# Red Hat Enterprise Linux 7
# ストレージ管理ガイド

---

Red Hat Enterprise Linux 7 における単一ノードストレージの導入と設定
エディッション 2


Red Hat Subject Matter ExpertsJosef Bacik
Kamil Dudka
Hans de Goede
Harald Hoyer
Doug Ledford
Daniel Novotny
Nathan Straz
David Wysochanski
Contributors
Michael Christie
Sachin Prabhu
Rob Evers
David Howells
David Lehman
Jeff Moyer
Eric Sandeen
Mike Snitzer

# Red Hat Enterprise Linux 7 ストレージ管理ガイド

## Red Hat Enterprise Linux 7 における単一ノードストレージの導入と設定 エディッション 2

Josef Bacik
Server Development Kernel File System
jwhiter@redhat.com
ディスククォータ

Kamil Dudka
Base Operating System Core Services - BRNO
kdudka@redhat.com
アクセス制御リスト

Hans de Goede
Base Operating System Installer
hdegoede@redhat.com
パーティション

Harald Hoyer
Engineering Software Engineering
harald@redhat.com
ファイルシステム

Doug Ledford
Server Development Hardware Enablement
dledford@redhat.com
RAID

Daniel Novotny
Base Operating System Core Services - BRNO
dnovotny@redhat.com
/proc ファイルシステム

Nathan Straz
Quality Engineering QE - Platform
nstraz@redhat.com
GFS2

David Wysochanski
Server Development Kernel Storage
dwysocha@redhat.com
LVM/LVM2

Michael Christie
Server Development Kernel Storage
mchristi@redhat.com
オンラインストレージ

Sachin Prabhu
Software Maintenance Engineering
sprabhu@redhat.com
NFS

Rob Evers
Server Development Kernel Storage
revers@redhat.com
オンラインストレージ

David Howells
Server Development Hardware Enablement
dhowells@redhat.com
FS-Cache

David Lehman
Base Operating System Installer
dlehman@redhat.com
インストール時のストレージ設定

Jeff Moyer
Server Development Kernel File System
jmoyer@redhat.com
ソリッドステートディスク

Eric Sandeen
Server Development Kernel File System
esandeen@redhat.com
ext3、ext4、XFS、暗号化ファイルシステム

Mike Snitzer
Server Development Kernel Storage
msnitzer@redhat.com
I/O スタックと制限

Red Hat Subject Matter Experts

Contributors

## 編集者

Jacquelynn East
Engineering Content Services
jeast@redhat.com

Don Domingo
Engineering Content Services
ddomingo@redhat.com

## 法律上の通知



## 概要

本ガイドは Red Hat Enterprise Linux 7 でストレージデバイスおよびファイルシステムを効果的に管理する方法を説明しています。本ガイドは、Red Hat Enterprise Linux または Fedora についての基礎から中等レベルの知識をお持ちのシステム管理者を対象にしています。

# 目次

# 前書き

# 第1章 概要

『ストレージ管理ガイド』 では Red Hat Enterprise Linux 7 で対応しているファイルシステムやデータストレージの機能などに関して詳細に解説しています。本ガイドは管理者が単一のノード (クラスター化していない) によるストレージソリューションを管理する場合に便利な参照となるよう構成されています。

ストレージ管理ガイドは、ファイルシステムとストレージ管理の 2 つに分割されています。

ファイルシステムの箇所では、Red Hat Enterprise Linux 7 がサポートする様々なファイルシステムについて説明します。まず、ファイルシステムの最適な使用方法について説明していきます。

ストレージ管理の箇所では、Red Hat Enterprise Linux 7 がサポートする様々なツールやストレージ管理タスクについて説明します。サポートされるツールやストレージ管理タスクについて説明し、これらの最適な使用方法について説明します。

## 1.1. Red Hat Enterprise Linux 7 の最新情報

Red Hat Enterprise Linux 7 は、以下のファイルシステムの拡張機能を特長としています。

### eCryptfs は含まれない

Red Hat Enterprise Linux 7 には eCryptfs は含まれません。ファイルシステムを暗号化する方法については、Red Hat のセキュリティガイドを参照してください。

### System Storage Manager

Red Hat Enterprise Linux 7 には、System Storage Manager という新規アプリケーションが含まれます。これは、複数のストレージテクノロジーを管理するコマンドラインインターフェースを提供します。さらに詳しくは、16章*System Storage Manager (SSM)*を参照してください。

### デフォルトファイルシステムとしての XFS

Red Hat Enterprise Linux 7 では、XFS はデフォルトのファイルシステムです。XFS ファイルシステムについてさらに詳しくは、6章*XFS ファイルシステム*を参照してください。

### ファイルシステムの再編成

Red Hat Enterprise Linux 7 は、新たなファイルシステム構造を導入しています。ディレクトリーの **`/bin`**、**`/sbin`**、**`/lib`**、および **`/lib64`** は **`/usr`** の下にネスト化されます。

### Snapper

Red Hat Enterprise Linux 7 では、Snapper と呼ばれる新規ツールを導入し、LVM および Btrfs のスナップショットを簡単に作成し、管理できるようになりました。さらに詳しくは、14章*Snapper*を参照してください。

### Btrfs (テクノロジープレビュー)

Btrfs は、統合された LVM 操作を含む、より良いパフォーマンスと拡張性を提供することを目的としたローカルのファイルシステムです。このファイルシステムは、Red Hat では完全にサポートされていないため、テクノロジープレビューになっています。Btrfs についてさらに詳しくは、3章*Btrfs (テクノロジープレビュー)* を参照してください。

## NFSv2 はサポート対象外

Red Hat Enterprise Linux 7 では、NFSv2 はサポート対象外になりました。

# パート I. ファイルシステム

ファイルシステムのセクションでは、ファイルシステムの構造について説明した後に、eCryptfs および Btrfs の 2 つのテクノロジープレビューを紹介します。次に、ext3、ext4、Global File System 2、XFS、NFS、および FS-Cache などを含む、Red Hat が完全にサポートするファイルシステムについて説明します。

それぞれのファイルシステムのサポート対象を比較したガイドについては、こちらから入手できます。https://access.redhat.com/site/articles/rhel-limits　この情報は、現時点では Red Hat Enterprise 6 までを対象にしていることに注意してください。この情報は Red Hat Enterprise Linux 7 が利用可能になる時点で更新されます。

# 第2章 ファイルシステムの構造およびメンテナンス

## 2.1. 共通の構造を共有する理由

ファイルシステムの構造は、オペレーティングシステムにおける最も基本的な組織レベルです。オペレーティングシステムがユーザー、アプリケーション、およびセキュリティーモデルと相互作用する方法は、オペレーティングシステムがストレージデバイス上のファイルをどのように組織しているかによってほぼ決まります。共通のファイルシステム構造を提供することで、ユーザーとプログラムが確実にファイルにアクセスして書き込みを実行できるようにします。

ファイルシステムはファイルを以下の 2 つの論理カテゴリーに分割します。

- 共有可能ファイル vs 共有不可ファイル
- 可変ファイル vs 静的ファイル

*共有可能 (Shareable)* ファイルはローカルからも、リモートホストからもアクセス可能であり、*共有不可 (unsharable)* ファイルはローカルでのみ利用できます。ドキュメントなどの*可変 (Variable)* ファイルはいつでも変更できますが、バイナリーなどの *静的 (static)* ファイルはシステム管理者からのアクションがない限りは変更されません。

このようにファイルをカテゴリー別に分類すると、各ファイルの機能とそれらのファイルを含むディレクトリーに割り当てられた権限を関連付けるのに役立ちます。オペレーティングシステムとそのユーザーがどのようにファイルと相互作用するかによって、ファイルの置かれるディレクトリーの設定が決定されます。たとえば、そのディレクトリーを読み込み専用か、または読み込み/書き込み権限でマウントするかや、各ユーザーがそのファイルに対して持つアクセスレベルが決定されます。この組織の最上位レベルが重要になり、その下にあるディレクトリーへのアクセスが制限されます。最上位レベルから下位にわたってアクセスルールが準拠されないと、セキュリティー上の問題が発生する可能性があります。

## 2.2. ファイルシステム階層標準 (FHS) の概要

Red Hat Enterprise Linux は、*ファイルシステム階層標準：Filesystem Hierarchy Standard* (*FHS*) のファイルシステム構造を使用します。これは、多くのファイルタイプとディレクトリーの名前、場所、および権限を定義します。

FHS ドキュメントは、FHS 準拠のファイルシステムに対する権威のあるリファレンスですが、この標準では定義していない、または拡張可能な分野が数多く残されています。このセクションでは、この標準の概要を説明し、この標準で扱われていないファイルシステムの他の部分について説明します。

FHS 準拠によってもたらされる最も重要な要素は以下の 2 点にまとめられます。

- FHS 準拠の他のシステムとの互換性
- **/usr/** パーティションを読み込み専用でマウントできること。**/usr/** には共通の実行可能ファイルが含まれており、ユーザーがこのパーティションを変更することができないため、この点は特に重要になります。さらに、**/usr/** は読み込み専用でマウントされるため、CD-ROM ドライブから、または読み込み専用の NFS マウント経由で他のマシンからマウントすることができます。

### 2.2.1. FHS の組織

ここで説明されているディレクトリーとファイルは、FHS ドキュメントで指定されているもののごく一部です。総合的な情報については、http://www.pathname.com/fhs/ で最新の FHS ドキュメントを参照してください。

**注記**

どのディレクトリーが利用可能であるかは、所定のシステムに何がインストールされているかによって異なります。以下の一覧は、インストールされている内容の一部の例に過ぎません。

#### 2.2.1.1. ファイルシステム情報の収集

**df** コマンドは、システムのディスク領域の使用量を報告します。出力は以下のようになります。

**例2.1 df コマンドの出力**

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                       11675568   6272120   4810348  57% / /dev/sda1
                  100691      9281     86211  10% /boot
none                     322856         0    322856   0% /dev/shm
```

デフォルトで、**df** はパーティションサイズを 1 キロバイトのブロック単位で示し、使用中/利用可能のディスク領域の容量をキロバイトで示します。この情報をメガバイトやギガバイトで表示するには、コマンド **df -h** を使用します。**-h** という引数は "human-readable (人に認識可能な)" 形式という意味です。**df -h** の出力は以下のようになります。

**例2.2 df -h コマンドの出力**

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                        12G  6.0G  4.6G  57% / /dev/sda1
   99M  9.1M   85M  10% /boot
none    316M     0  316M   0% /dev/shm
```

**注記**

上記の例では、マウント済みのパーティション **/dev/shm** は、システムの仮想メモリーファイルシステムを表します。

**du** コマンドは、ディレクトリー内のファイルによって使用されている領域の概算を示すもので、各サブディレクトリーのディスク使用量を表示します。**du** の出力の最後の行はディレクトリーの合計ディスク使用量を表示します。ディレクトリーの合計ディスク使用量を人間が認識できる形式でのみ表示するには、**du -hs** を使用します。他のオプションについては、**man du** を参照してください。

システムのパーティションとディスク領域の使用状況をグラフィカル形式で表示するには、**アプリケーション** → **システムツール** → **システムモニター** を順にクリックするか、またはコマンド**gmnoe-system-monitor** を使用して、Gnome **システムモニター** を使用します。**ファイルシステム** タブを選択すると、システムのパーティションが表示されます。以下の図は、**ファイルシステム** タブを示しています。

