE と新しい BE の間で共有することはできません。これらのファイルシステムは、常にソース BE からターゲット BE にコピーされます。これに対して、ユーザー定義のファイルシステムはデフォルトで共有されます。Live Upgrade では、OS を含むファイルシステムを共有不能 (または重要な) ファイルシステム、その他のファイルシステムを共有可能ファイルシステムと呼びます。ソース BE の vfstab にリストされている共有不能ファイルシステムは、新しい BE にコピーされます。共有可能ファイルシステムは、宛先スライスを指定した場合にのみコピーされます。宛先スライスを指定しない場合は、コピーされません。

lucreate コマンドは、Solaris ボリュームマネージャ機能のいくつかのサブセットをサポートします。たとえば、lucreate コマンドに -m オプションを使用すると、下記のことが可能です。

- ミラーを作成する
- ミラーから既存の Solaris ボリュームマネージャ連結を切り離す。同様に、ミラーに既存の Solaris ボリュームマネージャの連結を接続する。これらは、Solaris ボリュームマネージャまたは lucreate を使って作成されたミラーである。
- 単一スライス連結を作成し、それに単一のディスクスライスを接続する。
- 単一スライス連結から単一のディスクスライスを切り離す。
- 単一のミラーに複数の単一スライス連結を接続する。lucreate では、Solaris ボリュームマネージャで許可されている個数の連結を接続できる。

lucreate では、単一の連結に複数のディスクスライスまたは記憶装置を接続することはできません。同様に、単一の連結から複数のスライスまたは装置を切り離すことはできません。

ブート環境に Solaris ボリュームマネージャのボリュームを使用する場合は、ボリュームの操作には Solaris ボリュームマネージャのコマンドよりも lucreate コマンドが推奨されます。Solaris ボリュームマネージャソフトウェアはブート環境を認識しません。一方、lucreate コマンドには、たとえば Solaris ボリュームマネージャのボリュームを上書きしたり削除したりしてブート環境を破壊することを避ける検査機能が含まれています。

すでに Solaris ボリュームマネージャソフトウェアを使用して複雑な Solaris ボリュームマネージャのボリューム (たとえば、RAID-5 ボリューム) を作成した場合は、Live Upgrade がその使用をサポートします。ただし、これらの複雑なオブジェクトを作成したり操作したりするためには、Solaris ボリュームマネージャソフトウェアを使用する必要があります。上で述べたとおり、lucreate コマンドではなく Solaris ボ

リュームマネージャを使用すると、ブート環境を破壊する危険が伴います。それでも Solaris ボリュームマネージャソフトウェアを使用するという場合は、lufslist(1m) を使用して、ブート環境に使用されているデバイスを特定してください。

次に説明するように、-s オプションを特に使用する場合を除き、新しい BE の作成にはソース BE が必要です。デフォルトでは、ソース BE は現在の BE です。-s オプションを使えば、現在の BE 以外の BE を指定できます。

新しく BE を作成する場合は、lucreate を使用するとソース BE からファイルを除外したり、あるいは含めたりすることができます。この動作は、次で説明する -f、-x、-y、-Y、および -z オプションを使用して行います。これらのオプションの組み合わせについては、次の「オプション」を参照してください。

デフォルトでは、ソース BE 上のすべてのスワップパーティションは、ソース BE とターゲット BE 間で共有されます。-m オプションを使えば (下記を参照)、ソース BE 上のスワップパーティションの追加セットまたは新しいセットをターゲット BE と共有できます。

lucreate コマンド使用すると、BE に説明を追加できます。説明の指定は任意で、書式や長さに制限はありません。たとえば、テキスト文字列でもバイナリデータでもかまいません。BE の作成後は、ludesc(1M) ユーティリティーを使用して BE の説明を変更できます。

lucreate コマンドを実行するには、root 権限が必要です。

オプション

lucreate コマンドには、次のオプションを指定できます。BE 名は、30 文字を超えてはならず、使用できる文字は英数字とその他の ASCII 文字 (UNIX シェルにとって特殊な意味を持つ文字は除く) だけです。これについては、sh(1) の「クォート」セクションを参照してください。また、BE 名に使用できるのは 8 ビットで表現できるシングルバイトの文字だけです。空白文字を含めることはできません。

lucreate コマンド行で -m オプションまたは -M オプション (下記を参照) を省略すると、FMLI ベースのインタフェースが呼び出され、このインタフェースを介して BE 用のディスクスライスまたはデバイススライスを指定できます。

-A *BE_description*
BE に、説明 (*BE_description*) を追加します。*BE_description* には、テキスト文字列および、UNIX コマンド行に入力可能な任意の文字を使用できます。BE の説明に関する詳細は、ludesc(1M) を参照してください。

-c *BE_name*
現在の BE に *BE_name* という名前を割り当てます。このオプションは、省略可能であり、最初の BE を作成するときにだけ使用できます。lucreate を初めて実行する際に -c を省略した場合、次の規則に従ってデフォルト名が決定されます。

1. 物理ブートデバイスを決定できる場合、そのデバイスのベース名に基づいて新しいブート環境が命名されます。たとえば、物理ブートデバイスが /dev/dsk/c0t0d0s0 であった場合、新しいブート環境は「c0t0d0s0」と命名されます。

2. 物理ブートデバイスを決定できない場合、オペレーティングシステム名 (「uname -s」を使用) とオペレーティングシステムのリリースレベル (「uname -r」を使用) を組み合わせて新しいブート環境が命名されます。たとえば、「uname -s」から「SunOS」が返され、「uname -r」から「5.9」が返される場合、「SunOS5.9」という名前が新しいブート環境に割り当てられます。
3. ブートデバイス名、オペレーティングシステム名のいずれも決定できない場合、「current」という名前が新しいブート環境に割り当てられます。

最初のブート環境の作成後に -c オプションを使用した場合、指定した名前が現在のブート環境名と同じである場合にはそのオプションは無視されます。名前が異なる場合には、エラーメッセージが表示されて処理が終了します。

-C (*boot_device* | –)
lucreate が、どの物理記憶装置がブートデバイスであるか認識できない場合があります。たとえば、x86 マシン上のソース BE にミラー化されたルートデバイスがある場合、このようなことが発生します。-C は、ソース BE のブートに使用する物理ブートデバイスを指定します。このオプションを指定しないと lucreate は BE のブートに使用された物理デバイスを検出しようとします。ルート (/) ファイルシステムのあるデバイスが物理ディスクでない場合 (例: ルートが Solaris ボリュームマネージャのボリューム上にある場合)、lucreate は、その物理デバイスが妥当と推測し、次のような質問メッセージを出力します。

```
Is the physical device devname the boot device for
the logical device devname?
```

y を入力すると、処理が続けられます。

-C *boot_device* を使用すると、lucreate は物理デバイスを検索せずに、指定されたブートデバイスを使用します。-C オプションに – (ハイフン) を指定すると、lucreate が検出したものをブートデバイスとして処理が続けられます。デバイスが特定できない場合は、デバイス名を入力するよう求めるプロンプトが表示されます。

-C を省略した場合や、-C *boot_device* を指定したが lucreate が指定されたブートデバイスを検出できなかった場合は、エラーメッセージが返されます。

-C – の場合、lucreate は、正しいブートデバイスを検出するか、以降の質問メッセージでブートデバイスを指定するように求めるので、この形式を使用する方が安全です。

-f *exclude_list_file*
exclude_list_file の内容を使用して、新しく作成した BE から特定のファイル (ディレクトリを含む) を除外します。*exclude_list_file* にはファイル名およびディレクトリ名が行単位のリストで記載されています。行の項目がファイル名の場合、そのファイルだけが除外されます。項目がディレクトリ名の場合は、該当するディレクトリとその下にあるすべてのファイル (サブディレクトリを含む) が除外されます。

`-I`
完全性の検査を無視します。新しい BE を作成する前に、BE から誤って重要なシステムファイルを除外しないように `lucreate` は完全性検査を実行します。完全性検査を上書きする場合もこのオプションを使用します。このオプションを使用することによる長所は、 `-I` を使用すると BE をより速く作成できるということです。それに対して短所は、BE が期待した通りには動作しない危険があるということです。

`-l` *error_log*
エラーメッセージおよびその他の状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-m` *mount_point*:*device*[,*volume*]:*fs_option*
[`-m` *mount_point*:*device*:*fs_option*] ...
新しい BE の `vfstab`(4) 情報を指定します。`-m` オプションの引数として指定するファイルシステムは、同じディスク上にあっても、複数のディスクにまたがって存在していてもかまいません。

mount_point には、有効なマウントポイントを指定します。– (ハイフン) は、スワップパーティションであることを示します。*device* フィールドには、次のいずれかを指定します。

- ディスクスライスの名前 (書式は `/dev/dsk/c`*num*`t`*num*`d`*num*`s`*num*)。
- Solaris ボリュームマネージャのボリュームの名前 (書式は `/dev/md/dsk/d`*num*)。
- Veritas ファイルシステムの名前 (書式は `/dev/md/vxfs/d`*num*)。
- キーワード `merged`。指定したマウントポイントのファイルシステムがその親とマージされることを示します。
- キーワード `shared`。ソース BE のすべてのスワップパーティションが新しい BE と共有されることを示します。

物理ディスクデバイス名および Solaris ボリュームマネージャのボリューム名は、デバイスを一意に示す最短の名前に短縮できます。たとえば、マシンがデバイス `/dev/dsk/c0t0d0s0` に対して 1 つのディスクコントローラと 1 つのディスクドライブだけを持っている場合、`/dev/dsk/c0t0d0` は省略して `s0` という名前を使用できます。マシンが 1 つのコントローラと複数のディスクを持っている場合は `t0d0s0` という名前を、また、複数のコントローラを持っている場合は `c0t0d0s0` という名前を使用できます。Solaris ボリュームマネージャのボリュームは、その `d`*num* という指定から判断できます。たとえば、`/dev/md/dsk/d10` は単純に `d10` で表わすことができます。

`-m` オプションを指定すると、物理ディスクデバイス Solaris ボリュームマネージャの単一スライス連結に接続したり、Solaris ボリュームマネージャのボリュームをミラーに接続したりできます。どちらの操作も、後述する `attach` キーワードを使って実行できます。このオプションでは、特定の連結またはミラーを指定することができ、または `lucreate` に自動的に選択させることもできます。特定

の連結またはミラーを指定するには、論理デバイスを接続するデバイス名の後にコンマと Solaris ボリュームマネージャの論理デバイス名を付加します。この指定を省略した場合、lucreate によって、空いているデバイスのリストから特定の連結またはミラーが選択されます。「使用例」を参照してください。

fs_option フィールドには、下記に示すキーワードの 1 つまたは複数を指定できます。最初に示す 2 つのキーワードは、ファイルシステムの種類を指定します。ほかのキーワードは、ファイルシステムに対するアクションを指定します。複数のキーワードを指定する場合は、コンマで区切ります。

ufs
ファイルシステムを UFS ボリュームとして作成します。

vxfs
ファイルシステムを Veritas デバイスとして作成します。

preserve
指定した物理記憶装置のファイルシステムの内容を保存します。このキーワードを使用すると、デバイスのファイルシステムとその内容が、指定したマウントポイントに適切であると仮定されます。指定したマウントポイントに保存するデバイスとして指定できるのは 1 つだけです。このキーワードを指定すると、指定した記憶装置に新しいファイルシステムを作成し、ファイルシステムの内容を、ソース BE から指定したデバイスにコピーするというデフォルトの手順を省略できます。preserve、lucreate を使用すると、記憶装置の内容が指定したファイルシステムに適切かどうか検査します。この検査には限界があり、適合性を保証するものではありません。

mirror
指定した記憶装置上にミラーを作成します。指定する記憶装置は、正しく名前のついた (たとえば、/dev/md/dsk/d10、d10 など) ミラーとして働く論理デバイスである必要があります。後続の -m オプションでは、attach を指定して (下記を参照)、少なくとも 1 つの物理デバイスを新しいミラーに接続する必要があります。

attach
ボリュームに含まれる物理記憶装置を、指定したマウントポイントに関連するミラーまたは単一スライス連結に接続します。attach を使用する場合、ディスクを特定のミラーまたは連結に接続するときは、デバイス名の後ろにコンマ、続けて論理デバイスの名前を付与します。コンマと連結名を省略すると、 lucreate はその記憶装置のコンテナボリュームとして空いているミラーまたは単一スライス連結を選択します。「使用例」を参照してください。

lucreate は、1 つの物理ドライブを含む連結だけを作成して、そのような連結を最大 4 つまでミラーに接続できるようにします。

detach
指定したマウントポイントに関連するミラーまたは連結から物理記憶装置の接続を解除します。

ルートには、少なくとも 1 つのディスクまたはデバイススライスを指定する必要があります。この指定には、-m または -M オプション (下記を参照) を使用するか、FMLI ベースのインタフェースを使用します。新しい BE にファイルシステムを作成する場合は、ファイルシステムごとに -m 引数を指定する必要があります。たとえば、ソース BE にある 3 つのファイルシステム (たとえば、/、/usr、および /var) を新しい BE 上に個別のファイルシステムとして作成する場合は、-m 引数を 3 回指定します。-m 引数を 1 回だけ指定した場合、/、/usr、および /var は、1 つのファイルシステムとして新しい BE の / の下にマージされます。

-m オプションを使ってスワップパーティションを指定する場合は、いずれかの BE で現在スワップに使用しているデバイスおよび未使用のデバイスを指定できます。スワップ割り当てについては、次の方法から選択できます。

- ソース BE に関連するすべてのスワップデバイスを新しい BE が使用するように割り当てる場合は、スワップデバイスの指定をすべて省略できます。
- 新しい BE が、指定したスワップデバイスだけを使用してソース BE に関連するスワップデバイスを自動的に共有しないように割り当てる場合は、1 つまたは複数のスワップデバイスを指定します。
- 新しい BE が、指定したスワップデバイスを使用してソース BE とスワップデバイスを共有するように割り当てる場合は、構文 -m –:shared:swap を使用して 1 つまたは複数のスワップデバイスを指定します。

後述の「使用例」を参照してください。

-M *slice_list*
slice_list ファイルには、-m オプションのリストが含まれています。リストに指定する引数の形式は、-m オプションに指定する引数の形式と同じです。# 文字で始まる行は、コメント行で無視されます。-M オプションは、BE に多数のファイルシステムを指定するときに便利です。-m と -M オプションは同時に指定できます。たとえば、スワップパーティションを *slice_list* 内に指定し、/ と /usr スライスを -m オプションで指定することができます。

-m と -M オプションには、1 つのマウントポイントに対して複数のスライスを指定できます。lucreate は、存在しないスライスを無視し、使用可能な最初のスライスを選択します。「使用例」を参照してください。

-n *BE_name*
作成する BE の名前を指定します。*BE_name* は、システムにおいて一意の名前にする必要があります。

-o *outfile*
すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

-s (– | *BE_name*)
新しい BE を作成する際のソースを指定します。このオプションにより、新しい BE を作成するソースとして現在の BE 以外のものを指定できます。

引数としてハイフン (-) を指定すると、新しい BE が作成されますが、ファイルシステム内にファイルは作成されません。-s オプションにこのような引数を指定するのは、続けて luupgrade(1M) を使用して、空の BE にフラッシュアーカイブのインストールを行うためです。flar(1M) を参照してください。

-x *exclude*
新しく作成した BE から *exclude* ファイルまたは *exclude* ディレクトリを除外します。*exclude* がディレクトリ名の場合、lucreate はそのディレクトリと、その下にあるすべてのファイル (サブディレクトリを含む) を除外します。

-X
XML 出力を有効にします。XML の特性は DTD (/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

-y *include*
新しく作成した BE にファイル *include* またはディレクトリ *include* を取り込みます。*include* がディレクトリ名の場合、lucreate はそのディレクトリと、その下にあるすべてのファイル (サブディレクトリを含む) を取り込みます。

-Y *include_list_file*
include_list_file の内容を使用して、新しく作成した BE に特定のファイル (ディレクトリを含む) を取り込みます。*include_list_file* にはファイル名およびディレクトリ名が行単位のリストで記載されています。行の項目がファイル名の場合、そのファイルが取り込まれます。行の項目がディレクトリ名の場合、該当するディレクトリとその下にあるすべてのファイル (サブディレクトリを含む) が取り込まれます。

-z *filter_list_file*
filter_list_file にはファイル名およびディレクトリ名が行単位の項目リストで記載されています。各項目の先頭に + 記号が付いた場合は新しい BE に取り込む項目を、 - が付いた場合は新しい BE から除外する項目を示します。

ファイルの取り込みオプションと除外オプションの組み合わせ

lucreate コマンドを使用すると、新しい BE を作成したときに特定のファイルおよびディレクトリを取り込みまたは除外できます。ファイルおよびディレクトリの取り込みは、次のオプションを使用します。

- -y *include* オプション
- -Y *include_list_file* オプション
- -z *filter_list* オプション。リストファイルの項目名の先頭に + を付ける

ファイルおよびディレクトリの除外は、次のオプションを使用します。

- -x *exclude* オプション
- -f *exclude_list_file* オプション
- -z *filter_list* オプション。リストファイルの項目名の先頭に – を付ける

除外される項目の親ディレクトリが、取り込みオプション (たとえば、 -y *include*) で取り込まれる場合は、*exclude* で指定された特定のファイルまたはディレクトリだけが除外されます。逆に、取り込まれるファイルの親ディレクトリが除外オプションで指定されている場合は、*include* で指定されたファイルだけが取り込まれます。たとえば、次のように指定します。

```
-x /a -y /a/b
```

/a/b を除いたすべての /a が除外されます。また、次のように指定するとします。

```
-y /a -x /a/b
```

/a/b を除いたすべての /a が取り込まれます。

使用例

lucreate コマンドはさまざまな出力を生成します。次の例では、説明のために必要な場合を除き、出力例は省略しています。

例 1 新しいブート環境を初めて作成する

次のコマンドシーケンスを使って、ブート環境がまだ作成されていないマシン上に新しい BE を作成します。共有できないすべての (クリティカルな) ファイルシステムは / の下にマウントされます。

```
# lucreate -c first_disk -m /:/dev/dsk/c0t4d0s0:ufs -n second_disk
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

次のコマンドは、先のコマンドと同様、BE が作成されたことのないマシン上に新しいブート環境を作成します。ただし、次のコマンドは、先のコマンドと 2 つの点で異なります。-c オプションは省略され、/usr ファイルシステムは / からは分離して自身のディスクスライス上にマウントされます。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-n second_disk
lucreate: Please wait while your system configuration is determined.
(出力略)
lucreate: Creation of Boot Environment c0t4d0s0 successful.
```

-c オプションが指定されなかった場合、lucreate は、ルートデバイスのベース名である「c0t4d0s0」を、新しいブート環境に割り当てます。

次に -c オプションを指定して同じコマンドを実行します。

```
# lucreate -c first_disk -m /:/dev/dsk/c0t4d0s0:ufs \
-m /usr:/dev/dsk/c0t4d0s1:ufs -n second_disk
```

例 1　新しいブート環境を初めて作成する　　(続き)

```
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

BE を作成したあと、luupgrade(1M) を使って新しい BE 上で OS をアップグレードし、luactivate(1m) を使ってその BE をアクティブにします。マシンは次回リブートするときに、この BE からブートされます。first_disk のスワップパーティションとすべての共有可能ファイルシステムは、second_disk からも使用 (共有) できます。

```
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image
(出力略)
luupgrade: Upgrade of Boot Environment <second_disk> successful.

# luactivate second_disk
```

これらのコマンドの説明については、luupgrade(1M) と luactivate(1m) のマニュアルページを参照してください。

例 2　現在の BE 以外のソースを使って BE を作成する

-s オプションを使って、現在の BE 以外の BE をソース BE として指定します。

```
# lucreate -s third_disk -m /:/dev/dsk/c0t4d0s0:ufs \
-m /usr:/dev/dsk/c0t4d0s1:ufs -n second_disk
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

例 3　BE をフラッシュアーカイブから作成する

このためには、-s – オプションを指定した lucreate と luupgrade を実行する必要があります。

```
# lucreate -s - -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-n second_disk
brief messages
lucreate: Creation of Boot Environment <second_disk> successful.
```

-s オプションを指定した lucreate コマンドは、数秒で処理を終了します。この時点で luupgrade を実行して、フラッシュアーカイブをインストールできます。

```
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-J "archive_location http://example.com/myflash.flar"
```

このコマンドについては、luupgrade(1M) を参照してください。

例 4　スワップパーティションの共有および追加

もっとも単純な例として、lucreate コマンドでスワップパーティションを 1 つも指定しない場合、ソース BE のすべてのスワップパーティションは新しい BE と共有されます。たとえば、現在の BE がスワップパーティションとして /dev/dsk/c0t4d0s7 を使用すると仮定します。次のコマンドを入力します。

```
# lucreate -n second_disk -m /:/dev/dsk/c0t4d0s0:ufs
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

このコマンドの結果、その BE がアクティブになり、起動したとき、パーティション /dev/dsk/c0t4d0s7 は BE second_disk によって使用されます。

新しい BE にソース BE とは異なるスワップパーティションを使用したい場合は、1 つまたは複数の -m オプションを入力して新しいパーティションを指定します。ここで、現在の BE がスワップパーティションとして /dev/dsk/c0t4d0s7 を使用していると仮定します。次のコマンドを入力します。

```
# lucreate -m /:/dev/dsk/c0t0d0s0:ufs -m -:/dev/dsk/c0t4d0s1:swap \
 -m -:/dev/dsk/c0t4d0s2:swap -n second_disk
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

BE がアクティブになり起動すると、新しい BE second_disk は /dev/dsk/c0t4d0s1 および /dev/dsk/c0t4d0s2 を使用し、/dev/dsk/c0t4d0s7 は使用しません。スワップパーティションはソース BE が使用します。

新しい BE second_disk がソース BE のスワップパーティションを共有していて、さらにスワップパーティションを追加したいと仮定します。次のコマンドを入力します。

```
# lucreate -m /:/dev/dsk/c0t0d0s0:ufs -m -:/dev/dsk/c0t4d0s1:swap \
 -m -:shared:swap -n second_disk
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

BE がアクティブになり起動すると、新しい BE second_disk はスワップに /dev/dsk/c0t4d0s7 を使用して、ソース BE と共有し、さらに /dev/dsk/c0t4d0s1 を使用します。

例 5　複数のディスク上にあるスワップパーティションを共有する

次のコマンドは、BE を 2 台目のディスクに作成し、1 台目と 2 台目の両方のディスク上にあるスワップパーティションを共有できるようにします。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m -:/dev/dsk/c0t4d0s1:swap \
 -m -:/dev/dsk/c0t0d0s1:swap -n second_disk
```

例 5　複数のディスク上にあるスワップパーティションを共有する　*(続き)*

*(*出力略*)*

```
lucreate: Creation of Boot Environment <second_disk> successful.
```

上記のコマンドが完了すると、second_disk という BE は、/dev/dsk/c0t0d0s1 と /dev/dsk/c0t4d0s1 をスワップパーティションとして使用します。ただし、このスワップ割当は、second_disk からブートが行なわれるまで有効になりません。スワップパーティションが多数ある場合は、次の例のように -M オプションを使用すると便利です。

例 6　-m と -M オプションを組み合わせて使用する

この例では、スワップパーティションのリストを /etc/lu/swapslices ファイルに指定します。このファイルの場所と名前はユーザーが定義できます。/etc/lu/swapslices ファイルの内容を次に示します。

```
-:/dev/dsk/c0t3d0s2:swap
-:/dev/dsk/c0t3d0s2:swap
-:/dev/dsk/c0t4d0s2:swap
-:/dev/dsk/c0t5d0s2:swap
-:/dev/dsk/c1t3d0s2:swap
-:/dev/dsk/c1t4d0s2:swap
-:/dev/dsk/c1t5d0s2:swap
```

上記のファイルは次のコマンドで指定されます。

```
# lucreate -m /:/dev/dsk/c02t4d0s0:ufs -m /usr:/dev/dsk/c02t4d0s1:ufs \
-M /etc/lu/swapslices -n second_disk
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

BE second_disk は、/etc/lu/swapslices に指定されたパーティションをスワップとして使用します。

例 7　コピーと共有

次のコマンドは、現在の BE にあるユーザーファイルシステム /home (および、共有不能なファイルシステム / と /usr) を新しい BE にコピーします。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-m /home:/dev/dsk/c0t4d0s4:ufs -n second_disk
```

前述のコマンドとは異なり、次のコマンドには、/home の宛先を指定する -m オプションが指定されていません。コマンドを実行すると、/home は現在の BE と BE second_disk の間で共有されます。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-n second_disk
```

例 8　Solaris ボリュームマネージャのボリュームの使用

次のコマンドがあります。

1. ミラー d10 を作成して、このミラーをルートファイルシステムを受容体として確立します。
2. c0t0d0s0 および c0t1d0s0 を単一スライス連結 d1 および d2 にそれぞれ接続します。これらのボリュームの指定は任意であることに注意してください。
3. c0t0d0s0 および c0t1d0s0 に関連する連結をミラー d10 に接続します。
4. 現在の BE のルートファイルシステムをミラー d10 にコピーして、d10 の内容をすべて上書きします。

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0,d1:attach \
-m /:/dev/dsk/c0t1d0s0,d2:attach -n newBE
```

次のコマンドは、物理記憶装置用の連結が指定されていないという点のみ、先のコマンドと異なります。この例では、lucreate は空いている名前のリストから連結名を選択し、これらのボリュームを最初の -m オプションで指定されたミラーに接続します。

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0:attach \
-m /:/dev/dsk/c0t1d0s0:attach -n newBE
```

次のコマンドは、作成するミラーにデバイスを接続する前に、ミラーから物理ディスクの 1 つを切り離すという点で、先のコマンドとは異なります。また、物理ディスクの 1 つの内容が保存されます。このコマンドは次のことを行います。

1. ミラー d10 を作成して、このミラーをルートファイルシステムを受容体として確立します。
2. c0t0d0s0 を現在接続されているミラーから切り離します。
3. c0t0d0s0 および c0t1d0s0 を連結 d1 および d2 にそれぞれ接続します。これらの連結の指定は任意であることに注意してください。
4. c0t0d0s0 の内容を保存します。c0t0d0s0 には現在の BE のルートファイルシステムの有効なコピーが含まれていると仮定します。
5. c0t0d0s0 および c0t1d0s0 に関連する連結 (d1 および d2) をミラー d10 に接続します。

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0,d1:detach,attach,preserve \
-m /:/dev/dsk/c0t1d0s0,d2:attach -n newBE
```

上記のコマンドは、次のように短縮できます。

例 8　Solaris ボリュームマネージャのボリュームの使用　　*(続き)*

```
# lucreate -m /:d10:ufs,mirror \
-m /:c0t0d0s0:detach,attach,preserve \
-m /:c0t1d0s0:attach -n newBE
```

上の例では、デバイス名 (物理デバイス名および論理デバイス名) が短縮され、連結 (d1 および d2) の指定子が省略されていることに注意してください。

次のコマンドは、この一連の例で最初に紹介したコマンドに続くコマンドです。このコマンドは、連結 (c0t0d0s0 を含む) を 1 つのミラー (最初のコマンドでは d10) から切り離し、別のミラー (d20) に接続してその内容を保存します。

```
# lucreate -m /:/dev/md/dsk/d20:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0:detach,attach,preserve -n nextBE
```

次のコマンドは、2 つのミラーを作成し、一方のミラー上には新しい BE の / ファイルシステムを、もう一方のミラー上には新しい BE の /opt ファイルシステムを置きます。

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0,d1:attach \
-m /:/dev/dsk/c1t0d0s0,d2:attach \
-m /opt:/dev/md/dsk/d11:ufs,mirror \
-m /opt:/dev/dsk/c2t0d0s1,d3:attach \
-m /opt:/dev/dsk/c3t1d0s1,d4:attach -n anotherBE
```

例 9　FMLI ベースのインタフェースの呼び出し

lu インタフェースはすでに非推奨になっているため、この例が含められているのは歴史的な経緯からです。

次のコマンドには -m と -M オプションのいずれも指定されていないため、Live Upgrade 操作を行う FMLI ベースのインタフェースが呼び出されます。

```
# lucreate -n second_disk
```

上記のコマンドは、ターゲット BE second_disk のソース BE として現在の BE を使用します。FMLI インタフェースで、second_disk のターゲットスライスを指定します。次のコマンドは、これまでに紹介した例の応用です。

```
# lucreate -n second_disk -s third_disk
```

上記のコマンドには、ターゲット BE のソースが指定されています。前述の例と同様に、FMLI インタフェースが表示されるので、新しい BE のターゲットスライスを指定します。

例 10 ファイルシステムのマージ

次のコマンドは、ファイルシステム /usr/opt をファイルシステム /usr にマージします。まず次の例では、BE first_diskにあるディスクスライスを -m オプションの引数形式で表したものです。

```
/:/dev/dsk/c0t4d0s0:ufs
/usr:/dev/dsk/c0t4d0s1:ufs
/usr/opt:/dev/dsk/c0t4d0s3:ufs
```

次のコマンドは、BE second_diskを作成し、/usr/opt をその親の /usr にマージします。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-m /usr/opt:merged:ufs -n second_disk
```

例 11 ファイルシステムの分割

/、/usr、および /var のすべてが同じディスクスライス上にマウントされているソース BE があるとします。次のコマンドは、/、/usr、および /var がそれぞれ異なるディスクスライスにマウントされた BE second_diskを作成します。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
/var:/dev/dsk/c0t4d0s3:ufs -n second_disk
```

ファイルシステム (ルートなど) のコンポーネントを異なるディスクスライスに分割するこのような操作を、ファイルシステムの分割といいます。

例 12 代替スライスの指定

次のコマンドは、新しい BE second_disk の代替ディスクスライスとして複数の -m オプションを使用します。

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /:/dev/dsk/c0t4d0s1:ufs \
-m /:/dev/dsk/c0t4d0s5:ufs -n second_disk
(出力略)
lucreate: Creation of Boot Environment <second_disk> successful.
```

上記のコマンドは、/ ファイルシステムのディスクスライスの候補として s0、s1、s5 を指定しています。lucreate は、この 3 つのスライスの中から他の BE によって使用されていない最初のスライスを選択します。-s オプションが省略されているため、新しい BE は現在の BE をソースとして作成されます。

終了ステータス

次の終了ステータスが返されます。

0 正常終了。

>0 エラーが発生した。

ファイル

`/etc/lutab`
システム上にある BE のリスト

`/usr/share/lib/xml/dtd/lu_cli.dtd.`*<num>*
Live Upgrade の DTD (`-X` オプションを参照)

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

注意事項

すべての Solaris オペレーティング環境のアップグレードに当てはまりますが (Live Upgrade の機能は除きます)、ディレクトリを複数のマウントポイントに分割すると、ハードリンクは解除されます。たとえば、`/usr/test1/buglist` と `/usr/test2/buglist` がハードリンクされているとします。`/usr/test1` と `/usr/test2` を別々のファイルシステムに分割すると、これらのファイル間のリンクは切断されます。`lucreate` が複数のファイルシステムにまたがるハードリンクを検出すると、コマンドは警告メッセージを出力し、失われたハードリンクに代わるシンボリックリンクが作成されます。

`lucreate` は、共有不能なファイルシステムについて無効な構成が作成されても検出できません。たとえば、`/` と `/kernel` を別々のファイルシステムとして作成する `lucreate` コマンドを実行することは可能です。しかし、作成した BE はブートできません。ブート環境用のファイルシステムを作成するときは、Solaris オペレーティング環境用のファイルシステムを作成する場合と同じ規則が適用されます。

これまでの説明をふまえて、次のことに留意してください。

- ソース BE では、コピーしたり新しい BE と共有する各ファイルシステムの有効な `vfstab` エントリが必要です。
- 重複するパーティション (つまり、同じ物理ディスク領域を共有するパーティション) には、新しい BE を作成することはできません。このようなディスクに BE を作成すると、`lucreate` コマンドは問題を検出しませんが、作成された BE はブートできません。

注 – `-m` オプションの説明にあるとおり、Solaris ボリュームマネージャのボリュームをブート環境に使用する場合は、Solaris ボリュームマネージャのコマンドではなく `lucreate` を使用してこれらのボリュームを操作してください。Solaris ボリュームマネージャソフトウェアはブート環境を認識しません。`lucreate` コマンドには、たとえば誤って Solaris ボリュームマネージャのボリュームを上書きしたり削除したりしてブート環境を破壊することを避ける検査機能が含まれています。

Solaris 10 より以前の Solaris オペレーティングシステムについては、Live Upgrade は、配布中のリリース、および最大 3 つのリリースをさかのぼってサポートします。たとえば、Solaris 9 (Solaris 9 Upgrade を含む) の Live Upgrade バージョンを入手した場合、Solaris 9 に加えて Solaris 2.6、Solaris 7、および Solaris 8 版をサポートします。Live Upgrade は Solaris 2.6 より以前の Solaris をサポートしていません。

Solaris 10 オペレーティングシステムからは、Live Upgrade は、配布中のリリースおよび最大 2 つのリリースをさかのぼってサポートします。たとえば、Solaris 10 (Solaris 10 Upgrade を含む) の Live Upgrade バージョンを入手した場合、Solaris 10 に加えて Solaris 8 と Solaris 9 をサポートします。

Solaris Live Upgrade を正しく操作するためには、指定の OS バージョン用の特定のパッチリビジョンのセットがインストールされている必要があります。Live Upgrade をインストールする、または実行する前に、特定のパッチリビジョンのセットをインストールする必要があります。`http://sunsolve.sun.com` を参照して、最新のパッチリストがあることを確認してください。SunSolve Web サイトで infodoc 72099 を検索してください。

名前

lucurr – アクティブなブート環境名の表示

形式

```
/usr/sbin/lucurr [-l error_log] [-m mount_point]
     [-o outfile] [-X]
```

機能説明

`lucurr` コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

`lucurr` コマンドは、現在実行中のブート環境 (BE) の名前を表示します。システム上に BE が構成されていない場合は、 'No Boot Environments are defined ' というメッセージを表示します。`lucurr` で表示されるのは現在の BE の名前だけであり、次回のリブート時にアクティブになる BE の名前は表示されません。次回のリブート時にアクティブになる BE の情報が必要な場合は、lustatus(1M) または luactivate(1m) を使用してください。

`lucurr` コマンドを実行するには、root 権限が必要です。

オプション

`lucurr` コマンドには、次のオプションを指定できます。

`-l` *error_log*
: エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-m` *mount_point*
: *mount_point* (BE のルートファイルシステムのマウントポイント) を所有している BE の名前を返します。*mount_point* には、現在の BE のマウントポイントまたはそれ以外の BE のマウントポイントのどちらでも指定できます。後者の場合、このオプションを使用する前に、lumount(1M) または mount(1M) を使って BE のファイルシステムをマウントしておく必要があります。

`-o` *outfile*
: すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

`-X`
: XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

終了ステータス

次の終了ステータスが返されます。

`0`
: 正常終了。

`>0`
: エラーが発生した。

ファイル

`/etc/lutab`	システム上にある BE のリスト
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade の DTD (`-X` オプションを参照)

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

名前　ludelete – ブート環境の削除

形式

```
/usr/sbin/ludelete  [-l error_log] [-o outfile] BE_name
 [-X]
```

機能説明

ludelete コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、live_upgrade(5) のマニュアルページを参照してください。

ludelete コマンドは、BE 上でステータスが完了になっているブート環境 (BE) に関連する、すべてのレコードを削除します。完了している BE とは、lucreate(1M)、luupgrade(1M)、または lucompare(1M) 操作が進行中でない BE のことです。BE のステータスを確認するには、lustatus(1M) を使用します。アクティブな BE と、次回のリブート時にアクティブになる BE は削除できません。

ludelete は、削除される BE 上のファイルは変更しません。

ludelete コマンドを実行するには、root 権限が必要です。

オプション

ludelete コマンドには、次のオプションを指定できます。

-l *error_log*　エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

-o *outfile*　すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

-X　XML 出力を有効にします。XML の特性は DTD (/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

オペランド

BE_name　削除する BE の名前を指定します。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生した。

ファイル

/etc/lutab	システム上にある BE のリスト
/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>	Live Upgrade の DTD (-X オプションを参照)

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目 lu(1M), luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludesc(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), lutab(4), attributes(5), live_upgrade(5)

名前　lufslist – ブート環境の構成の一覧表示

形式　`/usr/sbin/lufslist [-l` *error_log*`] [-o` *outfile*`]` *BE_name* `[-X]`

機能説明　`lufslist` コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

`lufslist` コマンドは、ブート環境 (BE) の構成を一覧表示します。出力内容は、BE のマウントポイントごとのディスクスライス (ファイルシステム)、ファイルシステムタイプ、およびファイルシステムのサイズです。出力には、表示されている BE 内部の非大域ゾーンに属するすべての個別ファイルシステムも示されます。

次に示す `lufslist` の出力例は、表示されている BE 内部のゾーン `zone1` の個別のファイルシステムを示します。

```
# lufslist BE_name
Filesystem              fstype       size(Mb) Mounted on
------------------------------------------------------------------
/dev/dsk/c0t0d0s1       swap          512.11 -
/dev/dsk/c0t4d0s3       ufs          3738.29 /
/dev/dsk/c0t4d0s4       ufs           510.24 /opt

              zone zone1 within boot environment BE_name
/dev/dsk/c0t4d0s7       ufs           7000.48 /export
```

ファイルシステムタイプは `ufs`、`swap`、`vxfs` (Veritas ファイルシステム) のいずれかです。`Filesystem` の見出しの下には、ディスクスライスまたは論理デバイス (ボリューム管理ソフトウェアが使用するディスクメタデバイスなど) が表示されます。

`lufslist` コマンドを実行するには、root 特権または Primary Administrator 役割が必要となります。

オプション　`lufslist` コマンドには、次のオプションを指定できます。

`-l` *error_log*　エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-o` *outfile*　すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

`-X`　XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

オペランド　*BE_name*　ファイルシステムの情報を表示する BE の名前を指定します。その他の Live Upgrade 操作で使用されている BE は指定できません。

終了ステータス　次の終了ステータスが返されます。

`0` 正常終了。

`>0` エラーが発生した。

ファイル

`/etc/lutab`	システム上にある BE のリスト
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade の DTD (`-X` オプションを参照)

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5), `zones`(5)

名前

lumake – ブート環境の生成

形式

```
/usr/sbin/lumake [-l error_log] [-o outfile] [-s source_BE] -n BE_name
     [-X]

/usr/sbin/lumake [-l error_log] -t time [-o outfile]
     [-s source_BE] -n BE_name [-m email_address] [-X]
```

機能説明

lumake コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、live_upgrade(5) のマニュアルページを参照してください。

lumake コマンドは、ファイルをコピーすることにより、指定のブート環境 (BE) のファイルシステムを作成します。具体的には、アクティブな BE またはソース BE (-s) のファイルシステムからファイルをコピーします。ターゲット BE 上の既存のデータはすべて削除されます。ターゲット BE 上のすべてのファイルシステムは再作成されます。

ターゲット BE には、必ず既存の BE を指定します。新しい BE を作成するには、lucreate(1M) を使用します。

lumake コマンドを実行するには、root 権限が必要です。

オプション

lumake コマンドには、次のオプションを指定できます。

-n *BE_name* 生成する BE の名前を指定します。

-s *source_BE* オプションとしてソース BE の名前を指定します。このオプションを省略すると、現在の BE がソースとして使用されます。コピーを行う前に、BE のステータスが complete になっていることを確認してください。BE のステータスを確認するには、lustatus(1M) を使用します。

-l *error_log* エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

-o *outfile* すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

-t *time* 指定の時刻に指定の BE を生成するバッチジョブを設定します。時刻の指定には、at(1) と同じ形式を使用します。複数の Live Upgrade 操作を同時にスケジュールすることはできません。lucancel(1M) を使用すると、スケジュール済みの lumake 操作を取り消すことができます。

-m *email_address* lumake コマンドの完了時に、その出力内容を電子メールで指定アドレスに送信できるようにします。*email_address* の検査は行われません。バッチコマンドの完了通知を受け取るには、このオプションを -t とともに使用します。-t 以外のオプションと -m を組み合わせて使用すると、電子メールは送信されません。

`-X` XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

終了ステータス 次の終了ステータスが返されます。

`0` 正常終了。

`>0` エラーが発生した。

ファイル

`/etc/lutab`	システム上にある BE のリスト
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade の DTD (`-X` オプションを参照)

属性 次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目 luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

名前

lumount, luumount – ブート環境内のすべてのファイルシステムのマウントまたはマウント解除

形式

```
/usr/sbin/lumount [-l error_log] [-o outfile] BE_name
     [mount_point] [-X]

/usr/sbin/lumount

/usr/sbin/luumount [-f]
     { [-n] BE_name | [-m] mount_point | block_device}
     [-l error_log] [-o outfile] [-X]
```

機能説明

`lumount` および `luumount` コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の一部です。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

`lumount` および `luumount` コマンドを使えば、ブート環境 (BE) 内のすべてのファイルシステムをマウントまたはマウント解除できます。これにより、ある BE がアクティブになっていない間に、その BE 内のファイルを検査または変更することができます。`lumount` はデフォルトで、`/.alt.`*BE_name* という形式のマウントポイントにファイルシステムをマウントします。ここで、*BE_name* は、ファイルシステムをマウントする BE の名前です。「注意事項」 を参照してください。

`lumount` および `luumount` は、BE 内部のすべてのインストール済み非大域ゾーンもマウントまたはマウント解除します。`lumount` は、現在の BE 内の実行中、マウント済み、または準備のできた非大域ゾーンごとに、非大域ゾーンに属するマウント済み BE 内のすべてのファイルシステムを、非大域ゾーン内の指定されたマウントポイントにマウントします。これにより、マウント済みの BE 内に存在する対応するファイルシステムへの非大域ゾーン管理者アクセスが可能になります。

引数を指定しないで `lumount` を呼び出した場合、システム上のマウント済み BE の名前が返されます。

`lumount` および `luumount` コマンドを実行するには、root 特権または Primary Administrator 役割が必要となります。

オプション

`lumount` および `luumount` コマンドのオプションを、次に示します。

`-f`
: `luumount` 専用。強制的でないマウント解除の試行 (および失敗) のあと、BE のファイルシステムを強制的にマウント解除します。このオプションは、mount(1M) `-f` オプションに似ています。

`-l` *error_log*
: エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-m` *mount_point*
: `luumount` は、*mount_point* を所有する BE のファイルシステムをマウント解除します。次の「オペランド」に含まれる *mount_point* の説明を参照してください。`luumount` でマウントポイントを指定する場合、`-m` は省略可能です。

`-n` *BE_name*　ファイルシステムをマウント解除する BE の名前。次の「オペランド」に含まれる *BE_name* の説明を参照してください。`luumount` で BE 名を指定する場合、`-n` は省略可能です。

`-o` *outfile*　すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

`-X`　XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

`luumount` でユーザーが引数を指定し、かつ `-m`、`-n` のどちらも指定しなかった場合、このコマンドは、その引数が BE 名、マウントポイント、ブロックデバイスのいずれであるかを判定します。その引数がこれら 3 つの 1 つであり、かつその引数に関連する BE がファイルシステムをマウントしていた場合、`luumount` はその BE のファイルシステムをマウント解除します。 それ以外の場合、`luumount` はエラーを返します。

オペランド

BE_name　ファイルシステムをマウントまたはマウント解除する BE の名前。これは、現在のシステム上に存在している、アクティブ BE 以外の BE です。`lumount` または `luumount` コマンドを正しく完了するには、lustatus(1M) で取得される BE のステータスが complete でなければならないことに注意してください。また、BE のどのディスクスライスも、(mount(1M) を使って) マウントすることはできません。

mount_point　`lumount` の場合、デフォルトの `/.alt.`*BE_name* の代わりに使用するマウントポイント。*mount_point* が存在しない場合、`lumount` はそれを作成します。`luumount` の場合、*mount_point* に関連付けられた BE のファイルシステムがマウント解除されます。`luumount` でのマウント解除時にデフォルトのマウントポイントが自動的に削除されることに注意してください。ユーザーが指定したマウントポイントは削除されません。

block_device　`luumount` 専用。*block_device* は、`/dev/dsk/c0t4d0s0` のような、BE のルートスライスです。`luumount` は、*block_device* に関連付けられた BE のファイルシステムをマウント解除します。

使用例

例 1　マウントポイントの指定

次のコマンドは、マウントポイント `/test` を作成し、BE `second_disk` のファイルシステムを `/test` にマウントします。

```
# lumount second_disk /test
/test
```

その後、`cd` を実行して `/test` に移動し、`second_disk` のファイルシステムを参照できます。`/test` をマウントポイントとして指定しなかった場合、`lumount` は、`/.alt.second_disk` という名前のデフォルトのマウントポイントを作成します。

例1 マウントポイントの指定 (続き)

システムに非大域ゾーンをインストール済みの場合、このコマンドは、現在稼働しているシステム内の対応する非大域ゾーンにある second_disk 内のすべての非大域ゾーンも、マウントポイント /test (マウントポイントが指定されていない場合は /.alt.second_disk) でマウントします。

例2 ファイルシステムのマウント解除

次のコマンドは、BE second_disk のファイルシステムをマウント解除します。この例では、cd を実行して / に移動していますが、これは、second_disk 内のどのファイルシステムにも入っていない状態にするためです。

```
# cd /
# luumount second_disk
#
```

/dev/dsk/c0t4d0s0 が second_disk のルートスライスである場合には、次のコマンドを入力しても、上記のコマンドと同じ効果が得られます。

```
# cd /
# luumount /dev/dsk/c0t4d0s0
#
```

終了ステータス

次の終了ステータスが返されます。

0 正常終了。

>0 エラーが発生した。

ファイル

/etc/lutab	システム上にある BE のリスト
/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>	Live Upgrade の DTD (-X オプションを参照)

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), ludesc(1M), lufslist(1m), lumake(1M), lurename(1M), lustatus(1M), luupgrade(1M), lutab(4), attributes(5), live_upgrade(5), zones(5)

注意事項

BE 名にスラッシュ (/) が含まれていた場合、lumount は、デフォルトマウントポイント名で、それらのスラッシュをコロンに置き換えます。たとえば、次のように指定します。

```
# lumount 'first/disk'
/.alt.first:disk
```

名前

lurename – ブート環境名の変更

形式

```
/usr/sbin/lurename -e BE_name -n new_name [-l error_log]
    [-o outfile] [-X]
```

機能説明

`lurename` コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、`live_upgrade`(5) のマニュアルページを参照してください。

`lurename` コマンドは、ブート環境 (BE) の名前を *BE_name* から *new_name* に変更します。

new_name は 30 文字を超えてはならず、使用できる文字は、英数字とその他の ASCII 文字 (UNIX シェルにとって特殊な意味を持つ文字は除く) だけです。`sh`(1) の「クォート」セクションを参照してください。BE 名に使用できるのは、8 ビットで表現できるシングルバイトの文字だけです。空白文字を含めることはできません。*new_name* は、システム上で一意のものでなければなりません。

BE 名を変更する前に、BE のステータスが complete になっていることを確認してください。BE のステータスを確認するには、lustatus(1M) を使用します。なお、lumount(1M) または mount(1M) でファイルシステムがマウントされている BE の名前は変更できません。

BE 名の変更は、BE を Solaris の新しいリリースにアップグレードするときに有用です。たとえば、オペレーティングシステムのアップグレードに伴って、BE 名を `solaris7` から `solaris8` に変更できます。

`lurename` コマンドを実行するには、root 権限が必要です。

オプション

`lurename` コマンドには、次のオプションを指定できます。

`-e` *BE_name*
: 変更したい BE の名前を指定します。

`-l` *error_log*
: エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-n` *new_name*
: 新しい BE 名を指定します。*new_name* は、システム上で一意のものでなければなりません。

`-o` *outfile*
: すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

`-X`
: XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

終了ステータス

次の終了ステータスが返されます。

`0`
: 正常終了。

`>0`
: エラーが発生した。

ファイル

`/etc/lutab`	システム上にある BE のリスト
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade の DTD (`-X` オプションを参照)

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), `ludesc`(1M), lufslist(1m), lumake(1M), lumount(1M), lustatus(1M), luupgrade(1M), `lutab`(4), `attributes`(5), `live_upgrade`(5)

名前　lustatus – ブート環境のステータスの表示

形式

```
/usr/sbin/lustatus [-l error_log] [-o outfile] [BE_name]
     [-X]
```

機能説明

lustatus コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、live_upgrade(5) のマニュアルページを参照してください。

lustatus コマンドは、ブート環境 (BE) *BE_name* のステータスを表示します。BE を指定しない場合は、システム上のすべての BE のステータスを表示します。

lustatus コマンドでは、次の見出しの下に情報が出力されます。

Boot Environment Name　BE の名前です。

Is Complete　BE がブート可能かどうかを示します。現在何らかの操作が行われている場合や、lucreate(1M) または luupgrade(1M) の操作に失敗した場合、BE は未完了のステータスになります。たとえば、BE 上でコピー操作が実行中であったり、コピー操作がスケジュールされている場合、その BE のステータスは未完了と見なされます。

Active　BE が現在アクティブであるかどうかを示します。「アクティブな」BE とは、現在ブートされている BE です。

ActiveOnReboot　BE が次回のシステムのリブート時にアクティブになるかどうかを示します。

Can Delete　BE 上でコピー、比較、またはアップグレードの操作が行われていないことを示します。また、この BE のファイルシステムは、現在 1 つもマウントされていません。これらの条件がすべてそろったとき、この BE を削除できます。

Copy Status　BE の作成や再生成がスケジュールされているかどうか、あるいは、アクティブ (実行中) かどうかを示します。この見出しの下に、ACTIVE、COMPARING (lucompare(1M) を参照)、UPGRADING、または SCHEDULED というステータスが表示されている場合、Live Upgrade 機能を使ったコピー、名前の変更、アップグレードは実行できません。

lustatus の出力例を以下に示します。

```
BE_name             Complete  Active  ActiveOnReboot  CopyStatus
------------------------------------------------------------------------
disk_a_S7            yes       yes     yes              -
disk_b_S7database    yes       no      no              UPGRADING
disk_b_S8            no        no      no               -
```

未完了である disk_b_S8 のコピー、名前の変更、アップグレードは実行できません。同じく、Live Upgrade 操作が保留状態になっている disk_b_S7database のコピー、名前の変更、アップグレードも実行できないことに注意してください。

lustatus コマンドを実行するには、root 権限が必要です。

オプション

lustatus コマンドには、次のオプションを指定できます。

-l *error_log* エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

-o *outfile* すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

-X XML 出力を有効にします。XML の特性は DTD (/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

オペランド

BE_name ステータスを表示する BE の名前を指定します。*BE_name* を省略すると、システム内のすべての BE のステータスが表示されます。

終了ステータス

次の終了ステータスが返されます。

0 成功および完了しました。

>0 エラーが発生しました。

ファイル

/etc/lutab	システム上にある BE のリスト
/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>	Live Upgrade DTD (-X オプションを参照)

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludesc(1M), ludelete(1m), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), luupgrade(1M), lutab(4), attributes(5), live_upgrade(5)

名前

luupgrade – ブート環境上のソフトウェアのインストール、アップグレード、およびその他の機能の実行

形式

```
/usr/sbin/luupgrade [-iIufpPtTcC] [options]
```

機能説明

luupgrade コマンドは、Solaris オペレーティング環境の Live Upgrade 機能を提供するコマンド群の 1 つです。Live Upgrade 機能の説明については、live_upgrade(5) のマニュアルページを参照してください。

luupgrade コマンドは、指定したブート環境 (BE) にソフトウェアをインストールできるようにします。具体的な機能は次のとおりです。

- BE 上のオペレーティングシステムイメージをアップグレードする (-u オプション)。Solaris フラッシュアーカイブなど、有効な Solaris インストール媒体をイメージのソースとして指定できる
- BE 上に Solaris フラッシュアーカイブを抽出する (-f オプション)。flar(1M) を参照
- BE にパッケージを追加する (-p)。または、BE から パッケージを削除する (-P)
- BE にパッチを追加する (-t)。または、BE からパッチを削除する (-T)
- パッケージに関する情報をチェックします (-I)。または、パッケージの情報を確認する
- オペレーティングシステムのインストール媒体を確認する (-c)。

luupgrade を使用する前に、lucreate(1M) コマンドを使用して BE を作成しておく必要があります。アップグレードできるのは、現在の BE 以外の BE です。

これらの機能では、それぞれに固有のオプションを使用できます。各機能で使用可能なオプションについてはその機能の説明を参照してください。

luupgrade を正しく完了するには、lustatus(1M) で取得される BE のステータスが、complete でなければならないことに注意してください。なお、lumount(1M) または mount(1M) でディスクスライスがマウントされている BE に対しては、luupgrade を実行できません。

luupgrade を使用すると、luupgrade を呼び出すマシンで実行しているリリースとは別のマーケティングリリースの Solaris オペレーティングシステムイメージをインストールできます。この機能を使用するには、次の条件があります。

- 以前のリリースを実行しているマシンに、指定した Solaris オペレーティングシステムのリリースから Live Upgrade パッケージ (SUNWluu、SUNWlur、および SUNWlucfg) をインストールできます。これらのパッケージは、Live Upgrade パッケージのリリースより 3 つ前のリリースまでの Solaris を実行しているマシン上にインストールできます。Live Upgrade は Solaris 2.6 より前のリリースではサポートされていません。そのため、たとえば、Solaris 2.9 パッケージは Solaris 2.8、2.7、および 2.6 マシンにインストールできます。

- マシンにインストールされている Live Upgrade パッケージと同じリリースの Solaris オペレーティングシステムリリースにアップグレードできます。この機能により、マーケティングリリース範囲内の Solarisアップグレードリリースにアップグレードできます。たとえば、マシンに Solaris 9 FCS Live Upgrade パッケージがインストールされている場合は、 `luupgrade` を使用して BE を Solaris オペレーティングシステムの Update 3 リリースにアップグレードできます。

Live Upgrade パッケージのインストール方法については、『Solaris インストールガイド』を参照してください。

`luupgrade` コマンドを実行するには、root 権限が必要です。

常に指定可能なオプション

次のオプションは、どのような目的で `luupgrade` を実行する場合でも指定できます。

`-l` *error_log* エラーメッセージと状態メッセージを、現在の環境での書き込み先だけでなく、*error_log* にも書き込みます。

`-o` *outfile* すべてのコマンド出力を、現在の環境での書き込み先だけでなく、*outfile* にも書き込みます。

`-N` dry-run モード。コマンド引数の指定が正しいかどうかを確認できるようにします。`-c` (媒体の検査) には適用されません。

`-X` XML 出力を有効にします。XML の特性は DTD (`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>) に定義されています。<*num*> は、各 DTD ファイルのバージョン番号を示します。

オペレーティングシステムイメージのアップグレード

オペレーティングシステムのイメージをアップグレードするには、`luupgrade` コマンドに `-u` を指定します。この場合の構文は次のとおりです。

```
luupgrade -u -n BE_name [ -l error_log ] [ -o outfile ][-N]
 -s os_image_path [ -j profile_path [-D] ]
```

最初のオプション `-u` は、OS イメージのインストールを実行することを示します。この `luupgrade` を使用する場合に指定可能なその他のオプションは、次のとおりです。

`-n` *BE_name* OS をアップグレードする BE の名前のどちらでも指定できます。

`-s` *os_image_path* OS イメージが格納されているディレクトリのパス名を指定します。DVD や CD などのインストール媒体上のディレクトリか、NFS または UFS ディレクトリのどちらでも指定できます。

`-j` *profile_path* JumpStart プロファイルのパスを指定します。`luupgrade` で呼び出すプロファイルで使用する有効なキーワードのリストについては、後述のセクション「JumpStart プロファイルのキーワー

ド」を参照してください。JumpStart ソフトウェアについては、pfinstall(1M) のマニュアルページと Solaris のインストールマニュアルを参照してください。

-D　指定した BE のディスク構成に対して -j で指定されたプロファイルの値をテストします。アップグレードは行われません。このオプションの結果は、プロファイルをテストする事前実行です。luupgrade はその出力に指定されたログファイルを作成するので、コマンドの結果を調べることができます。

ブート環境をアップグレードする前に、次をのことを行ってください。

- analyze_patches を実行します。
- アップブレード対象のオペレーティングシステムと同じバージョンの Live Upgrade パッケージをインストールします。

Solaris Software DVD (以前の Solaris Installation CD) の /Misc ディレクトリから analyze_patches コマンドを入手できます。 このコマンドで、どのパッチがアップグレードの結果として削除されるかを調べます。アップグレード後、analyze_patches で調べておいたパッチを再インストールできます。

Live Upgrade パッケージ、SUNWluu、SUNWlur、および SUNWlucfg は、Solaris software DVD (Solaris のバージョンによっては CD) から入手できます。-u オプションを使って luupgrade を実行する前に、アップグレードしようとしている Solaris のバージョンのパッケージをインストールしたことを確認します。

複数のコンポーネントから成る媒体 (たとえば複数の DVD など) からアップグレードを行う場合 は、次に説明するように、-i オプションを指定して、2 番目あるいはそれ以降の媒体からソフトウェアをインストールできるようにします。

インストーラプログラムを実行したアップグレードの継続

インストーラプログラムを実行するには、luupgrade コマンドに -i を指定します。次に説明するように、このオプションは主に、オプション -u で luupgrade を実行したあとで実行します。オプション -i の構文は次のとおりです。

```
luupgrade -i -n BE_name [ -l error_log ] [ -o outfile ] [ -N ]
-s installation_medium [ -O "installer_options" ]
```

最初のオプション -iは、 -s で指定された途中からのインストールのためのインストーラプログラムを実行することを示します。この luupgrade を使用する場合に指定可能なその他のオプションは、次のとおりです。

-n *BE_name*　ソフトウェアのインストール先となる BE の名前を指定します。

-O "*installer_options*"　Solaris インストーラプログラムに直接渡されるオプションを指定します。インストーラオプションについては、installer(1M) を参照してください。

`-s` *installation_medium* インストール媒体のパス名を指定します。これは、DVD、CD、NFS、および UFS のいずれのディレクトリも可能です。

`-i` オプションを指定すると、`luupgrade` は指定した媒体上でインストールプログラムを検索し、実行します。

オプション `-i` の主要な用途は、複数の DVD などから成る複数のコンポーネントからのオペレーティングシステムイメージをアップグレードすることです。 ここでは、`-u` を使用して `luupgrade` を実行してから `-i` オプションを使用して `luupgrade` コマンドを実行します。「使用例」を参照してください。`-u` オプションはについては、上記を参照してください。

Solaris フラッシュアーカイブからのインストール

Solaris フラッシュアーカイブからオペレーティングシステムをインストールするには、`luupgrade` コマンドに `-f` を指定します。アーカイブをインストールすると、ターゲットとなる BE のすべてのファイルが上書きされることに注意してください。この場合の構文は次のとおりです。

```
luupgrade -f -n BE_name [ -l error_log ] [ -o outfile ] [ -N ] [-D]
-s os_image_path ( -a archive | -j profile_path | -J "profile" )
```

最初のオプション `-f` は、Solaris フラッシュアーカイブから OS のインストールを行うことを示します。この `luupgrade` を使用する場合に指定可能なその他のオプションは、次のとおりです。

`-n` *BE_name* OS のインストール先となる BE の名前のどちらでも指定できます。

`-D` 指定した BE のディスク構成に対して `-j` または `-J` で指定されたプロファイルの値をテストします。アップグレードは行われません。このオプションの結果は、プロファイルをテストする事前実行です。`luupgrade` はその出力に指定されたログファイルを作成するので、コマンドの結果を調べることができます。

`-s` *os_image_path* OS イメージが格納されているディレクトリのパス名を指定します。DVD や CD などのインストール媒体上のディレクトリか、NFS または UFS ディレクトリのどちらでも指定できます。

`-a` *archive* ローカルファイルシステム上でアーカイブが使用可能な状態になっているときは、Solaris フラッシュアーカイブのパスを指定します。`-a`、`-j`、または `-J` のいずれかを指定する必要があります。

`-j` *profile_path* Solaris フラッシュインストール用に構成された JumpStart プロファイルのパスを指定します。`luupgrade` で呼び出すプロファイルで使用する有効なキーワードのリストについては、後述のセクション「JumpStart プロファイルのキーワード」を参照してください。JumpStart ソフトウェアについて

は、pfinstall(1M) のマニュアルページと Solaris のインストールマニュアルを参照してください。-a、-j、または -J のいずれかを指定する必要があります。

-J *"profile"*
Solaris フラッシュインストール用に構成された JumpStart プロファイルのエントリを指定します。このオプションに有効な唯一のキーワードは、archive_location です。JumpStart ソフトウェアについては、pfinstall(1M) のマニュアルページと Solaris のインストールマニュアルを参照してください。-a、-j、または -J のいずれかを指定する必要があります。

-s で指定した OS イメージのバージョンは、-a、-j、-J のいずれかのオプションで指定した Solaris フラッシュアーカイブ内の OS のバージョンと同じでなければなりません。

パッケージの追加と削除

パッケージを追加する場合は、luupgrade コマンドに -p を指定します。パッケージを削除する場合は -P を指定します。この場合の構文は次のとおりです。

パッケージの追加:

```
luupgrade -p -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
 -s packages_path [ -O "pkgadd_options" ] [ -a admin ]
[ pkginst [ pkginst...]]
```

パッケージの削除:

```
luupgrade -P -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "pkgrm_options" ] [ pkginst [ pkginst...]]
```

最初のオプションは、パッケージを追加する場合は -p、削除する場合は -P を指定します。この luupgrade を使用する場合に指定可能なその他のオプションは、次のとおりです。

-n *BE_name*
パッケージの追加先または削除先となる BE の名前を指定します。

-s *packages_path*
(パッケージの追加の場合のみ) 追加するパッケージが格納されているディレクトリのパス名を指定します。-s オプションは -d オプションで代用することもできます。-d は、pkgadd(1M) との互換性を確保する目的でサポートされています。

-d *packages_path*
-s オプションと同じです。-s オプションの使用をお勧めします。

-O "*pkgadd_options*" または "*pkgrm_options*"
パッケージを追加する場合 (-p) は pkgadd に直接渡されるオプション、削

除する場合 (-P) は pkgrm に直接渡されるオプションを指定します。これらのコマンドのオプションについては、pkgadd(1M) のマニュアルページと pkgrm(1M) のマニュアルページを参照してください。

-a *admin*
: (パッケージの追加の場合のみ) admin ファイルのパスを指定します。pkgadd の -a オプションと同じ意味になります。ここでの -a オプションは、-O "-a *admin*" と同等です。

pkginst [*pkginst...*]
: 追加または削除するゼロ個以上のパッケージを指定します。パッケージを追加する場合、上記の -s オプションで指定したすべてのパッケージが追加されます (デフォルト)。複数のパッケージ名は空白文字で区切って指定します。

追加するパッケージは、すべて『SVR4 Advanced Packaging Guidelines』 に準拠していなければなりません。「警告」の項を参照してください。

パッチの追加と削除

パッチを追加する場合は、luupgrade コマンドに -t を指定します。パッチを削除する場合は -T を指定します。この場合の構文は次のとおりです。

パッチの追加:

```
luupgrade -t -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
-s patch_path [ -O "patchadd_options" ] [ patch_name [ patch_name...]]
```

パッチの削除:

```
luupgrade -T -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "patchrm_options" ] [ patch_name [ patch_name...]]
```

最初のオプションは、パッチを追加する場合は -t、削除する場合は -T を指定します。この luupgrade を使用する場合に指定可能なその他のオプションは、次のとおりです。

-n *BE_name*
: パッチの追加先となる BE の名前、またはパッチを削除する BE の名前を指定します。

-s *patch_path*
: (パッチの追加の場合のみ) 追加するパッチが格納されているディレ

クトリのパス名または patch_order ファイルのパス名を指定します。

-O "*patchadd_options*" または "*patchrm_options*"	パッチを追加する場合 (-p) は patchadd に直接渡されるオプション、削除する場合 (-P) は patchrm に直接渡されるオプションを指定します。これらのコマンドのオプションについては、patchadd(1M) のマニュアルページまたは patchrm(1M) のマニュアルページを参照してください。
patch_name [*patch_name...*]	追加または削除するゼロ個以上のパッチを指定します。パッチを追加する場合、上記の -s オプションで指定したすべてのパッチが追加されます (デフォルト)。複数のパッチ名は空白文字で区切って指定します。

追加するパッケージは、すべて『SVR4 Advanced Packaging Guidelines』 に準拠していなければなりません。「警告」 の項を参照してください。

パッケージ情報の確認と表示

-C を指定すると、BE 上のすべてのパッケージまたは指定したパッケージに対して pkgchk(1M) が実行されます。-I オプションを指定すると、pkginfo(1) が実行されます。

pkgchk の実行:

```
luupgrade -C -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "pkgchk_options" ][ pkginst [ pkginst...]]
```

pkginfo の実行:

```
luupgrade -I -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "pkginfo_options" ][ pkginst [ pkginst...]]
```

最初のオプションは、pkgchk の場合は -C、pkginfo の場合は -I を指定します。この luupgrade を使用する場合に指定可能なその他のオプションは、次のとおりです。

-n *BE_name*	パッケージを確認する BE、またはパッケージ情報を表示する BE の名前を指定します。
-O "*pkgchk_options*" または "*pkginfo_options*"	pkgchk に直接渡されるオプション (-C) または pkginfo に直接渡されるオプション (-I) を指定します。これら

のオプションについては、pkgchk(1M) のマニュアルページまたは pkginfo(1) のマニュアルページを参照してください。

pkginst [*pkginst...*]　チェックするゼロ個以上のパッケージ名、または情報を表示するゼロ個以上のパッケージ名を指定します。パッケージ名を省略すると、BE 上のすべてのパッケージ情報が返されます。複数のパッケージ名を指定する場合は、スペースで区切ります。

OS のインストール媒体の確認

-c オプションで luupgrade はローカルまたは、リモートの媒体、たとえば DVD や CD などが有効なインストール媒体かどうかチェックします。-c オプションは、指定した媒体に関する有用な情報を返します。 このような使用のための luupgrade コマンドの構文は次のようになります。

```
luupgrade -c [ -l error_log ] [ -o outfile ] -s path_to_medium
```

最初のオプション -c は、これから実行する処理 (インストール媒体のチェック) を示します。上記の -s オプションの働きは次のとおりです。

-s *path_to_medium*　パス名は、DVD や CD などのインストール媒体です。

JumpStart プロファイルのキーワード

このセクションでは、luupgrade のプロファイルで使用できる Solaris JumpStart キーワードを指定します。-j オプションに -u (アップグレード) オプション、または -f (フラッシュ) オプションを組み合わせて使用します。-u オプションの場合、必須のキーワードはありません。-f オプションは、完全なフラッシュアーカイブの場合は値を install_type: flash_install に、差分フラッシュアーカイブの場合は値を flash_update に指定する必要があります。-j オプションと -f オプションを合わせて使用する場合は、-a (アクティブな位置) オプションを指定するか、またはプロファイル中で archive_location キーワードを指定する必要があります。

archive_location キーワードは -J オプションに有効な唯一の引数です。

次のキーワードオプションは、-u オプションおよび -f オプションで使用するプロファイルに使用することがあります。

cluster　システムに追加するソフトウェアグループを指定します。

geo　システムにインストール、または追加する地域ロケールまたはロケールを指定します。使用可能な値の一覧については、『Solaris インストールガイド』 を参照してください。

isa_bits　64 ビットまたは 32 ビットのどちらのパッケージをインストールするかを指定します。有効な値は 64 と 32 です。

`locale` システムにインストールまたは追加するロケールパッケージを指定します。使用可能な値の一覧については、『Solaris インストールガイド』 を参照してください。

`package` システムに追加する、またはシステムから削除するパッケージを指定します。

次のキーワードは、`luupgrade` で使用するプロファイルに使用してはなりません。

- `boot_device`
- `dontuse`
- `fdisk`
- `filesys`
- `layout_constraint`
- `noreboot`
- `partitioning`
- `root_device`
- `usedisk`

すべての JumpStart プロファイルキーワードの説明、および JumpStart プロファイルを作成する手順については『Solaris インストールガイド』 を参照してください。

使用例

例 1 パッケージを削除して追加する

次の例は、ブート環境から複数のパッケージを削除して、追加し直します。

```
# luupgrade -P -n second_disk SUNWabc SUNWdef SUNWghi
```

同じパッケージを追加します。

```
# luupgrade -p -n second_disk -s /net/installmachine/export/packages \
SUNWabc SUNWdef SUNWghi
```

次の例では、上記のコマンドに `-O` オプションを追加することにより、引数を直接 `pkgadd` に渡しています。

```
# luupgrade -p -n second_disk -s /net/installmachine/export/packages \
-O "-r /net/testmachine/export/responses" SUNWabc SUNWdef SUNWghi
```

このコマンドのオプションについては、pkgadd(1M) のマニュアルページを参照してください。

例 2 結合イメージを使用して、新しい OS にアップグレードする

次の例は、ブート環境上のオペレーティング環境をアップグレードします。ソースイメージは、リモートディスクまたは DVD 上に結合イメージとしてあります。

例 2　結合イメージを使用して、新しい OS にアップグレードする　　(続き)

```
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image
```

上記のコマンドに続けて次のコマンドを実行すると、アップグレードした BE がアクティブになります。

```
# luactivate second_disk
```

次回のリブート時に、`second_disk` が現在のブート環境になります。luactivate(1m) のマニュアルページを参照してください。

例 3　複数の CD を使用して、新しい OS にアップグレードする

次の例は、これまでに紹介した例の応用です。OS のアップグレードは CD 2 枚に格納されています。SPARC マシン上でアップグレードを開始するには、次のコマンドを入力します。

```
# luupgrade -u -n second_disk -s /cdrom/cdrom0/s0
```

x86 マシンでは、`-s` の引数 `s0` を `s2` で置き換えます。

1 枚目の CD の内容の処理が完了したら、ドライブに 2 枚目の CD を挿入し、次のコマンドを入力します。

```
# luupgrade -i -n second_disk -s /cdrom/cdrom0 \
-O "-nodisplay -noconsole"
```

継続してインストールするときは、`-u` オプションよりも `-i` を使用してください。3 枚目以降も同様に `-i` オプションを使用してください。上記の `-O` オプションは、installer(1M) に渡されます。これらのオプションを省略すると、CD の挿入と読み取りが完了した時点でグラフィカルユーザーインタフェースが起動します。`-O` オプションについては installer(1M) を参照してください。

複数の CD からアップグレードを行う場合、すべての CD に対して `luupgrade` コマンドを入力し、その実行が完了した時点でアップグレードが終了します。各 CD からパッケージをインストールしたあと、次のようなメッセージが出力される場合があります。

```
WARNING: <num> packages must be installed on boot environment <disk_device>.
```

これは、上記の例のように、複数の CD からパッケージのインストールが必要であるということを示すものです。全パッケージのインストールが完了していないと、アップグレードの完了した BE を `luactivate` を使用して有効化 (ブート先として指定) することはできません。

例 4　JumpStart プロファイルを使用したアップグレード

次のコマンド例は、プロファイル /home2/profiles/test.profile をテストするのに -D オプションを使用しています。

```
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-j /home2/profiles/test.profile -D
```

このコマンドの結果が満足できる内容の場合は、上記のコマンドから -D オプションを省略して、アップグレードを実行します。

例 5　Solaris フラッシュアーカイブから新しい OS をインストールする

次の例は、 Solaris フラッシュアーカイブを使ってブート環境上のオペレーティング環境をアップグレードします。-J オプションで指定するファイルは、フラッシュインストールを指定する JumpStart プロファイルです。

```
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-J "archive_location http://example.com/myflash.flar"
```

次のコマンドが上記のコマンド例と異なるのは、 -j オプションが -J オプションに置き換わっている点だけです。どちらかのコマンドに -D オプションを追加すると、実際にフラッシュインストールを実行する前にプロファイルをテストできます。

```
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-j /net/example/flash_archives/flash_gordon
```

先の 2 つのコマンドは、完全フラッシュインストール、または差分フラッシュインストールとして動作します。差分フラッシュインストールか完全フラッシュインストールかは、プロファイル中の install_type キーワードの値で決まります。前述の「JumpStart プロファイルキーワード」を参照してください。

例 6　パッケージ情報を取得する

次の例は、pkgchk に -v オプションを指定して、SUNWluu、SUNWlur、および SUNWlucfg パッケージに対して pkgchk を実行する例です。

```
# luupgrade -C -n second_disk -O "-v" SUNWluu SUNWlur SUNWlucfg
```

次のコマンドは、上記の 2 つのパッケージに対して pkginfo を実行します。

```
# luupgrade -I -n second_disk -O "-v" SUNWluu SUNWlur SUNWlucfg
```

例 6 パッケージ情報を取得する　(続き)

どちらのコマンドの場合でも、パッケージ名を省略すると、指定した BE の全パッケージの情報が返されます。これらのコマンドのオプションについては、pkgchk(1M) のマニュアルページと pkginfo(1) のマニュアルページを参照してください。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生した。

ファイル

/etc/lutab	システム上にある BE のリスト
/usr/share/lib/xml/dtd/lu_cli.dtd.<*num*>	Live Upgrade の DTD (「常に指定可能なオプション」の -X オプションを参照)

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWluu

関連項目

installer(1M), luactivate(1m), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludelete(1m), ludesc(1M), lufslist(1m), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), lutab(4), attributes(5), live_upgrade(5), zones(5)

警告

パッケージまたはパッチを追加する場合は (-p、-P、-t、-T のいずれかを指定)、『SVR4 Advanced Packaging Guidelines』および『『Solaris 10 インストールガイド (基本編)』』の付録 C に記載されているガイドラインに準拠したパッケージまたはパッチを指定してください。これらに準拠したパッケージまたはパッチは、pkgadd(1M) または patchadd(1M) の -R オプション (説明については各ユーティリティーのマニュアルページを参照) に準拠することになります。ほぼすべての Sun のパッケージおよびパッチがこのガイドラインに準拠していますが、Sun 以外のベンダーのパッケージはガイドラインに準拠していない場合があります。また、Sun の古いパッケージおよびパッチの中には、-R オプションに対応していないものがあります。そのようなパッケージまたはパッチを見つけた場合は、Sun に報告してください。準拠していないパッケージまたはパッチを指定すると、luupgrade のパッケージまたはパッチ追加ソフトウェアを実行したときに問題が発生したり、現在の BE が変更されてしまう可能性があります。

注意事項

Solaris 10 より以前の Solaris オペレーティングシステムについては、Live Upgrade は、配布中のリリース、および最大 3 つのリリースをさかのぼってサポートします。たとえば、Solaris 9 (Solaris 9 Upgrade を含む) の Live Upgrade バージョンを入手

した場合、Solaris 9 に加えて Solaris 2.6、Solaris 7、および Solaris 8 版をサポートします。Live Upgrade は Solaris 2.6 より以前の Solaris をサポートしていません。

Solaris 10 オペレーティングシステムからは、Live Upgrade は、配布中のリリースおよび最大 2 つのリリースをさかのぼってサポートします。たとえば、Solaris 10 (Solaris 10 Upgrade を含む) の Live Upgrade バージョンを入手した場合、Solaris 10 に加えて Solaris 8 と Solaris 9 をサポートします。

Solaris Live Upgrade を正しく操作するためには、指定の OS バージョン用の特定のパッチリビジョンのセットがインストールされている必要があります。Live Upgrade をインストールする、または実行する前に、特定のパッチリビジョンのセットをインストールする必要があります。`http://sunsolve.sun.com` を参照して、最新のパッチリストがあることを確認してください。SunSolve Web サイトで infodoc 72099 を検索してください。

名前

luxadm – Sun Fire 880 記憶装置サブシステムと FC_AL デバイスの管理

形式

```
luxadm [options]... subcommand [options]... enclosure
     [,dev] | pathname...
```

機能説明

`luxadm` プログラムは、SENA、Sun Fire 880 内蔵記憶装置サブシステム、および各 FC_AL (Fiber Channel Arbitrated Loop) デバイスの管理コマンドです。`luxadm` は、コマンド行に指定される引数やオプションに応じて、さまざまな制御処理や照会処理を実行します。

コマンド行にはサブコマンドの指定が必要です。コマンド行には、サブコマンドに応じてオプションやパラメータを指定することができます。オプションには、通常 1 つ以上の格納装置名またはパス名を指定します。指定する文字数は、サブコマンドを一意に識別するのに必要なだけでかまいません。

サブコマンドが作用するデバイスをパス名で指定します。Sun StorEdge A5000 サブシステムでは、パス名の代わりに、デバイスまたはデバイスに対するポートのワールドワイド名 (WWN) を入力することによってディスクデバイスまたは格納装置サービスコントローラを指定することができます。また、Sun StorEdge A5000 の格納装置名および格納装置内の特定のデバイスを示す識別子 (オプション) を入力することによってデバイスを指定することもできます。各 FC_AL デバイスを指定するには、デバイスの WWN またはポートを入力します。

パス名

デバイスまたはコントローラを、完全な物理パス名または完全な論理パス名で指定します。

Sun StorEdge A5000 に対するデバイスの一般的な物理パス名を次に示します。

```
/devices/sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037000f96,
        0:a,raw
```

システム上のすべての Sun StorEdge A5000 IB (Interface Board)に対しては、物理パスへの論理リンクが `/dev/es` ディレクトリに格納されます。論理リンクの例として、`/dev/es/ses0` などがあります。

FC_AL デバイスや Sun StorEdge A5000 サブシステム IB を選択するために、パス名の代わりに WWN を使用することもできます。WWN は、デバイスを使用するためのポートまたはデバイスそのものを表す 16 桁の 16 進数の値です。一般的な WWN の値を次に示します。

```
2200002037000f96
```

WWN の形式については、「注意事項」を参照してください。

Sun Fire 880 内蔵記憶装置サブシステムのディスクの一般的な物理パス名を次に示します。

```
/devices/pci@8,600000/SUNW,qlc@2/fp@0,0/ssd@w2100002037a6303c,0:a
```

次に、一般的な論理パス名を示します。

```
/dev/rdsk/c2t8d0s2
```

各 FC_AL デバイスの一般的なパス名を示します。

```
/devices/sbus@3.0/SUNW,socal@d,10000/sf@0,0/ssd@w2200002037049fc3,0:a,raw
```

次に、一般的な論理パス名を示します。

```
/dev/rdsk/c1t0d0s2
```

格納装置

Sun StorEdge A5000 では、デバイスは格納装置名とスロット名で特定することができます。

box_name[,f*slot_number*]
box_name[,r*slot_number*]

box_name は、Sun StorEdge A5000 の格納装置名で、`enclosure_name` サブコマンドで指定します。オプションの *slot_number* パラメータを指定しないと、*box_name* には、Sun StorEdge A5000 サブシステム IB が指定されます。

`f` または `r` は、Sun StorEdge A5000 の格納装置の前面のスロット(f) か、背面のスロット (r) かを指定します。

slot_number は、Sun StorEdge A5000 の格納装置のデバイスのスロット番号を指定します。指定することができる番号の範囲は `0`〜 `6` または `0` 〜 `10` です。

Sun Fire 880 内蔵記憶装置サブシステムも、デバイスは格納装置名とスロット名で特定することができます。ただし、1 セットのディスクしかありません。

box_name[,s*slot_number*]

box_name は、Sun Fire 880 内蔵記憶装置サブシステムの格納装置名で、`enclosure_name` サブコマンドで指定します。オプションの *slot_number* パラメータを指定しないで使用した場合、*box_name* には、Sun Fire 880 内蔵記憶装置サブシステム格納装置のサービスデバイスが指定されます。Sun Fire 880 内蔵記憶装置サブシステムのスロット番号、`0` 〜 `11` のいずれかを指定するには `s` を使用します。

ディスクおよびサブシステムの論理名については、`disks`(1M) および `devlinks`(1M) を参照してください。

オプション

次のオプションはすべてのサブコマンドに対して指定することができます。

`-e` エキスパート (Expert) モードです。このオプションは初心者の方にはお勧めしません。

`-v` 詳細表示モード。

特定のサブコマンドだけに指定するオプションについては、「使用方法」の各サブコマンドの説明を参照してください。

オペランド　次のオペランドを指定できます。

enclosure　Sun StorEdge A5000 の *box_name* を指定します。

fibre_channel_HBA_port　ホストのコントローラポートへのパスを指定します。一般的なパスは、下記のとおりです。

```
/devices/pci@8,600000/pci@1/SUNW,qlc@4/fp@0,0:devctl
```

pathname　SENA IB、Sun Fire 880 内蔵記憶装置サブシステム、またはディスクデバイスの論理パスまたは物理パスを指定します。*pathname* は、SENA IB、SENA ディスク、または各 FC_AL デバイスの WWN でも指定することができます。

使用法

サブコマンド

`display` *enclosure*[,*dev*] . . . | *pathname* . . .
`display -p` *pathname* . . .
`display -r` *enclosure*[,*dev*] . . . | *pathname* . . .
`display - v` *enclosure*[,*dev*] . . . | *pathname* . . .