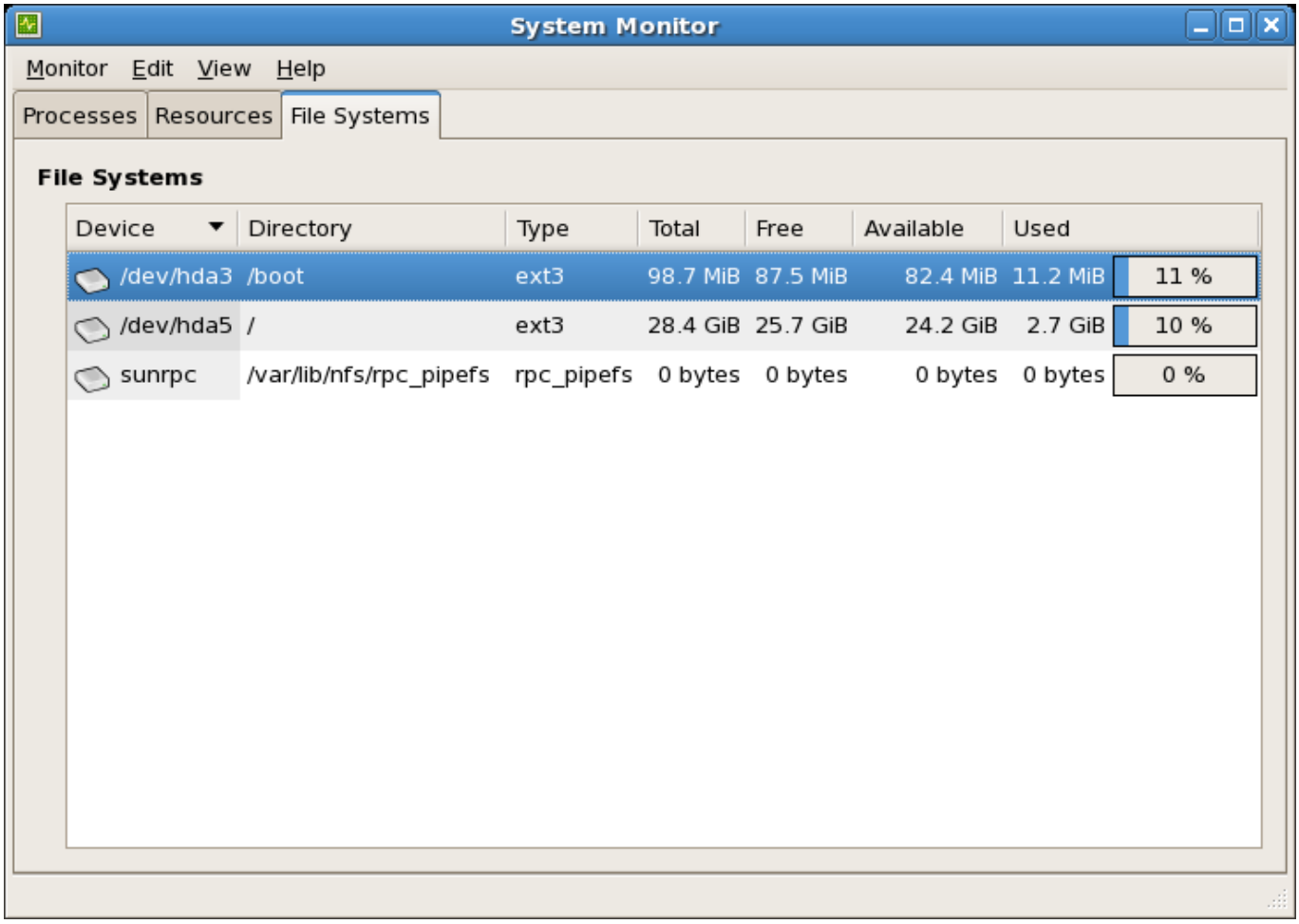



**図2.1 GNOME システムモニターの「ファイルシステム」タブ**

#### 2.2.1.2. /boot/ ディレクトリー

**/boot/** ディレクトリーには、システムを起動するために必要な Linux カーネルなどの静的ファイルが含まれます。これらのファイルはシステムが正常に起動するために不可欠です。

**警告**

**/boot/** ディレクトリーを削除しないでください。削除するとシステムを起動できなくなります。

#### 2.2.1.3. /dev/ ディレクトリー

**/dev/** ディレクトリーには、以下のデバイスタイプを表すデバイスノードが含まれます。

- システムに接続しているデバイス
- カーネルで提供される仮想デバイス

これらのデバイスノードはシステムが正常に機能するために不可欠です。**udevd** デーモンは、必要に応じて **/dev/** 内でデバイスノードを作成したり削除したりします。

**/dev/** ディレクトリーとサブディレクトリー内のデバイスは、**キャラクター** (マウスやキーボードなどの、シリアルストリームの入出力のみを提供) か、または **ブロック** (ハードドライブやフロッピーなど、ランダムにアクセス可能なデバイス) のいずれかです。GNOME または KDE をインストールしている場合、一部のストレージデバイスは接続した時点 (例: USB を使用) や挿入した時点 (例: CD や DVD ドライブを使

用) で自動的に検出されて、ポップアップウィンドウがそのコンテンツを表示します。

**表2.1 /dev ディレクトリー内にある共通ファイルの例**

| ファイル | 詳細 |
|---|---|
| /dev/hda | プライマリー IDE チャネル上のマスターデバイス |
| /dev/hdb | プライマリー IDE チャネル上のスレーブデバイス |
| /dev/tty0 | 1 番目の仮想コンソール |
| /dev/tty1 | 2 番目の仮想コンソール |
| /dev/sda | プライマリー SCSI または SATA チャネル上の 1 番目のデバイス |
| /dev/lp0 | 1 番目のパラレルポート |

有効なブロックデバイスは、以下の 2 つのタイプのエントリーのいずれかになります。

**マップされたデバイス**

ボリュームグループ内の論理ボリューム。たとえば、**/dev/mapper/VolGroup00-LogVol02**

**静的なデバイス**

従来のストレージボリューム。たとえば、**/dev/*sdbX***。ここで、*sdb* はストレージデバイス名で、*X* はパーティション番号になります。**/dev/sdb*X*** は、**/dev/disk/by-id/*WWID*** (詳細は「WWID」を参照) または **/dev/disk/by-uuid/*UUID*** (詳細は「UUID とその他の永続的となる識別子」を参照) になる場合があります。

### 2.2.1.4. /etc/ ディレクトリー

**/etc/** ディレクトリーは、マシンに対してローカルな設定ファイル用に確保されています。バイナリーはここに格納できないため、バイナリーは **/bin/** か **/sbin/** に移動する必要があります。

たとえば、**/etc/skel/** ディレクトリーは、「スケルトン」ユーザーファイルを保管しますが、このファイルは、最初にユーザーを作成する際にホームディレクトリーに追加するために使用されます。アプリケーションもこのディレクトリーにその設定ファイルを保存して実行時にそれを参照します。**/etc/exports** ファイルはリモートホストにエクスポートするファイルシステムを制御します。

### 2.2.1.5. /media/ ディレクトリー

**/media/** ディレクトリーには、USB ストレージメディア、DVD、CD-ROM、および Zip ディスクなどのリムーバブルメディア用のマウントポイントとして使用されるサブディレクトリーが含まれます。

### 2.2.1.6. /mnt/ ディレクトリー

**/mnt/** ディレクトリーは、NFS ファイルシステムマウントなど、一時的にマウントされたファイルシステムのために確保されます。すべてのリムーバブルストレージメディアには、**/media/** ディレクトリーを使用します。自動的に検出されたリムーバブルメディアは **/media** ディレクトリーにマウントされます。

**重要**

**/mnt** ディレクトリーはインストールプログラムで使用することはできません。

### 2.2.1.7. /opt/ ディレクトリー

**/opt/** ディレクトリーは、通常デフォルトインストールの一部ではないソフトウェアやアドオンパッケージ用に確保されています。**/opt/** にインストールするパッケージは、名前がそのパッケージと同じディレクトリーを作成します (例: **/opt/*パッケージ名*/**)。ほとんどの場合、そのようなパッケージは予想可能なサブディレクトリー構造に従って、ほとんどがそのバイナリーを **/opt/*パッケージ名*/bin/** に保存し、それらの **man** ページを **/opt/*パッケージ名*/man/** に保存します。

#### 2.2.1.8. /proc/ ディレクトリー

**/proc/** ディレクトリーは、カーネルから情報を抽出するか、またはカーネルに情報を送る特別なファイルを収納しています。その情報の例としては、システムメモリー、cpu 情報、およびハードウェア設定などがあります。**/proc/** の詳細については、「/proc 仮想ファイルシステム」を参照してください。

#### 2.2.1.9. /srv/ ディレクトリー

**/srv/** ディレクトリーには、Red Hat Enterprise Linux システムが提供するサイト特有のデータが含まれています。このディレクトリーは、ユーザーに対して FTP、WWW、または CVS などの特定のサービス用のデータファイルの場所を提供します。特定ユーザーにのみ関連するデータは **/home/** ディレクトリーに置く必要があります。

#### 2.2.1.10. /sys/ ディレクトリー

**/sys/** ディレクトリーは、カーネルに固有の新しい**sysfs** 仮想ファイルシステムを使用します。カーネルのホットプラグハードウェアデバイスに対するサポートの強化により、**/sys/** ディレクトリーは、**/proc/** で保管されている情報に似た情報を格納し、ホットプラグデバイス固有のデバイス情報を階層表示で示します。

#### 2.2.1.11. /usr/ ディレクトリー

**/usr/** ディレクトリーは、複数マシンで共有されるファイル用に使われます。**/usr/** ディレクトリーは多くの場合、独自のパーティション上に置かれ、読み込み専用でマウントされます。**/usr/** には少なくとも以下のサブディレクトリーが含まれます。

**/usr/bin**

このディレクトリーはバイナリー用に使用されます。

**/usr/etc**

このディレクトリーは、システム全体の設定ファイル用に使用されます。

**/usr/games**

このディレクトリーはゲームを保管します。

**/usr/include**

このディレクトリーは C ヘッダーファイル用に使用されます。

**/usr/kerberos**

このディレクトリーは、Kerberos 関連のバイナリーとファイル用に使用されます。

**/usr/lib**

このディレクトリーは、シェルスクリプトやユーザーに直接利用されるように設計されていないオブジェクトファイルとライブラリー用に使用されます。

Red Hat Enterprise Linux 7.0 の時点では、**/lib/** ディレクトリーは **/usr/lib** にマージされています。このディレクトリーには、**/usr/bin/** および **/usr/sbin/** のバイナリーを実行するために必要なライブラリーが収納されています。これらの共有ライブラリーイメージは、システムを起動したり root ファイルシステム内でコマンドを実行したりするために使用されます。

**/usr/libexec**

このディレクトリーには、他のプログラムから呼び出される小さなヘルパープログラムが収納されています。

**/usr/sbin**

このディレクトリーは、**/sbin/** に属さないシステム管理バイナリーを格納します。

Red Hat Enterprise Linux 7.0 では、**/sbin** は **/usr/sbin** にマージされました。これは、このディレクトリーにシステムの起動、復元または修復に不可欠なバイナリーを収納しています。**/usr/sbin/** のバイナリーはその使用に root 権限が必要です。さらに、**/sbin/** には、**/usr/** ディレクトリーがマウントされる**前に** システムで使用されるバイナリーが含まれます。**/usr/** がマウントされた後に使用されるすべてのシステムユーティリティーは、通常 **/usr/sbin/** に置かれます。

少なくとも、以下のプログラムが **/usr/sbin/** 内に格納される必要があります。

- **arp**
- **clock**
- **halt**
- **init**
- **fsck.***
- **grub**
- **ifconfig**
- **agetty**
- **mkfs.***
- **mkswap**
- **reboot**
- **route**
- **shutdown**
- **swapoff**
- **swapon**

**/usr/share**

このディレクトリーは、アーキテクチャーに固有ではないファイルを格納します。

**/usr/src**

このディレクトリーは、ソースコードを格納します。

**/var/tmp にリンクされた /usr/tmp**

このディレクトリーは、一時ファイルを格納します。

**/usr/** ディレクトリーには、**/local/** サブディレクトリーが含まれる必要もあります。FHS に準じて、このサブディレクトリーはソフトウェアをローカルでインストールする際にシステム管理者によって使用されます。さらに、このサブディレクリーはシステムの更新時に上書きされないように保護されます。**/usr/local** ディレクトリーは **/usr/** と似た構造を持ち、以下のサブディレクトリーが含まれます。

- **/usr/local/bin**
- **/usr/local/etc**
- **/usr/local/games**
- **/usr/local/include**
- **/usr/local/lib**
- **/usr/local/libexec**
- **/usr/local/sbin**
- **/usr/local/share**
- **/usr/local/src**

Red Hat Enterprise Linux による **/usr/local/** の使用は FHS とは少々異なります。FHS では、システムソフトウェアのアップグレード時に保護するソフトウェアを格納するために **/usr/local/** を使用する必要があるとしています。しかし、**RPM パッケージマネージャー** はソフトウェアアップグレードを安全に実行できるため、ファイルを **/usr/local/** に保存することによって保護する必要はありません。

代わりに、Red Hat Enterprise Linux は **/usr/local/** をマシンに対してローカルのソフトウェア用に使用します。たとえば **/usr/** ディレクトリーが読み込み専用の NFS 共有としてリモートホストからマウントされている場合も、パッケージまたはプログラムを **/usr/local/** ディレクトリー下にインストールすることができます。

### 2.2.1.12. /var/ ディレクトリー

FHS は Linux が **/usr/** を読み込み専用としてマウントすることを要求するため、ログファイルを書き込むプログラムや、**spool/** ディレクトリーまたは **lock/** ディレクトリーを必要とするすべてのプログラムはそれらを **/var/** ディレクトリーに書き込む必要があります。FHS では、**/var/** がスプールディレクトリーおよびファイル、ログデータ、一過性/一時的ファイルを含む可変データ用であるとしています。

**/var/** ディレクトリー内にある一部のディレクトリーを以下に示します。

- **/var/account/**
- **/var/arpwatch/**
- **/var/cache/**
- **/var/crash/**
- **/var/db/**
- **/var/empty/**
- **/var/ftp/**
- **/var/gdm/**

- **/var/kerberos/**
- **/var/lib/**
- **/var/local/**
- **/var/lock/**
- **/var/log/**
- **/var/spool/mail/** にリンクされた **/var/mail**
- **/var/mailman/**
- **/var/named/**
- **/var/nis/**
- **/var/opt/**
- **/var/preserve/**
- **/var/run/**
- **/var/spool/**
- **/var/tmp/**
- **/var/tux/**
- **/var/www/**
- **/var/yp/**

**messages** や **lastlog** などのシステムログファイルは、**/var/log/** ディレクトリーに置かれます。**/var/lib/rpm/** ディレクトリーは、RPM システムデータベースを収納します。ロックファイルは **/var/lock/** ディレクトリーに置かれますが、通常はそのファイルを使用するプログラム用のディレクトリー内にあります。**/var/spool/** ディレクトリーには、一部のプログラムのデータファイルを保存するサブディレクトリーがあります。これらのサブディレクトリーには以下が含まれます。

- **/var/spool/at/**
- **/var/spool/clientmqueue/**
- **/var/spool/cron/**
- **/var/spool/cups/**
- **/var/spool/exim/**
- **/var/spool/lpd/**
- **/var/spool/mail/**
- **/var/spool/mailman/**
- **/var/spool/mqueue/**
- **/var/spool/news/**
- **/var/spool/postfix/**

- **/var/spool/repackage/**
- **/var/spool/rwho/**
- **/var/spool/samba/**
- **/var/spool/squid/**
- **/var/spool/squirrelmail/**
- **/var/spool/up2date/**
- **/var/spool/uucp/**
- **/var/spool/uucppublic/**
- **/var/spool/vbox/**

## 2.3. 特殊な Red Hat Enterprise Linux ファイルの場所

Red Hat Enterprise Linux は特殊なファイルに対応するために、FHS の構造を少々拡張しています。

RPM に関連するほとんどのファイルは、**/var/lib/rpm/** ディレクトリー内に保管されます。RPM についてさらに詳しくは、**man rpm** を参照してください。

**/var/cache/yum/** ディレクトリーには、システムの RPM ヘッダー情報を含む、**パッケージアップデーター** が使用するファイルを収納しています。この場所は、システムの更新中にダウンロードされる RPM を一時的に保存するためにも使用されます。Red Hat Network の詳細情報については、オンラインの https://rhn.redhat.com/ にあるドキュメントを参照してください。

Red Hat Enterprise Linux に固有のもう 1 つの場所は、**/etc/sysconfig/** ディレクトリーです。このディレクトリーは、様々な設定情報を保存します。起動時に実行される多くのスクリプトはこのディレクトリーにあるファイルを使用します。

## 2.4. /proc 仮想ファイルシステム

ほとんどのファイルシステムとは異なり、**/proc** にはテキストファイルもバイナリーファイルも含まれていません。代わりに *仮想ファイル* を保管しています。そのため、通常 **/proc** は仮想ファイルシステムと呼ばれています。これらの仮想ファイルは、通常、大量の情報が含まれていてもサイズはゼロバイトです。

**/proc** ファイルシステムはそれ自体がストレージに使用されることはありません。その主要な目的は、ハードウェア、メモリー、実行中のプロセス、および他のシステムコンポーネントに対してファイルベースのインターフェースを提供することです。そのため、対応する **/proc** ファイルを確認することにより、数多くのシステムコンポーネントについてのリアルタイム情報を取得できます。**/proc** 内の一部のファイルは、カーネルを設定するように操作することもできます (ユーザーおよびアプリケーションの両方によって可能)。

以下の **/proc** ファイルはシステムストレージの管理と監視に関連しています。

**/proc/devices**

現在設定してある様々なキャラクターデバイスとブロックデバイスを表示します。

**/proc/filesystems**

現在カーネルによってサポートされているすべてのファイルシステムのタイプを一覧表示します。

**/proc/mdstat**

システム上の複数ディスクの設定または RAID の設定の現在の情報が含まれます (ある場合)。

**/proc/mounts**

現在システムで使用されているすべてのマウントを一覧表示します。

**/proc/partitions**

パーティションブロックの割り当て情報が含まれます。

**/proc** ファイルシステムについてさらに詳しくは、Red Hat Enterprise Linux 7 『導入ガイド』 を参照してください。

## 2.5. 未使用ブロックの破棄

バッチ破棄やオンライン破棄のオペレーションは、マウントされたファイルシステムの機能であり、ファイルシステムが使用していないブロックを破棄します。ソリッドステートドライブやシンプロビジョニングされたストレージの両方に役立ちます。

**バッチ破棄オペレーション** は、**fstrim** コマンドで明示的に実行されます。このコマンドは、ユーザーの条件と一致するファイルシステム内の未使用ブロックをすべて破棄します。両方のオペレーションタイプは、ファイルシステムの基盤となっているブロックデバイスが物理的な破棄オペレーションに対応している限り、Red Hat Enterprise Linux 6.2 以降の ext4 ファイルシステムでの使用が可能です。Red Hat Enterprise Linux 6.4 以降の XFS ファイルシステムについてもこれと同様です。物理的な破棄オペレーションは、**/sys/block/*device*/queue/discard_max_bytes** の値がゼロでない場合にサポートされます。

**オンライン破棄オペレーション**は、マウント時に **-o discard**オプション (**/etc/fstab** 内か、または **mount** コマンドのいずれかで指定) を使って指定され、ユーザーの介入なしでリアルタイムで実行されます。オンライン破棄オペレーションは、「使用済み」から「空き」に移行するブロックのみを破棄します。オンライン破棄オペレーションは、Red Hat Enterprise Linux 6.2 以降の ext4 ファイルシステムや、Red Hat Enterprise Linux 6.4 以降の XFS ファイルシステムでサポートされます。

システムのワークロードがバッチ破棄で対応できない場合やパフォーマンスを維持するためにオンライン破棄オペレーションが必要な場合以外は、バッチ破棄オペレーションを推奨します。

# 第3章 Btrfs (テクノロジープレビュー)

Btrfs は次世代 Linux ファイルシステムで、高度な管理、信頼性、および拡張性機能を提供します。Btrfs は、スナップショット、圧縮機能および統合デバイス管理を提供する点でユニークと言えます。

**重要**

Btrfs は Red Hat Enterprise Linux 7 のテクノロジープレビューです。

## 3.1. Btrfs ファイルシステムを作成する

基本的な Btrfs ファイルシステムを作成するには、以下のコマンドを使用します。

```
# mkfs.btrfs /dev/device
```

追加デバイスを備えた Btrfs ファイルシステムを作成し、メタデータおよびデータ用のマルチデバイスプロファイルを指定する方法についてさらに詳しくは、「複数デバイスの統合ボリューム管理」を参照してください。

## 3.2. Btrfs ファイルシステムをマウントする

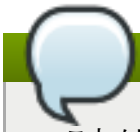
Btrfs ファイルシステムでデバイスをマウントするには、以下のコマンドを使用します。

```
# mount /dev/device /mount-point
```

その他の役に立つオプション:

**device=*/dev/name***

このオプションをマウントコマンドに追加すると、btrfs は Btrfs ボリューム用の指定されたデバイスをスキャンするように指示されます。このオプションは、Btrfs ではないデバイスのマウントを試みるとマウントが失敗するため、確実にマウントを成功させるために使用されます。



**注記**

これは、すべてのデバイスがファイルシステムに追加されるということではなく、それらのデバイスのスキャンのみが行なわれます。

**max_inline=*number***

このオプションを使用して、メタデータ B ツリーリーフ内のインラインデータに使用できる最大スペース (バイト単位) を設定します。デフォルトは 8192 バイトです。4k ページの場合、リーフに適合させる必要のある追加ヘッダーがあるため、このサイズは 3900 バイトに制限されます。

**alloc_start=*number***

このオプションを使用して、ディスクストレージの割り当てにおける開始点を設定します。

**thread_pool=*number***

このオプションを使用して、割り当てられるワーカースレッドの数を指定します。

**discard**

このオプションを使用して、解放されたブロック上で discard/TRIM を有効にします。

**noacl**

このオプションを使用して ACL の使用を無効にします。

**space_cache**

このオプションを使用して、ディスク上の空き領域のデータを保存し、ブロックグループのキャッシュを高速化します。これは、永続的な変更であり、古いカーネルでブートしても問題はありません。

**nospace_cache**

このオプションを使用して、上記の **space_cache** を無効にします。

**clear_cache**

このオプションを使用して、マウント時にすべての空き領域キャッシュをクリアにします。これは、安全なオプションですが、キャッシュ領域の再ビルドをトリガーします。そのため、再ビルドプロセスを終了させるためにファイルシステムをマウントした状態にしておきます。このマウントオプションは、1 回限り、かつ空き領域に関する問題が明らかになった後にのみ使用することが意図されています。

**enospc_debug**

このオプションは、「空き領域が残っていない」問題をデバッグするために使用されます。

**recovery**

このオプションを使用して、マウント時に自動リカバリーを有効にします。

## 3.3. Btrfs ファイルシステムのサイズ変更

本来 Btrfs ファイルシステムをサイズ変更することはできませんが、ファイルシステムが使用する各デバイスのサイズ変更は可能です。使用されているデバイスが 1 つだけの場合は、ファイルシステムのサイズ変更と同様のことが行なわれます。使用されているデバイスが複数ある場合、それらは必要な結果を得るために手動でサイズ変更する必要があります。

以下の 3 つのオプションの違いは、+ または - のいずれが先行するかという点です。+ がファイルシステムの変更数の前に置かれる場合、その数は増加することを意味し、- がファイルシステムの変更数の前に置かれる場合、その数は減少することを意味します。いずれも付けられない場合、ファイルシステムが指定される値に設定されます。

**注記**

単位サイズについて大文字と小文字を区別しない場合、GiB には **G** と **g** の両方を使用できます。

このコマンドは、テラバイトについて **t** を使用することはできず、ペタバイトについて**p** を使用することはできません。**k**、**m**、および **g** のみを使用できます。

### Btrfs ファイルシステムの拡大

ファイルシステムを拡大するには、このコマンドを使用します。

```
# btrfs filesystem resize amount /mount-point
```

*amount* には、2 GiB の増分について "+2g" などの設定されたサイズを指定したり、ブロックデバイス全体を埋めるまでにファイルシステムを拡大する場合は "max" と指定したりできます。

### btrfs ファイルシステムの縮小

以下のコマンドを使用して、btrfs ファイルシステムを縮小します。

```
# btrfs filesystem resize amount /mount-point
```

2 GiB の減少分として "-2g" とするなど、縮小分として必要なサイズに *amount* を設定します。

### ファイルシステムのサイズ設定

以下のコマンドを使用して、btrfs ファイルシステムを設定された値に設定します。

```
#btrfs filesystem resize amount /mount-point
```

ファイルシステムのサイズを 20 GiB にするために "20g" とするなど、ファイルシステムの必要なサイズに *amount* を設定します。

## 3.4. 複数デバイスの統合ボリューム管理

BTRFS ファイルシステムは、数多くのデバイスの上部に作成することができ、ファイルシステムの作成後に複数のデバイスを追加することができます。デフォルトでは、メタデータが 2 つのデバイス間でミラー化され、データは表示されるすべてのデバイス間でストライプ化されます。ただし、1 つのデバイスのみが表示される場合、メタデータはそのデバイス上に複製されます。

### 3.4.1. 複数デバイスによるファイルシステムの作成

**mkfs.btrfs** コマンドは、「Btrfs ファイルシステムを作成する」で詳細に説明されているように、データとしてオプション **-d** を受け入れ、メタデータとして **-m** を受け入れます。有効な仕様は以下の通りです。

- **raid0**
- **raid1**
- **raid10**
- **single**

**-m single** オプションは、メタデータの複製が行なわれていないことを通知します。この機能は、ハードウェア RAID を使用する場合に必要になることがあります。

**注記**

Raid10 が正常に実行されるには、少なくとも 4 つのデバイスが必要です。

**例3.1 raid10 btrfs ファイルシステムの作成**

4 つのデバイス間でファイルシステムを作成します (ミラー化されたメタデータ、ストライプ化されたデータ)。

```
# mkfs.btrfs /dev/device1 /dev/device2 /dev/device3 /dev/device4
```

ミラー化せずにメタデータをストライプ化します。

```
# mkfs.btrfs -m raid0 /dev/device1 /dev/device2
```

データとメタデータの両方に raid10 を使用します。

```
# mkfs.btrfs -m raid10 -d raid10 /dev/device1 /dev/device2 /dev/device3
/dev/device4
```

単一デバイス上でメタデータを複製しません。

```
# mkfs.btrfs -m single /dev/device
```

ドライブのサイズが異なる場合、各デバイスの容量を完全に使用するには **single** オプションを使用します。

```
# mkfs.btrfs /dev/device1 /dev/device2 /dev/device3
# mount /dev/device3 /mount-point
```

新規デバイスをすでに作成されたマルチデバイスファイルシステムに追加するには、以下のコマンドを使用します。

```
# mkfs.btrfs /dev/device1 /dev/device2 /dev/device3
# mount /dev/device3 /mount-point
```

Btrfs モジュールの再起動または再読み込みの後に、**btrfs device scan** コマンドを使用して、すべてのマルチデバイスファイルシステムを検出します。さらに詳しくは、「デバイスをスキャンする Btrfs マルチデバイス」 を参照してください。

### 3.4.2. デバイスをスキャンする Btrfs マルチデバイス

**btrfs device scan** を使用して、**/dev** 以下のすべてのブロックデバイスをスキャンし、BTRFS ボリュームをプローブします。ファイルシステム内で複数のデバイスと共に実行されている場合、Btrfs モジュールの読み込み後に実行されます。

すべてのデバイスをスキャンするには、以下のコマンドを使用します。

```
# btrfs device scan
```

単一デバイスをスキャンするには、以下のコマンドを使用します。

```
# btrfs device scan /dev/device
```

### 3.4.3. 新規デバイスの Btrfs ファイルシステムへの追加

**btrfs filesystem show** コマンドを使用して、すべての btrfs ファイルシステムとそれらに含まれるデバイスをリストします。

**btrfs device add** コマンドは、マウントされたファイルシステムに新規のデバイスを追加するために使用されます。

**btrfs filesystem balance** コマンドは、すべての既存デバイス間で割り当てられたエクステントのバランスを取ります (再ストライピング)。

新規デバイスを追加するためのこれらすべてのコマンドの例は以下の通りです。

**例3.2 新規デバイスの btrfs ファイルシステムへの追加**

最初に、btrfs ファイルシステムの作成とマウントを行います。btrfs ファイルシステムを作成する方法についての詳細は「Btrfs ファイルシステムを作成する」、btrfs ファイルシステムのマウント方法についての詳細は、「Btrfs ファイルシステムをマウントする」を参照してください。

```
# mkfs.btrfs /dev/device1
# mount /dev/device1
```