格納装置またはデバイス固有のデータを表示します。

サブシステムのデータは、格納装置の環境検知情報、およびディスクの状態を含む、すべてのサブシステムのデバイスの状態で構成されます。

ディスクデータは、照会、容量、および設定情報で構成されます。

`-p`　*pathname* で指定されたデバイスまたはサブシステムの性能情報を表示します。このオプションは、性能情報を保存するサブシステムに対してのみ指定することができます。

`-r`　pathname で指定された FC_AL デバイスのエラー情報を表示します。また、パス名が Sun StorEdge A5000 の場合は、そのループ上のすべてのデバイスのエラー情報を表示します。`-r` オプションは Sun StorEdge A5000 サブシステムまたは各 FC_AL デバイスに対してのみ指定することができます。

`-v`　モード検知データを含む、詳細表示モードで表示します。

`download` [`-s`] [`-w` *WWN*] [`-f` *filename_path*] *enclosure*. . .

prom イメージを、*filename_path* で指定された SENA サブシステムのインタフェースボード装置か、enclosure または pathname で指定された SPARCstorageArray コントローラにダウンロードします。SPARCstorage Array は、ダウンロードコードを使用するためにリセットする必要があります。

SENA のダウンロードが終了すると、SENA はリセットされ、ダウンロードコードが実行されます。ファイル名が指定されていない場合は、デフォルトの prom イメージが使用されます。SPARCstorage Array コントローラ用のデフォルトの prom イメージは、`/usr/lib/firmware/ssa/ssafirmware` ディレクトリに格納され

ます。SENA 用のデフォルトの prom イメージは /usr/lib/locale/C/LC_MESSAGES ディレクトリに ibfirmware というファイル名で格納されます。

Sun Fire 880 内蔵記憶装置サブシステムのダウンロードが終了すると、サブシステムはリセットされ、ダウンロードコードが実行されます。Sun Fire 880 内蔵記憶装置サブシステムのデフォルトのファームウェアイメージは、次のディレクトリに格納されます。

/usr/platform/SUNW,Sun-Fire-880/lib/images/int_fcbpl_fw。

-s 保存オプションです。-s オプションはダウンロードしたファームウェアを FEPROM に保存します。-s を省略すると、ダウンロードしたファームウェアは保存されず、電源の再投入後に消えてしまいます。

-s オプションは SPARCstorage Array コントローラには指定することができません。

-s オプションを使用すると、ダウンロードサブコマンドがサブシステム上の FEPROMを変更するため、-s オプションの使用には注意が必要です。

enclosure_name *new_name enclosure* | *pathname*

enclosure または pathname で指定された単数または複数の格納装置名を変更します。新しい名前は 16 文字以下で *new_name* に指定します。英数字のみが使用できます。このサブコマンドは Sun StorEdge A5000 に対してだけ指定することができます。

failover primary | secondary *pathname*

指定した論理ボリュームに、どの Sun StorEdge T3 array パートナーグループのコントローラがアクセスするかを選択します。primary を指定した場合、プライマリコントローラから論理ボリュームにアクセスします。secondary を指定した場合、*pathname* で指定したセカンダリコントローラから論理ボリュームにアクセスします。

fcal_s_download [-f *fcode-file*]

fcode-file で指定されたファイルに含まれる fcode をすべての FC/S SBus カードにダウンロードします。このコマンドは対話型で、fcode をダウンロードする前にユーザーに対して確認を促します。

fcal_s_download は、シングルユーザーモードでだけ使用してください。入出力操作が行われているホストアダプタに対して、そのアダプタを更新する目的で fcal_s_download を指定すると、アダプタのリセットの原因となります。新しく更新した fcode は、システムを再起動したときに実行され、表示できるようになります。

-f *fcode-file* -f *fcode-file* オプションを省略すると、各 FC100/S SBus カードの中の現在のバージョンのfcode が表示されます。

fcode_download -p

fcode_download -d *dir-name*

インストール済みの FC/S、FC100/S、FC100/P、または FC100/2P ホストバスアダプタ カードを検出し、*dir-name* の fcode を適切なカードにダウンロードします。このコマンドは対話型で、各ファイルタイプに対して適切なカードを決定します。各デバイスに fcode をダウンロードする前にユーザーに対して確認を促します。

`fcode_download` は、シングルユーザーモードで fcode だけを読み込むときに使用します。入出力操作があるホストアダプタに対して、そのアダプタを更新する目的で `fcode_download` を指定すると、アダプタのリセットの原因となります。新しく更新した fcode は、システムを再起動したときに実行され、表示できるようになります。

`-d` *dir-name*　*dir-name* ディレクトリに格納されている fcode ファイルを適切なアダプタカードにダウンロードします。`-d` オプションを省略すると、デフォルトの`usr/lib/firmware/fc_s` ディレクトリが使用されます。

`-p`　各カードに読み込まれている現在のバージョンの fcode を表示します。ダウンロードは行われません。

`inquiry` *enclosure*[,*dev*] . . . | *pathname* . . .
enclosure または pathname で指定されたデバイスに対する照会情報を表示します。

`insert_device` [*enclosure,dev* . . .]
このコマンドを使用して、電源を入れたまま 1 つまたは複数のデバイスを追加することができます。ホットプラグ操作中の制限に関しては「注意事項」を参照してください。このサブコマンドは Sun StorEdge A5000、RSM および各 FC_AL デバイスに対してのみ指定することができます。Sun StorEdge A5000に対して複数の格納装置が指定された場合は、複数のバスに対して同時に追加が行われます。このサブコマンドの引数を省略すると、すべての格納装置または各 FC_AL デバイスが追加されます。RSM に対して指定することができるコントローラは 1 つだけです。Sun StorEdge A5000 に対しては、このサブコマンドは、電源を入れたまま 1 つまたは複数のデバイスを追加するすべての過程をユーザーとの対話によって行います。複数のディスクが指定された場合は、それらのディスクが正しいかどうかの確認が行われ、ユーザーは継続するか中止するかを選択することができます。その後で、ディスクまたは格納装置の追加を実行するかどうかの確認が行われ、それらのデバイスの論理パス名が作成および表示されます。

`led` *enclosure,dev* . . . | *pathname*. . .
enclosure または pathname で指定されたディスクに対応している LED の現在の状態を表示します。このサブコマンドはこの機能を持つサブシステムに対してのみ指定することができます。

`led_blink` *enclosure,dev* . . . | *pathname* . . .
enclosure または pathname で指定されたディスクに対応している LED の点滅を開始するようサブシステムに指示します。このサブコマンドはこの機能を持つサブシステムに対してのみ指定することができます。

`led_off` *enclosure,dev . . .* | *pathname . . .*
enclosure または pathname で指定されたディスクに対応している LED を消灯させるようサブシステムに指示します。Sun StorEdge A5000 サブシステムでは、LED の消灯や点滅の停止ができる状態ときない状態があります。『Sun StorEdge A5000 設置・サービスマニュアル』(805-4111) を参照してください。このサブコマンドはこの機能を持つサブシステムに対してのみ指定することができます。

`led_on` *pathname . . .*
pathname で指定されたディスクに対応している LED を点灯させるようサブシステムに指示します。このサブコマンドはこの機能を持つサブシステムに対してだけ指定することができます。

`power_off` [`-F`] *enclosure*[,*dev*] . . . | *pathname . . .*
`power_off` *pathname* [*enclosure-port*] . . . | *controller tray-number*
Sun StorEdge A5000 に対してこのオプションを指定した場合は、Sun StorEdge A5000 サブシステムが省電力モードに切り替わります。Sun StorEdge A5000 ドライブは、省電力モードでは使用することができません。SPARCstorage Array 中の格納装置サービスカードに対してこのオプションを指定した場合は、RSM トレーの電源が切れます。Sun StorEdge A5000 中のドライブに対してこのオプションを指定した場合は、そのドライブがドライブ off/unmated モードに設定されます。ドライブ off/unmated 状態では、ドライブが停止し、バイパスモードになります。このコマンドは、Sun Fire 880 内蔵記憶装置サブシステムには使用できません。

`-F` 強制オプションは Sun StorEdge A5000 のみに適用されます。このオプションを指定すると、`luxadm` は、1 つまたは複数のデバイスを、それらがホストによって使用されていても電源切断しようとします。

警告: 現在使用されているデータを含むデバイスの電源を切断すると、予想不可能な結果を引き起こします。デバイスの電源を切断する際は、まず、通常の方法を (`-F` を指定せずに) 試してください。このオプションは、通常の確認を無効にすることによる結果を理解した上で使用してください。

`power_on` *enclosure* [, *dev*] . .
Sun StorEdge A5000 に対してこのオプションを指定した場合は、Sun StorEdge A5000 サブシステムが省電力モードから抜けます。Sun StorEdge A5000 ドライブは、省電力モードでは使用することができません。SPARCstorage Array RSM トレーの電源をプログラムによって投入する方法はありません。Sun StorEdge A5000 の中のドライブに対してこのオプションを指定した場合は、そのドライブが通常の起動状態に設定されます。このコマンドは、Sun Fire 880 内蔵記憶装置サブシステムには使用できません。

`probe` [`-p`]
接続された Sun StorEdge A5000 サブシステムおよび各 FC_AL デバイスすべてに関する情報を検索して表示します。この情報には、論理パス名、WWN、および格

納装置名が含まれます。このサブコマンドは、同じ格納装置名を持つ異なる Sun StorEdge A5000 を見つけると、ユーザーに対して警告を発します。

`-p` 物理パス名を表示に含めます。

`qlgc_s_download` [`-f` *fcode-file*]
fcode-file ファイルに格納されている fcode をすべての FC100/P、FC100/2P PCIホストアダプタカードにダウンロードします。このコマンドは対話型で、fcode を各ドライブにダウンロードする前にユーザーに対して確認を促します。
`qlgc_s_download` は、シングルユーザーモードでだけ使用してください。入出力操作があるホストアダプタに対して、そのアダプタを更新する目的で `qlgc_s_download` を指定すると、アダプタのリセットの原因となります。新しく更新した fcode は、システムを再起動したときに実行され、表示できるようになります。

`-f` *fcode-file* `-f` オプションを省略すると、各 FC100/P、FC100/2P PCI カードの、現在のバージョンの fcode が表示されます。

`release` *pathname*
指定されたディスクの予約を解除します。パス名は、ディスクの物理または論理パス名でなければなりません。SPARCstorage Array コントローラのパス名を指定した場合、SPARCstorage Array 中のすべてのディスクの予約が解除されます。

このサブコマンドは、履歴および診断目的以外では使用しないでください。

`remove_device` [`-F`] *enclosure*[,*dev*] . . . | *pathname* . . .
このコマンドを使用して、電源を入れたまま 1 つまたは複数のデバイスを削除することができます。このサブコマンドはすべての格納装置を削除する場合にも指定することができます。このサブコマンドは Sun StorEdge A5000、RSM、および各 FC_AL デバイスに対して指定することができます。ホットプラグ操作中の制限に関しては「注意事項」を参照してください。Sun StorEdge A5000 および各 FC_AL デバイスに対しては、このサブコマンドは、電源を入れたまま 1 つまたは複数のデバイスを削除するすべての過程をユーザーとの対話によって行います。複数のディスクが指定された場合は、それらのディスクが正しいかどうかの確認が行われ、ユーザーは継続するか中止するかを選択することができます。その後で、ディスクまたは格納装置の削除を実行するかどうかの確認が行われ、それらのデバイスの論理パス名が作成および表示されます。

多重ホストディスクの場合、実行する手順は次のとおりです。

- 最初のホストに対して、`luxadm remove_device` コマンドを発行します。続行するかどうかを確認するプロンプトが表示されたら、待機します。
- 2 番目のホストに対して、`luxadm remove_device` コマンドを発行します。続行するかどうかを確認するプロンプトが表示されたら、待機します。
- 最初のホストに対して、`remove_device` コマンドを続行します。デバイスを削除するプロンプトが表示されたら、これを実行します。

- ほかのホストに対して、`luxadm remove_device` コマンドを最後まで実行します。

`-F` 1 つまたは複数のデバイスに対してホットプラグを適用するように `luxadm` に指示します。これらのデバイスがホストで使用されている場合 (*busy* または *reserved* の場合) にも適用されます。ホットプラグオペレーションが強制的に実行されます。

警告: 現在使用されているデータが格納されているデバイスを取り外すと、予期しない結果が発生します。通常は、`-F` を指定しないでホットプラグを適用します。`-F` は、通常のホットプラグ検査を無効にしたときの結果がわかっているときにだけ使用します。

`reserve` *pathname* 指定されたディスクを、ホストによる排他的利用のために 予約します。パス名は、ディスクの物理または論理パス名でなければなりません。pathname が SPARCstorage Array コントローラのパス名の場合、SPARCstorage Array 中のすべてのディスクが予約されます。

このサブコマンドは、履歴および診断目的以外では使用しないでください。

`set_boot_dev` [`-y`] *pathname* システム PROM 中のブートデバイス変数を、*pathname* で指定された物理デバイス名に設定します。指定する物理デバイス名は、ブロック型特殊デバイス、または起動ファイルシステムのマウント先のディレクトリのパス名です。通常、このコマンドは、PROM の中のデフォルト起動デバイスを設定するためにユーザーに対して確認を促します。`-y` オプションを指定すると、ユーザーに対して確認を要求しません。

`start` *pathname* 指定されたSENA のディスクを起動します。

`stop` *pathname...* 指定されたディスク SENA のを停止します。

SENA、Sun Fire 800 内蔵記憶装置サブシステムおよび各 FC_AL デバイスのエキスパートモードサブコマンド

次のサブコマンドは、経験が豊富なユーザーのみが使用することができます。また、Sun StorEdge A5000 およびファイバチャネルループのみを対象にしています。これらのコマンドは、Sun StorEdge A5000 サブシステムとファイバチャネルループの知識が豊富なユーザーだけが使用することができます。

バスを操作するエキスパートサブコマンドにディスクを指定した場合は、そのサブコマンドは、指定したディスクに接続されたバスを操作します。

`-e bypass [-ab]` *enclosure,dev*
`-e bypass -f` *enclosure*

エンクロージャサービスコントローラに対して、指定したポートおよびデバイスの LRC (ループ冗長回路) をバイパス状態に設定するよう要求します。

このサブコマンドは、次のオプションをサポートします。

`-a` 指定したデバイスのポート `a` をバイパスします。

`-b` 指定したデバイスのポート `b` をバイパスします。

`-e dump_map` *fibre_channel_HBA_port*

指定したファイバチャネルポート上にあるターゲットデバイスまたはホストバスアダプタの WWN データを表示します。指定したポートにターゲットデバイスが存在しない場合、エラーが返されます。

`-e enable [-ab]` *enclosure,dev*
`-e enable -f` *enclosure*

エンクロージャサービスコントローラに対して、指定したポートおよびデバイスの LRC (ループ冗長回路) を有効な状態に設定するよう要求します。

このサブコマンドは、次のオプションをサポートします。

`-a` 指定したデバイスのポート `a` を有効にします。

`-b` 指定したデバイスのポート `b` を有効にします。

`-e forcelip` *enclosure*[*,dev*] . . . | *pathname* . . .

ループ初期化基本式 (LIP) の処理を使ってリンクを強制的に最初期化します。enclosure または pathname には、ループ上のどのデバイスでも指定することができます。複数のループ構成に対して特

定のパスを指定する場合は、パス名を使用します。

このコマンドは経験者向けのコマンドで、使用には注意が必要です。このコマンドは、ループ上のすべてのポートをリセットします。

`-e rdls` *enclosure*[,*dev*] . . . | *pathname* . . .
: enclosure または pathname で指定されたデバイスを含むループにある、すべての使用可能なデバイスのリンクエラー状態情報を読み込み、表示します。

その他のエキスパートモードサブコマンド

ホットプラグ操作中の制限に関しては「注意事項」を参照してください。次のサブコマンドは、経験が豊富なユーザーのみが使用することができます。

これらのコマンドは、Sun Fire 880 内蔵記憶装置サブシステムに使用することはできません。

`-e bus_getstate` *pathname*
: 指定されたバスの状態を取得、表示します。

`-e bus_quiesce` *pathname*
: 指定されたバスを休止します。

`-e bus_reset` *pathname*
: 指定されたバスのみをリセットします。

`-e bus_resetall` *pathname*
: 指定されたバスおよびすべてのデバイスをリセットします。

`-e bus_unquiesce` *pathname*
: 指定されたバスを休止解除します。指定されたデバイス。

`-e dev_getstate` *pathname*
: 指定されたデバイスの状態を取得、表示します。

`-e dev_reset` *pathname*
: 指定されたデバイスをリセットします。

`-e offline` *pathname*
: 指定されたデバイスをオフラインにします。

`-e online` *pathname*
: 指定されたデバイスをオンラインにします。

使用例

例 1　システム上にあるすべての Sun StorEdge A5000 および FC_AL の表示

システム上にあるすべての Sun StorEdge A5000 および FC_AL デバイスを検索、表示する例を次に示します。

```
example% luxadm probe
```

例 2　SENA または Sun Fire 880 内蔵記憶装置サブシステムの表示

SENA または Sun Fire 880 内蔵記憶装置サブシステムを表示する例を次に示します。

```
example% luxadm display /dev/es/ses0
```

例 3　2 つのサブシステムの表示

格納装置名を使って 2 つのサブシステムを表示する例を次に示します。

```
example% luxadm display BOB system1
```

例 4　最初のディスクに関する情報の表示

BOB という名前の格納装置の前面にある最初のディスクに関する情報を表示する例を次に示します。前面のディスクを指定する場合は f を指定します。背面のディスクを指定する場合は r を指定します。

```
example% luxadm display BOB,f0
```

例 5　Sun Fire 880 内蔵記憶装置サブシステムに関する情報の表示

Sun Fire 880 内蔵記憶装置サブシステムには、1 セットのディスクしかありません。この場合、そのスロットを指定する場合は *s* を指定します。

```
example% luxadm display BOB,s0
```

例 6　SENA ディスク、格納装置、または各 FC_AL デバイスに関する情報の表示

ポートの WWN が 2200002037001246 の Sun StorEdge A5000 ディスク、格納装置、または各 FC_AL デバイスに関する情報を表示する例を次に示します。

```
example% luxadm display 2200002037001246
```

例 7　サブコマンドとして一意に認識するための文字列

サブコマンドとして一意に認識できるだけの長さの文字列を使用する例を次に示します。

```
example% luxadm disp BOB
```

例 8　エラー情報の表示

格納装置 BOB があるループに関するエラー情報を表示する例を次に示します。

```
example% luxadm display -r BOB
```

例 9　インタフェースボードへの新しいファームウェアのダウンロード

格納装置 BOB のインタフェースボードに新しいファームウェアをダウンロードする例を次に示します (ダウンロードするファイルはデフォルトパスで指定されています)。

```
example% luxadm download -s BOB
```

例 10　SCSI 照会コマンドからの情報の表示

システム上の個々のディスクから SCSI 照会コマンドからの情報を表示する例を次に示します。サブコマンドとして一意に認識できる長さの文字列だけが使用されます。

```
example% luxadm inq /dev/rdsk/c?t?d?s2
```

例 11　ホットプラグによる取り付け

BOB,f1 という名前の格納装置の前面の最初のスロットに新しいドライブをホットプラグで取り付ける例を次に示します。

```
example% luxadm insert_device BOB,f0
```

SF880-1 という名前の Sun Fire 880 内蔵記憶装置サブシステムの最初のスロットに、新しいドライブをホットプラグで取り付ける例を次に示します。

```
example% luxadm insert_device SF880-1,s0
```

例 12　エキスパートサブコマンドの実行

エキスパートサブコマンドを実行する例を次に示します。このサブコマンドは、格納装置 BOB のあるループを強制的に初期化します。

```
example% luxadm -e forcelip BOB
```

例 13　エキスパートモードのホットプラグサブコマンドの使用

エキスパートモードのホットプラグサブコマンドを使用して SSA 上のディスクを削除する例を次に示します。ホットプラグ操作中の制限に関しては「注意事項」を参照してください。

最初の手順では、SCSI デバイスが 2 つ目の SCSI バスによってアクセスされないように、そのデバイスを予約します。

```
example# luxadm reserve /dev/rdsk/c1t8d0s2
```

例 14　ディスクをオフラインにする

次の 2 つの手順では、ディスクをオフラインにし、バスを休止します。

```
example# luxadm -e offline /dev/rdsk/c1t8d0s2
example# luxadm -e bus_quiesce /dev/rdsk/c1t8d0s2
```

例 15　バスの休止解除

ユーザーは、この時点でディスクを取り外し、バスを休止解除し、ディスクをオンラインに戻し、ディスクの予約を解除します。

```
example# luxadm -e bus_unquiesce /dev/rdsk/c1t8d0s2
example# luxadm -e online /dev/rdsk/c1t8d0s2
example# luxadm release /dev/rdsk/c1t8d0s2
```

環境

luxadm の実行に影響のある環境変数 LANG に関しては environ(5) を参照してください。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

–1　エラーが発生した。

ファイル

usr/lib/firmware/fc_s/fc_s_fcode

usr/lib/locale/C/LC_MESSAGES/ibfirmware

属性

次の属性については attributes(5) のマニュアルページを参照してください。

usr/sbin

属性タイプ	属性値
使用条件	SUNWluxop

関連項目

devlinks(1M), disks(1M), attributes(5), environ(5), ses(7D)

『Sun StorEdge A5000 設置・サービスマニュアル』(805-4111)

『RAID Manager 6.1 Installation and Support Guide Answerbook』

『RAID Manager 6.1 User's Guide Answerbook』

注意事項

Sun StorEdge A5000 に関するその他の情報については、『Sun StorEdge A5000 設置・サービスマニュアル』(805-4111) を参照してください。IEEE 拡張 WWW に関する情報については、『*Tutorial for SCSI use of IEEE company_ID*』(R. Snively 著) を参照してください。「関連項目」を参照してください。現在は、一部のデバイスドライバのみがホットプラグに対応しています。ホットプラグに対応していないディスクまたはバスにホットプラグが適用されると、メッセージが表示されます。

```
luxadm: can’t acquire "PATHNAME": No such file or directory
```

上記の形式で表示されます。

ルートファイルシステムまたは `/usr` ファイルシステムを含むバスやスワップデータを含むバスを休止する際は注意してください。そのようなバスを休止すると、デッドロックを引き起こす可能性があり、そのような場合は、システムの再起動が必要となります。

名前

m64config, SUNWm64_config – M64 グラフィックスアクセラレータの設定

形式

```
/usr/sbin/m64config [-defaults] [-depth 8 | 24 | 32]
     [-dev device-filename] [-file machine | system]
     [-prconf] [-propt]
     [-res video-mode [now | try] [noconfirm | nocheck ]]

/usr/sbin/m64config [-prconf] [-propt]

/usr/sbin/m64config [-help] [-res ?]
```

機能説明

`m64config` は、M64 グラフィックスアクセラレータおよび M64 対応の X11 ウィンドウシステムのデフォルトの一部を設定します。

形式の項に記された `m64config` の 1 番目の形式では、指定したオプションを OWconfig ファイルに保存します。これらのオプションは、次にウィンドウシステムをそのデバイスで実行するときに M64 デバイスを初期化するために使用されます。OWconfig ファイル内のオプションの更新は、異なるウィンドウセッションや再起動したシステムでも有効となります。

`-prconf`、`-propt`、`-help`、`-res ?` オプションだけを起動する 2 番目と 3 番目の形式では、OWconfig ファイルは更新されません。また、3 番目の形式では、その他のオプションはすべて無視されます。

オプションは、一度に 1 つの M64 デバイスに対してのみ指定することができます。複数の M64 デバイスに対してオプションを指定するには、`m64config` を複数回起動する必要があります。

`m64config` で指定できるのは、M64 固有のオプションだけです。デフォルトの表示色数、デフォルトの画像表示形式クラスなどを指定する通常のウィンドウシステムのオプションは、`openwin` コマンド行のデバイス修飾子で指定してください。詳細は、『OpenWindows Desktop Reference Manual』を参照してください。

ユーザーは、更新する OWconfig ファイルを指定することもできます。デフォルトでは、`/etc/openwin` ディレクトリツリーにあるマシン固有のファイルが更新されます。別のファイルを指定するには、`-file` オプションを使用します。たとえば、`/usr/openwin` ディレクトリツリーにあるシステム共通の OWconfig ファイルを代わりに更新することができます。

これらの標準 OWconfig ファイルのどちらもスーパーユーザーのみが書き込みを行えます。したがって、スーパーユーザーが所有する `m64config` プログラムは、`setuid` による root の権限で実行されます。

オプション

`-defaults`
すべてのオプションの値をそれぞれのデフォルト値に戻します。

`-depth 8 | 24 | 32`
表示色数をビット/ピクセルで指定します。指定可能な値は `8`、`24`、または `32` です (`32` を指定した場合、24 ビット/ピクセルが使用されます)。変更を反映させる

には、現在のウィンドウシステムのセッションからログアウトしてから再度ログインします。24 または 32 ビット/ピクセルでは、画面解像度を低くすることにより、ウィンドウシステムで TrueColor グラフィックスを実現できます。

サポートされている m64 デバイス上で 32 ビット/ピクセルを設定すると、8 ビットおよび 24 ビットカラーのウインドウが同時に使用できます。32 ビット/ピクセルに設定した場合、-propt オプションを指定してコマンドを実行すると depth に 32 が表示され、-prconf オプションを指定してコマンドを実行すると depth に 24 が表示されます。ウインドウの深さを調べるには、xwininfo ユーティリティを使用してください。xwininfo ユーティリティは通常、フレームバッファーソフトウェアを含むパッケージ (たとえば SUNWxwplt) に同梱されています。

24 ビット/ピクセルで利用できる解像度の最大値は、PGX カードに搭載されているメモリー量に依存します。2M バイトのメモリーを搭載した PGX カードで利用できる解像度の最大値は 800x600 です。4M バイトのメモリーを搭載したカードでは 1152x900 です。8M バイトのメモリーを搭載したカードでは 1920x1080 です。指定した解像度と色数の組み合わせに必要なメモリーが不足している場合、m64config はエラーメッセージを出力して終了します。

-dev *device-filename*
M64 特殊ファイルを指定します。指定しない場合、m64config はファイルが見つかるまで /dev/fbs/m640 から /dev/fbs/m648 を試行します。

-file machine| system
更新する OWconfig ファイルを指定します。machine を指定すると、/etc/openwin ディレクトリツリーにあるマシン固有の OWconfig ファイルが使用されます。system を指定すると、/usr/openwin ディレクトリツリーにある共通の OWcon-fig ファイルが使用されます。ファイルがない場合は、新たに生成されます。ほかのオプションを指定しない限り、このオプションは有効ではありません。デフォルトは *machine* です。

-help
m64config コマンド行のオプションと機能の概要を一覧で表示します。

-prconf
M64 ハードウェア構成を表示します。次に表示例を示します。

```
--- Hardware Configuration for /dev/fbs/m640 ---
ASIC: version 0x41004754
DAC: version 0x0
PROM: version 0x0
Card possible resolutions: 640x480x60, 800x600x75, 1024x768x60
    1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76
    1280x1024x60, 1152x900x66, 1152x900x76, 1280x1024x67
    960x680x112S, 960x680x108S, 640x480x60i, 768x575x50i
    1280x800x76, 1440x900x76, 1600x1000x66, 1600x1000x76
    vga, svga, 1152, 1280, stereo, ntsc, pal
```

```
Monitor possible resolutions: 720x400x70, 720x400x85, 640x480x60
    640x480x67, 640x480x72, 640x480x75, 800x600x56, 800x600x60
    800x600x72, 800x600x75, 832x624x75, 1024x768x85, 1024x768x60
    1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76,
     1152x900x66, 1152x900x76, 1280x1024x67, 960x680x112S
    vga, svga, 1152, 1280, stereo
Possible depths: 8, 24
Current resolution setting: 1280x1024x76
Current depth: 8
```

-propt
: -file オプションで指定された OWconfig ファイルに書かれた M64 オプションの値のうち、-dev オプションで指定されたデバイスに対するものすべてを表示します。m64config の呼び出しが終了した後に、OWconfig ファイルに書き込まれるオプションの値を表示します。次に表示例を示します。

```
--- OpenWindows Configuration for /dev/fbs/m640 ---
OWconfig: machine
Video Mode: not set
Depth: 8
```

-res *video-mode* [now | try [noconfirm | nocheck]]
: 指定した M64 デバイスに接続されているモニターを制御する際に使われる表示モードを指定します。表示モードはあらかじめ組み込まれています。表示モードの形式は *width*x-*height*x*rate* です。*width* はピクセル単位の画面幅、*height* はピクセル単位の画面の高さ、*rate* は画面を垂直方向に再描画する周期です。便宜上、-res にリフレッシュレートを指定する際は、値の直前に x の代わりに @ を使用することができます。たとえば、1280x1024@76 のように指定することができます。

  有効な表示モードのリストは、次のコマンドを実行することによって得ることができます。 m64config -res '?'. ? は引用符で囲んで文字として扱う必要があります。すべてのビデオボードおよびモニターがすべての解像度に対応しているわけではありません。m64config は、ボードが対応していない解像度に設定することを許しません。モニターが対応していない解像度に設定しようとすると、確認を促すメッセージを表示します。

  記号名

  便宜上、表示モードのいくつかには記号名が定義されています。*width*x*height*x*rate* の形式の代わりに、記号名を -res の引数として指定することができます。記号名 noneは、ウィンドウシステムを実行すると、画面の解像度は現在デバイスにプログラムされている表示モードになることを意味します。

記号名	対応する表示モード

svga	1024x768x60
1152	1152x900x76
1280	1280x1024x76
none	(デバイスでプログラムされている表示モード)

-res オプションには、表示モードの直後に次の追加引数を指定することができます。追加引数は、単独でも複数でも指定することができます。

nocheck　このオプションを指定すると、モニターセンスコードに基づく通常のエラーチェックが行われません。ユーザーによって指定された表示モードは、現在接続されているモニターに適切かどうかにかかわらず受け付けられます。このオプションは、 M64 デバイスに異なるモニターを接続する場合に便利です。このオプションを指定すると、noconfirm も指定されます。

noconfirm　-res オプションを指定した際に、システムが使用できない状態になり、表示出力がなくなる場合があります。このような状況は、特定のコードが読み込まれた際のモニターセンスコードにあいまいさがあった場合などに発生します。このような事態を避けるために m64config のデフォルトの動作では、この問題についての警告メッセージと、処理を継続するかどうかを確認するメッセージを表示します。noconfirm オプションを指定すると、m64config コマンドはこの確認をせずに、要求のあった表示モードにプログラムします。このオプションは、m64config がシェルスクリプトから実行されている場合に便利です。

now　OWconfig ファイルの表示モードを更新するとともに、M64 デバイスが指定した表示モードにただちにプログラムされます (この機能は、ウィンドウシステムを開始する前に表示モードを変更する際に便利です)。

対象となるデバイスが稼働している間 (たとえば、ウィンドウシステムの稼働中) に、この追加オプションを m64config に指定することはお勧めしません。予期しない結果になることもあります。now オプションを指定して m64config コマンドを実行する場合は、最初にウィンドウシステムを終了してください。now オプションがウィンドウシステムのセッション中に使用された場合、表示モードはただちに変更されますが、画面の幅や高さはそのセッションが終了して次のセッションに入るまで変更されません。さらに、立体表示モードではシステムが変更を認識しないことがあります。したがって、ウィンドウシステムの稼働中には絶対に now オプションを指定しないでください。

`try` このオプションを指定すると、指定した表示モードが試験的にプログラムされます。ユーザーは、指定した表示モードを使用する場合は、メッセージが表示されてから 10 秒以内に y と入力します。表示されたモードを使用しない場合は、10 秒以内に任意の文字を入力します。y または Returnキー以外の文字の入力は、すべて「使用しない」とみなされ、以前の表示モードに戻され、OWcon-fig ファイル中の表示モードは書き換えられません (その他の指定されたオプションは有効となります)。Return キーの入力があった場合は、新しい表示モードを保持するかどうかを yes または no で確認するメッセージが表示されます。

構成済みデバイスを使用中に (たとえば、ウィンドウシステムを実行している場合)、try サブオプションを `m64config` に指定してはなりません。予期せぬ結果になることがあります。`try` サブオプションを指定して `m64config` を実行する前には、ウィンドウシステムを停止しておく必要があります。

デフォルト設定

`m64config` コマンド行で指定されていないオプションについては、対応する OWconfig ファイル中のオプションは更新されず、ファイル内の値がそのまま使用されます。

ウィンドウシステムを実行する際に、`m64config` による M64 オプションの指定がまったくなかった場合は、デフォルト値が使用されます。オプションのデフォルトを以下に示します。

オプション	デフォルト値
`-dev`	`/dev/fbs/m640`
`-file`	machine
`-res`	none

`-res` オプションのデフォルト値 none とは、ウィンドウシステムが実行された場合に、画面解像度がそのデバイスに現在プログラムされている表示モードになることを意味しています。

注 : これによって、PROM によってデバイスの解像度を指定しているユーザーとの共用性が保てます。(GX などの) 一部のデバイスでは、PROM が表示モードを指定する唯一の手段です。これは、デフォルトの M64 表示モードは、最終的に PROM によって決まることを意味しています。

使用例

例1 モニターの種類の変更

モニターの種類を、垂直周波数 76 Hz で解像度 1280x1024 に変更する例を以下に示します。

```
example% /usr/sbin/m64config -res 1280x1024x76
```

ファイル

/dev/fbs/m640	デバイス特殊ファイル
/etc/openwin/server/etc/OWconfig	システム設定ファイル(ファイルを作成または更新する)
/usr/lib/fbconfig/SUNWm64_config	usr/sbin/m64config へのシンボリックリンク

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWm64cf

関連項目

attributes(5), m64(7D)

『OpenWindows Desktop Reference Manual』

名前　mdlogd – Solaris ボリュームマネージャデーモン

形式　`mdlogd`

機能説明　`mdlogd` は、Solaris ボリュームマネージャによって書き込まれたメッセージを検出するためにシステムコンソールを監視する単純なデーモンを実装します。Solaris ボリュームマネージャのメッセージが検出されると、`mdlogd` は一般 SNMP トラップを送信します。

トラップを有効にするには、SNMP フレームワークに対して `mdlogd` を構成する必要があります。『Solaris ボリュームマネージャの管理』 を参照。

使用方法　`mdlogd` は以下の SNMP MIB を実装します。

```
SOLARIS-VOLUME-MGR-MIB DEFINITIONS ::= BEGIN
        IMPORTS
                enterprises FROM RFC1155-SMI
                DisplayString FROM SNMPv2-TC;

        -- Sun Private MIB for Solaris Volume Manager


        sun       OBJECT IDENTIFIER ::= { enterprises 42 }
        sunSVM       OBJECT IDENTIFIER ::= { sun 104 }

        -- this is actually just the string from /dev/log that
        -- matches the md: regular expressions.
        -- This is an interim SNMP trap generator to provide
        -- information until a more complete version is available.

        -- this definition is a formalization of the old
        -- Solaris DiskSuite mdlogd trap mib.

        svmOldTrapString OBJECT-TYPE
                        SYNTAX DisplayString (SIZE (0..255))
                        ACCESS read-only
                        STATUS mandatory
                        DESCRIPTION
                        "This is the matched string that
                         was obtained from /dev/log."
        ::= { sunSVM 1 }

        -- SVM Compatibility ( error trap )

        svmNotice        TrapTRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log trap for NOTICE.
```

```
                                This matches 'NOTICE: md:'"
        ::= 1

        svmWarningTrap  TRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log trap for WARNING..
                                 This matches 'WARNING: md:'"
        ::= 2

        svmPanicTrap    TRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log traps for PANIC..
                                This matches 'PANIC: md:'"
        ::= 3
END
```

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWlvma, SUNWlvmr
インタフェースの安定性	廃止

関連項目

snmpdx(1M), attributes(5)

『Solaris ボリュームマネージャの管理』

名前

mdmonitord – daemon to monitor metadevices

形式

`/usr/sbin/mdmonitord [-t` *time_interval*`]`

機能説明

The `mdmonitord` utility is part of Solaris Volume Manager. It monitors and checks RAID1 (mirrors), RAID5 and hot spares.

There are two methods for checking:

- At fixed time intervals.
- When a RAID-1 (mirror), RAID-5, or hot spare fails. A failure generates an error event which triggers a check of these metadevices.

オプション

The following options are supported:

`-t` Time interval in seconds. The default value is 0, which causes probes to occur only upon an error. If you want to run `mdmonitord` at a regular interval, a value of 1800 (seconds, every half hour) is recommended as a starting point.

終了ステータス

The following exit values are returned:

`0` Successful completion.

>0 An error occurred.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

関連項目

`svcs`(1), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), svcadm(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `md.tab`(4), `attributes`(5), `smf`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

注意事項

Since frequent probes can affect performance, it is recommended that the intervals between probes be limited.

The `mdmonitord` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/system/mdmonitor
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the `svcs`(1) command.

名前

metaclear – アクティブなメタデバイスとホットスペア集合を削除する

形式

```
/usr/sbin/metaclear  -h

/usr/sbin/metaclear  [-s setname] -a [-f]

/usr/sbin/metaclear  component

/usr/sbin/metaclear  [-s setname] [-f] metadevice...
 hot_spare_pool...

/usr/sbin/metaclear  [-s setname] -r [-f] metadevice...
 hot_spare_pool...

/usr/sbin/metaclear  [-s setname] -p component

/usr/sbin/metaclear  [-s setname] -p metadevice
```

機能説明

metaclear コマンドを使用すると、指定したメタデバイスまたは *hot_spare_pool* が削除されるか、または指定したコンポーネントからすべてのソフトパーティションが削除されます。メタデバイスやホットスペア集合を削除した後、再度使用するには metainit を使用して再生成する必要があります。

現在使用中の (オープンしている) メタデバイスは削除できません。

オプション

以下のオプションのうち、-h 以外のオプションを実行するには、スーパーユーザーになる必要があります。

-a
: -s により指定されているセット内、またはデフォルトではローカルセット内にある、すべてのメタデバイスと構成されたホットスペア集合を削除します。

-f
: エラー状態のサブコンポーネントが含まれているメタデバイスを (強制的に) 削除します。

-h
: 使用方法に関するメッセージを表示します。

-p
: 指定したメタデバイスまたはコンポーネントからすべてのソフトパーティションを削除します。

-r
: 指定したメタデバイスとホットスペア集合を再帰的に削除しますが、ほかの要素が依存しているメタデバイスは削除しません。

-s *setname*
: metaclear を実行するディスクセットの名前を指定します。このオプションを使用すると、指定したディスクセット内で metaclear コマンドが実行されます。このオプションを使用しない場合は、ローカルのメタデバイスやホットスペア集合に対して、metaclear コマンドが実行されます。

オペランド

metadevice...
: 削除されるメタデバイスの名前を指定します。

component
: 削除されるソフトパーティションを含むコンポーネントの c*d*t*s* 名を指定します。

hot_spare_pool ... 削除されるホットスペア集合の名前を hsp*nnn* という書式で指定します。この場合、*nnn* には、000 から 999 までの数字を指定します。

使用例

例 1 さまざまなデバイスの削除

以下の例では、d10 という名前のメタデバイスを削除します。

```
# metaclear /dev/md/dsk/d10
```

以下の例では、システム上のローカルメタデバイスとホットスペア集合をすべて削除します。

```
# metaclear -a
```

以下の例では、ミラー d20 と、エラー状態のサブミラーを削除します。

```
# metaclear -f d20
```

以下の例では、hsp001 という名前のホットスペア集合を削除します。

```
# metaclear hsp001
```

以下の例では、ソフトパーティション d23 を削除します。

```
# metaclear d23
```

以下の例では、ソフトパーティションがほかのメタデバイスによって使用されていない場合、または開いていない場合に、スライス c2t3d5s2 上のすべてのソフトパーティションを削除します。

```
# metaclear -p c2t3d5s2
```

以下の例では、メタデバイスからソフトパーティションを削除します。

```
# metaclear -p d2
d3: Soft Partition is cleared
d4: Soft Partition is cleared
d5: Soft Partition is cleared
```

終了ステータス

以下の終了ステータスが返されます。

0 正常終了

>0 エラーが発生した

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

名前

metadb – メタデバイス状態データベースの複製の作成および削除

形式

```
/sbin/metadb  -h

/sbin/metadb  [-s setname]

/sbin/metadb  [-s setname] -a [-f] [-k system-file] mddbnn

/sbin/metadb  [-s setname] -a [-f] [-k system-file]
 [-c number] [-l length] slice...

/sbin/metadb  [-s setname] -d [-f] [-k system-file] mddbnn

/sbin/metadb  [-s setname] -d [-f] [-k system-file] slice...

/sbin/metadb  [-s setname] -i

/sbin/metadb  [-s setname] -p [-k system-file] [mddb.cf-file]
```

機能説明

metadb コマンドは、メタデバイス状態データベースの複製を作成または削除します。状態データベースの複製は、専用スライス上、あるいはあとで単純メタデバイス (連結またはストライプ)、または RAID5 メタデバイスの一部になるスライス上に作成できます。状態データベースの複製は、ファブリック接続ストレージ、SAN、またはシステムに直接接続されていないほかのストレージに格納しないでください。複製は、ブートプロセス中、従来の SCSI ドライブまたは IDE ドライブと同じ時点で利用できるストレージに格納する必要があります。注意事項を参照

メタデバイス状態データベースには、システム上のすべてのメタデバイスとホットスペア集合の構成情報が含まれています。さらに、メタデバイス状態データベースは、メタデバイスとホットスペア集合、およびそのコンポーネントの現在の状態を保存し続けています。Solaris ボリュームマネージャは、構成や状態が変化すると、メタデバイス状態データベースを自動的に更新します。状態の変化の例としては、サブミラーで障害が発生する場合が挙げられます。構成の変化の例としては、新しいメタデバイスの作成が挙げられます。

メタデバイス状態データベースは、実際には、データベースを複製した複数のコピーの集合です。それぞれのコピーは複製と呼ばれ、正確さを保証するために厳密な一貫性検査が行われます。

複写されたデータベースには、どのデータベースが有効で正確なデータを持っているかを判断する点において、複写に伴った問題点があります。この問題点を解決するために、ボリュームマネージャは多数決アルゴリズムを使用しています。このアルゴリズムでは、データベース複製の過半数が利用可能な状態にないと、複製を有効と判断することができません。このアルゴリズムでは、少なくとも最初に 3 つの複製を作成することを強くお勧めします。3 つの複製があり、そのうち少なくとも 2 つが使用可能であれば合意は成立します。複製が 1 つしかない場合にシステムがクラッシュすると、メタデバイス構成データのすべてが失われる可能性があります。

多数決アルゴリズムは、ほとんどの最新のデータが実際には 1 つの複製にある場合でも、多数決で決まらなければ失敗するという点で保守的であるといえます。この

アプローチによって、どのようにして起きた失敗かに関わらず、古くなったデータを誤って使用することがなくなります。多数決アルゴリズムでは、システムが半数以上の複製で稼動しつづけていること、システムは複製の半数以上が使用できない状態であると混乱してしまうこと、システムは全複製の半数以上がないとリブートしないことを考慮に入れてください。

オプションを指定せずに metadb コマンドを実行すると、メタデバイス状態データベースの状態が簡単に出力されます。出力されるフラグフィールドについての説明を参照するには、metadb -i を実行してください。

新しく状態データベースを作成するには、 metadb コマンドに -a オプションと -f オプションを指定し、その後に複製を置くスライスを指定します。-a オプションによって、(この場合は初期の) 状態データベースの複製が作成されます。-f オプションによって、状態データベースが存在しない場合は作成されます。(状態データベースが存在しない場合だけ、-a オプションと -f オプションを同時に指定してください。)

初期に作成された複製に加えて、追加の複製をシステムに加えることもできます。追加の複製は既存の複製と同じ情報を持ち、構成情報が失われるのを防ぎます。構成情報が失われると、メタデバイスが動作しなくなります。追加の複製を作成するには、metadb -a コマンドの後に複製を置くスライス名を指定します。同じスライスに置く複製はすべて同時に作成してください。

同一のスライス上にあるすべての複製を削除するには、metadb -d コマンドの後にスライス名を指定します。

metadb -i を実行すると、メタデバイス状態データベースの状態が表示されます。ハードウェア障害が発生したときや、状態データベースが追加または削除されたときに、状態データベースの状態が変わります。

エラー状態にある複製を修正するには、いったんその複製を削除してから追加し直します。

メタデバイス状態データベース (mddb) には、該当するディスクセット (ローカルディスクセットまたは共有ディスクセット) の複製の位置の一覧も含まれています。

ローカルセットの mddb には、該当するノードが属する各共有ディスクセットのホスト情報とドライブ情報も含まれている場合があります。ローカルセットの mddb に保存されているディスクセットのホスト情報とドライブ情報を除くと、ローカルディスクセットの mddb と共有ディスクセットの mddb は機能的に同じです。

mddb が書き込まれるのは、ミラーの再同期中、コンポーネントの障害発生中、あるいは構成の変更中です。1 つの複製に構成の変更または障害が発生する可能性もあり (mddb の削除またはディスクの障害)、ほかの複製はこの障害情報により更新されます。

オプション

-h と -i 以外のオプションを実行するには、スーパーユーザー になる必要があります。

metadb コマンドで使用できるオプションを以下に示します。ただし、すべてのオプションが同じコマンド行で使用できるとは限りません。サポートされているオプションの使用方法については、「形式」を参照してください。

-a
新しいデータベースデバイスを追加します。/kernel/drv/md.conf ファイルが新しい情報に自動的に更新され、/etc/lvm/mddb.cf ファイルも更新されます。複製を /etc/lvm/md.tab ファイルに定義し、コマンド行で、mddb*nn* という形式で名前を指定する方法でも複製を作成すること ができます (*nn* には、/etc/lvm/md.tab ファイルで定義した 2 桁の番号を指定します)。/etc/lvm/md.tab ファイルで複製を設定する方法については、md.tab(4) マニュアルページを参照してください。

-c *number*
各デバイスに置く複製の数を指定します。デフォルト値は 1 です。

-d
指定されたスライス *slice* 上にある複製をすべて削除します。/kernel/drv/md.conf ファイルが新しい情報に自動的に更新され、/etc/lvm/mddb.cf ファイルも更新されます。

-f
最初に状態ファイルを作成するときに使用します。このオプションは、強制的に全ての複製を削除する (削除により最低限必要な個数である 1 個を下回る数にする) ときにも使用します。(-a オプションと -f オプションは、状態データベースが存在しないときにだけ同時に指定できます。)

-h
使用方法に関するメッセージを表示します。

-i
複製の状態を表示します。-i オプションの出力では、デバイス名の前に状態データベースの状態を示す文字を表示します。次に、複製の状態に続いて表示される文字の意味を示します。

d　この複製には、デバイス ID が対応付けられていません

o　この複製は、mddb 構成の最後の変更より前にアクティブでした

u　この複製は最新です

l　この複製の位置情報は正常に読み込まれました

c　この複製の位置は /etc/lvm/mddb.cf に記述されていました

p　この複製の位置はカーネルでパッチされました

m　この複製はマスターです。つまり、これは入力として選択された複製です

r　この複製には、デバイス再配置情報がありません

t　この複製には、タグ付きデータが関連付けられています

W　この複製で、デバイス書き込みエラーが発生しました

a　この複製はアクティブです。コミットが行われています

M　この複製には、マスターブロックに問題がありました

D　この複製には、データブロックに問題がありました

F　この複製には、フォーマットに問題がありました

S　この複製は小さすぎて、現在のデータベースを格納できません

R　この複製で、デバイス読み取りエラーが発生しました

B　この複製に関連付けられているタグ付きデータは無効です

-k *system-file*　複製情報を書き込むカーネルファイルの名前を指定します。デフォルトの *system-file* は、`/kernel/drv/md.conf` です。このオプションは、ローカルディスクセットに対してのみ使用します。

-l *length*　各複製のサイズを指定します。デフォルトの *length* は 8192 ブロックであり、ほとんどのデバイス構成に適しています。128 ブロックより小さい複製のサイズを指定することお勧めしません。

-p　`/etc/lvm/mddb.cf` ファイルの情報に基づき、システムファイル (`/kernel/drv/md.conf`) を更新します。このオプションは通常、新しく作成したシステムファイルをはじめて起動する前に更新するために使用します。実行するシステム以外で新しいシステムが作成された場合は、引数にローカルマシン上の `mddb.cf` の位置を指定します。更新対象となるシステムファイルは、**-k** オプションで変更できます。このオプションは、ローカルディスクセットに対してのみ使用します。

-s *setname*　`metadb` コマンドを実行するディスクセットの名前を指定します。このオプションを使用すると、指定したディスクセット内で `metadb` が実行されます。このオプションを指定しない場合は、ローカルデータベースの複製に対して `metadb` が実行されます。

slice　`/dev/dsk/c0t0d0s3` などの物理スライス (パーティション) の論理名を指定します。

使用例

例 1　状態データベースの複製のはじめての作成

以下は、新しいシステムにはじめて状態データベースの複製を作成する例です。

```
# metadb -a -f c0t0d0s7 c0t1d0s3 c1t0d0s7 c1t1d0s3
```

例 1　状態データベースの複製のはじめての作成　　(続き)

-a オプションと -f オプションで、最初のデータベースと複製を強制的に作成します。これによって、システムの有効利用をするために、これらと同じスライスを使用してメタデバイスを作成できます。

例 2　2 つの新しいディスクへの 2 つの複製の追加

この例は、現在ボリュームマネージャが稼動しているシステムに接続された 2 つの新しいディスク上に 2 つの複製を追加する方法を示しています。

```
# metadb -a c0t2d0s3 c1t1d0s3
```

例 3　2 つの複製の削除

この例は、システムから 2 つの複製を削除する方法を示しています。この複製は、/dev/dsk/c0t2d0s3 上と /dev/dsk/c1t1d0s3 上に設定されていたものとします。

```
# metadb -d c0t2d0s3 c1t1d0s3
```

複製をすべて削除することもできますが、メタデバイスが存在している場合はすべてを削除しないでください。すべての複製を削除すると、既存のメタデバイスが使用不能になります。

ファイル

/etc/lvm/mddb.cf	メタデバイス状態データベースの複製の位置を記録するファイル
/etc/lvm/md.tab	メタデバイスデータベースを設定する作業領域ファイル
/kernel/drv/md.conf	システムのすべてのメタデバイスのデータベース複製情報。Solaris ボリュームマネージャ構成情報も含まれています。

終了ステータス

以下の終了ステータスが返されます。

0　　正常終了

>0　　エラーが発生した

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdr

関連項目

mdmonitord(1M), metaclear(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M),

metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

注意事項

複製は、ファブリック接続ストレージ、SAN、またはシステムに直接接続されていないほかのストレージに格納することはできません。複製は、ブートプロセス中、従来の SCSI ドライブまたは IDE ドライブと同じ時点で利用できるストレージに格納する必要があります。複製は以下の場所に格納できます:

- 専用のローカルディスクパーティション
- ボリュームの一部になるローカルパーティション
- UFS ロギングデバイスの一部になるローカルパーティション

名前

metahs – ホットスペアとホットスペア集合を管理する

形式

```
/usr/sbin/metahs  [-s setname] -a all component

/usr/sbin/metahs  [-s setname] -a hot_spare_pool [component]

/usr/sbin/metahs  [-s setname] -d hot_spare_pool [component]

/usr/sbin/metahs  [-s setname] -d all component

/usr/sbin/metahs  [-s setname] -e component

/usr/sbin/metahs  [-s setname] -r hot_spare_pool component-old

/usr/sbin/metahs  [-s setname] -r all component-old
 component-new

/usr/sbin/metahs  [-s setname] -i [hot_spare_pool]...
```

機能説明

metahs コマンドは、既存のホットスペアとホットスペア集合を管理します。このコマンドを使用して、ホットスペア集合の中にコンポーネント (スライス) を追加したり、削除、交換、もしくは使用可能状態にします。metainit コマンドと同様に、metahs コマンドも初期ホットスペア集合を生成することができます。しかし、metahs コマンドは、メタデバイスのコンポーネントの交換は行いません。コンポーネントの交換は、metareplace コマンドによって行います。

ホットスペアは、常に (使用可能、使用中、障害) のいずれかの状態になっています。「使用可能」のホットスペアは実行中であり、データを受け付けることができますが、現在書き込みや読み取りは行われていません。「使用中」のホットスペアは、現在書き込みや読み取りが行われています。「障害」のホットスペアは、停止しており、修理を行う必要があります。ホットスペアの状態を表示するには、metahs に -i オプションを指定して実行します。

Solaris 10 を 64 ビットカーネルで実行している場合、Solaris ボリュームマネージャは 1T バイトを超えるストレージデバイスと論理ボリューム (「大容量ボリューム」と呼ぶ) をサポートします (ホットスペアを含む)。

大容量ボリュームまたはホットスペアを持つシステムを 32 ビットの Solaris 10 カーネルでリブートした場合、この大型ボリュームは metastat 出力には表示されますが、アクセス、変更、または削除することはできません。新しい大容量ボリュームも作成できません。この状況では、大容量ボリューム上にあるボリュームまたはファイルシステムも同様に利用できません。大容量ボリュームを持つシステムを Solaris 10 より前のバージョンの Solaris でリブートした場合、Solaris ボリュームマネージャは起動しません。Solaris 10 より前のバージョンの Solaris オペレーティング環境下で Solaris ボリュームマネージャを実行する前には、すべての大容量ボリュームを削除しておく必要があります。

オプション

以下のオプションのうち、-i 以外のオプションを実行するには、スーパーユーザーになる必要があります。

以下のオプションがサポートされています。

-a all *component*	*component* をすべてのホットスペア集合に追加します。all は、大文字でも小文字でも構いません。
-a *hot_spare_pool* [*component*]	*component* を *hot_spare_pool* に追加します。*hot_spare_pool* が存在しない場合は、作成されます。
-d all *component*	すべてのホットスペア集合から *component* を削除します。ただし、「使用中」の *component* は削除できません。
-d *hot_spare_pool* [*component*]	*hot_spare_pool* が空であり、メタデバイスが参照していない場合は、その *hot_spare_pool* を削除します。*component* を指定した場合は、*hot_spare_pool* から *component* を削除します。ただし、「使用中」のホットスペアは削除できません。
-e *component*	*component* をホットスペアとして使用できるようにします。*component* が「障害」状態で修理が完了している場合は、*component* を使用可能にすることができます。
-i [*hot_spare_pool* . . .]	指定された *hot_spare_pool* または、指定がない場合はすべてのホットスペア集合の状態を表示します。
-r all *component-old component-new*	コンポーネントが含まれているすべてのホットスペア集合内の *component-old* を *component-new* に交換します。ただし、古いホットスペアが使用中の場合は、どのホットスペア集合のコンポーネントも交換することができません。
-r *hot_spare_pool component-old component-new*	*hot_spare_pool* 内の *component-old* を *component-new* に交換します。ただし、古いホットスペアが使用中の場合は、ホットスペア集合のコンポーネントに交換することはできません。

`-s` *setname*	`metahs` を実行するディスクセットの名前を指定します。このオプションを使用すると、指定したディスクセット内で `metahs` が実行されます。このオプションを指定しない場合は、ローカルのホットスペア集合に対して `metahs` が実行されます。

オペランド　以下のオペランドがサポートされています。

component	ディスクドライブ上にある物理的なスライス (パーティション) の論理名。たとえば `/dev/dsk/c0t0d0s2` などです。
hot_spare_pool	ホットスペア集合の名前を `hsp`*nnn* という書式で指定します。この場合、*nnn* には、000 から 999 までの数字を指定します。

使用例

例 1　ホットスペア集合へのホットスペアの追加

以下の例では、ホットスペア集合 `hsp003` にホットスペア `/dev/dsk/c0t0d0s7` を追加します。

```
# metahs -a hsp003 c0t0d0s7
```

ホットスペアが集合に追加されても、集合内に存在するホットスペアの順番は変わりません。新しいホットスペアは、既存のホットスペアリストの最後に追加されます。

例 2　現在定義されているすべてのホットスペア集合へのホットスペアの追加

この例では、現在定義されているホットスペア集合にホットスペアを追加します。

```
# metahs -a all c0t0d0s7
```

この例の中のキーワード `all` は、ホットスペア `/dev/dsk/c0t0d0s7` をすべてのホットスペア集合に追加することを意味します。

例 3　ホットスペアの削除

この例では、ホットスペア集合 `hsp003` からホットスペア `/dev/dsk/c0t0d0s7` を削除します。

```
# metahs -d hsp003 c0t0d0s7
```

ホットスペアを削除すると、集合に残されたホットスペアの位置は、新しい順番に変わります。たとえばこの例で、`/dev/dsk/c0t0d0s7` が 3 つあるホットスペアのうちの 2 番目に位置していたとすると、削除後は 3 番目のホットスペアが 2 番目の位置

例 3 ホットスペアの削除 *(続き)*

に移動します。

例 4 ポットスペアの交換

この例では、以前定義されたホットスペアを交換します。

```
# metahs -r hsp001 c0t1d0s0 c0t3d0s0
```

この例では、ホットスペア `/dev/dsk/c0t1d0s0` が `/dev/dsk/c0t3d0s0` と交換されます。ホットスペアの順番は変わりません。

終了ステータス

以下の終了ステータスが返されます。

`0` 正常終了

`>0` エラーが発生した

属性

以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

警告

32 ビットカーネルの Solaris オペレーティング環境を実行する予定の場合、あるいは、Solaris 10 より前のバージョンの Solaris オペレーティング環境を使用する予定の場合、大容量ボリューム (つまり、1T バイトを超えるボリューム) を作成してはいけません。

名前　　metainit – メタデバイスを構成する

形式

```
/sbin/metainit  -h

/sbin/metainit  [generic options] concat/stripe numstripes
 width component... [-i interlace]

/sbin/metainit  [width component... [-i interlace]]
 [-h hot_spare_pool]

/sbin/metainit  [generic options] mirror -m submirror
 [read_options] [write_options] [pass_num]

/sbin/metainit  [generic options] RAID -r component...
 [-i interlace] [-h hot_spare_pool] [-k] [-o original_column_count]

/sbin/metainit  [generic options] hot_spare_pool
 [hotspare...]

/sbin/metainit  [generic options] metadevice-name

/sbin/metainit  [generic options] -a

/sbin/metainit  [generic options] softpart -p [-e]
 component [-A alignment] size

/sbin/metainit  -r
```

機能説明　　`metainit` コマンドは、コマンド行で指定された情報に従って、メタデバイスとホットスペアを構成します。あるいは、`metainit` を実行して `/etc/lvm/md.tab` ファイルで指定した構成エントリを使用することもできます (`md.tab`(4) 参照)。すべてのメタデバイスは、使用前に `metainit` コマンドで構成しておく必要があります。

システムが 64 ビットの Solaris カーネルを実行している場合、Solaris ボリュームマネージャは 1T バイトを超えるストレージデバイスと論理ボリューム (「大型ボリューム」と呼ぶ) をサポートします。大型ボリュームのサポートは自動です。1T バイトを超えるデバイスを作成した場合、Solaris ボリュームマネージャは、ユーザーの操作なしで、そのデバイスを適切に構成します。

大型ボリュームを持つシステムを 32 ビットの Solaris カーネルでリブートした場合、この大型ボリュームは `metastat` 出力に表示されます。大型ボリュームは、アクセス、変更、または削除することはできません。新しい大型ボリュームも作成できません。この状況では、大型ボリューム上にあるすべてのボリュームまたはファイルシステムも同様に利用できません。大型ボリュームを持つシステムを Solaris 9 4/03 リリースより前のバージョンの Solaris でリブートした場合、Solaris ボリュームマネージャが起動しません。これより前のバージョンの Solaris オペレーティングシステム下で Solaris ボリュームマネージャを実行する前には、すべての大型ボリュームを削除しておく必要があります。

`/etc/lvm/md.tab` ファイルを編集してメタデバイスを構成する場合には、各行で 1 つの完結した構成エントリを指定します。次に、`-a` オプションを指定して

/etc/lvm/md.tab ファイルに入力したすべてのメタデバイスを有効にするか、または特定の構成エントリに対応するメタデバイス名を指定して metainit コマンドを実行します。

-a フラグと -n フラグの両方を付けて metainit を実行した場合、metainit は作成されていたはずのボリュームの状態を保持しません。md.tab にあるボリュームが md.tab にあるほかのボリュームに依存する場合、metainit -a -n を実行すると、それらのボリュームはすべてエラーとして報告されます。ただし、metainit -a の実行では、成功する可能性があります。md.tab(4) を参照してください。

Solaris ボリュームマネージャは /etc/lvm/md.tab ファイルを更新しません。完全な構成情報はメタデバイス状態データベースに格納されており、md.tab には格納されていません。md.tab を手入力で編集したときにだけ、情報がこのファイルに入ります。

ディスクミラーを設定するとき、最初の手順は、metainit を使用して、ルートスライスに一対一の連結を作成することです。使用例を参照。

オプション

次のオプションを指定できます。

一般オプション

-h 以外のオプションを実行するには、スーパーユーザーになる必要があります。

以下の汎用オプションがサポートされています。

-f
: スライスの 1 つがマウントされたファイルシステムまたはスワップとして使用されている場合でも、あるいは、作成しようとしているストライプのサイズがその基になるソフトパーティションのサイズより小さい場合でも、metainit コマンドの実行を強制的に続けます。このオプションは、ルート (/)、swap、/usr でミラーを構成するときに使用します。

-h
: 使用方法に関するメッセージを表示します。

-n
: メタデバイスを実際に設定せずに、コマンド行や md.tab エントリの構文を検査します。-a と同時に指定すると、すべてのデバイスが検査されますが、初期化は行われません。

-r
: ブート時にシェルスクリプト内でだけ使用されます。システム障害の発生前またはシャットダウン前に構成されたすべてのメタデバイスを再構成します。すでに構成されたメタデバイスの情報は、メタデバイス状態データベースに保存されています (metadb(1M) を参照)。

-s *setname*
: metainit を実行するディスクセットの名前を指定します。このオプションを指定しない場合は、ローカルのメタデバイスやホットスペアに対して metainit コマンドが作用します。

連結/ストライプ方式のオプション

以下に、サポートされる連結/ストライプ方式のオプションを示します。

concat/stripe 定義される連結、ストライプ、連結のストライプのメタデバイス名を指定します。

numstripes メタデバイス内の個別のストライプの数を指定します。単一のストライプでは、*numstripes* は常に 1 になります。連結では、*numstripes* はスライスと同じ数になります。連結ストライプでは、*numstripes* はストライプの数によって変化します。

width ストライプを構成するスライスの数を指定します。*width* が 1 よりも大きいときには、スライスはストライプされます。

component `/dev/dsk/c0t0d0s0` などのディスクドライブ上の物理スライス (パーティション) 用の論理名です。RAID レベル 5 メタデバイスでは、スライス間でパリティー情報のストライプ化を可能にするためには少なくとも 3 つのスライスが必要です。

`-i` *interlace* 飛び越しサイズを指定します。この値で、Solaris ボリュームマネージャに対してストライプされたメタデバイスまたは RAID レベル 5 メタデバイスにどの程度の量のデータを配置するかを指定して、次のスライスに移動します。*interlace* は、後ろに「`k` (K バイト)」、「`m` (M バイト)」、「`b` (ブロック)」の単位を付けて値を指定します。単位を表す文字は大文字でも小文字でもかまいません。*interlace* は 16 ブロック以上または 100M バイト以内にします。*interlace* を指定しないと、そのデフォルトは 16 K バイトになります。

`-h` *hot_spare_pool* メタデバイスに対応させる *hot_spare_pool* を指定します。コマンド行を使用する場合、ホットスペア集合は、`metainit` コマンドで事前に作成してから、メタデバイスに対応させる必要があります。*hot_spare_pool* は `hsp`*nnn* の形式で指定します。このとき nnn は 000 ～ 999 の範囲の数にします。作成される連結方式がサブミラーとして使用されるときには `/-h hsp`*nnn* を使用します。

ミラーオプション 以下に、サポートされるミラーのオプションを示します。

mirror `-m` *submirror* ミラーのメタデバイス名を指定します。`-m` は構成がミラーであることを指定します。*submirror* ははじめて作成する 1 面ミラーを形成するメタデバイス (ストライプ方式または単純連結方式) です。Solaris ボリュームマネージャは最大 4 面のミラー化をサポートします。ミラーを定義するときには、まず、`metainit` コマンドで 1 面ミラーとしてミラーを作成します。次に `metattach` コマンドを使用して残りのサブミラーを追加します。この方法によって、Solaris ボリュームマネージャは適切にミラーの同期を取ることができます。先に、`metainit` コマンドで 2 番目以降のサブミラーを作成します。

read_options 以下に、サポートされるミラーの読み取りのオプションを示します。

`-g` 幾何学的読み取りオプションを有効にします。これによって、順次読み取りのパフォーマンスが速くなります。

`-r` すべての読み取りを最初のサブミラーに対して行います。これは、最初のサブミラーを構成するデバイスが 2 番目のミラーのデバイスよりも十分に高速であるときにだけ使用します。このフラグは `-g` フラグと同時には使用できません。

`-g` フラグも `-r` フラグも指定されていない場合、読み取りはミラー内のすべてのサブミラーからラウンドロビン (巡回的) になります。これによって、サブミラー間の負荷のバランスがとれます。

write_options ミラーに対して以下の書き込みオプションがサポートされています。

`-S` ミラーに対して逐次書き込みを行います。最初のサブミラーへの書き込みが完了してから 2 番目のミラーへの書き込みが行われます。これは、部分的なセクター障害がハードウェアで許容されている場合には便利です。`-S` が指定されない場合、書き込みはすべてのミラーに同時に複写され、ディスパッチされます。

pass_num エントリの最後で 0 ～ 9 の範囲で指定する番号です。これによってリブート中にミラーが再同期を取る順序が決定されます。デフォルト値は 1 です。小さい方のパス番号が最初に再同期を取られます。同じパス番号の場合は、同時に実行されます。0 が使用された場合には、再同期はスキップされます。0 は読み取り専用または `swap` としてマウントされたミラーだけに使用します。

RAID レベル 5 オプション 以下に、使用可能な RAID レベル 5 のオプションを示します。

RAID `-r` RAID レベル 5 メタデバイスの名前を指定します。`-r` は構成が RAID レベル 5 であることを指定します。

`-k` RAID レベル 5 メタデバイスにおいて、すでにデータがあるので初期化 (ディスクブロックをゼロにする) しないようドライバに通知します。このオプションは、以前に作成された RAID レベル 5 デバイスを再作成するときにだけ使用します。

-k オプションを使用するときには十分に注意してください。このオプションは、ディスクブロックを OK 状態に設定します。したがって、メタデバイス内のディスクブロックに何らかのエラーが存在しても、Solaris ボリュームマネージャはデータの作成を開始する可能性があります。-k オプションを使用する代わりに、デバイスを初期化して、テープからデータを復元してもかまいません。

-o *original_column_count*

RAID レベル 5 メタデバイスで、-k オプションと共に使用して、初めに定義されたメタデバイスが拡張された場合に拡張前のスライス数を定義します。パリティーセグメントは連結されたデバイス間ではストライプされないので、これが必要になります。

-o オプションを使用するときには十分に注意してください。このオプションはディスクブロックを OK 状態に設定します。メタデバイス内のディスクブロックにエラーが存在しても、Solaris ボリュームマネージャはデータの作成を開始する可能性があります。-o オプションを使用する代わりに、デバイスを初期化し、テープからデータを復元することもできます。

ソフトパーティションオプション

以下に、サポートされるソフトパーティションのオプションを示します。

softpart -p [-e] *component* [-A *alignment*] *size*

softpart 引数は、ソフトパーティションの名前を指定します。-p は、その構成がソフトパーティションであることを指定します。

-e は、*component* で指定したディスク (c*t*d*) 全体のパーティションを再分割して、ソフトパーティション用に予約するべきことを指定します。指定したコンポーネントでは、スライス 7 にシステム (状態データベースの複製) 用の領域が予約され、スライス 0 にディスクの残りのすべての領域が割り当てられます。スライス 7 の最小のサイズは 4 M バイトですが、ディスクのジオメトリによっては大きくすることも可能です。新しく作成するソフトウェアパーティションは、デバイスのスライス 0 に置かれます。

component 引数は、ソフトウェアパーティションを作成するディスク (`c*t*d*`)、スライス (`c*t*d*s*`)、またはメタデバイス (`d*`) を指定します。*size* 引数は、ソフトパーティションに使用する領域を決定します。サイズの単位としては、K バイトには `K` または `k` を、M バイトには `M` または `m` を、G バイトには `G` または `g` を、T バイトには `T` または `t` (1T バイトが最大のサイズ) を、ブロック (セクター) には `B` または `b` を指定できます。すべての値は 2 のべき乗で表現されます。大文字と小文字のオプションは同じです。使用できる値は整数値だけです。

`-A` 整列オプションは、ソフトウェアパーティション境界整列の値を設定します。このオプションは、ソフトパーティションの開始オフセットを指定することが重要であるときに使用します。このオプションを指定すると、メタデバイスとそれを構成する物理デバイスの 2 つのアドレス空間の間で、同じデータ整列が行われます。たとえば、ハードウェアデバイスでチェックサムを実行している場合、そのデバイスへの入出力要求は Solaris ボリュームマネージャによって分割されるべきではありません。この場合、ハードウェア構成から取得した値を境界整列の値として使用します。このオプションをソフトウェア入出力負荷と組み合わせて使用すると、境界整列の値はアプリケーションの入出力負荷に対応します。これによって、入出力が不必要に分割されて、パフォーマンスに影響が出ることを防ぐことができます。

ソフトパーティションが、デバイス上の使用可能なすべての領域に渡って拡張されるように指定するには、サイズの代わりに、`all` というリテラルを指

定します。

ホットスペア集合オプション

以下に、サポートされるホットスペア集合のオプションを示します。

hot_spare_pool [*hotspare...*]

`metainit` コマンドの引数として使用すると、*hot_spare_pool* はホットスペア集合用の名前を定義し、*hotspare...* はその集合で利用可能にする物理スライスの論理名になります。*hot_spare_pool* は `hsp`*nnn* 形式の数値であり、*nnn* は 000 ～ 999 の範囲の数になります。

`md.tab` ファイルオプション

以下に、サポートされる `md.tab` ファイルのオプションを示します。

metadevice-name

`metainit` コマンドが *metadevice-name* だけを引数として実行されると、`/etc/lvm/md.tab` ファイルを検索してその名前と対応するエントリを見つけます。`md.tab` ファイルで見つかるエントリの順序は重要ではありません。たとえば、以下の `md.tab` エントリの場合を考えてみます。

```
d0 2 1 c1t0d0s0 1 c2t1d0s0
```

コマンド `metainit d0` を実行すると、`md.tab` ファイルにある構成情報をもとにメタデバイス `d0` が構成されます。

`-a`

`md.tab` ファイルに定義されたすべてのメタデバイスを有効にします。

`-a` フラグと `-n` フラグの両方を付けて `metainit` を実行した場合、`metainit` は作成されていたはずのボリュームの状態を保持しません。`md.tab` ファイルの最初の行でデバイス `d0` を作成して、後ろの行で `d0` の存在を仮定している場合、`metainit -a` を実行すると、成功する場合がありますが、`metainit -an` を実行すると、後ろの行は失敗します。

使用例

例 1　一対一の連結の作成

以下のコマンドは、ルートスライスに一対一の連結を作成します。ルートスライス、またはその他のマウント解除できないスライスのミラーを設定する場合、最初の手順はこのようなコマンドになります。ルート (`/`) などの既存のファイルシステムを持つボリュームを作成する場合、`-f` オプションが必要になります。

```
# metainit -f d1 1 1 c0t0d0s0
```

このコマンドは、ルートスライスを使用して、`d1` を一対一の連結にします。次に、以下のコマンドを入力します。

```
# metainit d0 -m d1
```

このコマンドは、ルートスライスの 1 面ミラーを作成します。

例2 連結

以下の例で示されるドライブは、すべて同じサイズ (525 M バイト) です。

この例では、4 つのスライスの連結であるメタデバイス /dev/md/dsk/d7 を示します。

```
# metainit d7 4 1 c0t1d0s0 1 c0t2d0s0 1 c0t3d0s0 1 /dev/dsk/c0t4d0s0
```

数値 4 は、この連結に 4 つの独立したストライプがあることを示しています。各ストライプは 1 つのスライスで構成されているので、各スライスの最初に数値 1 が示されています。上記の各デバイスの最初のディスクセクターには、ディスクラベルが含まれています。デバイス /dev/dsk/c0t2d0s0、/dev/dsk/c0t3d0s0、/dev/dsk/c0t4d0s0 のラベルを保存するために、連結の境界を越えてアクセスを割り当てるときには、メタディスクドライバがこれらのディスクの少なくとも最初のセクターをスキップする必要があります。最初のセクターだけをスキップすると、ディスクのジオメトリが不規則になるので、ディスクの最初のシリンダ全体がスキップされます。このため、上位のファイルシステムソフトウェアによってブロックの割り当てが最適化できます。

例3 ストライプ

この例では、2 つのスライスで構成されるメタデバイス/dev/md/dsk/d15 を示します。

```
# metainit d15 1 2 c0t1d0s0 c0t2d0s0 -i 32k
```

数値 1 は、1 つのストライプが作成されることを示しています。このストライプは 2 つのスライスで構成されるため、その後に数値 2 が続きます。オプションの -i の後ろには 32k が指定され、飛び越しサイズが 32 K バイトになることが示されます。飛び越しサイズが指定されなかった場合には、ストライプはデフォルト値の 16 K バイトを使用します。

例4 連結ストライプ

この例は、3 台のディスクの 2 つのストライプの連結で構成されるメタデバイス /dev/md/dsk/d75 を示しています。

```
# metainit d75 2 3 c0t1d0s0 c0t2d0s0 \
      c0t3d0s0 -i 16k \
      3 c1t1d0s0 c1t2d0s0 c1t3d0s0 -i 32k
```

最初の行では、-i とその後ろの 16k でストライプの飛び越し サイズが 16 K バイトであることを指定しています。2 番目の組み合わせは、ストライプの飛び越しサイズが 32 K バイトになることを指定しています。2 番目の組み合わせが 32 K バイトを

例 4 連結ストライプ (続き)

指定していなければ、この組み合わせはデフォルトの飛び越し値 16 K バイトを使用します。3 台のディスクの各組み合わせのブロックは、3 台のディスクにまたがって飛び越しされます。

例 5 ミラー化

この例は、2 つのサブミラーで構成される 2 面ミラー /dev/md/dsk/d50 を示しています。このミラーには既存のデータは含まれていません。

```
# metainit d51 1 1 c0t1d0s0
# metainit d52 1 1 c0t2d0s0
# metainit d50 -m d51
# metattach d50 d52
```

この例では、2 つのサブミラーである d51 と d52 は metainit コマンドで作成されます。これら 2 つのサブミラーは単純連結方式です。次に 1 面ミラーである d50 は d51 を指定して -m オプションで作成されます。2 番目のサブミラーは後で metattach コマンドを使用して追加されます。ミラーを作成するときには、ストライプ方式と連結方式の任意の組み合わせが使用できます。この例のデフォルトの読み取りオプションと書き込みオプションは、ラウンドロビン (巡回的) 読み取りアルゴリズムとすべてのサブミラーに対する並列書き込みです。

例 6 ディスクセット内のメタデバイスの作成

この例は、set1 というディスクセット内の 2 つのストライプの連結から構成されるメタデバイス /dev/md/dsk/d75 を示しています。

```
# metainit -s set1 d75 2 3 c2t1d0s0 c2t2d0s0 \
      c2t3d0s0 -i 32k
# metainit -s set1 d51 1 1 c2t1d0s0
# metainit -s set1 d52 1 1 c3t1d0s0
# metainit -s set1 d50 -m d51
# metattach -s set1 d50 d52
```

この例では、まず、metaset コマンドを使用してディスクセットを作成します。次に、metainit コマンドを使用して、そのディスクセット内にメタデバイスを作成します。2 つのサブミラー d51 と d52 は単純連結です。次に、d51 を指定した -m オプションを使用して、1 面ミラー d50 を作成します。次に、metattach コマンドを使用して、2 番目のサブミラーを接続します。ミラーを作成するとき、ストライプと連結の任意の組み合わせを使用できます。この例では、デフォルトの読み取りオプションと書き込みオプションはそれぞれ、ラウンドロビン方式の読み取りアルゴリズムとすべてのサブミラーへの並行書き込みです。

例 7　RAID レベル 5

この例は、3 つのスライスで構成される RAID レベル 5 デバイス d80 を示しています。

```
# metainit d80 -r c1t0d0s0 c1t1d0s0 c1t3d0s0 -i 20k
```

この例で、RAID レベル 5 メタデバイスは -r オプションによって飛び越しサイズ 20 K バイトで定義されます。データとパリティーセグメントは、c1t0d0s0、c1t2d0s0、および c1t3d0s0 のスライスにまたがってストライプ化されます。

例 8　ソフトパーティション

以下の例は、メタデバイス d100 に構築される、サイズが 100 M バイト (「100M」で示される) のソフトパーティションデバイス d1 を示しています。

```
# metainit d1 -p d100 100M
```

このコマンドは、100 M バイトのソフトパーティションをメタデバイス d100 に作成します。このメタデバイスは、RAID レベル 5、ストライプ、連結、またはミラーのいずれでもかまいません。

例 9　ディスク全体を占めるソフトパーティション

以下の例は、ディスク c3t4d0 に構築されるソフトパーティションデバイス d1 を示しています。

```
# metainit d1 -p -e c3t4d0 9G
```

この例では、このディスクのパーティションを再分割して、ソフトパーティションが、スライス c3t4d0s0 上の使用可能な 9GB すべてのディスク領域を使用するように定義しています。

例 10　利用可能な領域をすべて割り当てたソフトパーティション

以下の例は、ディスク c3t4d0 に構築されるソフトパーティションデバイス d1 を示しています。

```
# metainit d1 -p -e c3t4d0 all
```

この例では、このディスクのパーティションを切り直して、ディスクスライス c3t4d0s0 で利用可能な領域をすべてソフトパーティション用に定義しています。

例 11　ホットスペア

この例では、2 面ミラー /dev/md/dsk/d10 と 3 つのホットスペアコンポーネントを持つホットスペア集合を示します。このミラーには既存のデータはありません。

```
# metainit hsp001 c2t2d0s0 c3t2d0s0 c1t2d0s0
# metainit d41 1 1 c1t0d0s0 -h hsp001
# metainit d42 1 1 c3t0d0s0 -h hsp001
# metainit d40 -m d41
# metattach d40 d42
```

この例では、ホットスペアとして使用される 3 つの異なるディスクから 3 つのスライスを使用して、ホットスペア集合 hsp001 を作成しています。次に、2 つのサブミラー d41 と d42 が作成されます。これらは単純連結方式です。metainit コマンドで -h オプションを指定してホットスペア集合 hsp001 と各サブミラーを関連付けます。次に、1 面ミラーを -m オプションを使用して定義します。2 番目のサブミラーを metattach コマンドを使用して追加します。

例 12　ソフトパーティション境界整列の値の設定

この例では、ソフトパーティションの境界整列の値を 1M バイトに設定する方法を示しています。

```
# metainit -s red d13 -p c1t3d0s4 -A 1m 4m
```

この例では、境界整列が 1M バイトのソフトパーティション d13 を作成しています。metainit コマンドの -A オプションに指定した「1m」は、1M バイトのソフトパーティション境界整列を定義しています。

ファイル

/etc/lvm/md.tab　バッチ方式で作成するためのメタデバイスとホットスペアの構成の一覧が格納されています。

警告

この節では、さまざまな警告について説明します。

1T バイトを超えるデバイスとボリューム

32 ビットカーネルの Solaris オペレーティングシステムを実行する予定の場合、あるいは、Solaris 10 より前のバージョンの Solaris オペレーティングシステムを使用する予定の場合、大型ボリューム (つまり、1T バイトを超えるボリューム) を作成してはいけません。

多面ミラー

metainit コマンドを使用して一度に多面ミラーを作成しないでください。そのときには、metainit コマンドで 1 面ミラーを作成した後、metattach で残りのサブミラーを追加していきます。metattach コマンドが使用されないと、再同期処理は実行されず、データが破壊される可能性があります。

metainit によって、複数のサブミラーを持つミラーを作成すると、以下のメッセージが表示されます。

警告: この形式の metainit はお勧めできません。
サブミラーは同じデータを持っていない可能性があります。
詳しい情報は metainit(1M) の「マニュアルページ」を参照してください。

ソフトパーティションの切り捨て

ソフトパーティションの一番上にストライプを作成するとき、新しいストライプのサイズがその元になるソフトパーティションのサイズよりも小さくなることがあります。この状況が発生すると、metainit は失敗して、この問題に必要な処置を示すエラーメッセージが表示されます。

-f オプションを使用してこの問題を無視しようとすると、以下のようなメッセージが表示されます。

警告: この形式の metainit はお勧めできません。
ストライプによって基本となるデバイスのサイズが切り詰められています。
詳細は、metainit(1M) の「エラー」を参照してください。

書き込み時の書き込み問題

Solaris ボリュームマネージャでデータをミラー化する場合、必ずしも、ミラー側すべてにおいて、メモリーからディスクへの転送がまったく同時に行われるわけではありません。データをディスクに転送している間にバッファーの内容が変更された場合 (「書き込み時の書き込み」と呼ぶ)、各ミラーで格納されるデータが異なる可能性があります。

この問題は、ミラー書き込み用にデータの専用コピーを作成することで処理できますが、このコピー作成は効率的ではありません。別の手段として、書き込み時にメモリーが変更されたことを、メモリーページに関連するダーティービットを見て検出するという方法があります。Solaris ボリュームマネージャは (可能なときは) このダーティービット手法を 使用します。残念ながら、この手法は raw 入出力や直接入出力では動作しません。デフォルトでは、Solaris ボリュームマネージャは次のように不利な点はありますが、パフォーマンスを優先して調整されています。つまり、raw 入出力や直接入出力に関連するバッファーにアプリケーションが「書き込み時の書き込み」を行なった場合、ミラー化されたデータについては同期を取りません。ミラー化を行わない場合、どのデータが最終的にメディアに存在したかは保証されません。しかし、複数の読み取りは同じデータを戻します。ミラー化を行う場合、複数の読み取りは異なるデータを戻す可能性があります。以下の行を /etc/system に追加すると、すべての raw 入出力と直接入出力の書き込み操作において、バッファーのコピーが安定します。

```
set md_mirror:md_mirror_wow_flg=0x20
```

このフラグを設定すると、パフォーマンスが低下します。

終了ステータス

次の終了ステータスが返されます。

`0` 正常終了。

`>0` エラーが発生した。

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdr

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

制限事項

ミラー化を再帰的に行うことはできません。つまり、ミラーを別のミラーの定義で使用することはできません。

また、ロギングを再帰的に行うこともできません。つまり、トランスメタデバイスを別のメタデバイスの定義で使用することはできません。

ストライプ、連結、および RAID レベル 5 のメタデバイスは、スライスだけで構成される必要があります。

RAID レベル 5 のメタデバイスをミラー化することはできません。

ソフトパーティションは、raw デバイスか、ストライプ、RAID レベル 5、またはミラーに構築できます。

RAID レベル 5 またはストライプのメタデバイスは、ソフトパーティションに直接構築できます。

注意事項

トランスメタデバイスは UFS ロギングによって置き換えられました。既存のトランスデバイスはロギングを行わず、その元となるデバイスにデータを直接渡します。UFS ロギングについての詳細は、mount_ufs(1M) を参照してください。

名前

metaoffline, metaonline – サブミラーの状態をオフラインおよびオンラインにする

形式

```
/usr/sbin/metaoffline  -h

/usr/sbin/metaoffline  [-s setname] [-f] mirror submirror

/usr/sbin/metaonline  -h

/usr/sbin/metaonline  [-s setname] mirror submirror
```

機能説明

metaoffline コマンドは、Solaris ボリュームマネージャによるオフライン状態のサブミラーへの読み取りと書き込みを阻止します。ミラーに対する書き込みは、サブミラーのオフライン時には領域単位で記録されるだけで、サブミラーがオンライン状態に戻った時に書き込まれます。metaoffline コマンドは、オンラインバックアップ時に利用することができます。サブミラーをオフライン状態にしてバックアップを実行している間でも、そのサブミラーを持つミラーはアクセス可能です。ただし、そのミラーが 2 面のミラーである場合、サブミラーがオフライン状態の間、データの冗長性は失われます。metaoffline コマンドは、サブミラーとミラー間の論理的関連付けを切り離さない点が、metadetach コマンドとは異なります。ミラーからサブミラーを完全に削除したい場合には、metadetach コマンドを使用します。

オフライン状態になったサブミラーは、metaonline コマンドを実行するかシステムをリブートするまで、オフライン状態のままです。

metaonline コマンドを実行すると、サブミラーに対する読み取りや書き込みが再開されます。サブミラーのオフライン時に書き込まれた領域に対して、自動的に再同期処理が行われます。書き込みは、再同期中にそのサブミラーに対して行われますが、読み取るデータは、別のサブミラーから取り込みます。再同期処理が完了すると、処理されたサブミラー上で読み取りと書き込みが行われます。metaonline コマンドは、オフライン状態になっているミラーのサブミラーに対してのみ有効です。

metaoffline コマンドと metaonline コマンドは、アプリケーションベース回復 (ABR) モードの RAID 1 ボリュームには使用できません。

metaofflineコマンドでオフライン状態になったサブミラーは、読み取り専用としてのみマウント可能です。

オプション

以下のオプションのうち、-h 以外のオプションを実行するには、スーパーユーザーになる必要があります。

-f　保守を行う必要のあるスライスが含まれているサブミラーを強制的にオフラインにします。

-h　使用方法に関するメッセージを表示します。

-s *setname*　metaoffline と metaonline を実行するディスクセットの名前を指定します。このオプションを使用すると、指定したディスクセット内で metaoffline が実行されます。このオプションを使用しないと、ローカルのメタデバイスに対して metaoffline が実行されます。

mirror オフラインまたはオンラインにしたいサブミラーを持つミラーのメタデバイス名を指定します。

submirror オフラインまたはオンラインにしたいサブミラーのメタデバイス名を指定します。

使用例

例1 サブミラーのオフライン化

この例では、ミラー `d10` にあるサブミラー `d9` をオフラインにします。

```
# metaoffline d10 d9
```

終了ステータス

以下の終了ステータスが返されます。

`0` 正常終了

`>0` エラーが発生した

属性

以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

注意事項

`metaonline` コマンドと `metaoffline` コマンドは、アプリケーションベース回復 (ABR) モードのミラーには適用できません。

名前

metaparam – メタデバイスのパラメータを変更する

形式

```
/usr/sbin/metaparam  -h

/usr/sbin/metaparam  [-s setname] [concat/stripe または
 RAID5 のオプション] concat/stripe RAID

/usr/sbin/metaparam  [-s setname] [mirror options]
 mirror
```

機能説明

metaparam コマンドは、メタデバイスの現在のパラメータを表示または変更します。

metaparam コマンドの引数としてメタデバイスのみを指定すると、現在の設定値が表示されます。

metaparam コマンドは、メタデバイス (ボリューム) のほとんどのパラメータを変更できます。metaparam コマンドで変更できない唯一のパラメータは飛び越し値です。飛び越し値はメタデバイスが作成されたときに確立し、そのあとでは変更できません。

オプション

以下のオプションを実行するには、スーパーユーザーになる必要があります。

以下のオプションがサポートされています。

-h　使用方法に関するメッセージを表示します。

-s *setname*　metaparam を実行するディスクセットの名前を指定します。このオプションを使用した場合は、指定したディスクセット内で metaparam が実行されます。このオプションを使用しない場合は、ローカルのメタデバイスに対して metaparam が実行されます。

連結方式または Raid5 オプション

–h *hot_spare_pool* | none　メタデバイスが使用するホットスペア集合を指定します。none を指定すると、メタデバイスは割り当てられているホットスペア集合との関係を解除します。メタデバイスが現在使用しているホットスペア集合を metaparam によって交換することはできません。

concat/stripe | *RAID*　連結方式、ストライプ方式、ストライプ方式による連結のメタデバイス名、または RAID5 メタデバイス名を指定します。

ミラーオプション

–r roundrobin | geometric | first　ミラーの読み取りオプションを変更します。-r の後ろには、roundrobin、geometric、first のいずれかを指定します。roundrobin は metainit コマンドのデフォルト動作であり、ラウンドロビン (巡回的) 方式でディスクから読み取ります。geometric は順次読み取りのパフォーマンス

を向上させます。first は最初のサブミラーだけから読み取ります。

–w parallel | serial	ミラーの書き込みオプションを変更します。-w の後ろには parallel または serial を指定します。parallel は metainit コマンドのデフォルト動作であり、すべての書き込みを並列処理で実行します。serial は書き込みを逐次処理で実行します。
-p *pass_number*	リブート時にミラーを再同期処理する順序 (0 ～ 9) を指定します。デフォルト値は 1 です。番号が小さいパスから先に再同期処理が行われます。0 を入力すると、ミラーの再同期処理が行われません。0 を指定するのは、読み取り専用もしくは swap としてマウントされたミラーだけにしてください。
mirror	ミラーのメタデバイス名を指定します。

使用例

例 1　ホットスペア集合と RAID5 メタデバイスの関連付け

この例では、ホットスペア集合 hsp005 と RAID5 メタデバイス d80 を関連付けます。

```
# metaparam -h hsp005 d80
```

例 2　読み取りオプションの geometric への変更

この例では、ミラー d50 の読み取りオプションを、デフォルトの roundrobin から geometric に変更します。

```
# metaparam -r geometric d50
```

終了ステータス

以下の終了ステータスが返されます。

0　　正常終了

>0　　エラーが発生した

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metarecover(1M), metarename(1M),

metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

名前 metarecover – recover soft partition information

形式

```
/sbin/metarecover  [-n] [-v] [-s setname] component -p

/sbin/metarecover  [-n] [-v] [-s setname] component -p
 {-d}

/sbin/metarecover  [-n] [-v] [-s setname] component -p
 {-m}
```

機能説明 The `metarecover` command scans a specified component to look for soft partition configuration information and to regenerate the configuration.

オプション The following options are supported:

`-d` Recover soft partitions in the metadevice state database from the extent headers on the device. Options `-d` and `-m` are mutually exclusive.

`-m` Regenerate the extent headers and reapplies them to the underlying device based on the soft partitions listed in the metadevice state database. Options `-d` and `-m` are mutually exclusive.

`-n` Do not actually perform the operation. Show the output or errors that would have resulted from the operation, had it been run.

`-p` Regenerate soft partitions based on the metadevice state database or extent headers on the underlying device. If neither `-d` nor `-m` are specified, this option compares the soft partition information in the metadevice state database to the extent headers.

`-s` *setname* Specify the name of the diskset on which `metarecover` works. Using the `s` option causes the command to perform its function within the specified diskset. Without the `-s` option, the `metarecover` command operates on the metadevices and/or hot spare pools in the local diskset.

This option is required to recover former `sps` from a diskset component or raw-device. *setname* must be identical to the former *setname* in which the `sps` were created. The set numbers, however, seem irrelevant.

`-v` Verbose mode, displaying the changes being made.

オペランド The following operand is supported:

component Specifies the `c*t*d*s*` number of the disk or slice containing the partitions, or the device name (for example, `d10`) of the metadevice containing the partitions.

component can be a slice name, component name, `/dev/dsk path`, or `/dev/rdsk path`.

使用例

例 1 Updating Metadevice State Database Based on Disk Extent Headers

A disk containing soft partitions is moved from one system to another. The system administrator would like to use the existing soft partitions. `metarecover` updates the metadevice state database based on the extent headers on the disk.

```
# metarecover -v c0t3d0s2 -p -d
```

例 2 Updating Metadevice State Database Based on Incomplete Soft Partition Creation

A system crashes in the middle of creating a new soft partition. The soft partition is in the creating state and the driver does not let that device be opened. `metarecover` rewrites the extent headers for the partially created soft partition and mark it as `Okay`.

```
# metarecover -v c0t3d0s2 -p -m
```

例 3 Updating Extent Headers Based on Metadevice State Database

Someone accidentally overwrote a portion of a disk leaving extent headers destroyed. `metarecover` rewrites the extent headers to ensure a valid soft partition configuration, though user data is not recovered.

```
# metarecover -v d5 -m
```

例 4 Validating Soft Partition Configuration

To validate the existing soft partition configuration, use `metarecover` with only the `-p` flag.

```
# metarecover c0t3d0s2 -p
```

終了ステータス

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdr

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `md.tab`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

名前

metarename – メタデバイス名を変更したり、階層化されたメタデバイス名を交換する

形式

```
/usr/sbin/metarename  [-s setname] metadevice1  metadevice2

/usr/sbin/metarename  [-s setname] [-f] -x metadevice1
 metadevice2

/usr/sbin/metarename  -h
```

機能説明

metarename には 2 とおりの使用法があります。-x オプションを指定する方法と、指定しない方法です。-x オプションを指定しない方法では、既存のメタデバイスの名前を新しい名前に変更します。この機能により、メタデバイスの名前空間をより簡単に管理できます。ただし、マウントされているメタデバイスやオープン状態となっているメタデバイスの名前は変更できません。また既存の名前を割り当てることもできません。たとえば、名称変更したいマウントされているファイルシステムをもつメタデバイスは、まずファイルシステムのマウントを解除してから名称変更してください。

metarename の2 つ目の使用法は、 -x オプションを指定することにより、階層化された既存のメタデバイスの名前を、そのサブデバイスの 1 つと切り替える (交換する) ことです。Solaris ボリュームマネージャでは、階層化されたメタデバイスとはミラーもしくはトランスメタデバイスを指します。-x オプションを指定すると、ミラーのメタデバイス名をそのサブミラーの 1 つと交換したり、トランスメタデバイス名をそのマスターデバイスと交換したりできます。

したがって、metarename -x により既存のストライプや連結のミラー化やミラー化の解除を行なったり、トランスデバイスを削除したりすることがより簡単になります。

既存のストライプや連結をミラー化する場合は、そのデバイスへのアクセスを中止しなければなりません。たとえば、マウントされているファイルシステムをもつデバイスは、名称変更する前にそのファイルシステムのマウントを解除しなければなりません。

metarename -x コマンドを使用すると、既存のデバイスからトランスメタデバイスをトランス解除することもできます。この機能はマスターデバイスだけに適用できます。metarename コマンドでは、ロギングデバイスを削除できません。トランスデバイスの名称を変更する前に、まずロギングデバイスを切り離し、そのトランスメタデバイスへのアクセスを中止する必要があります。

メタデバイス自体や、そのサブコンポーネントがエラー状態である場合、またはメタデバイスにホットスペアと交換されているものがある場合、そのメタデバイスの名称を変更したり、交換することはできません。

メタデバイス名を交換できるのは、親子関係にあるものだけです。たとえば、マスターデバイスとなっているミラー中のストライプとそのトランスメタデバイスとでは、名前を直接交換することはできません。

トランスメタデバイスのメンバーを交換するときは、必ず -f オプションを使用してください。

この場合、交換できるのは、スライスではなくメタデバイスです。

オプション

以下のオプションがサポートされています。

-f	トランスメタデバイスのメンバーを強制的に切り替えます。
-h	ヘルプメッセージを表示します。
-s *setname*	metarename コマンドを実行するディスクセットの名前を指定します。-s オプションを指定すると、metarename コマンドは指定されたディスクセット内でのみ、その機能を実行します。このオプションを指定しないと、metarename コマンドはローカルのメタデバイス上でその機能を実行します。
-x	メタデバイス名 *metadevice1* と *metadevice2* を交換します。
metadevice1	変更または交換したいメタデバイス名を指定します。
metadevice2	新しいメタデバイス名を指定します。

使用例

例 1　メタデバイスの名称変更

この例では、メタデバイスの名称を d10 から d100 に変更します。この場合、d100 がまだ存在していない名前であれば、この変更は成功します。

```
# metarename d10 d100
```

例 2　2 面ミラーの作成

この例では、マウントされているファイルシステム /home2 をもつ既存のストライプ d1 から 2 面ミラーを作成します。

```
# metainit d2 1 1 c13d0s1
# metainit -f d20 -m d1
# umount /home2
# metarename -x d20 d1
# metattach d1 d2
# mount /home2
```

まず、2 番目の連結 d2 が作成されます (d1 はすでに存在しているため)。metainit コマンドは、d1 から d20 という 1 面ミラーを作成します。次に、ファイルシステムのマウントを解除し、d1 と d20 を交換して、d1 をトップレベルのデバイス (ミラー) にします。この後、2 番目のサブミラー d2 を接続して、2 面ミラーを生成します。最後に、ファイルシステムをマウントし直します。

例3 ミラー化されているファイルシステムをストライプとしてマウント

この例では、最初ファイルシステムは既存のミラー d1 にマウントされていますが、最後はストライプ d1 としてマウントされます。

```
# umount /fs2
# metarename -x d1 d20
# metadetach d20 d1
# metaclear -r d20
# mount /fs2
```

まず始めにファイルシステムのマウントを解除し、ミラー d1 とそのサブミラー d20 の名前を交換します。この結果、ミラーは d20 になります。次に、d20 から d1 を切断し、ミラー d20 とその他のサブミラーを削除します。最後にファイルシステムをマウントし直します。

例4 トランスメタデバイスの削除

この例では、マウントポイント /myhome にマウントされているトランスメタデバイス d10 を削除します。マスターデバイス d2 と、ロギングデバイス d5 は、どちらもストライプです。

```
# umount /myhome
# metadetach d10
# metarename -f -x d10 d2
# metaclear d2
# metaclear d5
# fsck /dev/md/dsk/d10
# mount /myhome
```

まず始めにファイルシステムのマウントを解除し、トランスメタデバイスのロギングデバイス d10 を切断します。次にトランスメタデバイスと、そのマスターデバイスとを交換するため、トランスメタデバイスが d2 となり、その配下のストライプが d10 となります。次に、トランスメタデバイス d2 とロギングデバイス d5 をクリアします。その後、d10 に必ず fsck コマンドを実行してからファイルシステムをマウントし直します。

終了ステータス

以下の終了ステータスが返されます。

0 正常終了

>0 エラーが発生した

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

制限事項

名称変更や、名前の交換ができるのは、メタデバイスのみです。物理スライスをメタデバイスに名称変更したり、メタデバイス名を物理的なスライス名に変更することはできません。

メタデバイスには、*d<xyz>* という形式の名前を指定します。この場合、*xyz* には `0` - `8192` までの数字が入ります。メタデバイスに論理名を指定することはできません。

注意事項

トランスメタデバイスは UFS ロギングによって置き換えられました。既存のトランスデバイスはロギングを行わず、その元となるデバイスにデータを直接渡します。UFS ロギングについての詳細は、mount_ufs(1M) を参照してください。

名前

metareplace – サブミラーまたは RAID5 メタデバイスのコンポーネントを使用可能状態にするか、または交換する

形式

```
/usr/sbin/metareplace  -h

/usr/sbin/metareplace  [-s setname] -e mirror component

/usr/sbin/metareplace  [-s setname] mirror component-old
 component-new

/usr/sbin/metareplace  [-s setname] -e RAID component

/usr/sbin/metareplace  [-s setname] [-f] RAID component-old
 component-new
```

機能説明

metareplace コマンドは、サブミラーまたは RAID5 メタデバイス内のコンポーネント (スライス) を使用可能状態にするか、または交換します。

コンポーネントを交換するとき、metareplace コマンドは自動 的に新しいコンポーネントと残りのメタデバイスとの再同期処理を行います。再同期処理が完了すると、交換されたコンポーネントは読み取りと書き込みが可能になります。障害のあるコンポーネントがホットスペアと交換されていた場合は、ホットスペアが使用可能になり、ほかのホットスペア交換に利用できるようになります。

新しいコンポーネントには、少なくとも古いコンポーネントと同等の容量が必要です。

コンポーネントが Last Erred 状態または Maintenance 状態になった場合は、処置が必要です。常に Maintenance 状態のコンポーネントを最初に交換し、次にデータの再同期処理と妥当性検査を行なってください。その後に、Last Erred 状態のディスクを交換します。データを損失しないように、すべてのデータのバックアップをとってから、Last Erred デバイスを交換してください。

オプション

以下のオプションのうち、-h 以外のオプションを実行するには、スーパーユーザーになる必要があります。

-e
: *component* を使用可能な状態にし、障害のあるコンポーネントに対して再同期処理を行います。障害のあるコンポーネントがホットスペアと交換されたいた場合は、ホットスペアが使用可能状態になり、ほかのホットスペア交換に使用できるようになります。このコマンドは、コンポーネントが人為的なエラー (たとえば間違ってディスクの電源をオフにしてしまった場合) によって失敗したり、コンポーネントが物理的に交換された場合などに便利です。この場合、交換用のコンポーネントは metareplace コマンドを実行する前に交換するディスクと一致するようにパーティションを区切っておく必要があります。

-f
: メタデバイスにエラー状態のコンポーネントが複数ある場合、エラーのあるコンポーネントを強制的に交換します。metastat コマンドによって「Maintenance」状態にすると表示されたコンポーネ

ントを最初に交換する必要があります。ただし、複数のコンポーネントがエラー状態にあるときに、このオプションを使用すると、誤ったデータが作成されることがあります。

`-h` ヘルプメッセージを表示します。

`-s` *setname* `metareplace` を実行するディスクセットの名前を指定します。このオプションを使用すると、指定したディスクセット内で `metareplace` が実行されます。このオプションを使用しないと、ローカルのメタデバイスに対して `metareplace` が実行されます。

mirror ミラーのメタデバイス名。

component ディスクドライブ上にある、物理的なスライス (パーティション) の論理名。たとえば `/dev/dsk/c0t0d0s2` などです。

component-old 交換したい物理的なスライスの名前。

component-new *component-old* と交換する物理的なスライスの名前。

RAID RAID5 デバイスのメタデバイス名。

使用例

例 1 RAID5 メタデバイスのエラー状態からの回復

この例では、RAID5 メタデバイスにあるひとつのコンポーネントがエラー状態になった場合の回復方法を示しています。

```
# metareplace d10 c3t0d0s2 c5t0d0s2
```

この例では、RAID5 メタデバイス `d10` にあるエラー状態のコンポーネント `c3t0d0s2` を新しいコンポーネント `c5t0d0s2` と交換しています。

例 2 物理ディスク交換後の `-e` の使用法

ここでは、サブミラー (この場合、ミラー `d11` のサブミラー) にある物理ディスクが交換されたあとの `-e` オプションの使用例を示します。

```
# metareplace -e d11 c1t4d0s2
```

注：交換用ディスクは `metareplace` コマンドを実行する前に、交換するディスクと一致するようにパーティションを分割しておく必要があります。

終了ステータス

以下の終了ステータスが返されます。

`0` 正常終了

`>0` エラーが発生した

属性

以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

名前

metaroot – ルート (/) メタデバイスのシステムファイルを設定する

形式

```
/usr/sbin/metaroot  -h

/usr/sbin/metaroot  [-n] [-k system-name] [-v vfstab-name]
 [-c mddb.cf-name] [-m md.conf-name] [-R root-path] device
```

機能説明

metaroot コマンドは、ルートファイルシステム (/) が適切なメタデバイス上に設定された状態でシステムを起動できるように、/etc/vfstab ファイルと /etc/system ファイルを編集します。ルートファイルシステムをサポートするメタデバイスは、単一スライスのストライプ、または、単一スライスのストライプ上にあるミラーのみです。

必要に応じて、metaroot コマンドはメタデバイス上に構成されたルートファイルシステム (/) からブートするように設定されているシステムの設定を、物理スライスを使用するようにリセットすることができます。

オプション

以下のオプションのうち、-h 以外のオプションを実行するには、スーパーユーザーになる必要があります。

以下のオプションがサポートされています。

-c *mddb.cf-name*
: メタデバイスデータベースの位置が記述されるデフォルトのファイル /etc/lvm/mddb.cf の代わりに、*mddb.cf-name* を使用します。

-h
: 使用方法に関するメッセージを表示します。

-k *system-name*
: デフォルトの /etc/system システム構成ファイルの代わりに、ユーザーが指定した *system-name* を編集します。

-m *md.conf-name*
: デフォルトの /kernel/drv/md.conf ではなく、*md.conf-name* により指定されている構成ファイルを編集します。

-n
: 実際に処理を行わずに、処理内容を出力します。

-R *root-path*
: metaroot がシステムファイルを変更する場合、*root-path* の下位にある相対位置でシステムファイルにアクセスします。

  -R オプションは、-c、-k、-m、または -v オプションと組み合わせて使用することはできません。

  注 – いかなる非大域ゾーンのルートファイルシステムも -R で参照しないでください。 この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティを損ねたり、非大域ゾーンのファイルシステムを損傷する可能性があります。 zones(5) のマニュアルページを参照してください。

-v *vfstab-name*
: ファイルシステムのデフォルトを設定するデフォルトテーブル /etc/vfstab の代わりに、*vfstab-name* を編集します。

オペランド

以下のオペランドがサポートされています。

device ルートファイルシステム (/) に使用されているディスクデバイス (スライス) とメタデバイスのどちらかを指定します。

使用例

例1 メタデバイス上のルートファイルシステムの指定

以下の例では、/etc/system と /etc/vfstab を編集して、メタデバイス d0 上にルートファイルシステムを定義します。

```
# metaroot d0
```

例2 SCSI ディスク上のルートファイルシステムの指定

以下の例では、/etc/system と /etc/vfstab を編集して、SCSIディスクデバイス /dev/dsk/c0t3d0s0 上にルートファイルシステムを定義します。

```
# metaroot /dev/dsk/c0t3d0s0
```

ファイル

/etc/system	システム構成情報ファイル。system(4) を参照してください。
/etc/vfstab	ファイルシステムのデフォルトを指定するファイル
/etc/lvm/mddb.cf	メタデバイス状態データベースの位置が記述されているファイル
/kernel/drv/md.conf	メタデバイスドライバ md 用の構成ファイル

終了ステータス

以下の終了ステータスが返されます。

0 正常終了

>0 エラーが発生した

属性

以下の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), attributes(5), md(7D)

『Solaris ボリュームマネージャの管理』

名前

metaset – ディスクセットを構成する

形式

```
/usr/sbin/metaset  -s setname [-M-a -h hostname]

/usr/sbin/metaset  -s setname -A{enable | disable}

/usr/sbin/metaset  -s setname [-A{enable | disable}]
 -a -h hostname...

/usr/sbin/metaset  -s setname -a [-l length] [-L] drivename...

/usr/sbin/metaset  -s setname -C {take | release |
 purge}

/usr/sbin/metaset  -s setname -d [-f] -h hostname...

/usr/sbin/metaset  -s setname -d [-f] drivename...

/usr/sbin/metaset  -s setname -j

/usr/sbin/metaset  -s setname -r

/usr/sbin/metaset  -s setname -w

/usr/sbin/metaset  -s setname -t [-f] [-u tagnumber]
 [y]

/usr/sbin/metaset  -s setname -b

/usr/sbin/metaset  -s setname -P

/usr/sbin/metaset  -s setname -q

/usr/sbin/metaset  -s setname -o [-h hostname]

/usr/sbin/metaset  [-s setname]

/usr/sbin/metaset  [-s setname] -a | -d [ [m] mediator_host_list]
```

機能説明

metaset コマンドは、指定されたディスクセット内のディスクのセットを管理します。名前が付けられたディスクセットには、ローカルセット内にないすべてのディスクセットが含まれています。ディスクセットにより高可用性構成が実現されますが、Solaris ボリュームマネージャそれ自体は実際には高可用性環境を実現していません。

単一所有者ディスクセット構成は SAN 上のストレージまたはファブリック接続ストレージを管理し、また指定したディスクのセットに対して名前空間の制御および状態データベースの複製の管理を行います。

共有ディスクセット構成では、複数のホストが同じディスクのセットに物理的に接続されています。一方のホストが故障すると、もう一方のホストがそのディスクに対して排他的にアクセスすることになります。各ホストは共有ディスクセットを制御できますが、一度に共有ディスクセットを制御できるのは 1 つのホストのみです。

ディスクセットに新しいディスクを追加する際には、Solaris ボリュームマネージャがディスクの形式を検査します。必要に応じて、状態データベースの複製用の適切な容量を使用して、適切に設定された予約済みのスライス 7 (または EFI ラベル付きのデバイス上のスライス 6) をディスクが持つよう、Solaris ボリュームマネージャはディスクのパーティションを再設定します。スライス 7 (または EFI ラベル付きのデバイス上のスライス 6) の正確なサイズは、ディスクのジオメトリに依存します。従来のディスクセットの場合、スライスは 4M バイトを下回ることはなく、シリンダの境界がどこに存在するかに応じて、6M バイトに近づくことが多くなります。複数所有者ディスクセットの場合、スライスは最小で 256M バイトです。スライス 7 の最小サイズは将来変更される可能性があります。この変更は、状態データベースの複製のサイズ、および状態データベースの複製に格納される情報を含む、さまざまな要因に基づいて行われます。

ディスクセットで使用するためには、ディスクには特定の基準を満たす専用のスライス (6 または 7) が必要です。

- スライスはセクター 0 から始まる必要がある
- スライスにはディスクラベル用の十分な容量が含まれている必要がある
- 状態データベースの複製をマウントすることはできない
- スライスは、スライス 2 を含むその他のデバイスとは重複しない

既存のパーティションテーブルがこれらの基準を満たしていない場合、または -L フラグが指定されている場合、Solaris ボリュームマネージャはディスクのパーティションを再設定します。各ドライブのごく一部が、Solaris ボリュームマネージャで使用するために、スライス 7 (または EFI ラベル付きのデバイス上のスライス 6) に予約されます。各ドライブの残りの容量は、スライス 0 に配置されます。パーティションを再設定することで、ディスク上の既存のデータはすべて失われます。

ディスクセットにドライブを追加したあとは、必要に応じてパーティションを再設定できますが、スライス 7 (または EFI ラベル付きのデバイス上のスライス 6) は決して変更されないという例外があります。

ディスクセットを作成し、ディスクセット内でメタデバイスを設定すると、メタデバイス名は、以下のような形式になります。

`/dev/md/`*setname*`/{dsk,rdsk}/d`*number*

setname はディスクセットの名前、*number* はメタデバイスの番号 (0 ～ 127) です。

Solstice DiskSuite ソフトウェアからアップグレードしたディスクセットがある場合、これらのセット上にあるデフォルトの状態データベースの複製のサイズは、Solaris ボリュームマネージャの 8192 ブロックサイズではなく、1034 ブロックになります。また、Solstice DiskSuite の使用時に追加されたディスク上のスライス 7 は、それに応じて、Solaris ボリュームマネージャの使用時に追加されたディスク上のスライス 7 よりも小さくなります。

ディスクセットに追加したディスクに、受け入れ可能な (シリンダ 0 から始まり、状態データベースの複製用の十分な空き容量がある) スライス 7 がある場合、それらは再フォーマットされません。

ローカルのディスクセットにあるホットスペア集合は、Solaris ボリュームマネージャの標準命名規則に従っています。共有ディスクセットにあるホットスペア集合は、以下のような命名規則を使用します。

setname/hsp*number*

ここで、*setname* はディスクセットの名前で、*number* はホットスペア集合の番号 (0〜999) です。

複数ノード環境

複数ノード環境でディスクセットを作成し、ディスクセットで作業を行うには、`root` がすべてのホスト上のグループ 14 のメンバーであるか、または `/.rhosts` ファイルにその他すべてのホスト名のエントリが含まれている必要があります。これは、SunCluseter 3.x 環境では必要ありません。

タグ付きデータ

タグ付きデータが発生するのは、ディスクセットの複製の異なるバージョンが存在する場合です。このタグ付きデータは、セットの所有者のノード名、所有者のハードウェアシリアル番号、および使用可能な複製に書き出された時刻から構成されています。システム管理者はこの情報を使用すると、どの複製に正しいデータが含まれているかを判断できます。

ディスクセットが偶数のストレージ格納装置を使用して構成され、複製がそれらの格納装置で均等に分散されている場合、(半分のストレージ格納装置の電源障害などにより) 最大半分の複製が失われる可能性があります。電源障害が発生した格納装置をリブートしたあとで、複製の半分は Solaris ボリュームマネージャにより認識されません。セットの再取得が行われた場合、`metaset` コマンドは「stale databases」のエラーを返し、すべてのメタデバイスが読み取り専用状態になります。

認識されない複製の一部は削除する必要があります。また、複製を削除する操作により、削除されていない複製も更新されます。デュアルホストのディスクセット環境では、2 番目のノードはセットの取得時に、既存の複製ではなく削除された複製にアクセスする可能性があります。これにより、ディスクセットの取得時に誤った複製レコードを取得する可能性があります。エラーメッセージが表示され、ユーザーの操作が必要になります。

ディスクセットを照会するには `-q` オプションを使用し、タグを選択してディスクセットを取得するには `-t`、`-u`、および `-y` オプションを使用します。各オプションを参照してください。

メディエータの構成

Solaris ボリュームマネージャは、2 組の一連のドライブのみを共有する 2 つのホストから構成される、ローエンド HA ソリューションをサポートしています。この種類の構成のホストは、メディエータまたはメディエータホストと呼ばれ、特別な

デーモン rpc.metamedd(1M) を実行します。メディエータホストには、ホストやドライブに障害が発生した場合もデータの可用性を確保するという追加的な責任があります。

メディエータ構成は、1 つのホストまたは一連のドライブに障害が発生しても、管理者の操作なしで稼働を続行できます。ホスト、および一連のドライブの両方に障害が発生した場合 (複数障害)、データの整合性は保証できません。この場合、データをアクセス可能にするには、管理者による操作が必要になります。詳細については mediator(7D) を参照してください。

-m オプションを使用してメディエータホストを追加または削除します。各オプションを参照してください。

オプション

サポートしているオプションは、次のとおりです。

-a

指定したディスクセットにドライブまたはホストを追加します。ただし、別のメタデバイスやディスクセット内で使用中であったり、マウントされていたり、Swap として使用されているドライブは、ディスクセット内に追加することはできません。ディスクセットに追加されるドライブは、パーティションが再設定され、メタデバイス状態データベースの複製 (そのディスクセット用の複製) もドライブ上に置かれます。しかし、スライス 7 (または EFI ラベル付きのデバイス上のスライス 6) がシリンダ 0 で始まっていて、状態データベースの複製を持てるだけの容量がある場合、ディスクパーティションの再設定は行われません。また、ディスクセットの一部として指定されたすべてのホスト上にドライブが見つからない場合は、ドライブは受け入れられません。つまり、特定のセットにあるホストがネットワークの障害によって通信不可能であったり、または管理上の理由によりダウンしている場合には、ドライブを追加することができません。

-a | -d | -m *mediator_host_list*

指定したディスクセットに対して、メディエータホストの追加 (-a) または削除 (-d) を行います。*mediator_host_list* は、追加されるメディエータホストの nodename(4)、および (追加する場合は) メディエータホストの最大 2 つの別名です。各メディエータホストのノード名と別名は、コンマのみで区切られます。名前付きディスクセットには最大 2 つのメディエータホストを指定できます。メディエータホストを削除するには、-m に対する引数としてそのホストのノード名のみを指定します。

1 つの metaset コマンドの実行で 2 つのメディエータホストを追加または削除できます。「使用例」を参照してください。

`-A {enable | disable}` ディスクセットの自動取得状態を指定します。あるセットに対して自動取得が有効になっている場合、そのディスクセットはブート時に自動的に取得され、そのディスクセット内のボリューム上にあるファイルシステムは、/etc/vfstab エントリを介してマウントできます。自動取得セットに対応付けることができるのは 1 つのホストのみであるため、自動取得セットに 2 つ目のホストを追加しようとしたり、複数のホストを持つディスクセットを自動取得として構成しようとしたりすると、エラーメッセージが表示され失敗します。特定のディスクセットの自動取得状態を無効にすると、そのディスクセットは通常の動作に戻ります。つまり、そのディスクセットは潜在的にホスト間で (同時ではなく) 共有され、/etc/vfstab を介してマウントすることはできません。

`-b` 複製を、複製配置アルゴリズムに従って配付します。これはいつでも実行できます。複製が正しく配付されたときは、何も返しません。ユーザーが metadb コマンドを使用して手動で 複製を削除もしくは追加していた場合には、このコマンドを使用して複製配置アルゴリズムに従って複製の配付を確実に行うことができます。

`-C {take | release | purge}` Sun Cluster 3 環境で使用している場合、クラスタ フレームワークとのやり取りは行いません。実際にこの方法では、Cluster Configuration Repository は変更されません。これらのオプションは、破壊されたディスクセット構成を修正するためにのみ使用するべきです。このオプションは、複数所有者ディスクセットでは使用できません。

take ディスクセットの所有権を取得しますが、ディスクセットが使用可能であることをクラスタ フレームワークには通知しません。

release クラスタ フレームワークに通知することなく、ディスクセットの所有権を解放します。このオプションは、対応する -C take

オプションを使用してディスクセットの所有権が取得されている場合にのみ使用するべきです。

purge　ディスクセットが削除されたことをクラスタフレームワークに通知することなく、ディスクセットを削除します。

-d　指定したディスクセットからドライブまたはホストを削除します。ただし、ディスクセット内で使用中のドライブは、削除できません。ディスクセット内のドライブをすべて削除しない限り、最後のホストを削除することはできません。ディスクセットの最後のホストを削除すると、ディスクセットが破壊されます。

その他のノードがセット内に存在する間にマスターノードを削除しようとすると、複数所有者ディスクセットではこのオプションは失敗します。

-f　次の 3 とおりの動作のうちの 1 つを強制的に行います。 1) -t と共に使用された時はディスクセットの所有権を取得します、2) ディスクセットから最後のディスクドライブを削除します、3) ディスクセットから最後のホストを削除します。(最後のドライブやホストを削除する場合には -d オプションが必要です。)

ディスクセットの所有権を強制的に取得すると、別のホストによって所有されているかどうかに関わらず、ディスクセットを所有できます。ディスクセット内のすべてのディスクが占有され (予約され)、FailFast が使用可能になります。その結果、別のホストがそのディスクセットの所有権を持っていると、そのホストはパニック状態になります。ホストが取得を行うと、メタデバイス状態データベースが読み込まれ、ディスクセット内の共有メタデバイスにアクセスできるようになります。

このオプションを使用してディスクセットの最後のドライブを削除できます。これは、このドライブには最後の状態データベースの複製が暗黙的に含まれているためです。

また、ディスクセットからホストを削除する場合も、-f オプションを使用できます。引数としていく

つかのホストをリストで指定すると、このオプションは1ホストでの管理用として使用できます。1ホストでの管理は、ホストが機能不能として認識されている場合、タイムアウトやコマンドの失敗を回避できるため便利です。引数としてすべてのホストをリストで指定すると、ディスクセットが完全に削除されます。通常このオプションは、1ホスト管理を行なったあとのクリーンアップのために、すべてのホストをリストで指定して使用します。

`-h` *hostname...*

ディスクセットに追加するホスト、またはディスクセットから削除するホストの名前を1つまたは複数指定します。最初のホストを追加すると、ディスクセットが作成されます。ディスクセット内のすべてのドライブを削除しない限り、最後のホストを削除することはできません。指定したホスト上にディスクセット内のすべてのドライブが見つからないと、このコマンドは実行されません。ホスト名は、`/etc/nodename` にある名前と同じです。

`-j`

複数所有者ディスクセットの所有者リストにホストを追加します。従来のディスクセットで使用される取得と解放という概念は、複数所有者セットには適用されませんが、これは複数の所有者が許可されているためです。

ホストがブートしオンライン状態になり、複数所有者ディスクセットを使用できるようになるためには、次の3つの構成レベルを満たす必要があります。

1. クラスタノードリストに含まれている必要があります。これは、クラスタまたはノードが1つである状況では自動的に行われます。
2. このマニュアルページのほかの箇所で説明されている `-a -h` オプションを使用して、複数所有者ディスクセットに追加する必要があります。
3. セットに追加する必要があります。最初にホストがセットに追加される場合、自動的に追加されます。

手動で再起動を行う時点で、管理者は次のコマンドを手動で発行する必要があります。

`metaset -s` *multinodesetname* `-j`

これにより所有者リストにホストが追加されます。クラスタ再構成のあとで、ホストがクラスタに再び入った場合、ノードは自動的にセット内に追加されます。`metaset -j` コマンドにより、ホストの追加先であるすべての複数所有者セットにホストが結合されます。ノードが 1 つである場合、ノードをディスクセットに追加すると、必要な再同期がすべて始まります。

`-L`
ディスクをディスクセットに追加する場合、標準の Solaris ボリュームマネージャのアルゴリズムを使用して強制的にディスクのパーティションを再設定します。「機能説明」を参照してください。

`-l` *length*
メタデバイス状態データベースの複製のサイズ (ブロック単位) を設定します。length を設定できるのは、新しいドライブを追加するときだけです。既存のドライブでは変更できません。デフォルト (および最大) のサイズは 8192 ブロックで、これが大部分の構成で適切です。128 ブロック未満の複製のサイズはお勧めしません。

`-M`
作成または変更されるディスクセットを、複数の同時所有者をサポートする複数所有者ディスクセットに指定します。

このオプションは、複数所有者ディスクセットを作成する際に必要です。複数所有者ディスクセットでのその他すべての操作に対するこのオプションの使用は任意で、何の影響もありません。既存のディスクセットを複数所有者セットに変換することはできません。

`-o`
ローカルホストまたは `-h` オプションで指定されたホストがディスクセットを所有している場合に、終了ステータス `0` を返します。

`-P`
`metaset` コマンドが実行されているノードから、名前付きディスクセットを削除します。ディスクセットは、このコマンドを実行するノードによって所有されてはいけません。ノードが実際にディスクセットを所有している場合、コマンドは失敗します。

ディスクセットを削除する必要があり、セットの所有権を取得できない場合に、`-P` オプションを使用します。

このオプションは、複数所有者ディスクセットでは使用できません。

`-q`
ディスクセットの所有権の取得時に検出可能な「タグ付きデータ」に関するタグの列挙リストを表示します。

このオプションは、複数所有者ディスクセットでは使用できません。

`-r`
ディスクセットの所有権を解放します。ディスクセット内のすべてのディスクが解放されます。ディスクセット内に設定されたメタデバイスにアクセスできなくなります。

このオプションは、複数所有者ディスクセットでは使用できません。

`-s` *setname*
`metaset` を実行するディスクセットの名前を指定します。*setname* が指定されていない場合は、すべてのディスクセットが返されます。

`-t`
ディスクセットの所有権を安全に取得します。ディスクセットが別のホストによって所有されている場合は、ここで指定したホストはディスクセットを所有することができません。ディスクセットが別のホストによって所有されていない場合は、ディスクセット内のすべてのディスクが `metaset` を実行したホストによって所有されます。メタデバイス状態データベースが読み込まれ、ディスクセット内に設定された共有メタデバイスにアクセスできるようになります。`-t` オプションは、無効なデータベースを持つディスクセットを取得します。データベースが無効であると、`metaset` はコード 66 で終了し、メッセージを表示します。その時点で可能な操作は、複製の追加と削除だけです。複製の追加や削除が完了すると、ディスクセットは解放され、再度ディスクセットへ取り込まれてデータを完全にアクセスできるようになります。

このオプションは、複数所有者ディスクセットでは使用できません。

`-u` *tagnumber*
タグが選択されていれば、続いて `-u` *tagnumber* を使用した取得を実行して、指定されたタグ番号と関連付けられたデータを選択できます。

w

複数所有者ディスクセットの所有者リストからホストを削除します。従来のディスクセットで使用される取得と解放という概念は、複数所有者セットには適用されませんが、これは複数の所有者が許可されているためです。

セットを解放する代わりに、ホストは次のコマンドを発行できます。

```
metaset -s multinodesetname -w
```

これにより所有者リストから削除されます。ホストはリブート時に自動的に離脱します。ただし、ホストがセットを使用できない状態にするべきだが、あとでセットに戻す可能性がある場合は、ホストを手動で削除することもできます。リブートにより離脱したホストは、再構成サイクルが実施されるまでは、セット内のほかのホストからは存在しているように見える可能性があります。

`metaset -w` は、ホストがメンバーであるすべての複数所有者セットの所有権からホストを削除します。その他のノードがディスクセット所有者リスト内に存在する間にマスターノードを削除しようとすると、このオプションは失敗します。このオプションは、ノード上で実行されているすべての再同期を取り消します。クラスタメンバーシップリストからノードを削除するクラスタ再構成プロセスは、効果的に所有権リストからホストを削除します。

-y

継続的な取得を実行します。take オプションが「タグ付きデータ」を検出すると、取得処理はコード 2 とともに終了します。続いて `-q` オプションを付けて `metaset` コマンドを実行すると、タグの列挙リストを参照できます。

使用例

例 1 ディスクセットの定義

この例では、ディスクセットを定義します。

```
# metaset -s relo-red -a -h red blue
```

この例で、ディスクセットの名前は `relo-red` です。セットに追加された 1 番目と 2 番目のホストの名前は、`red` と `blue` です。ホスト名は `/etc/nodename` に書かれています。1 番目のホストを追加すると、複数所有者ディスクセットが生成されます。ディスクセットはホストが 1 つだけでも作成でき、あとでホストを追加することができます。セットに含まれるすべてのドライブが削除されるまで、最後のホストは

例 1 ディスクセットの定義 *(続き)*

削除できません。

例 2 ディスクセットへのドライブの追加

この例では、ディスクセットにドライブを追加します。

```
# metaset -s relo-red -a c2t0d0 c2t1d0 c2t2d0 c2t3d0 c2t4d0 c2t5d0
```

この例で、以前生成したディスクセットの名前は relo-red です。ドライブの名前は、c2t0d0、c2t1d0、c2t2d0、c2t3d0、c2t4d0、および c2t5d0 です。ドライブ名の末尾にスライスの識別子 ("s*x*") は付きません。

例 3 複数のメディエータホストの追加

以下のコマンドは、指定したディスクセットに 2 つのメディエータホストを追加します。

```
# metaset -s mydiskset -a -m myhost1,alias1 myhost2,alias2
```

例 4 ノードからのディスクセットの削除

以下のコマンドは、ノードからディスクセット relo-red を削除します。

```
# metaset -s relo-red -P
```

例 5 ディスクセットへのタグ付きデータの照会

以下のコマンドは、タグ付きデータのリストのディスクセット relo-red を照会します。

```
# metaset -s relo-red -q
```

このコマンドにより以下の結果が得られます。

```
The following tag(s) were found:
 1 - vha-1000c - Fri Sep 20 17:20:08 2002
 2 - vha-1000c - Mon Sep 23 11:01:27 2002
```

例 6 タグの選択とディスクセットの取得

以下のコマンドはタグを選択し、ディスクセット relo-red を取得します。

```
# metaset -s relo-red -t -u 2
```

例7 複数所有者ディスクセットの定義

以下のコマンドは、複数所有者ディスクセットを定義します。

```
# metaset -s blue -M -a -h hahost1 hahost2
```

この例で、ディスクセットの名前は blue です。セットに追加される 1 番目と 2 番目のホストの名前は、それぞれ hahost1 と hahost2 です。ホスト名は /etc/nodename に書かれています。1 番目のホストを追加すると、複数所有者ディスクセットが生成されます。ディスクセットはホストが 1 つだけでも生成でき、あとでホストを追加できます。セットに含まれるすべてのドライブが削除されるまで、最後のホストは削除できません。

ファイル

/etc/lvm/md.tab　メタデバイス構成のリストが含まれています。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生した。

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

『Solaris ボリュームマネージャの管理』

注意事項

ディスクセット管理 (ホストおよびドライブの追加と削除を含む) では、ネットワークからディスクセット内のすべてのホストにアクセスできる必要があります。

名前 metassist – automated volume creation utility to support Solaris Volume Manager

形式

```
metassist -V

metassist -?

metassist create [-v n] [-c] -F config_file

metassist create [-v n] [-c | -d] -F request_file

metassist create [-v n] [-c | -d] [-f] [-n name] [-p datapaths]
 [-r redundancy] [-a available [,available,...]]
 [-u unavailable [,unavailable,...]] -s setname
 -S size

metassist create -?
```

機能説明 The `metassist` command provides assistance, through automation, with common Solaris Volume Manager tasks.

SUBCOMMANDS The following subcommands are supported:

`create` The `create` subcommand creates one or more Solaris Volume Manager volumes. You can specify this request on the command line or in a file specified on the command line.

If you create a volume using the command line, you can specify the characteristics of the volume in terms of the desired quality of service it will provide - its size, the number of redundant copies of the data it contains, the number of data paths by which it is accessible, and whether faulty components are replaced automatically. The diskset in which the volume will reside and the volume's size must be specified on the command line in this form of the command.

If you create a volume using a request in a file, you can specify the characteristics of the volume in terms of the quality of service they provide, as on the command line. Alternatively, the file can specify the types and component parts of the volume, (for example, mirrors, stripes, concatenations, and their component slices). The file may also specify volumes partly in terms of their types and partly in terms of their component parts, and may specify the characteristics of more than one volume. All volumes specified in a file must reside in the same diskset, whose name must be specified in the file.

If you specify the `-c` or `-d` option on the command line, the command runs without creating an actual volume or volumes. Instead, it outputs either a a Bourne shell command script (`-c` option) or a volume configuration (`-d` option). The command script, when run, creates the

specified volume or volumes. The volume configuration specifies the volume or volumes in complete detail, naming all their components.

The input file given on the command line can take one of the following forms:

- a volume request, which specifies a request for a volume with explicit attributes and components, or matching a given quality of service
- a volume configuration, produced by a previous execution of the command

オプション

The following option is mandatory if you specify a volume request or volume configuration in a file:

`-F` *config_file* | *request_file*
Specify the volume request or volume configuration file to process. If *config_file* or *request_file* is `-`, it is read from standard input.

The `-d` option cannot be specified when *inputfile* is a volume configuration file.

The following options are mandatory if you specify a volume request on the command line:

`-s` *set*
Specify the disk set to use when creating volumes. All the volumes and hot spare pools are created in this disk set. If necessary, disks are moved into the diskset for use in the volumes and hot spare pools. If the diskset doesn't exist the command creates it. This option is required. `metassist` works entirely within a named disk set. Use of the local, or unnamed disk set, is not allowed.

`-S` *size*
Specify the size of the volume to be created. The size argument consists of a numeric value (a decimal can be specified) followed by KB, MB, GB, or TB, indicating kilobytes, megabytes, gigabytes, or terabytes, respectively. Case is ignored when interpreting this option. This option is required.

The following options are optional command line parameters:

`-a` *device1*`,`*device2*`,...`
Explicitly specify the devices that can be used in the creation of this volume. Named devices may be controllers or disks. Only used when specifying a volume on the command line.

`-c`
Output the command script that would implement the specified or generated volume configuration. The command script is not run, and processing stops at this stage.

`-d`
Output the volume configuration that satisfies the specified or generated volume request. No command script is generated or executed, and processing stops at this stage.

`-f`
Specify whether the volume should support automatic component replacement after a fault. If this option is specified, a mirror is created and its submirrors are associated with a hot spare.

`-n` *name*
Specify the name of the new volume. See metainit(1m) for naming guidelines.

`-p` *n*
Specify the number of required paths to the storage volume. The value of *n* cannot be greater than the number of different physical paths and logical paths to attached storage. Only used when specifying a volume on the command line.

`-r` *n*
Specify the redundancy level (`0`-`4`) of the data. The default is `0`. Only used when specifying a volume on the command line. If redundancy is `0`, a stripe is created. If redundancy is `1` or greater, a mirror with this number of submirrors is created. In this case, the volume can suffer a disk failure on *n*`-1` copies without data loss. With the use of hot spares (see the `-f` option), a volume can suffer a disk failure on *n*+`hsps-1` volumes without data loss, assuming non-concurrent failures.

`-u` *device1*`,`*device2*`,...`
Explicitly specify devices to exclude in the creation of this volume. Named devices can be controllers or disks. You can use this option alone, or to exclude some of the devices listed as available with the `-a` option, Only used when specifying a volume on the command line.

`-v` *value*
Specify the level of verbosity. Values from `0` to `2` are available, with higher numbers specifying more verbose output when the command is run. `-v 0` indicates silent output, except for errors or other critical messages. The default level is `1`.

`-V`
Display program version information.

`-?`
Display help information. This option can follow a subcommand for subcommand-specific help.

使用例

例 1 Creating a Mirror

The following example creates a two-way, 36Gb mirror on available devices from controller 1 and controller 2. It places the volume in diskset `mirrorset`.

```
# metassist create -r 2 -a c1,c2 -s mirrorset -S 36G
```

例 2 Creating a Mirror with Additional Fault Tolerance

The following example creates a two-way, 36Gb mirror on available devices from controller 1 and controller 2. It provides additional fault tolerance in the form of a hot spare. It places the volume in diskset `mirrorset`.

```
# metassist create -f -r 2 -a c1,c2 -s mirrorset -S 36GB
```

例 3 Creating a Three-way Mirror and Excluding Devices

The following example creates a three-way, 180Gb mirror from storage devices on controller 1 or controller 2. It excludes the disks `c1t2d0` and `c2t2d1` from the volume. It places the volume in diskset `mirrorset`.

```
metassist create -r 3 -a c1,c2 -u c1t2d0, c2t2d1 \
    -s mirrorset -S 180GB
```

例 4 Determining and Implementing a Configuration

The following example determines and implements a configuration satisfying the request specified in a request file:

```
# metassist create -F request.xml
```

例 5 Determining a Configuration and Saving It in a volume-config File

The following example determines a configuration which satisfies the given request. It saves the configuration in a volume-config file without implementing it:

```
# metassist create -d -F request.xml > volume-config
```

例 6 Determining a Configuration and Saving It in a Shell Script

The following example determines a configuration which satisfies the given request. It saves the configuration in a shell script without implementing it:

```
# metassist create -c -F request.xml > setupvols.sh
```

例 7 Implementing the Given volume-config

The following example implements the given volume-config:

```
# metassist create -F config.xml
```

例 8 Converting the Given volume-config to a Shell Script

The following example converts the given volume-config to a shell script that you can run later:

```
# metassist create -c -F config.xml > setupvols.sh
```

終了ステータス

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ファイル

`/usr/share/lib/xml/dtd/volume-request.dtd`

`/usr/share/lib/xml/dtd/volume-defaults.dtd`

`/usr/share/lib/xml/dtd/volume-config.dtd`

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdr

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `md.tab`(4), `volume-config`(4), `volume-request`(4), `attributes`(5), `md`(7D)

注意事項

The quality of service arguments are mutually exclusive with the `-F` *inputfile* argument.

When specifying a request file or quality of service arguments on the command line, the `/etc/default/metassist.xml` file is read for global and per-disk set defaults.

Characteristics of this file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-defaults.dtd`.

Characteristics of the XML request file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-request.dtd`.

Characteristics of the XML configuration file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-config.dtd`.

This command must be run as root.

This command requires a functional Solaris Volume Manager configuration before it runs.

名前

metastat – メタデバイスまたはホットスペア集合の状態を表示する

形式

```
/usr/sbin/metastat  -h

/usr/sbin/metastat  [-a] [-B] [-c] [-i] [-p] [-q] [-s setname]
 [-t] [metadevice...] [hot_spare_pool...]

/usr/sbin/metastat  [-a] [-B] [-c] [-i] [-p] [-q] [-s setname]
 component...
```

機能説明

metastat コマンドは、それぞれのメタデバイス (ストライプ、連結、連結ストライプ、ミラー、RAID5、ソフトパーティション、およびトランスデバイスを含む) またはホットスペア集合の現在の状態、または、指定したメタデバイス、コンポーネント、またはホットスペア集合の現在の状態を表示します。

metattach コマンドを実行した後に metastat コマンドを実行すると、メタデバイスの状態を表示することができて便利です。

metastat は、システム上の各 Solaris ボリュームマネージャの状態を表示します。取りうる状態の種類は以下のとおりです。

正常 (Okay)	デバイスはエラーを報告していません。
保守要(Needs maintenance)	問題が検出されました。この場合、システム管理者が、障害が発生した物理デバイスを交換する必要があります。Needs maintenance を表示しているボリュームにはデータの損失が起こっていませんが、さらなる障害が発生するとデータが失われる危険があります。可能な限り迅速に措置を行なってください。
最後にエラー (Last erred)	問題が検出されました。データが失われた可能性があります。これは、サブミラーのコンポーネントに障害が発生し、ホットスペアにより置き換えられないため、Needs maintenance 状態に移行した場合に発生する可能性があります。対応するコンポーネントにも障害が発生した場合は Last erred 状態に移行し、有効なデータソースが残っていないため、データが失われた可能性があります。
利用不可 (Unavailable)	デバイスにアクセスできませんが、エラーは発生していません。このような状態になるのは、物理デバイスが Solaris の動的再構成 (DR) 機能により削除され、Solaris ボリュームマネージャのボリュームが使用不可能なままになっている場合が考えられます。また、システムの初期化時にアレイまたはディスクの電源が切られたり、32 ビットモードでのシステムのブート時に 1T バイトを超えるボリュームが存在したりする場合にも発生する可能性があります。

ストレージが使用可能になった後で、-i オプションを付けて metastat コマンドを実行し、メタデバイスの状態を更新します。これにより、アクセス可能なデバイスの使用不能状態がクリアされます。

Needs maintenance 状態または Last erred 状態でのディスクの交換およびボリュームの取り扱いについての手順は『Solaris ボリュームマネージャの管理』を参照してください。

オプション　以下のオプションがサポートされています。

-a　すべてのディスクセットを表示します。現在のホストにより所有されているディスクセット内のメタデバイスのみが表示されます。

-B　すべての 64 ビットメタデバイスおよびホットスペアの現在の状態を表示します。

-c　簡潔な出力を表示します。

1 つのメタデバイスにつき 1 行ずつ出力されます。出力では、各メタデバイスの基本構造および (存在する場合は) エラー状態が表示されます。

-c の出力の形式は -p の出力の形式とは異なります。-p オプションはメタデバイスの状態を表示せず、また人間が判読できる出力としては表示することを目的としていません。

-h　使用方法に関するメッセージを表示します。

-i　RAID1 (ミラー) ボリューム、RAID5 ボリューム、およびホットスペアの状態を検査します。最上位のメタデバイスから順に、アクセスできるかどうか各メタデバイスを照会により検査します。 問題が見つかると、エラーが発生した場合のように、メタデバイス状態データベースが更新されます。

-p　md.tab と同じ形式で、アクティブなメタデバイスとホットスペア集合のリストを表示します。md.tab(4) を参照してください。

-p 出力は、あとでの回復または設定用に現時点の構成の記録を取るために設計されています。

-q　メタデバイスの状態を、デバイス再配置情報なしに表示します。

-s *setname*　metastat を実行するディスクセットの名前を指定します。このオプションを使用すると、指定したディスクセット内で metastat が実行されます。このオプションを指定しない場合は、ローカルディスクセットのメタデバイスまたはホットスペア集合に対して、 metastat が実行されます。

`-t` 指定されたメタデバイスとホットスペア集合の現在の状態とタイムスタンプを表示します。タイムスタンプは、最後に状態が変更された日付と時刻を示します。

オペランド

以下のオペランドがサポートされています。

component 範囲、開始ブロック、およびブロックカウントを含む、ソフトパーティションのホストであるコンポーネントの状態を表示します。

hot_spare_pool 指定されたホットスペア集合の状態を表示します。

metadevice 指定されたメタデバイスの状態を表示します。トランスメタデバイスが指定された場合、マスターとログデバイスの状態も表示されます。トランスメタデバイスは UFS ロギングに置き換えられています。注を参照してください。

使用例

例 1 2 つのサブミラーを持つミラーを表示する出力

以下の例は、2 つのサブミラー `d70` と `d80` からなるミラー `d0` を生成した後、`metastat` コマンドから出力されるメッセージの一部です。

```
# metastat d0
d0: Mirror
    Submirror 0: d80
      State: Okay
    Submirror 1: d70
      State: Resyncing
    Resync in progress: 15 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2006130 blocks
    .
    .
    .
```

例 2 サブミラーを持つミラー上のソフトパーティション

以下の例に、ソフトパーティション上に構築される連結 `d2` 上にソフトパーティション `d3` を作成した後の `metastat` コマンドの出力を部分的に示します。

```
# metastat
d2: Concat/Stripe
    Size: 204800 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        d0                          0     No    Okay
```

例 2 サブミラーを持つミラー上のソフトパーティション (続き)

```
d0: Soft Partition
    Component: c0t3d0s0
    Status: Okay
    Size: 204800 blocks
        Extent              Start Block  Block count
             0                      129       204800

d3: Soft Partition
    Component: d2
    Status: Okay
    Size: 202752 blocks
         Extent              Start Block  Block count
              0                      129       202752
```

例 3 トランスメタデバイス

以下の例に、トランスメタデバイスを作成した後の metastat コマンドの出力を示します。

```
# metastat
d2: Concat/Stripe
    Size: 204800 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        d0                         0     No    Okay

d0: Soft Partition
    Component: c0t3d0s0
    Status: Okay
    Size: 204800 blocks
        Extent              Start Block  Block count
             0                      129       204800

d3: Soft Partition
    Component: d2
    Status: Okay
    Size: 202752 blocks
         Extent              Start Block  Block count
              0                      129       202752
```

例 4 複数所有者ディスクセット

以下の例に、複数所有者ディスクセットとアプリケーションベースのミラー再同期オプションを使用した、metastat コマンドの出力を示します。アプリケーションベースの再同期は、必要に応じて自動的に設定されます。

```
# metastat -s oban
oban/d100: Mirror
    Submirror 0: oban/d10
      State: Okay
    Submirror 1: oban/d11
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Resync option: application based
    Owner: None
    Size: 1027216 blocks (501 MB)

oban/d10: Submirror of oban/d100
    State: Okay
    Size: 1027216 blocks (501 MB)
    Stripe 0:
        Device     Start Block  Dbase     State Reloc Hot Spare
        c1t3d0s0          0     No         Okay

oban/d11: Submirror of oban/d100
    State: Okay
    Size: 1027216 blocks (501 MB)
    Stripe 0:
        Device     Start Block  Dbase     State Reloc Hot Spare
        c1t4d0s0          0     No         Okay
```

警告

metastat は、コマンドの入力時点での状態を表示します。したがって、以下に示す理由により、metastat -p コマンドの出力を使用して md.tab(4) ファイルを作成することはお勧めできません。

- metastat -p の実行結果に、使用中のホットスペアも含まれることがあります。
- また、複数のサブミラーを持つミラーが表示されることもあります。metainit と metattach によって多面ミラーを作成する方法 については、metainit(1m) を参照してください。
- metastat -p の実行後に、スライスがエラー状態になることがあります。

終了ステータス

以下の終了ステータスが返されます。

0 正常終了

`>0` エラーが発生した

属性

以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdr
安定性	発展中

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metasync(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

注意事項

トランスメタデバイスは UFS ロギングに置き換えられています。既存のトランスデバイスはロギングを行わず、基礎デバイスにデータを直接渡します。UFS ロギングの詳細については、mount_ufs(1M) を参照してください。

名前　metasync – リブート時にメタデバイスを再同期処理する

形式

```
/usr/sbin/metasync  -h

/usr/sbin/metasync  [-s setname] [buffer_size] metadevice

/usr/sbin/metasync  [-s setname] -r [buffer_size]

/usr/sbin/metasync  -p metadevice
```

機能説明

metasync コマンドは、指定された *metadevice* の再同期処理を開始します。再同期が必要なすべてのコンポーネントに対して再同期処理が行われます。RAID5 の初期化中や再同期処理中にシステムがクラッシュした場合、システムがリブートされるとそれらの処理が再開されます。

metasync による再同期処理中でも、アプリケーションはメタデバイスにアクセスすることができます。また、metasync は、カーネル内部からもコピーを行い、ユーティリティの処理効率が向上します。

ブート用スクリプトの中で -r オプションを使用すると、すべてのサブミラーが再同期処理されます。

オプション　以下のオプションがサポートされています。

-h　使用方法に関するメッセージを表示します。

-p *metadevice*　RAID5 メタデバイスのパリティー情報を生成し直します。

-s *setname*　metasync を実行するディスクセットの名前を指定します。-s オプションを使用すると、指定したディスクセット内で管理機能が実行されます。このオプションを使用しない場合は、ローカルのメタデバイスに対して metasync が実行されます。

-r　システムのリブート時に行う、特別な再同期処理を行います。metasync -r は、svc:/system/mdmonitor サービスからのみ起動するべきです。metasync コマンドは、再同期が必要なメタデバイスに対してのみ再同期処理を行います。metasync は、パス番号に従ってすべてのミラー再同期処理をスケジュールします。

svc:/system/mdmonitor サービスが使用するデフォルトのバッファーサイズの値を変更するには、/etc/system を編集して指定します。

```
set md_mirror:md_resync_bufsz = 2048
```

上記のように指定すると、再同期処理の速度が最大になります。

オペランド　*buffer_size*　ミラーの再同期に使用する内部コピーバッファーのサイズ (512 バイトディスクブロックの数) を指定します。デフォルトは 512 バイトディスクブロック 128 個分 (64 K バイト) です。2048 ブロックを超える値は使用できません。最良の性能を求める場合 (つまり、再同期処理

を最速で実行したい場合)、推奨されるサイズは 2048 ブロックです。

終了ステータス

以下の終了ステータスが返されます。

`0` 正常終了

`>0` エラーが発生した

属性

以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metattach(1M), `md.tab`(4), `md.cf`(4), `mddb.cf`(4), `attributes`(5), `md`(7D)

『Solaris ボリュームマネージャの管理』

注意事項

`metasync` サービスを管理するには、サービス管理機能 `smf`(5) を使用します。このとき、次のサービス識別子を使用します。

```
svc:/system/mdmonitor
```

このサービスに対する管理アクション (有効化、無効化、または再起動の要求など) を実行するには、svcadm(1M) を使用します。サービスの状態を照会するには、`svcs`(1) コマンドを使用します。

名前

metattach, metadetach – メタデバイスの接続または切り離し

形式

```
/usr/sbin/metattach [-h]

/usr/sbin/metattach [-s setname] mirror [metadevice]

/usr/sbin/metattach [-s setname] [-i interlace] concat/stripe
 component...

/usr/sbin/metattach [-s setname] RAID component...

/usr/sbin/metattach [-s setname] [-A alignment] softpart
 size | all

/usr/sbin/metadetach [-s setname] [-f] mirror submirror

/usr/sbin/metadetach [-s setname] [-f] trans
```

機能説明

metattach は、ミラーにサブミラーを追加したり、メタデバイスを拡張したり、ソフトパーティションを拡張したりします。メタデバイスを拡張するとき、サービスを中断する必要はありません。ミラーまたはトランスのサイズを拡張するには、サブミラーまたはマスターデバイスにスライスを追加する必要があります。

システムが 64 ビットの Solaris カーネルを実行している場合、Solaris ボリュームマネージャは 1T バイトを超えるストレージデバイスと論理ボリューム (「大型ボリューム」と呼ぶ) をサポートします。大型ボリュームのサポートは自動です。1T バイトを超えるデバイスを作成した場合、Solaris ボリュームマネージャは、ユーザーの操作なしで、そのデバイスを適切に構成します。

大型ボリュームを持つシステムを 32 ビットの Solaris カーネルでリブートした場合、この大型ボリュームは metastat 出力に表示されます。大型ボリュームはアクセス、変更、または削除することはできません。新しい大型ボリュームも作成できません。この状況では、大容量ボリューム上にあるボリュームまたはファイルシステムも同様に利用できません。大型ボリュームを持つシステムを Solaris 9 4/03 リリースより前のバージョンの Solaris でリブートした場合、Solaris ボリュームマネージャが起動しません。これより前のバージョンの Solaris オペレーティングシステム下で Solaris ボリュームマネージャを実行する前には、すべての大型ボリュームを削除しておく必要があります。

Solaris ボリュームマネージャは、1 面から 4 面までのミラーをサポートします。したがって、メタデバイスをミラーに接続できるのは、そのミラーが持つサブミラーが 3 つ以下の場合だけです。新しいメタデバイスがミラーに接続されると、metattach は自動的に、新しいサブミラーの再同期処理を開始します。

metadetach は、ミラーからサブミラーを切断したり、トランスメタデバイスからロギングデバイスを切断したりします。

サブミラーがミラーから切断されると、そのサブミラーはミラーの一部ではなくなるので、ミラーを通して行われたメタデバイスに対する読み取りや書き込みは、ミラーを通して行われなくなります。1 つだけ存在するサブミラーを切断することは

できません。保守の必要があると報告されている (metastat によって) スライスが含まれているサブミラーは、-f (強制) フラグを使用しないと切断することができません。

metadetach は、トランスからロギングデバイスを切断することもできます。この手順は、トランスボリュームをクリアする前に実行する必要があります。トランスメタデバイスは UFS ロギングによって置き換えられました。既存のトランスデバイスはロギングを行いません。その元となるデバイスにデータを直接渡します。UFS ロギングについての詳細は、mount_ufs(1M) を参照してください。

-f (強制) フラグを使用しない限り、使用中のトランスデバイスからロギングデバイスを切断することはできません。強制フラグを使用しても、ロギングデバイスは、トランスがアイドルになるまで、実際に切断されません。ロギングデバイスを切断するまで、トランスは切断中状態 (metastat) です。

オプション

次のオプションのうち、-h 以外のオプションを実行するには、スーパーユーザーになる必要があります。

次のオプションを指定できます。

-A *alignment*
ソフトウェアパーティション境界整列の値を設定します。このオプションは、ソフトパーティションの開始オフセットを指定することが重要であるときに使用します。このオプションを指定すると、メタデバイスとそれを構成する物理デバイスの 2 つのアドレス空間の間で、同じデータ整列が行われます。

たとえば、ハードウェアデバイスでチェックサムを実行している場合、そのデバイスへの入出力要求は Solaris ボリュームマネージャによって分割されるべきではありません。この場合、ハードウェア構成から取得した値を境界整列の値として使用します。このオプションをソフトウェア入出力負荷と組み合わせて使用すると、境界整列の値はアプリケーションの入出力負荷に対応します。これによって、入出力が不必要に分割されて、パフォーマンスに影響が出ることを防ぐことができます。

-f
保守を必要とするコンポーネントまたは使用中のコンポーネントが含まれているメタデバイスを強制的に切断します。このオプションを使用できるのは、ミラーが metareplace(1M) で修正できる保守状態であるときだけです。ミラーが metasync(1M) だけで修正できる保守状態である場合 (metastat(1M) の出力で判断)、metadetach -f ではまったく効果がありません。これは、メタデバイスの 1 つを切り離す前には、ミラーを再同期する必要があるためです。

-h
使用方法に関するメッセージを表示します。

-i *interlace*
ストライプの飛び越しの値を指定します。size 数値の後ろに「k」(キロバイト)、「m」(メガバイト)、「b」(512 バイトブロック) の単位を付けて指定しま

す。これらの単位は、大文字でも小文字でも構いません。*size* を指定しないと、メタデバイスの最後のストライプの飛び越しの値がデフォルトサイズとして使用されます。あるストライプの飛び越しの値を変更すると、その変更内容が以降のすべてのストライプに適用されます。

`-s` *setname*
`metattach` コマンドまたは `metadetach` コマンドを実行するディスクセットの名前を指定します。`-s` オプションを使用すると、指定したディスクセット内でコマンドが実行されます。このオプションを使用しない場合は、ローカルのメタデバイスに対してコマンドが実行されます。

オペランド　次のオペランドを指定できます。

component
`/dev/dsk/c0t0d0s2` などのディスクドライブ上の物理スライス (パーティション) を連結、ストライプ、連結ストライプ、RAID5 メタデバイスに追加させるための論理名。

concat/stripe
連結、ストライプ、連結ストライプのいずれかのメタデバイス名。

log
トランスメタデバイスに接続されるロギングデバイスのメタデバイス名。

metadevice
ミラーにサブミラーとして接続されるメタデバイス名。このメタデバイスは、`metainit` コマンドで事前に作成されている必要があります。

mirror
ミラーの名前。

RAID
RAID5 メタデバイスのメタデバイス名。

size | *all*
ソフトパーティションに追加する領域のサイズ。サイズの単位としては、K バイトには `K` または `k` を、M バイトには `M` または `m` を、G バイトには `G` または `g` を、T バイトには `T` または `t` を、ブロック (セクター) には B または `b` を指定できます。すべての値は 2 のべき乗で表現されます。大文字と小文字のオプションは同じです。使用できる値は整数値だけです。ソフトパーティションが、その元になるボリューム上の使用可能なすべての領域に渡って拡張されるように指定するには、`all` というリテラルを指定します。

softpart
既存のソフトパーティションのメタデバイス名。

submirror
ミラーから切断されるサブミラーのメタデバイス名。

trans
(マスターデバイスまたはロギングデバイスではなく) トランスメタデバイスのメタデバイス名。

使用例

例 1 メタデバイスへの新しいスライスの連結

この例は、新しい単一のスライスを既存のメタデバイス d8 に連結します。その後、growfs(1M) コマンドを使用してファイルシステムを拡張できます。

```
# metattach d8 /dev/dsk/c0t1d0s2
```

例 2 トランスメタデバイスからのロギングデバイスの切断

この例は、トランスメタデバイス d9 からロギングデバイスを切断します。ロギングデバイスは 1 台だけなので、指定する必要はありません。

```
# metadetach d9
```

例 3 RAID5 メタデバイスの拡張

この例は、RAID5 メタデバイス d45 にもう 1 つのスライスを追加することで RAID5 メタデバイスを拡張します。

```
# metattach d45 /dev/dsk/c3t0d0s2
```

追加スライスを RAID5 メタデバイスに追加すると、追加領域はデータ専用になります。新しいパリティーブロックは割り当てられません。ただし、追加されたスライス上のデータは、全体のパリティー計算に含まれるので、単一のデバイス障害に対しては保護されます。

例 4 ソフトパーティションの拡張

以下の例は、元になるデバイス上の使用可能なすべての領域を接続して、ソフトパーティション d42 を拡張します。

```
# metattach d42 all
```

ソフトパーティションに領域を追加するとき、追加の領域はそのスライスで利用可能な任意の領域から取得されるため、既存のソフトパーティションとは隣接していない可能性があります。

例 5 2 面ミラーへの領域の追加

この例では、各サブミラーにスライスを追加して、2 面ミラーに領域を追加します。その後、growfs(1M) コマンドを使用してファイルシステムを拡張できます。

```
# metattach d9 /dev/dsk/c0t2d0s5
# metattach d10 /dev/dsk/c0t3d0s5
```

例 5　2 面ミラーへの領域の追加　　(続き)

この例は、ミラーをその元になるデバイスのサイズまで拡張します。

```
# metattach d11
```

この例は、デバイスの UFS のサイズを増やして、その領域を使用できるようにします。

```
# growfs -M /export /dev/md/dsk/d11
```

例 6　ミラーからのサブミラーの切断

この例は、サブミラー d2 をミラー d4 から切断します。

```
# metadetach d4 d2
```

例 7　4 つのスライスのメタデバイスへの追加

この例は、既存のメタデバイス d9 に 4 つのスライスを追加します。その後、growfs(1M) コマンドを使用してファイルシステムを拡張できます。

```
# metattach d9 /dev/dsk/c0t1d0s2 /dev/dsk/c0t2d0s2 \
     /dev/dsk/c0t3d0s2 /dev/dsk/c0t4d0s2
```

例 8　ソフトパーティション境界整列の値の設定

以下の例は、ソフトパーティションを拡張するときに、ソフトパーティション境界整列の値を 1M バイトに設定する方法を示しています。

```
# metattach -s red -A 2m d13 1m
```

終了ステータス

次の終了ステータスが返されます。

0　　正常終了。

>0　　エラーが発生した。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

mdmonitord(1M), metaclear(1M), metadb(1M), metahs(1M), metainit(1m), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

『Solaris ボリュームマネージャの管理』

警告

この節では、1T バイトを超えるデバイスと多面ミラーに関する警告について説明します。

1T バイトを超えるデバイスとボリューム

32 ビットカーネルの Solaris オペレーティングシステムを実行する予定の場合、または、Solaris 9 4/03 より前のバージョンの Solaris オペレーティングシステムを使用する予定の場合、大型ボリューム (つまり、1T バイトを超えるボリューム) を作成してはいけません。

多面ミラー

ミラーからサブミラーを切断すると、メタデバイス上のデータが `metadetach` の実行前にミラー上に存在したデータと一致しないことがあります。特に、`-f` オプションが必要なときには、メタデバイスとミラーのデータが一致しないことがよくあります。

注意事項

トランスメタデバイスは UFS ロギングによって置き換えられました。既存のトランスデバイスはロギングを行いません。その元となるデバイスにデータを直接渡します。UFS ロギングについての詳細は、mount_ufs(1M) を参照してください。

名前　mkfile – ファイルの作成

形式　`mkfile [-nv] size [g | k | b | m] filename...`

機能説明　`mkfile` は、NFS マウント上のファイル、あるいはローカルファイルをスワップ領域として使用するのに適した 1 つまたは複数のファイルを作成します。root ユーザーとして `mkfile()` を実行すると、デフォルトでスティッキービットが設定され、ファイルはゼロでパディングされます。root 以外のユーザーは、`mkfile()` を実行するときに、`chmod`(1) を使用して、手動でスティッキービットを設定する必要があります。デフォルトでは `size` の単位はバイトですが、`g`、`k`、`b`、または `m` のフラグを使用すると、ギガバイト、キロバイト、ブロック、またはメガバイト単位でも指定できます。

オプション

`-n`　空の *filename* を作成します。サイズは出力されますが、ディスクブロックはデータが書き込まれるまで割り当てられません。このオプションで作成したファイルは、ローカルの UFS マウント上でスワップできません。

`-v`　詳細表示。作成したファイルの名前とサイズを報告します。

使用法　2G バイト (2^{31} バイト) 以上のファイルを検出した場合の `mkfile` の動作については、`largefile`(5) のマニュアルページを参照してください。

属性　次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目　`chmod`(1), swap(1m), `attributes`(5), `largefile`(5)

名前　mkfs – ファイルシステムの構築

形式　`mkfs [-F FSType] [generic_options] [-o FSType-specific_options] raw_device_file [operands]`

機能説明　`mkfs` ユーティリティは、`-F` *FSType* で指定したファイルシステムタイプに固有の `mkfs` モジュールを呼び出すことによって、*raw_device_file* 上にファイルシステムを構築します。

注: UFS ファイルシステムを作成するときは、通常 newfs(1M) コマンドを使用します。

generic_options はファイルシステムの種類によらない共通のオプションです。*FSType-specific_options* は、*keyword=value* の組みを、空白を入れずにコンマで区切ったリストで、*FSType* に固有です。*raw_device_file* は、ファイルシステムを作成するディスクパーティションです。この引数は必須で、(もしあれば) *specific_options* のすぐ後に指定する必要があります。*operands* は *FSType* に固有の引数です。詳細については、`mkfs` の *FSType* に固有なマニュアルページを参照してください (たとえば、`mkfs_ufs`(1M))。

オプション　`mkfs` の汎用オプションは次のとおりです。

`-F`　構築する *FSType* を指定します。`-F` を指定しないと、*FSType* は `/etc/vfstab` から一致する *raw_device_file* を探すことによって、あるいは、`/etc/default/fs` に指定されているデフォルトを調べることによって決定されます。

`-V`　コマンド行全体をエコーしますが、コマンドは実行しません。エコーされたコマンド行には、ユーザーが指定したオプションと引数、さらに、`/etc/vfstab` または `/etc/default/fs` から得られた情報が追加されます。このオプションは、コマンド行を確認および検証するときに使用します。

`-m`　ファイルシステムを作成するのに使用したコマンド行を表示します。ファイルシステムは、あらかじめ存在していなければなりません。このオプションは、ファイルシステムを構築したときに使用したパラメータを調べるのに使用します。

`-o`　*FSType* に固有なオプションを指定します。ファイルシステムタイプ (*FSType*) に固有な `mkfs` モジュールのマニュアルページを参照してください。

使用法　2G バイト (2^{31} バイト) 以上のファイルを検出した場合の `mkfs` の動作については、`largefile`(5) のマニュアルページを参照してください。

ファイル

`/etc/default/fs`　デフォルトのファイルシステムタイプ。デフォルト値は、`/etc/default/fs` 内で次のように設定されています。(例: `LOCAL=ufs`)

`LOCAL`　*FSType* を指定しない場合に、コマンドがデフォルトで使用するパーティション

`/etc/vfstab` 各ファイルシステム用のデフォルトのパラメータリスト

属性

次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

`mkfs_ufs`(1M), newfs(1M), `vfstab`(4), `attributes`(5), `largefile`(5)

`mkfs` の FSType に固有なモジュールのマニュアルページ

注意事項

このコマンドは、すべての *FSType* で使用できるわけではありません。

`lofiadm` を使用すると、`mkfs` コマンドには raw デバイスのように見えるファイルを作成できます。次に、`mkfs` コマンドを使用すると、そのデバイス上にファイルシステムを作成できます。`mkfs_ufs`(1M) と `mkfs_pcfs`(1M) を使用することにより、`lofiadm` で作成したデバイス上に UFS と PC (FAT) ファイルシステムを作成する例については、lofiadm(1m) のマニュアルページを参照してください。

名前　modinfo – ロードされているカーネルモジュールについての情報の表示

形式　`/usr/sbin/modinfo [-c] [-w] [-i module-id]`

機能説明　`modinfo` ユーティリティは、ロードされているモジュールに関する情報を表示します。表示形式は次のとおりです。

Id Loadaddr Size Info Rev Module Name

Id はモジュール ID、*Loadaddr* はテキスト開始位置のアドレス (16 進表記)、*Size* はテキストとデータと bss のバイト単位の合計サイズ (16 進表記)、*Info* はモジュール固有の情報、*Rev* はロード可能モジュールシステムのリビジョン番号、そして *Module Name* はモジュールのファイル名と内容の説明を表します。

モジュール固有の情報として表示される内容は、モジュールの種類により異なります。すなわち、ドライバであればブロックメジャー番号とキャラクタメジャー番号、システムコールであればシステムコール番号、その他の種類は指定されていません。

オプション　次のオプションを指定できます。

`-c` ロードされているモジュールのインスタンス数と、モジュールの現在の状態を表示します。

`-i` *module-id* 指定したモジュールに関する情報のみを表示します。

`-w` モジュールに関する情報を、80 文字の位置で切り落としません。

使用例　例 1　モジュールの状態の表示

次の例では、モジュール 2 に関する情報を表示しています。

```
example% modinfo -i 2
Id   Loadaddr   Size Info Rev Module Name
 2    ff08e000   1734   -   1  swapgeneric (root and swap configuration)
```

例 2　カーネルモジュールの状態の表示

次の例では、いくつかのカーネルモジュールの状態を表示しています。

```
example% modinfo
Id Loadaddr   Size Info Rev Module Name
 2  ff08e000   1734   -   1  swapgeneric
 4  ff07a000   3bc0   -   1  specfs (filesystem for specfs)
 6  ff07dbc0   2918   -   1  TS (time sharing sched class)
 7  ff0804d8    49c   -   1  TS_DPTBL (Time sharing dispatch table)
 8  ff04a000  24a30   2   1  ufs (filesystem for ufs)
 9  ff080978   c640 226   1  rpcmod (RPC syscall)
 9  ff080978   c640   -   1  rpcmod (rpc interface str mod)
10  ff08cfb8  2031c   -   1  ip (IP Streams module)
```

例2 カーネルモジュールの状態の表示 (続き)

```
10  ff08cfb8  2031c   2   1  ip (IP Streams device)
```

例3 -cオプションの使用

-cオプションを指定してmodinfoコマンドを実行すると、読み込まれているモジュールのインスタンスの数と、そのモジュールの現在の状態が表示されます。

```
example% modinfo -c
Id    Loadcnt Module Name                          State
  1          0 krtld                          UNLOADED/UNINSTALLED
  2          0 genunix                        UNLOADED/UNINSTALLED
  3          0 platmod                        UNLOADED/UNINSTALLED
  4          0 SUNW,UltraSPARC-IIi            UNLOADED/UNINSTALLED
  5          0 cl_bootstrap                   UNLOADED/UNINSTALLED
  6          1 specfs                         LOADED/INSTALLED
  7          1 swapgeneric                    UNLOADED/UNINSTALLED
  8          1 TS                             LOADED/INSTALLED
  9          1 TS_DPTBL                       LOADED/INSTALLED
 10          1 ufs                            LOADED/INSTALLED
 11          1 fssnap_if                      LOADED/INSTALLED
```

属性

次の属性についてはattributes(5)のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	開発中

関連項目

modload(1M), modunload(1M), attributes(5)

名前

modload – カーネルモジュールのロード

形式

```
modload  [-p] [-e exec_file] filename
```

機能説明

`modload` コマンドは、*filename* で示すロード可能モジュールを、稼動中のシステムにロードします。

filename は、`ld -r`で生成されたオブジェクトファイルを示します。 *filename* が絶対パス名の場合、 その絶対パスが指定するファイルがロードされます。*filename* の先頭文字がスラッシュ (/) ではない場合、 `-p` オプションが指定されていないと、現在のディレクトリに相対するパスを使って *filename* をロードします。

カーネルのモジュールパス `modpath` 変数は、`/etc/system` ファイルを使って設定できます。この変数のデフォルト値は、オペレーティングシステムをロードした時のパスです。通常は、`/kernel /usr/kernel` となります。したがって、次のように入力した場合、カーネルは `./drv/foo` ファイルを探します。

```
example# modload drv/foo
```

また、 次のように入力した場合には、カーネルはまず `/kernel/drv/foo` を探し、存在しなければ `/usr/kernel/drv/foo` を探します。

```
example# modload –p drv/foo
```

オプション

次のオプションを試用できます。

`-p`
: モジュールを検索するパスとして、カーネルの内部 modpath 変数を使用します。

`-e` *exec_file*
: モジュールのロードが正常に終了した後で実行すべき シェルスクリプトまたは実行可能イメージの名前を指定します。そのスクリプトまたはイメージにはいくつかの引数が渡されます。第 1 引数は常にモジュール ID (10 進数) です。 他の引数は、モジュールの種類により異なります。 ドライバには、ブロックメジャー番号とキャラクタメジャー番号、システムコールにはシステムコール番号、その他のモジュールタイプにはそれぞれにあったカーネルテーブルへのインデックスが指定されます (modinfo(1M) を参照) 。

属性

次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

`ld`(1), `add_drv`(1M), `kernel`(1M), modinfo(1M), modunload(1M), `system`(4), `attributes`(5), `modldrv`(9S), `modlinkage`(9S), `modlstrmod`(9S), `module_info`(9S)

『Writing Device Drivers』

注意事項

デバイスドライバを追加するには、`modload`ではなく `add_drv`(1M) を使います。デバイスドライバの追加方法については、『Writing Device Drivers』 を参照してください。

名前　modunload – モジュールのアンロード

形式　`modunload -i` *module_id* `[-exec` *exec_file*`]`

機能説明　`modunload` は、稼動中のシステムからロード可能モジュールをアンロードします。*module_id* はアンロードするモジュールの ID で、これは modinfo(1M) の出力情報中に得られる値と同じです。ID として `0` を指定した場合、自動ローディングされたモジュールのうちアンロード可能なものがすべてアンロードされます。modload(1M) を使ってロードしたモジュールは対象とはなりません。

オプション　次のオプションを指定できます。

`-e` *exec_file*　モジュールをアンロードする前に実行すべきシェルスクリプトまたは実行可能イメージの名前を指定します。 そのスクリプトまたはイメージにはいくつかの引数が渡されます。第 1 引数は常にモジュール ID (10 進数) です。 他の 2 つの引数は、モジュールの種類により異なります。ロード可能ドライバの場合は、ブロックメジャー番号が第 2 引数です。ロード可能システムコールの場合は、システムコール番号が第 2 引数です。ロード可能 exec クラスの場合は、`execsw` テーブルへのインデックスが第 2 引数です。ロード可能ファイルシステムの場合は、`vfssw` テーブルへのインデックスが第 2 引数です。ロード可能ストリームモジュールの場合は、`fmodsw` テーブルへのインデックスが第 2 引数です。ロード可能スケジューリングクラスの場合は、クラス配列へのインデックスが第 2 引数です。 該当しない引数の値としては、–1 が渡されます。

`-i` *module_id*　アンロードすべきモジュールを指定します。

属性　次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目　modinfo(1M), modload(1M), `update_drv`(1M), `attributes`(5)

注意事項　`modunload` コマンドは、ドライバモジュールに対してしばしば関連するドライバ構成ファイルをシステムに再度読み込ませる目的で、使用されてきました。この方法は Solaris 9 でも機能しますが、将来のリリースではサポートされなくなる可能性があります。ドライバ構成ファイルの再読み込みには、`update_drv`(1M) コマンドを使用することをお勧めします。

名前

mount, umount – ファイルシステムとリモート資源のマウントおよびマウント解除

形式

```
mount [-p | -v]

mount [-F FSType] [generic_options] [-o specific_options]
 [-O] special | mount_point

mount [-F FSType] [generic_options] [-o specific_options]
 [-O] special mount_point

mount -a [-F FSType] [-V] [current_options] [-o specific_options]
 [mount_point]...

umount [-f] [-V] [-o specific_options] special | mount_point

umount -a [-f] [-V] [-o specific_options] [mount_point]...
```

機能説明

mount は、ファイルシステムをファイルシステム階層内の *mount_point* (ディレクトリのパス名) に継ぎ足します。マウント操作を実行する前に *mount_point* の下に存在したファイルとディレクトリは、ファイルシステムのマウントを解除するまで見えなくなります。

umount は、現在マウントされているファイルシステムをマウント解除します。マウントを解除するファイルシステムは *mount_point* または *special* (ファイルシステムが存在しているデバイス) のどちらかで指定できます。

現在マウントされているファイルシステムのテーブルは、マウント済みファイルシステム情報ファイルにあります。このファイルは通常、/etc/mnttab 上にマウントされているファイルシステムによって提供されます。マウント済のファイルシステムについての情報は、mnttab(4) に格納されています。ファイルシステムをマウントすると、マウントテーブルにエントリが追加されます。(umount で) ファイルシステムをマウント解除すると、テーブルからエントリが削除されます。

special および *mount_point* 両方の引数と -F オプションを指定した場合、mount は *special* 以外のすべての引数の妥当性を検査し、適切な *FSType* に固有な mount モジュールを呼び出します。引数なしで呼び出した場合、mount は、マウントテーブル /etc/mnttab に記録されているすべてのマウント済みファイルシステムをリストします。引数を一部だけ、たとえば、*special* か *mount_point* のどちらか 1 つだけを指定した場合や、*special* と *mount_point* の両方を指定したが、*FSType* は指定しない場合、mount は /etc/vfstab を調べて、指定されていない引数を補完するエントリを探します。そのようなエントリが見つからず、また、*special* 引数が / で始まる場合、/etc/default/fs に指定されているデフォルトのローカルのファイルシステムタイプが使用されます。それ以外の場合、デフォルトのリモートファイルシステムタイプが使用されます。デフォルトのリモートファイルシステムタイプは、/etc/dfs/fstypes ファイルの最初のエントリによって決定されます。指定されていない引数を補完した後、mount は *FSType* に固有な mount モジュールを呼び出します。

mount または umount を使用してファイルシステムをマウントまたはマウント解除できるのはスーパーユーザーだけです。ただし、マウントされているファイルシステムとリソースを一覧表示するだけであれば、だれでも mount コマンドを使用することができます。

オプション　次のオプションを指定できます。

-F *FSType*
操作の対象となる *FSType* を指定します。*FSType* は、明示的に指定するか、あるいは、/etc/vfstab から決定できるか、/etc/default/fs または /etc/dfs/fstypes を調べることで決定できるものでなければなりません。

-a [*mount_points...*]
可能であれば、複数の mount または umount 操作を同時に実行します。

マウントポイントを指定しないと、mount は /etc/vfstab において mount at boot フィールドが yes に設定されているすべてのファイルシステムをマウントします。マウントポイントを指定すると、/etc/vfstab の mount at boot フィールドは無視されます。

マウントポイントを指定すると、umount は指定されたマウントポイントだけをマウント解除します。マウントポイントを指定しないと、umount は /etc/mnttab にあるすべてのファイルシステムのマウントを解除します。ただし、/、/usr、/var、/var/adm、/var/run、/proc、/dev/fd、および /tmp など、システムに必須のファイルシステムは除きます。

-f
ファイルシステムのマウントを強制的に解除します。

このオプションを指定しないと、umount は、ファイルシステム上のファイルがビジー状態である場合、ファイルシステムをマウント解除しません。このオプションを指定すると、オープンしているファイルのデータが失われる可能性があります。ファイルシステムをマウント解除した後にプログラムがファイルにアクセスしようとすると、エラー (EIO) が返されます。

-p
マウントされているファイルシステムの一覧を /etc/vfstab 形式で表示します。このオプションを指定するときは他のオプションを指定してはなりません。「使用上の留意点」を参照してください。

-v
マウントされているファイルシステムの一覧を詳細形式で表示します。このオプションを指定するときは他のオプションを指定してはなりません。

-V
コマンド行全体をエコーしますが、コマンドは実行しません。エコーされたコマンドには、ユーザーが指定したオプションと引数、さらに、/etc/mnttab から得られた情報が追加されます。このオプションは、コマンド行を確認および検証するときに使用します。

generic_options

ほとんどの *FSType* に固有なコマンドモジュールで使用できる共通のオプションです。次のオプションを指定できます。

`-m` ファイルシステムをマウントしますが、`/etc/mnttab` にエントリを作成しません。

`-g` ファイルシステムを広域的にマウントします。クラスタ化されているシステムでは、クラスタ内にあるすべてのノード上でファイルシステムを広域的にマウントします。クラスタ化されていないシステムでは、このオプションは何の効果もありません。

`-o` サブオプションとキーワード属性の組の並を、空白を入れずにコンマで区切った形式で、*FSType* 固有のオプションを指定します。これらのオプションはコマンドの *FSType* に固有なモジュールによって解釈されます (mount_ufs(1M) のマニュアルページを参照)。`/etc/vfstab` にエントリがあるファイルシステムで `-o` オプションを指定すると、`/etc/vfstab` 内でそのファイルシステムに指定されている `mount` コマンドのオプションはすべて無視されます。

次のオプションを指定できます。

`devices | nodevices` デバイス固有ファイルを開くことを許可または禁止します。デフォルトは `devices` です。

`devices` と一緒に `nosuid` を使用すると、その動作は `nosuid` と同等になります。

`exec | noexec` ファイルシステムでプログラムを実行することを許可または禁止します。ファイルシステム内のファイルに対して、`PROT_EXEC` 付きの `mmap`(2) を許可または禁止します。デフォルトは `exec` です。

`nbmand | nonbmand` 当該ファイルシステムに対して、非ブロック必須ロック意味論を許可または禁止します。デフォルトでは、非ブロック必須ロックは無効です。

ファイルシステムが `nbmand` オプションでマウントされている場合、アプリケーションは `fcntl`(2) インタフェースを使用して、非ブロック必須ロックをファイルにかけることができます。すると、システムは非ブロック必須ロック意味論を実施します。このオプションを有効にした場合、標準に適合するアプリケーションに予期せぬエラーが発生する可能性があります。

nbmand オプションは、/、/var、および /usr に使用してはなりません。

remount オプションを使用して、ファイルシステムの nbmand 設定を変更してはなりません。nbmand オプションはグローバルオプションとは相互排他的です。-g を参照してください。

ro | rw
: 読み取り専用または読み書きを指定します。デフォルトは rw です。

setuid | nosetuid
: setuid または setgid の実行を許可または禁止します。デフォルトは setuid です。

  setuid と一緒に nosuid を使用すると、その動作は nosuid と同等になります。

  nosuid は、nosetuid と nodevices と同等です。suid または nosuid を setuid または nosetuid および devices または nodevices と一緒に組み合わせた場合、ほとんどの制限オプションが有効になります。

  root= オプションを指定した NFS を使用して、ファイルシステムを共有している場合、このオプションを指定することを強く推奨します。このオプションを指定しないと、NFS クライアントは setuid プログラムをサーバーに追加したり、セキュリティホールを開けるデバイスを作成する可能性があります。

suid | nosuid
: setuid または setgid の実行を許可または禁止します。デフォルトは suid です。このオプションは、また、ファイルシステム内にあらわれる任意のデバイス固有エントリを開くことを許可または禁止します。

  nosuid は、nosetuid と nodevices と同等です。suid または nosuid を setuid または nosetuid および devices または nodevices と一緒に組み合わせた場合、ほとんどの制限オプションが有効になります。

  root= オプションを指定した NFS を使用して、ファイルシステムを共有している場合、このオプションを指定することを強く推奨します。このオ

プションを指定しないと、NFS クライアントは setuid プログラムをサーバーに追加したり、セキュリティホールを開けるデバイスを作成する可能性があります。

-O　オーバーレイマウント。既存のマウントポイント上にファイルシステムをマウントできます。これより既存のマウントポイントのファイルシステムにはアクセスできなくなります。このフラグを指定せずに既存のマウントポイント上にファイルシステムをマウントしようとすると、マウントは失敗して、device busy というエラーメッセージが表示されます。

-r　ファイルシステムを読み取り専用でマウントします。

使用法

2G バイト (2^{31} バイト) 以上のファイルを検出した場合の mount と umount の動作については、largefile(5) のマニュアルページを参照してください。

ファイル

/etc/mnttab　マウントされているファイルシステムのテーブル

/etc/default/fs　デフォルトのローカルのファイルシステムタイプ。デフォルト値は、/etc/default/fs 内で次のように設定されています。(例: LOCAL=ufs)

LOCAL:　*FSType* を指定しない場合に、コマンドがデフォルトで使用するパーティション

/etc/vfstab　各ファイルシステム用のデフォルトのパラメータリスト

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

mount_cachefs(1M), mount_hsfs(1M), mount_nfs(1M), mount_pcfs(1M), mount_tmpfs(1M), mount_ufs(1M), mountall(1M), umountall(1M), fcntl(2), mmap(2), mnttab(4), vfstab(4), attributes(5), largefile(5), lofs(7FS), pcfs(7FS)

注意事項

ファイルシステムがマウントされるディレクトリがシンボリックリンクの場合、ファイルシステムは、シンボリックリンク自身ではなく、シンボリックリンクが参照するディレクトリ上にマウントされます。

使用上の留意点

mount -p の出力は、cachefs については正しくありません。

名前

mount_ufs – mount ufs file systems

形式

```
mount -F ufs [generic_options] [-o specific_options] [-O] special | mount_point
mount -F ufs [generic_options] [-o specific_options] [-O] special mount_point
```

機能説明

The `mount` utility attaches a `ufs` file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *specific_options*. See mount(1M).

If *special* and *mount_point* are specified without any *specific_options*, the default is `rw`.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

オプション

See mount(1M) for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options*

Specify `ufs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`dfratime` | `nodfratime`

By default, writing access time updates to the disk may be deferred (`dfratime`) for the file system until the disk is accessed for a reason other than updating access times. `nodfratime` disables this behavior.

If power management is enabled on the system, do not set `nodfratime` unless `noatime` is also set. If you set `nodfratime` without setting `noatime`, the disk is spun up every time a file within a file system on the disk is accessed - even if the file is not modified.

`forcedirectio` | `noforcedirectio`	If `forcedirectio` is specified and supported by the file system, then for the duration of the mount, forced direct I/O will be used. If the filesystem is mounted using `forcedirectio`, data is transferred directly between user address space and the disk. If the filesystem is mounted using `noforcedirectio`, data is buffered in kernel address space when data is transferred between user address space and the disk. `forcedirectio` is a performance option that is of benefit only in large sequential data transfers. The default behavior is `noforcedirectio`.
`global` | `noglobal`	If `global` is specified and supported on the file system, and the system in question is part of a cluster, the file system will be globally visible on all nodes of the cluster. If `noglobal` is specified, the mount will not be globally visible. The default behavior is `noglobal`.
`intr` | `nointr`	Allow (do not allow) keyboard interrupts to kill a process that is waiting for an operation on a locked file system. The default is `intr`.
`largefiles` | `nolargefiles`	If `nolargefiles` is specified and supported by the file system, then for the duration of the mount it is guaranteed that all regular files in the file system have a size that will fit

in the smallest object of type `off_t` supported by the system performing the mount. The mount will fail if there are any files in the file system not meeting this criterion. If `largefiles` is specified, there is no such guarantee. The default behavior is `largefiles`.

If `nolargefiles` is specified, `mount` will fail for `ufs` if the file system to be mounted has contained a large file (a file whose size is greater than or equal to 2 Gbyte) since the last invocation of `fsck` on the file system. The large file need not be present in the file system at the time of the mount for the mount to fail; it could have been created previously and destroyed. Invoking `fsck` (see `fsck_ufs`(1M)) on the file system will reset the file system state if no large files are present. After invoking `fsck`, a successful mount of the file system with `nolargefiles` specified indicates the absence of large files in the file system; an unsuccessful mount attempt indicates the presence of at least one large file.

`logging` | `nologging`

If `logging` is specified, then logging is enabled for the duration of the mounted file system. Logging is the process of storing transactions (changes that make up a complete UFS operation) in a log before the transactions are

applied to the file system. Once a transaction is stored, the transaction can be applied to the file system later. This prevents file systems from becoming inconsistent, therefore reducing the possibility that `fsck` might run. And, if `fsck` is bypassed, logging generally reduces the time required to reboot a system.

The default behavior is `logging` for all UFS file systems.

The log is allocated from free blocks in the file system, and is sized approximately 1 Mbyte per 1 Gbyte of file system, up to a maximum of 64 Mbytes.

Logging is enabled on any UFS file system, including root (`/`), except under the following conditions:

- When logging is specifically disabled.
- If there is insufficient file system space for the log. In this case, the following message is displayed and file system is still mounted:

```
# mount /dev/dsk/c0t4d0s0 /mnt
  /mnt: No space left on device
  Could not enable logging for /mnt on  /dev/dsk/c0t4d0s0.
```

The log created by UFS logging is continually flushed as it fills up. The log is totally flushed when the file system is unmounted or as a result of the `lockfs -f` command.

`m`	Mount the file system without making an entry in `/etc/mnttab`.
`noatime`	By default, the file system is mounted with normal access time (`atime`) recording. If `noatime` is specified, the file system will ignore access time updates on files, except when they coincide with updates to the `ctime` or `mtime`. See `stat`(2). This option reduces disk activity on file systems where access times are unimportant (for example, a Usenet news spool). `noatime` turns off access time recording regardless of `dfratime` or `nodfratime`. The POSIX standard requires that access times be marked on files. `-noatime` ignores them unless the file is also modified.
`onerror` = *action*	This option specifies the action that UFS should take to recover from an internal inconsistency on a file system. Specify *action* as `panic`, `lock`, or `umount`. These values cause a forced system shutdown, a file system lock to be applied to the file system, or the file system to be forcibly unmounted, respectively. The default is `panic`.
`quota`	Quotas are turned on for the file system.
`remount`	Remounts a file system with a new set of options. All options

not explicitly set with `remount` revert to their default values.

`rq` Read-write with quotas turned on. Equivalent to `rw, quota`.

`-O` Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error “device busy”.

使用例

例 1 Turning Off (and On) Logging

The following command turns off logging on an already mounted file system. The subsequent command restores logging.

```
# mount -F ufs -o remount,nologging /export
# (absence of message indicates success)
# mount -F ufs -o remount,logging /export
```

In the preceding commands, the `-F ufs` option is not necessary.