次に、マウントされた btrfs ファイルシステムに 2 番目のデバイスを追加します。

```
# btrfs device add /dev/device2 /mount-point
```

これらのデバイス上のメタデータとデータは、依然として **/dev/device1** にのみ保存されます。これがすべてのデバイスに広がるようにバランスを取る必要があります。

```
# btrfs filesystem balance /mount-point
```

ファイルシステムのバランスの調整には時間がかかります。ファイルシステムのすべてのデータとメタデータを読み取り、それを新規デバイス全体に再度書き込むためです。

### 3.4.4. btrfs ファイルシステムの変換

非 RAID ファイルシステムを RAID に変換するには、デバイスを追加し、チャンク割り当てプロファイルを変更するバランスフィルターを実行します。

**例3.3 btrfs ファイルシステムの変換**

この場合のように、既存の単一デバイスシステムの **/dev/sdb1** を raid1 システムの 2 つのデバイスに変換し、単一ディスクの障害から保護するには、以下のコマンドを使用します。

```
# mount /dev/sdb1 /mnt
# btrfs device add /dev/sdc1 /mnt
# btrfs balance start -dconvert=raid1 -mconvert=raid1 /mnt
```

**重要**

メタデータが単一デバイスのデフォルトから変換されない場合、それは DUP のままになります。これは、ブロックのコピーが別々のデバイスに置かれることを保証するものではありません。データが変換されない場合、重複コピーは全く存在しません。

### 3.4.5. btrfs デバイスの削除

**btrfs device delete** コマンドを使用して、オンラインデバイスを削除します。また、安全に削除するために使用中のすべてのエクステントをファイルシステムの他のデバイスに再配信します。

**例3.4 btrfs ファイルシステム上でのデバイスの削除**

最初に、いくつかの btrfs ファイルシステムを作成し、マウントします。

```
# mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd /dev/sde
# mount /dev/sdb /mnt
```

一部のデータをファイルシステムに追加します。

最後に、必要なデバイスを削除します。

```
# btrfs device delete /dev/sdc /mnt
```

### 3.4.6. BTRFS ファイルシステムでの障害のあるデバイスの置き換え

「btrfs デバイスの削除」は、障害のあるデバイスを削除するために使用できます。ただし、これはスーパーブロックを依然として読み込み可能な場合に限られます。デバイスが見つからなかったり、スーパーブロックが壊れていたりすると、ファイルシステムは低下モードでマウントする必要があります。

```
# mkfs.btrfs -m raid1 /dev/sdb /dev/sdc /dev/sdd /dev/sde


  ssd is destroyed or removed, use -o degraded to force the mount
  to ignore missing devices


# mount -o degraded /dev/sdb /mnt


  'missing' is a special device name


# btrfs device delete missing /mnt
```

コマンド **btrfs device delete missing** は、ファイルシステムのメタデータで記述されているものの、ファイルシステムのマウント時には表示されない最初のデバイスを削除します。

**重要**

見つからないものがあることを考慮しても、特定の RAID レイアウトに必要なデバイスの最小数を下回ることは考えられません。障害のあるものを削除するために、新たなデバイスを追加することが必要になる場合があります。

たとえば、2 つのデバイスからなる raid1 レイアウトについては、デバイスが失敗した場合に以下を実行する必要があります。

1. 低下モードでのマウント
2. 新規デバイスの追加
3. 見つからないデバイスの削除

### 3.4.7. btrfs ファイルシステムの /etc/fstab への登録

initrd がないか、または btrfs デバイススキャンを実行しない場合、ファイルシステム内のすべてのデバイスを mount コマンドに明示的に渡すことで、マルチボリューム BTRFS ファイルシステムをマウントすることができます。

**例3.5 /etc/fstab エントリーの例**

適切な **/etc/fstab** エントリーの例は、以下のようになります。

```
/dev/sdb    /mnt    btrfs
device=/dev/sdb,device=/dev/sdc,device=/device/sdd,device=/dev/sde    0
```

UUID を使用することもでき、この方法はデバイスパスを使用するよりも安定性が高くなります。

## 3.5. SSD の最適化

btrfs ファイルシステムを使用して SSD を最適化することができます。これは 2 つの方法で実行することができます。

最初の方法として、**/sys/block/*device*/queue/rotational** が単一の指定されたデバイスに対してゼロである場合に **mkfs.btrfs** は単一デバイスでのメタデータの複製をオフにします。これは、コマンドラインで **-m single** を指定するのと同等の動作になります。**-m dup** オプションを指定すると、この設定はオーバーライドされ、メタデータの複製が強制実行されます。ただし、SSD ファームウェアが両方のコピーを失う可能性があるため、複製は不要です。スペースの無駄やパフォーマンスコストを考慮してもそのように言えます。

2 つ目として、SSD マウントオプションの **ssd**、 **nossd**、および **ssd_spread** などのグループを使用する方法があります。

**ssd** オプションは、いくつかのことを実行します。

- 大規模なメタデータクラスター割り当てを可能にします。
- 可能な場合は、データをより順次に割り当てます。
- キーとブロックの順番を一致させるため btree リーフの再書き込みを無効にします。
- 複数プロセスをバッチ処理を行なわずに、ログのフラグメントをコミットします。

**注記**

**ssd** マウントオプションは ssd オプションのみを有効にします。これを無効にするには、**nossd** オプションを使用します。

一部の SSD のパフォーマンスは、ブロック番号を再利用する場合に最もよくなることが多くあります。また、クラスター化によって未使用領域の大きなチャンクが厳密に割り当てられるとパフォーマンスが大幅に向上する場合もあります。デフォルトでは、**mount -o ssd** は、割り当てられたブロックが混在している可能性のある複数の空きブロックがある場合に、ブロックの複数のグループを検索します。コマンド **mount -o ssd_spread** は、割り当て済みのブロックが混在していないことを確認します。これにより、ローエンドの SSD のパフォーマンスが強化されます。

**注記**

**ssd_spread** オプションは、**ssd** と **ssd_spread** オプションの両方を有効にします。これらのどちらのオプションも無効にするには、**nossd** を使用します。

**ssd_spread** オプションは、いずれの ssd オプションも指定されず、デバイスのいずれかが非回転デバイスの場合は、自動的に設定されません。

これらのオプションは、これらを使用することによりパフォーマンスが改善されるか、または低下するかを調べるために、ご使用の特定のビルドですべてテストする必要があります。ビルドにより、SSD ファームウェアとアプリケーション負荷の組み合わせはそれぞれ異なるためです。

## 3.6. Btrfs の参考資料

man ページ **btrfs(8)** には、すべての重要な管理コマンドについての記載があります。とくに、以下が含まれます。

- スナップショットを管理するためのすべてのサブボリュームコマンド。
- デバイスを管理するための **device** コマンド。
- **scrub**、**balance**、および **defragment** コマンド。

man ページの **mkfs.btrfs(8)** には、関連するすべてのオプションを含め、BTRFS ファイルシステムを作成する方法についての情報が含まれます。

btrfs システム上の **fsck** に関する情報を記載した man ページの**btrfsck(8)**。

# 第4章 Ext3 ファイルシステム

基本的に、ext3 ファイルシステムは ext2 ファイルシステムの拡張されたバージョンです。各種の改善点により以下のような利点が提供されます。

**可用性**

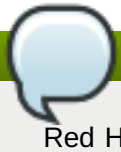
予期しない停電やシステムクラッシュ (*クリーンでないシステムシャットダウン*とも言われる) の発生後に、マシン上の各 ext2 ファイルシステムは **e2fsck** プログラムによって整合性をチェックする必要があります。これは時間を浪費するプロセスであり、大量のファイルを含む大型ボリュームでは、著しくシステムの起動時間を遅らせてしまいます。この期間中、そのボリュームにあるデータはどれも使用できなくなります。

稼働中のファイルシステムで **fsck -n** を実行することはできますが、変更を実行することはできず、部分的に書き込まれたメタデータが発生すると誤解を生じさせる結果になりかねません。

スタック内で LVM が使用されている場合は、もう 1 つの選択肢としてファイルシステムの LVM スナップショットを取り、スナップショットで **fsck** を実行できます。

最後に、ファイルシステムを読み込み専用で再マウントするオプションがあります。すべての保留中メタデータの更新 (および書き込み) は再マウントの前にディスクに強制的に入れられます。これにより、これ以前に破損していない限り、ファイルシステムの整合性を確保できます。この時点で **fsck -n** の実行が可能になります。

ext3 ファイルシステムで提供されるジャーナリングは、クリーンでないシステムシャットダウンの発生後でもこの種のファイルシステムのチェックが不要であることを意味します。ext3 の使用で整合性チェックが必要になる唯一の場面は、ハードドライブの障害が発生した場合などのごく稀なハードウェア障害のケースのみです。クリーンでないシャットダウンの発生後に ext3 ファイルシステムを復元する時間はファイルシステムのサイズやファイルの数量に左右されません。むしろ、整合性の維持のために使用される *ジャーナル* のサイズで決まります。デフォルトのジャーナルサイズの場合、ハードウェアのスピードにもよりますが復元に 1 秒ほどかかります。



**注記**

Red Hat でサポートされている ext3 の唯一のジャーナリングモードは**data=ordered** (デフォルト) です。

**データの整合性**

ext3 ファイルシステムは、クリーンでないシステムシャットダウンが発生した際にデータの整合性が失われることを防止します。ext3 ファイルシステムにより、データが受けることのできる保護のタイプとレベルを選択できるようになります。ファイルシステムの状態に関しては、ext3 のボリュームはデフォルトで高度なレベルのデータ整合性を維持するように設定されています。

**速度**

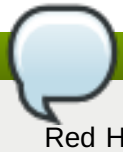
ext3 のジャーナリングはハードドライブのヘッド動作を最適化するため、一部のデータを複数回書き込んだとしても、ほとんどのケースで ext2 よりも高いスループットがあります。速度を最適化するために 3 つのジャーナリングモードから選択できますが、システムに障害が発生する可能性のある状況では、モードの選択とデータの整合性がトレードオフの関係になることがあります。

**注記**

Red Hat でサポートされている ext3 の唯一のジャーナリングモードは**data=ordered** (デフォルト) です。

**容易な移行**

ext2 から ext3 へ移行して、再フォーマットをせずに堅固なジャーナリングファイルシステムの利点を活かす操作は簡単です。このタスクの実行方法の詳細については、「Ext3 ファイルシステムへの変換」 を参照してください。



**注記**

Red Hat Enterprise Linux 7 は、統一された extN ドライバーを提供します。これは、ext2 および ext3 設定を無効にすることによって実行され、これらのオンディスク形式に対して **ext4.ko** を使用します。これは、使用される ext ファイルシステムの種類を問わず、カーネルのメッセージは常に ext4 を参照することを意味します。

以下のセクションでは、ext3 パーティションの作成とチューニングについて説明します。ext2 パーティションについては、以下にあるパーティション設定とフォーマットのセクションを飛ばして、直接「Ext3 ファイルシステムへの変換」 に進んでください。

## 4.1. Ext3 ファイルシステムの作成

インストール後も、新規の ext3 ファイルシステムを作成する必要がある場合があります。たとえば、システムに新しいディスクドライブを追加した場合、そのドライブにパーティション設定して ext3 ファイルシステムを使用します。

ext3 ファイルシステムを作成する手順は以下のようになります。

**手順4.1 ext3 ファイルシステムの作成**

1. **mkfs** を使用して、ext3 ファイルシステムのパーティションまたは LVM ボリュームをフォーマットします。
2. **e2label** を使用して、ファイルシステムにラベルを付けます。

さらに、特定の UUID をファイルシステムに追加することもできます。たとえば、UUID の 7cd65de3-e0be-41d9-b66d-96d749c02da7 をファイルシステム **/dev/sda8** に追加するには、以下のコマンドを使用します。

```
# mkfs -t ext3 -U 7cd65de3-e0be-41d9-b66d-96d749c02da7 /dev/sda8
# tune2fs -U 7cd65de3-e0be-41d9-b66d-96d749c02da7 /dev/sda8
```

ext3 を使用しているファイルシステムタイプ (例: ext4) に置き換え、-U オプションに選択した UUID を加え、/dev/sda8 を UUID を追加するファイルシステムに置き換えます。

## 4.2. Ext3 ファイルシステムへの変換

**tune2fs** コマンドは、**ext2** ファイルシステムを **ext3** に変換します。

**注記**

ext2 を ext3 に変換するには、常に **e2fsck** ユーティリティーを使用し、**tune2fs** の使用前後にファイルシステムをチェックしてください。ext2 の ext3 への変換を試みる前に、エラーが発生する場合に備えてすべてのファイルシステムをバックアップします。

さらに Red Hat では、ext2 から ext3 への変換を随時実行するのではなく、まず新しい ext3 ファイルシステムを作成してお持ちのデータを移行することをお勧めしています。