ファイル

`/etc/mnttab` table of mounted file systems

`/etc/vfstab` list of default parameters for each file system

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

関連項目

fsck(1m), `fsck_ufs`(1M), mount(1M), mountall(1M), `fcntl`(2), `mount`(2), `stat`(2), `mnttab`(4), `vfstab`(4), `attributes`(5), `fsattr`(5), `largefile`(5)

注意事項

Since the root (`/`) file system is mounted read-only by the kernel during the boot process, only the `remount` option (and options that can be used in conjunction with `remount`) affect the root (`/`) entry in the `/etc/vfstab` file.

名前

mountall, umountall – 複数のファイルシステムのマウントおよびマウント解除

形式

```
mountall  [-F FSType] [-l |  -r] [file_system_table]

umountall  [-k] [-s] [-F FSType] [-l |  -r] [-n]

umountall  [-k] [-s] [-h host] [-n]
```

機能説明

`mountall` は、ファイルシステムテーブルに指定されているファイルシステムをマウントするときに使用します。ファイルシステムテーブルは `vfstab`(4) 形式である必要があります。*file_system_table* を指定しない場合は、`/etc/vfstab` が使用されます。*file_system_table* として – を指定すると、`mountall` は標準入力からファイルシステムテーブルを読み取ります。`mountall` は、*file_system_table* において `mount at boot` フィールドが `yes` に設定されているファイルシステムだけをマウントします。

ファイルシステムテーブル内のファイルシステムごとに、次のロジックが実行されます。つまり、`/usr/lib/fs/`*FSType*`/fsckall` というファイル (*FSType* はファイルシステムのタイプ) が存在する場合、当該ファイルシステムをリストに保存しておき、後でまとめて `/usr/lib/fs/`*FSType*`/fsckall` スクリプトに引数として渡します。`/usr/lib/fs/`*FSType*`/fsckall` スクリプトは、引数リスト内にあるファイルシステムをすべて検査して、安全にマウントできるかどうかを決定します。*FSType* のファイルシステム用の `/usr/lib/fs/`*FSType*`/fsckall` スクリプトが存在しない場合、そのファイルシステムは fsck(1m) を使用して個々に検査されます。検査の結果、ファイルシステムがマウントできない状態であることが判明した場合、マウントを試行する前に `fsck` で修復されます。`fsckdev` フィールドのエントリが – であるファイルシステムは事前の検査なしにマウントされます。

`umountall` は、`root`、`/usr`、`/var`、`/var/adm`、`/var/run`、`/proc`、および `/dev/fd` を除き、マウントされているファイルシステムをすべてマウント解除します。*FSType* を指定すると、`mountall` と `umountall` のアクションは指定された *FSType* に制限されます。`-k` を指定した場合でも、`umountall` がビジー状態のファイルシステムをマウント解除するかどうかは保証されません。

オプション

次のオプションを指定できます。

`-F`
: マウントまたはマウント解除するファイルシステムの *FSType* を指定します。

`-h` *host*
: *host* からリモートでマウントされている、`/etc/mnttab` 内のファイルシステムをすべてマウント解除します。

`-k`
: `fuser -k` *mount-point* コマンドを使用します。詳細については、fuser(1M) のマニュアルページを参照してください。 `-k` オプションは `SIGKILL` シグナルをファイルを使用している各プロセスに送信します。このオプションはプロセスごとに終了シグナルを生成するので、終了メッセージがすぐに出力されないことがあります。`-k` を指定した場合でも、`umountall` がビジー状態のファイルシステムをマウント解除するかどうかは保証されません。

`-l` アクションをローカルのファイルシステムに制限します。

`-n` 指定したオプションで実行されるアクションをリスト表示します。ただし、実際にはアクションを実行しません。`-n` オプションを指定せずにコマンドを繰り返すと、コマンドを繰り返す間に `/etc/mnttab` ファイルは変更されなかったと仮定して、リストされたアクションが実行されます。

`-r` アクションをリモートのファイルシステムタイプに制限します。

`-s` 複数の `umount` 操作を同時に実行しないようにします。

ファイル

`/etc/mnttab`	マウント済みファイルシステムテーブル
`/etc/vfstab`	デフォルトのファイルシステムテーブル
`/usr/lib/fs/`*FSType*`/fsckall`	*FSType* タイプのファイルシステムをすべて検査するときに `mountall` が呼び出すスクリプト

属性

次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

fsck(1m), fuser(1M), mount(1M), `mnttab`(4), `vfstab`(4), `attributes`(5)

診断

ファイルシステムがマウント可能であり、クリーンな状態であれば、メッセージは出力されません。

エラーメッセージと警告メッセージは、fsck(1m) と mount(1M) の両方から送られます。

名前

newfs – UFS ファイルシステムの構築

形式

```
newfs  [-NSBTv] [mkfs-options] raw-device
```

機能説明

newfs は、ディスクパーティション上に UFS ファイルシステムを作成する mkfs(1M) プログラムをより使いやすいようにしたフロントエンドプログラムです。newfs は、最適化されたパラメータ値を算出して、mkfs を呼び出します。

対話形式で実行する場合 (つまり、標準入力が tty である場合)、newfs はファイルシステムを作成する前に確認のプロンプトを出力します。

-N オプションを指定せず、デバイスの i ノードがランダム化されていない場合、newfs は fsirand(1M) を呼び出します。

このコマンドを使用するにはスーパーユーザーであるか、またはこのコマンドを使用するのに適切なアクセス権を持っている必要があります。ただし、UFS ファイルシステムをフロッピーディスク上に作成するときは例外です (使用例 を参照)。

マルチテラバイトの UFS ファイルシステムの作成

マルチテラバイトの UFS ファイルシステムを作成する場合、次の制限があることに注意してください。

- *nbpi* の値は、特別にこれより大きな値を設定しない限り、1M バイトに設定されています。マルチテラバイトの UFS ファイルシステムでは、*nbpi* を 1M バイトより小さな値に設定することはできません。
- *fragsize* は *bsize* と等しい値に設定されます。
- デフォルトで、ロギングは有効です。

オプション

次のオプションを指定できます。

`-N`
: ファイルシステムを作成するときに使用するファイルシステムのパラメータを出力しますが、ファイルシステムの作成は行いません。この場合、fsirand(1M) は呼び出されません。

`-S`
: 指定された構成用のパラメータでファイルシステムを作成する場合に使用されるスーパーブロックを、可読形式で標準出力に出力します。

`-B`
: 指定された構成用のパラメータでファイルシステムを作成する場合に使用されるスーパーブロックを、バイナリ(機械可読) 形式で標準出力に出力します。

`-T`
: ファイルシステムの全体サイズが最終的に 1T バイトを超えることを許可するようにファイルシステムのパラメータを設定します。このオプションは、*fragsize* の値を *bsize* と同じに、また、-i オプションで値をさらに大きくするよう設定されていない場合は *nbpi* の値を 1M バイトに設定します。-f オプションまたは -i オプションを使用して、このオプションと両立しない *fragsize* または *nbpi* を指定すると、このオプションの *fragsize* または *nbpi* にユーザーが設定した値は無視されます。

デフォルトでは、このオプションを使用して作成されたすべてのファイルシステムにロギングが有効です。

`-v` 詳細表示。`newfs` はそのアクション (`mkfs` に渡されるパラメータも含む) を出力します。

mkfs-options デフォルトのパラメータを無効にするオプションは次のとおりです。

`-a` *apc* 不良ブロックの交換用に予約する、シリンダ当たりの代替セクター数。SCSI デバイスにのみ使用します。デフォルト値は `0` です。

このオプションは、EFI ラベルに準拠したディスクには適用されず、無視されます。

`-b` *bsize* ファイルシステムの論理ブロックのサイズ (バイト数)。4096 または 8192 で、デフォルトは 8192 です。sun4u アーキテクチャでは、4096 ブロックのサイズは使用できません。

`-c` *cgsize* シリンダグループ当たりのシリンダ数で、16 から 256 の値です。デフォルトは、ファイルシステムのセクター数を 1G バイトのセクター数で割ったものです。そのため、結果は常に 32 の倍数です。デフォルト値は常に 16 から 256 の範囲の値です。

`mkfs` は、この値を無効にすることがあります。詳細については、`mkfs_ufs`(1M) のマニュアルページを参照してください。

このオプションは、EFI ラベルに準拠したディスクには適用されず、無視されます。

`-C` *maxcontig* 連続して割り当てられる、1 つのファイルに属する論理ブロックの最大数。デフォルト値は、次の式で計算されます。

`maxcontig = ディスクドライブの最大転送サイズ / ディスクブロックサイズ`

ディスクドライブの最大転送サイズが決定できない場合は、カーネルのパラメータから次のようにして *maxcontig* のデフォルト値が計算されます。

`maxphys` が `ufs_maxmaxphys` (通常 1M バイト) より小さい場合、`maxcontig` は `maxphys` に設定されます。それ以外の場合は、`maxcontig` は `ufs_maxmaxphys` に設定されます。

maxcontig の値は、任意の正の整数に設定できます。

実際の値は、指定された値の中で、ハードウェアがサポートする最小のものになります。

このパラメータは tunefs(1M) を使用して後で変更できます。

-d *gap*
: 回転待ち。このオプションは廃止されました。この値は、入力した値に関わらず、常に 0 に設定されます。

-f *fragsize*
: ファイルに割り当てるディスク容量の最小値 (バイト数)。*bsize* を *fragsize* で割った数は 2 の乗数でなければなりません。次のようになります。

bsize / *fragsize* は、1、2、4、または 8。

つまり、論理ブロックのサイズが 4096 である場合、*fragsize* に有効な値は 512、1024、2048、および 4096 です。論理ブロックのサイズが 8192 である場合、有効な値は 1024、2048、4096、および 8192 です。デフォルト値は 1024 です。

1T バイトより大きなファイルシステム、または -T オプションを使用して作成したファイルシステムの場合、*fragsize* は強制的にブロックサイズ (*bsize*) と同じ値に変更されます。

-i *nbpi*
: i ノード当たりのバイト数で、ファイルシステム内にある i ノードの密度を指定します。この値をファイルシステムの合計サイズで割ることによって、作成できる i ノード数が決まります。

この値は、ファイルシステム内にあるファイルの予想平均サイズを反映します。i ノードを減らしたい場合は大きい値を指定します。逆に、i ノードを増やしたい場合は小さい値を指定します。*nbpi* のデフォルト値は次のとおりです。

ディスクサイズ	密度
1G バイト未満	2048
2G バイト未満	4096
3G バイト未満	6144
3G バイト〜 1T バイト	8192
1T バイトより大きい。または	

	-T を使用して作成した場合　1048576 ファイルシステムが growfs コマンドで拡張されている場合は、i ノードの数を増やすことができます。
-m *free*	ファイルシステム内で維持する空き領域の最小パーセンテージ。0% から 99% までで 0% と 99% も含みます。ユーザーはこの領域を設定できません。ファイルシステムがこのしきい値に達したら、ファイルシステムに書き込みを続けられるのはスーパーユーザーだけです。 デフォルト値は ((64 M バイト/パーティションのサイズ) * 100) で、小数点以下は切り捨てられます。1% 以上、10% 以内に制限されます。 このパラメータは tunefs(1M) コマンドを使用して後で変更できます。
-n *nrpos*	シリンダグループを分割する回転位置の数。デフォルト値は 8 です。 このオプションは、EFI ラベルに準拠したディスクには適用されず、無視されます。
-o *space* | *time*	ブロック割り当てにかかる時間を最小化するか、ディスク上の容量フラグメンテーションを最小化するかをファイルシステムに指示します。デフォルト値は time です。 このパラメータは tunefs(1M) コマンドを使用して後で変更できます。
-r *rpm*	分当たりのディスクの回転速度。デフォルト値はドライバまたはデバイスに固有です。 newfs コマンドの場合は *rpm*を、 mkfs コマンドの場合は *rps* を指定することに注意してください。 このオプションは、EFI ラベルに準拠したディスクには適用されず、無視されます。
-s *size*	ファイルシステムのサイズ (セクター単位)。デフォルトではパーティション全体を使用します。
-t *ntrack*	ディスク上のシリンダ当たりのトラック数。デフォルト値はディスクラベルから取得されます。

このオプションは、EFI ラベルに準拠したディスクには適用されず、無視されます。

オペランド　次のオペランドを指定できます。

raw-device　ファイルシステムを作成する /dev ディレクトリ上にある raw 特殊デバイスの名前 (たとえば、/dev/rdsk/c0t0d0s6)

使用法　2G バイト (2^{31} バイト) 以上のファイルを検出した場合の newfs の動作については、largefile(5) のマニュアルページを参照してください。

使用例

例 1　raw 特殊デバイスのパラメータを詳細に表示する

次の例は、raw 特殊デバイス c0t0d0s6 のパラメータを詳細に表示します。この例では、新しいファイルシステムの作成は行いません。

```
example# newfs -Nv /dev/rdsk/c0t0d0s6
mkfs -F ufs -o N /dev/rdsk/c0t0d0s6 1112940 54 1
5 8192 1024 16 10 60
2048 t 0 −1 8 /dev/rdsk/c0t0d0s6: 1112940 sectors in
1374 cylinders of 15 tracks, 54 sectors 569.8MB in 86 cyl
groups (16 c/g, 6.64MB/g, 3072 i/g) super-block backups
(for fsck -b #) at:
32, 13056, 26080, 39104, 52128, 65152, 78176, 91200, 104224, . . .
```

例 2　UFS ファイルシステムを作成する

次の例は、ボリュームマネージャが管理するフロッピーディスク上に UFS ファイルシステムを作成します。

```
example%  newfs /vol/dev/aliases/floppy0
newfs: construct a new file system /vol/dev/aliases/floppy0: (y/n)? y
/vol/dev/aliases/floppy0:    2880 sectors in 80 cylinders of 2 tracks,
    18 sectors   1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
        super-block backups (for fsck -F ufs -o b=#) at:
     32, 640, 1184, 1792, 2336, . . .
```

例 3　マルチテラバイトまでサイズを増大させる UFS ファイルシステムの作成

次の例は、サイズが増大して最終的にマルチテラバイトになる UFS ファイルシステムを作成します。

このコマンドは、ボリューム /dev/md/rdsk/d99 に 800G バイトのファイルシステムを作成します。

```
# newfs -T /dev/md/rdsk/d99
newfs: construct a new file system /dev/md/rdsk/d99: (y/n)? y
```

例 3　マルチテラバイトまでサイズを増大させる UFS ファイルシステムの作成　(続き)

```
   /dev/md/rdsk/d99: 1677754368 sectors in 45512 cylinders of
   144 tracks, 256 sectors
   819216.0MB in 1821 cyl groups (25 c/g, 450.00MB/g, 448 i/g) . . .
```

この後、ファイルシステムのボリュームサイズを増大させる場合は、`growfs` コマンドを使用してファイルシステムを拡張できます。この例では、ファイルシステムのサイズは 1.2T バイトに増大します。

```
# growfs -v /dev/md/rdsk/d99
/usr/lib/fs/ufs/mkfs -G /dev/md/rdsk/d99 2516631552 /dev/md/rdsk/d99:
   2516631552 sectors in 68268 cylinders of 144 tracks, 256  sectors
   1228824.0MB in 2731 cyl groups (25 c/g, 450.00MB/g, 448 i/g). . .
```

終了ステータス

次の終了ステータスが返されます。

`0`　正常終了

`1, 10`　使用方法のエラーまたは内部エラー。エラーを説明するメッセージが標準エラー出力に送られます。

`newfs` から呼び出された mkfs(1M) により、他の終了ステータスが返されることがあります。

属性

次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

fsck(1m), `fsck_ufs`(1M), `fsirand`(1M), mkfs(1M), `mkfs_ufs`(1M), `tunefs`(1M), `attributes`(5), `largefile`(5)

診断

`newfs: No such file or directory`　指定されたデバイスが存在していないか、ディスクパーティションが指定されていません。

special`: cannot open`　このコマンドを使用するには、デバイスへの書き込みアクセス権を持っていなければなりません。

名前

patchadd – Solaris オペレーティングシステムが稼働しているシステムへのパッチ適用

形式

```
patchadd  [-dun] [-G] [-B backout_dir] [-k keystore]
 [-P passwd] [-t] [-x proxy] {patch} |
{-M patch_location [patch_list]}
[-C net_install_image |  -R client_root_path |  -S service]

patchadd  -p [-C net_install_image |  -R client_root_path
 |  -S service]
```

機能説明

patchadd は、Solaris 2.*x* オペレーティング環境、および Solaris 2.*x* と互換性がある、2.*x* 以降の Solaris オペレーティング環境 (Solaris 10 など) を稼動しているシステムにパッチパッケージを適用します。patchadd は、Solaris 1.x システム用のパッチを適用するためには使用できません。patchadd を実行するには、スーパーユーザーになる必要があります。

patchadd コマンドには、次の使用方法があります。

- 1 つめの形式は、1 つまたは複数のパッチを、1 つのシステム、クライアント、サービス、またはネットインストールイメージの miniroot にインストールします。
- 2 つめの形式は、クライアント、サービス、またはネットインストールイメージの miniroot にインストールされたパッチを表示します。

Solaris オペレーティングシステムのバージョン 10 から、patchadd は、-M ソース指定子で指定された一連のパッチに対し、有効性チェックと依存性チェックを実行するようになりました。後述する「オペランド」の -M の説明を参照してください。

zones(5) に関しては、大域ゾーンで呼び出されると、デフォルトでは patchadd はすべてのゾーンにおいてすべての適切なパッケージにパッチを適用します。ゾーンがインストールされているシステム上のパッチ適用動作は次の要因によって異なります。

- -G オプションの使用 (次で説明)
- pkginfo ファイルの SUNW_PKG_ALLZONES 変数の設定 (pkginfo(4) を参照)
- 呼び出される patchadd のゾーンタイプ (大域または局所 (非大域))

上記の要因の相互関係を、次の「ゾーンの -G と pkginfo 変数の相互関係」に示します。

ゾーンがインストールされている Solaris システムでパッケージにパッチを追加すると、多数のゾーン関連のメッセージ、patchadd を大域ゾーンまたは局所ゾーンのどちらで呼び出すかによって異なる頻度と内容、SUNW_PKG_ALLZONES の設定、および -G オプションの使用が表示されます。

「形式」に示されている *patch*、`-M`、`-C`、`-R`、および `-S` 引数は、「オプション」のあとの「オペランド」で説明されています。

オプション　次のオプションを指定できます。

`-B` *backout_dir*
パッチのバックアウト (削除) 時に利用されるデータ (バックアウトデータ) を、パッケージデータベース以外のディレクトリに保存します。*backout_dir* は絶対パス名で指定してください。

`-d`
パッチが適用されるファイルのバックアップを作成しません。このオプションを指定すると、適用されたパッチを後で削除 (バックアウト) することはできません。

`-G`
現在のゾーンでのみパッケージにパッチを追加します。大域ゾーンで使用される場合、大域ゾーンでのみパッチはパッケージに追加され、既存の非大域ゾーン、または将来作成される非大域ゾーンには転送されません。非大域ゾーンで使用される場合、非大域ゾーンでのみパッチはパッケージに追加されます。次の「ゾーンの `-G` と `pkginfo` 変数の相互関係」を参照してください。

`-k` *keystore*
各パッチ内に見つかったデジタル署名を検証するのに、認証局の信頼された証明書を入手するための場所を *keystore* で指定します。キーストアが指定されていない場合、デフォルトのキーストアの場所で信頼された有効な証明書を探します。詳細は、pkgadd(1M) の「キーストアの場所」を参照してください。

`-n`
署名を無視してその検証を行いません。これは、パッチの内容が既知で信頼されている場合にのみ使用するべきです。本来、Solaris 8 のように、パッチ署名を検証する機能がないシステムにパッチを適用するためのオプションです。

`-p`
2 つめの形式で使用され、現在適用されているパッチのリストを表示します。

`-P` *passwd*
必要に応じて、`-k` で指定したキーストアを復号化するのに使用するパスワードを指定します。このオプションの引数の書式について詳細は、pkgadd(1M) の「パスフレーズの引数」を参照してください。

`-t`
Solaris 10 より以前のリリースにおいて返される `patchadd` 戻りコードを使用できるようにします。`zones`(5) がインストールされているシステム上では、戻り値 0 (ゼロ) は正常終了を示します。他の戻り値はエラーを示します。

`-u`
ほかの必須パッチまたは非互換パッチに対する検証をオフに設定します。このオプションを使用するときには細心の注意を払ってください。これを使用すると、予期しない不正な結果を引き起こす可能性があります。

`-x` *proxy*
パッケージをダウンロードする場合に使用する HTTP[S] プロキシを指定します。プロキシの書式は *host*:*port* です。ここで、*host* は HTTP[S] プロキシのホスト名、*port* はそのプロキシに関連付けられたポート番号です。このスイッチは、プロキシを指定するほかのすべての方法より優先します。デフォルトのプロキシを指定する代わりの方法についての詳細は、pkgadd(1M) の「環境」を参照してください。

オペランド

次のオペランドを指定できます。

ソース

`patchadd` は、パッチを抽出するために、ソースを指定する必要があります。次に示す構文を使ってソースを指定します。

patch
patch_id の絶対パス名または署名付きパッチを指す URI。
`/var/sadm/spool/patch/104945-02` は *patch* の一例です。
`https://syrinx.eng:8887/patches/104945-02` は、署名付きパッチを指す URI の一例です。

`-M` *patch_location* [*patch_list*]
インストールするパッチを、ディレクトリの場所または URL、およびオプションでパッチリストを含むファイル名で指定します。

patch_location としてディレクトリを使用する場合、そのディレクトリは絶対パス名で指定してください。URL の場合、スプールされたパッチを含むサーバー名およびパス名を指定します。省略可能な *patch_list* は、指定された場所にあるパッチのうち、インストール対象パッチを含んだファイルの名前です。

`-M` *patch_location patch_id* [*patch_id...*]
インストールするパッチをディレクトリの場所または URL、およびパッチ番号で指定します。

ディレクトリ場所または URL とパッチ番号を使用するには、スプールされたパッチが格納されているディレクトリの絶対パス名として、*patch_location* を指定します。URL の場合、スプールされたパッチを含むサーバー名およびパス名を指定します。該当するパッチのパッチ番号は *patch_id* で指定します。
`104945-02` は *patch_id* の一例です。また、`104945-02` は、`104945-02.jar` 内のパッチ ID の一例でもあります。

`patchadd` はパッチのリストを必要としません。ディレクトリ内に存在しているか、リスト内で指定されたか、あるいはコマンド行から入力された一連のパッチに対し、`patchadd` は有効性チェックと依存性チェックを実行します。具体的には、このコマンドは次の処理を行います。

- 各パッチがシステムに適用可能かどうかを判断します。たとえば、パッチ対象のパッケージがインストールされていない場合、patchadd はそのパッチの追加を試みません。
- 有効なパッチ間の依存関係を確認し、その関係に基づいてパッチのインストール順序を決定します。

大部分のユーザーにとって、patchadd のソースを指定するもっとも簡単な方法は、一連のパッチが格納された *patch_location* だけを指定することです。

宛先

デフォルトでは、patchadd は指定した宛先にパッチを適用します。宛先が指定されていない場合、現在のシステム (そのルートファイルシステムが / にマウントされているシステム) がパッチの宛先と仮定されます。次の方法で宛先を指定することもできます。

-C *net_install_image*
: setup_install_server で作成されたネットインストールイメージ上のミニルート上に置かれたファイルにパッチを適用します。*net_install_image* には Solaris 8 またはそれと互換性のあるバージョンの起動ディレクトリへの絶対パス名を指定します。「使用例」を参照してください。

  ミニルートへのインストールに推奨されているパッチをインストールする場合のみ、-C オプションを使用してください。ミニルートへのインストールが推奨されているパッチには、通常、パッケージコマンド、Sun 製インストールツールおよびパッチインストールツールのようなインストール関連のパッチが含まれています。ミニルートにパッチをたくさん適用しすぎると、ミニルートが大きくなり、Solaris のネットインストール時にメモリが足りなくなる可能性があります。ミニルートが大きくなりすぎないように、-B オプションと -C オプションを一緒に使用してください。上記の -B オプションの説明を参照してください。

  現在のリリース、および GRUB スタイルのブート (grub(5) を参照) をサポートする Solaris 10 のバージョンでは、ミニルートが圧縮されています。圧縮されたミニルートにパッチを適用するには、事前にある一定の手順を実行しておく必要があります。後述の「圧縮されたミニルートへのパッチ適用」を参照してください。

-R *client_root_path*
: patchadd で生成されたすべてのパッチファイルを *client_root_path* の下のディレクトリに配置します。*client_root_path* は、サーバーから見たクライアントのブート可能なルートを含むディレクトリです。*client_root_path* にはディレクトリツリーの先頭の絶対パスを指定します。この下に patchadd で生成されたすべてのパッチファイルがあります。-R オプションは -S オプションと一緒に指定することはできません。「注意事項」 を参照してください。

  注 – いかなる非大域ゾーンのルートファイルシステムも -R で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。zones(5) のマニュアルページを参照してください。

-S *service*
代わりとなるサービスを指定します (たとえば、Solaris_8)。このサービスはサーバーモデルおよびクライアントモデルの一部で、サーバーのコンソールからのみ使用可能です。サーバーは、smosservice(1M) で作成された /usr 共有ファイルシステムを持つことができ、これらのサービス領域は、それらが扱うクライアントに使用可能にすることができます。この -S オプションは -R オプションと同時に指定することはできません。「注意事項」 を参照してください。

圧縮されたミニルートへのパッチ適用

Solaris オペレーティングシステムの現在のリリースでは、圧縮されたミニルートが使用されます。x86 システムではすべてのミニルートが、SPARC システムでは一部のミニルートが、それぞれ圧縮されています。GRUB スタイルのブートをサポートする x86 版 Solaris 10 バージョンでも、圧縮されたミニルートが使用されます。使用する Solaris 10 システムが GRUB スタイルのブートをサポートするかどうかを確認するための簡単な方法については後述します。

圧縮されたミニルート (すべてまたは一部) を持つシステムにパッチを適用するには、-C 適用先指定子を使って patchadd を実行する際に、その前後でミニルートの展開と再圧縮を行う必要があります。次に示す手順とコマンド例を参考にしてください。

1. 圧縮されたミニルートを展開します。

```
# /boot/solaris/bin/root_archive unpackmedia \
/export/home/altuser/testdir /export/home/altuser/mr
```

2. -C を指定して patchadd を実行します。ミニルートにパッチが適用されます。

```
# patchadd -C /export/home/altuser/mr \
/var/sadm/spool/104945-02
```

3. ミニルートを再圧縮します。

```
# /boot/solaris/bin/root_archive packmedia \
/export/home/altuser/testdir /export/home/altuser/mr
```

この時点で、setup_install_server(1M) を使ってパッチ適用済みのミニルートを特定のインストールサーバー上にインストールできる状態になっています。このコマンドについては、root_archive(1M) を参照してください。

ある x86 版 Solaris 10 システムが GRUB スタイルのブートをサポートするかどうかを確認するには、そのインストール媒体上に /cdrom/boot ディレクトリが存在するか調べます。このディレクトリが存在していれば、GRUB がサポートされています。ネットワークインストールイメージでネットワークインストールを実行する場合は、Solaris_*relnum* ディレクトリと同じレベルに boot という名前のディレクトリが存在しているか確認します。 たとえば、ディレクトリ /export/Solaris_10 が存在する場合であれば、/export/boot ディレクトリの有無を確認します。ここでも、そのようなディレクトリが存在していれば、GRUB がサポートされています。

ゾーンの -G と pkginfo 変数の相互関係

次のリストに、大域ゾーンおよび局所 (非大域) ゾーンでパッチを追加する場合の -G オプションおよび SUNW_PKG_ALLZONES 変数 (pkginfo(4) を参照) の相互関係を示します。

大域ゾーン、-G を指定

SUNW_PKG_ALLZONES を真に設定しているパッケージがある場合: エラーです。変更はありません。

SUNW_PKG_ALLZONES を true に設定しているパッケージがない場合: 大域ゾーンでのみパッケージにパッチを適用します。

大域ゾーン、-G を指定しない

SUNW_PKG_ALLZONES を真に設定しているパッケージがある場合: すべてのゾーンで適切なパッケージにパッチを適用します。

SUNW_PKG_ALLZONES を true に設定しているパッケージがない場合: すべてのゾーンで適切なパッケージにパッチを適用します。

局所ゾーン、-G を指定または指定しない

SUNW_PKG_ALLZONES を真に設定しているパッケージがある場合: エラーです。変更はありません。

SUNW_PKG_ALLZONES を true に設定しているパッケージがない場合: 局所ゾーンでのみパッケージにパッチを適用します。

キーストアの場所

詳細については、pkgadd(1M) のマニュアルページのセクション「キーストアの場所」を参照してください。

キーストアおよび証明書の形式

詳細については、pkgadd(1M) のマニュアルページのセクション「キーストアおよび証明書の形式」を参照してください。

使用例

次に示す例では、/usr/sbin ディレクトリのコマンドを使用しているものとします。

例 1 スタンドアロンマシンに 1 つのパッチをインストールする

スタンドアロンマシンに 1 つのパッチをインストールする例を示します。

```
example# patchadd /var/spool/patch/104945-02
```

例 2 サーバーのコンソールからクライアントに 1 つのパッチをインストールする

サーバーのコンソールからクライアントに 1 つのパッチをインストールする例を示します。

```
example# patchadd -R /export/root/client1 /var/spool/patch/104945-02
```

例 3　サーバーのコンソールからサービスに 1 つのパッチをインストールする

サーバーのコンソールからサービスに 1 つのパッチをインストールする例を示します。

```
example# patchadd -S Solaris_8 /var/spool/patch/104945-02
```

例 4　patchadd を 1 回実行して複数のパッチをインストールする

1 回の patchadd の実行で複数のパッチをインストールする例を示します。

```
example# patchadd -M /var/spool/patch 104945-02 104946-02 102345-02
```

例 5　パッチのリストが記述されているファイルを指定して複数のパッチをインストールする

インストールするパッチのリストが記述されたファイルを指定して、複数のパッチをインストールする例を示します。

```
example# patchadd -M /var/spool/patch patchlist
```

例 6　クライアントに複数のパッチをインストールし、バックアウトデータをデフォルト以外のディレクトリに保存する

クライアントに複数のパッチをインストールし、パッチのバックアウト時に利用されるデータ (バックアウトデータ) をデフォルト以外のディレクトリに保存する例を示します。

```
example# patchadd -M /var/spool/patch -R /export/root/client1
     -B /export/backoutrepository 104945-02 104946-02 102345-02
```

例 7　Solaris 8 およびその互換バージョンのネットインストールイメージにパッチをインストールする

Solaris 8 およびその互換バージョンのネットインストールイメージにパッチをインストールする例を示します。

```
example# patchadd -C /export/Solaris_8/Tools/Boot \
    /var/sadm/spool/104945-02
```

例 8　圧縮されたミニルートにパッチをインストールする

次の例では、GRUB スタイルのブートをサポートする Solaris x86 マシンに含まれる圧縮されたミニルートに、パッチをインストールします。この例では、/export/Solaris_11/Tools/Boot に展開済みのミニルートが含まれていると仮定しています。このミニルートは、パッチ適用後に再圧縮する必要があります。

```
example# patchadd -C /export/Solaris_11/Tools/Boot \
/var/sadm/spool/104945-02
```

例 8 圧縮されたミニルートにパッチをインストールする (続き)

圧縮されたミニルートを使用する Solaris バージョンについては、前述の「圧縮されたミニルートへのパッチ適用」を参照してください。

例 9 圧縮されていないミニルートにパッチをインストールする

次の例では、圧縮されたミニルートを持たない Solaris マシン上のミニルートに、パッチをインストールします。

```
example# patchadd -C /export/Solaris_9/Tools/Boot \
/var/sadm/spool/104945-02
```

圧縮されたミニルートを使用する Solaris バージョンについては、前述の「圧縮されたミニルートへのパッチ適用」を参照してください。

例 10 クライアントにインストールされているパッチを表示する

クライアント上にインストールされているパッチを表示する例を示します。

```
example# patchadd -R /export/root/client1 -p
```

-R の使い方については、前述されているこのオプションへの注意事項に留意してください。

例 11 パッチのデジタル署名セットをインストールする

次の例では複数のパッチをインストールします。その一部は、提供されたキーストア、パスワード、および HTTP プロクシを使用して署名されています。

```
example# patchadd -k /etc/mycerts -p pass:abcd -x webcache.eng:8080 \
-M http://www.sun.com/solaris/patches/latest 101223-02 102323-02
```

終了ステータス

次の終了ステータスが返されます。

0 正常終了。

>0 エラーが発生した。

属性

次の属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWswmt, SUNWcsu

属性タイプ	属性値
インタフェースの安定性	開発中

関連項目

cpio(1), pkginfo(1), patchrm(1M), pkgadd(1M), pkgadm(1M), pkgchk(1M), pkgrm(1M), setup_install_server(1M), smpatch(1M), showrev(1M), pkginfo(4), attributes(5), grub(5), zones(5)

診断

パッチのインストール時によく発生する問題、出力されるエラーメッセージ、その対処方法について説明します。

パッチのインストールエラー

メッセージ

```
The prepatch script exited with return code retcode.
patchadd is terminating.
```

説明・対処法

パッチに付属している prepatch スクリプトが 0 以外の終了コードで終了しました。prepatch スクリプトのトレースを実行して、prepatch スクリプトが不正な終了コードで終了した原因を調べてください。問題を修正するには、prebackout スクリプトの先頭行に -x オプションを追加して再度 patchrm を実行してください。

メッセージ

```
The signature on patch patch_id was unable to be verified.
patchadd is terminating.
```

説明・対処法

使用しているキーストアおよびパッチの署名が与えられたのですが、パッチのデジタル署名が検証できませんでした。パッケージの署名を検証するのに必要なトラストアンカーがあるかどうか、またパッケージが勝手に変更されていないかどうか、キーストアを確認してください。

メッセージ

```
The postpatch script exited with return code retcode.
Backing out patch.
```

説明・対処法

パッチに付属している postpatch スクリプトが 0 以外の終了コードで終了しました。このスクリプトはおもに、パッチパッケージのオブジェクトに対応しないファイルを一掃するため (つまりパッケージの所有権およびアクセス権に問題があるとき) に使用されます。出力された検証エラーをすべて確認し、それぞれについて適切な処置を行なったあと、-u オプション付きで再度 patchadd を実行してください。検証エラーを無視してパッチがインストールされます。

メッセージ

```
Insufficient space in /var/sadm/patch to save old files.
(For 2.4 systems and previous)
```

説明・対処法

パッチ適用前のファイルを保存するための容量が /var/sadm/patch ディレクトリにありません。3 つの対処法があります。(1) patchadd を実行する際に、-B オプションを使用します。このオプションを patchadd に指定すると、ユーザーが指定したファイルシステムに、バックアウトデータ (パッチのバックアウト時に利用されるデータ) を保存します。(2) 不要なファイルを削除してディスク容量を確保します。(3) -d オプション付きで patchadd を実行して、パッチ適用前のファイルを保存しないようにします。

パッチ適用前のファイルを保存しないように選択した場合、後に patchrm を使用してパッチを削除することはできなくなります。以前に適用したパッチの保存領域を削除することによって、システム領域を確保するという方法もあります。後にパッチを削除する可能性がないと判断した場合は、patchadd によって保存されたファイルを削除しても構いません。パッチ *patch_id* について保存されたファイルを削除するには、次のように実行してください。

```
cd /var/sadm/patch/patch_id
rm -r save/*
rm .oldfilessaved
```

これらのコマンドを実行したあとは、パッチ *patch_id* をバックアウトすることはできません。

メッセージ

```
Insufficient space in /var/sadm/pkg/PKG/save to save old files.
(For 2.5 systems and later)
```

説明・対処法

/var/sadm/pkg/*PKG*/save ディレクトリに容量が不足しているため、パッチ適用前のファイルを保存できません。3 つの対処法があります。(1) patchadd を実行する際に、-B オプションを使用します。このオプションを patchadd に指定すると、ユーザーが指定したファイルシステムに、バックアウトデータ (パッチのバックアウト時に利用されるデータ) を保存します (1 つ前のメッセージに関する説明を参照してください)。(2) 不要なファイルを削除してディスク領域を確保します。(3) -d オプション付きで patchadd を実行して、パッチ適用前のファイルを保存しないようにします。パッチ適用前のファイルを保存しないように選択した場合、後に patchrm を使用してパッチを削除することはできなくなります。以前に適用したパッチの保存領域を削除することによって、システム領域を確保するという方法もあります。後にパッチを削除する可能性がないと判断した場合は、patchadd によって保存されたファイルを削除しても構いません。*patch_id* について保存されたファイルを削除するには、次のように実行してください。

```
cd /var/sadm/pkg/pkgabbrev/save
rm -r patch_id
```

これらのコマンドを実行したあとは、パッチ *patch_id* をバックアウトすることはできません。

メッセージ

```
Save of old files failed.
(For 2.4 systems and previous)
```

説明・対処法

パッチを適用する前に、パッチインストールスクリプトは cpio を使用してパッチ適用前のファイルを保存します。このエラーメッセージは、cpio が失敗したことを示します。このエラーメッセージの前に cpio からの出力が表示されているはずです。cpio の失敗を修正するために適切な処置を行う必要があります。通常、失敗の原因は、パッチ適用前のファイルを保存するのに十分なディスク領域がないことです。ディスク領域の不足に対処するには、次の 2 つの方法があります。(1) 不要なファイルを削除してディスク領域を確保します。(2) -d オプション付きで patchadd を実行してパッチ適用前のファイルを保存しないようにします。ただし、パッチ適用前のファイルを保存しないように選択した場合、パッチを削除することはできなくなります。

メッセージ

```
Pkgadd of pkgname package failed with error code code.
See /tmp/log.patch_id for reason for failure.
```

説明・対処法

メッセージ中に示されたパッチパッケージのインストールに失敗しました。patchadd はパッチをバックアウトして、システムをパッチ適用前の状態にします。ログファイルを参照してインストールに失敗した原因を確認し、必要な処置を行なって問題を解決したあと、再度パッチを適用してください。

メッセージ

```
Pkgadd of pkgname package failed with error code code.
Will not backout patch...patch re-installation.
Warning: The system may be in an unstable state!
See /tmp/log.patch_id for reason for failure.
```

説明・対処法

メッセージ中に示されたパッチパッケージのインストールに失敗しました。patchadd はパッチをバックアウトしません。patchrm を使用して手動でパッチをバックアウトし、再度パッチ全体を適用することをお勧めします。ログファイルを参照して pkgadd が失敗した原因を確認し、必要な処置を行なって問題を解決したあと、再度パッチを適用してください。

メッセージ

```
patchadd is unable to find the INST_RELEASE file. This file
must be present for patchadd to function correctly.
```

説明・対処法
システムに INST_RELEASE ファイルがありません。このファイルは初期インストール中またはアップグレード中に作成されます。

メッセージ

```
A previous installation of patch patch_id was invoked
that saved files that were to be patched. Since files
were saved, you must run this instance of patchadd
without the -d option.
```

説明・対処法
以前に -d オプションを使用しないでパッチがインストールされている場合、パッチを再インストールするときにも -d オプションなしで実行する必要があります。-d オプションを付けずに patchadd を実行してください。

メッセージ

```
A previous installation of patch patch_id was invoked
with the -d option. (i.e. Do not save files that would
be patched) Therefore, this invocation of patchadd
must also be run with the -d option.
```

説明・対処法
以前に -d オプションを使用してパッチがインストールされている場合、パッチを再インストールするときにも -d オプションを使用する必要があります。-d オプション付きで patchadd を実行してください。

その他の診断メッセージ

次に示すパッチインストール時のメッセージは、「説明・対処方法」で説明しているように必ずしもエラーではありません。ただし、これらのメッセージはパッチインストールのログファイルに記録されます。

メッセージ

```
Package not patched:
PKG=SUNxxxx
Original package not installed
```

説明・対処法
メッセージに示されているパッチコンポーネントは、システムにインストールされていないパッケージに対するパッチです。これは必ずしもエラーではありません。1 つのパッチが 1 つのバグを複数のパッケージに対して修正することもあります。

たとえば、オンラインバックアップと fddi パッケージの両方に対する 1 つのバグを修正するパッチを例として考えます。オンラインバックアップはイン

ストールされているけれども fddi パッケージがインストールされていない場合、次のようなメッセージが出力されます。

```
Package not patched:
PKG=SUNWbf
Original package not installed
```

システム上に fddi パッケージがインストールされている場合には、このメッセージをエラーとして対処する必要があります。必要な処置を行なってパッケージをインストールし、(パッチによってほかのパッケージがインストールされている場合は) パッチをバックアウトし、再度パッチをインストールしてください。

メッセージ

```
Package not patched:
PKG=SUNxxx
ARCH=xxxxxxx
VERSION=xxxxxxx
Architecture mismatch
```

説明・対処法

メッセージに示されているパッチコンポーネントは、ユーザーが使用しているシステムとは異なるアーキテクチャーのパッケージに対するパッチです。これは必ずしもエラーではありません。アーキテクチャー固有のパッケージに対するパッチには、該当する各アーキテクチャーごとに 1 つのコンポーネントが含まれている場合もあります。たとえば、sun4u アーキテクチャーのシステムを使用しているとします。このときに SUNWcar パッケージに対するパッチをインストールしようとすると、次のようなメッセージが出力されます。

```
Package not patched:
PKG=SUNWcar
ARCH=sparc.sun4c
VERSION=11.5.0,REV=2.0.18
Architecture mismatch

Package not patched:
PKG=SUNWcar
ARCH=sparc.sun4u
VERSION=11.5.0,REV=2.0.18
Architecture mismatch

Package not patched:
PKG=SUNWcar
ARCH=sparc.sun4e
VERSION=11.5.0,REV=2.0.18

Package not patched:
```

```
PKG=SUNWcar
ARCH=sparc.sun4
VERSION=11.5.0,REV=2.0.18
Architecture mismatch
```

これらのメッセージは、patchadd がアーキテクチャーを正しく認識しない場合にのみエラー状態を示しています。

メッセージ

```
Package not patched:
PKG=SUNxxxx
ARCH=xxxx
VERSION=xxxxxxx
Version mismatch
```

説明・対処法

パッチが適用されるソフトウェアバージョンがシステムにインストールされていません。たとえば Solaris 8 を実行している場合に Solaris 9 に対するパッチをインストールしようとすると、次のようなメッセージが出力されます。

```
Package not patched:
PKG=SUNWcsu
ARCH=sparc
VERSION=10.0.2
Version mismatch
```

このメッセージは、必ずしもエラーを示しているわけではありません。パッチを適用するパッケージのバージョンが不一致の場合は、正しいバージョンのパッチを入手するか、または正しいバージョンのパッケージをインストールしてください。そのあと、必要な場合はパッチをバックアウトしてから、再度パッチを適用してください。

メッセージ

```
Re-installing Patch.
```

説明・対処法

インストールしようとしているパッチはすでに適用されていますが、追加インストールされるパッケージが少なくとも 1 つパッチに含まれています。たとえば、AnswerBook がインストールされていないシステムに、 OpenWindows と AnswerBook の両方のコンポーネントが含まれているパッチを適用すると、そのパッチの AnswerBook コンポーネント部分は適用されません。後に pkgadd を使って AnswerBook をインストールしてパッチを再度適用すると、そのパッチの AnswerBook コンポーネント部分がシステムに適用されます。

メッセージ

```
patchadd Interrupted.
patchadd is terminating.
```

説明・対処法
patchadd の実行が中断されました (通常 CTRL-c が押されたため)。patchadd は処理中のファイルを削除し、終了します。

メッセージ

```
patchadd Interrupted.
Backing out Patch...
```

説明・対処法
patchadd の実行が中断されました (通常 CTRL-c が押されたため)。patchadd は処理中のファイルを削除し、パッチをバックアウトし、終了します。

注意事項

クライアントまたはサーバーにパッチをインストールするには、patchadd を 2 回実行する必要があります。 1 回は -R オプションを付けて実行し、もう 1 回は -S オプションを付けて実行します。これによって、パッチが /usr と / (ルート) パーティションの両方に確実にインストールされます。パッチに /usr と / (ルート) のパッケージが含まれている場合に、上記の方法を実行する必要があります。

patchadd を実行すると、pkgadd が起動され、*pkg*/install ディレクトリにあるインストールスクリプトを実行します。checkinstall スクリプトは、所有権を install というユーザー名に設定して実行されます。ユーザー名が install に設定されていない場合、pkgadd は checkinstall スクリプトを noaccess として実行します。SVR4 ABI には、checkinstall は情報収集を行うスクリプトとしてのみ使用されると記述されています。checkinstall スクリプトのアクセス権を初期設定から変更すると、pkgadd はファイルをオープンして読み取ることができなくなり、次のようなメッセージが出力されてパッチのインストールが異常終了します。

```
pkgadd: ERROR: checkinstall script did not complete successfully.
```

このため、checkinstall スクリプトのアクセス権は変更しないでください。パッチが正しくインストールされたときのパッチインストールログファイルの内容は、patchadd が pkgadd の出力をリダイレクトしたものになります。パッチが正しくインストールされると、pkgadd は次のようなメッセージを出力し、それがログファイルに書き込まれます。

```
This appears to be an attempt to install the same architecture
and version of a package which is already installed. This
installation will attempt to overwrite this package.
This message does not indicate a failure, it represents the
correct behavior by pkgadd when a patch installs correctly.
```

このメッセージは失敗を示しているものではなく、パッチが正しくインストールされたときの pkgadd による正しい動作を示しています。

クライアント・サーバーマシンでは、既存のクライアントにまたはクライアントのルートディレクトリ (templates 領域) にパッチパッケージは適用 (インストール) されません。このため、すべてのクライアントマシンにおいて直接 patchadd を使用して、適切な時にクライアントマシンにパッチを適用する必要があります。クライ

アントへのパッチの適用については、前述の説明を参照してください。パッケージユーティリティー (pkgadd, pkgrm, pkgchk) に影響するバグは、patchadd または patchrm の動作に影響する場合があります。patchadd および patchrm は、上記のパッケージユーティリティーを使用してパッチパッケージをインストールおよびバックアウトしています。これらのパッケージユーティリティーのバグを修正するパッチが提供されているかどうかを確認し、提供されている場合はそのパッチを先に適用してから、ほかのパッチを適用することをお勧めします。現在提供されている、パッケージユーティリティーに対するパッチは、次のとおりです。

Solaris 2.5.1 (Sparc 版):
104578

Solaris 2.5.1 (Intel 版):
104579

Solaris 2.6 (Sparc 版):
106292

Solaris 2.6 (Intel 版):
106293

警告

パッチの中には「遅延実行 (deferred activation)」パッチに分類されるものが あります(Deferred Activation のように、頭文字が大文字で記述されることもあります)。次に示す条件下では、このようなパッチには特別な処理が必要になります。そのパッチが遅延実行パッチに該当するかどうかは、パッチの README に記載されて います。 README ファイル内で Deferred Activation という文字列を検索して ください。

遅延実行処理を必要とするパッチをインストールしたり削除したりする場合は、次の ことを確認してください。

- ゾーンを実行しているシステムでは、パッチの追加や削除の際には、 すべての非大域ゾーンが停止状態になっていること。
- 遅延実行処理を安全に完了するには、ループバックファイルシステム (loopback file system、lofs) が必要です。 Sun Cluster 3.1 または Sun Cluster 3.2 を実行しているシステムでは、 多くの場合、lofsがオフにされています。これは、lofs が実行されていると、 HA-NFS 機能に制限が生じるためです。そのため、遅延実行を必要とするパッチを インストールまたは削除する前に、/etc/system ファイル内の次の行をコメントに して、ループバックファイルシステムを実行しておく必要があります。

  ```
  exclude:lofs
  ```

  次に、システムをリブートしてパッチをインストールまたは削除します。
  パッチの処理が終わったら、上記の行のコメントを外し、リブートして通常の処理を 続けてください。

名前　patchrm – システムからのパッチ削除とパッチ適用前のファイル復元

形式
```
patchrm [-f] [-G] [-B backout_dir]
     [-C net_install_image | -R client_root_path | -S service]
     [-t] patch_id
```

機能説明　patchrm は、Solaris 2.x オペレーティング環境、および Solaris 2.x と互換性がある、2.x 以降の Solaris オペレーティング環境 (Solaris 8 など) を稼動しているシステムから、パッチパッケージを削除し、以前に保存しておいた (パッチ適用前の) ファイルを復元します。patchrm は Solaris 1.x システム用のパッチには使用できません。patchrm を実行するにはスーパーユーザーになる必要があります。

zones(5) に関しては、大域ゾーンで呼び出されると、デフォルトでは patchrm はすべてのゾーンにおいてすべての適切なパッケージにパッチを適用します。ゾーン環境におけるパッチの削除動作は次の要因により異なります。

- -G オプションの使用 (次で説明)
- pkginfo ファイルの SUNW_PKG_ALLZONES 変数の設定 (pkginfo(4) を参照)
- 呼び出される patchrm のゾーンタイプ (大域またはローカル (非大域))

上記の要因の相互関係を、次の「ゾーンの -G と pkginfo 変数の相互関係」に示します。

ゾーンがインストールされている Solaris システムでパッケージにパッチを追加すると、多数のゾーン関連のメッセージ、patchrm を大域ゾーンまたは局所ゾーンのどちらで呼び出すかによって異なる頻度と内容、SUNW_PKG_ALLZONES の設定、および -G オプションの使用が表示されます。

オプション　次のオプションを指定できます。

-B *backout_dir*　パッケージデータベースとは別のディレクトリにバックアウトデータ (パッチのバックアウト時に利用されるデータ) が保存されているパッチを、バックアウトします。パッチのインストール時に patchadd コマンドでデフォルトのバックアウトデータのディレクトリを変更した場合のみ、このオプションを使用します。*backout_dir* は絶対パス名で指定してください。

-C *net_install_image*　setup_install_server によって作成された、ネットインストールイメージ上の miniroot にあるファイルに対して適用されたパッチを削除 (バックアウト) します。*net_install_image* には、Solaris 2.6 およびその互換バージョンのブートディレクトリへの絶対パス名を指定してください。「使用例」を参照してください。

-f　他のパッチに置き換えられているかどうかに関係なく、指定したパッチを削除 (バックアウト) します。

-G	現在のゾーンでのみパッケージからパッチを削除します。大域ゾーンで使用される場合、大域ゾーンでのみパッチはパッケージから削除され、既存の非大域ゾーン、または将来作成される非大域ゾーンのパッケージからは削除されません。非大域ゾーンで使用される場合、非大域ゾーンでのみパッチはパッケージから削除されます。次の「ゾーンの -G と pkginfo 変数の相互関係」を参照してください。
-R *client_root_path*	patchrm によって生成されるすべてのパッチファイルをディレクトリ *client_root_path* の下に置きます。*client_root_path* は、サーバーから見たクライアントのブート可能なルートを含むディレクトリです。*client_root_path* には、patchrm で生成されたすべてのパッチファイルが置かれるディレクトリツリーの、先頭の絶対パス名を指定します。-R オプションは -S オプションと一緒に指定することはできません。 注 – いかなる非大域ゾーンのルートファイルシステムも -R で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。zones(5) のマニュアルページを参照してください。
-S *service*	代替サービス (たとえば Solaris_2.3) を指定します。このサービスはサーバーモデルおよびクライアントモデルの一部で、サーバーのコンソールからのみ使用可能です。サーバーは、smosservice(1M) で作成された /usr 共有ファイルシステムを持つことができ、これらのサービス領域は、それらが扱うクライアントに使用可能にすることができます。この -S オプションは -R オプションと同時に指定することはできません。
-t	Solaris 10 より以前のリリースにおいて返される patchrm 戻りコードを使用できるようにします。zones(5) がインストールされているシステム上では、戻り値 0 (ゼロ) は正常終了を示します。他の戻り値はエラーを示します。

ゾーンの -G と pkginfo 変数の相互関係

次のリストに、大域ゾーンおよびローカル (非大域の) ゾーンでパッチを削除する場合の -G オプションおよび SUNW_PKG_ALLZONES 変数 (pkginfo(4) を参照) の相互関係を示します。

大域ゾーン、-G を指定	SUNW_PKG_ALLZONES を真に設定しているパッケージがある場合: エラーです。変更はありません。

	SUNW_PKG_ALLZONES を true に設定しているパッケージがない場合: 大域ゾーンでのみパッケージからパッチを削除します。
大域ゾーン、-G を指定しない	SUNW_PKG_ALLZONES を真に設定しているパッケージがある場合: すべてのゾーンで適切なパッケージからパッチを削除します。
	SUNW_PKG_ALLZONES を true に設定しているパッケージがない場合: すべてのゾーンで適切なパッケージからパッチを削除します。
局所ゾーン、-G を指定または指定しない	SUNW_PKG_ALLZONES を真に設定しているパッケージがある場合: エラーです。変更はありません。
	SUNW_PKG_ALLZONES を true に設定しているパッケージがない場合: ローカルゾーンでのみパッケージからパッチを削除します。

オペランド　次のオペランドを指定できます。

patch_id　パッチ番号。104945-02 は *patch_id* の一例です。

使用例　以下に示す例では、パッチ 104945-02 がシステムにインストールされていることを前提としています。また、/usr/sbin ディレクトリのコマンドを使用しているものとします。

例 1　スタンドアロンシステム上の 1 つのパッチを削除する

スタンドアロンシステム上のパッチを 1 つ削除 (バックアウト) する例を示します。

```
example# patchrm 104945-02
```

例 2　サーバーのコンソールからクライアントシステム上の 1 つのパッチを削除する

サーバーのコンソールから、クライアントシステム上のパッチを 1 つ削除 (バックアウト) する例を示します。

```
example# patchrm -R /export/root/client1 104945-02
```

-R の使い方については、前述されているこのオプションへの注意事項に留意してください。

例 3　サーバーの OS サービス領域上の 1 つのパッチを削除する

サーバーの OS サービス領域にあるパッチを 1 つ削除 (バックアウト) する例を示します。

```
example# patchrm -S Solaris_2.3 104945-02
```

例 4　ネットインストールイメージ上の 1 つのパッチを削除する

ネットインストールイメージ上のパッチを 1 つ削除 (バックアウト) する例を示します。

```
example# patchrm -C /export/Solaris_2.6/Tools/Boot 104945-02
```

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生した。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWswmt, SUNWcsu

診断

パッチのバックアウト時によく発生する問題、出力されるエラーメッセージ、その対処方法について説明します。

メッセージ

```
prebackout patch exited with return code code.
patchrm exiting.
```

説明・対処法　パッチに付属している prebackout スクリプトが **0** 以外の終了コードで終了しました。prebackout スクリプトのトレースを作成して、prebackout スクリプトが失敗した原因を調べてください。問題を修正するには、prebackout スクリプトの先頭行に **-x** オプションを追加して再度 patchrm を実行してください。

メッセージ

```
postbackout patch exited with return code code.
patchrm exiting.
```

説明・対処法　パッチに付属している postbackout スクリプトが **0** 以外の終了コードで終了しました。postbackout ス

クリプトを参照して postbackout スクリプトが失敗した原因を調べてください。問題を修正するには、postbackout スクリプトの先頭行に -x オプションを追加して、必要な場合は postbackout スクリプトだけを再度実行してください。

メッセージ

```
Only one service may be defined.
```

説明・対処法　複数の OS サービスからパッチをバックアウトしようとしています。複数の OS サービスからパッチをバックアウトするには、各 OS サービスごとに別々に patchrm を実行してください。

メッセージ

```
The -S and -R arguments are mutually exclusive.
```

説明・対処法　ネイティブでない OS サービスと *client_root_path* からパッチをバックアウトしようとしています。これら 2 つの引数は互いに排他的です。ネイティブでない usr パーティションからパッチをバックアウト (削除) するには -S オプションを使用する必要があります。クライアントの (ネイティブまたは非ネイティブの) ルートパーティションからパッチをバックアウトするには -R オプションを使用する必要があります。

メッセージ

```
The service service cannot be found on this system
```

説明・対処法　ネイティブでない OS サービスからパッチをバックアウト (削除) しようとしましたが、指定された OS サービスはシステムにインストールされていません。正しい OS サービスを指定してください。

メッセージ

```
Only one client_root_path may be defined.
```

説明・対処法　-R オプションを指定して複数の *client_root_path* を指定しています。1 回の patchrm の実行につき -R オプションは 1 回だけ使用できます。

メッセージ

```
The dir directory cannot be found on this system.
```

説明・対処法　-R オプションを使用して、マウントされていないまたはシステムに存在しないディレクトリを指定しています。正しいディレクトリ名を指定して、パッチのバックアウトを再度実行してください。

メッセージ

```
Patch patch_id has not been successfully installed to this system.
```

説明・対処法　システムにインストールされていないパッチをバックアウト (削除) しようとしています。パッチが適用されたファイルをパッチ適用前のバージョンに復元するには、最初のインストール時に使用した CD から元のファイルを復元してください。

メッセージ

```
Patch patch_id has not been successfully applied to this system.
Will remove directory dir.
```

説明・対処法　システムに適用されていないパッチをバックアウト (削除) しようとしています。パッチは適用されていませんが、(失敗した patchadd によって作成された) /var/sadm/patch/*patch_id* ディレクトリが残っています。パッチをバックアウトすることはできません。パッチが適用されたファイルをパッチ適用前のバージョンに復元するには、最初のインストール時に使用した CD から元のファイルを復元してください。

メッセージ

```
This patch was obsoleted by patch patch_id.
Patches must be backed out in the reverse order in
which they were installed. Patch backout aborted.
```

説明・対処法　不適切な順番でパッチをバックアウト (削除) しようとしています。パッチは正しい順序でバックアウトする必要があります。バックアウトしようとしているパッチ以外のパッチにも影響が及んでいる可能性があります。

メッセージ

```
Patch patch_id is required to be installed by an already
installed patch_id.
It cannot be backed out until the required patch is backed out first.
```

説明・対処法　インストール (適用) されている必要があるとメッセージ中に示されているパッチをバックアウト (削除) してから、目的のパッチをバックアウトしてください。

メッセージ

```
The installation of patch patch_id was interrupted.
```

説明・対処法　以前に行なったパッチのインストールが中断されています。目的のパッチをバックアウト (削除) する前に、インストールが中断されたパッチをインストールする必要があります。

メッセージ

```
Patch patch_id was installed without backing up the original
files. It cannot be backed out.
```

説明・対処法　パッチの適用時に patchadd コマンドの -d オプションを指定したか、またはディスク領域を確保するためにパッチの保存領域が削除されています。このため、元のファイルが保存されていないので patchrm を使用できません。元のファイルはインストール CD からのみ復元できます。

メッセージ

```
pkgadd of pkgname package failed return code code.
See /var/sadm/patch/patch_id/log for reason for failure.
```

説明・対処法　メッセージに示されているパッチパッケージのインストールに失敗しています。ログファイルを参照してインストールに失敗した原因を確認し、問題を修正後に、バックアウトスクリプトを再度実行してください。

メッセージ

```
Restore of old files failed.
```

説明・対処法　パッチが適用されたファイルをパッチ適用前のバージョンに復元するために、バックアウトスクリプトが cpio コマンドを使用しています。上記のメッセージの前に cpio コマンドからの出力が表示されているはずです。cpio が失敗した原因を調べて必要な処置を行なってください。これは、Solaris 2.4 およびそれ以前のバージョンのシステムで発生するエラーです。

関連項目 cpio(1), pkginfo(1), patchadd(1M), pkgadd(1M), pkgchk(1M), pkgrm(1M), showrev(1M), pkginfo(4), attributes(5), zones(5)

注意事項 クライアント・サーバーマシンでは、既存のクライアントからまたはクライアントのルートディレクトリ (templates 領域) からパッチパッケージは削除 (バックアウト) されません。このため、すべてのクライアントマシンにおいて直接 patchrm を使用して、クライアントマシンから適切な時にパッチを削除する必要があります。パッケージユーティリティ (pkgadd, pkgrm, pkgchk) に影響するバグは、patchadd または patchrm の動作に影響する場合があります。patchadd および patchrm は、上記のパッケージユーティリティを使用してパッチパッケージをインストールおよびバックアウトしています。これらのパッケージユーティリティのバグを修正するパッチが提供されているかどうかを確認し、提供されている場合はそのパッチを先に適用してから、他のパッチを適用することをお勧めします。現在提供されている、パッケージユーティリティーに対するパッチは、次のとおりです。

Solaris 2.1:	100901
Solaris 2.2:	101122
Solaris 2.3:	101331
Solaris 2.4 (SPARC 版):	102039
Solaris 2.4 (Intel 版):	102041
Solaris 2.5.1 (Sparc 版):	104578
Solaris 2.5.1 (Intel 版):	104579
Solaris 2.6 (Sparc 版):	106292
Solaris 2.6 (Intel 版):	106293

警告 パッチの中には「遅延実行 (deferred activation)」パッチに分類されるものが あります(Deferred Activation のように、頭文字が大文字で記述されることもあります)。次に示す条件下では、このようなパッチには特別な処理が必要になります。そのパッチが遅延実行パッチに該当するかどうかは、パッチの README に記載されて います。 README ファイル内で Deferred Activation という文字列を検索して ください。

遅延実行処理を必要とするパッチをインストールしたり削除したりする場合は、次の ことを確認してください。

- ゾーンを実行しているシステムでは、パッチの追加や削除の際には、 すべての非大域ゾーンが停止状態になっていること。
- 遅延実行処理を安全に完了するには、ループバックファイルシステム (loopback file system、lofs) が必要です。 Sun Cluster 3.1 または Sun Cluster 3.2 を実行しているシステムでは、 多くの場合、lofsがオフにされています。これは、lofs が実行されていると、 HA-NFS 機能に制限が生じるためです。そのため、遅延実

行を必要とするパッチを インストールまたは削除する前に、/etc/system ファイル内の次の行をコメントに して、ループバックファイルシステムを実行しておく必要があります。

```
exclude:lofs
```

次に、システムをリブートしてパッチをインストールまたは削除します。
パッチの処理が終わったら、上記の行のコメントを外し、リブートして通常の処理を 続けてください。

名前

pgxconfig, GFXconfig, TSIgfxp_config – PGX32 (Raptor GFX) グラフィックスアクセラレータの設定

形式

```
/usr/sbin/pgxconfig  [-dev device-filename] [-res video-mode
 [try |  noconfirm |  nocheck]] [-file machine |  system]
 [-depth 8 |  24] [-24only true | false] [-cachedpixmap true | false]
 [-defaults]

/usr/sbin/pgxconfig  [-propt] [-prconf]

/usr/sbin/pgxconfig  [-help] [-res ?]

/usr/sbin/pgxconfig  [-i]
```

機能説明

`pgxconfig` ユーティリティーは PGX32 (Raptor GFX) グラフィックスアクセラレータの設定用コマンドで、これには X11 ウィンドウシステムの一部を PGX32 (Raptor GFX) 用にデフォルト設定する機能も含まれます。以前のバージョンでは、このユーティリティーは `GFXconfig` という名称でした。

形式の項に記された `pgxconfig` の第 1 の形式では、指定したオプションを OWconfig ファイルに保存します。次回の PGX32 (Raptor GFX) デバイス上でのウィンドウシステム起動時に、ここで指定したオプションに従って PGX32 (Raptor GFX) デバイスを初期化します。OWconfig ファイルに保存されたオプションの更新内容は、異なる複数のウィンドウシステムセッションや再起動後のシステムでも有効となります。

その他の形式 (2、3、4 番目の形式) を使用した場合はオプション `-prconf`, `-propt`, `-help`, `-res ?` を呼び出すだけで、OWconfig ファイルに保存されているオプションを更新することはありません。更に、3 番目の形式を使用した場合はこれ以外のオプションはすべて無視されます。

`-i` オプションを使用すると、`pgxconfig` は、対話型モードで起動します。

一度にオプションを指定できる PGX32 (Raptor GFX) デバイスの数は、1 つだけです。

`pgxconfig` コマンドで指定できるのは、PGX32 (Raptor GFX) 固有のオプションだけです。デフォルトの表示色数、デフォルトのビジュアルクラスなどを指定する通常のウィンドウシステムのオプションは、`openwin` コマンド行のデバイス修飾子で指定してください。SUNWxwman パッケージに含まれている `Xsun`(1) マニュアルページを参照してください。

ユーザーは、更新する OWconfig ファイルを指定することもできます。デフォルトでは `/usr/openwin` ディレクトリツリーにあるマシン固有のファイルが更新されます。別のファイルを指定するには、`-file` オプションを使用します。たとえば、`/etc/openwin` ディレクトリツリーにあるシステム共通の OWconfig ファイルを代わりに更新することができます。

いずれの標準 OWconfig ファイルでも、書き込み権限があるのはスーパーユーザーだけです。

オプション　次のオプションを指定できます。

`-cachedpixmap` true | false	`false` に設定した場合、PGX32 (Raptor GFX) デバイスは、OpenWindows を実行するときにだけに 24 ビットを使用します。デフォルト値は `true` です。 アプリケーションの中には、ディスプレイデバイスに書き込むときにキャッシュされたピックスマップを使用するものがあります。このような手法を使用すると、出力が歪曲され、X サーバーがクラッシュする可能性があります。このような問題が発生した場合は、`-cachedpixmap` オプションを `false` に設定してください。
`-defaults`	すべてのオプションの値をそれぞれのデフォルト値に戻します。
`-depth` 8 | 24	スクリーンデプスをピクセル当り 8 または 24 ビットに設定します。24 ビット/ピクセルに設定すると、ウィンドウシステムで TrueColor グラフィックスを使用できます。
`-dev` *device-filename*	PGX32 (Raptor GFX) 特殊ファイルを指定します。デフォルトは `/dev/fbs/gfxp0`、または使用可能な場合であれば `/dev/fbs/raptor0` です。
`-file machine | system`	更新する `OWconfig` ファイルを指定します。*machine* が指定された場合は、`/etc/openwin` ディレクトリツリーにあるマシン固有の `OWconfig` ファイルが更新されます。*system* が指定された場合は、`/usr/openwin` ディレクトリツリーにある共通の `OWconfig` ファイルが更新されます。指定されたファイルがない場合は、新たに作成されます。ほかのオプションを指定していない限り、このオプションは有効ではありません。デフォルトは *machine* です。

`-help`
: `pgxconfig` コマンド行のオプションと機能の概要を一覧で表示します。

`-i`
: `pgxconfig` コマンドを対話型モードで起動します。

`-prconf`
: PGX32 (Raptor GFX) のハードウェア構成を表示します。以下に表示例を示します。

```
--- Hardware Configuration for /dev/fbs/gfxp0 ---
DAC: version 0x0
Type:
Board:
PROM: version 0x0
PROM Information:
RAM:
EDID Data:
Monitor Sense ID:
Card possible resolutions:  640x480x60, 800x600x75, 1024x768x60
               1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76
               1280x1024x60, 1152x900x66, 1152x900x76, 1280x1024x67
               960x680x112S, 960x680x108S, 640x480x60i, 768x575x50i,
               1280x800x76, 1440x900x76, 1600x1000x66, 1600x1000x76,
               vga, svga, 1152, 1280, stereo, ntsc, pal
Monitor possible resolutions:  720x400x70, 720x400x88, 640x480x60
               640x480x67, 640x480x72, 640x480x75, 800x600x56,
               800x600x60, 800x600x72, 800x600x75, 832x624x75,
               1024x768x87, 1024x768x60, 1024x768x70, 1024x768x75,
               1280x1024x75, 1280x1024x76, 1152x900x66, 1152x900x76,
               1280x1024x67, 960x680x112S, vga, svga, 1152, 1280
               stereo
Current resolution setting: 1280x1024x76
Possible depths:
Current depth: 8
```

`-propt`
: `-file` オプションで指定された `OWconfig` ファイルに書かれた PGX32 (Raptor GFX) オプションの値のうち、`-dev` オプションで指定されたデバイスに対するものすべてを表示します。`pgxconfig` の呼び出しが終了した後に、`OWconfig` ファイルに書き込まれるオプションの値を表示します。以下に表示例を示します。

```
--- OpenWindows Configuration for /dev/fbs/gfxp0 ---
```

```
OWconfig: machine
Video Mode: not set
Depth: 8+24
```

`-res` *video-mode* [`try` | `noconfirm` | `nocheck`]

PGX32 (Raptor GFX) デバイスに接続されているモニターの制御に使用する組み込みのビデオモードを指定します。

video-mode には、以下のいずれかの書式で指定できます。

*width*x*height*x*rate*

width はピクセル単位の画面の幅、*height* はピクセル単位の画面の高さ、*rate* は垂直方向の画面再描画周期です。`-res` では、再描画周期 rate の前の x は、@ でも代用できます。つまり周期の指定に限っては、たとえば 1280x1024@76 のような形式にも対応しています。オプションに `-res ?` を付けて `pgxconfig` を実行すると (コマンド形式の項に記された 3 番目の形式)、ビデオモードの一覧が表示されます。ビデオボードとモニターの両方が、すべての解像度をサポートしているわけではありません。`noconfirm` または `nocheck` オプションを指定しな

いで、ボードのサポートしていない解像度を入力した場合には、`pgxconfig` はその要求を許可しません。また、`nocheck` オプションを指定しないで、モニターのサポートしていない解像度を指定した場合には、その値を適用する前に確認を求めてきます。

Symbolic names

便宜上、一部のビデオモードには記号名が定義されています。*width*x*height*x*rate* の形式の代わりに、記号名を `-res` の引数として指定することができます。記号名 `none` は、ウィンドウシステムを実行すると、画面の解像度が現在デバイスにプログラムされているビデオモードになることを意味します。

`svga`	1024x768x60
`1152`	1152x900x76
`1280`	1280x1024x76
`vga`	640x480x60
`none`	デフォルトのコン

ソールの
解像度

-res オプションには、ビデオモードの直後に次の追加引数を指定することができます。追加引数は、単独でも複数でも指定できます。

noconfirm
: -res オプションを指定した際に、システムが使用可能であっても、表示出力のない状態になる場合があります。このような状況は、特定のコードが読み込まれた際のモニターセンスコードにあいまいさがあった場合などに発生します。このような事態を避けるために pgxconfig のデフォルトの動作では、この問題についての警告メッセージと、処理を継続するかどうかを確認するメッセージを表示します。noconfirm オプションを指定すると、pgxconfig コマンドはこの確認をせずに、要求のあったビデオモードにプログラムします。このオプションは、pgxconfig がシェルスクリプトから実行されている場合に便利です。

nocheck
: このオプションを指定すると、モニターセンスコードに基づく通常のエラーチェックが行われません。ユーザーによって指定されたビデオモードは、現在接

続されているモニターに適切かどうかにかかわらず受け付けられます。このオプションは、PGX32 (Raptor GFX) デバイスに異なるモニターを接続する場合に便利です。このオプションの指定は、noconfirm の指定も兼ねます。

try

このオプションを指定すると、指定したビデオモードを適用する前にテストすることができます。まず、指定したモードに基づいたテストパターンが表示されます。テストパターンが正常に表示された場合は、「y」(次いでキャリッジリターン) を入力します。「y」以外の文字を入力 (次いでキャリッジリターンを入力) した場合は、「no」となります。

構成済みデバイスの使用中は (たとえば、ウィンドウシステムを実行している場合)、try サブオプションを pgxconfig に指定してはなりません。予期せぬ結果になることがあります。try サブオプションを指定して pgxconfig を実行する前には、ウィンドウシステムを停止しておく必要があります。

`-res ?`	PGX32 およびモニターがサポートする解像度の一覧を表示します。
`-24only`	Openwindows の実行時に、PGX32 (Raptor GFX) デバイスが 24 ビットカラーのみを使用するようにします。

デフォルト

`pgxconfig`コマンド行で指定されていないオプションについては、対応する OWconfig ファイル中のオプションは更新されず、ファイル内の値がそのまま使用されます。ただし、`-depth` と `-24only`については、その限りではありません。

ウィンドウシステムを実行する際に、`pgxconfig` による PGX32 (Raptor GFX) のオプションの指定がまったくなかった場合は、デフォルト値が使用されます。オプションのデフォルト値は次のとおりです。

`-dev` `/dev/fbs/gfxp0`

`-file` `system`

`-res` none

`-res` オプションのデフォルト値 none とは、ウィンドウシステムが実行された場合に、画面解像度がそのデバイスに現在プログラムされているビデオモードになることを意味しています。

使用例

例 1 モニターの種類を変更する。

モニターの種類を、垂直周波数 76 Hz で解像度 1280 x 1024 に変更する例を以下に示します。

```
example# /usr/sbin/pgxconfig -res 1280x1024x76
```

ファイル

`/dev/fbs/gfxp0`	デバイス特殊ファイル
`/usr/openwin/server/etc/OWconfig`	システム構成ファイル
`/etc/openwin/server/etc/OWconfig`	マシン構成ファイル

関連項目

PGX32 PCI グラフィックスカード インストールマニュアル

名前

pkgadd – システムへのソフトウェアパッケージの転送

形式

```
pkgadd  [-nv] [-a admin] [-G] [-x proxy] [ [-M] -R root_path]
 [-r response] [-k keystore] [-P passwd] [-V fs_file]
 [-d device |  -d datastreampkginst |  all] [pkginst
 |  -Y category [, category]...]

pkgadd  -s [-d device |  -d datastreampkginst |  all]
 [pkginst |  -Y category [, category]...]
```

機能説明

pkgadd は、ソフトウェアパッケージの内容をインストール用の配布媒体またはディレクトリからシステムに転送します。-d *device* ソース指定子が指定されなかった場合、pkgadd は、デフォルトスプールディレクトリ (/var/spool/pkg) 内でパッケージを検索します。-s オプションを指定すると、pkgadd はパッケージをインストールするのではなく、パッケージをスプールディレクトリに書き込みます。

pkgadd ユーティリティーは、インストールされるパッケージと同じサイズの一時容量を必要とします。pkgadd は、$TMPDIR 環境変数があるかどうかを検査して、使用する一時ディレクトリを決定します。$TMPDIR が定義されていない場合、pkgadd は stdio.h に指定された P_tmpdir を使用します。P_tmpdir には、デフォルトの /var/tmp/ という値が定義されています。

別製品や Sun 以外のパッケージの中には、最新バージョンの pkgadd との完全な互換性が確保されていないものがあります。そのようなパッケージでは、インストールの開始時だけでなく、インストール中もユーザーの関与が必要かまたは、要求したスクリプトを root ユーザーで実行する必要があります。

Solaris 2.4 より前にリリースされた古いパッケージをインストールするには、次の環境変数を設定する必要があります。NONABI_SCRIPTS=TRUE

この環境変数を設定した場合、インストール中のキーボードから pkgadd の対話とパッケージ要求スクリプトの root での実行が可能になります。

noaccess [デフォルト] または install ユーザーではなく root ユーザーとして実行する必要のあるパッケージ要求スクリプトの場合、rscript_alt パラメータを admin(4) ファイルで使用し、適切な選択をします。admin(4) を参照してください。

Solaris 8 および Solaris 9 では、要求スクリプトを実行するデフォルトのユーザーは、root または nobody のどちらかであり、オペレーティングシステムのパッチレベルによって異なります。現在のリリースでは、デフォルトのユーザーは noaccess です。

大域ゾーン (zones(5) を参照) で pkgadd を実行する場合、要求スクリプト (pkgask(1M)) を含むパッケージは大域ゾーンにのみ追加されます。このパッケージは既存の非大域ゾーン、または将来作成される非大域ゾーンには転送されません。この動作は次に説明する -G オプションの結果に類似しています。

「形式」に示された -d、-Y、および *pkginst* 引数については、「オペランド」と次の「オプション」で説明します。

オプション　サポートされているオプションについて、次に説明します。-d *device* ソース指定子については、「オペランド」で説明します。

-a *admin*　デフォルトのインストール管理ファイルの代わりに使用するインストール管理ファイル *admin* を定義します。トークン none は *admin* ファイルの使用を無効にするため、ユーザーの関与が必要になります。完全なパス名を指定しない場合、pkgadd はまず現在の作業ディレクトリで管理ファイルを探します。指定した管理ファイルが現在の作業ディレクトリにないと、pkgadd は /var/sadm/install/admin ディレクトリで管理ファイルを探します。

-G　現在のゾーンのみにパッケージを追加します。大域ゾーンで使用される場合、そのパッケージは大域ゾーンのみに追加され、既存の非大域ゾーン、または将来作成される非大域ゾーンには転送されません。非大域ゾーンで使用される場合、そのパッケージは非大域ゾーンのみに追加されます。

パッケージの pkginfo ファイル内で SUNW_PKG_ALLZONES が true に設定されている場合、このオプションを指定すると、パッケージのインストールは失敗します。pkginfo(4) を参照してください。

-k *keystore*　パッケージ内に見つかったデジタル署名を検証するのに、認証局の信頼された証明書を入手するための場所を *keystore* で指定します。キーストアが指定されていない場合、デフォルトのキーストアの場所で信頼された有効な証明書を探します。詳細は、「キーストアの場所」を参照してください。

-M　クライアントのマウントポイントを決定するときに、$*root_path*/etc/vfstab ファイルを使用しないようにします。このオプションは、マウントポイントがサーバー上で適切であり、そのサーバーが Solaris 2.5 以前のリリースで安定して動作しているものと仮定します。

-n　非対話型モードでインストールします。インストールされたファイルの一覧は出力されません。デフォルトは対話モードです。

-P *passwd*　必要に応じて、-k で指定したキーストアを復号化するのに使用するパスワード。このオプションの引数の書式について詳細は、パスフレーズの引数 を参照してください。

-r *response*　直前の pkgask(1M) セッションからの出力が入っているファイルまたはディレクトリを指定します。このファイルは、対話モードにおけるパッケージからの質問に対する応答を提供します。*response* はフルパス名でなければなりません。

-R *root_path*　*root_path* として使用するディレクトリのフルパス名を定義します。すべてのファイル (パッケージシステム情報ファイルを含む) は *root_path* から始まるディレクトリツリーに再配置されます。*root_path* は、サーバーからクライアントにインストールするときに指定します (たとえば、`/export/root/client1`)。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

-s *spool*　パッケージをインストールするのではなく、*spool* ディレクトリに書き込みます。

-v　`pkgadd` によって実行されたすべてのスクリプトを追跡します。これらのスクリプトは *pkginst*`/install` ディレクトリにあります。このオプションは、手続き型スクリプトや非手続き型スクリプトをデバッグするときに使用します。

-V *fs_file*　クライアントのファイルシステム構成として *fs_file* を指定します。たとえば、$*root_path*`/etc/vfstab` ファイルが存在しない (あるいは、信頼できない) 場合に使用します。

-x *proxy*　パッケージをダウンロードする場合に使用する HTTP[S] プロキシを指定します。プロキシの書式は *host*:*port* です。ここで、*host* は HTTP[S] プロキシのホスト名、*port* はそのプロキシに関連付けられたポート番号です。このスイッチは、プロキシを指定するほかのすべての方法より優先します。デフォルトのプロキシを指定する代わりの方法については、環境 を参照してください。

オプションまたはオペランドを指定しないで実行すると、`pkgadd` は `/var/spool/pkg` (デフォルトのスプールディレクトリ) を使用します。

オペランド　次のオペランドがサポートされています。

ソース　デフォルトでは、pkgadd はインストールまたはスプールするパッケージのインスタンスを探す場合、`/var/spool/pkg` ディレクトリを探します。任意で、インストールまたはスプールされたパッケージインスタンスのソースを指定することもできます。その場合、下記を指定します。

-d *device*
-d *datastream pkgname,...* | `all`　*device* からパッケージをインストールまたはコピーします。*device* には次のいずれかを指定できます。

- ディレクトリの完全パス、またはテープ、フロッピーディスク、またはリムーバブルディスクの識別子 (`/var/tmp`、`/floppy/`*floppy_name* など)。
- デバイス別名 (`/floppy/floppy0` など)。
- `pkgtrans` によって作成されたデータストリーム (`pkgtrans`(1) を参照)。
- `pkgtrans` によって作成されたデータストリームを指す URL。サポートされている URI (Universal Resource Identifier) は、`http:` と `https:` です。

上に示した `-d` 指定子の 2 番目の形式は、データストリーム指定時に使用する構文です。その場合、パッケージ名のコンマ区切りリスト、キーワード `all` のいずれかを指定する必要があります。

インスタンス

デフォルトでは、`pkgadd` は指定したソースを検索し、ユーザーが、ソースで見つかったどのパッケージをインストールするか選択できるような対話式メニューがあります。代わりの方法としては、次のオペランドを指定して、インストールするパッケージインスタンスを指定できます。

pkginst

インストールするパッケージインスタンスまたはインスタンスのリストを指定します。ソースメディアで使用可能なすべてのパッケージを指定するには、トークン `all` を使用できます。パッケージのすべてのインスタンスを指定するには、書式 *pkginst*`.*` を使用できます。

アスタリスク文字 (*) はあるシェルにとっては特殊文字なので、その場合はエスケープする必要があります。C シェルの場合、アスタリスクは単一引用符 (') で囲むか、前にバックスラッシュ (\) をつける必要があります。

`-Y` *category*[,*category*...]

パッケージの `pkginfo`(4) ファイルに保存されている `CATEGORY` パラメータの値に基づいてパッケージをインストールします。`CATEGORY` の値が指定したカテゴリの 1 つに一致するソースメディア上のすべてのパッケージが、インストールまたはスプール用に選択されます。

キーストアの場所

`pkgadd` または `patchadd` のようなパッケージツールまたはパッチツールは、パッケージまたはパッチで見つかったすべての署名の検証を行う場合、信頼された証明書のセットを使用します。パッケージまたはパッチに署名が 1 つも含まれていない場合は署名の検証が省略されます。証明書はいろいろな場所にある可能性があります。`-k` *keystore* が指定され、*keystore* がディレクトリである場合、証明書を使用する基本のディレクトリとして *keystore* が仮定されます。*keystore* がファイルの場

合、このファイル自身が必要なキーおよび証明書をすべて持っていると仮定されます。-k が指定されない場合、基本ディレクトリとして /var/sadm/security が使用されます。

指定した基本ディレクトリの中で、検索される保存場所は、検索を行なっているアプリケーション、および検索しているストアの種類に基づいて異なります。次のようなディレクトリが、指定された順番に検索されます。

1. <*store_dir*>/<*app_name*>/<*store_type*>
2. <*store_dir*>/<*store_type*>

ここで、<*store_dir*> が -k で指定されたディレクトリである場合、<*app_name*> は検索を行なっているアプリケーションの名前で、<*store_type*> は keystore (非公開鍵の場合)、certstore (信頼されていない公開鍵証明書の場合)、truststore (信頼された認証局の証明書の場合) のいずれかです。

たとえば、pkgadd が -k /export/certs で実行された場合、信頼できる場所を検索するのに次の場所が順番に検索されます。

1. /export/certs/pkgadd/truststore
2. /export/certs/truststore

この検索順序により、管理者はほとんどのアプリケーションにそれぞれ 1 つの場所を、また、特定のアプリケーションに特別な証明書の場所を特定できます。

キーストアおよび証明書の形式

pkgtrans および patchadd のようなパッケージ用ユーティリティーおよびパッチ用ユーティリティーは、パッケージおよびパッチに署名し、また必要に応じて検証するためにキーおよび証明書のセットにアクセスする必要があります。

キーストアの場所 に指定された次の検索パターンにしたがって見つかったキーストアファイルは、それぞれ自己保有型の PKCS#12 形式ファイルである必要があります。

pkgtrans を使用してパッケージに署名する場合、 certstore に 1 つ以上の公開鍵証明書が含まれている場合、 パッケージまたはパッチに署名するのに -a オプションで識別できる、または選択できるためには、各公開鍵には friendlyName 属性が含まれている必要があります。さらに、-a で選択した公開鍵証明書、および certstore で見つかった公開鍵証明書は、キーストア内に関連する秘密鍵を持っていなくてはなりません。

証明書およびキーを PKCS#12 キーストアにエクスポートおよびインポートするためには、いくつかのブラウザおよびユーティリティーを使用できます。たとえば、OpenSSL ツールキットと一緒に pkgadd を使用するためには、信頼される証明書は Mozilla からエクスポート可能で、さらに PKCS#12 キーストアにインポート可能です。

パスフレーズの引数

pkgtrans および pkgadd は、パスワードの引数を受け入れます。一般的には -p が パスワードを指定します。これにより、パスワードをいろいろなソースから入手できます。これら 2 つのコマンドのオプションはどちらも 1 つの引数を使用し、その書式は下記で示します。パスワードの引数が 1 つも指定されていないがパスワードが必要な場合、ユーザーはパスワードを入力するよう促されます。一般的にこれはエコー機能がオフの状態で、現在の端末から読みとられます。

`pass:`*password* 実際のパスワードは *password* です。パスワードは ps のようなユーティリティーには見えるため、この書式はセキュリティーがそれほど重要でない場合にのみ使用するべきです。

`env:`*var* パスワードを環境変数 *var* から取得します。プラットフォームによっては、ほかのプロセスの環境が見えるため、このオプションは注意して使用する必要があります。

`file:`*pathname* *pathname* に含まれる 1 行目がパスワードです。*pathname* は通常のファイルを参照する必要がありません。というのは、デバイスまたは名前のついたパイプを参照できます。たとえば、標準入力からパスワードを読みとるためには、 `file:/dev/stdin` を使用します。

`console` `/dev/tty` からパスワードを読みとります。

使用例

例 1 Solaris DVD からパッケージをインストールする

次の例は、Solaris DVD からパッケージをインストールします。インストールするパッケージの名前を入力するように求められます。

```
example# pkgadd -d /cdrom/cdrom0/s0/Solaris_10/Product
```

例 2 データストリームから一連のパッケージをインストールする

次に示すコマンド例は、-d ソース指定子で指定されたデータストリーム内のすべてのパッケージをインストールします。このコマンドを実行する前に、pkgtrans(1) コマンドを使ってこのデータストリームを作成しておく必要があります。

```
example# pkgadd -d /var/tmp/datastream all
```

キーワード all は、指定されたデータストリーム内で見つかったすべてのパッケージがインストールされることを表します。

終了ステータス

`0` 正常終了。

`1` 致命的なエラー。

`2` 警告。

`3` 割り込み。

4 管理。

5 管理。対話が必要。pkgadd -n を使用してはならない。

10 すべてのパッケージのインストール後に再起動する。

20 当該パッケージのインストール後に再起動する。

環境

HTTPPROXY HTTP プロキシホストを指定します。管理ファイル設定、および http_proxy 環境変数より優先します。

HTTPPROXYPORT HTTPPROXY で指定したホストに接続するときに使用するポートを指定します。HTTPPROXY が設定されていない場合は無視されます。

http_proxy プロキシホストおよびプロキシポートを指定するための URL 書式です。管理ファイル設定より優先します。

ファイル

/var/sadm/install/logs/ pkgadd がソフトウェアインストールのインスタンスのログを記録する場所。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWpkgcmdsu
インタフェースの安定性	開発中

関連項目

pkginfo(1), pkgmk(1), pkgparam(1), pkgproto(1), pkgtrans(1), installf(1M), pkgadm(1M), pkgask(1M), pkgchk(1M), pkgrm(1M), removef(1M), admin(4), pkginfo(4), attributes(5), zones(5)

『Application Packaging Developer's Guide』

http://www.openssl.org

注意事項

スプールディレクトリにパッケージを転送するときは、-r、-n、および -a オプションは使用できません。

-r オプションには、ファイル名と同様に、ディレクトリ名も指定できます。このディレクトリには、関連するパッケージと名前を共有する複数の応答ファイルを格納できます。これにより、1 回の pkgadd 呼び出しで、複数の対話型パッケージを追加できます。複数の対話型パッケージを追加するには、パッケージごとに応答ファイルが必要です。パッケージと同じ名前 (たとえば、pkinst1 や pkinst2) の応答ファイルを作成しておけば、-r オプションのあとに、これらの応答ファイルが入ったディレクトリを指定するだけで済みます。

-n オプションを指定した場合、インストールを完了するのに対話が必要になると、インストールが中断します。

デフォルトの *admin* ファイルの制限が厳しすぎる場合、パッケージのインストール時に非対話型モードで操作できるように、管理ファイルを変更する必要があります。詳細については、admin(4) のマニュアルページを参照してください。

パッケージストリームが -d で指定されている場合、ストリーム中にデジタル署名が見つかった場合、デフォルトの動作として、証明書と署名を検証しようとします。この動作は、admin ファイル設定で置き換えられます。詳細は、admin(4) を参照してください。

名前

pkgask – stores answers to a request script

形式

`pkgask [-d` *device*`] [-R` *root_path*`] -r` *response pkginst*`...`

機能説明

`pkgask` allows the administrator to store answers to an interactive package (one with a `request` script, that is, a user-created file that must be named `request`). Invoking this command generates a response file that is then used as input at installation time. The use of this response file prevents any interaction from occurring during installation since the file already contains all of the information the package needs.