**ext2** ファイルシステムを **ext3** に変換するには、root としてログインしてから、ターミナルで以下のコマンドを入力します。

```
# tune2fs -j block_device
```

*block_device* には、変換される ext2 ファイルシステムが入ります。

**df** コマンドを発行して、マウントしたファイルシステムを表示します。

## 4.3. Ext2 ファイルシステムに戻す

ext2 ファイルシステムに戻すには、以下の手順を使います。

分かりやすくするために、このセクションのサンプルコマンドではブロックデバイスに以下のような値を使用します。

**/dev/mapper/VolGroup00-LogVol02**

**手順4.2 ext3 から ext2 に戻す**

1. root でログインしてから以下を入力することにより、パーティションをアンマウントします。

   ```
   # umount /dev/mapper/VolGroup00-LogVol02
   ```

2. 以下のコマンドを入力してファイルシステムタイプを ext2 に変更します。

   ```
   # tune2fs -O ^has_journal /dev/mapper/VolGroup00-LogVol02
   ```

3. 以下のコマンドを入力して、パーティションのエラーをチェックします。

   ```
   # e2fsck -y /dev/mapper/VolGroup00-LogVol02
   ```

4. 次に、以下を入力してパーティションを ext2 ファイルシステムとして再度マウントします。

   ```
   # mount -t ext2 /dev/mapper/VolGroup00-LogVol02 /mount/point
   ```

   上記のコマンドでは、*/mount/point* を実際のパーティションのマウントポイントに置き換えます。

**注記**

パーティションの root レベルに **.journal** ファイルが存在する場合は、それを削除します。

パーティションを永久的に ext2 に変更したい場合は、**/etc/fstab** ファイルの更新を忘れないでください。更新しないと、ブート後に元に戻ります。

# 第5章 Ext4 ファイルシステム

ext4 ファイルシステムは ext3 ファイルシステムを拡張性のあるエクステンションにしたものです。Red Hat Enterprise Linux 6 では最大 16 テラバイトのみに対応していましたが、ext4 は 16 テラバイトを超えるファイルおよびファイルシステムのサイズに対応します。また、サブディレクトリー数は無制限に対応することができます (ext3 ファイルシステムの場合は最大 32,000 までの対応)。ただし、リンク数が 65,000 を超えると 1 にリセットされ増加しなくなります。bigalloc 機能は現在サポートされていません。

**注記**

ext3 と同様、**fsck** を実行する場合には ext4 ボリュームもアンマウントしなければなりません。 詳細については *4章Ext3 ファイルシステム* をご覧ください。

### 主な特長

ext4 はエクステントを使用し (ext2 および ext3 で使用された従来のブロックマッピングスキームとの対比)、サイズの大きいファイルを使用する場合のパフォーマンスが向上されているため、そのメタデータのオーバーヘッドが減少します。また、ext4 では未使用のブロックグループと inode テーブルのセクションにそれぞれラベル付けが行なわれます。これにより、ファイルシステムのチェック時にこれらを省略することができます。また、ファイルシステムチェックの速度が上がり、ファイルシステムが大きくなるほど便宜性は顕著になります。

### 割り当て機能

ext4 ファイルシステムには以下のような割り当てスキームが備わっています。

- 永続的な事前割り当て
- 遅延割り当て
- マルチブロック割り当て
- ストライプ認識割り当て

遅延割り当ておよび他のパフォーマンス最適化のため、ext4 のディスクへのファイル書き込み動作は ext3 の場合とは異なります。ext4 では、プログラムがファイルシステムへの書き込みを実行しても、**fsync()** 呼び出しを発行しない限り、その書き込みがオンディスクになる保証はありません。

ext3 では、**fsync()** の呼び出しがなくてもファイルが新たに作成されるとそのほぼ直後にデフォルトでディスクに書き込みが強制されます。この動作により、書き込まれたデータがオンディスクにあることを **fsync()** 使って確認しないというプログラムのバグが表面化しませんでした。一方、ext4 ファイルシステムはディスクへの変更書き込みの前に数秒間待機することが多く、書き込みを結合して再度順序付けを行うことにより ext3 を上回るディスクパフォーマンスを実現しています。

**警告**

ext3 とは異なり、ext4 ファイルシステムではトランザクションコミット時にディスクへのデータの書き込みを強制しません。このため、バッファーされた書き込みがディスクにフラッシュされるまでに時間がかかります。他のファイルシステムと同様、永続的なストレージにデータが書き込まれたことを確認するには、**fsync()** などのデータ整合性チェックの呼び出しを使用してください。

**ext4 のその他の機能**

ext4 ファイルシステムでは次の機能にも対応しています。

- *拡張属性* (**xattr**) — システムで、ファイルごとにいくつかの追加の名前と値のペアを関連付けられるようになります。
- *クォータジャーナリング機能* — クラッシュ発生後の時間のかかるクォータ整合性チェックが不要になります。

**注記**

ext4 で対応しているジャーナリングモードは **data=ordered** のみです (デフォルト)。

- *サブセカンドのタイムスタンプ* — サブセカンドのタイムスタンプを指定します。

## 5.1. ext4 ファイルシステムを作成する

ext4 ファイルシステムを作成するには、**mkfs.ext4** コマンドを使用します。一般的にはデフォルトのオプションがほとんどの場面での最適な設定になります。

```
# mkfs.ext4 /dev/device
```

以下にこのコマンドのサンプル出力を示します。ファイルシステムの配列や機能が出力結果として表示されます。

**例5.1 mkfs.ext4 コマンドの出力**

```
~]# mkfs.ext4 /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
245280 inodes, 979456 blocks
48972 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1006632960
30 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
 32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

ストライプ化されたブロックデバイスの場合 (RAID5 アレイ)、ファイルシステムを作成する際にストライプの配列を指定することができます。適切なストライプ配列を使用することで ext4 ファイルシステムのパフォーマンスが大幅に強化されます。

LVM ボリュームや MD ボリューム上にファイルシステムを作成する場合、**mkfs.ext4** によって最適な配列が選択されます。オペレーティングシステムに配列情報をエクスポートするハードウェア RAID の中にもこうした最適な配列を選択するものがあります。

ストライプ配列を指定する場合は、**mkfs.ext4** の **-E** オプション (拡張されたファイルシステムのオプション) に次のようなサブオプションを付けて使用します。

**stride=*value***

RAID のチャンクサイズを指定します。

**stripe-width=*value***

1 RAID デバイス内のデータディスク数を指定するか、1 ストライプ内のストライプユニット数を指定します。

いずれのサブオプションの場合も ***value*** はファイルシステムのブロック単位で指定しなければなりません。たとえば、4k のブロックファイルシステムの 64k ストライプ (16 x 4096) の場合は以下のコマンドを使用します。

```
# mkfs.ext4 -E stride=16,stripe-width=64 /dev/device
```

ファイルシステムの作成方法については **man mkfs.ext4** を参照してください。

**重要**

**tune2fs** を使用すると ext3 ファイルシステム上で ext4 の機能のいくつかを有効にし、ext3 ファイルシステムのマウントに ext4 のドライバーを使うことができます。ただし、これらの操作は十分なテストを行なわれておらず Red Hat Enterprise Linux 7 ではこの使用を **サポートしていません**。このため、この方法で変換またはマウントした ext3 ファイルシステムの一貫したパフォーマンスや予測可能な動作については保証できません。

特定の UUID をファイルシステムに追加することもできます。詳細については、「Ext3 ファイルシステムの作成」 を参照してください。

## 5.2. ext4 ファイルシステムをマウントする

ext4 ファイルシステムはオプション付けずにマウントすることができます。

```
# mount /dev/device /mount/point
```

ext4 ファイルシステムは動作に影響を与えるマウントオプションにも対応しています。たとえば、**acl** パラメーターはアクセス制御リストを有効にし、**user_xattr** パラメーターはユーザーによる拡張属性を有効にします。両方のオプションを有効にするには、以下のようにそれぞれのパラメーターに **-o** を付けて使用します。

```
# mount -o acl,user_xattr /dev/device /mount/point
```

ext3 の場合と同様に、オプションの **data_err=abort** は、エラーがファイルデータに生じた場合には、ジャーナルを中止するために使用することができます。

```
# mount -o data_err=abort /dev/device /mount/point
```

**tune2fs** ユーティリティーを使用すると、管理者の方はファイルシステムのスーパーブロック内にデフォルトのマウントオプションを設定することができるようになります。 詳細については **man tune2fs** をご覧ください。

### 書き込みバリア

書き込みキャッシュが有効になっているデバイスへの電力供給が停止した場合でもファイルシステムの整合性を維持できるようにするため、ext4 ではデフォルトで書き込みバリアを使用します。 このため、書き込みキャッシュのないデバイスやバッテリー駆動の書き込みキャッシュを持つデバイスの場合には、以下のように **nobarrier** オプションを使ってバリアを無効にします。

```
# mount -o nobarrier /dev/device /mount/point
```

書き込みバリアについての詳細は、23章*書き込みバリア*を参照してください。

## 5.3. ext4 ファイルシステムのサイズを変更する

ext4 ファイルシステムのサイズを大きくする前に、基礎となるブロックデバイスが将来的にファイルシステムを保持するのに十分なサイズであることを確認してください。該当するブロックデバイスのサイズを変更する場合は、ブロックデバイスに適した方法を使ってサイズ変更を行ってください。

ext4 ファイルシステムは **resize2fs** を使用するとマウントしたままの状態でサイズを大きくすることができます。

```
# resize2fs /mount/device size
```

**resize2fs** コマンドは**アンマウントしている**ext4 ファイルシステムのサイズを小さくすることもできます。

```
# resize2fs /dev/device size
```

ext4 ファイルシステムのサイズを変更する場合、特定の単位を示すサフィックスが使用されていない限り、**resize2fs** ユーティリティーはファイルシステムのブロックサイズ単位でサイズを読み込みます。

- **s** — 512 バイトのセクター
- **K** — キロバイト
- **M** — メガバイト
- **G** — ギガバイト

**注記**

拡張時のサイズパラメーターはオプションになります (不要になる場合が多い)。**resize2fs** は、論理ボリュームやパーティションなどに使用できるコンテナーの全領域に渡って自動的に拡張を行います。

ext4 ファイルシステムのサイズ変更に関する詳細は、**man resize2fs** をご覧ください。

## 5.4. ext4 ファイルシステムのその他のユーティリティー

Red Hat Enterprise Linux 7 には ext4 ファイルシステム管理用のユーティリティーが他にもあります。

**e2fsck**

ext4 ファイルシステムの修復時に使用します。ext4 のディスク構造の更新により ext3 ファイルシステムよりも効率的なチェックと修復が行えるようになりました。

**e2label**

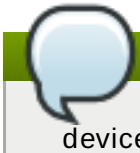
ext4 ファイルシステムのラベル変更を行います。このツールは ext2 および ext3 のファイルシステムでも動作します。

**quota**

ext4 ファイルシステム上のユーザーおよびグループごとのディスク領域 (ブロック) やファイル (inode) の使用量を制御し、それを報告します。**quota** の詳細については **man quota** および「ディスククォータの設定」をご覧ください。

**fsfreeze**

ファイルシステムへのアクセスを中断するには、フリーズを実行するためにコマンド **# fsfreeze -f *mount-point*** を使用し、フリーズ解除には **# fsfreeze -u *mount-point*** を使用します。これにより、ファイルシステムへのアクセスが停止し、安定したイメージがディスク上に作成されます。



**注記**

device-mapper ドライブに **fsfreeze** を使用する必要はありません。

詳細は、man ページの **fsfreeze(8)** を参照してください。

「ext4 ファイルシステムをマウントする」で説明している通り、**tune2fs** ユーティリティーでは ext2、ext3 および ext4 の各ファイルシステムの設定可能なファイルシステムパラメーターを調整することもできます。また、次のツールは ext4 ファイルシステムのデバッグや分析を行う際にも役に立ちます。

**debugfs**

ext2、 ext3、 ext4 の各ファイルシステムのデバッグを行います。

**e2image**

ext2、 ext3、 ext4 の重要なファイルシステムメタデータを任意のファイルに保存します。

上記のユーティリティーについての詳細は各 **man** ページをご覧ください。

# 第6章 XFS ファイルシステム

XFS は、拡張性とパフォーマンスが高いファイルシステムで、元々は Silicon Graphics, Inc. で設計されたファイルシステムでした。XFS は、Red Hat Enterprise Linux 7 のデフォルトのファイルシステムです。

### 主な特長

XFS はクラッシュからの迅速なリカバリーを容易にする *メタデータジャーナリング*に対応します。また、XFS ファイルシステムはマウント後のアクティブな状態でのデフラグや拡張も可能です。さらに Red Hat Enterprise Linux 7 では XFS 固有のバックアップや復元を行うユーティリティーにも対応しています。

### 割り当て機能

XFS には以下のような割り当てスキームが備わっています。

- エクステント (領域) ベースの割り当て
- ストライプを認識できる割り当てポリシー
- 遅延割り当て
- 領域の事前割り当て

遅延割り当てやその他のパフォーマンス最適化は、ext4 に対するのと同様に XFS ファイルシステムに影響を与えます。つまり、プログラムによる XFS ファイルシステムへの書き込みは、書き込み後にそのプログラムが **fsync()** 呼び出しを実行しない限りオンディスクになるとは限りません。

ファイルシステム (ext4 および XFS) での遅延割り当ての影響についてさらに詳しくは、5章*Ext4 ファイルシステム* の 『割り当て機能』 を参照してください。

**注記**

ディスク領域に空きがある場合でも、ファイルの作成および展開を時折実行すると、ENOSPIC で予期しない書き込みエラーが出て失敗します。これは XFS の設計がパフォーマンス中心型であるために生じます。ただし、これは残りの領域が数ブロック程度である場合にのみ生じるため、実際には問題にはなりません。

### XFS ファイルシステムのその他の機能

XFS ファイルシステムは次のような機能にも対応しています。

***拡張属性* (xattr)**

このシステムにより、ファイルごとの名前と値の複数の組み合わせを追加で関連付けられるようになります。これはデフォルトで有効にされます。

***クォータのジャーナリング***

クラッシュ後に行なわれる長い時間のかかるクォータの整合性チェックが不要になります。

***プロジェクト/ディレクトリーのクォータ***

ディレクトリーツリー全体にクォータ制限を適用することができます。

***サブセカンド (1 秒未満) のタイムスタンプ***

タイムスタンプをサブセカンド (1 秒未満) 単位にできます。

## 6.1. XFS ファイルシステムの作成

XFS ファイルシステムを作成するには、**mkfs.xfs /dev/*device*** コマンドを使用します。通常、一般的な使用にはデフォルトのオプションが最適となります。

既存のファイルシステムを含むブロックデバイス上で **mkfs.xfs** を使用する場合は、**-f** オプションを使ってそのファイルシステムの上書きを強制します。

**例6.1 mkfs.xfs コマンドの出力**

以下に **mkfs.xfs** コマンドの出力例を示します。

```
meta-data=/dev/device            isize=256    agcount=4, agsize=3277258
blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=13109032,
imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=6400, version=2
         =                       sectsz=512   sunit=0 blks, lazy-
count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

**注記**

XFS ファイルシステムの作成後にサイズを縮小することはできません。ただし、**xfs_growfs** コマンドを使ってサイズを拡大することはできます (「XFS ファイルシステムのサイズの拡大」を参照)。

ストライプ化されたブロックデバイス (RAID5 アレイなど) の場合、ファイルシステムの作成時にストライプの配列を指定することができます。適切なストライプ配列を使用することで XFS ファイルシステムのパフォーマンスが飛躍的に高まります。

ファイルシステムを LVM ボリュームや MD ボリューム上に作成すると、**mkfs.xfs** によって最適な配列が選択されます。配列情報をオペレーティングシステムにエクスポートするハードウェア RAID の場合、同様に最適な配列の選択を行うものがあります。

デバイスがストライプの配列情報をエクスポートする場合、**mkfs** (ext3、ext4、および xfs 用) がこの配列を自動的に使用します。ストライプの配列が、ストレージにストライプの配列があるにもかかわらず mkfs によって検出されない場合、以下のオプションを使用し、mkfs の実行時にストライプの配列を手動で指定することができます。

**su=*value***

ストライプユニットまたは RAID のチャンクサイズを指定します。***value*** はバイト単位で指定します。オプションで **k**、**m**、 **g** などを後ろに付けます。

**sw=*value***

1 RAID デバイス内のデータディスクの数を指定するか、または 1 ストライプ内のストライプユニットの数を指定します。

以下の例では、チャンクサイズが 64k、ストライプユニット数が 4 つの RAID デバイスを指定しています。

```
# mkfs.xfs -d su=64k,sw=4 /dev/device
```

XFS ファイルシステムの作成方法についてさらに詳しくは、**man mkfs.xfs** および 『Red Hat Enterprise Linux パフォーマンスチューニングガイド』 の 『Basic Tuning for XFS』 の章を参照してください。

## 6.2. XFS ファイルシステムのマウント

XFS ファイルシステムは追加のオプションを付けなくてもマウントすることができます。たとえば、以下のようになります。

```
# mount /dev/device /mount/point
```

Red Hat Enterprise Linux 7 のデフォルトは inode64 です。

**注記**

mke2fs とは異なり、mkfs.xfs は設定ファイルを使用せず、すべてコマンドラインに指定されます。

### 書き込みバリア

書き込みキャッシュが有効にされているデバイスへの電力供給が停止した場合でもファイルシステムの整合性を確保できるよう XFS はデフォルトで書き込みバリアを使用します。書き込みキャッシュがないデバイスや、書き込みキャッシュがバッテリー駆動型のデバイスなどの場合には、**nobarrier** オプションを使ってバリアを無効にします。

```
# mount -o nobarrier /dev/device /mount/point
```

書き込みバリアについてさらに詳しくは、23章*書き込みバリア*を参照してください。

## 6.3. XFS クォータの管理

XFS クォータサブシステムでディスク領域 (ブロック) およびファイル (inode) の使用量に関する制限の管理を行います。こうした項目の使用量に関する制御や報告は、ユーザー、グループ、ディレクトリーまたはプロジェクトの各レベルで行います。また、ユーザーのクォータ、グループのクォータ、ディレクトリーまたはプロジェクトのクォータはそれぞれ個別に有効にできますが、グループとプロジェクトのクォータについては相互排他的となります。

ディレクトリー単位またはプロジェクト単位で管理を行う場合、XFS は特定のプロジェクトに関連付けられたディレクトリー階層のディスク使用量を管理します。これにより、XFS は複数のプロジェクト間にある組織をまたがった「グループ」の境界を認識します。ディレクトリー単位またはプロジェクト単位でクォータの管理を行うことにより、ユーザーのクォータやグループのクォータを管理する場合よりも広範囲な制御レベルを得ることができます。

XFS のクォータは、マウント時に特定のマウントオプションを付けて有効にします。各マウントオプションは **noenforce** として指定することもできます。これを指定すると、一切の制限を適用することなく使用量に関する報告を可能にします。有効なクォータのマウントオプションは以下の通りです。

- **uquota**/**uqnoenforce** - ユーザーのクォータ
- **gquota**/**gqnoenforce** - グループのクォータ
- **pquota**/**pqnoenforce** - プロジェクトのクォータ

クォータを有効にすると、**xfs_quota** ツールを使ってディスク使用量の制限を設定し、その報告が可能になります。デフォルトでは、**xfs_quota** は *ベーシックモード* でインタラクティブに実行されます。ベーシックモードのサブコマンドは使用量の報告のみを実行します。このコマンドはすべてのユーザーが使用できます。**xfs_quota** の基本的なサブコマンドには以下が含まれます。

**quota** ***username/userID***

任意の ***username*** または数字による ***userID*** の使用量と制限を表示します。

**df**

空き/使用済みブロック数および空き/使用済み inode 数を表示します。

一方、**xfs_quota** には *エキスパートモード* もあります。このモードのサブコマンドで制限を実際に設定することができるため、このサブコマンドを使用できるのは上位の特権を持つユーザーに限られます。エキスパートモードのサブコマンドをインタラクティブに使用するには、**xfs_quota -x** を実行します。エキスパートモードのサブコマンドには以下が含まれます。

**report** ***/path***

特定のファイルシステムのクォータ情報を表示します。

**limit**

クォータの制限を変更します。

ベーシックモードまたはエキスパートモードのサブコマンドの詳細の一覧は、サブコマンドの **help** をご覧ください。

サブコマンドはすべて **-c** オプションを使うとコマンドラインからも直接実行させることができます。エキスパートのサブコマンドの場合は **-x** を使用できます。

**例6.2 サンプルのクォータレポートの表示**

たとえば、**/home** (**/dev/blockdevice** 上) のクォータレポートのサンプルを表示するには、**xfs_quota -x -c 'report -h' /home** コマンドを使用します。これにより、次のような出力が表示されます。

```
User quota on /home (/dev/blockdevice)
                        Blocks
User ID      Used   Soft   Hard Warn/Grace
---------- ---------------------------------
root            0      0      0  00 [------]
testuser   103.4G      0      0  00 [------]
...
```

ユーザー **john** (ホームディレクトリーは **/home/john**) に対して、inode 数のソフトとハードの制限をそれぞれ 500 と 700 に設定するには、次のコマンドを使用します。

```
# xfs_quota -x -c 'limit isoft=500 ihard=700 john' /home/
```

この場合、マウント済みの xfs ファイルシステムである *mount_point* を渡します。

デフォルトでは、**limit** サブコマンドはターゲットがユーザーであると認識します。このため、ターゲットをグループとして制限を設定する場合は **-g** オプションを使います (上記の例の場合も同様)。さらに、フロジェクトの場合は **-p** を使います。

ブロックのソフトとハードの制限は、**isoft** または **ihard** の代わりに **bsoft** または **bhard** を使って設定することもできます。

**例6.3 ブロックのソフトとハードの制限の設定**

たとえば、**/target/path** ファイルシステム上の **accounting** グループに対してソフトとハードのブロック制限をそれぞれ 1000m と 1200m に設定する場合は次のコマンドを使用します。

```
# xfs_quota -x -c 'limit -g bsoft=1000m bhard=1200m accounting'
/target/path
```

**注記**

コマンドの **bsoft** と **bhard** ではバイト単位でカウントします。

**重要**

**man xfs_quota** では、リアルタイムのブロック (**rtbhard**/**rtbsoft**) がクォータの設定時の有効な単位として説明されていますが、本リリースではリアルタイムのサブボリュームは有効にされません。このため、**rtbhard** と **rtbsoft** のオプションは適用できません。

## プロジェクト制限の設定

プロジェクトで管理されているディレクトリー群に対して制限を設定する前に、まずそのディレクトリー群を **/etc/projects** に追加します。複数のプロジェクト ID とプロジェクト名をマッピングさせるために、プロジェクト名を **/etc/projectid** に追加できます。プロジェクトが **/etc/projects** に追加されると、そのプロジェクトディレクトリーを次のコマンドを使って初期化できます。

```
# xfs_quota -x -c 'project -s projectname' project_path
```

初期化したディレクトリー群を持つプロジェクトのクォータ設定は次のように実行します。

```
# xfs_quota -x -c 'limit -p bsoft=1000m bhard=1200m projectname'
```

クォータ設定の汎用ツール (**quota**、**repquota**、および **edquota** など) を使用しても XFS のクォータを操作することができます。ただし、これらのツールは XFS のプロジェクトのクォータでは使用できません。

**重要**

Red Hat は、他の利用可能なすべてのツールの中でも、**xfs_quota** の使用を推奨します。

XFS のクォータの設定方法については、**man xfs_quota**、**man projid(5)**、および **man projects(5)** を参照してください。

## 6.4. XFS ファイルシステムのサイズの拡大

**xfs_growfs** コマンドを使用すると、マウント中の XFS ファイルシステムを拡大することができます。

```
# xfs_growfs /mount/point -D size
```

**-D *size*** オプションでファイルシステムを指定の***size*** (ファイルシステムのブロック数) まで大きくします。 **-D *size*** オプションを指定しない場合、**xfs_growfs** はデバイスで対応できる最大サイズにファイルシステムを拡大します。

**-D *size*** を指定して XFS ファイルシステムを拡大する前に、基礎となるブロックデバイスが将来的にファイルシステムを保持できるサイズであることを確認してください。サイズ変更する場合は対象となるブロックデバイスに適した方法を使用してください。

**注記**

XFS ファイルシステムはマウントしている状態でも拡大することはできますが、逆にそのサイズを縮小することはできません。

ファイルシステムを拡大する方法については **man xfs_growfs** を参照してください。

## 6.5. XFS ファイルシステムの修復

XFS ファイルシステムを修復するには、**xfs_repair** を使用します。

```
# xfs_repair /dev/device
```

**xfs_repair** ユーティリティーは拡張性に優れ、多くの inode を持つ非常に大きなファイルシステムさえも効率的に修復できるよう設計されています。他の Linux ファイルシステムとは異なり、**xfs_repair** は、XFS ファイルシステムが正常にアンマウントされていなかった場合でもブート時には実行されません。正常にアンマウントされなかった場合、**xfs_repair** はマウント時にログを再生し、ファイルシステムの整合性を確保します。

**警告**

**xfs_repair** ユーティリティーは、ダーティーログを持つ XFS ファイルシステムを修復できません。ログを消去するには、XFS ファイルシステムをマウントしてからアンマウントします。ログが破損していて再生できない場合は **-L** (ログのゼロ化を強制) オプションを使ってログの消去を行います (**xfs_repair -L */dev/device***)。ただし、これを行うと破損が悪化したりデータを損失したりする場合があります。

XFS ファイルシステムの修復方法についての詳細は、**man xfs_repair** を参照してください。

## 6.6. XFS ファイルシステムの一時停止

ファイルシステムへの書き込み動作を一時停止にしたり再開したりする場合は **xfs_freeze** を使用します。書き込み動作を一時停止にすることで、ハードウェアベースのデバイススナップショットを使用して、整合性のある状態のファイルシステムをキャプチャーできるようになります。

**注記**

**xfs_freeze** ユーティリティーは **xfsprogs** パッケージで提供されます。これは x86_64 でのみ利用できます。

XFS ファイルシステムを一時停止 (フリーズ) するには、以下のコマンドを使用します。

```
# xfs_freeze -f /mount/point
```

XFS ファイルシステムのフリーズを解除するには、次のコマンドを使用します。

```
# xfs_freeze -u /mount/point
```

LVM のスナップショットを取るには、**xfs_freeze** を使ってファイルシステムを一時停止にする必要はありません。LVM 管理ツールが、スナップショットを取る前に XFS ファイルシステムを自動的に一時停止します。