オプション

The following options are supported

`-d` *device*
: Run the request script for a package on *device*. *device* can be a directory pathname or the identifiers for tape, floppy disk or removable disk (for example, `/var/tmp`, `/dev/diskette`, and `/dev/dsk/c1d0s0`). The default device is the installation spool directory.

`-R` *root_path*
: Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

  注 – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones`(5).

`-r` *response*
: Identify a file or directory which should be created to contain the responses to interaction with the package. The name must be a full pathname. The file, or directory of files, can later be used as input to the pkgadd(1M) command.

オペランド

The following operands are supported:

pkginst
: Specify the package instance, or list of instances for which `request` scripts will be created. The token `all` may be used to refer to all packages available on the source medium.

終了ステータス

`0`
: Successful completion.

`>0`
: An error occurred.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

関連項目

`pkginfo`(1), `pkgmk`(1), `pkgparam`(1), `pkgproto`(1), `pkgtrans`(1), `installf`(1M), pkgadd(1M), `pkgchk`(1M), pkgrm(1M), `removef`(1M), `admin`(4),`attributes`(5)

『Application Packaging Developer's Guide』

注意事項

The `-r` option can be used to indicate a directory name as well as a filename. The directory name is used to create numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when you will be adding multiple interactive packages with one invocation of pkgadd(1M). Each package would need a response file. To create multiple response files with the same name as the package instance, name the directory in which the files should be created and supply multiple instance names with the `pkgask` command. When installing the packages, you will be able to identify this directory to the pkgadd(1M) command.

If the default `admin` file is too restrictive, the administration file may need to be modified to allow for total non-interaction during a package installation. See`admin`(4) for details.

名前　pkgrm – システムからのパッケージの削除

形式

```
pkgrm  [-nv] [-a admin] [ [-A |  -M] -R root_path] [-V fs_file]
 [pkginst... |  -Y category[,category...]]

pkgrm  -s spool [pkginst... |  -Y category[,category...]]
```

機能説明　pkgrm はすでにインストールしてある、あるいは、部分的にインストールしたパッケージをシステムから削除します。このとき、削除するパッケージに対して依存性のあるパッケージが存在するかどうかを検査します。パッケージに依存性がある場合に行われる処理は admin ファイルに定義されています。

コマンドのデフォルトの動作状態は対話モードです。つまり、処理中にプロンプトメッセージが表示されるので、管理者はどのような処理が行われるかを確認できます。非対話モードで処理を実行するには、-n オプションを使用します。

-s オプションを使用すると、特定のディレクトリにスプールされているパッケージを削除できます。

別製品や Sun 以外のパッケージの中には、最新バージョンの pkgrm との完全な互換性が確保されていないものがあります。このようなパッケージでは、削除の開始時だけでなく、削除処理中にもユーザーの関与が必要です。

Solaris 2.4 より前にリリースされた古いパッケージをインストールするには、次の環境変数を設定します。

```
NONABI_SCRIPTS=TRUE
```

この環境変数を設定していると、削除処理の全工程で、キーボードから pkgrm との対話が可能になります。

オプション　次のオプションを指定できます。

-a *admin*　デフォルトのインストール管理ファイルの代わりに、インストール管理ファイル *admin* を使用します。pkgrm はまず現在の作業ディレクトリで管理ファイルを探します。指定した管理ファイルが現在の作業ディレクトリにないと、pkgrm は /var/sadm/install/admin ディレクトリで管理ファイルを探します。

-A　パッケージファイルをクライアントのファイルシステムから無条件に削除します。ファイルが他のパッケージによって共有されている場合、そのファイルは、デフォルトではクライアントのファイルシステムから削除されません。

-M　クライアントのマウントポイントを決定するときに、$*root_path*/etc/vfstab ファイルを使用しないようにします。このオプションは、マウントポイントがサーバー上で適切であり、そのサーバーが Solaris 2.5 以前のリリースで安定して動作しているものと仮定します。

`-n` 非対話モード。対話の必要がある場合、コマンドは終了します。

このオプションを使用するには、コマンドを呼び出すときに、少なくとも 1 つのパッケージのインスタンスを指定しておく必要があります。非対話モードでパッケージを削除するには、特定の状態が存在しているか、制限のない `admin` ファイルを使用する必要があります。

`-R` *root_path* *root_path* として使用するディレクトリのフルパス名を定義します。すべてのファイル (パッケージシステム情報ファイルを含む) は *root_path* から始まるディレクトリツリーに再配置されます。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティを損ねたり、非大域ゾーンのファイルシステムを損傷する可能性があります。`zones`(5) のマニュアルページを参照してください。

`-s` *spool* 指定したパッケージを *spool* ディレクトリから削除します。スプールされているパッケージのデフォルトディレクトリは `/var/sadm/pkg` です。

`-v` `pkgrm` によって実行されたすべてのスクリプトを追跡します。これらのスクリプトは *pkginst*`/install` ディレクトリにあります。このオプションは、手続き型スクリプトや非手続き型スクリプトをデバッグするときに使用します。

`-V` *fs_file* クライアントのファイルシステム構成として *fs_file* を指定します。たとえば、$*root_path*`/etc/vfstab` ファイルが存在しない (あるいは、信頼できない) 場合に使用します。

`-Y` *category* インストールまたはスプールされているパッケージの `pkginfo`(4) ファイルに格納されている `CATEGORY` パラメータの値に基づいてパッケージを削除します。このオプションでは、`CATEGORY` が `system` に設定されているパッケージをファイルシステムから削除することはできません。

オペランド 次のオペランドを指定できます。

pkginst 削除するパッケージを指定します。パッケージのすべてのインスタンスを削除するには、*pkginst*`.*` という形式を使用します。

シェルによっては、アスタリスク文字 (*) が特殊な意味を持つことがあり、* をエスケープしなければならない場合があります。C シェルでは、* は単一引用符 (') で囲むか、バックスラッシュ (\) を前につける必要があります。

使用例

例 1 SUNWjunk のすべてのインスタンスを client1 から削除する

次の例は、SUNWjunk のすべてのインスタンスを client1 から削除します。

```
example% pkgrm -R /export/root/client1 SUNWjunk*
```

-R の使い方については、前述されているこのオプションへの注意事項に留意してください。

終了ステータス

次の終了ステータスが返されます。

0 正常終了

1 致命的なエラー

2 警告

3 割り込み

4 管理

10 すべてのパッケージの削除後に再起動する

20 当該パッケージの削除後に再起動する

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

pkginfo(1), pkgmk(1), pkgparam(1), pkgproto(1), pkgtrans(1), installf(1M), pkgadd(1M), pkgask(1M), pkgchk(1M), removef(1M), admin(4),pkginfo(4), attributes(5)

『Application Packaging Developer's Guide』

名前　pmconfig – 電源管理システムの設定

形式　`/usr/sbin/pmconfig [-r]`

機能説明　pmconfig ユーティリティーは、電源管理システムと保存停止・復元再開機能を設定します。ユーザーは、/etc/default/power のキーワードである PMCHANGEPERM で許可されている場合にのみ pmconfig を使用して電源管理設定を変更する権限を持ちます。/etc/default/power のキーワード CPRCHANGEPERM で許可されている場合にのみ pmconfig を使用して保存停止・復元再開機能を変更する権限を持ちます。/etc/default/power のキーワードである PMCHANGEPERM と CPRCHANGEPERM の詳細については、以下に記述する「ファイル」セクションを参照してください。

ユーザーの権限に基づいて、pmconfig は、最初に電源管理システムまたは保存停止・復元再開 (またはその両方) をリセットします。それから新しい電源管理システムまたは保存停止・復元再開 (またはその両方) の設定を /etc/power.conf から読み取り、その新しい設定を有効にするコマンドを発行します。pmconfig ユーティリティーはシステムの起動時に実行されます。また、このユーティリティーは、/etc/power.conf ファイルを手動で変更した後でコマンド行から実行することもできます。/etc/power.conf ファイルを編集した場合、その変更を有効にするには、pmconfig を実行する必要があります。

電源管理および保存停止・復元再開の設定変更に、より便利なインタフェースは、dtpower(1M) です。

オプション　以下のオプションを使用することができます。

`-r`　電源管理システムと保存停止・復元再開状態をデフォルトの状態にリセットして終了します。このオプションを使用するには、電源管理システムと保存停止・復元再開の設定に対する権限が必要です。

終了ステータス　以下の終了ステータスが返されます。

`0`　正常終了

`>0`　エラーが発生

ファイル

`/etc/power.conf`	システムの電源管理設定ファイル
`/etc/default/power`	システムの電源管理システムと保存停止・復元再開機能に対する権限を制御するファイル。PMCHANGEPERM キーワードは電源管理設定の権限を制御し、 CPRCHANGEPERM キーワードは保存停止・復元再開機能の設定権限を制御します。

以下に指定できる値を示します。

`all`	すべてのユーザーが設定を変更できます。
`-`	スーパーユーザーだけが設定を変更できます。

<user1, user2,...>	このユーザーリストに指定されているユーザーまたはスーパーユーザーは、設定の変更ができます。このリストには、空白またはコンマ (,) を使用して複数のユーザーを指定できます。このリストは、< と > で囲む必要があります。
console-owner	システムコンソールデバイスノードを所有するユーザーまたはスーパーユーザーだけが設定を変更できます。

デフォルトの値は、PMCHANGEPERM=console-owner および CPRCHANGEPERM=console-owner です。

属性

以下の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWpmu
インタフェースの安定性	不安定

関連項目

svcs(1), powerd(1M), power.conf(4), svcadm(1M), attributes(5), smf(5), cpr(7), pm(7D)

『電源管理システム ユーザーマニュアル』

注意事項

pmconfig サービスはサービス管理機能 smf(5) により次のサービス識別子の下で管理されます。

```
svc:/system/power:default
```

有効化、無効化、再起動要求など、このサービスに関する管理操作は、svcadm(1M) を使用して実行できます。サービスの状態は svcs(1) コマンドを使用して照会できます。

診断

プログラムが設定ファイルを開くことができない場合は、標準エラー出力にエラーメッセージを出力します。プログラムが設定ファイル内の構文エラーを検出した場合は、エラーメッセージと設定ファイル内のエラーの行番号を出力します。プログラムは、その行の残りの情報は処理せずに次の行を処理します。エラーを含んでいる行の設定情報の中で、すでに処理されているものは使用されます。電源管理システムおよび (または) 保存停止・復元再開機能の設定を変更する権限がない場合、また設定ファイル中にユーザーが権限を持っていないエントリがある場合、権限を持つエントリについてだけ処理が行われ、権限のない部分についてはエラーが出力されます。

名前　pooladm – 資源プール機能の有効化および無効化

形式

```
/usr/sbin/pooladm  [-n] [-s] [-c] [filename] |-x

/usr/sbin/pooladm  [-d |  -e]
```

機能説明　pooladm コマンドは、プールやセットに対する管理操作を行います。pooladm は、指定された名前のファイルを読み取り、そこに含まれているプール構成を有効にします。

pooladm は、現在のプール実行時構成を更新する前に、その構成が正しいか検査します。

オプションを指定しないで実行すると、pooladm は、現在実行中のプール構成を表示します。

オプション　サポートしているオプションは、次のとおりです。

-c　指定された場所にある構成をインスタンス化します。ファイル名が指定されなかった場合のデフォルトは、/etc/pooladm.conf になります。

-d　プール機能を無効にして、プールを操作できないようにします。

-e　プール機能を有効にして、プールを操作できるようにします。

-n　現在のアクティブファイルを実際に更新することなく構成を検査します。構文エラーがないことと、現在のシステム上で構成をインスタンス化できることを検査します。アプリケーション固有のプロパティーの検査は実行されません。

-s　指定された場所を現在の動的構成の内容で更新します。

このオプションを指定した場合、更新する構成に対する更新権限が必要となります。このオプションを -c オプションと併用した場合、動的な構成が更新されてから、静的な場所が更新されます。

-x　現在アクティブになっているプール構成を削除します。定義されたすべての資源を破棄し、それまで区分化されていたすべてのコンポーネントを、それぞれのデフォルト資源に戻します。

オペランド　次のオペランドがサポートされています。

filename　このファイル内に格納されている構成を使用します。

使用例

例 1　構成をインスタンス化する

次のコマンドは、/home/admin/newconfig に保存されている構成をインスタンス化します。

```
example# /usr/sbin/pooladm -c /home/admin/newconfig
```

例2 インスタンス化することなく構成を検査する

次のコマンドは、/home/admin/newconfig に保存されている構成をインスタンス化しようとします。発生したエラー条件はすべて表示しますが、アクティブ構成を実際に変更することはありません。

```
example# /usr/sbin/pooladm -n -c /home/admin/newconfig
```

例3 現在の構成を削除する

次のコマンドは、現在のプール構成を削除します。

```
example# /usr/sbin/pooladm -x
```

例4 プール機能を有効にする

次のコマンドは、プール機能を有効にします。

```
example# /usr/sbin/pooladm -e
```

例5 SMF を使ってプール機能を有効にする

次のコマンドは、サービス管理機能を使ってプール機能を有効にします。smf(5) を参照してください。

```
example# /usr/sbin/svcadm enable svc:/system/pools:default
```

例6 アクティブ構成を指定された場所に保存する

次のコマンドは、アクティブ構成を /tmp/state.backup に保存します。

```
example# /usr/sbin/pooladm -s /tmp/state.backup
```

ファイル

/etc/pooladm.conf pooladm の構成ファイル。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWpool
インタフェースの安定性	以下を参照。

この呼び出しは開発中です。出力は不安定 (Unstable) です。

関連項目

poolcfg(1m), poolbind(1M), psrset(1M), svcadm(1M), pset_destroy(2), libpool(3LIB), attributes(5), smf(5)

『Solaris のシステム管理 (Solaris コンテナ : 資源管理と Solaris ゾーン)』

注意事項

スケジューリングクラスなど、区分化可能な資源へのバインディングのかたちで存在していない資源バインドは、pooladm -x 操作で必ずしも変更されるとは限りません。

デフォルトでは、プール機能は Solaris 起動時に有効になりません。pooladm -e は、プール機能を明示的に有効にします。プールが有効になると、プロセッサ区分化とプロセスバインドに関する特定の API の動作が変更されます。libpool(3LIB) を参照してください。

プロセッサセットが作成されたシステム上でプール機能を有効にすることはできません。プール機能を有効にする前に、psrset(1M) コマンドまたは pset_destroy(2) を使ってプロセッサセットを手動で破棄してください。

資源プール機能は smf(5) サービスであるため、その有効化および無効化を標準の SMF インタフェース経由で行うこともできます。

名前　poolcfg – 資源プール構成ファイルの作成および変更

形式

```
/usr/sbin/poolcfg  -c command [-d |  [filename]]

/usr/sbin/poolcfg  -f command_file [-d |  [filename]]

/usr/sbin/poolcfg  -h
```

機能説明

poolcfg コマンドは、プールやセットに対する構成操作を行います。これらの操作は既存の構成に対して実行され、「指定された構成ファイルを変更する」という方法を取ります。-d オプションを使用した場合、カーネル状態に対して変更が加えられます。結果として得られた構成を実際に有効化するには、pooladm(1M) コマンドを使用します。

プール構成ファイルは、poolcfg 自体を使ったか、あるいは libpool(3LIB) を使って直接構築された構造化ファイルです。

このツールを使って作成された構成は、pooladm が特定のターゲットホスト上で構成をインスタンス化する際に使用できます。

オプション　サポートしているオプションは、次のとおりです。

-c *command*	*command* に特定の編集コマンドを指定します。「使用法」を参照してください。
-d	カーネル状態に対して直接操作を行います。*filename* は使用できません。
-f *command_file*	*command_file* からコマンドを取得します。*command_file* はいくつかの編集コマンドから構成されます。コマンドは 1 行に 1 つずつ記述されます。
-h	編集コマンドの構文に関する詳細情報を表示します。

使用法

スクリプト　スクリプトは、複数の編集コマンドで構成されます。スクリプト内の各行に、編集コマンドを 1 つずつ記述します。次のような書式を使用します。

info [*entity-name*]	構成 (または指定された部分) を、人間が読める形式で標準出力に表示します。エンティティーが指定されなかった場合、システムの情報が表示されます。したがって、`poolcfg -c 'info' afile` は、`poolcfg -c 'info system name' afile` と同等の呼び出しになります。
create *entity-name* [*property-list*]	指定されたタイプと名前のエンティティーを作成します。

destroy *entity-name*	指定されたエンティティーを削除します。
modify *entity-name* [*property-list*]	指定されたエンティティー上の指定された一連のプロパティーを変更します。
associate *pool-name* [*resource-list*]	特定のプールに 1 つ以上の資源を接続するか、1 つ以上の既存の接続を置き換えます。
transfer to [*resourcetype*] *name*[*component-list*]	1 つ以上の個別コンポーネントを特定の資源に転送します。
transfer [*quantity*] from [*resourcetype*] [*src*] to [*tgt*]	特定数の資源を *src* から *tgt* に転送します。
transfer [*quantity*] to [*resourcetype*] [*tgt*] from [*src*]	特定数の資源を *src* から *tgt* に転送します。
discover	システムエンティティーを 1 つ作成します。このエンティティーには、1 つのプールエンティティーと現在のシステム構成に一致する資源が格納されます。発見されたすべての資源タイプのすべての資源がファイル内に記録されます。その際、単一のプールが各資源タイプのデフォルト資源を参照するように記録されます。 `poolcfg` がカーネルに対して直接操作を行う場合、このコマンドは何も行いません。`-d` オプションを参照してください。 このコマンドはできるだけ使用しないでください。構成を作成するために推奨されている方法は、pooladm(1M) の `-s` オプションを使って動的構成をエクスポートすることです。
rename *entity-name* to *new-name*	システム上の特定のエンティティーの名前を別の名前に変更します。

プロパティーリスト　プロパティーリストは次のように指定します。

```
( proptype name = value [ ; proptype name = value ]* )
```

ここで、特定のプロパティータイプと名前のペアについて、シーケンス内で最後に定義されたものが有効となります。プロパティーを削除するには、~ proptype name を使用します。

資源リスト

資源リストは次のように指定します。

```
( resourcetype name [ ; resourcetype name ]* )
```

ここで、特定の資源について、シーケンス内で最後に指定されたものが有効となります。資源リストでは、削除用の構文は存在しません。

コンポーネントリスト

コンポーネントリストは次のように指定します。

```
( componenttype name [ ; componenttype name ]* )
```

ここで、特定のコンポーネントについて、シーケンス内で最後に指定されたものが有効となります。コンポーネントリストでは、削除用の構文は存在しません。

認識されるエンティティー

system　マシンレベルのエンティティー

pool　資源関連付けの集合に名前を付けたもの

資源タイプ

pset　プロセッサセット資源

プロパティータイプ

boolean　2 つの値 true、false のいずれかを取ります。

int　64 ビット符号付き整数値。

uint　64 ビット符号なし整数値。

string　文字列は引用符 (") で区切ります。formats(5) で定義されている文字エスケープシーケンスがサポートされます。

float　科学用の表記はサポートされません。

使用例

例 1　poolcfg スクリプトの記述

次の poolcfg スクリプトは、Accounting という名前のプールと、プロセッサセット small-1 を作成します。まず、プロセッサセットが作成されます。次に、プールが作成され、それにプロセッサセットが関連付けられます。

```
create pset small-1 ( uint pset.min = 1 ; uint pset.max = 4)
create pool Accounting
associate pool Accounting ( pset small-1 )
```

例 2　pool_0 のレポート

次のコマンドは、pool_0 に関するレポートを、人間が読める形式で標準出力に出力します。

例 2 pool_0 のレポート (続き)

```
 # poolcfg -c 'info pool pool_0' /etc/pooladm.conf
```

例 3 pool_0 とその関連付けの削除

次のコマンドは、pool_0 とその関連付けを削除します。ただし、すでに関連付けられていた資源は削除しません。

```
# poolcfg -c 'destroy pool pool_0' /etc/pooladm.conf
```

例 4 現在の構成の表示

次のコマンドは現在の構成を表示します。

```
$ poolcfg -c 'info' /etc/pooladm.conf
system example_system
               int system.version 1
               boolean system.bind-default true
               string system.comment Discovered by libpool

               pool pool_default
                    boolean pool.default true
                    boolean pool.active true
                    int pool.importance 5
                    string pool.comment
                    string.pool.scheduler FSS
                    pset pset_default

               pset pset_default
                    int pset.sys_id -1
                    string pset.units population
                    boolean pset.default true
                    uint pset.max 4294967295
                    uint pset.min 1
                    string pset.comment
                    boolean pset.escapable false
                    uint pset.load 0
                    uint pset.size 2

                    cpu
                         int cpu.sys_id 0
                         string cpu.comment

                    cpu
                         int cpu.sys_id 2
                         string cpu.comment
```

例 5　カーネル内で ID 2 の cpu をプロセッサセット pset1 に移動する

次のコマンドは、カーネル内で、ID 2 の cpu をプロセッサセット pset1 に移動します。

```
# poolcfg -dc 'transfer to pset pset1 ( cpu 2 )'
```

例 6　カーネル内で 2 つの cpu をプロセッサセット pset1 からプロセッサセット pset2 に移動する

次のコマンドでは、カーネル内で、2 つの cpu をプロセッサセット pset1 からプロセッサセット pset2 に移動します。

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWpool
インタフェースの安定性	以下を参照。

この呼び出しは開発中です。出力は不安定 (Unstable) です。

関連項目

pooladm(1M), poolbind(1M), libpool(3LIB), attributes(5), formats(5)

『Solaris のシステム管理 (Solaris コンテナ : 資源管理と Solaris ゾーン)』

名前　powerd – 電源管理デーモン

形式　`/usr/lib/power/powerd [-n]`

機能説明　`powerd` デーモンは pmconfig(1M) で起動し、システム動作の監視、および保存停止 - 復元再開機能を使用した自動停止を行います。システムが保存停止されると、電源が落とされる前に、完全な現在の状態がディスクに保存されます。リブート時に、システムは自動的に復元再開操作を開始し、システムが保存停止する直前と同じ状態に復元されます。

システムの停止の直前に、powerd デーモンは停止について syslogd(1M) に通知し、syslogd(1M) はこれをブロードキャストします。

オプション　以下のオプションを使用することができます。

`-n`　通知なし。デーモンは syslogd(1M) に通知しないでシステムを停止します。

ファイル　`/etc/power.conf`　電源管理設定情報ファイル

属性　以下の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWpmu
インタフェースの安定性	不安定

関連項目　pmconfig(1M), `dtpower`(1M), syslogd(1M), `power.conf`(4), `attributes`(5), `cpr`(7), `pm`(7D)

『電源管理システムユーザーマニュアル』

名前

prodreg – Solaris Product Registry の管理

形式

`prodreg [--help] | [`*subcommand operand* `...]`

機能説明

`prodreg` は、Solaris Product Registry のコンポーネントのブラウズ、登録解除、およびアンインストールに使用するユーティリティです。

一部のインストーラは、`libwsreg`(3LIB) を使って情報を登録します。Solaris Product Registry には、このインストール済みソフトウェアに関する情報が含まれています。

Solaris Product Registry のデータベースのパスは、インストール済みファイルシステムのルートを基点としています (通常、ファイルシステムのルートを基点とする場合はスラッシュ (`/`) で始まります)。ただし、Live Upgrade インストールの実行時には、代替ルートを基点とする異なった Solaris Product Registry インストールデータベースが使用されることがあります。`live_upgrade`(5) を参照してください。

Registry データベースは、インストーラに、インストール済みソフトウェアの情報を伝達します。Registry や `prodreg` ユーティリティが直接インストールまたはアンインストールを実行することはありません。`prodreg` は、`prodreg` 自体またはその他の方法で起動され、外部で実行されるインストーラをサポートします。

`prodreg` コマンドでは、サブコマンドを使って、コマンド行または GUI ビューアと同等の機能を使用できます。GUI ビューアには 2 つのバージョンがあります。1 つは Java Swing GUI (デフォルト)、もう 1 つは Java AWT GUI です。後者は、Java Swing がサポートされていない環境で使用します。

`unregister` サブコマンドは、CLI (コマンド行インタフェース) からは実行できても GUI からは実行できない唯一の機能です。たとえば、アンインストーラを使用せずに手動でソフトウェアを削除した場合、Product Registry に破損が生じる可能性があります。削除されたソフトウェアのエントリが、この後に実行されるインストーラを混乱させることがあるためです。このような場合は、`unregister` サブコマンドを使って無効なエントリを強制的に削除します。再帰オプションや強制オプションを使用してソフトウェアの登録を解除するときは、Registry 内の有効なエントリを誤って削除しないように注意してください。

`prodreg` コマンドは、あくまでも、その時点の Registry の内容を表示するものです。これは、GUI から起動した場合もコマンド行インタフェースブラウザから起動した場合も変わりません。`prodreg` ビューアの起動後または起動と同時に、ソフトウェアをインストールまたはアンインストールした場合、コマンドの出力結果と実際の Solaris Product Registry の内容が一致しなくなることがあります。

サブコマンド

`prodreg` コマンドには、サブコマンドを指定せずにオプションを指定できます。サブコマンドの指定を省略すると、`swing` サブコマンドが指定されたものと見なされます。

サポートされているサブコマンドは次のとおりです。

awt

Java awt GUI を起動します。

awt サブコマンドの形式は次のとおりです。

```
awt [-R alt_root | --help]
```

browse

コマンド行インタフェースから、テキスト形式で Solaris Product Registry の内容を表示します。この出力結果から、Product Registry ツリー内のすべてのコンポーネントの情報を、親コンポーネントおよび子コンポーネントの情報を含めて確認できます。このサブコマンドを繰り返し指定することにより、Product Registry 内を対話的にブラウズできます。

データベースコンポーネントはツリー形式で表され、1 つ以上の子コンポーネントを持つ場合もあります。ルート以外のコンポーネントは、親コンポーネントを 1 つ持ちます。このサブコマンドを実行すると、Solaris Product Registry データベース内の指定のコンポーネントの親コンポーネントおよび子コンポーネントが表示されます。

prodreg browse サブコマンドを実行するたびに、Registry 内のコンポーネントとその子コンポーネント、さらにそのコンポーネントからルートに至る親 (祖先) コンポーネントが表示されます。prodreg GUI を使用する場合は、任意のノードを選択し、クリックして展開します。コマンド行インタフェースでも同様の処理が可能です。あるノードの子ノードを順番にブラウズしていくことで、Registry を展開できます。

prodreg browse コマンドを実行して、まず Registry のルートをブラウズします。その後、コンポーネントを順に選択して、ブラウズする範囲を展開していきます。このように対話形式でブラウズする場合はブラウズ番号を指定すると便利ですが、スクリプト内に指定することはできません。ブラウズ番号は、セッションまたは使用するシステムごとに異なります。なぜなら、ブラウズ番号は、特定のシステム上で特定のユーザーによって最初に使用されるときに生成されるものだからです。

browse サブコマンドの形式は次のとおりです。

```
browse [-R alt_root] [-u uuid [-i instance | -p location]]
browse [-R alt_root] -n bnum [-i instance | -p location]
browse [-R alt_root] -m name
browse --help
```

各コンポーネントについて、次の情報が出力されます。

BROWSE # コンポーネントのブラウズ番号です。`prodreg browse` や `info` サブコマンドの引数として使用できます。

+/-/. 子コンポーネントがすべて非表示になっているコンポーネントは、「+」で表されます。1 個以上の子コンポーネントが表示されているコンポーネントは、「-」で表されます。子コンポーネントを持たないコンポーネントは、「.」で表されます。下の世代のコンポーネントほど、フィールド内の記号の位置が右寄りに表示されます。

UUID コンポーネントの一意の識別子です。

コンポーネントのインスタンス番号です。ソフトウェアコンポーネントは複数回インストールできます。ソフトウェアレジストリによって、各コンポーネントに 1 つずつ一意のインスタンスが割り当てられます。

NAME Solaris Product Registry データベース内の各コンポーネントに割り当てられている、各言語対応の名前です。この名前は、Registry 内で一意とは限りません。同じ名前を持つコンポーネントが複数存在している可能性があります。

`browse` サブコマンドで Registry データベースを表示するときは、4 種類のオプションを指定できます。あるコンポーネントに複数のインスタンスがある場合、どのインスタンスの情報が出力されたのか判別できないことがあります。要求は明確にする必要があります。`-u` オプションや `-n` オプションを使って、オペランド instance または location を指定すると、出力する情報を明確にできます。

- オペランド情報の指定がない場合は、レジストリツリーのルートとその子コンポーネントが表示されます。ここを起点にして、対話形式で Registry データベース全体をブラウズできます。
- ブラウズ番号が指定された場合は、該当コンポーネントが表示されます。
- `uuid` が指定された場合は、該当コンポーネントが表示されます。

- 名前が指定された場合は、該当コンポーネントが表示されます。

info

コンポーネントの識別情報を指定して、その属性を表示します。Solaris Product Registry 内のすべてのコンポーネントについて情報を表示できます。

Product Registry 内のコンポーネントは属性を持ちます。これらの属性は、*name* 文字列と単一の *value* 文字列から構成されています。

このサブコマンドは、Solaris Product Registry 内のコンポーネントの属性情報を出力します。コンポーネントの指定方法は、UUID、名前、ブラウズ番号のいずれかを必ず指定するという点を除けば、`browse` サブコマンドの場合と同じです。

このサブコマンドで指定したコンポーネントが複数のインスタンスを持つ場合や、Registry 内に同じ名前のコンポーネントが複数存在する場合は、コンポーネントの指定があいまいになります。このような場合、属性情報ではなく、指定した条件に合致するコンポーネントの一覧が出力されます。

デフォルトでは、コンポーネントの属性が 1 行に 1 つずつ出力されます。まず属性名に続いてコロン (`:`) と空白文字が出力され、RETURN キーを押すと属性値が出力されます。その他、指定できるオプションとして、`-a` および `-d` があります。

`info` サブコマンドの形式は次のとおりです。

```
info --help
info [-R alt_root] -u uuid [-i instance | -p location]
info [-R alt_root] -n bnum [-i instance | -p location]
info [-R alt_root] -m name [-a attr | -d ]
```

help | `--help` | `-?`

ヘルプテキストを表示します。

`help` サブコマンドの形式は次のとおりです。

```
help | --help | -?
```

swing

Java Swing GUI を起動します。Java Swing GUI を使用できない環境では失敗します。

`swing` サブコマンドの形式は次のとおりです。

```
swing  [-R alt_root | --help]
```

version | --version | -V　現在のバージョン文字列を出力します。

version サブコマンドの形式は次のとおりです。

```
version | --version | -V
```

unregister　Registry 内のエントリの登録を解除します。

Solaris Product Registry からコンポーネントを削除します。-u オプションで指定した UUID に対応するコンポーネントは、単一のインスタンスである必要があります。同じ UUID のコンポーネントが複数存在する場合、削除処理は実行されません。代わりに、指定した UUID に一致するインスタンスの一覧が表示されます。この場合は、登録を解除するコンポーネントインスタンスを一意に指定できる -p または -i オプションを使って、サブコマンドを再発行してください。

Registry 内に、登録を解除しようとするコンポーネントに依存するコンポーネントが存在する場合、unregister サブコマンドは失敗します。

ユーザーが Registry に対する書き込み権を持っていない場合、unregister サブコマンドは失敗します。これについては、wsreg_can_access_registry(3WSREG) を参照してください。ユーザーが Solaris Product Registry に登録されたコンポーネントではなくシステムコンポーネントの登録を解除しようとした場合、unregister サブコマンドは失敗します。システムコンポーネントに該当するのは、PKG 属性を持つコンポーネントと、いくつかの特別な Registry ノードです。次にその一部を紹介します。

```
UUID                                    Name
====================================  ============================
root                                    System Registry
a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  Solaris System Software
8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  Unclassified Software
b96ae9a9-1dd1-11b2-a3f2-0800209a5b6b  System Software Localizations
b1c43601-1dd1-11b2-a3f2-0800209a5b6b  Additional System Software
a8dcab4f-1dd1-11b2-a3f2-0800209a5b6b  Software Localizations
```

unregister サブコマンドの -f オプションを使用するときは、これから登録を解除するコンポーネントに依存するコンポーネントがないかどうかを事前に注意深く確認してください。-r オプションは、-f オプション以上に注意を要するオプションです。このオプションを指定してコンポーネントの登録を解除すると、そのコンポーネント

に依存するすべての子コンポーネントおよびソフトウェアコンポーネントの登録が解除されます。依存コンポーネントを確認するには、次のコマンドを使用します (*uuid* には UUID を指定)。

```
prodreg info -u uuid -a "Dependent Components"
```

必須コンポーネントを確認するには、次のコマンドを使用します。

```
prodreg info -u <uuid> -a "Required Components"
```

コンポーネントの名前、UUID、およびインスタンスが出力されます。

`unregister` サブコマンドの形式は次のとおりです。

```
unregister [-R alt_root] [-fr] -u uuid [-p location | -i instance]
unregister --help
```

uninstall

アンインストールプログラムを起動します。

Registry 内のコンポーネントが個別にアンインストーラを備えている場合があります。`-u` オプションでコンポーネントを指定すると、そのアンインストーラが起動します。この指定は、アンインストーラを持たないコンポーネントに対しては無効です。`-u` オプションで指定された UUID を持つコンポーネントが複数存在する (そのコンポーネントのインスタンスが複数インストールされている) 場合、該当するコンポーネントインスタンスの一覧が出力されます。このように `-u` オプションで指定された UUID があいまいな場合は、`-i` または `-p` オプションを指定して、サブコマンドを再発行してください。ただし、これらからアンインストールするコンポーネントに依存するコンポーネントが存在する場合、コマンドは失敗します。

`-x` オプションを指定してアンインストーラを起動することもできます。この場合、依存コンポーネントを持つコンポーネントをアンインストールするかどうかの確認は行われません。

ユーザーが Registry に対する書き込み権を持っていない場合、`uninstall` コマンドは実行されません。これについては、`wsreg_can_access_registry`(3WSREG) を参照してください。

`uninstall` コマンドの形式は次のとおりです。

```
uninstall [-R alt_root] [-f] -u uuid -p location
uninstall [-R alt_root] -i instance[arguments ...]
uninstall --help
```

オプション　awt サブコマンドは次のオプションをサポートします。

`--help`　ヘルプテキストを表示します。ビューアは起動しません。

`-R` *alt_root*　指定された代替ルートを使って、GUI ビューアに表示するデータベースの場所を決定します。

alt_root の仕様については、「オペランド」を参照してください。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

`browse` サブコマンドは次のオプションをサポートします。

`-help`　ヘルプテキストを表示します。browse サブコマンドは実行しません。

`-i` *instance*　指定されたコンポーネントインスタンスを出力します。

`-m` *name*　指定された名前のコンポーネントインスタンスを出力します。

`-n` *bnum*　指定されたブラウズ番号のコンポーネントインスタンスを出力します。

`-p` *location*　指定された位置にインストールされているコンポーネントインスタンスを出力します。コンポーネントがインストールされている位置を確認するには、info サブコマンドを使用します。

`-R` *alt_root*　指定された代替ルートを使って、データベースの場所を決定します。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

`-u` *uuid*　指定された UUID のコンポーネントインスタンスを出力します。

`info` サブコマンドは次のオプションをサポートします。

`-a` *attr*　オペランド attr で指定された属性だけを出力します。指定されたコンポーネントの属性をすべて出力するわけではありません。

`-d` `isDamaged` という名前の属性だけを出力します。指定されたコンポーネントの属性をすべて出力するわけではありません。値が `true` に設定されている場合、この属性は Registry 内のコンポーネントを示します。

`--help` ヘルプテキストを出力します。info サブコマンドは実行しません。

`-i` *instance* オペランド instance により、同じ `uuid` またはブラウズ番号を持つ複数のコンポーネントインスタンスを識別します。

`-m` *name* オペランド name により、Registry 内の 1 つ以上のコンポーネントを指定します。

`-n` *bnum* ブラウズ番号 bnum のコンポーネントインスタンスの属性を出力します。該当するインスタンスが複数存在する場合は、`-i` または `-p` オプションを指定して、あいまいさを解決する必要があります。

`-p` *location* インストール先を指定して、同じ UUID またはブラウズ番号を持つ複数のコンポーネントインスタンスを識別します。

`-R` *alt_root* 指定された代替ルートを使って、データベースの場所を決定します。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

`-u` *uuid* 指定された UUID を持つコンポーネントインスタンスの属性を出力します。該当するインスタンスが複数存在する場合は、`-i` または `-p` オプションを指定して、あいまいさを解決する必要があります。

`swing` サブコマンドは次のオプションをサポートします。

`--help` ヘルプテキストを出力します。swing サブコマンドは実行しません。

`-R` *alt_root* 指定された代替ルートを使って、データベースの場所を決定します。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

`uninstall` サブコマンドは次のオプションをサポートします。

`-f` 強制的にアンインストールを行います。指定された UUID を持つコンポーネントインスタンスが複数存在する場合は、該当するすべてのコンポーネントインスタンスをアンインストールします。

`--help` ヘルプテキストを出力します。unregister サブコマンドは実行しません。

`-i` *instance* 指定された UUID のあいまいさを解決します。

`-p` *location* 指定された UUID のあいまいさを解決します。*location* には、ソフトウェアコンポーネントがインストールされている位置を指定します。

`-R` *alt_root* 指定された代替ルートを使って、データベースの場所を決定します。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

`-u` *uuid* 指定された UUID を持つコンポーネントの登録を解除します。同じ UUID を持つコンポーネントが複数回インストールされている場合は、`-i` または `-p` オプションを使って、指定のあいまいさを解決します。

`unregister` サブコマンドは次のオプションをサポートします。

`-f` 強制的に登録を解除します。指定されたコンポーネントに依存するコンポーネントが存在する場合も登録を解除します。

`--help` ヘルプテキストを出力します。unregister サブコマンドは実行しません。

`-i` *instance* 指定された UUID のあいまいさを解決します。

`-p` *location* 指定された UUID のあいまいさを解決します。*location* には、ソフトウェアコンポーネントがインストールされている位置を指定します。

`-r` 指定されたコンポーネントの子コンポーネントと依存コンポーネントを含めて、再帰的にコンポーネントの登録を解除します。

`-R` *alt_root* 指定された代替ルートを使って、データベースの場所を決定します。

注 – いかなる非大域ゾーンのルートファイルシステムも `-R` で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。`zones`(5) のマニュアルページを参照してください。

`-u` *uuid* 指定された UUID を持つコンポーネントの登録を解除します。同じ UUID を持つコンポーネントが複数回インストールされている場合は、`-i` または `-p` オプションを使って、指定のあいまいさを解決します。

オペランド　次のオペランドを指定できます。

alt_root　代替ルートを示すファイルのパス名です。Solaris Product Registry データベースは、代替ルートを基点としています。この位置を基点とするデータベースが存在しない場合は、自動的に作成されます。

注 – いかなる非大域ゾーンのルートファイルシステムも *alt_root* で参照しないでください。この操作によって、大域ゾーンのファイルシステムを損傷したり、大域ゾーンのセキュリティーを損ねたり、非大域ゾーンのファイルシステムを損傷したりする可能性があります。zones(5) のマニュアルページを参照してください。

attr　属性の名前です。このオペランドは、`info` サブコマンド専用です。*attr* がコンポーネントに関連している場合、属性名と値が表示されます。

bnum　ブラウズ番号です。

Solaris Product Registry の各コンポーネントには、ブラウズ番号が割り当てられます。この番号は、対話形式でブラウズを行うユーザーのために生成されます。ただし、この番号は、システムを再起動または再インストールすると変更されます。この番号を `browse` または `info` サブコマンドの補助以外の目的で使用または保存することは避けてください。ブラウズ番号は、`prodreg browse` サブコマンドを実行すると出力されます。これ以外の値を、`browse` または `info` サブコマンドの入力値として指定することはできません。

instance　ソフトウェアは、複数の場所にインストールできます。Solaris Product Registry は、それぞれに一意のインスタンス番号を割り当てます。Registry 内の各コンポーネントのインスタンス番号を表示するには、browse サブコマンドを使用します。このオペランドは、インストール済みソフトウェアの (おそらく同一ではない) 複数のコピーが存在する場合、これらを識別する目的で使用します。

location　ファイルシステム内の特定のファイルまたはディレクトリのパスです。登録されたソフトウェアのインストール先を示します。たとえば、ソフトウェアのインストール先が `/usr/local` からの相対位置である場合、このオペランドの値は `/usr/local` になります。インストール先の情報は、インストーラによって使用されます。インストーラの位置を指定したり、ソフトウェアコンポーネントのインスタンスが複数存在する場合は、指定のあいまいさを解決する目的でも使用されます。

name　Solaris Product Registry 内の各ソフトウェアコンポーネントには、名前が割り当てられています。この名前は、`browse` サブコマンドで確認できます。また、一部のサブコマンドではオペランドとしてソフトウェアの名前を指定できます。ただし、こうした名前が一意であるとは限りません。指定された名前のコンポーネントが複数存在する場合、条件に合致

するコンポーネントの一覧が出力されます。名前が各言語対応になっている場合、すなわち、言語設定によって名前が異なる場合もあります。

uuid Solaris Product Registry 内の各ソフトウェアコンポーネントには、一意の識別子が割り当てられています。この識別子が、Registry データベース内のエントリにアクセスする際のハンドルになります。UUID は、インストールされているコンポーネントインスタンスの数やコンポーネントの各言語対応の名前に関係なく、そのコンポーネントに対応しています。

使用例

例 1 prodreg コマンドによるブラウズ

ブラウズを実行するには、prodreg の browse サブコマンドを使用します。この要求を繰り返し実行することにより、GUI を使ってコンポーネントおよびそれに含まれるコレクションを展開していく場合と同様に、ツリーの内容を確認することができます。ブラウズ番号を使ったブラウズは、こうした再帰的ブラウズ処理以外では実行しないでください。ブラウズ番号は、ブラウズ操作の結果として生成されるものです。

引数なしで browse サブコマンドを実行すると、Registry のトップからブラウズすることになります。出力内容は、そのシステムにインストールされているソフトウェアによって異なります。

```
$ prodreg browse
BROWSE # +/-/.  UUID                                 #  NAME
======== =====  ==================================== =  ============
1        -      root                                 1  System
                                                       Registry
2         +     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b 1  Solaris 10
                                                       System
                                                       Software
3         +     8f64eabf-1dd2-11b2-a3f1-0800209a5b6b 1  Unclassified
                                                       Software
```

ルートコンポーネントとその子コンポーネントのブラウズ番号、UUID、インスタンス番号、および名前が出力されます。コンポーネントの祖先 (ルートに至る親コンポーネント) も表示されます。「+/-/.」列は、ツリー内のコンポーネントが展開済みの親コンポーネントであるか (-)、子コンポーネントを持つ子コンポーネントであるか (+)、子コンポーネントを持たないコンポーネントであるか (.) を示します。

例 2 ツリー内のコンポーネントの情報を要求

UUID、名前、およびブラウズ番号の各フィールドを使って、ツリー内のコンポーネントの情報を要求できます。次の例では、UUID を指定してコンポーネントをブラウズします。

例 2 ツリー内のコンポーネントの情報を要求 *(続き)*

```
$ prodreg browse -u a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b
BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ===========
1         -      root                                  1  System
                                                          Registry
2          -     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                          System
                                                          Software
4           +    b96ae9a9-1dd1-11b2-a3f2-0800209a5b6b  1  System
                                                          Software
                                                          Localizations
5           +    SUNWCall                              1  Entire
                                                          Distribution
```

例 3 名前を指定してノードをブラウズ

次の例では、名前を指定してノードをブラウズします。

```
$ prodreg browse -m "System Software Localizations"
BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ===========
1         -      root                                  1  System
                                                          Registry
2          -     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                          System
                                                          Software
4           -    b96ae9a9-1dd1-11b2-a3f2-0800209a5b6b  1  System
                                                          Software
                                                          Localizations
316          .   SUNWceuow                             1  Central
                                                          Europe OW
                                                          Support
317          .   SUNWcsfw                              1  Simplified
                                                          Chinese
                                                          freeware
                                                          message
318          .   SUNWceuox                             1  Central
                                                          Europe
                                                          64-bit OS
                                                          Support
```

例 4 再帰的ブラウズ

追加出力は省略されています。再帰的ブラウズには、ブラウズ番号を使用すると便利です。ブラウズ番号は、prodreg コマンドを実行するシステム、コマンドを実行するユーザー、およびコマンドを実行するログインセッションによって異なるため、保存できません。

```
$ prodreg browse -n 3

BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ===========
1         -      root                                  1  System
                                                          Registry
2          -     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                          System
                                                          Software
5           -    SUNWCall                              1  Entire
                                                          Software
                                                          Distribution
6            .   SUNWrsmo                             1  RSMPI
                                                          Operations
                                                          Registration
                                                          Module
7            +   SUNWCjvx                             1  JavaVM
                                                          (64-bit)
8            .   SUNWrsmx                             1  Remote
                                                          Shared
                                                          Memory
                                                          (64-bit)
9            +   SUNWCacc                             1  System
                                                          Accounting
```

例 5 あいまいな値によるブラウズ

要求された値があいまいな場合、条件に合致するインスタンスの一覧が表示されます。次の例では、同じ名前を持つ異なったソフトウェアコンポーネントが 2 つ存在します。

```
$ ./prodreg browse -m JavaVM
The request failed because multiple components correspond to the
criteria given. Use the list of possible components given below,
select one and try again.

BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ===========
12        .      org.spybeam.javavm                    1  JavaVM
51        .      SUNWCjv                               1  JavaVM
```

例 5 あいまいな値によるブラウズ (続き)

次のいずれかの要求を再発行してください。

```
$ prodreg browse -u SUNWCjv
```

または

```
$ prodreg browse -u org.spybeam.javavm
```

例 6 複数回インストールされたソフトウェアのブラウズ

特定のソフトウェアコンポーネントが複数回インストールされている場合も、要求があいまいになります。次の例では、Example ソフトウェアが 3 回インストールされています。

```
$ prodreg browse -m Example
The request failed because multiple components correspond to the
criteria given. Use the list of possible components given below,
select one and try again.
BROWSE #  +/-/.  UUID                                  #  NAME
========  =====  ====================================  =  ===========
7         .      org.spybeam.example                   2  Example
7         .      org.spybeam.example                   3  Example
7         .      org.spybeam.example                   1  Example
The component requested could not be found.
```

例 7 特定のインスタンスによるブラウズ

特定のインスタンスを指定して要求を繰り返し実行することにより、あいまいさを解決できます。-p オプションでインストール先を指定する方法もあります。次の例では、Example ソフトウェアの 1 番目のインスタンスをブラウズします。

```
$ prodreg browse -u org.spybeam.example -i 1
```

例 8 info サブコマンドの使用

コンポーネントのインストール先やその他の属性情報を取得したい場合は、info サブコマンドを使用します。このサブコマンドにも、あいまいさを解決するオプションを指定できます。実行すると、コンポーネントのすべての属性が 1 行に 1 つずつ出力されます。

```
$ prodreg info -m Example
The request failed because multiple components correspond to the
criteria given. Use the list of possible components given below,
select one and try again.
BROWSE # +/-/. UUID                                  # NAME
```

例 8 infoサブコマンドの使用 (続き)

```
======== ===== ================================= = ===========
7        .     org.spybeam.example               2 Example
7        .     org.spybeam.example               3 Example
7        .     org.spybeam.example               1 Example
The component requested could not be found.
```

次のコマンドでは、Exampleコンポーネントのインスタンス 1 の情報がすべて出力されます (出力結果は省略)。

```
$ prodreg info -u org.spybeam.example -i 1
```

例 9 インストール先の情報を取得

コンポーネントのインストール先やその他の属性情報を取得したい場合は、infoサブコマンドを使用します。browseサブコマンドの場合と同様、infoサブコマンドにも、あいまいさを解決するオプションを指定できます。infoサブコマンドを実行すると、コンポーネントのすべての属性が 1 行に 1 つずつ出力されます。単一の属性を要求することもできます。

次のコマンドでは、インストール先の属性値が出力されます。

```
$ prodreg info -n 23 -a Location
```

例 10 損傷を受けたソフトウェアの特定および登録解除

所定のアンインストーラを使用せずにインストール済みのソフトウェアを削除すると、Registry内のソフトウェアが損傷を受ける可能性があります。コンポーネントが損傷を受けた場合、すでに削除されたソフトウェアがまだインストールされているものと認識されます。所定のアンインストーラを実行せずにファイルやパッケージを直接削除すると、同様にコンポーネントが損傷を受けることがあります。通常、次の規則に従う必要があります。インストールプログラムを使ってインストールしたソフトウェアは、所定のアンインストールプログラムを使ってアンインストールする必要があります。

次の例では、損傷を受けたソフトウェアコンポーネントを特定し、再インストールできる状態に修復します。

Examplesoftをブラウズすると、次の内容が出力されます。

```
$ prodreg browse -m Examplesoft
BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ============
1         -      root                                  1  System
                                                        Registry
```

例 10 損傷を受けたソフトウェアの特定および登録解除 (続き)

```
2          +     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                          System
                                                          Software
3          +     8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  1  Unclassified
                                                          Software
4          -     95842091-725a-8501-ef29-0472985982be  1  ExampleSoft
233         .    90209809-9785-b89e-c821-0472985982be  1  Example Doc
234         .    EXSOzzt                               1
235         .    EXSOblob                              1  Example Data
```

Examplesoft の子コンポーネント、EXSOzzt に注目してください。登録済みであるにもかかわらず、ソフトウェアのパッケージコンポーネント名が表示されていません。この場合、Examplesoft ソフトウェアが損傷を受けている可能性があります。次のコマンドで損傷の有無を確認します。

```
$ prodreg info -u 95842091-725a-8501-ef29-0472985982be \
     -i 1 -d
isDamaged=TRUE
```

isDamaged が TRUE なので、Examplesoft の一部が損傷を受けています。次のコマンドで、Examplesoft を構成するパッケージを一覧表示します。

```
$ prodreg info \
     -u 95842091-725a-8501-ef29-0472985982be\
     -i 1 -a PKGS pkgs:
EXSOzzt EXSOblob
```

pkginfo コマンドで、EXSO がインストールされているかどうかを確認します。

```
$ pkginfo EXSOzzt
ERROR: information for "EXSOzzt" was not found
$ pkginfo EXSOblob
application EXSOblob       Example Data
```

一連のコマンドの実行結果から、EXSOzzt パッケージが、おそらく pkgrm コマンドによって削除されていることがわかります。この場合、Examplesoft ソフトウェアは正常に機能しません。ソフトウェアを修復するには、Examplesoft に登録されているアンインストーラを実行する必要があります。アンインストーラは、ソフトウェアの登録を解除し、pkgrm コマンドを実行します。この操作には root ユーザーのアクセス権が必要です。したがって、通常、アンインストーラの実行にも、root ユーザーのアクセス権が必要になります。

```
# prodreg uninstall -u 95842091-725a-8501-ef29-0472985982be -i 1
The install program requested could not be found.
```

例 10 損傷を受けたソフトウェアの特定および登録解除 *(続き)*

アンインストールプログラムにアクセスして、ソフトウェアをアンインストールすることができませんでした。アンインストールプログラムが手動で削除されている可能性があります。そこで、uninstallprogram 属性を要求して、アンインストーラの位置を確認します。

```
$ prodreg info -m ExampleSoft -a uninstallprogram
uninstallprogram: /usr/bin/java -mx64m -classpath
/var/sadm/prod/org.example.ExampleSoft/987573587 uninstall_ExampleSoft
```

登録された位置にアンインストーラがあるかどうかを確認します。

```
# ls /var/sadm/prod/org.example.ExampleSoft/987573587
/var/sadm/prod/org.example.ExampleSoft/987573587:
No such file or directory
```

所定の位置にアンインストーラがない場合、2 通りの対処法があります。1 つは、バックアップストレージからアンインストーラを読み込んで手動で実行する方法です。Registry に格納されているコマンド行を使用してください。

```
# /usr/bin/java -mmx64m -classpath \
     /var/sadm/prod/org.example.ExampleSoft/987573587 \
     uninstall_ExampleSoft
```

バックアップストレージにない場合は、手動でソフトウェアの登録を解除します。

```
# prodreg unregister -u 95842091-725a-8501-ef29-0472985982be -i 1
```

このコマンドでは、EXSOblob パッケージは削除されません。手動で行う必要があります。

```
# pkgrm EXSOblob
```

例 11 複数のコンポーネントの削除

コンポーネント A は子コンポーネント B および C を持ち、子コンポーネント C は子コンポーネント D および E を持っているとします。この例では、これらのコンポーネントを一括削除します。この処理は、コンポーネントの階層全体を再インストールする必要があるときに、アンインストーラが見つからない、あるいは実行できない場合に便利です。

```
$ prodreg browse -u UUID-of-C
BROWSE #  +/-/.  UUID                                  #  NAME
========  =====  ====================================  =  ============
1         -      root                                  1  System
                                                        Registry
2          +     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                        System
```

例 11 複数のコンポーネントの削除 *(続き)*

```
                                                        Software
3          +     8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  1  Unclassified
                                                        Software
1423       -     UUID-of-A                             1  Example A
1436        .    UUID-of-B                             1  Example B
1437        -    UUID-of-C                             1  Example C
1462         .   UUID-of-D                             1  Example D
1463         .   UUID-of-E                             1  Example E

# prodreg uninstall -u UUID-of-A -i 1
```

uninstall サブコマンドが失敗することもあります。たとえば、Java クラスが削除されている、ユーザーのアクセス権が十分でない、システムに Java ソフトウェアがインストールされていない、などの条件下で失敗します。この問題には、再帰的登録解除のコマンドで対処できます。しかし、このコマンドは、対象コンポーネントの子コンポーネントおよび対象コンポーネントに依存するコンポーネントの登録をすべて解除するため、使用に際してはきわめて慎重に臨む必要があります。誤って必要なコンポーネントの登録を解除してしまわないように、対象コンポーネントの全情報をあらかじめ確認しておくことをお勧めします。次の例では、UUID-of-A を指定してコンポーネントの情報を表示します。

```
$ prodreg info -u UUID-of-A
Title: Example A Software
Version: 5.8.0.2001.11.02
Location: /usr
Vendor: Example Vendor
uninstallprogram: /usr/bin/java -mx64m -classpath
/var/sadm/prod/org.example.ExampleA/90820965 uninstall_ExampleA
vendorurl: http://www.example.org
description: Example A Software has many uses
Supported Languages: en

Child Components:
Name                        UUID                                  #
--------------------------  ------------------------------------  -
Example B                   UUID-of-B                             1
Example C                   UUID-of-C                             1

Required Components:
Name                        UUID                                  #
--------------------------  ------------------------------------  -
Example B                   UUID-of-B                             1
Example C                   UUID-of-C                             1
```

例 11　複数のコンポーネントの削除　　(続き)

「Dependent Components」フィールドが表示されていないので、Example A に依存するソフトウェアは存在しません。念のため、UUID-of-B と UUID-of-C の依存コンポーネントおよび子コンポーネント、UUID-of-B または UUID-of-C の子コンポーネントに依存するコンポーネントをすべて確認しておきます。

ブラウズツリーを表示して UUID-of-A のすべての子孫を確認できたら、今度は、Example A のすべての子孫の依存コンポーネント属性を確認します。

```
$ prodreg info -u UUID-of-B -i 1 -a "Dependent Components"
Dependent Components:
Name                         UUID                                  #
---------------------------  ------------------------------------  -
Example A                    UUID-of-A                             1

$ prodreg info -u UUID-of-C -i 1 -a "Dependent Components"
Dependent Components:
Name                         UUID                                  #
---------------------------  ------------------------------------  -
Example A                    UUID-of-A                             1

$ prodreg info -u UUID-of-D -i 1 -a "Dependent Components"
Dependent Components:
Name                         UUID                                  #
---------------------------  ------------------------------------  -
Example C                    UUID-of-C                             1

$ prodreg info -u UUID-of-E -i 1 -a "Dependent Components"
Dependent Components:
Name                         UUID                                  #
---------------------------  ------------------------------------  -
Example C                    UUID-of-C                             1
```

Example A の再帰的登録解除を実行しても、Example A とその子孫の登録以外は解除されません。

```
# prodreg unregister -r -u UUID-of-A -i 1
```

例 12　損傷を受けたコンポーネントの再インストール

この例では、依存するソフトウェアを持つコンポーネント、Software ZZZ が存在します。Software ZZZ は損傷を受けたため、再インストールする必要があります。再インストールするには、Software ZZZ の登録を解除しなければなりません。

この場合、まず Software ZZZ の依存コンポーネントを確認します。

例 12 損傷を受けたコンポーネントの再インストール (続き)

```
$ prodreg info -m "Software ZZZ" -a "Dependent Components"
Dependent Components:
Name                          UUID                                  #
---------------------------  ------------------------------------  -
Software Foobar               d9723500-9823-1432-810c-0100e09832ff  1
```

通常、Software ZZZ の登録を解除するには、依存コンポーネントである Software Foobar をアンインストールする必要があります。しかし、Software Foobar の再インストールが不可能または不適切であることがわかっているため、Software ZZZ の再帰的登録解除は実行できません。再帰的登録解除を実行すると、Software Foobar の登録まで解除されてしまいます。この場合は、Software ZZZ の強制登録解除を実行します。Software ZZZ の UUID は、90843fb1-9874-3a20-9b88-984b32098432 です。

```
# prodreg unregister -f -u 90843fb1-9874-3a20-9b88-984b32098432 -i 1
```

続いて、Software ZZZ を再インストールします。

```
# /usr/bin/java -cp /usr/installers/org.example.softwarezzz
```

使用上の留意点

ソフトウェアを手動で削除したり、pkgrm(1M) を使用して直接削除したりすると、Registry の情報が一致しなくなります。Registry の損傷を防ぐため、インストールプログラムを使ってインストールしたソフトウェアはアンインストールプログラムを使ってアンインストールしてください。

環境

次の環境変数は、prodreg の実行に影響を及ぼします。

PKG_INSTALL_ROOT　システムの PKG_INSTALL_ROOT パスとして使用するディレクトリの完全パス名を定義します。この環境変数が定義されている場合、すべての製品およびパッケージの情報ファイルは、最初に PKG_INSTALL_ROOT パスで検索されます。この環境変数が定義されていない場合、デフォルトのシステムパス / が使用されます。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

>0　エラーが発生した。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWwsr2, SUNWwsrv
インタフェースの安定性	開発中

関連項目

pkgadd(1M), pkgrm(1M), wsreg_can_access_registry(3WSREG), libwsreg(3LIB), live_upgrade(5), attributes(5)

『Application Packaging Developer's Guide』

注意事項

prodreg の GUI やコマンド行インタフェースには、Solaris Product Registry とパッケージデータベースが両方とも表示されます。どちらも Registry 内のコンポーネントのように見えますが、一部、登録解除やアンインストールができないものがあります。所定のアンインストーラを持たないパッケージは、prodreg uninstall コマンドでアンインストールできません。Solaris パッケージは、prodreg unregister コマンドで登録解除できません。GUI または CLI の prodreg ビューアに表示されないパッケージは、pkgrm(1M) コマンドで削除する必要があります。

ソフトウェアを削除するときは、pkgrm(1M) を使って個々のパッケージを削除するのではなく、付属のアンインストールソフトウェアを使用することをお勧めします。アンインストールソフトウェアを使用すれば、Registry 内の情報の登録解除からパッケージの削除まで、ソフトウェアの全リソースの削除を総合的に行うことができます。

prodreg uninstall コマンドは、外部プログラムを起動します。Product Registry の代替ルートを指定する際は、こうした外部プログラムのコマンド行の規約に従う必要があります。インストールプログラムが prodreg と同じ環境で実行されるように、PKG_INSTALL_ROOT 環境変数を使用する方法もあります。アンインストールプログラムは、通常、Java のインストールを必要とする Java クラスです。Java ソフトウェアが Solaris のディストリビューションから削除された場合、または見つからない場合、Java ベースのアンインストーラは実行できません。

root ユーザーのアクセス権がないと実行できないサブコマンドは、prodreg unregister と prodreg uninstall の 2 つだけです。なぜなら、登録の解除には Product Registry の変更が伴い、アンインストールにはパッケージの削除が伴うからです。これ以外のコマンドは Registry の内容を読み取るだけなので、root ユーザーのアクセス権がなくても実行できます。インストーラが root ユーザーのアクセス権を必要とする pkgadd(1M) や pkgrm(1M) などのコマンドを実行する場合と同様に、prodreg uninstall コマンドの実行にも root ユーザーのアクセス権が必要になります。

コンポーネントの属性については、さまざまなマニュアルに説明が記載されていますが、詳しくは『Application Packaging Developer's Guide』を参照してください。Solaris Product Registry 自体の属性については、次の用語集を参照してください。

依存コンポーネント	そのコンポーネントに依存するコンポーネント。
位置	ソフトウェアのインストール先からの相対位置。
pkgs	コンポーネントのパッケージ。これらのパッケージは、コンポーネントの登録後に pkgadd コマンドで追加され、コンポーネントの登録解除前に pkgrm コマンドで削除されます。

必須コンポーネント	そのコンポーネントが依存するコンポーネント。
ソース	インストール媒体。
サポート言語	登録済みタイトルがあるロケール。
タイトル	`prodreg browse` コマンドで指定される名前。シェルの実行ロケールの言語に対応させることができます。
固有名	以前のバージョンの Solaris Product Registry で使用されていた名前。通常、Registry 内のコンポーネントのパッケージ名に設定されます。
ベンダー	コンポーネントの製造元ベンダー。
バージョン	コンポーネントのバージョン文字列。

Registry 内に、実際のシステムにはインストールされていないソフトウェアコンポーネントが存在する場合があります。こうしたコンポーネントはいくつかの方法で検出できます。一番簡単なのは、`info` サブコマンドを使用して、コンポーネントが損傷を受けているかどうかを調べる方法です。`info` サブコマンドで、ソフトウェアがインストールされた場所を調べ、現在もその場所にあるかどうかを確認する方法もあります。

名前

prstat – アクティブプロセスの統計を報告する

形式

```
prstat  [-acJLmRtTv] [-C psrsetlist] [-j projlist]
 [-k tasklist] [-n ntop[,nbottom]] [-p pidlist] [-P cpulist]
 [-s key | -S key] [-u euidlist] [-U uidlist] [-z zoneidlist]
 [-Z] [interval [count]]
```

機能説明

prstat ユーティリティーは、システム上のすべてのアクティブプロセスを繰り返し検査し、選択された出力モードと並び替え順に基づいて統計を報告します。prstat には、指定した PID、UID、ゾーン ID、CPU ID、プロセッサセット ID に一致するプロセスのみを検査するオプションがあります。

-j、-k、-C、-p、-P、-u、-U、および -z オプションには、引数としてリストを指定できます。リストの項目は、コンマで区切られるか、あるいは引用符で囲まれてコンマか空白で区切られます。

オプションを指定しない場合、prstat はすべてのプロセスを検査し、CPU 使用状況の順に統計を報告します。

オプション

サポートしているオプションは、以下のとおりです。

-a
: プロセスとユーザーに関する情報を報告します。このモードでは、prstat はプロセスとユーザーに関する別個のレポートを同時に表示します。

-c
: 前のレポートに重ねるのではなく、レポートの下に新しいレポートを表示します。

-C *psrsetlist*
: 指定したリスト内のプロセッサセットにバインドされたプロセスまたは lwp のみを報告します。各プロセッサセットは、psrset(1M) によって報告される整数で識別されます。表示される平均負荷率は、指定したプロセッサセットの平均負荷率の合計です (pset_getloadavg(3C) を参照)。-L オプションが使用されていない場合でも、指定したリスト内のプロセッサセットにバインドされた 1 つ以上のLWP を持つプロセッサが報告されます。

-j *projlist*
: 指定したリスト内にプロジェクト ID があるプロセスまたは lwp だけを報告します。各プロジェクト ID はプロジェクト名または数値のプロジェクト ID のどちらでも指定できます。project(4) を参照してください。

-J
: プロセスとプロジェクトに関する情報を報告します。このモードでは、prstat はプロセスとプロジェクトに関する別個のレポートを同時に表示します。

-k *tasklist*
: *tasklist* 内にタスク ID があるプロセスまたは lwp だけを報告します。

`-L` 各軽量プロセス (LWP) の統計を報告します。デフォルトでは、`prstat` は各プロセスの LWP の数だけを報告します。

`-m` マイクロステートプロセスアカウンティング情報を報告します。`-v` モードで表示されるすべてのフィールドに加え、このモードには、プロセスがシステムトラップ、テキストページフォルト、データページフォルトの処理に費やした時間、ユーザーロックの待機および CPU の待機 (待ち時間) に費やした時間のパーセンテージも含まれます。

`-n` *ntop*[,*nbottom*] 出力の行数を制限します。*ntop* 引数はプロセスまたは `lwp` 統計が報告される行数を決定し、*nbottom* 引数は、`-a`、`-t`、`-T`、または `-J` オプションが指定されている場合に、ユーザー、タスク、またはプロジェクト統計が報告される行数を決定します。デフォルトでは、`prstat` はウィンドウまたはターミナルに合わせた出力の行数を表示します。`-c` オプションを指定した場合、またはファイルへの出力を指定した場合、`ntop` および `nbottom` のデフォルト値は `15` および `5` です。

`-p` *pidlist* 指定したリスト内にプロセス ID があるプロセスだけを報告します。

`-P` *cpulist* 指定したリスト内の CPU で最後に実行されたプロセスまたは lwp だけを報告します。各 CPU は、`psrinfo`(1M) で報告される整数によって識別されます。

`-R` `prstat` をリアルタイムスケジューリングクラスに配置します。このオプションが使用される場合、`prstat` は、タイムシェアリングプロセスおよび対話型プロセスよりも優先されます。このオプションはスーパーユーザーのみが使用できます。

`-s` *key* 出力行 (つまり、プロセス、`lwp`、またはユーザー) を *key* (キー) の降順にソートします。引数として使用できるのは 1 つの *key* (キー) のみです。

使用可能なキー値は 5 つあります。

cpu プロセスの CPU 使用状況順にソートします。デフォルト値です。

pri プロセス優先順位順に並び替えます。

rss 常駐セットサイズ順に並び替えます。

size プロセスイメージのサイズ順に並び替えます。

time プロセス実行時間順に並び替えます。

-S *key*	出力行を *key* (キー) の昇順にソートします。使用可能な *key* (キー) 値は、-s オプションと同じです。-s を参照してください。
-t	各ユーザーの全体的な使用状況の概要を報告します。概要には、ユーザーが所有するプロセスまたは LWP の総数、合計プロセスイメージサイズ、合計常駐セットサイズ、合計 CPU 時間、最近の CPU 時間とシステムメモリーのパーセンテージが含まれます。
-T	プロセスとタスクに関する情報を報告します。このモードでは、prstat はプロセスとタスクに関する別個のレポートを同時に表示します。
-u *euidlist*	指定したリスト内に実効ユーザー ID があるプロセスだけを報告します。各ユーザー ID はログイン名または数値のユーザー ID のどちらでも指定できます。
-U *uidlist*	指定したリスト内に実ユーザー ID があるプロセスだけを報告します。各ユーザー ID はログイン名または数値のユーザー ID のどちらでも指定できます。
-v	プロセス使用状況を詳細に報告します。この出力形式には、プロセスがユーザーモード、システムモード、および休止状態で費やした時間のパーセンテージが含まれます。自発的および強制的なコンテキスト切り替えの数、システムコール、および受信したシグナルの数も含まれます。報告されない統計は、- 記号でマークされます。
-z *zoneidlist*	指定したリスト内にゾーン ID があるプロセスまたは LWP だけを報告します。各ゾーン ID は、ゾーン名または数値のゾーン ID のどちらでも指定できます。zones(5) を参照してください。
-Z	プロセスとゾーンに関する情報を報告します。このモードでは、prstat はプロセスとゾーンに関する別個のレポートを同時に表示します。

出力

prstat レポートの列ヘッダーとその意味を次のリストで説明します。

PID	プロセスのプロセス ID。
USERNAME	実ユーザー (ログイン) 名または実ユーザー ID。
SIZE	すべてのマップ済みのファイルおよびデバイスを含む、プロセスの合計仮想メモリーのキロバイト (K)、メガバイト (M)、またはギガバイト (G) 単位のサイズ。
RSS	キロバイト (K)、メガバイト (M)、またはギガバイト (G) 単位のプロセスの常駐セットサイズ (RSS)。RSS 値は、proc(4) によって提供さ

れる見積りですが、これは実際の常駐セットサイズよりも少なく見積もる場合があります。容量計画のためにより正確な使用情報を取得するには、代わりに、pmap(1) に対して -x オプションを使用することをお勧めします。

STATE　プロセスの状態

- cpu*N*　プロセスは、CPU *N* 上で実行されています。
- sleep　休止状態。プロセスは、イベントが完了するのを待っている
- run　実行可能状態。プロセスは、実行待ち行列上にあります。
- zombie　ゾンビ状態。プロセスは終了していて、親プロセスは待っていない
- stop　プロセスは停止されています。

PRI　プロセスの優先順位。 数値が大きいほど優先順位が高くなります。

NICE　優先順位の計算に使用される nice 値。特定のスケージューリングクラスのプロセスのみが、nice 値を持ちます。

TIME　プロセスの累積実行時間

CPU　プロセスによって使用された最近の CPU 時間のパーセンテージ。非大域ゾーンで実行中で、プール機能がアクティブな場合、パーセンテージは、ゾーンがバインドされたプールによって使用中のプロセッサセット内にあるプロセッサのパーセンテージになります。

PROCESS　プロセスの名前 (実行されたファイルの名前)。

LWPID　情報が出力されている lwp の lwp ID。

NLWP　プロセス内の lwp の数。

一部のオプションを使用すると、前述の多数の列ヘッダーに加えて、次の列ヘッダーが表示されます。

NPROC　指定したコレクションにあるプロセスの数。

MEMORY　プロセスの指定したコレクションによって使用されたメモリーのパーセンテージ。

-v または -m オプションを指定すると、次の列が表示されます

USR　プロセスがユーザーモードで費やした時間のパーセンテージ。

SYS　プロセスがシステムモードで費やした時間のパーセンテージ。

TRP　プロセスがシステムトラップの処理で費やした時間のパーセンテージ。

TFL　プロセスがテキストページフォルトの処理で費やした時間のパーセンテージ。

DFL　プロセスがデータページフォルトの処理で費やした時間のパーセンテージ。

LCK　プロセスがユーザーロックの待機で費やした時間のパーセンテージ。

SLP　プロセスが休止状態で費やした時間のパーセンテージ。

LAT　プロセスが CPU の待機で費やした時間のパーセンテージ。

VCX　コンテキストの自発的な切り替え数。

ICX　コンテキストの強制的な切り替え数。

SCL　システムコールの数。

SIG　受信されたシグナルの数。

-L オプションの下に、プロセス内の lwp ごとに 1 行が表示され、一部のレポートフィールドにはプロセスではなく、lwp の値が表示されます。

オペランド

次のオペランドがサポートされています。

count　統計を繰り返す回数を指定します。デフォルトでは、prstat は終了シグナルが受信されるまで統計を報告します。

interval　抽出間隔を秒数で指定します。デフォルトの間隔は 5 秒です。

使用例

例 1　もっともアクティブな 5 つのスーパーユーザープロセスの報告

次のコマンドは、CPU1 および CPU2 上で実行中のもっともアクティブな 5 つのスーパーユーザープロセスを報告します。

```
example% prstat -u root -n 5 -P 1,2 1 1

PID   USERNAME  SIZE   RSS STATE  PRI  NICE      TIME  CPU PROCESS/LWP
 306   root     3024K 1448K sleep   58    0   0:00.00 0.3% sendmail/1
 102   root     1600K  592K sleep   59    0   0:00.00 0.1% in.rdisc/1
 250   root     1000K  552K sleep   58    0   0:00.00 0.0% utmpd/1
 288   root     1720K 1032K sleep   58    0   0:00.00 0.0% sac/1
   1   root      744K  168K sleep   58    0   0:00.00 0.0% init/1
TOTAL:       25, load averages:  0.05, 0.08, 0.12
```

例 2　プロセス使用状況の詳細情報の表示

次のコマンドは、ユーザー root および john が所有する最小の常駐セットサイズのプロセスに関するプロセス使用状況の詳細情報を表示します。

例 2 プロセス使用状況の詳細情報の表示 *(続き)*

```
example% prstat -S rss -n 5 -vc -u root,john

 PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWP
   1 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 init/1
 102 root     0.0 0.0  -   -   -   -  100  -    0   0   3   0 in.rdisc/1
 250 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 utmpd/1
1185 john     0.0 0.0  -   -   -   -  100  -    0   0   0   0 csh/1
 240 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 powerd/4
 TOTAL:       71, load averages:  0.02, 0.04, 0.08
```

終了ステータス

次の終了ステータスが返されます。

`0` 正常終了。

`1` エラーが発生しました。

属性

次の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

`proc`(1), `psrinfo`(1M), `psrset`(1M), `sar`(1M), `pset_getloadavg`(3C), `proc`(4), `project`(4), `attributes`(5), `zones`(5)

注意事項

`prstat` によって表示されるシステム使用状況のスナップショットは、瞬間的な状況を示すものに過ぎず、表示された時点ではすでに正確でない場合があります。`-m` オプションが指定された場合、`prstat` は各プロセスのマイクロステートアカウンティングをオンに設定しようとします。`prstat` が終了すると、元の状態が復元されます。マイクロステートアカウンティング機能の詳細については、`proc`(4) を参照してください。

プロセスのグループの SIZE および RSS 列に報告される合計メモリーサイズは、共有メモリーセグメントを持つプロセスが使用する実際のメモリーサイズよりも多く見積もる場合があります。

名前　prtconf – システム構成の出力

形式
```
/usr/sbin/prtconf  [-V] |  [-F] |  [-x] |  [-bpv]
 |  [-acDPv] [dev_path]
```

機能説明　prtconf コマンドはシステム構成情報を出力します。この出力には、メモリーの総量やシステム周辺機器の構成 (デバイスツリー形式) が含まれます。

コマンド行にデバイスパスを指定した場合、prtconf は、そのデバイスノードに関する情報だけを表示します。

オプション　次のオプションを指定できます。

-a　コマンド行に指定したデバイスの祖先であるデバイスノードをすべて (つまり、ルートノードまで) 表示します。

-b　プラットフォームを特定するため、ファームウェアのデバイスツリーのルート設定を表示します。これらの設定の内容は「name」、「compatible」、「banner-name」、および「model」です。

-c　コマンド行に指定したデバイスノードをルートとするデバイスサブツリーを表示します。つまり、コマンド行に指定したデバイスノードの子孫であるデバイスノードをすべて表示します。

-D　デバイスツリー内のシステム周辺機器ごとに、周辺機器を管理するために使用されているデバイスドライバの名前を表示します。

-F　SPARC プラットフォームのみのオプション。コンソールフレームバッファが存在する場合、そのデバイスパス名を返します。フレームバッファが存在しない場合、prtconf はゼロ以外の終了ステータスを返します。このフラグは単独で使用する必要があります。prtconf は、コンソールの名前のみ、フレームバッファデバイス、あるいはゼロ以外の終了ステータスを返します。たとえば、SUNW, Ultra-30 上のコンソールフレームバッファが ffb であれば、prtconf は次の値を返します。/SUNW,ffb@1e,0:ffb0。このオプションは、/dev/fb から実際のコンソールデバイスへのシンボリックリンクを作成するときに使用できます。

-p　SPARC プラットフォーム上のファームウェア (PROM) または x86 プラットフォーム上のブートシステムが提供するデバイスツリーから取得した情報を表示します。このオプションを使用して表示されるデバイスツリー情報は初期構成のスナップショットであり、後で行われた構成変更は正確に反映されない場合があります。

-P　疑似デバイスの情報も出力します。デフォルトでは、疑似デバイスの情報は出力されません。

-v　詳細表示モードを指定します。

-V　プラットフォームに固有な PROM (SPARC プラットフォーム) またはブートシステム (x86 プラットフォーム) のバージョン情報を表示します。このフラグ

は単独で使用する必要があります。出力は文字列です。文字列の形式は決まっておらず、プラットフォームに固有です。

`-x` 当該システム上のファームウェアが 64 ビット対応であるかどうかを報告します。既存のプラットフォームの中には、64 ビットカーネルを実行するためにファームウェアをアップグレードする必要があるものもあります。この操作が当該プラットフォームで適用されない場合、つまり、ファームウェアがすでに 64 ビット対応である場合、`prtconf` は何も表示せずに、ゼロの終了ステータスを返して終了します。この操作が当該プラットフォームで適用される場合、つまり、ファームウェアが 64 ビット対応でない場合、`prtconf` は標準出力に説明メッセージを表示して、ゼロ以外の終了ステータスを返して終了します。64 ビットカーネルを実行するためにファームウェアをアップグレードする必要があるかどうかの詳細については、プラットフォームのハードウェアマニュアルを参照してください。

このフラグは他のすべてのフラグを無効にするので、単独で使用する必要があります。

オペランド

次のオペランドを指定できます。

dev_path ターゲットとなるデバイスマイナーノード、デバイスへのパス、または、デバイスリンクへのパスです。これらデバイスノードの構成情報が表示されます。

終了ステータス

次の終了ステータスが返されます。

`0` 正常終了

`0` 以外 `-F` オプションを指定した場合 (SPARC のみ)、ゼロ以外の戻り値は、出力デバイスがフレームバッファではないことを意味します。`-x` オプションを指定した場合、ゼロ以外の戻り値は、ファームウェアが 64 ビット対応ではないことを意味します。上記以外の場合には、ゼロ以外の戻り値はエラーが発生したことを意味します。

属性

以下の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWesu
インタフェースの安定性	不安定

関連項目

fuser(1M), modinfo(1M), `sysdef`(1M), `attributes`(5)

『Sun ハードウェアマニュアル』

SPARC のみ

`openprom`(7D)

注意事項

prtconf コマンドの出力はシステムにインストールされている PROM のバージョンに大きく依存します。したがって、出力は潜在的にあらゆる状況の影響を受けます。

driver not attached というメッセージは、現在、デバイスの当該インスタンスにドライバが接続されていないことを意味します。一般に、ドライバは必要に応じてロードおよびインストールされ、(そして、ハードウェアのインスタンスに接続され)、デバイスが使用されないときはアンインストールとアンロードされます。

x86 プラットフォーム上で prtconf -vp を使用すると、prtconf -v の部分的な情報が表示されます。prtconf -vp の整数設定の値を正しく解釈するには、バイトスワップが必要になる場合があります。

x86 プラットフォーム上で prtconf -vp を使用すると、prtconf -v の部分的な情報が表示されます。prtconf -vp の整数設定の値を正しく解釈するには、バイトスワップが必要になる場合があります。

名前　prtdiag – システムの診断情報の出力

形式　`/usr/sbin/prtdiag [-v] [-l]`

機能説明　`prtdiag` は、`sun4u` および `sun4v` システム上で、システム設定と診断情報を表示します。

診断情報は、システム内で障害の発生した現場交換可能ユニット (FRU) を表示します。

`prtdiag` に必要なインタフェース、出力、およびディレクトリ階層における位置は、まだ確定されていません。将来のリリースで変更される可能性があります。

`prtdiag` は、Sun Enterprise 10000 サーバーで実行された場合は、診断情報と環境状態を表示しません。それらの情報を得るには、システムサービスプロセッサ(SSP)上の`/var/opt/SUNWssp/adm/${SUNW_HOSTNAME}/messages`ファイルを参照してください。

オプション　サポートしているオプションは、次のとおりです。

`-l`　ログの出力。システムで障害またはエラーが発生した場合は、syslogd(1M) のみに対してこの情報を出力します。

`-v`　詳細表示 (Verbose) モードです。最近発生した AC 電源障害の時刻、最近発生した重大なハードウェアエラーの情報、および (必要に応じて) 環境状態を表示します。重大なハードウェアエラーの情報は、FRU を修理する場合や、詳細な診断を行う場合に有用です。

終了ステータス　次の終了値が返されます。

`0`　システム内で何も異常が検出されなかった。

`1`　システム内で異常が検出された。

`2`　メモリー不足などの、内的な `prtdiag` のエラーが発生した。

属性　次の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWkvm
インタフェースの安定性	不安定*

* 出力は不安定 (Unstable) です。

関連項目　`uname`(1), modinfo(1M), prtconf(1M), `psrinfo`(1M), `sysdef`(1M), syslogd(1M), `attributes`(5), `openprom`(7D)

注意事項

すべての Solaris プラットフォーム上で、すべての診断およびシステム情報が得られるわけではないため、これらの情報が `prtdiag` で表示されないことがあります。こうしたプラットフォームでは、システムコントローラを使用してより詳しい情報を得ることができます。

名前

raidctl – RAID ハードウェアユーティリティー

形式

```
raidctl -C "disks" [-r raid_level] [-z capacity] [-s stripe_size] [-f]
     controller

raidctl -d [-f] volume

raidctl -F filename [-f] controller...

raidctl -a {set | unset} -g disk {volume | controller}

raidctl -p "param=value" [-f] volume

raidctl -c [-f] [-r raid_level] disk1 disk2 [disk3...]

raidctl -l -g disk controller

raidctl -l volume

raidctl -l controller...

raidctl [-l]

raidctl -S [volume | controller]

raidctl -S -g disk controller

raidctl -h
```

機能説明

raidctl ユーティリティーはハードウェア RAID 構成ツールであり、RAID ボリュームを作成、削除、または表示する CLI (コマンド行インタフェース) をエンドユーザーに提供することで、さまざまな RAID コントローラをサポートしています。このユーティリティーはまた、ボリュームのプロパティーの設定、ボリュームやコントローラへのホットスペア (HSP) ディスクの割り当て、および RAID コントローラのファームウェア/fcode/BIOS の更新にも使用できます。

raidctl ユーティリティーには、基本ファイルシステムのアクセス権で制御される特権が必要です。特権を持つユーザーだけが RAID システム構成を操作できます。特権を持たないユーザーが raidctl を実行しようとすると、終了ステータス 1 が返されてコマンドは失敗します。

raidctl ユーティリティーは、完全な機能を備えた RAID コントローラを管理するためのコマンド行オプションのセットを定義しています。RAID コントローラによってサポートされている機能が異なることがあるため、特定の RAID コントローラですべてのオプションがサポートされるわけではありません。ユーザーは raidctl を使用することで、特定のコントローラのタイプとファームウェアのバージョンを一覧表示して、サポートされる機能を確認できます。

現在、raidctl は次の RAID コントローラをサポートしています。

- LSI1020、LSI1030、LSI1064、および LSI1068 SCSI HBA。これらは x86 (32/64 ビット) および SPARC プラットフォーム上で、mpt ドライバによって維持管理されます。

オプション　次のオプションを指定できます。

`-C` *"disks"* [`-r` *raid_level*] [`-z` *capacity*] [`-s` *stripe_size*] [`-f`] *controller*
指定したディスクを使用して RAID ボリュームを作成します。

このオプションを使用して RAID ボリュームを作成すると、新しく作成されたボリュームの識別情報が自動的に生成され、raidctl はそれをユーザーに報告します。

このオプションによって指定される引数には、作成されるボリュームの構成に使われる要素が含まれます。要素にはディスクまたはサブボリュームのいずれかを指定できます。ディスクは空白で区切り、サブボリュームはディスクのセットを丸括弧でグループ化します。すべてのディスクは「`C.ID.L`」の形式にしてください (たとえば「`0.1.2`」は、チャネル 0 の物理ディスクであり、ターゲット ID は 1、論理ユニット番号は 2 であることを表す)。この引数は、(`-r` オプションを省略した場合でも) `-r` オプションで指定した RAID レベルと一致している必要があります。つまり、指定できる引数は次のように限定されます。

RAID 0 の場合	少なくとも 2 つのディスク
RAID 1 の場合	2 つのディスクのみ
RAID 1E の場合	少なくとも 3 つのディスク
RAID 5 の場合	少なくとも 3 つのディスク
RAID 10 の場合	少なくとも 2 つのサブボリューム。各サブボリュームは 2 つのディスクで構成されている必要があります
RAID 50 の場合	少なくとも 2 つのサブボリューム。各サブボリュームは 3 つ以上のディスクで構成されている必要があり、各サブボリューム内のディスク容量は同じであるべきです

たとえば、「0.0.0 0.1.0」の形式は、指定された 2 つのディスクが RAID ボリューム (RAID 0 または RAID 1 ボリュームのいずれか) を構成することを意味します。「(0.0.0 0.1.0)(0.2.0 0.3.0)」は、最初の 2 つのディスクと最後の 2 つのディスクが 2 つのサブボリュームを構成し、これら 2 つのサブボリュームが RAID 10 ボリュームを構成することを意味します。その他の例については、「使用例」の節を参照してください。

`-r` オプションは、作成されるボリュームの RAID レベルを指定します。指定できるレベルは 0、1、1E、5、10、50 です。このオプションを省略すると、`raidctl` はデフォルトで RAID 1 ボリュームを作成します。

`-z` オプションは、作成されるボリュームの容量を指定します。単位として、テラバイト、ギガバイト、またはメガバイト (たとえば 2t、10g、20m など) を使用

できます。このオプションを省略すると、raidctl は指定されたディスクで作成できるボリュームの最大容量を計算し、この値を使用してボリュームを作成します。

-s オプションは、作成されるボリュームのストライプサイズを指定します。指定できる値は 512、1k、2k、4k、8k、16k、32k、64k、または 128k です。このオプションを省略すると、raidctl はボリュームに適した値 (たとえば 64k) を選択します。

たとえば LSI1020、LSI1030、LSI1064、または LSI1068 HBA において、場合によっては RAID ボリュームの作成が特定のディスク上のデータ消失を引き起こすことがあり、raidctl はユーザーに対してボリューム作成の確認を要求します。ユーザーへの確認要求なしでボリューム作成を強制させるには、-f オプションを使用します。

controller 引数は、指定したディスクがどの RAID コントローラに属するかを特定するために使用します。-l オプションを使用すれば、コントローラの ID 番号を一覧表示できます。

-d [-f] *volume*
volume として指定された RAID ボリュームを削除します。ボリュームは標準的な形式 (たとえば c0t0d0) で指定します。

ボリュームを削除すると、すべてのデータが失われます。このため、-f オプションを指定しないかぎり、raidctl はボリュームを削除する前にユーザーに対して確認を要求します。

LSI1020、LSI1030、LSI1064、または LSI1068 HBA から RAID 1 ボリュームを削除すると、一次ディスクと二次ディスクが「分離」します。ボリュームが SYNCING 状態にあった場合、一次側にはデータが含まれていますが、二次側には含まれていません。ボリュームの状態が OPTIMAL であった場合、両方のディスクにデータの完全なイメージが含まれています。

-F *filename* [-f] *controller…*
指定したコントローラで動作しているファームウェアを更新します。-f オプションを指定しないかぎり、raidctl ユーティリティーはユーザーに対してこのアクションの確認を要求します。

-a {set | unset} -g *disk* {volume | controller}
ボリュームを指定した場合、-a で指定した値によって、raidctl はディスクをボリューム専用のローカルホットスペアディスクとして設定するか、その設定を解除します。コントローラを指定した場合、raidctl はディスクをグローバルなホットスペアディスクとして設定するか、その設定を解除します。

-p *"param=value"* [-f] *volume*
指定した RAID ボリュームのプロパティー値を変更します。現在、キャッシュ書き込みポリシーのみが (「on」または「off」に) 変更可能です。このため、*param* には文字列「wp」(SET_WR_POLICY) のみを、*value* には「on」または「off」のいずれかのみを指定できます。

RAID ボリュームのプロパティーを変更すると RAID コントローラの内部動作に影響する可能性があるため、-f オプションを指定しないかぎり、raidctl は変更を適用する前にユーザーに対して確認を要求します。

-c [-f] [-r *raid_level*] *disk1 disk2* [*disk3*...]
指定したディスクを使用してボリュームを作成します。これは、似た機能を持つ -C オプションの代替手段です。このオプションは互換性のために残されていますが、LSI1020、LSI1030、LSI1064、および LSI1068 HBA で RAID 0、RAID 1、または RAID 1E ボリュームを作成することしかできません。ほかの HBA では、ユーザーは -C オプションのみを使用できます。