XFS ファイルシステムのフリーズおよびフリーズ解除に関する詳細は、**man xfs_freeze** をご覧ください。

## 6.7. XFS ファイルシステムのバックアップと復元

XFS ファイルシステムのバックアップと復元には、**xfsdump** と **xfsrestore** の 2 種類のユーティリティーが必要になります。

XFS ファイルシステムのバックアップまたはダンプを実行するには、**xfsdump** ユーティリティーを使用します。Red Hat Enterprise Linux 7 は、テープドライブや通常のファイルイメージへのバックアップに対応しています。また、同じテープへの複数ダンプの書き込みも可能です。**xfsdump** ユーティリティーを使用すると、1 つのダンプを複数のテープに記録できますが、通常のファイルに書き込めるのは 1 つのダンプのみになります。さらに、**xfsdump** は増分バックアップに対応しているため、サイズ、サブツリーまたは inode フラグなどを使用してファイルをフィルターすることで、バックアップからファイルを除外することもできます。

増分バックアップに対応するため、**xfsdump** では *ダンプのレベル* を使って特定ダンプの比較対象となるベースダンプを決定します。ダンプレベル (**0-9**) は **-l** オプションで指定します。フルバックアップを行うには、次のようにしてファイルシステム (***/path/to/filesystem***) で **レベル 0** のダンプを実行します。

```
# xfsdump -l 0 -f /dev/device /path/to/filesystem
```

**注記**

バックアップ先は **-f** で指定します。たとえば、テープドライブには通常**/dev/st0** がバックアップ先として使用されます。**xfsdump** のダンプ先はテープドライブ、通常のファイル、またはリモートテープデバイスのいずれかにできます。

一方、増分バックアップでは最後の **レベル 0** ダンプから変更があったファイルのみをダンプします。**レベル 1** ダンプは。フルダンプ後に行なわれる最初の増分ダンプです。その次の増分ダンプは**レベル 2** となり、最大レベルは **レベル 9** です。**レベル 1** ダンプをテープドライブに実行するには、次のようにします。

```
# xfsdump -l 1 -f /dev/st0 /path/to/filesystem
```

これとは反対に、**xfsrestore** ユーティリティーは **xfsdump** で生成したダンプからファイルシステムを復元します。**xfsrestore** ユーティリティーにはデフォルトの **シンプル** モードと **累積** モードの 2 種類があります。特定のダンプは、*セッション ID* または *セッションラベル* のいずれかを使って行なわれます。このため、ダンプの復元には対応するセッションの ID またはラベルが必要になります。すべてのダンプ (フルダンプと増分ダンプの両方) のセッション ID およびセッションラベルを表示させるには、**-I** オプションを使用します。

```
# xfsrestore -I
```

出力は以下のようになります。

**例6.4 すべてのダンプのセッション ID とセッションラベル**

```
file system 0:
 fs id:  45e9af35-efd2-4244-87bc-4762e476cbab
 session 0:
  mount point: bear-05:/mnt/test
  device:  bear-05:/dev/sdb2
  time:  Fri Feb 26 16:55:21 2010
  session label: "my_dump_session_label"
  session id: b74a3586-e52e-4a4a-8775-c3334fa8ea2c
  level:  0
  resumed: NO
  subtree: NO
  streams: 1
  stream 0:
   pathname: /mnt/test2/backup
   start:  ino 0 offset 0
   end:  ino 1 offset 0
   interrupted: NO
   media files: 1
   media file 0:
    mfile index: 0
    mfile type: data
    mfile size: 21016
    mfile start: ino 0 offset 0
    mfile end: ino 1 offset 0
    media label: "my_dump_media_label"
    media id: 4a518062-2a8f-4f17-81fd-bb1eb2e3cb4f
xfsrestore: Restore Status: SUCCESS
```

### xfsrestore のシンプルモード

**シンプル** モードを使用すると、ユーザーは**レベル 0** のダンプからファイルシステム全体を復元することができます。**レベル 0** ダンプのセッション ID (***session-ID***) を確認してから、次のコマンドを使って ***/path/to/destination*** にファイルシステムの完全な復元を行います。

```
# xfsrestore -f /dev/st0 -S session-ID /path/to/destination
```

**注記**

ダンプの場所は **-f** オプションで指定し、復元するダンプの指定は **-S** または **-L** オプションで行います。セッション ID を指定するには **-S** オプションを使用し、セッションラベルには **-L** オプションを使用します。各ダンプのセッションラベルとセッション ID の両方を表示させるには **-I** オプションを使います。

### xfsrestore の累積モード

**xfsrestore** の **累積** モードを使用すると、**レベル 1** から **レベル 9** への復元など、特定の増分バックアップからのファイルシステムの復元を行うことができます。増分バックアップからのファイルシステムの復元は、**-r** オプションを追加するだけで実行できます。

```
# xfsrestore -f /dev/st0 -S session-ID -r /path/to/destination
```

### インタラクティブな操作

**xfsrestore** ユーティリティーは、任意のダンプからの特定ファイルの抽出、追加、または削除を実行します。**xfsrestore** をインタラクティブに使用する場合は以下のように**-i** オプションを使用します。

```
xfsrestore -f /dev/st0 -i /destination/directory
```

**xfsrestore** による指定デバイスの読み込みが終了すると、インタラクティブなダイアログが開始されます。このダイアログで使用できるコマンドは **cd**、**ls**、**add**、**delete**、および **extract** などになります。コマンドの詳細な一覧については **help** をご覧ください。

XFS ファイルシステムのダンプおよび復元方法についての詳細は、**man xfsdump** および **man xfsrestore** をご覧ください。

## 6.8. XFS ファイルシステムのその他のユーティリティー

Red Hat Enterprise Linux 7 には XFS ファイルシステム管理用のユーティリティーが他にもあります。

**xfs_fsr**

マウントしている XFS ファイルシステムのデフラグを行う際に使用します。 引数を指定せずに呼び出すと、**xfs_fsr** はマウントしているすべての XFS ファイルシステム内にあるすべての通常ファイルのデフラグを行います。このユーティリティーでは、ユーザーによるデフラグの一時停止や、一時停止した部分からのデフラグの再開が可能です。

さらに、**xfs_fsr** ***/path/to/file*** のように指定すると、**xfs_fsr** は、1 ファイルのみのデフラグも可能にします。Red Hat では、ファイルシステム全体に対して定期的なデフラグを行なわないように推奨します。通常 XFS は断片化を回避するため、またファイルシステム全体でのデフラグの副次的な影響として、空き領域の断片化が生じる可能性があるためです。

**xfs_bmap**

XFS ファイルシステム内のファイル群で使用されているディスクブロックのマップを表示します。指定したファイルによって使用されているエクステントや、該当するブロックがないファイルの領域 (ホール) を一覧表示します。

**xfs_info**

XFS ファイルシステムの情報を表示します。

**xfs_admin**

XFS ファイルシステムのパラメーターを変更します。**xfs_admin** ユーティリティーで変更できるのは、アンマウントされているデバイスやファイルシステムのパラメーターのみです。

**xfs_copy**

XFS ファイルシステム全体のコンテンツを 1 つまたは複数のターゲットに同時にコピーします。

また、XFS ファイルシステムのデバッグや分析を行う際に以下のユーティリティーが役に立ちます。

**xfs_metadump**

XFS ファイルシステムのメタデータをファイルにコピーします。Red Hat では、アンマウントされているファイルシステムや読み取り専用のマウントされているファイルシステムをコピーする場合にのみ **xfs_metadump** ユーティリティーの使用をサポートしています。これ以外のファイルシステムにこのユーティリティーを使用すると、破損したダンプまたは整合性のないダンプが生成される可能性があります。

**xfs_mdrestore**

XFS メタダンプイメージ (**xfs_metadump** で生成されたイメージ) をファイルシステムのイメージに復元します。

**xfs_db**

XFS ファイルシステムのデバッグを行います。

上記のユーティリティーについてさらに詳しくは、各 **man** ページをご覧ください。

## 6.9. ext4 から XFS への移行

Red Hat Enterprise 7 における大幅な変更として、デフォルトのファイルシステムが ext4 から XFS へ変更されたことが挙げられます。このセクションでは、XFS ファイルシステムの使用または管理について想定されるいくつかの相違点を強調します。

**注記**

ext4 ファイルシステムは依然として Red Hat Enterprise Linux 7 で完全にサポートされており、必要な場合はインストール時にこれを選択することができます。ext4 から XFS への移行は可能ですが、必須であるという訳ではありません。

### 6.9.1. Ext3/4 のコマンドとツール、および XFS のコマンドとツール

以下の表は、ext3 と ext4 に共通のコマンドと、XFS に固有のコマンドを示しています。

**表6.1 ext3/4 および XFS に共通のコマンド**

| タスク | ext3/4 | XFS |
|---|---|---|
| ファイルシステムの作成 | **mkfs.ext4** または **mkfs.ext3** | **mkfs.xfs** |
| ファイルシステムのチェック | **e2fsck** | **xfs_repair** |
| ファイルシステムのサイズ変更 | **resize2fs** | **xfs_growfs** |
| ファイルシステムのイメージの保存 | **e2image** | **xfs_metadump** および **xfs_mdrestore** |
| ファイルシステムのラベル付けまたはチューニング | **tune2fs** | **xfs_admin** |
| ファイルシステムのバックアップ | **dump** および **restore** | **xfsdump** および **xfsrestore** |

以下の表は、XFS ファイルシステムで機能する汎用ツールを示していますが、XFS バージョンにはさらに多くの固有機能があるため、これらの使用を推奨します。

**表6.2 ext2 と XFS の汎用ツール**

| タスク | ext4 | XFS |
|---|---|---|
| クォータ | **quota** | **xfs_quota** |
| ファイルマッピング | **filefrag** | **xfs_bmap** |

すべての XFS 管理ツールについて詳細な man ページが用意されており、これらのコマンドの多くは、6章*XFS ファイルシステム* で詳細に説明されています。

### 6.9.2. Ext3/4 と XFS の動作および管理上の相違

**ファイルシステムの修復**

ext3/4 とは異なり、XFS はブート時にファイルシステムのチェックを実行しません。完全にジャーナリングされたファイルシステムであるため、このチェックは不要になります。**fsck.xfs** シェルスクリプトも提供されますが、これは initscript の要件のみに対応するため、いずれの有用なアクションも実行しません。

XFS ファイルシステムの修復またはチェックが要求される場合、**xfs_repair** コマンドを使用します。読み取り専用のチェックには **-n** オプションを使用します。

**xfs_repair** コマンドは、ダーティーログを持つファイルシステムでは機能しません。このようなファイルシステムを修復するには、まず **mount** と **unmount** を実行し、ログを再生する必要があります。ログが破損していたり、再生できない場合には、**-L** オプションを使用して、ログのゼロ化を実行できます。

XFS ファイルシステムのファイルシステムの修復についてさらに詳しくは、「XFS」を参照してください。

**メタデータのエラー動作**

ext3/4 ファイルシステムでは、メタデータのエラーが生じた場合、デフォルトで継続させた状態で動作を設定することができます。XFS に回復不能なメタデータのエラーが生じた場合、ファイルシステムはシャットダウンし、**EFSCORRUPTED** エラーが戻されます。システムログには、発生したエラーの詳細情報が記載されます。さらに、必要な場合には **xfs_repair** を実行するこ

とを推奨します。

**クォータ**

XFS クォータは再マウントできないオプションです。クォータを有効にするには、**-o quota** オプションを初回マウント時に指定する必要があります。

クォータパッケージの標準ツールは基本的なクォータ管理タスクを実行できますが (*setquota* および *repquota* などのツール)、**xfs_quota** ツールは、プロジェクトクォータの管理など、XFS 固有の機能に使用できます。

**quotacheck** コマンドは、XFS ファイルシステムに一切影響を与えません。クォータアカウンティングがオンに設定される初回時に、XFS は自動の **quotacheck** を内部で実行します。XFS クォータのメタデータはファーストクラスのジャーナリングされたメタデータオブジェクトであるため、クォータシステムは、クォータが手動でオフに設定されるまで常に一貫性を維持します。

**ファイルシステムのサイズ変更**

XFS ファイルシステムには、ファイルシステムを小さくするユーティリティーがありません。ただし、XFS ファイルシステムは、**xfs_growfs** コマンドを使ってオンラインで拡張できます。

**Inode 数**

256 バイトの inodes を持つ 1T を超えるファイルシステムの場合や、512 バイトの inodes を持つ 2T を超える ファイルシステムの場合、XFS の inode 数は 2 の 32 乗を上回る場合があります。このように inode の数が大きくなると、32　ビットの stat 呼び出しは、**EOVERFLOW** を出して失敗します。通常、アプリケーションはこれらの inode の大きな数を適切に処理できるはずですが、必要な場合は、inode 数を 2 の 32 乗よりも小さくなるように強制するために、**-o inode32** を指定して XFS をマウントすることができます。

**注記**

これは、64 ビットの数字がすでに割り当てられている inodes には影響しません。

このオプションは割り当て動作を変更するものであり、inodes の割り当てに利用できる領域が低レベルのディスクブロックにない場合、ENOSPC が早期に発生する可能性があります。inode32 オプションは、特定の環境で必要とされない限り使用しないでください。

**予測型の事前割り当て**

XFS は、ファイルの書き込み時の EOF を超えたブロックの割り当てに *予測型の事前割り当て* を使用します。これは、同時ストリーミングの書き込みワークロードのためのファイル断片化を防ぐため、NFS サーバーで使用されます。デフォルトでは、この事前割り当ては、ファイルのサイズと共に拡大し、「du」の出力に表示されます。予測型の事前割り当てが設定されたファイルが 5 分間ダーティーにならない場合、この事前割り当ては破棄されます。inode がその前にキャッシュから解放される場合、inode が解放 (reclaim) される時点で事前割り当ては破棄されます。