-r オプションは、ターゲットボリュームの RAID レベルを指定するために使用できます。-r オプションを省略すると、raidctl は RAID 1 ボリュームを作成します。

ディスクは Solaris の標準的な形式 (たとえば c0t0d0) で指定する必要があります。

このオプションで RAID 1 ボリュームを作成すると、*disk2* の内容が *disk1* の内容に置き換わります。

ユーザーがこのオプションで RAID ボリュームを作成すると、RAID ボリュームは *disk1* の識別情報を引き受けます。ほかのディスクは見えなくなり、RAID ボリュームは 1 つのディスクとして表示されます。

このオプションによるボリュームの作成は、デフォルトで対話形式で進められます。ボリュームを作成するには、ユーザーはプロンプトに明示的に答える必要があります。ユーザーへの確認要求なしでボリューム作成を強制させるには、-f オプションを使用します。

-l -g *disk controller*
指定したコントローラの、指定したディスクに関する情報を表示します。出力には次の情報が含まれます。

Disk	ディスクを C.ID.L の形式で表示します。
Vendor	ベンダー ID の文字列を表示します。
Product	製品 ID の文字列を表示します。
Capacity	ディスクの合計容量を表示します。
Status	ディスクの現在の状態を表示します。状態は「GOOD」(正常動作中) または「FAILED」(機能していない) のいずれかです。

HSP　ディスクがグローバルホットスペアディスク、ローカルホットスペアディスク、または通常のものとして設定されているかを示します。ローカルホットスペアディスクの場合、このディスクが割り当てられているすべてのボリュームが表示されます。

`-l` *volume*
指定したボリュームに関する情報を表示します。出力には次の情報が含まれます。

Volume　ボリュームを標準的な形式で表示します。

Sub　指定したボリュームが RAID 10 または RAID 50 ボリュームの場合、サブボリュームを表示します。

Disk　指定したボリュームを構成するディスクをすべて表示します。

Stripe Size　ボリュームのストライプサイズを表示します。

Status　指定したボリューム、または指定したボリュームを構成するサブボリューム/ディスクの状態を表示します。ボリュームの場合、状態は「`OPTIMAL`」(正常動作中)、「`DEGRADED`」(機能制限付きで動作中)、「`FAILED`」(機能していない)、または「`SYNC`」(ディスクが同期中) のいずれかです。ディスクの場合、状態は「`GOOD`」または「`FAILED`」のいずれかです。

Cache　キャッシュの入出力書き込み動作が有効になっているかどうかを示します。キャッシュは「`ON`」または「`OFF`」のいずれかです。

RAID level　RAID レベルを表示します。RAID レベルは 0、1、1E、5、10、または 50 のいずれかです。

`-l` *controller ...*
指定したコントローラに関する情報を表示します。出力には次の情報が含まれます。

Controller　RAID コントローラの ID 番号を表示します。

Type　RAID コントローラの製品タイプを表示します。

fw_version　コントローラのファームウェアのバージョンを表示します。

`[-l]`
`raidctl` ユーティリティーが操作できる、RAID に関連するすべてのオブジェクトを一覧表示します。これには、使用可能なすべての RAID コントローラ、RAID ボリューム、および物理ディスクが含まれます。`-l` オプションは省略できます。

出力には次の情報が含まれます。

Controller　RAID コントローラの ID 番号を表示します。

Volume　論理 RAID ボリューム名を表示します。

Disk　　RAID ディスクを C.ID.L の形式で表示します。

-S [volume | controller]
使用可能なすべての RAID デバイス、RAID コントローラ、ボリューム、およびディスクを含む、RAID 構成情報のスナップショットを取得します。

出力の各行には、RAID デバイスとその関連情報が空白で区切られて表示されます。すべてのボリュームとディスクは、最後に指定したコントローラに属します。

出力には次の情報が一覧表示されます。

Controller	コントローラの ID 番号と、コントローラのタイプを表す文字列 (二重引用符に囲まれている) が表示されます。
Volume	RAID ボリューム名、構成要素のディスクの数、構成要素のディスクの C.ID.L 形式、RAID レベル、および状態を表示します。状態は「OPTIMAL」、「DEGRADED」、「FAILED」、または「SYNCING」のいずれかです。
Disk	ディスクの C.ID.L 形式と状態を表示します。状態は「GOOD」、「FAILED」、または「HSP」(ディスクが予備用ディスクとして設定されている) のいずれかです。

ボリュームまたはコントローラを指定すると、指定したボリュームまたはコントローラの情報のスナップショットしか取得されません。

-S -g *disk controller*
指定したディスクの情報のスナップショットを取得します。

-h
使用法についての文字列を出力します。

使用例

例 1　RAID 構成の作成

次のコマンドは、コントローラ 0 に 10G の RAID 0 ボリュームを作成し、ストライプサイズは 64k に設定されます。

```
# raidctl -C "0.0.0 0.2.0" -r 0 -z 10g -s 64k 0
```

次のコマンドは、コントローラ 2 に RAID 1 ボリュームを作成します。

```
# raidctl -C "0.0.0 1.1.0" -r 1 2
```

次のコマンドは、コントローラ 2 に RAID 5 ボリュームを作成します。

```
# raidctl -C "0.0.0 0.1.0 0.2.0" -r 5 2
```

例 1 RAID 構成の作成 *(続き)*

次のコマンドは、コントローラ 0 に RAID 10 ボリュームを作成します。

```
# raidctl -C "(0.0.0 0.1.0)(0.2.0 0.3.0)" -r 10 0
```

次のコマンドは、コントローラ 0 に RAID 50 ボリュームを作成します。

```
# raidctl -C "(0.0.0 0.1.0 0.2.0)(0.3.0 0.4.0 0.5.0)" -r 50 0
```

例 2 RAID 構成の表示

次のコマンドは、使用可能なすべてのコントローラ、ボリューム、およびディスクを表示します。

```
# raidctl -l

Controller: 0
Controller: 2
        Volume:c2t0d0
        Disk: 0.0.0
        Disk: 0.1.0
        Disk: 0.2.0
        Disk: 0.3.0(HSP)
```

次のコマンドは、コントローラ 2 に関する情報を表示します。

```
# raidctl -l 2

Controller      Type            Fw_version
------------------------------------------------------------
c2              LSI 1030        1.03.39.00
```

次のコマンドは、指定したボリュームに関する情報を表示します。

```
# raidctl -l c2t0d0

Volume                  Size    Stripe  Status  Cache   RAID
        Sub                      Size                    Level
                Disk
------------------------------------------------------------
c2t0d0                 10240M  64K     OPTIMAL ON      RAID5
                0.0.0   5120M           GOOD
                0.1.0   5120M           GOOD
                0.2.0   5120M           GOOD
```

次のコマンドは、コントローラ 2 のディスク 0.3.0 に関する情報を表示します。

```
# raidctl -l -g 0.3.0 2

Disk    Vendor  Product         Capacity        Status  HSP
```

```
------------------------------------------------------------
0.3.0   MAXTOR  ATLAS10K4_36SC  34732M          GOOD    c2t0d0
```

例 3 RAID 構成の削除

次のコマンドは、ボリュームを削除します。

```
# raidctl -d c0t0d0
```

例 4 コントローラのフラッシュイメージの更新

次のコマンドは、コントローラ 0 のフラッシュイメージを更新します。

```
# raidctl -F lsi_image.fw 0
```

例 5 ホットスペアディスクの設定または設定解除

次のコマンドは、コントローラ 2 のディスク 0.3.0 をグローバルホットスペアディスクとして設定します。

```
# raidctl -a set -g 0.3.0 2
```

次のコマンドは、コントローラ 2 のディスク 0.3.0 を、ボリューム c2t0d0 に対するローカルホットスペアディスクとして設定します。

```
# raidctl -a set -g 0.3.0 c2t0d0
```

次のコマンドは、コントローラ 2 のディスク 0.3.0 を、グローバルホットスペアディスクから通常のディスクへ変換します。

```
# raidctl -a unset -g 0.3.0 2
```

次のコマンドは、ディスク 0.3.0 のボリューム c2t0d0 に対するローカルホットスペアディスクの設定を解除します。

```
# raidctl -a unset -g 0.3.0 c2t0d0
```

例 6 ボリュームプロパティーの設定

次のコマンドは、ボリュームの書き込みポリシーを「off」に設定します。

```
# raidctl -a set -p "wp=off" c0t0d0
```

例 7 -c オプションによるボリュームの作成

次のコマンドは、RAID 1 ボリュームを作成します。

```
# raidctl -c c0t0d0 c0t1d0
```

次のコマンドは、RAID 0 ボリュームを作成します。

例 7　-c オプションによるボリュームの作成　　(続き)

```
# raidctl -c -r 0 c0t1d0 c0t2d0 c0t3d0
```

例 8　RAID 構成情報のスナップショットの取得

次のコマンドは、すべての RAID デバイスのスナップショットを取得します。

```
# # raidctl -S

1 "LSI 1030"
c1t1d0 2 0.2.0 0.3.0 1 DEGRADED
0.2.0 GOOD
0.3.0 FAILED
```

次のコマンドは、ボリューム c1t0d0 についてのスナップショットを取得します。

```
# raidctl -S c1t0d0

c1t0d0 2 0.0.0 0.1.0 1 OPTIMAL
```

次のコマンドは、コントローラ 1 のディスク 0.1.0 についてのスナップショットを取得します。

```
# raidctl -S -g 0.1.0 1

0.1.0 GOOD
```

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

1　コマンド行入力が不正、またはアクセスが拒否された。

2　要求操作は失敗した。

属性

次の属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	開発中

関連項目

attributes(5), mpt(7D)

『Solaris のシステム管理 (基本編)』

警告

Solaris マルチパス入出力機能 (MPxIO としても知られる) を使用する場合は、内部 SAS ディスク上に RAID ボリュームを作成しないでください。Solaris マルチパス環境で新しい RAID ボリュームを作成すると、既存デバイスの GUID と一致しない新しい GUID が root デバイスに与えられます。これによって `/etc/vfstab` 内の root デバイスのエントリが一致しなくなり、ブート障害を引き起こします。

名前　rctladm – システムの資源制御の大域状態の表示または変更

形式　`rctladm [-lu] [-e action] [-d action] [name...]`

機能説明　`rctladm` コマンドを使えば、稼働中のシステム上に存在するアクティブな資源制御を検査および変更できます。資源制御のインスタンスは「*rctl*」と記述します。`rctl` については、`setrctl`(2) を参照してください。Solaris オペレーティングシステムの現行リリースでサポートされている `rctl` の一覧については、`resource_controls`(5) を参照してください。`rctl` 違反のロギングをシステム全体で有効化/無効化したり、アクティブな `rctl` (とその状態) を一覧表示したりすることができます。

オプションなしの `rctladm` コマンドは、`-l` オプション付きの `rctladm` と同等です。次の `-l` に対する説明を参照してください。

オプション　サポートしているオプションは、次のとおりです。

`-d` *action*
`-e` *action*　指定された `rctl` 上で大域アクションを無効化 (`-d`) または有効化 (`-e`) します。`rctl` が指定されなかった場合、何のアクションも実行されず、エラー状態が返されます。無効化オプションで特殊なトークン `all` を使用すれば、特定の資源制御上のすべての大域アクションを無効にすることができます。

syslog アクションには、特定の重要度レベルを割り当てることでアクションレベルを設定できます。それには、`syslog=`*level* と指定します。ここで、*level* は、`syslog`(3C) で有効な重要度レベルとして記載されている文字列トークンのうちの 1 つです。重要度レベルの共通の `LOG_` 接頭辞は、省略できます。

`-l`　`rctl` に関する情報を一覧表示します。名前、大域イベントアクションと大域状態、および大域フラグが表示されます。1 つ以上の名前オペランドが指定された場合、それらの名前に一致する `rctl` だけが表示されます。

`-u`　`/etc/rctladm.conf` の内容に基づいて資源制御を構成します。名前オペランドはすべて無視されます。

オペランド　次のオペランドがサポートされています。

name　操作対象となる `rctl` の名前。複数の `rctl` 名を指定できます。名前が 1 つも指定されず、かつ一覧表示アクションが指定された場合には、すべての `rctl` が表示されます。有効化/無効化アクションを指定する場合、1 つ以上の `rctl` 名を指定する必要があります。

使用例

例1 特定の違反に対するシステムロギングの有効化

次のコマンドは、task.max-lwps に対するすべての違反のシステムロギングを有効化します。

```
# rctladm -e syslog task.max-lwps
#
```

例2 特定資源の現在の状態の検査

次のコマンドは、task.max-lwps 資源の現在の状態を検査します。

```
$ rctladm -l task.max-lwps
task.max-lwps               syslog=DEBUG
$
```

終了ステータス

次の終了ステータスが返されます。

0 正常終了。

1 致命的なエラーが発生した。処理が失敗したそれぞれの資源制御を知らせるメッセージが、標準エラーに書き込まれます。オペランドに指定されたその他の資源制御の処理は、すべて成功しました。

2 無効なコマンド行オプションが指定された。

ファイル

/etc/rctladm.conf rctladm が実行されるたびに、rctladm.conf の内容が現在の構成に基づいて更新されます。

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWesu

関連項目

setrctl(2), getrctl(2), prctl(1), rctlblk_get_global_flags(3C), rctlblk_get_global_action(3C), attributes(5), resource_controls(5)

注意事項

デフォルトでは、rctl の違反は大域ログ作成では記録されません。

名前　reboot – restart the operating system

形式　`/usr/sbin/reboot [-dlnq] boot_arguments`

機能説明

`reboot` ユーティリティはカーネルを再起動します。カーネルは PROM モニターによってメモリーに読み込まれ、読み込まれたカーネルに制御が渡されます。

スーパーユーザーはいつでも `reboot` を実行できますが、通常は shutdown(1M) を使用して、これからサービスが停止されることを、すべてのユーザーに事前に警告する必要があります。詳細については、shutdown(1M) のマニュアルページを参照してください。

`reboot` ユーティリティは `sync`(1M) 操作をディスクに実行して、マルチユーザー状態で再起動を実行します。詳細については、init(1M) のマニュアルページを参照してください。x86 システムでは、リブートの成功を確実にするために、`reboot` が、ブートアーカイブを必要に応じて変更することがあります。

`reboot` ユーティリティは通常、再起動の記録をシステムログデーモン syslogd(1M) に送信し、シャットダウンの記録をログインアカウンティングファイル `/var/adm/wtmpx` に保存します。これらの処理を抑制するには、`-n` または `-q` オプションを指定します。

通常、システムはパワーアップ時やクラッシュ後に再起動します。

オプション　次のオプションを指定できます。

`-d`　再起動する前に、システムクラッシュダンプを行います。システムクラッシュダンプを構成する方法については、`dumpadm`(1M) のマニュアルページを参照してください。

`-l`　システムログデーモン syslogd(1M) へのメッセージ (誰が `reboot` を実行したかを示す) の送信を抑制します。

`-n`　`sync`(2) の呼び出しは行わず、syslogd(1M) または `/var/adm/wtmpx` に再起動の記録を保存しないようにします。カーネルは再起動前にファイルシステムとの同期を取ろうとします。ただし、`-d` オプションも指定した場合は例外です。`-d` と `-n` を一緒に指定した場合、カーネルはファイルシステムとの同期を取りません。

`-q`　(Quick)。実行中のプロセスを停止せずに、ただちに再起動します。

オペランド　次のオペランドを指定できます。

boot_arguments　オプションの *boot_arguments* 文字列を指定すると、`uadmin`(2) 関数に引数を指定できます。これらの引数は再起動時にブートプログラムとカーネルに渡されます。引数の形式と一覧については、boot(1M) のマニュアルページと `kernel`(1M) のマニュアルページを参照してください。これらの引数を指定する場合、引数間の空白は、シェルのために引用符で囲まれていない限り、

単一のスペースに置換されます。*boot_arguments* がハイフン (-) で始まる場合、区切り文字列 (2 つのハイフン) を *boot_arguments* のハイフンの前に指定して、reboot 引数リストの終わりを示す必要があります (「使用例」の項を参照)。

使用例

例 1　-r と -v 引数を boot に渡す

次の例では、区切り文字列「--」 (2 つのハイフン) を使用して、reboot のオプションを boot(1M) の引数から分離する必要があります。

```
example# reboot -dl -- -rv
```

例 2　特定のディスクとカーネルを使用して再起動する

次の例は、特定のディスクとカーネルを使用して再起動します。

```
example# reboot disk1 kernel.test/unix
```

ファイル

/var/adm/wtmpx　　ログインアカウンティングファイル

属性

以下の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

mdb(1), boot(1M), dumpadm(1M), fsck(1m), halt(1M), init(1M), kernel(1M), shutdown(1M), sync(1M), syslogd(1M), sync(2), uadmin(2), reboot(3C), attributes(5)

注意事項

reboot ユーティリティは /etc/rc*num*.d 内のスクリプトや inittab(4) 内の停止アクションを実行しません。システムサービスを完全に停止するためには、shutdown(1M) または init(1M) を使用して Solaris システムを再起動します。

名前　rpc.metad – リモートメタセットサービス

形式　`/usr/sbin/rpc.metad`

機能説明　`rpc.metad` は、メタデバイスのディスクセット情報のローカルコピーを管理するのに使用される `rpc`(4) デーモン (サーバープロセスとして機能する) です。`rpc.metad` デーモンは inetadm(1m) によって制御されます。

終了ステータス　以下の終了ステータスが返されます。

`0`　正常終了

`>0`　エラーが発生した

属性　以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目　`svcs`(1), inetadm(1m), inetd(1M), metaset(1M), rpc.metamhd(1M), svcadm(1M), `rpc`(3NSL), `services`(4), `attributes`(5), `smf`(5)

『Solaris ボリュームマネージャの管理』

注意事項　`rpc.metad` サービスを管理するには、サービス管理機能 `smf`(5) を使用します。このとき、次のサービス識別子を使用します。

```
svc:/network/rpc/meta:default
```

このサービスに対する管理アクション (有効化、無効化、または再起動の要求など) を実行するには、svcadm(1M) を使用します。このサービスを起動または再起動する責任は inetd(1M) に委託されています。このサービスの構成を変更したり、構成情報を表示したりするには、inetadm(1m) を使用します。サービスの状態を照会するには、`svcs`(1) コマンドを使用します。

名前

rpc.metamedd – remote mediator services

形式

```
/usr/sbin/rpc.metamedd
```

機能説明

`rpc.metamedd` is an `rpc`(4) server which is used to manage mediator information for use in 2–string HA configurations. The `rpc.metamedd` daemon is controlled by inetadm(1m).

終了ステータス

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu
Interface Stability	Evolving

関連項目

`svcs`(1), inetadm(1m), inetd(1M), svcadm(1M), `rpc`(4), `services`(4), `attributes`(5), `smf`(5)

Sun Cluster documentation, 『Solaris ボリュームマネージャの管理』

注意事項

The `rpc.metamedd` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/network/rpc/metamed:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1m) to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs`(1) command.

名前

rpc.metamhd – リモートマルチホストディスクサービス

形式

```
/usr/sbin/rpc.metamhd
```

機能説明

`rpc.metamhd` は、複数のホストに接続されているディスクを管理するのに使用される `rpc`(4) デーモン (サーバープロセスとして機能する) です。`rpc.metamhd` デーモンは inetadm(1m) によって制御されます。

終了ステータス

以下の終了ステータスが返されます。

`0` 正常終了

`>0` エラーが発生した

属性

以下の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWmdu

関連項目

`svcs`(1), inetadm(1m), inetd(1M), metaset(1M), rpc.metad(1M), svcadm(1M), `rpc`(3NSL), `services`(4), `attributes`(5), `smf`(5)

『Solaris ボリュームマネージャの管理』

注意事項

`rpc.metamhd` サービスを管理するには、サービス管理機能 `smf`(5) を使用します。このとき、次のサービス識別子を使用します。

```
svc:/network/rpc/metamh:default
```

このサービスに対する管理アクション (有効化、無効化、または再起動の要求など) を実行するには、svcadm(1M) を使用します。このサービスを起動または再起動する責任は inetd(1M) に委託されています。このサービスの構成を変更したり、構成情報を表示したりするには、inetadm(1m) を使用します。サービスの状態を照会するには、`svcs`(1) コマンドを使用します。

名前　rsh, restricted_shell – 制限付きシェルのコマンドインタプリタ

形式　`/usr/lib/rsh [-acefhiknprstuvx] [`*argument*`]...`

機能説明　rsh は、標準のコマンドインタプリタである sh と比べて機能の一部が制限されており、ログインが許される実行環境も、sh の環境と比べて制限されています。機能の詳細や使用方法に関しては、sh(1) の説明を参照してください。

シェルは、呼び出されると環境変数 SHELL を調べます。この環境変数が存在していて、その値のファイル名部分が rsh であれば、シェルは制限付きシェルとなります。

rsh の機能は、以下の動作ができない点を除き sh と同一です。

- ディレクトリの変更 (cd(1) を参照)
- $PATH の値の設定
- / を含むパスまたはコマンド名の指定
- 出力先のリダイレクト (> および >>)

これらの制限は、*.profile* の解釈後に有効となります。

制限付きシェルは、次のいずれかの方法で呼び出せます。

1. /etc/passwd ファイルの最後のエントリのファイル名部分を rsh と記述する (passwd(4) を参照)
2. 環境変数 SHELL が存在していて、その値のファイル名の部分が rsh である。環境変数 SHELL は、 .login ファイル内に設定されている必要がある
3. シェルの呼び出し時に、 引数 0 のファイル名の部分が rsh である
4. シェルを -r オプション付きで呼び出す

実行するコマンドがシェル手続きである場合は、rsh は sh を呼び出して、コマンドを実行します。したがって、利用できるコマンドの種類には制限はありますが、一般ユーザーは標準シェルの全機能を利用できるシェル手続きを使用することができます。このスキーマは、一般ユーザーが 同じディレクトリにおいて書き込み権と実行権を持っていないことを想定しています。

.profile の作成者 (profile(4) を参照) が、確実な設定処理を実行してユーザーを適切なディレクトリ (おそらく、ログインディレクトリではない) に置くことにより、ユーザーの動作を完全に制御できるという点が、これらの規約の実際の効果となります。

システム管理者は、制限付きシェルで安全に起動できるコマンドのディレクトリ (つまり /usr/rbin) を設定することがよくあります。 システムによっては、制限付きエディタ red を提供するものもあります。

終了ステータス　構文エラーなどのエラーを検出した場合、シェルは 0 以外の終了ステータスを返します。シェルを対話型以外で使用している場合、シェルファイルの実行は中止されます。対話型で使用している場合は、シェルは最後に実行されたコマンドの 終了ステータスを返します。

属性　次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目　Intro(1), cd(1), login(1), rsh(1), sh(1), exec(2), passwd(4), profile(4), attributes(5)

注意事項　制限付きシェル /usr/lib/rsh を、リモートシェル /usr/bin/rsh と混同しないように注意してください。リモートシェルに関しては rsh(1) を参照してください。

名前　scadm – システムコントローラ (SC) の管理

形式

```
/usr/platform/platform-name/sbin/scadm
 subcommand [option] [argument]...
```

機能説明

scadm ユーティリティーは、システムコントローラ (SC) を管理します。このユーティリティーを使用すると、ホストサーバーが SC と対話できるようになります。

scadm ユーティリティーは、スーパーユーザーとして実行しなければなりません。

scadm のインタフェース、出力、およびディレクトリ階層内の位置は確定したものではなく、変更されることがあります。

platform-name はプラットフォームの実装名です。uname -i コマンドを使用して、プラットフォーム実装を識別します。uname(1) を参照してください。

scadm ユーティリティーにはいくつかのサブコマンドがあります。一部のサブコマンドには、そのサブコマンドに関連付けられている特定のオプションと引数があります。「サブコマンド」、「オプション」、「オペランド」、および「使用例」を参照してください。

サブコマンド

コマンド行の scadm コマンドの後に SPACE を入力し、その後にサブコマンドを指定します。

次のサブコマンドを指定できます。

consolehistory

SC のコンソールログを表示します。SC は、すべてのコンソール出力を捕捉する動的なログを維持管理します。このログは FIFO (first-in, first-out) バッファーとして管理されます。バッファーがいっぱいになると、新しいコンソール出力が古いコンソール出力に取って代わります。デフォルトでは、コンソールログファイルの最後の 8K バイトのみが表示されます。

省略可能な -a 引数は、コンソールログファイルの全体が表示されることを表します。

consolehistory サブコマンドの読み取り速度を超えるスピードで、SC がこのログに情報を記録する可能性があります。したがって、ログデータの一部が、表示される前に失われてしまう可能性があります。そのようなことが起こった場合、consolehistory サブコマンドは、「scadm: lost <数> bytes of console log data」とログ出力内に表示し、データの一部が失われたことを通知します。

consolehistory サブコマンドの書式を次に示します。

```
scadm consolehistory [-a]
```

consolehistory サブコマンドは、すべてのプラットフォーム上で使えるわけではありません。このコマンドをサポートしないプラットフォーム上でこのコマンドが使用された場合、scadm は次のメッセージを出力します。

```
scadm: command/option not supported
```

そして、0 以外の状態で終了します。

date

SC の日付と時刻を表示します。

date サブコマンドの書式を次に示します。

```
scadm date
```

download

SC のファームウェアをプログラムします。

ファームウェアについては、ブートモニターとメインイメージの 2 つの部分があります。

デフォルトでは、scadm コマンドのダウンロードを実行すれば、メインファームウェアイメージがプログラムされます。boot 引数を指定すると、ブートモニターのプログラミングが選択されます。

download サブコマンドの書式を次に示します。

```
scadm download [boot] file
```

fruhistory

SC によって維持管理される「fru (field replacable unit)」ログの内容を表示します。デフォルトでは、fru 履歴ログファイルの最後の 8K バイトのみが表示されます。このログには、SC の「showfru」コマンドのスナップショットが格納されます。このスナップショットは、システムがリセットされるか SC によってホットプラグイベントが検出されるたびに取得されます。

省略可能な -a 引数は、fru ログファイルの全体が表示されることを表します。

fruhistory サブコマンドの読み取り速度を超えるスピードで、SC がこのログに情報を記録する可能性があります。したがって、ログデータの一部が、表示される前に失われてしまう可能性があります。そのようなことが起こった場合、fruhistory サブコマンドは、「scadm: lost <数> bytes of fru log data」とログ出力内に表示し、データの一部が失われたことを通知します。

fruhistory サブコマンドの書式を次に示します。

```
scadm fruhistory [-a]
```

`fruhistory` サブコマンドは、すべてのプラットフォーム上で使えるわけではありません。このコマンドをサポートしないプラットフォーム上でこのコマンドが使用された場合、`scadm` は次のメッセージを出力します。

```
scadm: command/option not supported
```

そして、0 以外の状態で終了します。

help　コマンドの一覧を表示します。

`help` サブコマンドの書式を次に示します。

```
scadm help
```

loghistory　SC イベントログの最新のエントリを表示します。省略可能な `-a` 引数を指定すると、イベントログ履歴の全体が表示されます。`-a` 引数は、大規模ログファイルをサポートするプラットフォーム上でのみ使用できます。大規模ログファイルをサポートしないプラットフォーム上では、このフラグは何の効果も持ちません。

`loghistory` サブコマンドの読み取り速度を超えるスピードで、SC がこのログに情報を記録する可能性があります。したがって、ログデータの一部が、表示される前に失われてしまう可能性があります。そのようなことが起こった場合、`loghistory` サブコマンドは、「scadm: lost <数> events」とログ出力内に表示し、データの一部が失われたことを通知します。

`loghistory` サブコマンドの書式を次に示します。

```
scadm loghistory [-a]
```

resetrsc　SC をリセットします。実行できるリセットには、ハードリセットとソフトリセットの 2 種類があります。デフォルトで実行されるのがハードリセットです。ソフトリセットを選択する場合は `-s` オプションを指定します。

`resetrsc` サブコマンドの書式を次に示します。

```
scadm resetrsc [-s]
```

send_event　テキストベースのイベントを手動で送信します。SC は、イベントを SC イベントログに転送できます。`-c` オプションを構成して、重要な警告を電子メール、ログイン中の SC ユーザー、および `syslog` に送信できます。重要なイベントは `syslog`(3C) に記録されます。関連テキストメッセージの文字数は `80` 文字以内です。

`send_event` サブコマンドの書式を次に示します。

```
scadm send_event [-c] "message"
```

set
: SC 構成変数の値を設定します。

  SC 構成変数には、次のものが含まれます。SC IP アドレス `netsc_ipaddr` や、SC カスタマ情報 `sc_customerinfo` などです。SC 構成変数の全リストについては、`scadm help` コマンドの出力結果を参照してください。

  `set` サブコマンドの書式を次に示します。

  `scadm set` *variable value*

show
: SC 構成変数の現在の設定を表示します。変数を指定しないで `scadm` を実行すると、すべての変数設定が表示されます。

  `show` サブコマンドの書式を次に示します。

  `scadm show` [*variable*]

shownetwork
: SC の現在のネットワーク構成パラメタを表示します。

  `shownetwork` サブコマンドの書式を次に示します。

  `scadm shownetwork`

useradd
: SC にユーザーアカウントを追加します。SC では 16 名までの個別ユーザーがサポートされます。

  `useradd` サブコマンドの書式を次に示します。

  `scadm useradd` *username*

userdel
: SC からユーザーアカウントを削除します。

  `userdel` サブコマンドの書式を次に示します。

  `scadm userdel` *username*

userpassword
: 指定したユーザーアカウントのパスワードを設定します。このパスワードは、現在設定されている既存のパスワードがあった場合にはそれより優先されます。新しいパスワードを設定する前に、古いパスワードは検証されません。

  `userpassword` サブコマンドの書式を次に示します。

  `scadm userpassword` *username*

userperm
: ユーザーのアクセス権のレベルを設定します。

  `userperm` サブコマンドの書式を次に示します。

  `scadm userperm` *username* `[aucr]`

usershow　指定したユーザーアカウントの詳細を表示します。ユーザー名を指定しないと、すべてのユーザーアカウントの詳細が表示されます。

`usershow` サブコマンドの書式を次に示します。

```
scadm usershow username
```

version　SC のバージョン番号とそのコンポーネントを表示します。

`version` サブコマンドの書式を次に示します。

```
scadm version [-v]
```

オプション　`resetrsc`、`send_event`、および `version` サブコマンドには、関連するオプションがあります。コマンド行のサブコマンドの後に SPACE を入力し、その後にオプションを指定します。

`resetrsc` サブコマンドでは次のオプションを指定できます。

`-s`　ハードリセットの代わりにソフトリセットを実行します。ハードリセットは SC ハードウェアを物理的にリセットします。ソフトリセットの場合は、SC ソフトウェアがブートファームウェアにジャンプして、リセットをシミュレーションします。

`send_event` サブコマンドでは次のオプションを指定できます。

`-c`　重要なイベントを送信します。`-c` を指定しないで `-send_event` を実行すると、警告が送信されます。

`version` サブコマンドでは次のオプションを指定できます。

`-v`　バージョン番号と関連情報の詳細な内容が表示されます。

`consolehistory`、`fruhistory`、および `loghistory` サブコマンドは、次のオプションをサポートします。

`-a`　ログの全体を表示します。これらのサブコマンドは通常、最新のログデータだけを表示します。このフラグを指定すると、ログの全体が表示されるようになります。

オペランド　`download`, `send_event`, `set`, `show`, `useradd`, `userdel`, `userperm`, `usershow`, `userpassword`, `userperm` の各サブコマンドには関連する引数 (オペランド) があります。

サブコマンドにオプションを指定する場合は、コマンド行のオプションの後に SPACE を入力し、その後に引数を指定します。サブコマンドにオプションを指定しない場合は、コマンド行のサブコマンドの後に SPACE を入力し、その後に引数を指定します。引数を 2 つ以上指定する場合は、それぞれの引数の後に SPACE を入力して区切ります。

`download` サブコマンドでは次の引数を指定できます。

boot　フラッシュのブートモニター部分をプログラムします。フラッシュのメイン部分は、引数ファイルを使用せずにプログラムされます。

file　ダウンロード用のブートまたはメインファームウェアイメージの格納場所へのパスを *file* で指定します。

file の例は、次のとおりです。

```
/usr/platform/platform_type/lib/image/alommainfw
```

または

```
/usr/platform/platform_type/lib/image/alombootfw
```

`send_event` サブコマンドでは次の引数を指定できます。

"*message*"　*message* に指定されているテキストを使用して、イベントを記述します。*message* は引用符で囲みます。

`set` サブコマンドでは次の引数を指定できます。

variable　SC 構成 *variable* を設定します。

value　SC 構成変数の *value* を設定します。

`show` サブコマンドでは次の引数を指定できます。

variable　指定した特定の変数の値を表示します。

`useradd` サブコマンドでは次の引数を指定できます。

username　新しい SC アカウント *username* を追加します。

`userdel` サブコマンドでは次の引数を指定できます。

username　SC アカウント *username* を削除します。

`userperm` サブコマンドでは次の引数を指定できます。

`-aucr`　SC ユーザーアカウントのアクセス権を設定します。アクセス権を指定しない場合、4 つのアクセス権はすべて無効になり、読み取り専用アクセス権が割り当てられます。

アクセス権の定義を次に示します。

a　ユーザーは SC 構成変数を管理または変更できます。

u　ユーザーは、ユーザーコマンドを使用して SC アカウントを変更できます。

c　ユーザーはコンソールに接続できます。

r　ユーザーはSCのリセットと、ホストの電源投入および切断ができます。

username　SCアカウント*username*に対するアクセス権を変更します。

`usershow`サブコマンドでは次の引数を指定できます。

username　SCアカウント*username*に関する情報を表示します。*username*を指定しないと、すべてのアカウントの情報が表示されます。

`userpassword`サブコマンドでは次の引数を指定できます。

username　*username*に対するSCパスワードを設定します。

`userperm`サブコマンドでは次の引数を指定できます。

username　*username*に対するSCアクセス権を変更します。

使用例

例1　SCの日付と時刻を表示する

次のコマンドは、SCの日付と時刻を表示します。

```
scadm date
```

例2　SCの構成変数を設定する

次のコマンドは、SCの構成変数`netsc_ipaddr`を`192.168.1.2`に設定します。

```
scadm set netsc_ipaddr 192.168.1.2
```

例3　SCの現在の構成設定を表示する

次のコマンドは、SCの現在の構成設定を表示します。

```
scadm show
```

例4　変数の現在の設定を表示する

次のコマンドは、`sys_hostname`という変数の現在の設定を表示します。

```
scadm show sys_hostname
```

例5　テキストベースの重要なイベントを送信する

次のコマンドは、重要なイベントをSCイベントログに送信し、現在のSCユーザーに警告するとともに、イベントを`syslog`(3C)に送信します。

```
scadm send_event -c "The UPS signaled a loss in power"
```

例 6　参照用のテキストベースのイベントを送信する

次のコマンドは、重要ではない参照用のテキストベースのイベントを SC イベントログに送信します。

```
scadm send_event "The disk is close to full capacity"
```

例 7　ユーザーを SC に追加する

次のコマンドは、ユーザー rscroot を SC に追加します。

```
scadm useradd rscroot
```

例 8　SC からユーザーを削除する

次のコマンドは、ユーザー olduser を SC から削除します。

```
scadm userdel olduser
```

例 9　ユーザーの詳細を表示する

次のコマンドは、すべてのユーザーアカウントの詳細を表示します。

```
scadm usershow
```

例 10　特定のユーザーの詳細を表示する

次のコマンドは、ユーザーアカウント rscroot の詳細を表示します。

```
scadm usershow rscroot
```

例 11　ユーザーのアクセス権のレベルを設定する

次のコマンドは、ユーザー rscroot に全アクセス権 aucr を設定します。

```
scadm userperm rscroot aucr
```

例 12　ユーザーのアクセス権のレベルを設定する

次のコマンドは、ユーザー newuser にコンソールアクセス権 c だけを設定します。

```
scadm userperm newuser c
```

例 13 ユーザーのアクセス権のレベルを設定する

次のコマンドは、ユーザー newuser のアクセス権を読み取り専用に設定します。

```
scadm userperm newuser
```

例 14 現在のネットワークパラメタを表示する

次のコマンドは、SC の現在のネットワーク構成パラメタを表示します。

```
scadm shownetwork
```

例 15 コンソール履歴を表示する

次のコマンドは、SC イベントログのコンソールの内容を表示します。

```
scadm consolehistory [-a]
```

例 16 fru 履歴を表示する

次のコマンドは、SC イベントログの「fru (field replacable unit)」の内容を表示します。

```
scadm fruhistory [-a]
```

例 17 ログ履歴を表示する

次のコマンドは、SC イベントログの最新のエントリを表示します。

```
scadm loghistory [-a]
```

例 18 詳細情報を表示する

次のコマンドは、SC とそのコンポーネントの詳細なバージョン情報を表示します。

```
scadm version -v
```

終了ステータス

次の終了値が返されます。

0 正常終了。

0 以外 エラーが発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWkvm
インタフェースの安定性	開発中

関連項目

uname(1)、syslog(3C)、attributes(5)

名前

share – ローカル資源をリモートシステムからマウントできるようにする

形式

```
share  [-F FSType] [-o specific_options] [-d description]
 [pathname]
```

機能説明

share コマンドはローカル資源をエクスポートして、*FSType* のリモートファイルシステムからマウントできるようにします。オプション -F *FSType* を指定しない場合、/etc/dfs/fstypes 内の最初のファイルシステムタイプがデフォルトとして使用されます。NFS に固有なオプションについては、share_nfs(1M) のマニュアルページを参照してください。*pathname* は共有するディレクトリのパス名です。引数をなにも指定しないと、share は共有しているすべてのファイルシステムを表示します。

オプション

次のオプションを指定できます。

-F *FSType*
: ファイルシステムタイプを指定します。

-o *specific_options*
: *specific_options* を使用すると、共有する資源へのアクセスを制御できます。NFS に固有なオプションについては、share_nfs(1M) のマニュアルページを参照してください。*specific_options* は次のとおりです。

  rw
  : すべてのクライアントが *pathname* を読み書き両用で共有します。これはデフォルトの動作です。

  rw=*client*[:*client*]...
  : 指定されたクライアントだけが *pathname* を読み書き両用で共有します。他のシステムは *pathname* にアクセスできません。

  ro
  : すべてのクライアントが *pathname* を読み取り専用で共有します。

  ro=*client*[:*client*]...
  : 指定されたクライアントだけが *pathname* を読み取り専用で共有します。他のシステムは *pathname* にアクセスできません。

  複数のオプションはコロンで区切ります。

-d *description*
: -d オプションは、共有する資源の説明に使います。

使用例

例1 ファイルシステムを読み取り専用で共有する

次の例では、起動時に、(すべてのクライアントが) /disk ファイルシステムを読み取り専用で共有します。

```
share -F nfs -o ro /disk
```

例 2　複数のオプションを呼び出す

次の例では、netgroup_name 内のメンバーは読み取り専用で、hostname 内のユーザーは読み書き両用で、/export/manuals ファイルシステムを共有します。

```
share -F nfs -o ro=netgroup_name:rw=hostname /export/manuals
```

ファイル

/etc/dfs/dfstab	起動時に実行する share コマンドのリスト
/etc/dfs/fstypes	ファイルシステムタイプのリスト (デフォルトは NFS)
/etc/dfs/sharetab	共有するファイルシステムのシステム記録

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

mountd(1M), nfsd(1M), share_nfs(1M), shareall(1M), unshare(1M), attributes(5)

注意事項

エクスポート (古い用語): SunOS 4.x では、ファイルシステムの共有は「エクスポート」と呼ばれていました。そのため、share コマンドは exportfs(1B) または /usr/sbin/exportfs として呼び出されました。

share コマンドを同じファイルシステムに複数回呼び出した場合、最後に呼び出した share コマンドが前の share コマンドに優先します。つまり、最後の share コマンドで設定したオプションが前のオプションに置き換わります。たとえば、/somefs を usera に読み書き両用のアクセス権を与えた場合、/somefs を userb にも読み書き両用のアクセス権が与えられます。

```
example% share -F nfs -o rw=usera:userb /somefs
```

この動作はルートファイルシステムの共有だけに制限されず、すべてのファイルシステムに適用されます。

名前

shareall, unshareall – 複数リソースの共有または共有解除

形式

```
shareall  [-F FSType [,FSType]...] [-| file]

unshareall  [-F FSType [,FSType]...]
```

機能説明

引数を指定せずに shareall を使用すると、*file* (share コマンド行のリストを含む) 内に指定されているすべてのリソースが共有されます。オペランドにハイフン (–) を指定すると、share コマンド行は標準入力から読み取られます。*file* とハイフンのどちらも指定しないと、ファイル /etc/dfs/dfstab がデフォルトで使用されます。

複数のファイルシステムタイプをコンマで区切り、-F の引数として指定すると、特定のファイルシステムタイプのみのリソースが共有されます。

unshareall は、現在共有されているすべてのリソースの共有を解除します。-F フラグを指定しないと、すべてのファイルシステムタイプのリソースについて共有が解除されます。

オプション

-F *FSType*　ファイルシステムを指定します。デフォルトは、/etc/dfs/fstypes 内の最初のエントリです。

ファイル

/etc/dfs/dfstab

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

share(1M), unshare(1M), attributes(5)

名前　showmount – リモートマウントの表示

形式　`/usr/sbin/showmount [-ade] [`*host*`]`

機能説明　`showmount` は、ホスト (*host*) からファイルシステムをリモートマウントしているクライアントを表示します。この情報は、*host* 上の`mountd`(1M) サーバーが管理し、クラッシュ後もファイル `/etc/rmtab` 内に保存されます。*host* のデフォルト値は `hostname`(1) が返す値です。

`showmount` コマンドは、NFS バージョン 4 のクライアントの名前を表示しません。

オプション　次のオプションを指定できます。

`-a`　すべてのリモートマウント情報を次の形式で表示します。

ホスト名 : ディレクトリ

ホスト名はクライアントの名前で、ディレクトリはマウントされているファイルシステムの元です。

`-d`　クライアントがリモートマウントしたディレクトリを表示します。

`-e`　共有されているファイルシステムの一覧を表示します。

ファイル　`/etc/rmtab`

属性　次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプA	属性値
使用条件	SUNWnfscu

関連項目　`hostname`(1), `mountd`(1M), `attributes`(5)

使用上の留意点　クライアントがクラッシュしても、そのエントリがサーバー上のリモートマウントリストから削除されることはありません。

名前　shutdown – システムの停止とシステム状態の変更

形式　`/usr/sbin/shutdown [-y] [-g` *grace-period*`] [-i` *init-state*`]`
`[`*message*`]`

機能説明　`shutdown` は、マシンの状態を変更するために、スーパーユーザーが実行します。通常は、マルチユーザー状態 (state 2) からほかの状態に移行するために使用されます。

デフォルトでは、`shutdown` は、コンソール以外はオペレーティングシステムにアクセスできないシステム状態にします。この状態はシングルユーザー状態と呼ばれます。

デーモンを停止させたりプロセスを強制終了させたりする前に、`shutdown` は警告メッセージを出力するとともに、デフォルトでは確認のための最終メッセージも出力します。*message* は、標準の警告メッセージ 'The system will be shut down in ...' に続いて出力される文字列です。この文字列に 2 つ以上の語句を含める場合は、一重引用符 (') または二重引用符 (") で囲む必要があります。

この警告メッセージとユーザーが作成した *message* は、`shutdown` コマンドが開始されるまでの残り時間が 7200 秒、3600 秒、1800 秒、1200 秒、600 秒、300 秒、120 秒、60 秒、および 30 秒の時点で出力されます。「使用例」の項を参照してください。

各システム状態の定義は次のとおりです。

state 0　オペレーティングシステムを停止します。

state 1　state 1 は管理状態と呼ばれます。state 1 では、マルチユーザーの操作に必要なファイルシステムがマウントされ、マルチユーザーファイルシステムにアクセスする必要があるログインを使用できます。システムがファームウェアモードから state 1 に移行する時はコンソールだけがアクティブであり、ほかのマルチユーザー (stete 2) サービスは使用できません。マルチユーザー状態から state 1 への移行時にすべてのユーザープロセスが停止されるわけではないことに注意してください。

state s, S　state s (または S) はシングルユーザー状態と呼ばれます。この状態への移行時にはすべてのユーザープロセスが停止されます。シングルユーザー状態では、マルチユーザーログインに必要なファイルシステムのマウントが解除され、システムへのアクセスはコンソールを介してしか行えません。マルチユーザーファイルシステムへアクセスする必要があるログインは使用できません。

state 5　電源を切っても安全なようにマシンを停止し、可能であれば電源を自動的に切断します。`rc0` プロシージャがこのために呼び出されます。

state 6　オペレーティングシステムを停止したあと、`/etc/inittab` の `initdefault` エントリに定義されている状態でリブートします。`rc6` プロシージャがこのために呼び出されます。

オプション　次のオプションを指定できます。

`-y`　ユーザーの介入なしでコマンドが実行されるように、確認の問い合わせにあらかじめ応答します。

`-g` *grace-period*　デフォルトの 60 秒を、スーパーユーザーが変更できるようにします。秒数を指定します。

`-i` *init-state*　`init` が移行する状態を指定します。デフォルトは、システム状態 `s` です。

使用例

例 1　`shutdown` の使用

この例では、ホスト `foo` で `shutdown` が実行されており、120 秒後に停止がスケジュールされています。警告メッセージは、最終の確認メッセージの 2 分前、1 分前、および 30 秒前に出力されます。

```
example# shutdown -i S -g 120 "===== disk replacement ====="
Shutdown started.   Tue Jun   7  14:51:40 PDT  1994

Broadcast Message from root (pts/1) on foo Tue Jun  7 14:51:41. . .
The system will be shut down in 2 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:52:41. . .
The system will be shut down in 1 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:53:41. . .
The system will be shut down in 30 seconds
===== disk replacement =====
Do you want to continue? (y or n):
```

ファイル　`/etc/inittab`　`init` によるプロセスディスパッチを制御します。

属性　次の属性については、`attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目　boot(1M), halt(1M), init(1M), `killall`(1M), reboot(1m), `ufsdump`(1M), `init.d`(4), `inittab`(4), `nologin`(4), `attributes`(5)

注意事項　システムが `S` (または `s`) 状態に移行するときは、`/etc/nologin` ファイル (`nologin`(4) のマニュアルページを参照) が作成されます。その後 state 2 (マルチユーザー状態) へ移行する時点で、このファイルは `/etc/rc2.d` ディレクトリにあるスクリプトによって削除されます。

名前

snmpdx – Sun Solstice Enterprise Master Agent

形式

```
/usr/lib/snmp/snmpdx [-hy] [-a filename] [-c config-dir]
     [-d debug-level] [-i filename] [-m GROUP -m SPLIT]
     [-o filename] [-p port] [-r filename]
```

機能説明

The Master Agent, `snmpdx`, is the main component of Solstice Enterprise Agent (SEA) technology. It runs as a daemon process and listens to User Datagram Protocol (UDP) port 161 for SNMP requests. The Master Agent also opens another port to receive SNMP trap notifications from various subagents. These traps are forwarded to various managers, as determined by the configuration file.

Upon invocation, `snmpdx` reads its various configuration files and takes appropriate actions by activating subagents, determining the subtree Object Identifier (OID) for various subagents, populating its own Management Information Bases (MIBs), and so forth. The Master Agent invokes subagents, registers subagents, sends requests to subagents, receives responses from subagents, and traps notifications from subagents.

The Master Agent is invoked by the service management facility `smf`(5) at boot time if `svc:/application/management/snmpdx` is enabled (see NOTES) and contents of the resource configuration file `/etc/snmp/conf/snmpdx.rsrc` are non-trivial.

注 – The SMA (Systems Management Agent) is the default SNMP agent in the Solaris operating system. See `netsnmp`(5). `snmpdx` is Obsolete and may not be supported in a future release of Solaris.

オプション

The following options are supported:

`-a`*filename*	Specify the full path of the access control file used by the Master Agent. The default access control file is `/etc/snmp/conf/snmpdx.acl`.
`-c`*config-dir*	Specify the full path of the directory containing the Master Agent configuration files. The default directory is `/etc/snmp/conf`.
`-d`*debug-level*	Debug. Levels from 0 to 4 are supported, giving various levels of debug information. The default is 0 which means no debug information is given.
`-h`	Help. Print the command line usage.
`-i`*filename*	Specify the full path of the enterprise-name OID map. This file contains the PID used by the Master Agent for recovery after a crash. It contains tuples of the UNIX process ID, port number, resource name, and agent name. The default file is `/var/snmp/snmpdx.st`.
`-m GROUP | -m SPLIT`	Specify the mode to use for forwarding of SNMP requests.

`GROUP` Multiple variables can be included in each request from the Master Agent to the subagents. This results in, at mose, one send-request per agent.

`SPLIT` Each variable in the incoming request results in one send-request to each subagent.

The default is `GROUP`.

`-o`*filename* Specify the full path of the file containing the tuple (enterprise-name, OID). For example, (Sun Microsystems, 1.3.1.6.1.4.32). The Master Agent uses this file as a base for look-up in the trap-filtering and forwarding process. The default file is `/etc/snmp/conf/enterprises.oid`.

`-p`*port* Specify the port number. The default port number is 161.

`-r`*filename* Specify the full path of the resource file to be used by the Master Agent. This file stores information about the subagents that the Master Agent invokes and manages. The default resource file is `/etc/snmp/conf/snmpdx.rsrc`.

`-y` Set a recovery indicator to invoke the recovery module. The recovery process discovers which subagents in the previous session are still active; those subagents not active are re-spawned by the Master Agent.

ファイル

`/etc/snmp/conf/enterprises.oid`	Enterprise-name OID map
`/etc/snmp/conf/snmpdx.acl`	Access control file
`/etc/snmp/conf/snmpdx.rsrc`	Resource configuration file
`/var/snmp/snmpdx.st`	Master Agent status file
`/var/snmp/mib/snmpdx.mib`	Master Agent MIB file

終了ステータス

The following error values are returned:

`0` Successful completion.

non-zero An error occurred.

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsasnm
Interface Stability	Obsolete

関連項目 attributes(5), netsnmp(5), smf(5)

注意事項 The snmpdx service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/application/management/snmpdx
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

名前　snoop – ネットワークパケットの取得および検査

形式

```
snoop  [-aqrCDNPSvV] [-t [r |  a |  d]] [-c maxcount]
 [-d device] [-i filename] [-n filename] [-o filename]
 [-p first [, last]] [-s snaplen] [-x offset [, length]]
 [expression]
```

機能説明　snoop はネットワークからのパケットを取得し、その内容を表示します。snoop は、ネットワークパケットフィルタとストリームバッファーモジュールの両方を使用して、効率的にネットワークからパケットを取得します。取得したパケットは、受信した順に表示することも、あとで検査するためファイル (RFC 1761 準拠) に保存することもできます。

snoop は、パケットを単一行の要約形式または詳細な複数行形式で表示できます。要約形式では、もっとも高いレベルのプロトコルに関連したデータのみが表示されます。たとえば、NFS パケットには NFS 情報だけが表示されます。RPC、UDP、IP、および Ethernet のフレーム情報は抑止されますが、verbose (詳細表示) オプションのいずれかを選択してあれば表示できます。

LDAP または NIS などのネームサービスがない場合、snoop は数値 IP アドレスとしてホスト名を表示します。

snoop には、対話型のインタフェースが必要です。

オプション

-C　カーネルのパケットフィルタまたは snoop 独自のフィルタのいずれかに対するフィルタ式から生成されたコードをリスト表示します。

-D　要約行の取得中にドロップしたパケットの数を表示します。

-N　取得したファイルから IP アドレスと名前の対照ファイルを作成します。これは、取得したファイルの名前と -i オプションとともに設定する必要があります。アドレスと名前の対照ファイル名は、取得ファイル名に .names を付加したものです。このファイルは、取得するサイトで IP アドレスからホスト名へのマッピングを記録し、取得したファイルの可搬性を高めます。取得ファイルをどこか別の場所で分析する場合には、.names ファイルを生成します。このフラグが使用されたときは、パケットは表示されません。

-P　プロミスキュアスでないモードでパケットを取得します。ブロードキャスト、マルチキャスト、またはホストマシン宛に送られたパケットだけが見えます。

-S　要約行のリンクレイヤーフレーム全体のサイズをバイト単位で表示します。

-V　詳細要約モードです。冗長度の程度では、要約モードと詳細モードの中間です。パケット内のもっとも高いレベルのプロト

コルの要約行だけを表示する代わりに、パケット内のそれぞれのプロトコルレイヤーの要約行を表示します。たとえば、NFS パケットでは ETHER、IP、UDP、RPC および NFS レイヤーそれぞれに対して 1 行表示します。詳細要約モードは `grep` を介してパイプすることで、容易に必要とするパケットを抽出できます。たとえば、RPC 要約行だけを表示するには、次のように入力します。`example#` **`snoop -i rpc.cap -V | grep RPC`**

`-a` パケットを `/dev/audio` で聴きます (警告: うるさいかもしれません)。

`-c` *maxcount* *maxcount* パケットを取得したあとに終了します。指定しない場合は、残りのディスク容量がなくなるまで、または Control + C で割り込むまで取得し続けます。

`-d` *device* たとえば `eri0` または `hme0` のように、*device* で指定したインタフェースを使用してネットワークからのパケットを受信します。プログラム `netstat`(1M) は `-i` フラグ付きで呼び出されると、マシンにあるすべてのインタフェースを一覧表示します。通常 `snoop` は、最初に検出したループバックしないインタフェースを自動的に選択します。

`-i` *filename* 以前 *filename* で取得したパケットを表示します。このオプションを指定しない場合、`snoop` はネットワークインタフェースからパケットを読み取ります。*filename* `.names` ファイルが存在する場合、ファイルは自動的に `snoop` IP アドレスと名前のマッピングテーブル (`-N` フラグを参照) にロードされます。

`-n` *filename* *filename* を IP アドレスと名前のマッピングテーブルとして使用します。このファイルは `/etc/hosts` ファイル (IP アドレスの次にホスト名) の形式と同じである必要があります。

`-o` *filename* 取得したパケットを、取得したとおりに *filename* に保存します。(この *filename* を「取得ファイル」と呼びます。)取得ファイルの形式は、RFC 1761 に準拠しています。パケットの取得中、ファイルに保存されたパケット数のカウントが表示されます。ファイルに保存せずにパケットのカウントだけを行う場合は、ファイル名を `/dev/null` とします。

`-p` *first* [, `last`] 取得ファイルから、表示するパケットを 1 つ以上選択します。ファイルの最初の *first* パケットの番号は 1 です。

`-q` ネットワークパケットをファイルに取得するとき、パケットのカウントを表示しません。これにより、パケット取得のパフォーマンスが向上します。

`-r` IP アドレスを記号名に解決しません。これにより、`snoop` がパケットを取得および表示している間に、ネットワークトラ

フィックが生じなくなります。ただし、-n オプションが使用されている、およびアドレスがマッピングファイルに存在する場合には、対応する名前が使用されます。

-s *snaplen*　各パケットの *snaplen* バイトより後ろを切り捨てます。通常はパケット全体が取得されます。このオプションは、特定のパケットヘッダー情報だけが必要な場合に便利です。パケットの切り捨てはカーネル内で行われるため、ストリームパケットバッファーを効率的に利用できます。これにより、トラフィックが過多の期間中のバッファーオーバーフローによってパケットがドロップされる可能性が減ります。また、大量のトレースを取得ファイルに取得するときにディスク容量を節約できます。IP ヘッダーのみ (オプションなし) を取得するには、34 の *snaplen* を使用します。UDP では 42、TCP では 54 を使用します。RPC ヘッダーは 80 バイトの *snaplen* で取得できます。NFS ヘッダーは 120 バイトで取得できます。

-t [-r | -a | -d]　タイムスタンプの表示。タイムスタンプの精度は 4 マイクロ秒以内です。デフォルトでは、-d (デルタ) 形式 (以前のパケットを受信してからの時間) で時間が表示されます。オプション -a (絶対的) では、時計時刻になります。オプション -r (相対的) では、最初のパケットを表示した時間からの相対的な時間になります。これを -p オプションとともに使用すると、任意に選択したパケットに対する相対的な時間を表示できます。

-v　詳細表示 (Verbose) モードです。パケットヘッダーを数多くの詳細付きで表示します。この表示方式では 1 つのパケットについて多くの行を消費するため、選択したパケットに対してのみ使用するべきです。

-x*offset* [, *length*]　パケットデータを 16 進数および ASCII 形式で表示します。*offset* 値および *length* 値は、表示されるパケットの一部を選択します。パケット全体を表示するには、*offset* の 0 の値を使用します。*length* 値を指定しない場合、パケットの残りの部分が表示されます。

オペランド

expression　パケットをネットワーク、または取得ファイルのいずれかから選択します。式が true となるパケットのみ選択されます。式が指定されない場合は、真であるとみなされます。

フィルタ式を指定した場合、snoop はカーネルパケットフィルタ、またはそれ自体の内部フィルタのためのコードを生成します。ネットワークインタフェースでパケットを取得する場合、カーネルパケットフィルタ用のコードが生成されます。このフィルタは、バッファーモジュールの上位であるストリームモジュールとして実装されます。バッファーモジュールはいっぱいになるまでパケットを蓄積し、

snoop に渡します。カーネルパケットフィルタは、カーネル内の不要なパケットをパケットバッファーまたは snoop に到達する前に拒否するので、非常に効率的です。カーネルパケットフィルタは、その実装に関していくつかの制限があります。カーネルが処理できないフィルタ式を作成することが可能です。この場合、snoop はフィルタの分割を試み、カーネルで可能な限りのフィルタリングを試みます。残りのフィルタリングは、snoop 用のパケットフィルタによって行われます。-C フラグを使用して、カーネル用のパケットフィルタまたは snoop 用のパケットフィルタのいずれかに生成されたコードを表示できます。パケットが -i オプションを使用して取得ファイルから読み込まれた場合、snoop 用のパケットフィルタのみ使用されます。

フィルタの *expression* は、1 つまたは複数のブール型プリミティブで構成されます。複数のプリミティブは、ブール型演算子 (AND、OR、および NOT) で組み合わせることができます。ブール型演算子の通常の優先順位規則が適用されます。これらの演算子の評価の順序は、括弧で制御することができます。シェルは括弧およびほかのフィルタ式の文字を認識するため、多くの場合フィルタ式を引用符で囲む必要があります。より効率的なフィルタの設定についての情報は、例 2 を参照してください。

プリミティブに含まれるのは次のとおりです。

host *hostname*

発信元アドレスまたは着信先アドレスが hostname と同じ場合に true になります。*hostname* 引数は、アドレス表記でもかまいません。名前がほかの式のプリミティブの名前と競合しない場合は、キーワード host を省略できます。たとえば、「pinky」では、ホスト pinky が着信先または発信元であるパケットが選択され、「pinky and dinky」では、2 つのホスト pinky と dinky との間で交換されるパケットが選択されます。

使用されるアドレスのタイプは、host プリミティブに先行するプリミティブに依存します。使用可能な修飾子は、"inet"、"inet6"、"ether"、または修飾子なしです。以降では

これら 3 つのプリミティブについて説明します。プリミティブが 1 つもない状態は、「inet host hostname or inet6 host hostname」と同等です。つまり、snoop はホスト名と関連付けられた IP アドレスをすべてフィルタリングしようとします。

inet または *inet6*

`host` プリミティブを変更する修飾子は、次のとおりです。*inet* の場合、`snoop` は名前の検索によって返された IPv4 アドレスすべてをフィルタリングしようとします。*inet6* の場合、`snoop` は名前の検索によって返された IPv6 アドレスすべてをフィルタリングしようとします。

ipaddr、*atalkaddr*、または *etheraddr*

アドレス表記、IP ドット式、AppleTalk ドット式、および Ethernet コロンが認識されます。たとえば、

- 「`172.16.40.13`」は、その IP のすべてのパケットに一致します。
- 「`2::9255:a00:20ff:fe73:6e35`」は、発信元または着信先としてこの IPv6 アドレスを持つすべてのパケットに一致します。
- 「`65281.13`」は、その AppleTalk アドレスのすべてのパケットに一致します。
- 「`8:0:20:f:b1:51`」は、発信元または着信先として Ethernet アドレスのすべてのパケットに一致します。

文字で始まる Ethernet アドレスは、ホスト名であると解釈さ

	れます。これを避けるため、アドレスを指定する場合は最初にゼロをつけます。たとえば、Ethernet アドレスが「aa:0:45:23:52:44」の場合、最初に 0 を追加して「0aa:0:45:23:52:44」のように指定します。
from または src	発信元アドレス、ポート、または RPC 応答にのみ一致する、後に続く host、net、*ipaddr*、*atalkaddr*、*etheraddr*、port または rpc プリミティブを変更する修飾子。
to または dst	着信先アドレス、ポート、または RPC 呼び出しにのみ一致する、後に続く host、net、*ipaddr*、*atalkaddr*、*etheraddr*、port または rpc プリミティブを変更する修飾子。
ether	名前を Ethernet アドレスに解決する、後に続く host プリミティブを変更する修飾子。通常、IP アドレスのマッチングが行われます。このオプションは、IPoIB (IP over InfiniBand) などのメディアではサポートされていません。
ethertype *number*	Ethernet タイプフィールドに値 *number* がある場合は true になります。「ether[12:2] = *number*」と同等です。
ip、ip6、arp、rarp、pppoed、pppoes	パケットが適切な ethertype の場合に true になります。
pppoe	パケットの ethertype が pppoed または pppoes のいずれかの場合に true になります。
broadcast	パケットがブロードキャストパケットの場合に true になります。Ethernet の「ether[2:4]

	= 0xffffffff 」と同等です。このオプションは、IPoIB (IP over InfiniBand) などのメディアではサポートされていません。
multicast	パケットがマルチキャストパケットの場合に true になります。Ethernet の「ether[0] & 1 = 1」と同等です。このオプションは、IPoIB (IP over InfiniBand) などのメディアではサポートされていません。
bootp、dhcp	パケットが、BOOTPS (67) の発信元ポートおよび BOOTPC (68) の着信先ポート、または BOOTPC (68) の発信元ポートおよび BOOTPS (67) の着信先ポートのいずれかで断片化されていない UDP パケットの場合に true になります。
apple	パケットが Apple Ethertalk パケットの場合に true になります。「ethertype 0x809b or ethertype 0x80f3」と同等です。
decnet	パケットが DECNET パケットの場合に true になります。
greater *length*	パケットが *length* よりも長い場合に true になります。
less *length*	パケットが *length* よりも短い場合に true になります。
udp、tcp、icmp、icmp6、ah、esp	IP または IPv6 プロトコルが適切なタイプの場合に true になります。
net *net*	IP 発信元アドレスまたは着信先アドレスのいずれかが *net* のネットワーク番号を持つ場合に true になります。from または to 修飾子を使用して、発信元アドレスまたは着信先アドレスの一方で当該ネットワー

	ク番号が使われているパケットを選択することもできます。
`port` *port*	発信元ポートまたは着信先ポートが *port* の場合に true になります。*port* は `/etc/services` からのポート番号または名前のいずれかです。 `tcp` または `udp` プリミティブを使用して、TCP または UDP ポートだけを選択することもできます。`from` または `to` 識別子を使用して、発信元または着信先としてだけ *port* が発生しているパケットを選択することもできます。
`rpc` *prog* [, *vers* [, `proc`]]	パケットが *prog* によって識別されるプロトコルの RPC 呼び出しまたは応答パケットである場合に true になります。*prog* は、`/etc/rpc` からの RPC プロトコルの名前またはプログラム番号のいずれかです。*vers* および `proc` を使用すると、プログラム *version* および *procedure* 番号をさらに修飾できます。たとえば、「`rpc nfs,2,0`」は NFS NULL 手続きのすべての呼び出しおよび応答を選択します。`to` または `from` 修飾子を使用すると、呼び出しパケットまたは応答パケットだけのいずれかを選択します。
`ldap`	パケットがポート 389 上の LDAP パケットの場合に true になります。
`gateway` *host*	パケットが *host* をゲートウェイとして使用した場合、つまり Ethernet 発信元アドレスまたは着信先アドレスが *host* であって IP アドレスではない場合に true

になります。「`ether host` *host* and not host *host*」と同等です。

`nofrag`

パケットが断片化されていない、または一連の IP フラグメントの最初である場合に true になります。「`ip[6:2] & 0x1fff = 0`」と同等です。

expr relop expr

relop が `>`、`<`、`>=`、`<=`、 `=`、`!=` のうちの 1 つであり、`expr` が数字、パケットフィールドセレクタ、`length` プリミティブ、および算術演算子 `+`、 `–`、 `*`、`&`、 `|`、 `^`、および `%` からなる演算式である関係が成立する場合に true になります。`expr` 内の算術演算子は関係演算子の前に評価され、算術演算子間では加算の前に乗算などの通常の優先順位規則が適用されます。括弧を使用して評価の順序を制御することもできます。パケット内のフィールドの値を使用するには、次の構文を使用します。

base`[expr [: size ] ]`

ここで `expr` は、`ether`、`ip`、`ip6`、`udp`、`tcp`、または `icmp` のいずれかである *base* オフセットからのパケットを、オフセットの値で評価します。`size` 値は、フィールドのサイズを指定します。指定しない場合は 1 とみなされます。その他の正当な値は、2 および 4 です。たとえば、

```
ether[0] & 1 = 1
```

は `multicast` と同等、

```
ether[2:4] = 0xffffffff
```

は `broadcast` と同等です。

`ip[ip[0] & 0xf * 4 : 2] = 2049`

は `udp[0:2] = 2049` と同等、

`ip[0] & 0xf > 5`

はオプション付きの IP パケットを選択します。

`ip[6:2] & 0x1fff = 0`

は IP フラグメントを除去します。

`udp and ip[6:2]&0x1fff = 0 and udp[6:2] != 0`

UDP チェックサムを持つすべてのパケットを検索します。

`length` プリミティブを使用して、パケットの長さを取得することもできます。たとえば、「`length > 60` 」は「`greater 60` 」と同等であり、「`ether[length − 1]`」はパケットの最後のバイトを取得します。

`and` — 2 つのブール型値の間の論理 AND 演算を実行します。2 つのブール型式を並べた場合は AND 演算を行うものとみなされます。たとえば「`dinky pinky`」は「`dinky AND pinky`」と同等です。

`or` または `,` — 2 つのブール型値の間の論理 OR 演算を実行します。代わりにコンマを使用することもでき、たとえば「`dinky,pinky`」は「`dinky OR pinky`」と同じです。

`not` または `!` — 後続のブール型値で論理 NOT 演算を実行します。この演算子は、AND または OR の前に評価されます。

slp	パケットが SLP パケットの場合に true になります。
sctp	パケットが SCTP パケットの場合に true になります。
ospf	パケットが OSPF パケットの場合に true になります。

使用例

例 1 snoop コマンドの使用

すべてのパケットを取得し、パケットを受信した順に表示します。

```
example# snoop
```

ホスト funky が発信元または着信先であるパケットを取得し、受信した順に表示します。

```
example# snoop funky
```

funky および pinky の間のパケットを取得し、ファイルに保存します。その後、最初に取得したパケットからの相対的な時間 (秒単位) を使用して、パケットを検査します。

```
example# snoop -o cap funky pinky
example# snoop -i cap -t r | more
```

別の取得ファイル内の選択したパケットを表示します。

```
example# snoop -i pkts -p 99,108
 99   0.0027   boutique -> sunroof     NFS C GETATTR FH=8E6
100   0.0046   sunroof -> boutique     NFS R GETATTR OK
101   0.0080   boutique -> sunroof NFS C RENAME FH=8E6C MTra00192 to .nfs08
102   0.0102   marmot -> viper        NFS C LOOKUP FH=561E screen.r.13.i386
103   0.0072   viper -> marmot       NFS R LOOKUP No such file or directory
104   0.0085   bugbomb -> sunroof    RLOGIN C PORT=1023 h
105   0.0005   kandinsky -> sparky    RSTAT C Get Statistics
106   0.0004   beeblebrox -> sunroof  NFS C GETATTR FH=0307
107   0.0021   sparky -> kandinsky    RSTAT R
108   0.0073   office -> jeremiah      NFS C READ FH=2584 at 40960 for 8192
```

パケット 101 をさらに詳細表示します。

```
example# snoop -i pkts -v -p101
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 101 arrived at 16:09:53.59
ETHER:  Packet size = 210 bytes
ETHER:  Destination = 8:0:20:1:3d:94, Sun
ETHER:  Source      = 8:0:69:1:5f:e,  Silicon Graphics
```

例 1 snoop コマンドの使用 *(続き)*

```
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4, header length = 20 bytes
IP:   Type of service = 00
IP:         ..0. .... = routine
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 196 bytes
IP:   Identification 19846
IP:   Flags = 0X
IP:   .0.. .... = may fragment
IP:   ..0. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 18DC
IP:   Source address = 172.16.40.222, boutique
IP:   Destination address = 172.16.40.200, sunroof
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 1023
UDP:  Destination port = 2049 (Sun RPC)
UDP:  Length = 176
UDP:  Checksum = 0
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 665905
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100003 (NFS), version = 2, procedure = 1
RPC:  Credentials: Flavor = 1 (Unix), len = 32 bytes
RPC:     Time = 06-Mar-90 07:26:58
RPC:     Hostname = boutique
RPC:     Uid = 0, Gid = 1
RPC:     Groups = 1
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
NFS:  ----- SUN NFS -----
NFS:
NFS:  Proc = 11 (Rename)
```

例 1 snoop コマンドの使用　(続き)

```
NFS:  File handle = 0000164300000001000800003O5A1C47
NFS:                597A0000000800002046314AFC450000
NFS:  File name = MTra00192
NFS:  File handle = 0000164300000001000800003O5A1C47
NFS:                597A0000000800002046314AFC450000
NFS:  File name = .nfs08
NFS:
```

sunroof と boutique の間の NFS パケットだけを表示するには、次のように入力します。

```
example# snoop -i pkts rpc nfs and sunroof and boutique
1   0.0000   boutique -> sunroof    NFS C GETATTR FH=8E6C
2   0.0046    sunroof -> boutique   NFS R GETATTR OK
3   0.0080   boutique -> sunroof   NFS C RENAME FH=8E6C MTra00192 to .nfs08
```

これらのパケットを新規取得ファイルに保存するには、次のように入力します。

```
example# snoop -i pkts -o pkts.nfs rpc nfs sunroof boutique
```

カプセル化されたパケットを表示するには、カプセル化を示す指示子があります。

```
example# snoop ip-in-ip
sunroof -> boutique ICMP Echo request    (1 encap)
```

-V をカプセル化されたパケットに使用した場合は、次のようになります。

```
example# snoop -V ip-in-ip
sunroof -> boutique  ETHER Type=0800 (IP), size = 118 bytes
sunroof -> boutique  IP D=172.16.40.222 S=172.16.40.200 LEN=104, ID=27497
sunroof -> boutique  IP  D=10.1.1.2 S=10.1.1.1 LEN=84, ID=27497
sunroof -> boutique  ICMP Echo request
```

例 2 より効率的なフィルタの設定

より効率的なフィルタを設定するには、次に示すフィルタは式の最後の方で使用し、式の最初の部分がカーネルに設定されるようにするべきです。greater、less、port、rpc、nofrag、および relop です。カーネルで設定できないプリミティブを使用する場合、OR が存在するとフィルタリングを分割するのが難しくなります。代わりに、括弧を使用して OR にするべきプリミティブを強制します。

funky と pinky の間のパケットで、port 80 上のタイプが tcp または udp のパケットを取得するには、次のように入力します。

```
example# snoop funky and pinky and port 80 and tcp or udp
```

例 2 より効率的なフィルタの設定 *(続き)*

プリミティブ port はカーネルフィルタによって処理できず、また式には OR も存在するため、OR を式の最後に移動し、tcp と udp の間で括弧を使用して OR を強制すると、より効率的にフィルタリングできます。

```
example# snoop funky and pinky and (tcp or udp) and port 80
```

終了ステータス

0 正常終了。

1 エラーが発生しました。

ファイル

/dev/audio	システムの主なオーディオデバイスへのシンボリックリンクです。
/dev/null	NULL ファイルです。
/etc/hosts	ホスト名データベースです。
/etc/rpc	RPC プログラム番号データベースです。
/etc/services	インターネットサービスとエイリアスです。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWrcmdc

関連項目

netstat(1M), hosts(4), rpc(4), services(4), attributes(5), audio(7I), bufmod(7M), dlpi(7P), pfmod(7M), tun(7M)

Callaghan, B.、 Gilligan, R. 著、 『RFC 1761, Snoop Version 2 Packet Capture File Format』、 Network Working Group 発行、 1995 年 2 月。

警告

リアルタイムにパケットを解釈するための処理オーバーヘッドは非常に高くなります。結果として、ドロップされるパケットの数も多くなる可能性があります。より信頼できる取得を行うために、生のパケットを -o オプションを使用してファイルに出力し、オフラインでパケットを分析してください。

フィルタリングしないパケットの取得は、特に取得したパケットをリアルタイムで解釈する場合、ホストコンピュータに大きな処理負荷がかかります。この処理負荷は、詳細表示オプションが使用されるとさらに増加します。snoop を頻繁に使用するとほかの処理に対してコンピューターリソースが提供されない可能性があるため、本稼働サーバーでは使用しないようにするべきです。snoop を頻繁に使用するのは、専用コンピュータに限定することをお勧めします。

snoop は IP フラグメントを再構成しません。高いレベルのプロトコルの解釈は、最初の IP フラグメントの最後で停止します。

snoop の副作用として、余分のパケットが生成される場合があります。たとえば、IP アドレスをホスト名に変換して表示するために、ネットワークネームサービス (NIS または NIS+) が使用される場合があります。ファイルに取得してあとで表示するようにすると、取得のセッションが完了するまでアドレスから名前のマッピングが延期されます。NFS でマウントされたファイルへの取得によっても余分のパケットが生成されます。

snaplen (-s オプション) に小さな値を設定すると、高いレベルのプロトコルの解釈に必要なヘッダー情報が削除される場合があります。切り取られる正確な値は、使用するネットワークとプロトコルによって異なります。10 Mbps Ethernet 上の UDP を使用する NFS バージョン 2 トラフィックの場合、snaplen を 150 バイト未満に設定しないでください。100 Mbps Ethernet 上の TCP を使用する NFS バージョン 3 トラフィックの場合、snaplen を 250 バイト以上に設定するべきです。

snoop は、RPC 応答を完全に解釈するために、RPC 要求からの情報を必要とします。取得ファイルまたはパケットレンジ内の RPC 応答に先行する要求がない場合、RPC 応答ヘッダーだけが表示されます。

名前　　su – スーパーユーザーまたは別のユーザーに変更

形式　　su [-] [*username* [*arg*]...]

機能説明　　su コマンドを使用すると、ログオフをしないで別のユーザーまたは別の役割になることができます。デフォルトの *username* は root (スーパーユーザー) です。

su を使用するときは、適切なパスワードを入力する必要があります (ユーザーがすでに root の場合は不要)。パスワードが正しければ、指定した *username* 用に設定されている実ユーザー ID、実効ユーザー ID、グループ ID、補助グループリストを持つ新しいシェルプロセスが生成されます。新しいシェルは、*username* のパスワードファイルエントリ (passwd(4) を参照) のシェルフィールドで指定されたシェルです。シェルが指定されていないと、/usr/bin/sh (sh(1) を参照) になります。スーパーユーザー権限が必要な場合に、exec(2) によってスーパーユーザーのシェルを起動できないときは、代わりに /sbin/sh が使用されます。元のユーザー ID の権限に戻る場合は、EOF 文字 (CTRL-D) を入力して新しいシェルを終了します。

コマンド行に引数を指定すると、その引数が新しいシェルに渡されます。sh などのプログラムを実行している場合は、*arg* (引数) に -c *string* と指定すると、シェルによって *string* が実行されます。また、引数に -r を指定すると、ユーザーに制限付きのシェルが与えられ提供されます。

ログイン環境を作成するために、コマンド「"su –"」は次の作業を実行します。

- すでに伝播されている環境変数に加えて、指定されたユーザーの環境から環境変数 LC* と LANG を伝播します。
- ユーザーの環境から環境変数 TZ を伝播します。環境変数 TZ がユーザーの環境に存在しない場合、su は、/etc/default/login にある TIMEZONE パラメータの TZ 値を使用します。
- MAIL を /var/mail/*new_user* に設定します。

su の最初の引数に – (ダッシュ) を指定すると、指定したユーザーとして実際にログインした場合と同じ環境が渡されます。最初の引数に – (ダッシュ) を指定しない場合、$PATH 以外の環境が渡されます。$PATH は /etc/default/su 中で PATH と SUPATH によって制御されます。また、引数に – (ダッシュ) を指定した場合は、ユーザーのプロジェクト ID が設定されます。settaskid(2) のマニュアルページを参照してください。

su を実行して別のユーザーになる操作は、すべてログファイル /var/adm/sulog に記録されます (sulog(4) を参照)。

セキュリティ　　su は、pam(3PAM) を使って、認証、アカウント管理、セッション管理を行います。su で使用するモジュールを指定する PAM 構成ポリシーは、/etc/pam.conf に記述されています。次の例は、su コマンド (UNIX 認証、アカウント管理、セッション管理モジュールを実行) エントリが記述されている pam.conf ファイルの抜粋を示します。

```
su   auth        requisite   pam_authtok_get.so.1
su   auth        required    pam_dhkeys.so.1
su   auth        required    pam_unix_auth.so.1

su   account     required    pam_unix_roles.so.1
su   account     required    pam_unix_projects.so.1
su   account     required    pam_unix_account.so.1

su   session     required    pam_unix_session.so.1
```

su サービスのエントリがない場合は other のサービスのエントリを使用します。複数の認証モジュールが記述されている場合、複数のパスワードが必要になることがあります。

使用例

例 1 以前にエクスポートした環境を維持してユーザー bin になる

以前にエクスポートした環境のままユーザー bin になるには、以下のコマンドを実行します。

```
example% su bin
```

例 2 ユーザー bin になり bin のログイン環境に変更する

ユーザー bin になり、bin がログインした場合と同じ環境に切り替えるには、以下のコマンドを実行します。

```
example% su - bin
```

例 3 ユーザー bin の環境とアクセス権でコマンドを実行する

ユーザー bin の一時的な環境とアクセス権を使ってコマンド *command* を実行する場合は、以下のように入力します。

```
example% su - bin -c "command args"
```

環境

セキュリティ上の理由から、接頭辞 LD_ が付いている環境変数は削除されました。このため、以前にユーザー bin でエクスポートした接頭辞 LD_ 付きの環境変数は、su bin を実行しても使用できません。

LC_* 変数 (LC_CTYPE、LC_MESSAGES、LC_TIME、LC_COLLATE、LC_NUMERIC、LC_MONETARY) (environ(5) 参照) が設定されていない環境では、環境変数 LANG によって、各ロケールカテゴリの su の動作が決定します。もし、LC_ALL が設定されていれば、その内容が LANG 変数やその他の LC_* 変数より優先されます。上記の変数がどれも設定されていなければ、C ロケール (米国の形式) によって su の動作が決定します。

LC_CTYPE　su が文字を処理する方法を決定します。LC_CTYPE に有効な値が設定されていると、そのロケールで有効な文字を含むテキストやファイル名を表示または処理できます。また、拡張 UNIX コード

(EUC) の表示または処理も可能です (1 ～ 3 バイト幅の文字を使用)。このほか、1 カラム幅、2 カラム幅、またはそれ以上のカラム幅の EUC 文字も処理できます。C ロケールでは、ISO 8859-1 の文字だけが有効です。

LC_MESSAGES　診断メッセージや情報メッセージの表示方法を決定します。また、メッセージの言語とスタイル、肯定・否定の応答形式も決定します。C ロケールでは、メッセージはプログラム自身が使用しているデフォルトの形 (通常、米語) で表示されます。

ファイル

$HOME/.profile　sh および ksh ユーザーのログインコマンド

/etc/passwd　システムのパスワードファイル

/etc/profile　システム全体の sh と ksh のログインコマンド

/var/adm/sulog　ログファイル

/etc/default/su　このファイル中に指定できるデフォルトパラメータは以下のとおりです。

- SULOG　定義されている場合、別のユーザーになるための su の試行はすべて、指定されたファイルに記録されます。
- CONSOLE　定義されている場合、root になるための su の試行はすべて、コンソールに記録されます。
- PATH　デフォルトパス (/usr/bin:)
- SUPATH　root になるために su を起動しているユーザーのデフォルトパス (/usr/sbin:/usr/bin)
- SYSLOG　すべての su 試行を記録するのに syslog(3C) の LOG_AUTH を使うかどうかを指定します。LOG_NOTICE メッセージは root への su に対して生成され、LOG_INFO メッセージはその他のユーザーへの su に対して生成されます。LOG_CRIT メッセージは、su に失敗した場合に生成されます。

/etc/default/login　このファイルのデフォルトのパラメータは、次のとおりです。

- TIMEZONE　シェルの TZ 環境変数を設定します。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値

使用条件	SUNWcsu

関連項目

csh(1), env(1), ksh(1), login(1), roles(1), sh(1), syslogd(1M), exec(2), getprojent(3PROJECT), setproject(3PROJECT), pam(3PAM), pam_authenticate(3PAM), pam_acct_mgmt(3PAM), pam_setcred(3PAM), pam.conf(4), passwd(4), profile(4), sulog(4), syslog(3C), attributes(5), environ(5)

名前　　svcadm – サービスインスタンスの操作

形式

```
/usr/sbin/svcadm [-v] enable [-rst] {FMRI | pattern}...

/usr/sbin/svcadm [-v] disable [-st] {FMRI | pattern}...

/usr/sbin/svcadm [-v] restart {FMRI | pattern}...

/usr/sbin/svcadm [-v] refresh {FMRI | pattern}...

/usr/sbin/svcadm [-v] clear {FMRI | pattern}...

/usr/sbin/svcadm [-v] mark [-It] instance_state
     {FMRI | pattern}...

/usr/sbin/svcadm [-v] milestone [-d] milestone_FMRI

/usr/sbin/svcadm [-v] restarter_FMRI {FMRI | pattern}...
```

機能説明　　svcadm は、サービス管理機能 (smf(5) を参照) 内で実行中のサービスに対するアクションの要求を発行します。サービスに対するアクションは、それに割り当てられているサービスリスタータエージェントによって実行されます。デフォルトのサービスリスタータは svc.startd です (svc.startd(1M) を参照)。

オプション　　次のオプションを指定できます。

-v　アクションを詳細形式で標準出力に出力します。

サブコマンド

一般的な操作　　次に示すサブコマンドは、サービスインスタンスの一般的な管理に使用されます。

1 つ以上のオペランドを取るサブコマンドの場合、オペランドがサービスインスタンスではなくサービスを指定しており、かつそのサービスのインスタンスが 1 つだけ存在するときは、svcadm はそのインスタンスを操作します。省略形の *FMRI* (障害管理リソース識別子) またはパターンが複数のサービスに一致する場合は、警告メッセージが表示され、そのオペランドは無視されます。

サービスに複数のインスタンスが存在する場合、svcadm はゼロ以外の終了ステータスを返します。

enable [-rst] {*FMRI* | *pattern*}...　　オペランドで指定されているサービスインスタンスを有効にします。割り当てられているリスタータは、各サービスインスタンスをオンライン状態にしようとします。このアクションには、サービスインスタンスの「general」プロパティーグループを変更するためのアクセス権が必要です (smf_security(5) を参照)。

-r オプションが指定されている場合、svcadm は各サービスインスタンスを有効にし、その依存関係を再帰的に有効にします。

-s オプションが指定されている場合、svcadm は各サービスインスタンスを有効にしたあと、各サービスインスタンスが online または degraded 状態になるまで待ちます。管理者の操作なしではサービスがこれらの状態に到達できないと判定した場合、svcadm は状態を待たずに戻ります。

-t オプションが指定されている場合、svcadm は各サービスインスタンスを一時的に有効にします。一時的な有効はリブートまでに限り継続します。このアクションには、サービスインスタンスの「restarter_actions」プロパティーグループを変更するためのアクセス権が必要です (smf_security(5) を参照)。デフォルトでは、enable はリブート後も持続します。

disable [-st] {*FMRI* | *pattern*}...

オペランドで指定されているサービスインスタンスを無効にします。割り当てられているリスタータは、各サービスインスタンスを無効状態にしようとします。このアクションには、サービスインスタンスの「general」プロパティーグループを変更するためのアクセス権が必要です (smf_security(5) を参照)。

-s オプションが指定されている場合、svcadm は各サービスインスタンスを無効にしたあと、各サービスインスタンスが無効状態になるまで待ちます。管理者の操作なしではサービスがこの状態に到達できないと判定した場合、svcadm は状態を待たずに戻ります。

-t オプションが指定されている場合、svcadm は各サービスインスタンスを一時的に無効にします。一時的な無効はリブートまでに限り継続します。このアクションには、サービスインスタンスの「restarter_actions」プロパティーグループを変更するためのアクセス権が必要です (smf_security(5) を参照)。デフォルトでは、disable はリブート後も持続します。

restart {*FMRI* | *pattern*}...

オペランドで指定されているサービスインスタンスの再起動を要求します。このアクションには、サービスインスタンス
の「restarter_actions」プロパティーグループを

変更するためのアクセス権が必要です (smf_security(5) を参照)。

`refresh` {*FMRI* | *pattern*}...

オペランドで指定されている各サービスインスタンスについて、サービスの実行中の構成のスナップショットを現在の構成の値で更新するよう、割り当てられているリスタータに要求します。これらの値の一部はすぐに有効になります (たとえば、依存関係の変更など)。その他の値は、次回のサービスの `restart` まで有効になりません。詳細については、リスタータおよびサービスのマニュアルを参照してください。

サービスが svc.startd(1M) によって管理されている場合は、`refresh` メソッドが存在していればそれが呼び出され、サービスに自身の構成を読み込み直すように要求します。その他のリスタータについては、リスタータのマニュアルを参照してください。

このアクションには、サービスインスタンスの「restarter_actions」プロパティーグループを変更するためのアクセス権が必要です (smf_security(5) を参照)。

`clear` {*FMRI* | *pattern*}...

オペランドで指定されている各サービスインスタンスについて、インスタンスが `maintenance` 状態の場合は、割り当てられているリスタータにサービスが修復されたことを通知します。インスタンスが `degraded` 状態の場合は、サービスを `online` 状態にするよう、割り当てられているリスタータに要求します。このアクションには、サービスインスタンスの「restarter_actions」プロパティーグループを変更するためのアクセス権が必要です (smf_security(5) を参照)。

特別な操作

次に示すサブコマンドは、サービスの開発および一時的な管理操作に使用されます。

`mark [-It]` *instance_state* {*FMRI* | *pattern*}...

instance_state が「maintenance」の場合、svcadm は、オペランドで指定されている各サービスについて、サービスを `maintenance` 状態にするよう、割り当てられているリスタータに要求します。各リスタータに対して実行されるアクションの詳細については、

svc.startd(1M) および inetd(1M) のマニュアルページを参照してください。

instance_state が「degraded」の場合、`svcadm` は、オペランドで指定されているサービスのうちオンライン状態のものについて、サービスを `degraded` 状態にするよう、サービスに割り当てられているリスタータに要求します。

`-I` オプションが指定されている場合、要求には「即時 (immediate)」としてフラグが設定されます。

`-t` オプションは保守要求にのみ有効です。このオプションが指定されている場合、要求には「一時 (temporary)」としてフラグが設定され、その効果は次回のリブートまでに限り継続します。

`milestone` [`-d`] *milestone_FMRI*

milestone_FMRI がキーワード「none」の場合は、マスターリスタータ `svc:/system/svc/restarter:default` を除くすべてのサービスが一時的に無効になります。

milestone_FMRI がキーワード「all」の場合は、すべてのサービスについて一時的な enable 要求および disable 要求が無効化されます。

milestone_FMRI が次のいずれかの場合:

```
svc:/milestone/single-user:default
svc:/milestone/multi-user:default
svc:/milestone/multi-user-server:default
```

指定されたサービスおよびそれが (直接または間接的に) 依存しているすべてのサービスについて、一時的な enable 要求および disable 要求が無効化されます。その他のサービスはすべて一時的に無効になります。

システムの現在のマイルストーンを milestone サブコマンドで変更する場

合、システムの現在の実行レベルは変更されません。システムの実行レベルを変更するには、`/sbin/init` を直接呼び出します。

このアクションには、`svc:/system/svc/restarter:default` サービスインスタンスの「options_ovr」プロパティーグループを変更するためのアクセス権が必要です (`smf_security`(5) を参照)。

`-d` オプションが指定されている場合は、要求されたマイルストーンへの変更がすぐに、上記のとおり実行されます。また、指定されたマイルストーンはデフォルトのブートマイルストーンとなり、リブート後も持続します。デフォルトのマイルストーンは、マスターリスタータ `svc:/system/svc/restarter:default` の `options/milestone` プロパティーで定義されます。このプロパティーが存在しない場合、デフォルトは「all」です。このアクションには、`svc:/system/svc/restarter:default` サービスインスタンスの「options」プロパティーグループを変更するためのアクセス権が必要です (`smf_security`(5) を参照)。

オペランド　次のオペランドを指定できます。

FMRI　1 つまたは複数のインスタンスを指定する *FMRI*。インスタンス名を指定するか、サービス名の連続する部分を指定することにより、*FMRI* を省略形にすることもできます。たとえば、次のような *FMRI* を仮定します。

```
svc:/network/smtp:sendmail
```

この場合、次に示す省略形はすべて有効です。

```
sendmail
:sendmail
smtp
```

```
smtp:sendmail
network/smtp
```

これに対し、次のような省略形は無効です。

```
mail
network
network/smt
```

FMRI がサービスを指定している場合、コマンドはそのサービスのすべてのインスタンスに適用されます。*FMRI* の省略形は不安定なので、スクリプトやその他の長期に渡って使用するツールには使用しないでください。

pattern `fnmatch`(5) で説明されている展開規則に従ってサービスインスタンスの *FMRI* と照合されるパターン。このパターンが「svc:」で始まっていない場合は、「svc:/」が先頭に付加されます。

省略形の *FMRI* またはパターンが複数のサービスに一致する場合は、警告メッセージが表示され、そのオペランドは無視されます。

使用例

例 1 サービスインスタンスの再起動

次のコマンドは NFS サーバーを再起動します。デフォルトのサービスインスタンスの完全な *FMRI* は次のとおりです。`svc:/network/nfs/server:default`

ただし、完全な *FMRI* を次のような省略形にすることもできます。

```
# svcadm restart nfs/server
```

例 2 標準 HTTP サーバーの無効化

次のコマンドは、省略形の *FMRI* を使用して標準 HTTP サーバーを無効にします。

```
$ svcadm disable http
```

例 3 インスタンスおよびその依存インスタンスの有効化

次のコマンドは、`foo:bar` インスタンスおよびそれが依存しているすべてのインスタンスを有効にします。

```
$ svcadm enable -r foo:bar
```

例4 インスタンスの同期的な有効化

次のコマンドは foo:bar インスタンスを有効にします。インスタンスがオンラインになるまで、あるいはサービスがオンライン状態に到達できないと svcadm が判定するまで、コマンドは戻りません。

```
$ svcadm enable -s foo:bar
```

例5 実行中のサービスの制限と復元

次のコマンドは、実行中のサービスをシングルユーザーモードに制限します。

```
# svcadm milestone milestone/single-user
```

次のコマンドは、実行中のサービスを復元します。

```
# svcadm milestone all
```

終了ステータス

次の終了ステータスが返されます。

0 正常終了。

1 致命的なエラーが発生した。1 つ以上のエラーメッセージが標準エラーに表示されます。

2 無効なコマンド行オプションが指定された。

3 svcadm は、サービスインスタンス自体の問題により、待機中のサービスインスタンスが管理者の操作なしでは目的の状態に到達できないと判定しました。

4 svcadm は、サービスの依存関係の問題により、待機中のサービスインスタンスが管理者の操作なしでは目的の状態に到達できないと判定しました。

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	下記を参照

対話型出力は不安定です。呼び出しおよび非対話型出力は開発中です。

関連項目

svcprop(1), svcs(1), inetd(1M), init(1M), svccfg(1M), svc.startd(1M), libscf(3LIB), contract(4), attributes(5), smf(5), smf_security(5)

注意事項

svcadm がサービスおよびその依存関係の状態の変化を待機する時間は、それらのメソッドのタイムアウトによって暗黙に制限されます。たとえば、サービスがデフォルトのリスタータを使用している場合、その start メソッドがハングアップすると、タイムアウトの期限が切れたときサービスは保守状態に移行します。すると svcadm は、管理者の操作なしではこのサービスがオンライン状態に到達できないと判定します。

ファイルに (直接または間接的に) 依存しているサービスを同期的に有効にしようとすると、そのファイルの入っているディレクトリを検索するために必要な特権を呼び出し元が持っていない場合は、依存関係が満たされていないことを示す終了ステータスで失敗する場合があります。この制限事項は、Solaris の将来のリリースでは取り除かれる可能性があります。

名前　svccfg – サービス構成のインポート、エクスポート、および変更

形式

```
/usr/sbin/svccfg  [-v] [-s FMRI]

/usr/sbin/svccfg  [-v] [-s FMRI] subcommand [args]...

/usr/sbin/svccfg  [-v] [-s FMRI] -f command-file
```

機能説明　`svccfg` コマンドは、サービス構成リポジトリ内のデータを操作します。`svccfg` は、個別のサブコマンドを使用するか、一連のサブコマンドが格納されているコマンドファイルを指定することで、対話形式で呼び出すことができます。

通常、リポジトリ内の既存のサービスに加えた変更は、次にサービスインスタンスが更新されるまでサービスに反映されません。詳細については、svcadm(1M) マニュアルページの `refresh` サブコマンドを参照してください。

オプション　サポートしているオプションは、次のとおりです。

`-f` *command-file*　`svccfg` サブコマンドを *command-file* から読み込んで実行します。

`-s` *FMRI*　サブコマンドを実行する前に、*FMRI* (障害管理リソース識別子) で指定されているエンティティーを選択します。

`-v`　詳細表示。

サブコマンド　サポートされているサブコマンドは次のとおりです。

end
exit
quit　すぐに終了します。

set [`-v`|`-V`]　オプション動作を設定します。オプションが何も指定されていない場合、set は現在有効になっているオプションを表示します。

`-v`　詳細表示モードを有効にします。

`-V`　詳細表示モードを無効にします。

repository *repfile*　*repfile* を非公開リポジトリとして使用します。デフォルトでは、`svccfg`(1M) はシステムリポジトリを使用します。

サービスプロファイルのサブコマンド

apply *file*　*file* がサービスプロファイルの場合は、ファイル内で指定されているサービスインスタンスが、指定に従って有効または無効になります。サービスプロファイルについては、`smf`(5) を参照してください。このコマンドには、サービスインスタンスの「general/enabled」プロパティーを変更するための特権が必要です。プロパティーを変更するために必要な特権について

は、smf_security(5) を参照してください。*file* がサービスプロファイルでない場合、このサブコマンドは失敗します。

extract [>*file*] リポジトリ内のサービスインスタンスの有効化状態を表すサービスプロファイルを、標準出力に出力します。出力はファイルにリダイレクトされる場合があります。

サービスマニフェストのサブコマンド

import *file* *file* がサービスマニフェストの場合は、それによって指定されているサービスおよびインスタンスがリポジトリにインポートされます。ファイルに従って、ほかのサービスに依存関係が作成される場合があります。サービスマニフェストについては、smf(5) を参照してください。サービス構成を作成および変更するために必要な特権については、smf_security(5) を参照してください。

既存のサービスおよびインスタンスについては、最後に import スナップショットが作成された以降に変更されていないプロパティーは、マニフェストで指定されているものにアップグレードされます。競合 (リポジトリとマニフェストの両方で変更されたプロパティー) は、標準エラーストリームに報告されます。「general/enabled」プロパティーと「general/restarter」プロパティーは、管理者によって選択された設定を表すので、svccfg でアップグレードされることはありません。

archive リポジトリ内のすべてのサービス、インスタンス、およびそれらの持続的プロパティーに関する完全なサービス説明を XML でダンプします。これには、サービス状態などの一時的なプロパティーは含まれません。このダンプは、再配置可能なリポジトリバックアップとして適しています。

validate *file* *file* は import と同様の方法で処理されますが、リポジトリは変更されません。エラーが検出された場合、svccfg(1M) はゼロ以外の終了ステータスで終了します。

export *service_FMRI* [>*file*] 指定されたサービスおよびそのインスタンスに関するサービス説明が、標準出力に書き込まれるか、指定されたファイルにリダイレクトされます。ブール型

の「external」プロパティーが true に設定されている依存関係は、別のサービスのために作成されたものと見なされ、除外されます。

inventory *file*
: *file* がサービスマニフェストであると判定された場合は、*file* で指定されているサービスおよびインスタンスの FMRI が出力されます。各サービスについて、そのインスタンスの FMRI に続いてサービスの FMRI が表示されます。

エンティティーの選択、変更、およびナビゲーションのサブコマンド

「エンティティー」とは、スコープ、サービス、またはサービスインスタンスを指します。

select {*name* | *fmri*}
: 引数で現在の選択内容の子が指定されている場合は、それが現在の選択内容になります。それ以外の場合、引数は FMRI であると解釈され、引数で指定されているエンティティーが現在の選択内容になります。

unselect
: 現在の選択内容の親が現在の選択内容になります。

list [*pattern*]
: 現在の選択内容の子エンティティーのうち、展開パターン *pattern* に一致する名前を持つものが表示されます (`fnmatch`(5) を参照)。プロパティーを持つエンティティー、すなわちサービスおよびサービスインスタンスには、「:properties」も一覧表示されます。

add *name*
: 現在の選択内容の子として、指定された名前で新しいエンティティーが作成されます。エンティティーを作成するために必要な特権については、`smf_security`(5) を参照してください。

delete [`-f`] {*name* | *fmri*}
: *name* で指定されている現在の選択内容の子、または *fmri* で指定されているエンティティーが削除されます。`-f` フラグを指定しない限り、「online」状態または「degraded」状態のサービスインスタンスを削除しようとすると失敗します。サービスまたはサービスインスタンスが「framework」タイプの「dependents」プロパティーグループを持っている場合は、その中の「astring」タイプまたは「fmri」タイプの各プロパティーが調べられ、そのプロパティーがサービスまたはサービスインスタンスの名前を指定する 1 つの値を持っているときは、指定されたサービスまたはサービスインスタンスの、プロパティーと同じ名前を持つ dependency プロパティーグループが削除されます。サービス構成を削除するために必要な特権について

は、smf_security(5) を参照してください。

プロパティーの検査および変更のサブコマンド

listpg [*pattern*]	現在の選択内容のプロパティーグループの名前、タイプ、およびフラグを表示します。引数が指定されている場合、それは展開パターンと見なされ、引数に一致する名前を持つプロパティーグループだけが一覧表示されます。
addpg *name type* [*flags*]	指定された名前 *name* と *type* で、プロパティーグループを現在の選択内容に追加します。*flags* は、プロパティーグループの作成に使用するフラグを指定する文字列です。「P」は SCF_PG_FLAG_NONPERSISTENT を表します (scf_service_add_pg(3SCF) を参照)。プロパティーグループを作成するために必要な特権については、smf_security(5) を参照してください。
delpg *name*	現在の選択内容のプロパティーグループ *name* を削除します。プロパティーグループを削除するために必要な特権については、smf_security(5) を参照してください。
listprop [*pattern*]	現在の選択内容のプロパティーグループとプロパティーを一覧表示します。プロパティーグループの場合は、名前、タイプ、およびフラグが一覧表示されます。プロパティーの場合は、名前 (プロパティーグループ名とスラッシュ (/) が前に付加される)、タイプ、および値が一覧表示されます。使用可能なプロパティータイプのリストについては、scf_value_create(3SCF) を参照してください。引数が指定され

	ている場合、それは展開パターンと見なされ、引数に一致する名前を持つプロパティーグループおよびプロパティーだけが一覧表示されます。
setprop *pg/name* = [*type:*] *value* setprop *pg/name* = [*type:*] ([*values ...*])	現在の選択内容の *pg* プロパティーグループの *name* プロパティーを、タイプ *type* の指定された値に設定します。使用可能なプロパティータイプのリストについては、`scf_value_create`(3SCF) を参照してください。プロパティーがすでに存在している場合は、そのプロパティーに既存の *type* が *type* と一致しないと、このサブコマンドは失敗します。値は二重引用符で囲まれる場合があります。二重引用符やバックスラッシュを含んでいる文字列値は、二重引用符で囲む必要があります。また、文字列値に含まれている二重引用符とバックスラッシュは、バックスラッシュでエスケープする必要があります。*name* で指定されたプロパティーが存在しない場合は、*type* が指定されていればプロパティーが作成されます。プロパティーを作成または変更するために必要な特権については、`smf_security`(5) を参照してください。
delprop *pg*[*/name*]	現在の選択内容のプロパティーグループまたはプロパティーのうち、*name* で指定されているものを削除します。プロパティーを削除するために必要な特権については、`smf_security`(5) を参照してください。
editprop	現在の選択内容のプロパティーグループおよびプロパティーを再生成するためのコマンドのコメント

が一時ファイルに格納され、それを編集するために EDITOR 環境変数で指定されているプログラムが呼び出されます。完了すると、一時ファイル内のコマンドが実行されます。デフォルトのエディタは vi(1) です。プロパティーを作成、変更、または削除するために必要な特権については、smf_security(5) を参照してください。

addpropvalue *pg/name* [*type:*] *value*

指定された値をプロパティーに追加します。*type* が指定されている場合で、プロパティーが存在しているときは、そのプロパティーのタイプが *type* と一致しないと、このサブコマンドは失敗します。値は二重引用符で囲まれる場合があります。二重引用符やバックスラッシュを含んでいる文字列値は、二重引用符で囲む必要があります。また、文字列値に含まれている二重引用符とバックスラッシュは、バックスラッシュでエスケープする必要があります。存在していないプロパティーは作成されますが、その場合は *type* 指定子が指定されている必要があります。使用可能なプロパティータイプのリストについては、scf_value_create(3SCF) を参照してください。プロパティーを変更するために必要な特権については、smf_security(5) を参照してください。

delpropvalue *pg/name globpattern*

name で指定されたプロパティーの値のうち、指定された *globpattern* に一致するものをすべて削除します。一致する値がない場合でも、このサブコマンドは成功します。プロパティーを変更するために必

要な特権については、smf_security(5) を参照してください。

setenv [-i | -s] [-m *method_name*] *envvar value*

method_name プロパティーグループのタイプが「method」の場合は、そのプロパティーグループの「environment」プロパティーを変更することにより、サービスまたはインスタンスのメソッドの環境変数を設定します。*method_name* が指定されていない場合で、-i オプションが使用されているときは、現在インスタンスが選択されていれば「method_context」プロパティーグループが使用されます。-s オプションが使用されている場合で、現在サービスが選択されているときは、その「method_context」プロパティーグループが使用されます。-s オプションが使用されている場合で、現在インスタンスが選択されているときは、その親の「method_context」プロパティーグループが使用されます。-i オプションと -s オプションのどちらも使用されていない場合は、現在選択されているエンティティー内で「start」プロパティーグループが検索されます。また、現在インスタンスが選択されている場合は、その親も検索されます。「inetd_start」プロパティーグループが見つからない場合は、同様の方法で検索されます。

プロパティーが見つかると、「*envvar*=」で始まる値がすべて削除され、値「*envvar=value*」が追加されます。プロパティーを変更す

るために必要な特権については、`smf_security`(5) を参照してください。

unsetenv [`-i` | `-s`] [`-m` *method_name*] *envvar value*

method_name プロパティーグループのタイプが「method」の場合は、そのプロパティーグループの「environment」プロパティーを変更することにより、サービスまたはインスタンスのメソッドの環境変数を削除します。*method_name* が指定されていない場合で、`-i` オプションが使用されているときは、現在インスタンスが選択されていれば「method_context」プロパティーグループが使用されます。`-s` オプションが使用されている場合で、現在サービスが選択されているときは、その「method_context」プロパティーグループが使用されます。`-s` オプションが使用されている場合で、現在インスタンスが選択されているときは、その親の「method_context」プロパティーグループが使用されます。`-i` オプションと `-s` オプションのどちらも使用されていない場合は、現在選択されているエンティティー内で「start」プロパティーグループが検索されます。また、現在インスタンスが選択されている場合は、その親も検索されます。「inetd_start」プロパティーグループが見つからない場合は、同様の方法で検索されます。

プロパティーが見つかると、「*envvar*=」で始まる値がすべて削除されます。プロパティーを変更するために必要な特権については、`smf_security`(5) を参照してく

ださい。

スナップショットのナビゲーションおよび選択のサブコマンド

listsnap　現在選択されているインスタンスで使用可能なスナップショットを表示します。

selectsnap [*name*]　現在のスナップショットを、*name* で指定されたスナップショットに変更します。*name* が指定されていない場合は、現在選択されているスナップショットを選択解除します。スナップショットは読み取り専用です。

revert [*snapshot*]　現在選択されているインスタンスとそのサービスのプロパティーを、指定されたスナップショットに記録されているプロパティーに戻します。引数が何も指定されていない場合は、現在選択されているスナップショットを使用し、成功時にそれを選択解除します。変更したプロパティー値は、svcadm(1M) の `refresh` サブコマンドを使用してアクティブにすることができます。プロパティーを変更するために必要な特権については、`smf_security`(5) を参照してください。

FMRI を受け入れるコマンドはすべて、省略形や展開パターンも受け入れます。インスタンス名を指定するか、サービス名の連続する部分を指定することにより、インスタンスおよびサービスを省略形にすることもできます。たとえば、次のような *FMRI* を仮定します。

```
svc:/network/smtp:sendmail
```

この場合、次に示す省略形はすべて有効です。

```
sendmail
:sendmail
smtp
smtp:sendmail
network/smtp
```

これに対し、次のような省略形は無効です。

```
mail
network
network/smt
```

FMRI の省略形は不安定なので、スクリプトやその他の長期に渡って使用するツールには使用しないでください。パターンが複数のインスタンスまたはサービスに一致する場合は、エラーメッセージが表示され、処理は行われません。

使用例

例 1 サービス説明のエクスポート

サービス説明をローカルシステムにエクスポートするには、次のように指定します。

```
$ svccfg export dumpadm >/tmp/dump.xml
```

例 2 サービスインスタンスの削除

サービスインスタンスを削除するには、次のように指定します。

```
$ svccfg delete network/inetd-upgrade:default
```

例 3 サービス説明のインポート

サービス説明を非公開リポジトリに対話形式でインポートするには、次のように指定します。

```
$ svccfg
       svc:> repository /tmp/repository
       svc:> import /home/hjs/svc/box-factory.xml
       svc:> end
```

例 4 start メソッドの変更

start メソッドの LD_PRELOAD を変更し、デバッグ機能で libumem(3LIB) の使用を有効にするには、次のように指定します。

```
$ svccfg -s system/service setenv LD_PRELOAD libumem.so
$ svccfg -s system/service setenv UMEM_DEBUG default
```

環境変数

EDITOR　editprop サブコマンドが使用されたときに実行するコマンド。デフォルトのエディタは vi(1) です。

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

1　1 つまたは複数のサブコマンドが失敗した。エラーメッセージが標準エラーストリームに書き込まれます。

2　無効なコマンド行オプションが指定された。

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
インタフェースの安定性	下記を参照

対話型出力は不安定です。呼び出しおよび非対話型出力は開発中です。

関連項目

svcprop(1), svcs(1), svcadm(1M), svc.configd(1M), libscf(3LIB), libumem(3LIB), scf_service_add_pg(3SCF), scf_value_create(3SCF), contract(4), attributes(5), fnmatch(5), smf(5), smf_method(5), smf_security(5)

名前

svc.startd – サービス管理機能のマスターリスタータ

形式

```
/lib/svc/bin/svc.startd

svc:/system/svc/restarter:default
```

機能説明

svc.startd はサービス管理機能 (SMF) のマスターリスタータデーモンであり、すべてのサービスのデフォルトリスタータです。svc.startd は、管理要求、システム障害、またはアプリケーション障害に基づいて、サービスの開始、停止、および再起動を行います。

svc.startd は、各サービスの依存関係に基づいて障害管理を担うほか、サービスの状態の維持も行います。

svc.startd はシステムの起動時に自動的に呼び出されます。なんらかの障害が発生した場合は再起動されます。直接 svc.startd を呼び出さないでください。

すべてのリスタータに共通する構成および動作の詳細については、smf_restarter(5) を参照してください。

svcs(1) は、サービス構成機能によって管理されているすべてのサービスについて状態を報告します。svcadm(1M) を使用すると、サービスのリスタータに関連してサービスインスタンスを操作できます。

環境変数

「SMF_」という接頭辞を持つ環境変数は予約済みであり、上書きが可能です。

svc.startd は、smf_method(5) で指定されている「SMF_」環境変数をメソッドに渡します。デフォルトでは、PATH は /usr/sbin:/usr/bin に設定されます。デフォルトでは、svc.startd に渡されるその他すべての環境変数は、init(1M) から継承されたものです。

重複するエントリは削減されて単一のエントリになります。使用される値は不定です。「<名前>=」という接頭辞を持たない環境エントリは無視されます。

リスタータオプション

svc.startd の構成はコマンド行オプションでは設定されません。代わりに、サービス構成リポジトリから構成が読み込まれます。svccfg(1M) を使用すると、すべてのオプションおよびプロパティーを設定できます。

次に示す options プロパティーグループ内の構成変数は、開発者および管理者が使用できます。

boot_messages

astring は、ブート中にコンソールに出力するメッセージのデフォルトレベルを示します (scf_value_is_type で定義される。scf_value_create(3SCF) を参照)。サポートされているメッセージオプションには、quiet と verbose があります。quiet オプションは、ブート中に最小限のメッセージをコンソールに出力します。verbose オプションは、開始されるサービスごとに 1 つのメッセージを出力して、成功また

は失敗を示します。boot -m オプションを使用すると、ブート時に boot_messages 設定を上書きできます。kernel(1M) のマニュアルページを参照してください。

logging

svc.startd のグローバルサービスログ記録レベルを制御します。*astring* (scf_value_is_type で定義される。scf_value_create(3SCF) を参照) は、syslog (syslog(3C)) および svc.startd のグローバルログファイル /var/svc/log/svc.startd.log に記録するメッセージのデフォルトレベルを示します。サポートされているメッセージオプションには、quiet、verbose、および debug があります。quiet オプションは、管理者の介入を必要とするエラーメッセージを、コンソール、syslog、および svc.startd のグローバルログファイルに送信します。verbose オプションは、管理者の介入を必要とするエラーメッセージをコンソール、syslog、および svc.startd のグローバルログファイルに、管理者の介入を必要としないエラーに関する情報を svc.startd のグローバルログファイルに、それぞれ送信します。さらに、サービスが起動されるたびに単一のメッセージがコンソールに送信されます。debug オプションは、svc.startd デバッグメッセージを svc.startd のグローバルログファイルに、管理者の介入を必要とするエラーメッセージをコンソール、syslog、および svc.startd のグローバルログファイルに、サービスが起動されるたびに単一のメッセージをコンソールに、それぞれ送信します。

milestone

デフォルトのブートレベルとして使用するマイルストーンを決定する FMRI です。指定できるオプションは、次に示すメジャーマイルストーンです。

svc:/milestone/single-user:default
svc:/milestone/multi-user:default
svc:/milestone/multi-user-server:default

あるいは特殊値 all または none も指定できます。all は、すべてのサービスに依存する理想化されたマイルストーンを表します。none は、マスター svc:/system/svc/restarter:default 以外にはどのサービスも実行されていない、特殊なマイルストーンです。デフォルトでは、svc.startd は、すべてのサービスに依存する統合的なマイルストーン all を使用します。このプロパティーが指定された場合、inittab(4) の initdefault 設定はすべて上書きされます。

`system/reconfigure` 再構成リブートが要求されたことを示します。再構成リブートに応じて動作する必要のあるサービスでは、このプロパティーが存在し 1 に設定されているかどうかをチェックして、再構成ブートが要求されたかどうかを確認できます。

このプロパティーは `svc.startd` によって管理されます。管理者はこのプロパティーを変更しないでください。

`svc.startd` を無効にするなどの構成エラーは、`syslog` によってログに記録されますが、無視されます。

サービスの状態 `svc.startd` によって管理されるサービスは、`smf`(5) で説明されている状態のいずれかになります。このリスタータによって状態の定義が変更されることはありません。

サービスの報告機能 管理対象サービスによって実行されるログ記録のほかに、`svc.startd` には、サービスの報告およびログ記録を行う共通の機構が用意されています。

報告機能プロパティー `svc.startd` は、管理しているすべてのサービスについて、共通のプロパティーセットを更新します。これらのプロパティーは、サービスインスタンスの状態に基づいてアクションを実行するための共通インタフェースとして使用できます。`svcs`(1) コマンドを使用すると、これらのプロパティーを簡単に表示できます。

`restarter/state`
`restarter/next_state` インスタンスの現在の状態および次の状態 (移行中の場合)。

`restarter/auxiliary_state` 現在のインスタンス状態について詳細情報を示す説明文。`svc.startd` によって管理されるサービスには、次に示す補助状態を使用できます。

maintenance

fault_threshold_reached
stop_method_failed
administrative_request

`restarter/state_timestamp` 現在の状態に到達した時刻。

`restarter/contract` サービスインスタンスを実行している主プロセス契約 ID (存在する場合)。

ログ

デフォルトでは、`svc.startd` は、メソッドの標準出力および標準エラーのファイル記述子に加えて、サービスに対する重要なリスタータアクションを

/var/svc/log/*service*:*instance*.log に記録します。/var/svc/log/svc.startd.log や syslog など、システムのグローバルな場所へのロギングレベルは、options/logging プロパティーによって制御されます。

サービスの定義

svc.startd によって管理されるサービスを開発または構成するとき、サービスインスタンスとリスタータの相互作用を操作するために、一連の共通プロパティーが使用されます。

メソッド

svc.startd で提供されている fork/exec モデル用メソッドの一般的な形式は、smf_method(5) に示されています。svc.startd によって管理されるサービスでは、次に示すメソッドが必須メソッドまたは省略可能メソッドとしてサポートされています。

refresh　サービスを中断することなく、リポジトリまたは config ファイルから適切な構成パラメータを読み込み直します。多くの場合、これはシステムデーモン用の SIGHUP を使用して実装されます。再起動しないとサービスが構成の変更を認識できない場合、refresh メソッドは提供されません。

このメソッドは省略可能です。

start　サービスを開始します。アプリケーションがコンシューマに対して利用可能になったときに初めて成功を返します。競合するインスタンスがすでに実行されている場合や、サービスを開始できない場合は、失敗を返します。

このメソッドは必須です。

stop　サービスを停止します。場合によっては、一部またはすべてのサービスがすでに停止されているときに stop メソッドを呼び出すこともできます。メソッドが戻るときにサービスが完全に停止していない場合だけ、エラーを返します。

このメソッドは必須です。

サービスが必須メソッド内で何もアクションを実行する必要がない場合、そのメソッドに :true トークンを指定する必要があります。

svc.startd は、サービスや特定のメソッドに対して指定されたメソッドコンテキストをすべて尊重します。smf_method(5) で説明されているメソッド拡張トークンは、svc.startd によって呼び出されるすべてのメソッドで使用できます。

プロパティー

smf(5) で使用できる一般的なプロパティーの概要。これらの一般的なプロパティーは、次のような特定の方法で svc.startd と対話します。

general/enabled　enabled が true に設定されている場合、リスタータは、サービスの依存関係がすべて満たされるとサービスの開始を試みます。false に設定されている場合、サービスは実行されず、無効にされた状態のままになります。

general/restarter　この FMRI プロパティーが空または svc:/system/svc/restarter:default に設定されている場合、サービスは svc.startd によって管理されます。それ以外の場合は、指定されたリスタータが (利用可能になったときに) サービスの管理を担当します。

general/single_instance　single_instance が true に設定されている場合、svc.startd は、常にこのサービスの 1 つのインスタンスのみ、オンラインまたは機能低下状態に移行することを許可します。

また、svc.startd によって管理されるサービスは、下記に示す省略可能プロパティーを startd プロパティーグループ内に定義できます。

startd/duration　duration プロパティーは、サービスのモデルを定義します。このプロパティーは、transient、wait とも呼ばれる child モデルサービス、または contract (デフォルト) に設定できます。

startd/ignore_error　ignore_error プロパティーは、設定されている場合、無視するイベントをコンマ区切りのリストで指定します。このリストで有効な文字列値は core と signal です。デフォルトでは、すべてのエラーで再起動が行われます。

startd/need_session　need_session プロパティーが true に設定されている場合は、インスタンスを独自のセッションで起動する必要があることを示します。デフォルトでは、そのようにはなりません。

startd/utmpx_prefix　utmpx_prefix 文字列プロパティーは、メソッドを実行する前にインスタンスに有効な utmpx エントリが必要であることを定義します。デフォルトでは、utmpx エントリは作成されません。

サービス障害　メソッドからゼロ以外の終了コードが返されると、svc.startd はメソッドが失敗したと見なします。$SMF_EXIT_ERR_CONFIG または $SMF_EXIT_ERR_FATAL が返された場合、svc.startd はすぐにサービスを保守状態にします。その他すべての障害では、

svc.startd はサービスをオフライン状態にします。サービスがオフラインの場合で、その依存関係が満たされているときは、svc.startd は再びサービスの開始を試みます (smf(5) を参照)。

メソッドの失敗が 3 回連続して発生した場合や、サービスの再起動が 1 秒あたり 1 回以上発生している場合は、svc.startd はサービスを保守状態にします。

サービス障害の条件は、startd/duration プロパティーで定義されるサービスモデルと、startd/ignore_error プロパティーの値との組み合わせによって定義されます。

次に示す条件のいずれかが発生すると、contract モデルサービスは障害状態になります。

- サービスのすべてのプロセスが終了する
- サービスのプロセスのいずれかでコアダンプが生成される
- サービス外部のプロセスからサービスプロセスに致命的な信号が送信される (たとえば、管理者が pkill コマンドを使用してサービスプロセスを終了する)

startd/ignore_error に core または signal、あるいはその両方を指定すると、サービスでは最後の 2 つの条件を無視できます。

サービスを transient として定義すると、svc.startd はそのサービスのプロセスを追跡しません。したがって、contract モデルサービスについて記述されている潜在的な障害は、transient サービスについては障害と見なされません。transient サービスは、メソッド失敗条件のいずれかが発生した場合だけ保守状態になります。

wait モデルサービスは、サービスに関連付けられている子プロセスが終了するたびに再起動されます。「wait」モデルサービスの場合、子プロセスの終了はエラーとは見なされず、障害が繰り返し発生しても保守状態への移行にはつながりません。

従来のサービス

svc.startd は、起動時の実行レベル移行中に呼び出されるサービスを引き続きサポートしています。同等の実行レベルのマイルストーンを構成しているすべての管理対象サービスがオンライン状態に移行したあと、各 /etc/rc?.d ディレクトリが処理されます。/etc/rc?.d ディレクトリにある標準の init スクリプトは、そのシーケンス番号の順に実行されます。

マイルストーンと実行レベルのマッピングは次のとおりです。

milestone/single-user	シングルユーザー (S)
milestone/multi-user	マルチユーザー (2)
milestone/multi-user-server	ネットワークサービス付きマルチユーザー (3)

また、svc.startd は、スクリプトごとに 1 つのインスタンスをリポジトリに挿入することにより、これらの従来のサービスが SMF で表示されるようにします。これ

らの従来のインスタンスは、svcs(1) などの標準の SMF インタフェースを使用して表示でき、常に LEGACY-RUN 状態で表示されます。変更したり、ほかのサービスの依存関係として指定したりすることはできません。管理者に役立つように、従来のサービスの初期の開始時刻が取り込まれます。

ファイル

/var/svc/log	svc.startd がログファイルを格納するディレクトリ。
/etc/svc/volatile	/var が読み書き用にマウントされる前のブートの初期段階で svc.startd がログファイルを格納するディレクトリ。

使用例

例 1　詳細ログをオンに設定する

詳細ログをオンに設定するには、次のように入力します。

```
# /usr/sbin/svccfg -s system/svc/restarter:default
svc:/system/svc/restarter:default> addpg options application
svc:/system/svc/restarter:default> setprop options/logging = \
astring: verbose
svc:/system/svc/restarter:default> exit
```

この要求は、次に svc.startd を再起動したときに有効になります。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

svcs(1), svcprop(1), kernel(1M), init(1M), svcadm(1M), svccfg(1M), svc.configd(1M), setsid(2), syslog(3C), libscf(3LIB), scf_value_create(3SCF), contract(4), init.d(4), process(4), inittab(4), attributes(5), smf(5), smf_method(5)

名前　swap – スワップ管理インタフェース

形式

```
/usr/sbin/swap  -a swapname [swaplow] [swaplen]

/usr/sbin/swap  -d swapname [swaplow]

/usr/sbin/swap  -l

/usr/sbin/swap  -s
```

機能説明　swap ユーティリティーは、メモリーマネージャーが使用するシステムスワップ領域を追加、削除、および監視する方法を提供します。

オプション　サポートしているオプションは、以下のとおりです。

-a *swapname*　指定されたスワップ領域を追加します。このオプションは、スーパーユーザーのみ使用できます。*swapname* はスワップファイルの名前です。たとえば、`/dev/dsk/c0t0d0s1` または通常ファイルです。*swaplow* は、ファイル内でスワップ領域を開始する場所のオフセット (512 バイトブロック単位) です。*swaplen* は、スワップ領域として望ましい長さ (512 バイトブロック単位) です。*swaplen* の値は 16 より小さくすることはできません。たとえば、*n* ブロックが指定された場合、(*n*-1) ブロックが実際のスワップの長さになります。*swaplen* は、長さが少なくとも 1 ページは必要です。メモリーのページのサイズは、`pagesize` コマンドを使用して決定することができます。`pagesize`(1) を参照してください。スワップファイルの最初のページは自動的にスキップされ、スワップファイルは長さが少なくとも 1 ページは必要なため、最小サイズは 2 ページサイズバイトの倍数にすることをお勧めします。メモリーのページのサイズは、マシンに依存します。

swaplow + *swaplen* は、スワップファイルのサイズ以下である必要があります。*swaplen* を指定しないと、*swaplow* から始まり指定したファイルの末尾で終わる領域が追加されます。*swaplow* も *swaplen* も指定されない場合、最初のページ以外のファイル全体が使用されます。スワップ領域は通常、システムの起動中に `/sbin/swapadd` スクリプトによって自動的に追加されます。このスクリプトは、`/etc/vfstab` ファイルで指定したすべてのスワップ領域を追加します。これらの指定する構文については、`vfstab`(4) を参照してください。

NFS またはローカルファイルシステム *swapname* を使用するには、最初に mkfile(1m) を使用してファイルを作成することをお勧めします。ローカルファイルシステムのスワップファイルの場合、`swap -a` コマンドを実行するだけで、稼働中のシステムに追加できます。NFS マウントされたスワップファイルの場合、サーバーはファイルをエクスポートする必要があります。このためには次の手順を実行します。

1. `/etc/dfs/dfstab` に次の行を追加します。

   ```
   share -F nfs -o rw=clientname,root=clientname path-to-swap-file
   ```

2. shareall(1M) を実行します。
3. クライアントに次の行を `/etc/vfstab` に追加させます。

   ```
   server:path-to-swap-file - local-path-to-swap-filenfs \
       - - - local-path-to-swap-file - - swap - - -
   ```

4. クライアントに `mount` を実行させます。

   ```
   # mount local-path-to-swap-file
   ```

5. クライアントは次に `swap -a` を実行して、スワップ空間を追加できます。

   ```
   # swap -a local-path-to-swap-file
   ```

`-d` *swapname*
: 指定されたスワップ領域を削除します。このオプションは、スーパーユーザーのみ使用できます。*swapname* はスワップファイルの名前です。たとえば、`/dev/dsk/c0t0d0s1` または通常ファイルです。*swaplow* は、削除されるスワップ領域のオフセット (512 バイトブロック単位) です。*swaplow* を指定しないと、領域の削除は第 2 ページから開始されます。コマンドが完了すると、この領域からスワップブロックを割り当てることはできなくなり、以前このスワップ領域で使用されていたすべてのスワップブロックは、別のスワップ領域へ移動されています。

`-l`
: すべてのスワップ領域の状態をリスト表示します。出力には 5 つの列があります。

  `path` スワップ領域のパス名です。

  `dev` ブロック特殊デバイスの場合は、十進数のメジャー/マイナーデバイス番号、それ以外の場合はゼロです。

  `swaplo` 領域の *swaplow* 値 (512 バイトブロック単位) です。

  `blocks` 領域の *swaplen* 値 (512 バイトブロック単位) です。

  `free` この領域で現在割り当てられていない 512 バイトブロックの数です。

  このリストには物理メモリー形式でのスワップ空間は含まれません。この空間は特定のスワップ領域と関連付けられていないためです。

  swapname の (`swap -d` による) 削除処理中に `swap -l` を実行した場合、スワップの状態の列の 6 番目に文字列 INDEL が表示されます。

`-s` 全体のスワップ空間の使用量および利用可能量についての概要情報を出力します。

`allocated` バッキングストアとして使用されるよう現在割り当てられているスワップ空間の、バイト単位の合計容量です。

`reserved` 現在は割り当てられていないが、あとから使用できるようにメモリーマッピングによって確保されているスワップ空間のバイト単位の合計容量です。

`used` 割り当て済みまたは予約済みのスワップ空間のバイト単位の合計容量です。

`available` あとから予約や割り当てに使用可能なスワップ空間のバイト単位の合計。

これらの数には、`-l` オプションによって表示されるすべての構成済みスワップ領域のスワップ空間に加えて、物理メモリー形式でのスワップ空間も含まれます。

使用法

32 ビットオペレーティングシステムでは、スワップデバイスが 2G バイト以上の場合、最初の 2G バイト -1 だけが使用されます。64 ビットオペレーティングシステムでは、2G バイトを超えるブロックデバイスを十分に活用でき、2^{63} –1 バイトまでスワップ用に使用できます。

環境変数

`swap` の実行に影響を与える次の環境変数についての詳細は、`environ`(5) を参照してください。`LC_CTYPE` および `LC_MESSAGE`。

属性

次の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

`pagesize`(1), mkfile(1m), shareall(1M), `getpagesize`(3C), `vfstab`(4), `attributes`(5), `largefile`(5)

警告

追加するスワップ領域が既存のファイルシステムとオーバーラップするかどうかの確認は行われません。

名前　sys-unconfig – システム構成の解除

形式　`/usr/sbin/sys-unconfig`

機能説明　`sys-unconfig` コマンドは、システムの構成を「出荷時設定」に戻し、再び構成可能な状態にします。システムの構成には、ホスト名、NIS (Network Information Service) ドメイン名、タイムゾーン、IP アドレス、IP サブネットマスク、および root パスワードが含まれます。これは、ブート時に `sysidnet`(1M)、`sysidns`(1M)、および `sysidsys`(1M) プログラムによって実行されるのと逆の処理です。`sysidtool`(1M) のマニュアルページを参照してください。

`sys-unconfig` の処理内容は次のとおりです。

- 現在の `/etc/inet/hosts` ファイル情報を `/etc/inet/hosts.saved` に保存する
- 現在の `/etc/vfstab` ファイルに NFS マウントのエントリが含まれている場合は、`/etc/vfstab.orig` に `/etc/vfstab` ファイルを保存する
- デフォルトの `/etc/inet/hosts` ファイルを復元する
- 実行時に構成されたすべてのインタフェースについて、`/etc/hostname.interface` ファイル内のデフォルトのホスト名を削除する。どのインタフェースが構成されたかを確認するには、コマンド *ifconfig-a* を実行します。生成される出力に示されているすべてのインタフェースに対応する `/etc/hostname.`*interface* ファイル (ループバックインタフェース (lo0) を除く) が削除されます。
- `/etc/defaultdomain` 内のデフォルトのドメイン名を削除する
- `/etc/TIMEZONE` 内のタイムゾーンを PST8PDT に戻す
- NIS (Network Information Service) または NIS+ (Network Information Service Plus) が構成されている場合、これらを無効にする
- ファイル `/etc/inet/netmasks` を削除する
- ファイル `/etc/defaultrouter` を削除する
- スーパーユーザー用に設定されているパスワードを `/etc/shadow` から削除する
- ファイル `/etc/.rootkey` を削除する
- すべてのシステム構成アプリケーションを実行する。これらのアプリケーションは、あらかじめ `sysidconfig -a` *application* を実行して定義されています (`sysidconfig`(1M) のマニュアルページを参照)。`sys-unconfig` の実行時に、すべてのシステム構成アプリケーションに 1 つの引数 `-u` が渡されます。
- ファイル `/etc/resolv.conf` を削除する
- `/var/ldap/ldap_client_cache`、`/var/ldap/ldap_client_file`、`/var/ldap/ldap_client_cred`、および `/var/ldap/cachemgr.log` を削除することにより LDAP を無効にする
- `sshd`(1M) のキーを再生成する

sys-unconfig は、その終了時にシステムを停止します。sys-unconfig は潜在的に危険なユーティリティーであるため、スーパーユーザーだけが実行できます。

ファイル

/etc/default/init	プロセス制御の初期化
/etc/defaultdomain	
/etc/defaultrouter	
/etc/hostname.*interface*	
/etc/inet/hosts	ホスト名データベース
/etc/inet/netmasks	ネットワークマスクデータベース
/etc/nodename	
/etc/.rootkey	スーパーユーザーの秘密鍵
/etc/shadow	シャドウパスワードファイル
/etc/vfstab	仮想ファイルシステムテーブル
/var/nis/NIS_COLD_START	
/var/yp/binding/*/ypservers	

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWadmap

関連項目

init(1M), kdmconfig(1m), sysidconfig(1M), sysidtool(1M), hosts(4), netmasks(4), shadow(4), attributes(5)

注意事項

sys-unconfig は、ディスクレスクライアントでは使用できません。

名前

syslogd – log system messages

形式

```
/usr/sbin/syslogd  [-d] [-f configfile] [-m markinterval]
 [-p path] [-t |  -T]
```

機能説明

syslogd reads and forwards system messages to the appropriate log files or users, depending upon the priority of a message and the system facility from which it originates. The configuration file /etc/syslog.conf (see syslog.conf(4)) controls where messages are forwarded. syslogd logs a mark (timestamp) message every *markinterval* minutes (default 20) at priority LOG_INFO to the facility whose name is given as mark in the syslog.conf file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (< >); priorities are defined in <sys/syslog.h>.

syslogd reads from the STREAMS log driver, /dev/log, and from any transport provider specified in /etc/netconfig, /etc/net/transport/hosts, and /etc/net/transport/services.

syslogd reads the configuration file when it starts up, and again whenever it receives a HUP signal (see signal.h(3HEAD), at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. syslogd exits when it receives a TERM signal.

As it starts up, syslogd creates the file /var/run/syslog.pid, if possible, containing its process identifier (PID).

If message ID generation is enabled (see log(7D)), each message will be preceded by an identifier in the following format: [ID *msgid facility*.*priority*]. *msgid* is the message's numeric identifier described in msgid(1M). *facility* and *priority* are described in syslog.conf(4). [ID 123456 kern.notice] is an example of an identifier when message ID generation is enabled.

If the message originated in a loadable kernel module or driver, the kernel module's name (for example, ufs) will be displayed instead of unix. See EXAMPLES for sample output from syslogd with and without message ID generation enabled.

In an effort to reduce visual clutter, message IDs are not displayed when writing to the console; message IDs are only written to the log file. See 使用例.

The /etc/default/syslogd file contains the following default parameter settings, which are in effect if neither the -t nor -T option is selected. See FILES.

The recommended way to allow or disallow message logging is through the use of the service management facility (smf(5)) property:

```
svc:/system/system-log/config/log_from_remote
```

This property specifies whether remote messages are logged. log_from_remote=true is equivalent to the -t command-line option and false is equivalent to the -T command-line option. The default value for -log_from_remote is true. See NOTES, below.

`LOG_FROM_REMOTE`
Specifies whether remote messages are logged. `LOG_FROM_REMOTE=NO` is equivalent to the `-t` command-line option. The default value for `LOG_FROM_REMOTE` is `YES`.

オプション

The following options are supported:

`-d`	Turn on debugging. This option should only be used interactively in a root shell once the system is in multi-user mode. It should *not* be used in the system start-up scripts, as this will cause the system to hang at the point where `syslogd` is started.
`-f` *configfile*	Specify an alternate configuration file.
`-m` *markinterval*	Specify an interval, in minutes, between mark messages.
`-p` *path*	Specify an alternative log device name. The default is `/dev/log`.
`-T`	Enable the `syslogd` UDP port to turn on logging of remote messages. This is the default behavior. See ファイル.
`-t`	Disable the `syslogd` UDP port to turn off logging of remote messages. See ファイル.

使用例

例 1 `syslogd` Output Without Message ID Generation Enabled

The following example shows the output from `syslogd` when message ID generation *is not* enabled:

```
Sep 29 21:41:18 cathy unix: alloc /: file system full
```

例 2 `syslogd` Output with ID generation Enabled

The following example shows the output from `syslogd` when message ID generation *is* enabled. The message ID is displayed when writing to log file`/var/adm/messages`.

```
Sep 29 21:41:18 cathy ufs: [ID 845546 kern.notice]
                                    alloc /: file system full
```

例 3 `syslogd` Output with ID Generation Enabled

The following example shows the output from `syslogd` when message ID generation *is* enabled when writing to the console. Even though message ID is enabled, the message ID is not displayed at the console.

```
Sep 29 21:41:18 cathy ufs: alloc /: file system full
```

例 4 Enabling Acceptance of UDP Messages from Remote Systems

The following commands enable `syslogd` to accept entries from remote systems.

```
# svccfg -s svc:/system/system-log setprop config/log_from_remote = true
# svcadm refresh svc:/system/system-log
```

ファイル

`/etc/syslog.conf`	Configuration file
`/var/run/syslog.pid`	Process ID
`/etc/default/syslogd`	Contains default settings. You can override some of the settings by command-line options.
`/dev/log`	STREAMS log driver
`/etc/netconfig`	Transport providers available on the system
`/etc/net/transport/hosts`	Network hosts for each transport
`/etc/net/transport/services`	Network services for each transport

属性

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

関連項目

`logger`(1), `svcs`(1), `msgid`(1M), svcadm(1M), svccfg(1M), `syslog`(3C), `syslog.conf`(4), `attributes`(5), `signal.h`(3HEAD), `smf`(5), `log`(7D)

注意事項

The `mark` message is a system time stamp, and so it is only defined for the system on which `syslogd` is running. It can not be forwarded to other systems.

When `syslogd` receives a `HUP` signal, it attempts to complete outputting pending messages, and close all log files to which it is currently logging messages. If, for some reason, one (or more) of these files does not close within a generous grace period, `syslogd` discards the pending messages, forcibly closes these files, and starts reconfiguration. If this shutdown procedure is disturbed by an unexpected error and `syslogd` cannot complete reconfiguration, `syslogd` sends a mail message to the superuser on the current system stating that it has shut down, and exits.

Care should be taken to ensure that each window displaying messages forwarded by `syslogd` (especially console windows) is run in the system default locale (which is `syslogd`'s locale). If this advice is not followed, it is possible for a `syslog` message to alter the terminal settings for that window, possibly even allowing remote execution of arbitrary commands from that window.

The `syslogd` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/system/system-log:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the `svcs`(1) command.

When `syslogd` is started by means of svcadm(1M), if a value is specified for `LOG_FROM_REMOTE` in the `/etc/defaults/syslogd` file, the SMF property `svc:/system/system-log/config/log_from_remote` is set to correspond to the `LOG_FROM_REMOTE` value and the `/etc/default/syslogd` file is modified to replace the `LOG_FROM_REMOTE` specification with the following comment:

```
# LOG_FROM_REMOTE is now set using svccfg(1m), see syslogd(1m).
```

If neither `LOG_FROM_REMOTE` nor `svc:/system/system-log/config/log_from_remote` are defined, the default is to log remote messages.

On installation, the initial value of `svc:/system/system-log/config/log_from_remote` is `false`.

名前　ttymon – 端末ポートのポートモニター

形式

```
/usr/lib/saf/ttymon

/usr/lib/saf/ttymon -g [-d device] [-h] [-t timeout]
     [-l ttylabel] [-p prompt] [-m modules] [-T termtype]
```

機能説明

ttymon は STREAMS ベースの TTY ポートモニターです。その機能には、ポートの監視、ポートの端末モード、ボーレートおよび回線規約の設定、そしてポートに関連したサービスへの ユーザーまたはアプリケーションの接続があります。通常、ttymon は、サービス・アクセス・ファシリティ (SAF) の一部として、サービス・アクセス・コントローラ、sac(1M) の制御下で実行するように構成されています。ttymon は sacadm(1M) コマンドを実行して構成されます。ttymon の各インスタンスは複数のポートを監視できます。ttymon のインスタンスによって監視されるポートは、ポートモニターの管理ファイルに指定されています。管理ファイルは pmadm(1M) コマンドおよび ttyadm(1M) コマンドを実行して構成されます。ttymon のインスタンスは、sac コマンドによって呼び出されると、ポートの監視を始めます。各ポートについて、ttymon は回線規約が指定されていれば、最初に回線規約を初期化してから、速度と端末設定を初期化します。/etc/logindevperm 中のエントリに指定されているポートには、デバイスのオーナー、グループ、アクセス権が設定されます (logindevperm(4) 参照)。初期化に使用される値は、TTY 設定ファイルの適切なエントリから取り込まれます。このファイルは sttydefs(1M) コマンドによって管理されています。通常、ポート上のデフォルトの回線規約は、オートプッシュファシリティの autopush(1M) コマンドによって設定されます。

次に ttymon はプロンプトを書き込んでユーザーの入力を待ちます。ユーザーが BREAK キーを押して速度が適切でないことを知らせると、ttymon は次の速度を試行してプロンプトを再び書き込みます。正しい入力を受信すると、ttymon は、(もしあれば) ポートのサービスごとの構成ファイルを解釈し、必要に応じて utmpx エントリを作成します (utmpx(4) 参照)。そして、サービス環境を確立し、ポートに関連するサービスを呼び出します。有効な入力は、キャリッジリターンで終わり、少なくとも 1 つの復帰改行以外の文字が入った文字列からなります。サービスが終了すると、ttymon は utmpx エントリが存在していればそのエントリを消去して、ポートを初期状態に戻します。

autobaud がポートで動作可能であれば、ttymon は自動的にポートのボーレートを判別しようとします。ユーザーがキャリッジリターンを入力してからでないと、ttymon はボーレートを認識してプロンプトをプリントできません。現在のところ、*autobaud* が判別できるボーレートは 110、1200、2400、4800、および 9600 です。

ポートが双方向性ポートとして構成されている場合、ttymon によって、ユーザーはサービスに接続できるようになり、またポートが使用されていなければ、uucico(1M)、cu(1C)、または ct(1C) はダイアルアウト用にポートを使用できるようになります。ポートが双方向性であれば、ttymon は文字の読み込みが終わるのを待ってからプロンプトをプリントします。

ポート用に *connect-on-carrier* フラグをセットしている場合、ttymon は、接続要求を受け取るとただちに ポートに関連付けられたサービスを呼び出します。プロンプトメッセージは送信されません。

ポートが動作禁止になっていると、ttymon はこのポートにおいてサービスを呼び出しません。動作禁止メッセージが指定されている場合には、ttymon は接続要求を受け取ると動作禁止メッセージを送信します。ttymon が動作禁止になっている場合、ttymon のインスタンスの制御下にあるすべてのポートが動作禁止になります。

サービスの呼び出し

ポートに ttymon が呼び出すサービスは、ttymon 管理ファイルに指定されています。ttymon は、このポートに呼び出されるサービスを指示する文字列を走査して、%d または %% の 2 文字のシーケンスを捜します。%d を見つけると、ttymon は、これらの 2 文字をこのポートの完全パス名 (デバイス名) に置き換えることによって、実行されるサービスコマンドを変更します。%% を見つけると、これらは単一の % に置き換えられます。サービスを呼び出すと、読み書きするためにポートデバイスに対して 0、1、および 2 の各ファイル記述子がオープンされます。サービスは、ttymon に登録したときに用いたユーザー名についてのユーザー ID 、グループ ID、および現在のホームディレクトリを使用して呼び出されます。2 つの環境変数、HOME および TTYPROMPT は、ttymon によってサービスの環境に追加されます。HOME はサービスを呼び出すときに用いるユーザー名のホームディレクトリに設定されています。TTYPROMPT はこのポートのサービスに構成されたプロンプト文字列に設定されています。ttymon が呼び出すサービスが、プロンプトが実際に ttymon によって出されていたかを判別し、出されている場合にはそのプロンプトが実際には何であったかを判断する機能を持つように、設定されています。

サービスアクセスコントローラの制御下にある ttymon により監視されるポートに設定できるオプションについては、ttyadm(1M) を参照してください。

システムコンソールの呼び出し

システムコンソール上での ttymon の呼び出しは、サービス svc:/system/console-login によって smf(5) の元で管理されます。次に示すように、呼び出しを制御するためにプロパティグループ ttymon 内の多くのプロパティを提供します。

```
NAME                 TYPE              TTYMON OPTION
-------------------------------------------------------------
device               astring           [-d device]
nohangup             boolean           [-h]
label                astring           [-l label]
modules              astring           [-m module1,module2]
prompt               astring           [-p prompt]
timeout              count             [-t timeout]
terminal_type        astring           [-T termtype]
```

いずれかの値が空の文字列であるかゼロに設定された整数である場合、オプションは ttymon 呼び出しに渡されません。-g オプションはつねにこの呼び出しに指定されます。-d オプションは、設定されていない場合、つねにデフォルトの /dev/console になります。

「使用例」を参照してください。

セキュリティ

ttymon は pam(3PAM) を使って、セッション管理を行います。PAM 構成ポリシーは ttymon で使用するモジュールを明記しています。このポリシーは /etc/pam.conf で見ることができます。以下に UNIX セッション管理モジュールを使用する ttymon コマンドのエントリの入った pam.conf ファイルの抜粋を示します。

```
ttymon session  required /usr/lib/security/pam_unix_session.so.1
```

ttymon サービスのエントリがない場合には other のサービスのエントリを使用します。

オプション

次のオプションを指定できます。

-g
: ttymon の特殊な呼び出しは -g オプションを指定して行います。コマンドのこの書式を呼び出せるのは、ポートに正しいボーレートおよび端末設定を設定してから login サービスに接続する必要があるアプリケーションだけにする必要があります。SAC の制御下では前もって構成することはできません。-g は、以下のようなオプションの組み合わせとともに使用できます。

-d*device*
: *device* は ttymon の接続先とすべきポートの完全パス名です。このオプションが指定されていなければ、ファイル記述子 0 は TTY ポートに対する呼び出しプロセスによって設定される必要があります。

-h
: -h フラグが指定されていないと、ttymon は、速度をデフォルト速度または指定速度に設定する前に、速度を 0 に設定することによって回線のハングアップを強制的に行います。

-l*ttylabel*
: *ttylabel* は ttydefs ファイルの速度および TTY 定義に対するリンクです。この定義によって、初期の実行速度、初期の TTY 設定の内容およびユーザーが BREAK キーを押して速度が適切でないことを指示する場合に、次に試行する速度が ttymon に通知されます。デフォルト速度は 9600 ボーです。

-m*modules*
: ポートを初期化すると、ttymon はポートのモジュールすべてをポップして、指定した順序で *modules* をプッシュします。*modules* はコンマで区切ったプッシュ可能なモジュールのリストです。通常、ポートのデフォルトのモジュールは、オートプッシュファシリティによって設定されます。

-p*prompt*
: ユーザーはこれを用いるとプロンプト文字列を指定できます。デフォルトのプロンプトは Login: です。

-t *timeout*
: プロンプトの送信後、*timeout* 秒内に何も入力がない場合には ttymon を終了します。

-T*termtype*
: *termtype* に TERM 環境変数を設定します。

使用例

例1 端末タイプの設定

次の例では、システムコンソール ttymon 呼び出しのための端末タイプ (-T) オプションの値を設定します。

```
svccfg -s svc:/system/console-login setprop \
    ttymon/terminal_type = "xterm"
svcadm refresh svc:/system/console-login:default
```

環境

LC_* 変数 (LC_CTYPE、LC_MESSAGES、LC_TIME、LC_COLLATE、LC_NUMERIC、LC_MONETARY) (environ(5) 参照) のいずれも環境に設定されていなければ、それぞれ対応する ロケール のカテゴリにおける ttymon の動作は、環境変数 LANG によって決定されます。もし、LC_ALL が設定されていれば、その内容が LANG 変数やその他の LC_* 変数より優先されます。上記の変数が環境にまったく設定されていなければ、C ロケール (米国スタイル) が ttymon の動作を決定します。

LC_CTYPE ttymon の文字の処理方法を決定します。LC_CTYPE に有効な値が設定されていると、ttymon は、そのロケールにあった文字を含むテキストやファイル名を表示および処理できます。ttymon は拡張 UNIX コード(EUC)も表示および処理できます。この場合、文字は 1 バイト幅、 2 バイト幅、3 バイト幅のいずれも使用できます。また、ttymon は 1、2、またはそれ以上のカラム幅の EUC 文字も処理することができます。C ロケールにおいては、ISO 8859-1 の文字だけが有効です。

ファイル

/etc/logindevperm

属性

次の属性については attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu
安定性	下記を参照

コマンド行の形式は変更ありません。SMF プロパティは開発中です。

関連項目

ct(1C), cu(1C), autopush(1M), pmadm(1M), sac(1M), sacadm(1M), sttydefs(1M), ttyadm(1M), uucico(1M), pam(3PAM), logindevperm(4), pam.conf(4), utmpx(4), attributes(5), environ(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5)

『Solaris のシステム管理 (基本編)』

注意事項

ポートが複数の ttymon によって監視されている場合は、ttymon は入力を争うような方法でプロンプトメッセージを送信できます。

pam_unix(5) モジュールはサポートされません。同様の機能は、pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), および pam_unix_session(5) で提供されています。

名前

unshare – ローカルシステムをリモートシステムからマウント不可能にする設定

形式

```
unshare  [-F FSType] [-o specific_options] [pathname
 |  resourcename]
```

機能説明

unshare コマンドは、共有されているローカルリソースを、ファイルシステムタイプ *FSType* として利用できないようにします。オプション -F*FSType* を省略した場合、/etc/dfs/fstypes ファイル内の最初のファイルシステムタイプがデフォルトで使用されます。*Specific_options*、および *resourcename* の意味は、個々の分散ファイルシステムによって異なります。

オプション

-F *FSType*　　ファイルシステムタイプを指定します。

-o *specific_options*　　-F オプションによって指定されるファイルシステムに固有なオプションを指定します。

ファイル

/etc/dfs/fstypes

/etc/dfs/sharetab

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目

share(1M), shareall(1M), attributes(5)

注意事項

共有されている情報で *pathname* または *resourcename* のいずれかが見つからない場合、標準エラー出力にエラーメッセージが送られます。

unshare コマンドが正常に終了すると、unshare コマンドに指定されたファイルシステムをマウントしているクライアントは、そのファイルシステムにアクセスできなくなります。

名前　wall – すべてのユーザーへの通知

形式　`/usr/sbin/wall [-a] [-g grpname] [filename]`

機能説明　`wall` はファイルの終わりまで標準入力を読み取ります。その後、現在ログインしているすべてのユーザーに以下の文字で始まるメッセージを送信します。

```
Broadcast Message from . . .
```

filename が指定されていれば、メッセージはこのファイルから読み込まれます。通常、リモートログインセッションに対応していない仮想端末は無視されます。したがって、ウィンドウシステムを使用しているときは、メッセージはコンソールウィンドウにだけに表示されます。ただし、`-a` オプションを指定すると、メッセージを仮想端末などにも送信します。

一般的には、システムをシャットダウンする前に、すべてのユーザーに警告を与える場合に、このメッセージを用います。

ユーザーが起動させた保護を無効にする場合、送信側はスーパーユーザーである必要があります (`mesg`(1) 参照)。

`wall` は、他のユーザーの端末上で書き込み権を持つために、グループ ID `tty` に対して `setgid()` を実行します。 (`setuid`(2) 参照)。

`wall` は、ユーザーの端末に送信する前に非表示可能文字を検出します。制御文字は、適切な ASCII 文字が後に続く ^ として表示されます。すなわち、高位ビットが設定された文字は meta 表記法で表示されます。たとえば、`\003` は `^C`、また `\372` は `M–z` と表示されます。

オプション　次のオプションを指定できます。

`-a`　コンソールと仮想端末にメッセージを送ります。

`-g` *grpname*　グループデータベース (`group`(4) のマニュアルページを参照) ごとに、grpname で指定されたグループのユーザーだけにメッセージを送ります。

環境　`LC_*` 変数 (`LC_CTYPE`、`LC_TIME`、`LC_COLLATE`、`LC_NUMERIC`、`LC_MONETARY`) が環境に設定されていなければ、それぞれ対応する ロケール のカテゴリにおける `wall` の動作は、環境変数 `LANG` によって決定されます。 (`environ`(5) 参照)。もし、`LC_ALL` が設定されていれば、その内容が `LANG` 変数やその他の `LC_*` 変数より優先されます。上記の変数が環境にまったく設定されていなければ、 C ロケール (米国スタイル) が `wall` の動作を決定します。

ファイル　`/dev/tty*`

属性　次の属性については `attributes`(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	SUNWcsu

関連項目 mesg(1), write(1), setuid(2), attributes(5), environ(5)

注意事項 wall は、ユーザーの tty ファイルでオープンが失敗すると、Cannot send to . . . と表示します。

名前

zfs – ZFS ファイルシステムの構成

形式

```
zfs [-?]

zfs create [[-o property=value]]... filesystem

zfs create [-s] [-b blocksize] [[-o property=value]]... -V size volume

zfs destroy [-rRf] filesystem|volume|snapshot

zfs clone snapshot filesystem|volume

zfs promote filesystem

zfs rename filesystem|volume|snapshot
     [filesystem|volume|snapshot]

zfs snapshot [-r] filesystem@name|volume@name

zfs rollback [-rRf] snapshot

zfs list [-rH] [-o prop[,prop] ]... [ -t type[,type]]...
     [ -s prop [-s prop]]... [ -S prop [-S prop]...]
     filesystem|volume|snapshot|/pathname|./pathname ...

zfs set property=value filesystem|volume ...

zfs get [-rHp] [-o field[,field]...]
     [-s source[,source]...] all | property[,property]
     ... filesystem|volume|snapshot ...

zfs inherit [-r] property filesystem|volume... ...

zfs mount

zfs mount [-o options] [-O] -a

zfs mount [-o options] [-O] filesystem

zfs unmount [-f] -a

zfs unmount [-f] filesystem|mountpoint

zfs share -a

zfs share filesystem

zfs unshare [-f] -a

zfs unshare [-f] filesystem|mountpoint

zfs send [-i snapshot1] snapshot2

zfs receive [-vnF ] filesystem|volume|snapshot

zfs receive [-vnF ] -d filesystem
```

機能説明

zfs コマンドは、ZFS ストレージプール内の ZFS データセットを構成します。詳細は、zpool(1M) を参照してください。データセットの識別には、ZFS 名前空間内部の一意のパスが使用されます。次に例を示します。

```
pool/{filesystem,volume,snapshot}
```

この場合、データセット名の最大長は MAXNAMELEN (256 バイト) です。

使用できるデータセットは、次のうちのいずれかです。

file system 標準の POSIX ファイルシステム。ZFS ファイルシステムは、標準ファイルシステムの名前空間内でマウントでき、その他のファイルシステムと同様に動作します。

volume ブロック型デバイスとしてエクスポートされた論理ボリューム。このタイプのデータセットは特殊な状況でのみ使用し、通常はファイルシステムを使用するようにしてください。ボリュームは、非大域ゾーン内では使用できません。

snapshot 特定の時点におけるファイルシステムまたはボリュームの読み取り専用バージョン。これは、*filesystem@name* または *volume@name* として指定されます。

ZFS ファイルシステム階層

ZFS ストレージプールは、データセット用の領域を提供するデバイスの論理コレクションです。ストレージプールは、ZFS ファイルシステム階層のルートでもあります。

プールのルートにはファイルシステムとしてアクセスできるため、マウントやマウント解除、スナップショットの作成、およびプロパティーの設定などの操作を実行できます。ただし、物理ストレージ特性の管理には、zpool(1M) コマンドを使用します。

プールの作成および管理の詳細は、zpool(1M) を参照してください。

スナップショット

スナップショットは、ファイルシステムまたはボリュームの読み取り専用コピーです。スナップショットはきわめてすばやく作成することができ、最初はプール内で追加の領域を消費しません。アクティブなデータセット内のデータが変更されると、スナップショット用のデータが増加していきます。これは、本来ならアクティブなデータセットと共有されていたデータです。

スナップショットには、任意の名前を付けることができます。ボリュームのスナップショットは、クローンやロールバックを作成することは可能ですが、個別にアクセスすることはできません。

ファイルシステムのスナップショットには、ファイルシステムのルートにある「.zfs/snapshot」ディレクトリからアクセスできます。スナップショットは、要求に応じて自動的にマウントされます。また、一定の時間が経過するとスナップショットをマウント解除することもできます。「.zfs」ディレクトリの可視設定は、「snapdir」プロパティーを使って制御できます。

クローン

クローンとは、別のデータセットと同一の初期内容を持つ書き込み可能なボリュームまたはファイルシステムです。スナップショットの場合と同様に、クローンは瞬間的に作成され、最初は追加の領域を消費しません。

クローンは、スナップショットだけから作成できます。スナップショットのクローンを作成すると、親子間に暗黙の依存関係が作成されます。クローンをデータセット階層内の別の場所に作成した場合でも、クローンが存在するかぎり、元のスナップショットを破棄することはできません。この依存関係は、「origin」プロパティーからわかります。そのような依存関係が存在する場合には、`destroy` コマンドを実行すると表示されます。

「`promote`」サブコマンドを使えば、クローンの親子依存関係を逆転させることができます。これを行うと、「元」のファイルシステムが、指定されたファイルシステムのクローンになります。このため、クローンの作成元のファイルシステムを破棄できるようになります。

マウントポイント

ZFS ファイルシステムは簡単な操作で作成できるため、システムごとのファイルシステムの数はかなり多くなる可能性があります。この状況に対応するため、`/etc/vfstab` ファイルを編集しなくても、ZFS はファイルシステムのマウントおよびマウント解除を自動的に管理します。自動管理されるすべてのファイルシステムは、ブート時に ZFS によりマウントされます。

デフォルトでは、ファイルシステムは */path* 以下にマウントされます。ここで、*path* は ZFS 名前空間内のファイルシステムの名前です。ディレクトリが、必要に応じて作成および破棄されます。

また、「mountpoint」プロパティーにファイルシステムのマウントポイントを設定することも可能です。このディレクトリは必要に応じて作成され、「`zfs mount -a`」コマンドの呼び出し時に ZFS によりファイルシステムが自動的にマウントされます (`/etc/vfstab` の編集は不要)。このマウントポイントプロパティーは継承可能です。このため、`pool/home` が `/export/stuff` のマウントポイントを保持する場合、`pool/home/user` は自動的に `/export/stuff/user` のマウントポイントを継承します。

ファイルシステムの mountpoint プロパティーが「none」の場合、ファイルシステムはマウントされません。

必要に応じ、従来のツール (`mount`、`umount`、`/etc/vfstab`) を使って ZFS ファイルシステムを管理することもできます。ファイルシステムのマウントポイントが「legacy」に設定されている場合、ZFS はファイルシステムの管理を試みません。管理者が、ファイルシステムのマウントおよびマウント解除を担当します。

ゾーン

ZFS ファイルシステムを非大域ゾーンに追加するには、zonecfg の「`add fs`」サブコマンドを使用します。非大域ゾーンに追加する ZFS ファイルシステムでは、mountpoint プロパティーを legacy に設定する必要があります。

追加したファイルシステムの物理プロパティーの制御は、大域管理者が行います。ただし、ゾーン管理者は、追加したファイルシステム内部のファイルを、ファイルシステムのマウント方法に対応した方法で作成、変更、または破棄できます。

データセットを非大域ゾーンに委任する場合は、zonecfg の「`add dataset`」サブコマンドを使用します。データセットをあるゾーンに委任し、同じデータセットの子を別のゾーンに委任することはできません。ゾーン管理者は、データセットおよびそのすべての子のプロパティーを変更できます。ただし、「quota」プロパティーの制御は、大域管理者が行います。

ZFS のエミュレートされたボリュームをデバイスとして非大域ゾーンに追加するには、zonecfg の「`add device`」サブコマンドを使用します。ただし、その物理プロパティーを変更できるのは大域管理者だけです。

`zonecfg` の構文の詳細は、zonecfg(1M) を参照してください。

データセットが非大域ゾーンに委任されると、「zoned」プロパティーが自動的に設定されます。ゾーン管理者により、マウントポイントが受け入れられない値に設定されている可能性があるため、ゾーンファイルシステムを大域ゾーンにマウントすることはできません。

大域管理者は「zoned」プロパティーを強制的にクリアできます。 ただし、この操作は特に慎重に行うようにしてください。大域管理者は、このプロパティーをクリアする前に、すべてのマウントポイントが受け入れられるものであることを確認してください。

ネイティブプロパティー

プロパティーは、ネイティブプロパティーとユーザー定義プロパティーの 2 つの種類に分けられます。ネイティブプロパティーは、内部統計データのエクスポートや ZFS の動作制御を行います。また、ネイティブプロパティーは編集可能であるか、読み取り専用です。ユーザープロパティーは ZFS の動作には影響を及ぼしませんが、ユーザーの環境で意味のある方法でデータセットに注釈を付けるのに利用できます。ユーザープロパティーの詳細は「ユーザープロパティー」の項を参照してください。

すべてのデータセットには、さまざまな動作を制御するプロパティーに加え、データセットに関する統計データを出力するプロパティーがあります。プロパティーは、子により上書きされないかぎり、親から継承されます。スナップショットのプロパティーは編集できません。スナップショットは常に、継承可能なプロパティーを継承します。スナップショットに適用不可能なプロパティーは、表示されません。

数値プロパティーの値は、人間が読み取ることのできるサフィックス (k、KB、M、Gb など) を使って指定できます。使用可能な最大サフィックスは Z (ゼッタバイト) です。次の指定はすべて有効 (および等価) です。

`"1536M"、"1.5g"、"1.50GB"`

数値以外のプロパティーの値では、大文字と小文字が区別され、「mountpoint」と「sharenfs」を除き、小文字にする必要があります。

最初のプロパティーセットは、データセットに関する統計データを表示するものです。これらのプロパティーは、設定不可能であり、継承も行われません。ネイティブプロパティーは、特に注記がないかぎり、すべてのデータセットタイプに適用されます。

type
: データセットのタイプ。「filesystem」、「volume」、「snapshot」、または「clone」です。

creation
: このデータセットが作成された時刻。

used
: このデータセットおよびそのすべての子孫が使用する容量を調べます。この値は、このデータセットの割り当ておよび予約に基づいて計算されます。使用される領域にこのデータセットの予約は含まれませんが、子孫のデータセットがある場合はそれらの予約も考慮されます。データセットがその親から継承して使用する容量、およびこのデータセットが再帰的に破棄されるときに解放される容量は、使用済み領域および予約の中で大きな割合を占めます。

  スナップショット (「スナップショット」の節を参照) を作成したときは、それらの領域は最初はスナップショットとファイルシステムの間で共有されます。それまでに作成したスナップショットと領域が共有されることもあります。ファイルシステムが変化していくにつれて、それまで共有されていた領域がスナップショット固有になり、スナップショットが使用する領域に計上されます。また、スナップショットを削除すると、ほかのスナップショットに固有の (および使用される) 容量を増やすことができます。

  使用している容量、使用できる容量、または参照する容量では、保留状態の変更は考慮されません。保留状態の変更は通常、数秒以内に計上されます。`fsync`(3C) や `O_SYNC` を使用してディスクへの変更をコミットしても、領域の使用状況の情報がすぐに更新されることが保証されているわけではありません。

available
: データセットおよびその子すべてが使用可能な容量。プール内にその他のアクティビティーが存在しないものとして計算されます。容量はプール内で共有されるため、プールの物理サイズ、割り当て、予約、プール内のほかのデータセットなどのさまざまな要因によって、利用できる容量が制限されることがあります。

  このプロパティーは、列名の短縮形「avail」を使用しても参照できます。

referenced	このデータセットでアクセス可能なデータ量。これは、プール内のほかのデータセットと共有される場合も、共有されない場合もあります。スナップショットまたはクローンを作成したときには、それらの作成元のファイルシステムまたはスナップショットと同じ領域を最初は参照しています。内容が同じであるためです。 このプロパティーは、列名の短縮形「refer」を使用しても参照できます。
compressratio	このデータセットに対して達成された圧縮比。乗数で表記されます。「zfs set compression=on *dataset*」を実行すると、圧縮を有効にできます。デフォルト値は「off」です。
mounted	ファイルシステムの場合は、ファイルシステムが現在マウントされているかどうかを示します。このプロパティーは、「yes」または「no」になります。
origin	ファイルシステムまたはボリュームのクローンを作成した場合は、クローンの作成元のスナップショット。クローンが存在するかぎり、`-r` や `-f` オプションを使用しても、作成元は破棄できません。

次からのプロパティーセットは、データセット間で容量を割り当てる方法を制御するものです。これらのプロパティーは継承されませんが、子孫に影響を及ぼします。

quota=*size* | *none*	データセットおよびその子孫が使用できる容量を制限します。このプロパティーにより、使用される容量に対して強い制限値が設定されます。これには、子孫の消費する容量すべて(ファイルシステムとスナップショットを含む)が含まれます。割り当てがすでに設定されているデータセットの子孫に割り当てを設定した場合は、祖先の割り当ては上書きされずに、制限が追加されます。 ボリュームには割り当てを設定できません。「volsize」プロパティーが暗黙的な割り当てとして機能します。
reservation=*size* | *none*	データセットおよびその子孫に保証される最小容量。使用している容量がこの値を下回っているデータ

セットは、予約に指定された容量を使用していると見なされます。予約は、親データセットが使用する容量に計上されるので、親データセットの割り当てと予約を減らすことになります。

このプロパティーは、列名の短縮形「reserv」を使用しても参照できます。

volsize=*size*

ボリュームの場合に、ボリュームの論理サイズを指定します。デフォルトでは、ボリュームを作成するときに、同じ容量の予約が設定されます。`volsize` への変更があった場合には、予約にも対応する変更が反映されます。`volsize` に設定可能な値は、`volblocksize` の倍数だけです。この値をゼロにすることはできません。

コンシューマの予期しない動作を防ぐため、予約はボリュームの論理サイズと等価に保たれます。予約を使用しない場合、ボリュームの使用方法によっては、ボリュームの容量が不足して未定義の動作またはデータ破壊が発生する可能性があります。このような影響は、ボリュームの使用中にボリュームサイズを変更した場合にも発生することがあります。特に、サイズを縮小した場合にはその可能性が高くなります。 ボリュームサイズを調整するときは、特に注意するようにしてください。

推奨されてはいませんが、「`zfs create -V`」コマンドに `-s` オプションを指定するか、ボリュームの作成後に予約を変更することにより、「疎ボリューム」(「シンプロビジョニング」とも呼ばれる) を作成できます。「疎ボリューム」とは、予約がボリュームサイズよりも小さいボリュームのことです。このため、領

域上のプールが小さい場合、疎ボリュームへの書き込みが ENOSPC で失敗する可能性があります。疎ボリュームの場合、`volsize` を変更しても予約には反映されません。

volblocksize=*blocksize*

ボリュームの場合に、ボリュームのブロックサイズを指定します。ボリュームが書き込まれたあとで、`blocksize` を変更することはできません。このため、このプロパティーはボリュームの作成時に設定してください。ボリュームのデフォルト `blocksize` は、8K バイトです。512 バイト～ 128K バイトの範囲で、任意の 2 の累乗を指定できます。

このプロパティーは、列名の短縮形「volblock」を使用しても参照できます。

recordsize=*size*

ファイルシステムに格納するファイルの推奨ブロックサイズを指定します。このプロパティーは、レコードサイズが固定されているファイルにアクセスするデータベースワークロードだけで使用するように設計されています。ZFS では、標準的なアクセスパターンに最適化された内部アルゴリズムに従って、ブロックサイズが自動的に調整されます。

作成されるファイルのサイズが非常に大きく、それらのファイルにさまざまなパターンの小さなブロック単位でアクセスするデータベースの場合には、このようなアルゴリズムが最適でないことがあります。「recordsize」にデータベースレコードのサイズ以上の値を設定すると、パフォーマンスが大きく向上することがあります。このプロパティーを汎用目的のファイルシステムに使用

	することは、パフォーマンスが低下する可能性があるため、できるだけ避けてください。 指定するサイズは、512 バイト～128K バイトの 2 の累乗にしてください。 ファイルシステムの `recordsize` を変更すると、その後に作成されたファイルだけが影響を受けます。既存のファイルに影響はありません。 このプロパティーは、列名の短縮形「recsize」を使用しても参照できます。
mountpoint=*path* | *none* | *legacy*	このファイルシステムで使用されるマウントポイントを制御します。使用方法の詳細は、「マウントポイント」の節を参照してください。 ファイルシステムの mountpoint プロパティーを変更すると、そのマウントポイントを継承するファイルシステムおよびそのすべての子がマウント解除されます。新しい値が「legacy」である場合、マウント解除された状態が継続します。それ以外のときは、プロパティーの古い値が legacy または none だった場合、またはプロパティーが変更される前にマウントされていた場合は、新しい場所で自動的に再マウントされます。また、共有されていたすべてのファイルシステムは、共有が解除されてから新しい場所で共有されます。
sharenfs=*on* | *off* | *opts*	ファイルシステムを NFS 経由で共有するかどうか、および使用するオプションを制御します。「sharenfs」プロパティーが「off」であるファイルシステムは、share(1M)、unshare(1M)、`dfstab`(4) などの従来の

ツールを使って管理します。これらのツールを使って管理しない場合、ファイルシステムは「zfs share」および「zfs unshare」コマンドにより自動的に共有および共有解除されます。このプロパティーを「on」に設定すると、share(1M) コマンドがオプションなしで呼び出されます。「on」に設定しない場合、share(1M) コマンドの呼び出し時に、このプロパティーの内容と等価なオプションが使用されます。

データセットの「sharenfs」プロパティーが変更されると、プロパティーが以前に「off」に設定されていたか、変更前に共有されていた場合にのみ、そのデータセットおよびそのプロパティーを継承するすべての子により新しいオプションが再度共有されます。新規プロパティーが「off」の場合、ファイルシステムは共有を解除されます。

shareiscsi=*on* | *off*

「shareiscsi」は、「sharenfs」プロパティーと同様に、ZFS ボリュームが iSCSI ターゲットとしてエクスポートされるかどうかを示します。このプロパティーで使用可能な値は、「on」、「off」、および「type=disk」です。デフォルト値は「off」です。将来は、その他のターゲットタイプもサポートされる可能性があります。たとえば「tape」などです。

ファイルシステムに「shareiscsi=on」を設定して、そのファイルシステム内のすべての ZFS ボリュームがデフォルトで共有されるようにしたい場合があるかもしれません。しかし、ファイルシステム上でこのプロパティーを設定しても、直接的な効果はありません。

checksum=*on*|*off*|*fletcher2*,|*fletcher4*|*sha256*	データの完全性を検証するために使用するチェックサムを制御します。デフォルト値「on」では、適切なアルゴリズムが自動的に選択されます。 現在、アルゴリズムは*fletcher2*ですが、将来のリリースで変更される可能性があります。値が「off」の場合、ユーザーデータの完全性チェックが無効になります。チェックサムの無効化は、推奨されていない操作です。
compression=*on*|*off*|*lzjb*	このデータセットで使用される圧縮アルゴリズムを制御します。現在のところ、存在するアルゴリズムは「*lzjb*」だけですが、将来のリリースでは変更される可能性があります。デフォルト値は「off」です。 このプロパティーは、列名の短縮形「compress」を使用しても参照できます。
atime=*on*|*off*	ファイルを読み取るときにファイルのアクセス時刻を更新するかどうかを制御します。このプロパティーをオフに設定すると、ファイルを読み取るときに書き込みトラフィックが生成されなくなるため、パフォーマンスが大幅に向上する可能性があります。ただし、メールプログラムなどのユーティリティーが予期しない動作をすることがあります。デフォルト値は「on」です。
devices=*on*|*off*	このファイルシステムでデバイスノードを開くことができるかどうかを制御します。デフォルト値は「on」です。
exec=*on*|*off*	このファイルシステム内部からプロセスを実行可能かどうかを制御します。デフォルト値は「on」です。

setuid=*on* | *off*	設定された UID ビットが、このシステムで順守されるかどうかを制御します。デフォルト値は「on」です。
readonly=*on* | *off*	このデータを変更できるかどうかを制御します。デフォルト値は「off」です。 このプロパティーは、列名の短縮形「rdonly」を使用しても参照できます。
zoned=*on* | *off*	データセットを非大域ゾーンから管理するかどうかを制御します。詳細は、「ゾーン」の節を参照してください。デフォルト値は「off」です。
snapdir=*hidden* | *visible*	ファイルシステムのルートで、「.zfs」ディレクトリを非表示にするか、表示するかを制御します。詳細は、「スナップショット」の節を参照してください。デフォルト値は「hidden」です。
aclmode=`discard` | `groupmask` | `passthrough`	`chmod`(2) の実行時の ACL の変更方法を制御します。「aclmode」プロパティーが「`discard`」であるファイルシステムでは、ファイルのモードを表さない ACL エントリがすべて削除されます。「aclmode」プロパティーの「`groupmask`」(デフォルト) は、ユーザーまたはグループのアクセス権を低下させます。アクセス権は、グループアクセス権ビットと同程度にまで低下します。ただし、アクセス権がファイルまたはディレクトリの所有者と同じ UID を持つユーザーエントリである場合を除きます。この場合、ACL アクセス権は、所有者のアクセス権ビットと同程度にまで引き下げられます。「aclmode」プロパティーが「`passthrough`」であるファイルシステムでは、ファイルまたはディレクトリの新規モードを表す必須の ACL エントリを生成する以外、ACL に変更は加えられません。

aclinherit=discard | noallow | secure | passthrough

ファイルとディレクトリが作成されるときに ACL エントリをどのように継承するかを制御します。「aclinherit」プロパティーが「discard」であるファイルシステムは、 ACL エントリを一切継承しません。「aclinherit」プロパティーが「noallow」であるファイルシステムは、「deny」アクセス権を指定する継承可能な ACL エントリだけを継承します。プロパティー値「secure」(デフォルト) は、ACL エントリの継承時に「write_acl」および「write_owner」アクセス権を削除します。「aclinherit」プロパティー値が「passthrough」であるファイルシステムは、継承時に ACL エントリに加えられた変更を除く、継承可能なすべての ACL エントリを継承します。

canmount=on | off

このプロパティーが「off」に設定されている場合、ファイルシステムはマウントできず、「zfs mount -a」を実行しても無視されます。これは、「mountpoint」プロパティーを「none」に設定することに似ていますが、継承可能な通常の「mountpoint」プロパティーをデータセットが引き続き保持する点が異なります。これによってデータセットを、もっぱらプロパティー継承の機構として使用できるようになります。1 つの使用例として、論理的に分けられた 2 つのデータセットが同じマウントポイントを保持するようにできます。これにより、両方のデータセットの子は、同じディレクトリ内に表示されますが、継承する特性は異なります。デフォルト値は「on」です。

このプロパティーは継承されません。

xattr=on | off	このファイルシステムで拡張属性が有効かどうかを制御します。デフォルト値は「on」です。

iscsioptions

この読み取り専用の隠しプロパティーは、iSCSI ターゲットデーモンにより IQN などの持続的情報の格納に使用されます。zfs コマンドを使って、これを表示または変更することはできません。この内容は、外部のコンシューマを対象としたものではありません。

一時的なマウントポイントプロパティー

ファイルシステムのマウント時に、mount(1M) (従来のマウント)、または「zfs mount」コマンド (通常のファイルシステム) を使用して、プロパティーに合わせたマウントオプションが設定されます。プロパティーとマウントオプションは、次のような関係になっています。

```
PROPERTY                MOUNT OPTION
 devices                 devices/nodevices
 exec                    exec/noexec
 readonly                ro/rw
 setuid                  setuid/nosetuid
 xattr                   xattr/noxattr
```

さらに、-o オプションを使って、ディスクに格納されたプロパティーに影響を及ぼすことなく、これらのオプションをマウントごとに設定できます。コマンド行に指定した値により、データセットに格納された値が上書きされます。-nosuid オプションは、「nodevices,nosetuid」の別名です。これらのプロパティーは、「zfs get」コマンドにより、「temporary」とレポートされます。データセットのマウント時にプロパティーが変更されると、新規設定により一時設定がすべて上書きされます。

ユーザープロパティー

ZFS は、標準のネイティブプロパティーに加えて、任意のユーザープロパティーもサポートします。ユーザープロパティーは ZFS の動作には影響を与えませんが、アプリケーションや管理者が使用して、データセットに注釈を付けることができます。

ユーザープロパティー名には、ネイティブプロパティーと区別するためにコロン (「:」) 文字を含める必要があります。ユーザープロパティーに含めることができるのは、小文字の英字、数字、および句読文字のコロン (「:」)、ダッシュ (「-」)、ピリオド (「.」)、および下線 (「_」) です。想定されている規則では、プロパティー名は 2 つの部分に分割します (例:「*module*:*property*」)。ただし、この名前空間は ZFS によって強制されているものではありません。ユーザープロパティー名には、最大 256 文字を使用できます。名前の先頭にダッシュ (「-」) を付けることはできません。

ユーザープロパティーをプログラムで使用する場合、プロパティー名の *module* 要素には、逆順にした DNS ドメイン名を使用することを強くお勧めします。これは、それぞれ単独で開発された 2 つのパッケージが、異なる目的で同じプロパ

ティー名を使用する可能性を減らすためです。「com.sun.」で始まるプロパティー名は、Sun Microsystems が使用するために予約されています。

ユーザープロパティーの値は任意の文字列であり、常に継承されます。また、決して検証されることがありません。プロパティー上で動作するコマンド (「zfs list」、「zfs get」、「zfs set」など) はすべて、ネイティブプロパティーとユーザープロパティーの両方の操作に使用できます。ユーザープロパティーをクリアーするには、「`zfs inherit`」コマンドを使用します。プロパティーがどの親データセット内でも定義されていない場合は、そのプロパティー全体が削除されます。プロパティー値は、1024 文字以内に制限されています。

スワップまたはダンプデバイスとしてのボリュームの使用

スワップ領域を設定するには、特定のサイズの ZFS ボリュームを作成してから、そのデバイスでスワップを有効にします。詳細は、「例」の節を参照してください。

ZFS ファイルシステム上のファイルには、スワップを作成しないでください。ZFS のスワップファイル設定は、サポートされていません。

ZFS ボリュームをダンプデバイスとして使用する方法は、サポートされていません。

サブコマンド

状態を変更するサブコマンドはすべて、元の形式でプールに永続的に記録されます。

`zfs ?`
ヘルプメッセージを表示します。

`zfs create` [[`-o` property=value]...] *filesystem*
新しい ZFS ファイルシステムを作成します。ファイルシステムは、親から継承した「mountpoint」プロパティーに従って自動的にマウントされます。

`-o` property=value
データセットの作成時に「`zfs set property=value`」が呼び出された場合と同様に、指定されたプロパティーを設定します。編集可能なすべての ZFS プロパティーは、作成時にも設定可能です。複数の `-o` オプションを指定できます。複数の `-o` オプション内で同じプロパティーを指定した場合は、エラーが発生します。

`zfs create` [`-s`] [`-b` *blocksize*] [[`-o` property=value]...] `-V` *size volume*
指定したサイズのボリュームを作成します。ボリュームは、`/dev/zvol/{dsk,rdsk}/`*path* 内にブロックデバイスとしてエクスポートされます。ここで、*path* は ZFS 名前空間内のボリュームの名前です。このサイズは、デバイスによりエクスポートされる論理サイズを表します。デフォルトでは、同サイズの予約が作成されます。

size は、ボリュームが *blocksize* に関係なく整数のブロックを持つように、もっとも近い 128K バイトに自動的に切り上げられます。

`-s` 予約なしで疎ボリュームを作成します。疎ボリュームの詳細は、「ネイティブプロパティー」の節の「volsize」を参照してください。

`-o` property=value データセットの作成時に「`zfs set property=value`」が呼び出された場合と同様に、指定されたプロパティーを設定します。編集可能なすべての ZFS プロパティーは、作成時にも設定可能です。複数の `-o` オプションを指定できます。複数の `-o` オプション内で同じプロパティーを指定した場合は、エラーが発生します。

`-b` *blocksize* 「`-o volblocksize=`*blocksize*」と同等です。このオプションを「`-o volblocksize`」と組み合わせて指定した場合の動作は、定義されていません。

`zfs destroy` [`-rRf`] *filesystem*|*volume*|*snapshot*

指定されたデータセットを破棄します。デフォルトでは、このコマンドは共有中のファイルシステムすべての共有を解除し、マウント中のファイルシステムすべてをマウント解除し、アクティブな依存関係 (子、スナップショット、クローン) を持つデータセットの破棄を拒否します。

`-r` すべての子を再帰的に破棄します。スナップショットが指定された場合、子孫ファイルシステム内でこの名前を持つすべてのスナップショットを破棄します。

`-R` ターゲット階層外にあるクローンファイルシステムを含む、すべての依存関係を再帰的に破棄します。スナップショットが指定された場合、子孫ファイルシステム内でこの名前を持つすべてのスナップショットを破棄します。

`-f` 「`unmount -f`」コマンドを使用してすべてのファイルシステムを強制的にマウント解除します。非ファイルシステムやマウント解除されたファイルシステムは、このオプションの影響を受けません。

`-r` や `-f` オプションを適用すると、プールのかなりの部分を破棄することが可能で、使用中のマウントされたファイルシステムで予期しない動作が発生する場合があるため、これらのオプションは特に注意深く使用するようにしてください。

`zfs clone` *snapshot filesystem*|*volume*

指定したスナップショットのクローンを作成します。詳細は、「クローン」の節を参照してください。ターゲットのデータセットは、ZFS 階層内の任意の場所に配置できます。作成されたデータセットは元のデータセットと同タイプになります。

`zfs promote` *filesystem*

特定のクローンファイルシステムへの移行を促し、そのファイルシステムが「元」のスナップショットに依存しないようにします。これにより、そのク

ローンの作成元のファイルシステムを破棄できるようになります。クローンの親子依存関係が逆転し、「元」のファイルシステムが、指定されたファイルシステムのクローンになります。

クローンされたスナップショットと、そのスナップショットより前のすべてのスナップショットは、移行を促されたクローンによって所有されるようになります。それらが使用する領域は、「元」のファイルシステムから移行を促されたクローンに移されます。これは、それらのスナップショットを収容するための領域が不足しないようにするためです。この操作を行なっても新しい領域は消費されませんが、領域のアカウンティングは調整されます。移行を促されたクローンは、自身の衝突するスナップショット名を持ってはいけません。「rename」サブコマンドを使えば、衝突するスナップショット名を変更できます。

`zfs rename` *filesystem*|*volume*|*snapshot filesystem*|*volume*|*snapshot*
指定したデータセットの名前を変更します。スナップショットを除き、新規ターゲットはZFS階層内の任意の場所に配置できます。スナップショットの名前を変更できるのは、親のファイルシステムまたはボリューム内だけです。スナップショットの名前を変更する場合、スナップショットの親ファイルシステムを2番目の引数として指定する必要はありません。名前の変更されたファイルシステムは、新しいマウントポイントを継承できます。この場合、ファイルシステムは、マウント解除されてから新しいマウントポイントで再マウントされます。

`zfs snapshot` [`-r`] *filesystem@name*|*volume@name*
指定した名前のスナップショットを作成します。詳細は、「スナップショット」の節を参照してください。

`-r` すべての子孫データセットのスナップショットを再帰的に作成します。スナップショットは原子的に取得されるため、再帰的スナップショットはすべて同じ時点のものになります。

`zfs rollback` [`-rRf`] *snapshot*
指定したデータセットを以前のスナップショットにロールバックします。データセットをロールバックすると、スナップショット作成時から変更されたすべてのデータは破棄され、データセットがスナップショット作成時の状態に戻ります。デフォルトでは、このコマンドを使って、最新のスナップショット以外のスナップショットにロールバックすることはできません。最新でないスナップショットにロールバックする場合は、`-r`オプションを使って中間スナップショットをすべて破棄する必要があります。このファイルシステムは、必要に応じて、マウント解除および再マウントされます。

`-r` 指定したスナップショット以降のスナップショットをすべて再帰的に破棄します。

`-R` 指定したスナップショット以降のスナップショットを再帰的に破棄すると共に、これらのスナップショットのクローンもすべて破棄します。

`-f` 「`unmount -f`」コマンドを使用してすべてのファイルシステムを強制的にマウント解除します。

`zfs list` [`-rH`] [`-o` *prop*[,*prop*] ...] [`-t` *type*[,*type*] ...] [`-s` *prop* [`-s` *prop*] ...] [`-S` *prop* [`-S` *prop*] ...] *filesystem*|*volume*|*snapshot*|*/pathname*|*. /pathname* ...

指定したデータセットのプロパティー情報を、表形式で表示します。指定した場合、プロパティー情報を絶対パス名または相対パス名で表示できます。デフォルトでは、すべてのデータセットが表示されます。次のフィールドが含まれます。

```
name,used,available,referenced,mountpoint
```

`-H` スクリプティングモードで使用します。ヘッダーを出力せず、フィールドを任意の空白文字ではなく単一のタブで区切ります。

`-r` データセットのすべての子をコマンド行に再帰的に表示します。

`-o` *prop* 表示するプロパティーのコンマ区切りのリスト。プロパティーは、「ネイティブプロパティー」の節で示したプロパティーのいずれか、またはデータセット名を表示する特殊な値「name」を使用する必要があります。

`-s` *prop* プロパティーの値に基づいて、出力を列で昇順にソートする場合に使用するプロパティー。プロパティーは、「プロパティー」の節で示したプロパティーのいずれか、またはデータセット名をソートする特殊な値「name」にする必要があります。複数の `-s` プロパティーオプションを使用することで、一度に複数のプロパティーを指定できます。複数の `-s` オプションは、左から右の順で、重要度の高さが判断されます。

次に、ソート基準の一覧を示します。

- 数値型は、数値順にソートされる。
- 文字列型は、アルファベット順にソートされる。
- 行に適さない型は、指定したソート順に関係なく、その行がリテラルソート順の一番下になる。
- ソートオプションを指定しない場合は、「`zfs list`」の既存の動作が維持される。

`-S` *prop* `-s` オプションと同じですが、プロパティーで降順にソートされる点が異なります。

`-t` *type* 表示するタイプのコンマ区切りのリスト。ここで、「type」は、「filesystem」、「snapshot」、「volume」のいずれかになります。たとえば、「`-t snapshot`」を指定すると、スナップショットだけが表示されます。

`zfs set` *property=value filesystem*|*volume* ...

プロパティーを、各データセット用に指定した値に設定します。一部のプロパティーのみ、編集可能です。設定可能なプロパティーおよび有効な値については、「プロパティー」の節を参照してください。数値は、正確な値を指定することも、サフィックス「B」、「K」、「M」、「G」、「T」、「P」、「E」、

「Z」(それぞれ、バイト、キロバイト、メガバイト、ギガバイト、テラバイト、ペタバイト、エクサバイト、ゼタバイトを表す) を使って人間の読み取り可能な形式で指定することもできます。スナップショットにプロパティーを設定することはできません。

`zfs get` [`-rHp`] [`-o` *field*[,*field*]...] [`-s` *source*[,*source*]...] *all* | *property*[,*property*]... *filesystem*|*volume*|*snapshot* ...

指定したデータセットのプロパティーを表示します。データセットが指定されていない場合、このコマンドはシステムのすべてのデータセットのプロパティーを表示します。プロパティーごとに、次の列が表示されます。

```
name      Dataset name
property  Property name
value     Property value
source    Property source. Can either be local, default,
          temporary, inherited, or none (-).
```

デフォルトではすべての列が表示されます。これは、`-o` オプションを使って制御できます。このコマンドには、「ネイティブプロパティー」および「ユーザープロパティー」で説明されているコンマ区切りのプロパティーリストを指定できます。

特殊な値「all」を使って、指定したデータセットのプロパティーをすべて表示できます。

`-r` 任意の子のプロパティーを再帰的に表示します。

`-H` スクリプトによる解析がより容易な形式で、出力を表示します。ヘッダーがすべて省略され、フィールドが任意の数の空白ではなく、タブ 1 つで明示的に区切られます。

`-o` *field* 表示する列のコンマ区切りのリスト。デフォルト値は、「name,property,value,source」です。

`-s` *source* 表示するソースのコンマ区切りのリスト。このリストにないソースからのプロパティーは、無視されます。各ソースは、「local、default、inherited、temporary、none」のいずれかでなければなりません。デフォルト値はすべてのソースです。

`-p` 解析可能な (絶対) 値で数を表示します。

`zfs inherit` [`-r`] *property filesystem*|*volume* ...

指定したプロパティーをクリアして、そのプロパティーが祖先から継承されるようにします。祖先にプロパティーが設定されていない場合は、デフォルト値が使用されます。デフォルト値のリスト、および継承可能なプロパティーの詳細については、「プロパティー」の節を参照してください。

`-r` すべての子で指定したプロパティーを再帰的に継承します。

`zfs mount`
現在マウントされているすべての ZFS ファイルシステムを表示します。

`zfs mount`[`-o` *opts*] [`-O`] `-a`
使用可能なすべての ZFS ファイルシステムをマウントします。これは、ブートプロセスの一部として自動的に呼び出されます。

`-o` *opts* マウント時に一時的に使用する、コンマ区切りのマウントオプションリスト (省略可能)。詳細は、「一時的なマウントポイントプロパティー」の節を参照してください。

`-O` オーバーレイマウントを実行します。詳細は、mount(1M) を参照してください。

`zfs mount` [`-o` *opts*] [`-O`] *filesystem*
特定の ZFS ファイルシステムをマウントします。ファイルシステムは作成時または mountpoint プロパティーの変更時に自動的にマウントされるため、通常、これは必要ありません。詳細は、「マウントポイント」の節を参照してください。

`-o` *opts* マウント時に一時的に使用する、コンマ区切りのマウントオプションリスト (省略可能)。詳細は、「一時的なマウントポイントプロパティー」の節を参照してください。

`-O` オーバーレイマウントを実行します。詳細は、mount(1M) を参照してください。

`zfs unmount -a`
現在マウントされているすべての ZFS ファイルシステムをマウント解除します。これは、シャットダウンプロセスの一部として自動的に呼び出されます。

`zfs unmount` [`-f`] *filesystem*|*mountpoint*
指定したファイルシステムをマウント解除します。このコマンドには、システム上の ZFS ファイルシステムのマウントポイントのパスを指定することもできます。

`-f` ファイルシステムを、使用中であっても強制的にマウント解除します。

`zfs share -a`
使用可能なすべての ZFS ファイルシステムを共有します。これは、ブートプロセスの一部として自動的に呼び出されます。

`zfs share` *filesystem*
「sharenfs」プロパティーに従って、特定の ZFS ファイルシステムを共有します。ファイルシステムは、「sharenfs」プロパティーの設定時に共有されます。

`zfs unshare -a`
現在共有されているすべての ZFS ファイルシステムを共有解除します。これは、シャットダウンプロセスの一部として自動的に呼び出されます。

`zfs unshare` [`-F`] *filesystem*|*mountpoint* 指定したファイルシステムの共有を解除します。このコマンドには、システム上で共

有される ZFS ファイルシステムのパスを指定することもできます。

`-F` ファイルシステムを、使用中であっても強制的に共有解除します。

`zfs send` [`-i` *snapshot1*] *snapshot2*

snapshot2 のストリーム表現を作成します。これは、標準出力に書き込まれます。出力は、`ssh`(1) などを使用して、ファイルまたは別のマシンにリダイレクトできます。デフォルトでは、完全なストリームが生成されます。

`-i` *snapshot1* *snapshot1* から *snapshot2* への増分ストリームを生成します。増分ソース *snapshot1* は、スナップショット名の最後の構成要素 (たとえば、「@」のあとの部分) として指定できます。これは、*snapshot2* と同じファイルシステムに由来すると見なされます。

ストリームの形式は、発展途上にあります。下位互換性は保証されていません。ZFS の将来のバージョンでは、使用しているストリームを受信できなくなる可能性があります。

`zfs receive` [`-vnF`] *filesystem*|*volume*|*snapshot*
`zfs receive` [`-vnF`] `-d` *filesystem*

スナップショットを作成します。内容は、標準入力から受信したストリームの指定に基づきます。ストリーム全体を受信する場合、新しいファイルシステムも作成されます。ストリームは「`zfs send`」サブコマンドによって作成されます。このサブコマンドはデフォルトで、完全なストリームを作成します。「`zfs receive`」の別名として、「`zfs recv`」を使用できます。

増分ストリームが受信された場合、対象となるファイルシステムがすでに存在しており、かつその最新のスナップショットがその増分ストリームのソースに一致する必要がありま

す。対象となるファイルシステムおよびその子ファイルシステムがすべてマウント解除されるため、受信操作中にアクセスすることはできません。

このサブコマンドが作成するスナップショット (ストリーム全体を受信する場合はファイルシステムも) の名前は、引数タイプおよび -d オプションによって決まります。

引数がスナップショット名の場合には、指定した *snapshot* が作成されます。引数がファイルシステム名またはボリューム名の場合には、送信されたスナップショットと同名のスナップショットが、指定した *filesystem* または *volume* 内部で作成されます。-d オプションを指定すると、指定した *filesystem* に送信したスナップショットの名前を追加してスナップショット名が決定されます。-d オプションを指定すると、指定したファイルシステム内のすべての必須ファイルシステムが作成されます。

-d 前の段落で説明したように、送信したスナップショットの名前を使って、新規スナップショットの名前が決定されます。

-v ストリームおよび受信操作の所要時間に関する詳細な情報を出力します。

-n ストリームを実際には受信しません。これを -v オプションと組み合わせて使用すると、受信操作で使用する名前を決定するのに役に立ちます。

-F 受信操作を実行する前に、*filesystem* のロールバックを最

新のスナップショットに強制します。

使用例

例 1 ZFS ファイルシステム階層を作成する

次のコマンドは、「pool/home」という名前のファイルシステムと、「pool/home/bob」という名前のファイルシステムを作成します。マウントポイント「/export/home」が、親のファイルシステムに対して設定され、子のファイルシステムがそれを自動的に継承します。

```
# zfs create pool/home
# zfs set mountpoint=/export/home pool/home
# zfs create pool/home/bob
```

例 2 ZFS スナップショットを作成する

次のコマンドは、「yesterday」という名前のスナップショットを作成します。このスナップショットは、要求に応じて、「pool/home/bob」ファイルシステムのルートにある「.zfs/snapshot」ディレクトリにマウントされます。

```
# zfs snapshot pool/home/bob@yesterday
```

例 3 複数のスナップショットの作成と破棄

次のコマンドは、「pool/home」とその子孫ファイルシステムの、「yesterday」という名前のスナップショットを作成します。各スナップショットは必要に応じて、そのファイルシステムのルートの「.zfs/snapshot」ディレクトリにマウントされます。2 つ目のコマンドは、新しく作成されたスナップショットを破棄します。

```
# zfs snapshot -r pool/home@yesterday
# zfs destroy -r pool/home@yesterday
```

例 4 圧縮を無効にする

次のコマンドは、「pool/home」以下のすべてのファイルシステムで圧縮を無効にしますが、「pool/home/anne」に対しては明示的に有効にします。

```
# zfs set compression=off pool/home
# zfs set compression=on pool/home/anne
```

例 5 ZFS データセットを一覧表示する

次のコマンドは、システム内のアクティブなファイルシステムをすべて一覧表示します。

例 5 ZFS データセットを一覧表示する *(続き)*

```
# zfs list
```

```
NAME                     USED  AVAIL  REFER  MOUNTPOINT
pool                     100G   60G       -  /pool
pool/home                100G   60G       -  /export/home
pool/home/bob             40G   60G     40G  /export/home/bob
pool/home/bob@yesterday    3M     -     40G  -
pool/home/anne            60G   60G     40G  /export/home/anne
```

例 6 ZFS ファイルシステムに割り当てを設定する

次のコマンドは、「pool/home/bob」に 50G バイトの割り当てを設定します。

```
# zfs set quota=50G pool/home/bob
```

例 7 ZFS プロパティーを一覧表示する

次のコマンドは、「pool/home/bob」のプロパティーをすべて一覧表示します。

```
# zfs get -o property,value,source all pool/home/bob
```

```
PROPERTY       VALUE                      SOURCE
type           filesystem                 -
creation       Tue Feb 14 15:51 2006      -
used           571K                       -
available      50.0G                      -
referenced     376K                       -
compressratio  1.00x                      -
mounted        yes                        -
quota          50G                        local
reservation    none                       default
recordsize     128K                       default
mountpoint     /pool/home/bob             default
sharenfs       off                        default
shareiscsi     off                        default
checksum       on                         default
compression    on                         local
atime          on                         default
devices        on                         default
exec           on                         default
setuid         on                         default
readonly       off                        default
zoned          off                        default
```

例 7　ZFS プロパティーを一覧表示する　　(続き)

```
    snapdir         visible                 default
    aclmode         groupmask               default
    aclinherit      secure                  default
    xattr           on                      default
```

次のコマンドは、プロパティー値を 1 つ取得します。

```
# zfs get -H -o value compression pool/home/bob
on
```

次のコマンドは、「pool/home/bob」のローカル設定を持つプロパティーをすべて一覧表示します。

```
# zfs get -r -s local -o name,property,value all pool/home/bob

   NAME             PROPERTY      VALUE
   pool             compression   on
   pool/home        checksum      off
```

例 8　ZFS ファイルシステムをロールバックする

次のコマンドは、「pool/home/anne」の内容を「yesterday」という名前のスナップショットに戻し、中間のスナップショットをすべて削除します。

```
# zfs rollback -r pool/home/anne@yesterday
```

例 9　ZFS クローンを作成する

次のコマンドは、初期内容が「pool/home/bob@yesterday」と同一の書き込み可能なファイルシステムを作成します。

```
# zfs clone pool/home/bob@yesterday pool/clone
```

例 10　ZFS クローンの移行を促す

次の各コマンドは、クローン、クローンの移行の促進、および名前変更を実行することで、あるファイルシステムに対する変更内容をテストしたあと、その変更後のファイルシステムで元のファイルシステムを置き換える方法を示しています。

```
# zfs create pool/project/production
  populate /pool/project/production with data
# zfs snapshot pool/project/production@today
# zfs clone pool/project/production@today pool/project/beta
  make changes to /pool/project/beta and test them
# zfs promote pool/project/beta
```

例 10 ZFS クローンの移行を促す (続き)

```
# zfs rename pool/project/production pool/project/legacy
# zfs rename pool/project/beta pool/project/production
  once the legacy version is no longer needed, it can be
  destroyed
# zfs destroy pool/project/legacy
```

例 11 ZFS プロパティーを継承する

次のコマンドにより、「pool/home/bob」および「pool/home/anne」は「checksum」プロパティーを親から継承します。

```
# zfs inherit checksum pool/home/bob pool/home/anne
```

例 12 ZFS データをリモートで複製する

次のコマンドは、リモートマシンにストリーム全体を送信してから、増分ストリームを送信して、「poolB/received/fs@a」および「poolB/received/fs@b」内にそれぞれ復元します。「poolB」には、ファイルシステム「poolB/received」を含める必要があります。また、初期状態で「poolB/received/fs」を含んでいてはなりません。

```
# zfs send pool/fs@a | \
   ssh host zfs receive poolB/received/fs@a
# zfs send -i a pool/fs@b | ssh host \
   zfs receive poolB/received/fs
```

例 13 zfs receive -d オプションを使用する

次のコマンドは、「poolA/fsA/fsB@snap」のストリーム全体をリモートマシンに送信し、「poolB/received/fsA/fsB@snap」内に受信します。受信するスナップショット名の「fsA/fsB@snap」部分は、送信するスナップショットの名前に基づいて決定されます。「poolB」には、ファイルシステム「poolB/received」を含める必要があります。「poolB/received/fsA」が存在しない場合、空のファイルシステムとして作成されます。

```
# zfs send poolA/fsA/fsB@snap | \
   ssh host zfs receive -d poolB/received
```

例 14 ZFS ボリュームをスワップデバイスとして作成する

次の例は、5G バイトの ZFS ボリュームを作成し、そのボリュームをスワップデバイスとして追加する方法を示します。

例 14 ZFS ボリュームをスワップデバイスとして作成する (続き)

```
# zfs create  -V 5gb tank/vol
# swap -a /dev/zvol/dsk/tank/vol
```

例 15 ユーザープロパティーを設定する

次の例では、ユーザー定義の 「com.example:department」 プロパティーをデータセットに設定します。

```
# zfs set com.example:department=12345 tank/accounting
```

例 16 ZFS ボリュームを iSCSI ターゲットデバイスとして作成する

次の例は、ZFS ボリュームを iSCSI ターゲットとして作成する方法を示します。

```
# zfs create -V 2g pool/volumes/vol1
 # zfs set shareiscsi=on pool/volumes/vol1
 # iscsitadm list target
 Target: pool/volumes/vol1
 iSCSI Name:
 iqn.1986-03.com.sun:02:7b4b02a6-3277-eb1b-e686-a24762c52a8c
 Connections: 0
```

iSCSI ターゲットの作成後に、iSCSI イニシエータを設定します。Solaris iSCSI イニシエータの詳細は、『Solaris のシステム管理 (デバイスとファイルシステム)』を参照してください。

例 17 ZFS ファイルシステムに sharenfs プロパティーオプションを設定する

次のコマンドでは、tank/home ファイルシステムに「sharenfs」プロパティーオプションを設定することで、IP アドレスのセットに対して rw アクセスを有効にし、システム neo に対して root アクセスを有効にする方法を示しています。

```
# zfs set sharenfs='rw=@123.123.0.0/16,root=neo' tank/home
```

ホストの名前解決に DNS を使用している場合は、完全指定のホスト名を指定してください。

終了ステータス 次の終了ステータスが返されます。

0 正常終了。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

属性

次の属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWzfsu
インタフェースの安定性	開発中

関連項目

ssh(1), mount(1M), share(1M), unshare(1M), zonecfg(1M), zpool(1M), chmod(2), stat(2), fsync(3C), dfstab(4), attributes(5)

ZFS の Web ベース管理ツールおよびその他の ZFS 機能を使用するための情報については、『Solaris ZFS 管理ガイド』を参照してください。

名前　zoneadm – ゾーンの管理

形式

```
zoneadm -z zonename [-u uuid-match] subcommand
     [subcommand_options]

zoneadm [-R root] [-z zonename] [-u uuid-match] list
     [list_options]

zoneadm [-R root] -z zonename [-u uuid-match] mark incomplete
```

機能説明　zoneadm ユーティリティーは、システムゾーンを管理するために使用されます。ゾーンは、オペレーティングシステムによって実行時に管理されるアプリケーションコンテナです。

セキュリティ　ゾーン 0 以外のゾーンに実装されたプロセスは、すべての子プロセスを含めて、ゾーンを変更できません。

オプション　次のオプションを指定できます。

-R *root*
代替ルート (ブート環境) を指定します。このオプションは、「list」および「mark」サブコマンドとの組み合わせでのみ使用できます。

-u *uuid-match*
libuuid(3LIB) によって割り当てられる、ゾーンの一意識別子。このオプションが存在し、引数が空の文字列でない場合、UUID に一致するゾーンがあると -z オプションで指定されたゾーンの代わりに選択されます。

-z *zonename*
ゾーンの文字列識別子。

サブコマンド　破壊的な動作や作業内容の消失を伴う可能性のあるサブコマンドには、強制的にその処理を実行するために -F フラグが用意されています。端末デバイスから入力しているときに、-F フラグを指定しないでそのようなコマンドを実行した場合は、フラグを指定するかどうかを確認されます。それ以外の状況で -F フラグを指定しないでそのようなコマンドを実行した場合、その操作は許可されず、診断メッセージが標準エラーに書き出されます。ゾーンのインストールまたはアンインストールが中断した場合、ゾーンの状態は不完全なままになります。そのようなゾーンをリセットして構成済みの状態に戻すには、アンインストールを使用します。

サポートされているサブコマンドは次のとおりです。

attach [-F] [-n *path*]
attach サブコマンドは、あるシステムから切り離されたゾーンを取得し、そのゾーンを新しいシステムに接続します。したがって、detach サブコマンドを先に実行しておかないと、「接続」を行うことができません。構成済み状態の新しいゾーンが得られたら、attach サブコマンドを使用して、インストールを行う代わりにゾーンのルートを設定します。-F オプションを使えば、確認を行わずにゾーンを強制的に「インストール済み」の状態にすることができます。あるソースシステムからゾーンを適切にホストできないターゲットシステムへの移動

を行なった場合にゾーンがサポート不可能な状態に陥る危険性があるため、このオプションは注意して使用してください。-n オプションを使えば、コマンドを実行することなしに attach サブコマンドを起動することができます。このオプションは、「detach -n」サブコマンドの出力を入力として使用し、ネットワークデバイスの非互換やホストのゾーン未サポートなど、あらゆる衝突問題を特定するのに便利です。パスには「-」を指定できますが、その場合、入力が標準入力から読み取られます。

接続するゾーンは、事前に zonecfg (zonecfg(1M) を参照) コマンドを使って構成しておく必要があります。ただしそれは、「attach - n」を実行する場合には当てはまりません。

ゾーンを接続するには次のコマンドを使用します。

```
# zoneadm -z my-zone attach
```

boot [-- *boot_options*]
指定されたゾーンを起動 (アクティブに) します。

次の *boot_options* がサポートされています。

-i *altinit*
代替実行可能ファイルを原始プロセスとして選択します。*altinit* は実行可能ファイルへの有効なパスです。デフォルトの原始プロセスは init(1M) です。

-m *smf_options*
smf_options には、サービス管理機能のブート動作を制御する次の 2 つのカテゴリのオプションが含まれています。復旧オプションとメッセージオプションです。

メッセージオプションは、ブート中に smf(5) が表示するメッセージの種類と量を決定します。サービスオプションは、システムのブートに使用されるサービスを決定します。-m サブオプションの一覧については、kernel(1M) を参照してください。

-s
マイルストン svc:/milestone/single-user:default に対してのみ起動します。このマイルストーンは、init のレベル s と同等です。svc.startd(1M) および init(1M) を参照してください。

clone [-m *copy*] [-s *zfs_snapshot*] *source_zone*
既存のインストール済みゾーンをコピーすることでゾーンのインストールを行います。このサブコマンドは、ゾーンをインストールするための代替手段となります。

-m *copy*
「ZFS クローン」が可能な場合でも、このクローンを強制的にコピーにします。

-s *zfs_snapshot*
クローンのソースとして使用する ZFS スナップショットの名前を指定します。*snapshot* は、以前の「zoneadm clone」インストールから取得したソースゾーンの *snapshot* でなければいけません。

ソースゾーンを停止しないと、このサブコマンドを使用できません。

detach [-n]
指定されたゾーンを切り離します。ゾーンの切り離しは、あるシステムから別のシステムへゾーンを移動する際の、最初のステップです。ゾーンを切り離し、*zonepath* ディレクトリを新しいホストに移動したあと、ゾーンを新しいホストに接続する、というのが、完全なゾーン移行手順となります。ゾーンが切り離されると、そのゾーンは構成済みの状態になります。切り離された構成済みゾーンをインストールまたはクローンしようとするとエラーメッセージが表示され、その install または clone サブコマンドの処理を続行できなくなります。-n オプションを使えば、コマンドを実行することなしに detach サブコマンドを起動することができます。この場合、「attach -n」サブコマンドの実行に必要な情報が生成されます。このサブコマンドは、ネットワークデバイスの非互換やホストのゾーン未サポートなど、あらゆる衝突問題を特定するのに便利です。この情報は標準出力に送られますが、ファイルに保存したり、「attach -n」サブコマンドにパイプしたりすることもできます。

ゾーンを切り離すには次のコマンドを使用します。

```
# zoneadm -z my-zone detach
```

ソースゾーンを停止しないと、このサブコマンドを使用できません。

halt
指定されたゾーンを停止します。halt を指定した場合、そのゾーンの停止スクリプトは実行されません。また、ゾーンの実行時資源を削除します。

次のコマンドを使用すると、

```
zlogin zone shutdown
```

停止スクリプトが実行されてゾーンが完全に停止します。

help [*subcommand*]
一般ヘルプを表示します。*subcommand* を指定した場合は、*subcommand* に関するヘルプが表示されます。

install [-x *nodataset*] [*brand-specific options*]
指定されたゾーンをシステムにインストールします。このサブコマンドが実行される前に、自動的にゾーンの確認が行われます。この確認手順が失敗した場合は、インストールが拒否されます。verify サブコマンドを参照してください。

-x *nodataset*
ZFS ファイルシステムを作成しません。

ブランドゾーンには、ブランドのソフトウェアがゾーンにどのようにインストールされるかを管理する、追加のオプションが含まれる場合があります。ブランド固有の情報については、brands(5) を参照してください。

`list` [*list_options*]
現在のゾーンの名前、またはゾーンが表示されるように指定されている場合はそのゾーンの名前を表示します。

デフォルトでは、実行中のすべてのゾーンの一覧を表示します。このサブコマンドを `zoneadm -z` *zonename* オプションと一緒に使用した場合は、指定したゾーンの状態に関係なく、そのゾーンだけが表示されます。この場合、`-i` および `-c` オプションは許可されません。

`-i` オプションと `-c` オプションのどちらも指定されなかった場合、実行中のすべてのゾーンの一覧を表示します。

次の *list_options* がサポートされています。

`-c`
設定済みのすべてのゾーンを表示します。このオプションは、`-i` オプションよりも優先されます。

`-i`
インストール済みのすべてのゾーンを表示します。

`-p`
機械可読な出力を要求します。出力形式は行のリストです。1 つのゾーンは 1 行に出力され、各フィールドはコロンで区切られます。これらのフィールドは次のとおりです。

```
zoneid:zonename:state:zonepath:uuid:brand:ip-type
```

`zonepath` にコロンが含まれている場合、そのコロンはバックスラッシュ (“\:”) でエスケープできます。これは、環境変数 IFS を持つシェルの read(1) 関数を使用することで解析できます。*uuid* 値は、ゾーンのインストール時に libuuid(3LIB) によって割り当てられ、代替ブート環境に同じゾーンが存在する (またはリネームされた) 場合にゾーンを識別するのに役立ちます。「`zoneadm list -p`」コマンドの出力を解析するすべてのソフトウェアは、将来追加される可能性のあるすべてのフィールドを扱うことができる必要があります。

`-v` と `-p` オプションを同時に指定することはできません。`-v` と `-p` のどちらのオプションも使用しない場合は、ゾーン名だけが表示されます。

`-v`
ゾーン名、ID、現在の状態、ルートディレクトリ、ブランドタイプ、IP タイプ、およびオプションなどの冗長情報を表示します。

`-v` と `-p` オプションを同時に指定することはできません。`-v` と `-p` のどちらのオプションも使用しない場合は、ゾーン名だけが表示されます。

`mark incomplete`
インストールされているゾーンの状態を「incomplete」に変更します。このコマンドは、システムにおける管理上の変更によって、ゾーンが使用できない、または一貫性のない状態になった場合に役立つ可能性があります。この変更は (ゾーンをアンインストールしないかぎり) 取り消せません。

`move` *new_zonepath*
zonepath を *new_zonepath* に移動します。ゾーンを停止しないと、このサブコマンドを使用できません。*new_zonepath* はローカルファイルシステムでなければいけません。また、*zonepath* の通常の制約が適用されます。

`ready`
アプリケーションを実行するためにゾーンを準備します。このゾーンでユーザープロセスを開始するためのサブコマンドではありません。

`reboot`
ゾーンを再起動します。`halt` `boot` シーケンスと同等です。指定したゾーンがアクティブでない場合、このサブコマンドは失敗します。

`uninstall [-F]`
指定されたゾーンをシステムからアンインストールします。このサブコマンドを使用するときは、注意が必要です。指定されたゾーンの *zonepath* にあるすべてのファイルが削除されます。`-F` フラグを使用すれば、強制的に処理を実行できます。

`verify`
指定されたゾーンの設定を確認して、このマシンに安全にインストールできることを確認します。`resource/property` 形式による確認を次に示します。

`zonepath`
`zonepath` とその親ディレクトリが存在し、それらが適切なモードの root によって所有されていることを確認します。適切なモードとは、`zonepath` が `700` である、その親がグループ書き込み可や誰でも書き込み可でない、などの状態であることを指します。`zonepath` が NFS マウントを越えていないことを確認します。`zonepath` に「root」という名前のサブディレクトリが存在しないことを確認します。

`zonepath` が存在しなくても、`verify` が失敗することはありません。次回のインストールのときに適切なアクセス権で作成されることが警告されるだけです。それ以降の `verify` で問題が発生した場合には、失敗する可能性があります。

`zonepath` はシンボリックリンクであってはいけません。

`fs`
`fs` 資源の *type* 値を確認します。値が `proc`、`mntfs`、`autofs`、`cachefs`、または `nfs` の場合、あるいはファイルシステムに関連付けられているマウントバイナリが `/usr/lib/fs/<`*fstype*`>/mount` に存在しない場合には、エラーを報告します。

directory が相対パスの場合は、エラーになります。

また、raw に指定されているパスが相対パスの場合、または指定されたファイルシステムタイプの fsck バイナリが /usr/lib/fs/<*fstype*>/fsck に存在しない場合は、エラーになります。対応する fsck バイナリは存在していても、raw パスが指定されていない場合には、エラーになります。

net
すべての物理ネットワークインタフェースが存在することを確認します。すべてのネットワークアドレス資源は、次のいずれかになります。

- 有効な IPv4 アドレス。後続の「/」とプレフィックス長は任意です。
- 有効な IPv6 アドレス。後続の「/」とプレフィックス長は必須です。
- IPv4 アドレスに解決されるホスト名。

IPv6 アドレスに解決されるホスト名はサポートされていません。

物理インタフェース名はネットワークインタフェース名です。

ゾーンは排他的 IP か共有 IP のいずれかに構成できます。共有 IP ゾーンでは、物理プロパティーとアドレスプロパティーの両方を設定する必要があります。排他的 IP ゾーンでは、物理プロパティーを設定する必要があり、アドレスプロパティーは設定できません。

rctl
定義されている資源制御値が現在のマシン上で有効であることも確認します。つまり、特権レベルが privileged であること、制限値が現在定義されているシステム値より低いこと、および定義されているアクションが資源制御で有効になっていることを確認します。

使用例

例 1 -m オプションを使用する

次のコマンドは、-m オプションの使用法を示しています。

```
# zoneadm boot -- -m verbose
```

例 2 -i オプションを使用する

次のコマンドは、-i オプションの使用法を示しています。

```
# zoneadm boot -- -i /sbin/init
```

例 3 -s オプションを使用する

次のコマンドは、-s オプションの使用法を示しています。

```
# zoneadm boot -- -s
```

終了ステータス

次の終了ステータスが返されます。

0
正常終了。

1
エラーが発生した。

2
無効な使用法。

属性

次の属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWzoneu
インタフェースの安定性	開発中

関連項目

read(1), svcs(1), zlogin(1), zonename(1), init(1M), kernel(1M), svcadm(1M), svc.startd(1M), svc.startd(1M), zonecfg(1M), libuuid(3LIB), attributes(5), brands(5), smf(5), zones(5)

注意事項

zones(5) サービスは、サービス管理機能 smf(5) によって、次のサービス識別子として管理されます。

```
svc:/system/zones:default
```

有効化、無効化、または再起動要求など、このサービスに関する管理操作は、svcadm(1M) を使用して実行できます。サービスの状態は svcs(1) コマンドを使用して照会できます。

新しい非大域ゾーンをインストールするときには、Solaris オペレーティングシステムの新規インストールを行います。Solaris を新規インストールするときに、ユーザーとの対話は必要ありません。つまり、ユーザーの操作を必要としません。このため、大域ゾーンおよびすべての非大域ゾーンにインストールするパッケージには、要求スクリプトを含めることができません (pkgask(1M) を参照)。パッケージに要求スクリプトが含まれている場合は、非大域ゾーンを作成するときにユーザーの介入が必要になります。要求スクリプトを含むパッケージは、大域ゾーンのみに追加されます。pkgadd(1M) を参照してください。

名前　zonecfg – ゾーン構成の設定

形式

```
zonecfg -z zonename
zonecfg -z zonename subcommand
zonecfg -z zonename -f command_file
zonecfg help
```

機能説明

zonecfg ユーティリティーは、ゾーンの構成を作成および変更します。ゾーン構成は、いくつかの資源およびプロパティーで構成されます。

ユーザーインタフェースを簡素化するために、zonecfg では有効範囲の概念が利用されます。デフォルトの有効範囲は大域です。

次の形式の zonecfg コマンドは、対話形式で使用されます。

```
zonecfg -z zonename subcommand
```

zonecfg 経由で変更されたパラメータは、稼働中のゾーンには影響しません。変更を適用するには、ゾーンを再起動する必要があります。

ゾーンの作成と変更のほか、zonecfg ユーティリティーは大域ゾーンの資源管理設定を永続的に指定するためにも使用できます。

以降の文では、「rctl」を「resource control」(資源制御) の略語として使用します。resource_controls(5) を参照してください。

資源　次の資源タイプがサポートされています。

attr	汎用属性。
capped-memory	物理メモリー、スワップメモリー、およびロックされたメモリーの制限。
dataset	ZFS データセット。
dedicated-cpu	稼働している間、このゾーン専用に割り当てられるシステムプロセッサのサブセット。
device	デバイス。
fs	ファイルシステム。
inherit-pkg-dir	大域ゾーンから継承されるディレクトリ。ソフトウェアパッケージの内容がこのディレクトリに転送された場合、非大域ゾーンはそのソフトウェアパッケージを読み取り専用モードで継承します。非大域ゾーンのパッケージデータベースは、それらのパッケージを反映するために更新されます。zoneadm を使用してゾーンをインストールしたあとに、このような資源を変更または削除することはできません。
net	ネットワークインタフェース。

`rctl`	資源制御。

プロパティー　各資源タイプには、1 つまたは複数のプロパティーが割り当てられます。また、いくつかの大域プロパティー (特定の資源のプロパティーではなく、構成全体のプロパティー) も割り当てられます。

次のプロパティーがサポートされています。

(大域)	`zonename`
(大域)	`zonepath`
(大域)	`autoboot`
(大域)	`bootargs`
(大域)	`pool`
(大域)	`limitpriv`
(大域)	`brand`
(大域)	`cpu-shares`
(大域)	`max-lwps`
(大域)	`max-msg-ids`
(大域)	`max-sem-ids`
(大域)	`max-shm-ids`
(大域)	`max-shm-memory`
fs	`dir、special、raw、type、options`
inherit-pkg-dir	`dir`
net	`address、physical`
device	`match`
rctl	`name、value`
attr	`name、type、value`
dataset	`name`
dedicated-cpu	`ncpus、importance`
capped-memory	`physical、swap、locked`

これらのプロパティー名のプロパティー値は、単純値、複合値、またはリスト値で指定します。許可されるタイプはプロパティーごとに異なります。単純値は文字列で、引用符で囲むこともできます。複合値は次の構文で指定します。

(*<name>*=*<value>*,*<name>*=*<value>*,...)

ここで、各 *<value>* は単純値であり、*<name>* 文字列は与えられたプロパティー内で一意になります。リストは次の構文で指定します。

[*<value>*,...]

ここで、*<value>* は、単純値、複合値のいずれかです。値 (単純値または複合値) が 1 つだけのリストは、リスト構文を使わずにその値を指定することと等価です。つまり、「foo」は「[foo]」と等価です。リストは空でもかまいません (「[]」と表記)。

プロパティー値の解釈において、`zonecfg` は `fnmatch`(5) で指定されている正規表現に対応しています。「使用例」を参照してください。

次にプロパティータイプについて説明します。

global: zonename
: ゾーンの名前。

global: zonepath
: ゾーンのファイルシステムのパス。

global: autoboot
: システムが起動するときに、ゾーンを自動的に起動するかどうかを指定するブール値。ゾーンサービスが無効になっている場合は、このプロパティーの設定に関係なく、ゾーンは自動的に起動されません。ゾーンサービスを有効にするには、次のように `svcadm` コマンドを使用します。

```
# svcadm enable
svc:/system/zones:default
```

ゾーンサービスを無効にする場合は、`enable` を `disable` に置き換えます。svcadm(1M) を参照してください。

global: bootargs
: ゾーンの起動時に渡される引数 (オプション)。ただし、`zoneadm boot` コマンドにオプションが指定された場合は、そちらが優先されます。有効な引数は、zoneadm(1M) で説明されています。

global: pool
: 起動時にこのゾーンをバインドする資源プールの名前。このプロパティーには `dedicated-cpu` 資源との互換性はありません。

global: limitpriv
: このゾーン内のいずれかのプロセスが入手できる特権のセットの最大数。プロパ

ティーは、priv_str_to_set(3C) で説明されているコンマ区切りの特権セットの仕様で構成するようにしてください。名前の前にダッシュ (-) または感嘆符 (!) を付けると、結果として得られるセットから特権を除外できます。このコンテキストでは、特別な特権文字列「zone」はサポートされていません。特別な文字列「default」がプロパティーで最初のトークンとして出現する場合、zones(5) で説明されている、リソースとセキュリティーの分離を保持する安全な特権のセットに展開します。プロパティーが不足しているか空である場合、この同じ安全な特権のセットと同等になります。

システム管理者は、ゾーンの設定を行う際に細心の注意を払う必要があります。一部の特権は、ゾーンのブートに必須のためこの機構を介して除外できません。さらに、ゾーン内のプロセスがほかのゾーンのプロセスに過度に影響を与えてしまうため、ゾーンに対して付与することができない特権もあります。zoneadm(1M) は、ゾーンを「ブート」または「準備」しようとする際、無効な特権がゾーンの特権セットに追加されるまたはゾーンの特権セットから削除される場合を示しています。

特権の説明は、privileges(5) を参照してください。コマンド「ppriv -l」(ppriv(1) を参照) は、すべての Solaris 特権の一覧を生成します。ppriv で表示されるように特権を指定できます。privileges(5) では、特権がPRIV_*privilege_name* の形式で一覧表示されています。たとえば、このプロパティーで特権 *sys_time* を指定した場合、これは privileges(5) では PRIV_SYS_TIME として表示されています。

global: brand	ゾーンのブランドタイプ。ブランドが割り当てられていないゾーンは「ネイティブ」ゾーンとして見なされます。
global: ip-type	ゾーンは大域ゾーンと IP インスタンスを共有する (デフォルト) か、独自の排他的な IP インスタンスを持つことができます。 このプロパティーは「`shared`」および「`exclusive`」の値を取ります。
fs: dir、special、raw、type、options	ファイルシステムをマウントする方法や場所などを決めるために必要な値。mount(1M)、`mount`(2)、fsck(1m)、および `vfstab`(4) を参照してください。
inherit-pkg-dir: dir	ディレクトリパス。
net: address、physical	ネットワークインタフェースのネットワークアドレスと物理インタフェース名。ネットワークアドレスは、次のいずれかになります。 ■ 有効な IPv4 アドレス。後続の「`/`」とプレフィックス長は任意です。 ■ 有効な IPv6 アドレス。後続の「`/`」とプレフィックス長は必須です。 ■ IPv4 アドレスに解決されるホスト名。 IPv6 アドレスに解決されるホスト名はサポートされていません。 物理インタフェース名はネットワークインタフェース名です。 ゾーンは排他的 IP か共有 IP のいずれかに構成できます。共有 IP ゾーンでは、物理プロパティーとアドレスプロパティーの両方を設定する必要があります。排他的 IP ゾーンでは、物理プロパティーを設定する必要があり、アドレスプロパティーは設定できません。
device: match	照合するデバイス名。
rctl: name、value	資源制御の名前、特権、制限、アクション。`prctl`(1) および rctladm(1M) を参

	照してください。rctl の値の望ましい設定方法は、特定の rctl に関連する大域的なプロパティー名を使用する方法です。
attr: name、 type、 value	汎用属性の名前、型、および値。`type` は、`int`、`uint`、`boolean`、`string` のいずれかでなければなりません。値には、その型の値を指定する必要があります。`uint` は符号なしの (負でない) 整数です。
dataset: name	ゾーン内からアクセスする ZFS データセットの名前。zfs(1M) を参照してください。
global: cpu-shares	このゾーンに割り当てられている公平配分スケジューラ (FSS) の配分数。このプロパティーは、`zone.cpu-shares` rctl を設定するための望ましい方法です。
global: max-lwps	このゾーンの、同時に使用できる LWP の最大数。このプロパティーは、`zone.max-lwps` rctl を設定するための望ましい方法です。
global: max-msg-ids	このゾーンに許容されるメッセージキュー ID の最大数。このプロパティーは、`zone.max-msg-ids` rctl を設定するための望ましい方法です。
global: max-sem-ids	このゾーンに許容されるセマフォー ID の最大数。このプロパティーは、`zone.max-sem-ids` rctl を設定するための望ましい方法です。
global: max-shm-ids	このゾーンに許容される共有メモリー ID の最大数。このプロパティーは、`zone.max-shm-ids` rctl を設定するための望ましい方法です。
global: max-shm-memory	このゾーンに許容される共有メモリーの最大容量。このプロパティーは、`zone.max-shm-memory` rctl を設定するための望ましい方法です。この数値には、K、M、G、T の単位を適用できます。たとえば、1M は 1 メガバイトです。
dedicated-cpu: ncpus、 importance	このゾーンが排他的に使用するために割り当てられるべき CPU の数。ゾーンは、

	起動時にプールおよびプロセッサのセットを作成します。資源プールの詳細については、pooladm(1M) および poolcfg(1m) を参照してください。ncpu プロパティーには、プロセッサ数を単一の値か範囲 (たとえば 1-4) で指定できます。「importance」プロパティーは省略可能であり、これが設定されると、poold(1M) によって使用される pset.importance の値を指定します。この資源を使用する場合、ゾーンに割り当てるのに十分な数の空きプロセッサがゾーンのブート時に必要であり、足りないとゾーンはブートしません。このゾーンに割り当てられたプロセッサを、大域ゾーンやほかのゾーンが使用することはできません。この資源には pool プロパティーとの互換性はありません。この資源の単一のインスタンスのみをゾーンに追加できます。
capped-memory: physical、swap、locked	このゾーンが使用できるメモリーの上限。これらの各数値には、K、M、G、T の単位を適用できます。たとえば、1M は 1 メガバイトです。これらの各プロパティーは省略可能ですが、この資源を追加するときは少なくとも 1 つのプロパティーを設定する必要があります。この資源の単一のインスタンスのみをゾーンに追加できます。physical プロパティーは、このゾーンの max-rss を設定します。これは大域ゾーンで実行中の rcapd(1M) によって強制されます。swap プロパティーは、zone.max-swap rctl を設定するための望ましい方法です。locked プロパティーは、zone.max-locked-memory rctl を設定するための望ましい方法です。

次の表は、資源、プロパティー名、および型の一覧です。

```
resource          property-name   type
(global)          zonename        simple
(global)          zonepath        simple
(global)          autoboot        simple
```

```
(global)          bootargs        simple
(global)          pool            simple
(global)          limitpriv       simple
(global)          brand           simple
(global)          ip-type         simple
(global)          cpu-shares      simple
(global)          max-lwps        simple
(global)          max-msg-ids     simple
(global)          max-sem-ids     simple
(global)          max-shm-ids     simple
(global)          max-shm-memory  simple
fs                dir             simple
                   special         simple
                   raw             simple
                   type            simple
                   options         list of simple
inherit-pkg-dir   dir             simple
net               address         simple
                   physical        simple
device            match           simple
rctl              name            simple
                   value           list of complex
attr              name            simple
                   type            simple
                   value           simple
dataset           name            simple
dedicated-cpu     ncpus           simple or range
                   importance      simple

capped-memory    physical        simple with scale
                  swap            simple with scale
                  locked          simple with scale
```

この表について、いくつか説明を補足します。「rctl」資源タイプの複合プロパティー「value」は、名前/値の 3 つの組み合わせで構成されます。名前は「priv」、「limit」、および「action」で、それぞれ単純値を指定します。「attr」資源の「name」プロパティーの構文には、ゾーン名と同じような制限がありますが、相違点もあります。つまり、英数字で始める必要がありますが、英数字以外にハイフン (-)、下線 (_)、およびピリオド (.) も使用できます。「zone」で始まる属性名はシステム用に予約されています。また、「autoboot」大域プロパティーの値は、必ず「true」または「false」にしてください。

オプション

次のオプションを指定できます。

`-f` *command_file*
: `zonecfg` コマンドファイルの名前を指定します。*command_file* は、`zonecfg` サブコマンドを 1 行に 1 つずつ記述するテキストファイルです。

-z zonename ゾーンの名前を指定します。ゾーン名では、大文字と小文字が区別されます。ゾーン名は英数字で始める必要がありますが、英数字以外に下線 (_)、ハイフン (-)、およびピリオド (.) も使用できます。global という名前と SUNW で始まるすべての名前は、予約されているので使用できません。

サブコマンド

add および select サブコマンドを使用して、特定の資源を選択できます。選択した時点で、有効範囲がその資源に変更されます。end および cancel サブコマンドを使用して、その資源の指定を終了することができます。指定を終了した時点で、有効範囲が大域に戻ります。add、remove、set などの一部のサブコマンドでは、有効範囲ごとにセマンティクスが異なります。

破壊的な動作や作業内容の消失を伴う可能性のあるサブコマンドには、強制的にその処理を実行するために -F オプションが用意されています。端末デバイスから入力しているときに、-F オプションを指定しないでそのようなコマンドを実行した場合は、オプションを指定するかどうかを適切なタイミングで確認されます。それ以外の状況で -F オプションを指定しないでそのようなコマンドを実行した場合、その操作は許可されず、診断メッセージが標準エラーに書き出されます。

サポートされているサブコマンドは次のとおりです。

add *resource-type* (global scope)
add *property-name property-value* (resource scope)
大域有効範囲の場合は、特定の資源タイプの指定を開始します。有効範囲がその資源タイプに変わります。

資源固有の有効範囲では、指定された名前と値を持つプロパティーを追加します。プロパティー値の構文は、プロパティータイプによって異なります。通常は、単純値または単純値のリストを使用します。リストの場合は、角括弧で囲み、各値をコンマで区切ります ([foo,bar,baz])。「プロパティー」を参照してください。

cancel
資源の指定を終了し、有効範囲を大域に戻します。指定途中の資源をすべて破棄します。cancel は、資源の有効範囲だけに適用できます。

clear *property-name*
プロパティーの値を消去します。

commit
現在の構成をメモリーから安定した記憶領域に確定します。zoneadm で構成を使用するには、その構成を確定する必要があります。メモリー内構成を確定するまでは、revert サブコマンドを使って変更を取り消すことができます。zonecfg セッションが終了するときには、commit 処理が自動的に実行されます。構成が正しく設定されていないと確定できないので、この処理では自動的に構成が確認されます。

`create [-F] [ -a` *path* `|-b | -t` *template*`]`
指定されたゾーンのメモリー内構成を作成します。新しいゾーンの設定を開始するときは、`create` を使用します。このゾーンを安定した記憶領域に保存する方法については、`commit` を参照してください。

既存の構成を上書きする場合は、`-F` オプションを指定して、強制的に処理を実行します。*template* と同じ構成を作成する場合は、`-t` *template* オプションを指定します。*template* は設定済みゾーンの名前です。

新しいホストで切り離されたゾーンを設定できるようにするには、`-a` *path* オプションを使用します。*path* パラメータは、この新しいホスト上に移動されている、切り離されたゾーンのゾーンパスの位置です。切り離されたゾーンが設定されたあとは、「`zoneadm attach`」コマンド (zoneadm(1M) を参照) を使用してインストールするようにしてください。新しいゾーンのすべての検証は、ゾーンの設定中ではなく `attach` 処理中に行われます。

空の構成を作成する場合は、`-b` オプションを使用します。引数が指定されていない場合、`create` は Sun のデフォルト設定を適用します。

`delete [-F]`
指定された構成をメモリー内および安定した記憶領域から削除します。この操作はすぐに実行されるので、確定する必要はありません。削除した構成を元に戻すことはできません。

この操作を強制的に実行する場合は、`-F` オプションを使用します。

`end`
資源の指定を終了します。このサブコマンドは、資源の有効範囲だけに適用できます。`zonecfg` は、現在の資源が完全に指定されていることを確認します。完全に指定されている場合は、メモリー内構成に追加され (メモリー内構成を安定した記憶領域に保存する方法については `commit` を参照)、有効範囲は大域に戻ります。指定が完全でない場合は、対応するエラーメッセージが通知されます。

`export [-f` *output-file*`]`
標準出力に構成を出力します。*output-file* に構成を出力する場合は、`-f` オプションを使用します。このオプションを使用すると、コマンドファイルに適した形式で出力が作成されます。

`help [usage] [`*subcommand*`] [syntax] [`*command-name*`]`
一般ヘルプまたは特定項目のヘルプを出力します。

`info zonename | zonepath | autoboot | brand | pool | limitpriv`
`info [`*resource-type* `[`*property-name=property-value* `]*]`
現在の構成に関する情報を表示します。*resource-type* が指定されている場合は、そのタイプの資源についてのみ情報を表示します。*property-name* 値ペアが指定されている場合は、その条件を満たしている資源の情報だけを表示します。資源の有効範囲では、引数が無視され、`info` は現在追加または変更されている資源の情報を表示します。

`remove` *resource-type*{*property-name*=*property-value*}(global scope)
大域有効範囲では、指定された資源タイプを削除します。`[]` 構文は、0 個以上の項目を角括弧の中に指定できることを意味します。資源の単一インスタンスを削除するだけの場合、その資源が一意に識別されるように、プロパティーの名前と値ペアを十分に指定する必要があります。プロパティーの名前と値ペアを 1 つも指定しなかった場合、すべてのインスタンスが削除されます。2 つ以上のペアを指定した場合は、`-F` オプションを指定しないかぎり、確認を要求されます。

`select` *resource-type* {*property-name*=*property-value*}
指定された資源タイプのうち、指定された *property-name* と *property-value* の対の条件に一致するものを、変更対象として選択します。このサブコマンドは、大域有効範囲だけに適用できます。有効範囲がその資源タイプに変わります。`{}` 構文は、1 つまたは複数の項目をこの中に指定できることを意味します。資源が一意に識別されるように、*property-name property-value* ペアを十分に指定する必要があります。

`set` *property-name*=*property-value*
指定されたプロパティー名を、指定された値に設定します。プロパティーには、`zonename` や `zonepath` のような大域プロパティーと、資源固有のプロパティーがあります。このサブコマンドは、大域有効範囲と資源固有の有効範囲の両方で使用できます。

`verify`
現在の構成が次の点で正しいかどうかを確認します。

- 各資源に必須プロパティーがすべて指定されていること。
- `zonepath` が指定されていること。

`revert [-F]`
構成を、最後に確定されたときの状態に戻します。この操作を強制的に実行する場合は、`-F` オプションを使用します。

`exit [-F]`
`zonecfg` のセッションを終了します。必要な場合は、確定操作が自動的に試行されます。EOF 文字を使用して `zonecfg` を終了することもできます。この操作を強制的に実行する場合は、`-F` オプションを使用します。

使用例

例 1 新しいゾーンの環境を作成する

次の例では、`zonecfg` を使って新しいゾーンの環境を作成します。`/usr/local` は、大域ゾーンから `/opt/local` にマウントされるループバックです。`/opt/sfw` は、大域ゾーンからマウントされるループバックです。3 つの論理ネットワークインタフェースが追加されています。公平配分スケジューラ (fair-share scheduler、FSS) による CPU 共有の数に対する制限が、`rctl` 資源タイプを使ってゾーンに設定されています。この例では、特定の資源を変更するために選択する方法も示しています。

```
example# zonecfg -z myzone3
my-zone3: No such zone configured
Use ’create’ to begin configuring a new zone.
```

例 1 新しいゾーンの環境を作成する (続き)

```
zonecfg:myzone3> create
zonecfg:myzone3> set zonepath=/export/home/my-zone3
zonecfg:myzone3> set autoboot=true
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/usr/local
zonecfg:myzone3:fs> set special=/opt/local
zonecfg:myzone3:fs> set type=lofs
zonecfg:myzone3:fs> add options [ro,nodevices]
zonecfg:myzone3:fs> end
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/mnt
zonecfg:myzone3:fs> set special=/dev/dsk/c0t0d0s7
zonecfg:myzone3:fs> set raw=/dev/rdsk/c0t0d0s7
zonecfg:myzone3:fs> set type=ufs
zonecfg:myzone3:fs> end
zonecfg:myzone3> add inherit-pkg-dir
zonecfg:myzone3:inherit-pkg-dir> set dir=/opt/sfw
zonecfg:myzone3:inherit-pkg-dir> end
zonecfg:myzone3> add net
zonecfg:myzone3:net> set address=192.168.0.1/24
zonecfg:myzone3:net> set physical=eri0
zonecfg:myzone3:net> end
zonecfg:myzone3> add net
zonecfg:myzone3:net> set address=192.168.1.2/24
zonecfg:myzone3:net> set physical=eri0
zonecfg:myzone3:net> end
zonecfg:myzone3> add net
zonecfg:myzone3:net> set address=192.168.2.3/24
zonecfg:myzone3:net> set physical=eri0
zonecfg:myzone3:net> end
zonecfg:my-zone3> set cpu-shares=5
zonecfg:my-zone3> add capped-memory
zonecfg:my-zone3:capped-memory> set physical=50m
zonecfg:my-zone3:capped-memory> set swap=100m
zonecfg:my-zone3:capped-memory> end
zonecfg:myzone3> exit
```

例 2 ネイティブではないゾーンを作成する

次の例では、新しい Linux ゾーンを作成します。

```
example# zonecfg -z lxzone
lxzone: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:lxzone> create -t SUNWlx
zonecfg:lxzone> set zonepath=/export/zones/lxzone
```

例 2　ネイティブではないゾーンを作成する　　(続き)

```
zonecfg:lxzone> set autoboot=true
zonecfg:lxzone> exit
```

例 3　排他的 IP ゾーンを作成する

次の例では、bge1 および bge33000 に対する排他的なアクセスが与えられており、かつ IP 層において、システム上に構成されているほかのゾーンから隔離されているゾーンを作成します。

IP アドレスおよびルーティングは、新しいゾーンの中で sysidtool(1M) を使用して構成されます。

```
example# zonecfg -z excl
excl: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:excl> create
zonecfg:excl> set zonepath=/export/zones/excl
zonecfg:excl> set ip-type=exclusive
zonecfg:excl> add net
zonecfg:excl:net> set physical=bge1
zonecfg:excl:net> end
zonecfg:excl> add net
zonecfg:excl:net> set physical=bge33000
zonecfg:excl:net> end
zonecfg:excl> exit
```

例 4　ゾーンを資源プールに関連付ける

次の例は、既存のゾーンを既存の資源プールに関連付ける方法を示しています。

```
example# zonecfg -z myzone
zonecfg:myzone> set pool=mypool
zonecfg:myzone> exit
```

資源プールの詳細については、pooladm(1M) および poolcfg(1m) を参照してください。

例 5　ゾーンの名前を変更する

次の例は、既存のゾーンの名前を変更する方法を示したものです。

```
example# zonecfg -z myzone
zonecfg:myzone> set zonename=myzone2
zonecfg:myzone2> exit
```

例 6　ゾーンの特権セットを変更する

次の例では、次回ゾーンのブート時に制限される既存のゾーンのプロセスの特権セットを変更する方法を示しています。この場合の特権セットは、システムの日付と時間を変更する特権に加え、通常ゾーンが持っている標準的な安全特権セットになります。

```
example# zonecfg -z myzone
zonecfg:myzone> set limitpriv="default,sys_time"
zonecfg:myzone2> exit
```

例 7　大域ゾーンの zone.cpu-shares プロパティーを設定する

次のコマンドでは、大域ゾーンの zone.cpu-shares プロパティーを設定します。

```
example# zonecfg -z global
zonecfg:global> set cpu-shares=5
zonecfg:global> exit
```

例 8　パターンマッチングを使用する

次のコマンドでは、zonecfg のパターンマッチングのサポートを説明しています。ゾーン flexlm で、次のように入力します。

```
zonecfg:flexlm> add device
zonecfg:flexlm:device> set match="/dev/cua/a00[2-5]"
zonecfg:flexlm:device> end
```

大域ゾーンで、次のように入力します。

```
global# ls /dev/cua
a     a000  a001  a002  a003  a004  a005  a006  a007  b
```

ゾーン flexlm で、次のように入力します。

```
flexlm# ls /dev/cua
a002  a003  a004  a005
```

終了ステータス

次の終了ステータスが返されます。

0　正常終了。

1　エラーが発生した。

2　無効な使用法。

属性

次の属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWzoneu
インタフェースの安定性	不安定

関連項目

ppriv(1), prctl(1), zlogin(1), mount(1M), pooladm(1M), poolcfg(1m), poold(1M), rcapd(1M), rctladm(1M), svcadm(1M), sysidtool(1M), zfs(1M), zoneadm(1M), priv_str_to_set(3C), vfstab(4), attributes(5), brands(5), fnmatch(5), lx(5), privileges(5), resource_controls(5), zones(5)

『Solaris のシステム管理 (Solaris コンテナ : 資源管理と Solaris ゾーン)』

注意事項

zonecfg で使用するすべての文字データは、US-ASCII エンコーディングにする必要があります。

名前

zpool – ZFS ストレージプールの構成

形式

```
zpool [-?]

zpool create [-fn] [-R root] [-m mountpoint] pool vdev ...

zpool destroy [-f] pool

zpool add [-fn] pool vdev

zpool remove pool vdev

zpool  list [-H] [-o field[,field]*] [pool] ...

zpool iostat [-v] [pool] ... [interval [count]]

zpool status [-xv] [pool] ...

zpool offline [-t] pool device ...

zpool online pool device ...

zpool clear pool [device] ...

zpool attach [-f] pool device new_device

zpool detach pool device

zpool replace [-f] pool device [new_device]

zpool scrub [-s] pool ...

zpool export [-f] pool

zpool import [-d dir] [-D]

zpool import [-d dir] [-D] [-f] [-o opts] [-R root] pool | id [newpool]

zpool import [-d dir] [-D] [-f] -a

zpool upgrade

zpool upgrade -v

zpool upgrade [-a | pool]

zpool history [pool] ...
```

機能説明

zpool コマンドは、ZFS ストレージプールを構成します。ストレージプールは、ZFS データセット用の物理ストレージおよびデータ複製を提供するデバイスの集合体です。

ストレージプール内部のすべてのデータセットは、同一の領域を共有します。データセットの管理については、zfs(1M) を参照してください。

仮想デバイス (vdevs)

「仮想デバイス」とは、特定のパフォーマンスおよびフォルト特性に従って編成された単一のデバイスまたはデバイスの集合体を指します。次の仮想デバイスがサポートされています。

disk
ブロックデバイス。通常は「/dev/dsk」内に存在します。ZFS では、個別のスライスまたはパーティションを使用することが可能です。ただし、推奨されているのは、ディスク全体を使用する操作モードです。ディスクは、フルパスで指定することも、短縮名 (「/dev/dsk」以下のパスの相対部分) を使用することもできます。スライスやパーティションの指定を省略すると、ディスク全体を指定できます。たとえば、「c0t0d0」は「/dev/dsk/c0t0d0s2」と等価です。ディスク全体を指定すると、ZFS によりディスクのラベルが必要に応じて自動的に設定されます。

file
通常のファイル。ファイルをバッキングストアとして使用することは決してお勧めできません。ファイルのフォルトトレランスは、それが存在するファイルシステムと同程度に過ぎません。これは、主に実験用途を念頭に置いて設計されています。ファイルは、フルパスで指定する必要があります。

ミラー
2 つ以上のデバイスのミラー。データの複製方法は、ミラーのすべてのコンポーネントで同じです。サイズ *X* のディスク *N* 台で構成されるミラーは、*X* バイトの格納が可能であり、(*N-1*) 台のデバイスで障害が発生してもデータの完全性が保たれます。

raidz
raidz1
raidz2
RAID-5 の一種。これを使用すると、パリティー分配の向上、および「RAID-5 書き込みホール」(電力喪失によりデータおよびパリティーに矛盾が発生する) の除去が可能になります。データおよびパリティーは、raidz グループ内の全ディスク間でストライプ化されます。

raidz グループは、シングルパリティー、ダブルパリティーのいずれかを持つことができます。これはそれぞれ、raidz グループがデータを一切失うことなく、1 つの障害または 2 つの障害に耐えられることを意味します。raidz1 vdev タイプはシングルパリティーの raidz グループを、raidz2 vdev タイプはダブルパリティーの raidz グループを、それぞれ指定します。raidz vdev タイプは raidz1 の別名です。

サイズ *X* のディスク *N* 台 (パリティーディスク *P* 台を含む) で構成される raidz グループには、約 (*N-P*)***X* バイトを格納可能です。また、1 台のデバイスに障害が発生してもデータの完全性は損なわれません。raidz グループ内のデバイスの最小数は、パリティーディスクの数に 1 を加えたものです。推奨の数は、3 から 9 までです。

spare
特定のプールで使用可能なホットスペアを追跡する特殊な擬似 vdev。詳細は、「ホットスペア」節を参照してください。

仮想デバイスを入れ子にすることはできません。このため、ミラーまたは raidz 仮想デバイスに含めることができるのは、ファイルまたはディスクだけです。ミラーをミラー (またはその他の組み合わせを使用) することはできません。

プールでは、構成の最上位に任意の数の仮想デバイス (「ルート vdevs」と呼ばれる) を含めることができます。最上位の全デバイス間でデータが動的に配分されることで、デバイス間のデータ均衡が保たれます。新しい仮想デバイスが追加されると、ZFS は使用可能になった新規デバイスにデータを自動的に格納します。

コマンド行で、複数の仮想デバイスを 1 つずつ空白で区切って指定できます。キーワード「mirror」や「raidz」は、グループが終了し、別のグループが開始することを示す場合に使用します。たとえば、次のコマンドは、それぞれが 2 台のディスクのミラーである 2 つのルート vdevs を作成します。

```
# zpool create mypool mirror c0t0d0 c0t1d0 mirror c1t0d0 c1t1d0
```

デバイスの障害と復旧

ZFS では、デバイス障害およびデータ破壊を処理するための豊富な機構がサポートされています。すべてのメタデータおよびデータのチェックサムが実行され、破壊が検出されると、ZFS により不正なデータが正常なコピーを使って自動的に修復されます。

これらの機能を活用するため、プールでミラー化グループまたは raidz グループを使って、なんらかの形式の冗長性を利用する必要があります。ZFS では、非冗長設定 (各ルート vdev は単なるディスクまたはファイル) での実行がサポートされていますが、この方法は決してお勧めできません。ビット破壊が 1 回発生するだけで、データの一部またはすべてが使用不可になる可能性があるためです。

プールの健全性状態は、次の 3 つの状態のいずれかで表されます。オンライン、機能低下、またはフォルトです。オンラインプールでは、すべてのデバイスが正常に動作しています。機能低下プールでは、1 つ以上のデバイスで障害が発生していますが、冗長性設定のためにデータを引き続き使用できます。フォルトプールでは、1 つ以上のデバイスで障害が発生しており、冗長性が不十分であるために欠落したデータを複製できません。

ホットスペア

ZFS では、デバイスを「ホットスペア」としてプールに関連付けることができます。これらのデバイスはプール内でアクティブには使用されませんが、特定のアクティブデバイスで障害が発生すると、そのデバイスは特定のホットスペアで自動的に置き換えられます。ホットスペアを含むプールを作成するには、「spare」 vdev と任意の数のデバイスを指定します。たとえば、

```
# zpool create pool mirror c0d0 c1d0 spare c2d0 c3d0
```

スペアは複数のプール間で共有できます。スペアを追加するには「zpool add」コマンドを、削除するには「zpool remove」コマンドを、それぞれ使用します。スペア置換が起動されると新しい「spare」vdev が構成内に作成されますが、元のデバイスが交換されるとそのスペアは削除されます。この時点で、別のデバイスで障害が発生すると、そのホットスペアは再度使用可能となります。

処理中のスペア置換を取り消すには、そのホットスペアを切り離します。障害の発生した元のデバイスが切り離されると、そのデバイスが構成内で占めていた位置にホットスペアが配置され、そのホットスペアがすべてのアクティブプールのスペアリストから削除されます。

代替ルートプール

「zpool create -R」および「zpool import -R」コマンドを使用すると、異なるルートパスを持つプールの作成およびインポートを実行できます。デフォルトでは、システムでプールを作成またはインポートする際、プールが永続的に追加され、システムのブート時に常に使用できるようになります。リムーバブルメディアを使用する場合や復旧を行う場合には、この動作は望ましくないこともあります。代替ルートプールは、システム上で持続しません。エクスポートされるか、システムがリブートするまでの間だけ存在します。その時点で、代替ルートプールを再度インポートする必要があります。

また、プール内のすべてのマウントポイントに、指定したルートが接頭辞として付加されるため、プールをファイルシステム内の特定の領域に制限できます。これがもっとも役立つのは、リムーバブルメディアから未知のプールをインポートする場合です。どのファイルシステムのマウントポイントも信頼することができないためです。

代替ルートプール作成時のデフォルトマウントポイントは、通常のデフォルト「/*pool*」ではなく「/」です。

サブコマンド

状態を変更するサブコマンドはすべて、元の形式でプールに永続的に記録されます。

`zpool` コマンドの提供するサブコマンドを使用すると、ストレージプールの作成と破棄、ストレージプールへの容量の追加、およびストレージプールに関する情報提供を行えます。サポートされているサブコマンドは次のとおりです。

`zpool -?`
ヘルプメッセージを表示します。

`zpool create` [`-fn`] [`-R` *root*] [`-m` *mountpoint*] *pool vdev* ...
コマンド行で指定した仮想デバイスを含む新規ストレージプールを作成します。プール名の先頭は文字でなければなりません。また、プール名に含めることができるのは、英数字、下線(「_」)、ダッシュ(「-」)、およびピリオド(「.」)です。プールは予約されている「mirror」、「raidz」、および「spare」と、パターン「c[0-9]」で始まるデバイスを設定できます。`vdev` の仕様については、「仮想デバイス」の節を参照してください。

このコマンドは、指定された各デバイスがアクセス可能であり、現在別のサブシステムにより使用されていないことを確認します。ある種の用法(現在マウントされている、専用のダンプデバイスとして指定されているなど)では、ZFS によるデバイスの使用が妨げられます。`-f` オプションを使用すると、その他の用法(既存の UFS ファイルシステムの使用など)を上書きできます。

このコマンドは、プールの複製方法が一貫しているかどうかもチェックします。-f を指定せずに、単一のプール内で冗長ストレージと非冗長ストレージの結合を試みたり、ディスクとファイルの混在を試みると、エラーが発生します。-f を指定せずに、単一の raidz またはミラーグループ内でサイズの異なるデバイスを使用しても、エラーのフラグが付けられます。

-R オプションを指定しない場合の、デフォルトのマウントポイントは「/*pool*」です。このマウントポイントは存在しないか、空でなければなりません。さもないと、ルートデータセットをマウントすることはできません。これは、-m オプションを使って上書きできます。

-f
使用中と表示されていたり、競合する複製レベルが指定されていたとしても、vdev の使用を強制します。この方法で、すべてのデバイスを上書きできるわけではありません。

-n
使用する設定を、実際にプールを作成せずに表示します。実際のプール作成は、権限の不足またはデバイス共有のために失敗する可能性があります。

-R *root*
代替 *root* を使ってプールを作成します。「代替ルートプール」の節を参照してください。この操作の一環として、ルートデータセットのマウントポイントが「/」に設定されます。

-m *mountpoint*
ルートデータセットのマウントポイントを設定します。デフォルトのマウントポイントは、「/*pool*」です。このマウントポイントは絶対パス、「legacy」または「none」でなければなりません。データセットマウントポイントの詳細は、zfs(1M) を参照してください。

zpool destroy [-f] *pool*
指定したプールを破棄し、すべてのデバイスを解放してほかの用途で使用できるようにします。このコマンドは、プールを破棄する前に、アクティブなデータセットすべてのマウント解除を試みます。

-f
プール内に含まれるアクティブなデータセットすべてのマウント解除を強制的に行います。

zpool add [-fn] *pool vdev ...*
指定された仮想デバイスを指定したプールに追加します。*vdev* の仕様については、「仮想デバイス」の節を参照してください。-f オプションの動作、および実行されるデバイス検査については、「zpool create」サブコマンドを参照してください。

-f
使用中と表示されていたり、競合する複製レベルが指定されていたとしても、vdev の使用を強制します。この方法で、すべてのデバイスを上書きできるわけではありません。

`-n` vdev を実際に追加することなく、使用される設定を表示します。実際のプール作成は、権限の不足またはデバイス共有のために失敗する可能性があります。

現在設定されているディスクは、定足数デバイスとして zpool に追加しないでください。ディスクがいったん zpool に追加されると、そのディスクは定足数デバイスとして設定可能になります。

`zpool remove` *pool vdev*

指定された vdev をプールから削除します。このコマンドが現在サポートしているのは、ホットスペアの削除だけです。ミラーの一部となっているデバイスは、「zpool detach」コマンドを使用して削除できます。raidz と最上位 vdev は、プールから削除できません。

`zpool list` [`-H`] [`-o` *field*[,*field**]] [*pool*] ...

指定したプールについて、健全性状態および領域使用状況を一覧表示します。引数を指定しない場合、システム内のすべてのプールが表示されます。

`-H` スクリプトモード。ヘッダーを表示せず、フィールドを任意の空白文字ではなく単一のタブで区切ります。

`-o` *field* 表示するフィールドのコンマ区切りのリスト。各フィールドには、次のいずれかを指定する必要があります。

```
name            Pool name
size            Total size
used            Amount of space used
available       Amount of space available
capacity        Percentage of pool space used
health          Health status
```

デフォルトはすべてのフィールドです。

このコマンドにより、ストレージプールで使用可能な実際の物理容量がレポートされます。物理容量は、内部の全データセットが実際に使用可能な総容量とは異なる場合があります。raidz 設定で使用される容量は、書き込まれるデータの特性により異なります。また、ZFS は、zfs(1M) コマンドが考慮に入れる内部アカウンティングのための容量を確保します。しかし、`zpool` コマンドでは内部アカウンティングを考慮しません。容量に余裕のある適切なサイズのプールの場合、これらの影響は見えません。小さなプールまたは容量全体がほぼ使用されているプールの場合、これらの相違はより目立つようになります。

`zpool iostat` [`-v`] [*pool*] ... [*interval* [*count*]]

指定したプールの I/O 統計を表示します。間隔を指定した場合、`Ctrl-C` キーを押すまで、統計が *interval* 秒ごとに出力されます。*pools* を指定しない場合、システム内のすべてのプールの統計が表示されます。*count* を指定した場合、このコマンドは *count* レポートの出力後に配置されます。

-v 詳細な統計。プール全体の統計と共に、プール内の各 *vdevs* の使用状況統計をレポートします。

`zpool status` [`-xv`] [*pool*] ...
指定したプールの詳細な健全性状態を表示します。*pool* を指定しない場合、システムの各プールの状態が表示されます。

消し込みまたは再同期化が進行中の場合、このコマンドは、実行済みの割合 (パーセント) および推定完了時間をレポートします。プール内のデータ量およびシステムのその他のワークロードは変化する場合があるため、これらはどちらも概算値に過ぎません。

-x エラーが発生しているか、使用不可能なプールの状態だけを表示します。

-v 詳細なデータエラー情報を表示し、前回のプール消し込みが完了して以降のデータエラーすべての完全なリストを出力します。

`zpool offline` [`-t`] *pool device* ...
指定した物理デバイスをオフラインにします。オフライン状態にある *device* に対して、読み取りや書き込みは行われません。

このコマンドは、スペアには適用されません。

-t 「temporary」。リブートすると、指定した物理デバイスは以前の状態に戻ります。

`zpool online` *pool device* ...
指定した物理デバイスをオンラインにします。

このコマンドは、スペアには適用されません。

`zpool clear` *pool* [*device*] ...
プール内のデバイスエラーをクリアします。引数を指定しない場合、プール内のすべてのデバイスエラーがクリアされます。1 つ以上のデバイスを指定すると、指定したデバイスに関連するエラーだけがクリアされます。

`zpool attach` [`-f`] *pool device new_device*
new_device を既存の `zpool` デバイスに接続します。既存のデバイスを、raidz 構成の一部にすることはできません。*device* が現在ミラー構成の一部ではない場合、*device* は *device* および *new_device* の 2 方向のミラーに自動的に変換されます。*device* が 2 方向ミラーの一部である場合、*new_device* を接続すると 3 方向ミラーが作成され、以下同様に作成されます。どの場合でも、*new_device* の再同期化がすぐに開始されます。

-f *new_device* が使用中と表示されている場合でも、これを強制的に使用します。この方法で、すべてのデバイスを上書きできるわけではありません。

`zpool detach` *pool device*
device をミラーから切り離します。データの有効な複製がほかに存在しない場合、この操作は拒否されます。

`zpool replace` [`-f`] *pool old_device* [*new_device*]
old_device を *new_device* で置き換えます。これは、*new_device* を接続し、再同期化の実行まで待機してから *old_device* を切り離す操作と同じです。

new_device のサイズは、ミラーまたは raidz 構成内の全デバイスの最小サイズ以上でなければなりません。

new_device を指定しない場合、デフォルトの *old_device* が使用されます。この置換形式は、既存のディスクで障害が発生したためにディスクを物理的に交換したあとで実行する場合に役立ちます。この場合、実際には別のディスクであっても、新規ディスクには以前のデバイスと同じ `/dev/dsk` パスが使用されます。これは、ZFS により認識されます。

`-f` *new_device* が使用中と表示されている場合でも、これを強制的に使用します。この方法で、すべてのデバイスを上書きできるわけではありません。

`zpool scrub` [`-s`] *pool* ...
消し込みを開始します。消し込みにより、指定したプール内のすべてのデータが検査され、チェックサムが適正に検証されます。複製された (ミラーまたは raidz) デバイスの場合、消し込み中に検出されたすべての損傷は、ZFS により自動的に修復されます。「`zpool status`」コマンドは、消し込みの進捗状況をレポートします。また、完了時には消し込み結果の概要を出力します。

消し込みおよび再同期化は、非常に類似した操作です。相違点は、再同期化では ZFS が期限切れであることを認識しているデータだけが検査されますが (新規デバイスをミラーに接続する場合や、既存のデバイスを置換する場合など)、消し込みではすべてのデータが検査されるため、ハードウェア障害やディスク障害に起因する非表示のエラーも検出されます。

消し込みと再同期化は、I/O 集約的な操作であるため、ZFS では一度に 1 つの操作だけが許可されます。進行中の消し込みが存在する場合に「`zpool scrub`」コマンドを実行すると、進行中の消し込みが終了して、新規消し込みが開始されます。進行中の再同期化が存在する場合、ZFS は、再同期化が完了するまで書き込みの開始を許可しません。

`-s` 消し込みを停止します。

`zpool export` [`-f`] *pool* ...
指定したプールをシステムからエクスポートします。すべてのデバイスにエクスポート済みのマークが付けられますが、別のサブシステムからは引き続き使用中と見なされます。十分な数のデバイスが存在するかぎり、このデバイスをシステム間 (エンディアンの異なるシステムを含む) で移動またはインポートすることが可能です。

プールをエクスポートする前に、プール内のすべてのデータセットがマウント解除されます。

プールを可搬性のあるものにするため、zpool コマンドに単なるスライスではなく、ディスク全体を指定する必要があります。これにより、ZFS は可搬性のある EFI ラベルをディスクに設定します。これを行わない場合、エンディアンの異なるプラットフォーム上のディスクドライバは、ディスクを認識しません。

-f 「unmount -f」コマンドを使用して、すべてのデータセットを強制的にマウント解除します。

zpool import [-d *dir*] [-D]
インポートに使用可能なプールを一覧表示します。-d オプションを指定しない場合、このコマンドは「/dev/dsk」内のデバイスを検索します。-d オプションは、複数回指定できます。このオプションを指定すると、すべてのディレクトリが検索されます。デバイスがエクスポートされたプールの一部として表示される場合、このコマンドは、プールの名前、数値識別子、および各デバイスやファイルの *vdev* レイアウトと現在の健全性を含むプールの概要を表示します。-D オプションを指定しないかぎり、破棄されるプールや「-zpool destroy」コマンドを使って以前に破棄されたプールは表示されません。

数値識別子は一意であり、エクスポートされた同名のプールを複数利用可能な場合にプール名の代わりに使用できます。

-d *dir* デバイスまたはファイルを *dir* 内で検索します。-d オプションは複数回指定できます。

-D 破棄されたプールだけを一覧表示します。

zpool import [-d *dir*] [-D] [-f] [-o *opts*] [-R *root*] *pool* | *id* [*newpool*]
特定のプールをインポートします。プールは、名前または数値識別子を使って識別できます。*newpool* を指定すると、名前 *newpool* を使ってプールがインポートされます。それ以外の場合は、エクスポートされた名前と同じ名前を使ってインポートが行われます。

「zpool export」を最初に実行せずにシステムからデバイスを削除すると、そのデバイスは潜在的にアクティブな状態として表示されます。これがエクスポートの失敗を示すのか、デバイスが別のホストにより実際に使用されていることを示すのかを見分けることはできません。この状態のプールをインポートするには、-f オプションを指定する必要があります。

-d *dir* デバイスまたはファイルを *dir* 内で検索します。-d オプションは複数回指定できます。

-D 破棄されたプールをインポートします。-f オプションも指定する必要があります。

-f プールが潜在的にアクティブであると表示されている場合でも、インポートを強制的に実行します。

`-o` *opts*　プール内のデータセットのマウント時に使用するマウントオプションのコンマ区切りのリスト。データセットのプロパティーとマウントオプションの説明については、zfs(1M) を参照してください。

`-R` *root*　代替 *root* を使ってプールをインポートします。「代替ルートプール」の節を参照してください。

`zpool import` [`-d` *dir*] [`-D`] [`-f`] `-a`
検索ディレクトリ内で検出されたプールをすべてインポートします。十分な数のデバイスを使用可能なプールがすべてインポートされることを除き、前のコマンドと同じです。`-D` オプションを指定しないかぎり、破棄されるプールや「`-zpool destroy`」コマンドを使って以前に破棄されたプールはインポートされません。

`-d` *dir*　デバイスまたはファイルを *dir* 内で検索します。`-d` オプションは複数回指定できます。

`-D`　破棄されたプールだけをインポートします。`-f` オプションも指定する必要があります。

`-f`　プールが潜在的にアクティブであると表示されている場合でも、インポートを強制的に実行します。

`zpool upgrade`
ディスク上にある ZFS とは別のバージョンを使ってフォーマットされたプールをすべて表示します。以前のバージョンは引き続き使用できますが、一部の機能は利用できない場合があります。これらのプールは、「`zpool upgrade -a`」を使用してアップグレードできます。より新しいバージョンでフォーマットされたプールも表示されますが、システム上でこれらのプールにアクセスすることはできません。

`zpool upgrade -v`
現在のソフトウェアでサポートされている ZFS バージョンを表示します。現在の ZFS バージョンおよびサポートされる以前のバージョンすべてが、各バージョンで提供される機能に関する説明と共に表示されます。

`zpool upgrade` [`-a` | *pool*]
指定したプールを最新のオンディスクバージョンにアップグレードします。これを実行すると、以前のバージョンのソフトウェアを実行するシステム上で、プールにアクセスすることはできなくなります。

`-a`　すべてのプールをアップグレードします。

`zpool history` [*pool*] ...
指定したプール (プールを指定しない場合はすべてのプール) のコマンド履歴を表示します。

使用例

例 1 RAID-Z ストレージプールを作成する

次のコマンドは、6 台のディスクで構成される単一の raidz ルート *vdev* を含むプールを作成します。

```
# zpool create tank raidz c0t0d0 c0t1d0 c0t2d0 c0t3d0 c0t4d0 c0t5d0
```

例 2 ミラー化されたストレージプールを作成する

次のコマンドは、2 つのミラーを持つプールを作成します。各ミラーには 2 台のディスクが含まれます。

```
# zpool create tank mirror c0t0d0 c0t1d0 mirror c0t2d0 c0t3d0
```

例 3 スライスを使って ZFS ストレージプールを作成する

次のコマンドは、2 つのディスクスライスを使ってミラー化されていないプールを作成します。

```
# zpool create tank /dev/dsk/c0t0d0s1 c0t1d0s4
```

例 4 ファイルを使って ZFS ストレージプールを作成する

次のコマンドは、ファイルを使ってミラー化されていないプールを作成します。推奨される使用法ではありませんが、実験目的で使用する場合にファイルを使って作成したプールは便利です。

```
# zpool create tank /path/to/file/a /path/to/file/b
```

例 5 ミラーを ZFS ストレージプールに追加する

次のコマンドは、2 つのミラー化ディスクをプール「*tank*」に追加します。ただし、プールは 2 方向ミラーで構成済みであるものとします。プール内のすべてのデータセットは、追加された領域をすぐに利用できます。

```
# zpool add tank mirror c1t0d0 c1t1d0
```

例 6 使用可能な ZFS ストレージプールを一覧表示する

次のコマンドは、システムで使用可能なすべてのプールを一覧表示します。ここでは、デバイスの欠落が原因で、プール *zion* に障害が発生しています。

このコマンドを実行すると、次のような結果が表示されます。

例 6 使用可能なZFSストレージプールを一覧表示する (続き)

```
# zpool list
  NAME              SIZE    USED   AVAIL    CAP  HEALTH     ALTROOT
     pool            67.5G   2.92M   67.5G     0%  ONLINE     -
     tank            67.5G   2.92M   67.5G     0%  ONLINE     -
     zion                -       -       -     0%  FAULTED    -
```

例 7 ZFSストレージプールを破棄する

次のコマンドは、プール「*tank*」およびその内部のデータセットをすべて破棄します。

```
    # zpool destroy -f tank
```

例 8 ZFSストレージプールをエクスポートする

次のコマンドは、プール *tank* 内のデバイスをエクスポートします。エクスポートされたデバイスは、再配置したり、あとでインポートしたりすることが可能です。

```
# zpool export tank
```

例 9 ZFSストレージプールをインポートする

次のコマンドは、使用可能なプールを表示してから、プール「tank」をインポートしてシステムで使用できるようにします。

このコマンドを実行すると、次のような結果が表示されます。

```
# zpool import
  pool: tank
    id: 15451357997522795478
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        tank        ONLINE
          mirror    ONLINE
            c1t2d0  ONLINE
            c1t3d0  ONLINE
# zpool import tank
```

例 10 すべての ZFS ストレージプールを最新のバージョンにアップグレードする

次のコマンドは、すべての ZFS ストレージプールを最新バージョンのソフトウェアにアップグレードします。

```
# zpool upgrade -a
This system is currently running ZFS version 2.
```

例 11 ホットスペアを管理する

次のコマンドは、使用可能なホットスペアを 1 つ含む新しいプールを作成します。

```
# zpool create tank mirror c0t0d0 c0t1d0 spare c0t2d0
```

いずれかのディスクで障害が発生すると、このプールは機能低下状態に陥ります。障害が発生したデバイスを交換するには、次のコマンドを使用します。

```
# zpool replace tank c0t0d0 c0t3d0
```

いったんデータが回復されると、スペアは自動的に削除され、障害が発生した別のデバイスで使用可能となります。ホットスペアをプールから完全に削除するには、次のコマンドを使用します。

```
# zpool remove tank c0t2d0
```

終了ステータス

次の終了ステータスが返されます。

`0` 正常終了。

`1` エラーが発生した。

`2` 無効なコマンド行オプションが指定された。

属性

次の属性についての詳細は、マニュアルページの `attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	SUNWzfsu
インタフェースの安定性	開発中

関連項目

zfs(1M), `attributes`(5)