早期に出される ENOSPC の問題が予測型の事前割り当てによるものであると見られる場合、事前割り当ての固定量を、**-o allocsize=*amount*** マウントオプションで指定することができます。

**断片化に関連したツール**

断片化は、遅延割り当てや予測型の事前割り当てなど、各種のヒューリスティックおよび動作によるものであるため、XFS ファイルシステムに関する大きな問題となることはほとんどありません。ただし、ファイルシステムの断片化を測定したり、ファイルシステムのデフラグを行うための

のツールは存在します。それらの使用については奨励されません。

**xfs_db frag** コマンドは、ファイルシステムの割り当てを、パーセントで表される単一の断片化の数に抽出することを試行します。コマンドの出力については、その意味を理解するのに多くの専門知識が必要になります。たとえば、75% という断片化の割合は、ファイルごとの平均エクステント数が 4 であることのみを意味します。そのため、xfs_db フラグの出力が役立つとは見なされておらず、断片化の問題についてはより注意深い分析を行うことが推奨されます。

**警告**

**xfs_fsr** コマンドは、個別ファイル、またはファイルシステム上のすべてのファイルをデフラグするのに使用できます。後者については、ファイルの場所が破損したり、空き領域の断片化が発生したりする可能性があるのでとくに推奨されません。

# 第7章 Global File System 2

Red Hat Global File System 2 (GFS2) はネイティブのファイルシステムで、直接　Linux カーネルファイルシステムのインターフェースと相互作用します (VFS 層)。クラスターファイルシステムとして実装すると、GFS2 は配信メタデータと複数のジャーナルを採用します。

GFS2 は 64 ビットのアーキテクチャーをベースとし、理論的には 8 エクサバイトのファイルシステムを収容することができます。ただし、現在対応している GFS2 ファイルシステムの最大サイズは 100 テラバイトです。システムで 100 テラバイトを超える GFS2 ファイルシステムが必要な場合には Red Hat サービス担当者にご連絡ください。

ファイルシステムのサイズを確定する際は復元時のニーズを考慮してください。非常に大きなファイルシステムでの **fsck** コマンドの実行は時間がかかり、メモリーを大量に消費することになります。また、ディスクやディスクのサブシステムで障害が発生すると、その復元時間は使用するバックアップメディアの速度によって制限されます。

Red Hat Cluster Suite で設定する場合、Red Hat GFS2 のノードは Red Hat Cluster Suite の設定および管理ツールで設定し、管理することができます。Red Hat GFS2 により、Red Hat クラスター内の GFS2 ノード群でデータを共有でき、このノード群全体で整合性のあるファイルシステム名前空間の単一ビューが提供されます。これにより、異なる複数のノード上の複数プロセスが GFS2 のファイル群を共有できるようになり、その共有方法は同じノード上の複数のプロセスが 1 つのローカルファイルシステム上のファイル群を共有する場合と同様で、両者にはっきり識別できる違いはありません。Red Hat Cluster Suite の詳細については、『Cluster Administration』 ガイドを参照してください。

GFS2 はリニアボリュームまたはミラーボリュームとなる論理ボリューム (LVM で作成) 上に作成してください。Red Hat Cluster Suite の LVM で作成した論理ボリュームは CLVM (クラスター全体での LVM の実装) で管理し、CLVM デーモンの **clvmd** で有効にして Red Hat Cluster Suite のクラスター内で実行します。このデーモンにより、LVM2 を使用してクラスター全体で複数の論理ボリュームを管理できるようになり、クラスター内のすべてのノードが論理ボリュームを共有できるようになります。論理ボリュームマネージャーについては、『論理ボリュームマネージャの管理』 ガイドを参照してください。

**gfs2.ko** カーネルモジュールでは GFS2 ファイルシステムを実装しているため、GFS2 クラスターノード群にロードされます。

クラスター化したストレージでの GFS2 ファイルシステムの作成と設定、およびクラスター化していないストレージでの GFS2 ファイルシステムの作成と設定についての詳細については Red Hat の 『Global File System 2』 ガイドを参照してください。

# 第8章 NFS (Network File System)

*ネットワークファイルシステム* (*NFS*) を利用すると、リモートのホストはネットワーク経由でファイルシステムをマウントし、そのファイルシステムをローカルにマウントしているファイルシステムと同じように操作することができるようになります。また、システム管理者側はリソースをネットワーク上の中央サーバーに統合することができるようになります。

この章では、基本的な NFS の概念と補足的な情報に焦点を絞って説明します。

## 8.1. 動作について

現在、NFS には 2 つのバージョンがあります。NFS バージョン 3 (NFSv3) は安全な非同期書き込みに対応し、NFSv2 に比べエラーの処理機能が強化されています。また、64 ビットのファイルサイズやオフセットに対応するため、クライアントによる 2Gb を超えるファイルデータへのアクセスが可能になります。

NFS バージョン 4 (NFSv4) はファイアーウォールを介してインターネット上で動作します。**rpcbind** サービスを必要としなくなり、ACL に対応し、ステートフルな操作を活用します。Red Hat Enterprise Linux 7 では、NFSv3 および NFSv4 のクライアントに対応しています。NFS 経由でファイルシステムをマウントする場合、サーバーが NFSv4 に対応している場合には Red Hat Enterprise Linux は NFSv4 をデフォルトで使用します。

**注記**

NFS バージョン 2 (NFSv2) は Red Hat のサポート対象外になりました。

NFS の全バージョンで IP ネットワーク経由で実行する *Transmission Control Protocol* (*TCP*) を使用することができ、NFSv4 の場合は TCP が必須になります。NFSv3 では IP ネットワーク経由で実行する *User Datagram Protocol* (UDP) を使用してクライアントとサーバー間のステートレスなネットワーク接続を提供することができます。

UDP で NFSv3 を使用する場合、ステートレスな UDP 接続のプロトコルのオーバーヘッドは TCP より少なくなります (通常の状況を想定)。つまり、クリーンで適度なトラフィックのネットワーク上では、UDP の方がパフォーマンスがよくなります。ただし、UDP はステートレスのため、予期しないサーバーダウンなどが発生すると UDP クライアントはサーバーの要求でネットワークを飽和させ続けます。また、UDP の場合にフレームがなくなると、RPC 要求全体を再転送しなければならなくなります。一方、TCP の場合、再送信が必要なのは失ったフレームのみなります。こうした理由から NFS サーバーへの接続には TCP プロトコルが推奨されます。

マウントおよびロックプロトコルが NFSv4 プロトコルに組み込まれています。サーバーは、一般的に使用されている TCP ポート 2049 でもリッスンします。このように、NFSv4 は **rpcbind** [1]、**lockd**、**rpc.statd** デーモンとの対話が必要ありません。**rpc.mountd** デーモンはエクスポートのセットアップには NFS サーバーで必要ですが、送信オペレーションは行われません。

**注記**

Red Hat Enterprise Linux では、TCP が NFS バージョン 2 および 3 のデフォルトの転送プロトコルになります。UDP は互換性に必要となる場合は使用できますが、その使用範囲についてはできるだけ限定することを推奨しています。NFSv4 には TCP が必須となります。

RPC/NFS デーモンはすべて **'-p'** コマンドラインオプションがあり、ポートを設定することができるため、ファイアウォールの設定が容易になります。

TCP ラッパーによってクライアントにアクセスが許可されると、NFS サーバーは **/etc/exports** 設定ファイルを参照してそのクライアントのエクスポート済みファイルシステムへのアクセスの可否を確認します。アクセス可能なことが確認されると、そのユーザーは全ファイルおよびディレクトリーの操作を行えるようになります。

**重要**

ファイアーウォールを有効にしている Red Hat Enterprise Linux のデフォルトインストールで NFS を正しく動作させるため、IPTables はデフォルトの TCP ポート 2049 で設定してください。IPTables が正しく設定されていないと NFS は正常に動作しません。

NFS の初期化スクリプトおよび **rpc.nfsd** プロセスでは、システム起動中の指定ポートへのバインドが可能になりました。ただし、このポートが使用できない場合や、別のデーモンと競合してしまう場合はエラーが発生しやすくなる可能性があります。

### 8.1.1. 必須サービス

Red Hat Enterprise Linux では、NFS ファイル共有を提供するためにカーネルベースのサポートとデーモンのプロセスの組み合わせを使用します。NFS のすべてのバージョンはクライアントとサーバー間の *Remote Procedure Call* (*RPC*) に依存します。Red Hat Enterprise Linux 7 での RPC サービスは**rpcbind** サービスで制御されます。NFS ファイルシステムの共有やマウントには、実装されている NFS のバージョンに応じて次のようなサービスが連携して動作することになります。

**注記**

**portmap** サービスは、Red Hat Enterprise Linux の旧バージョンで RPC プログラム番号を IP アドレスとポート番号の組み合わせにマッピングする際に使用されていました。このサービスは Red Hat Enterprise Linux 7 では IPv6 に対応するよう **rpcbind** に置き換えられています。この変更についてさらに詳しくは、以下のリンクを参照してください。

- 『TI-RPC / rpcbind support』: http://nfsv4.bullopensource.org/doc/tirpc_rpcbind.php
- 『IPv6 support in NFS』: http://nfsv4.bullopensource.org/doc/nfs_ipv6.php

**nfs**

**service nfs start** により NFS サーバーおよび該当の RPC プロセスが起動し、共有 NFS ファイルシステムの要求が処理されます。

**nfslock**

**service nfslock start** によって RPC プロセスを起動する必須サービスがアクティベートされます。この RPC プロセスにより NFS クライアントはサーバー上にあるファイルをロックできるようになります。

**rpcbind**

**rpcbind** でローカルの RPC サービスからポート予約が受け取られると、これらのポートはリモートの RPC サービスによってアクセス可能であることが公開されます。**rpcbind** は RPC サービスの要求に応答し、要求された RPC サービスへの接続のセットアップを行います。NFSv4 では **rpcbind** は使用されません。

以下の RPC プロセスは NFS サービスと連携して動作します。

**rpc.mountd**

NFS サーバーはこのプロセスを使用して NFSv3 クライアントの **MOUNT** 要求を処理します。要求されている NFS 共有が現在 NFS サーバーで公開されているか、またその共有へのクライアントのアクセスが許可されているかをチェックします。マウントの要求が許可されると rpc.mountd サーバーは **Success** ステータスで応答し、この NFS 共有用の**File-Handle** を NFS クライアントに戻します。

**rpc.nfsd**

**rpc.nfsd** では、サーバーが公開している明示的な NFS のバージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的なデマンドに対応するため Linux カーネルと連携して動作します。このプロセスは **nfs** サービスに対応します。

**lockd**

**lockd** はクライアントとサーバーの両方で実行されるカーネルスレッドです。*Network Lock Manager* (NLM) プロトコルを実装し、NFSv3 のクライアントがサーバー上でファイルのロックを行えるようにします。NFS サーバーが実行中で NFS ファイルシステムがマウントされていれば、このプロセスは常に自動的に起動します。

**rpc.statd**

*Network Status Monitor* (NSM) RPC プロトコルを実装します。NFS サーバーが正常にシャットダウンされず再起動された場合に NFS クライアントに通知します。 **rpc.statd** は **nfslock** サービスによって自動的に起動されるため、ユーザー設定を必要としません。このプロセスは NFSv4 では使用されません。

**rpc.rquotad**

リモートユーザーのユーザークォータ情報を提供します。**rpc.rquotad** は **nfs** サービスによって自動的に起動するため、ユーザー設定を必要としません。

**rpc.idmapd**

**rpc.idmapd** は、ネットワーク上の NFSv4 の名前 (***user@domain*** 形式の文字列) とローカルの UID および GID とのマッピングを行う NFSv4 クライアントアップコールおよびサーバーアップコールを提供します。**idmapd** を NFSv4 で正常に動作させるには、**/etc/idmapd.conf** ファイルを設定する必要があります。NFSv4 を使用する場合はこのサービスが必須となりますが、全ホストに同じ DNS ドメイン名を共有させる場合は必要ありません。ローカルドメインの使用については、ナレッジベース記事の https://access.redhat.com/site/solutions/130783 を参照してください。

## 8.2. pNFS

Red Hat Enterprise Linux 6.4 から、NFS v4.1 標準の一部として Parallel NFS (pNFS) に対応しています。pNFS アーキテクチャーは、NFS の拡張性を向上させ、それに加えてパフォーマンスが強化される場合もあります。つまり、サーバーが pNFS も実装した場合、クライアントは同時に複数サーバーを介してデータにアクセスできるようになります。3 つのストレージプロトコルまたはレイアウト (ファイル、オブジェクト、ブロック) に対応します。

**注記**

このプロトコルは、ファイル、オブジェクト、ブロックの 3 つの pNFS レイアウトタイプに対応します。ただし、Red Hat Enterprise Linux 6.4 クライアントはファイルレイアウトタイプのみに対応しているため、オブジェクトとブロックのレイアウトタイプはテクノロジープレビューに含まれます。

この機能を有効にするには、pNFS が有効にされているサーバーからのマウントで以下のマウントオプションのいずれかを使用してください。

```
-o v4.1
```

サーバーで pNFS を有効にした後に、**nfs_layout_nfsv41_files** カーネルは最初のマウントで自動的に読み込まれます。出力のマウントエントリーには **minorversion=1** が含まれているはずです。以下のコマンドを使用してモジュールが読み込まれたことを確認します。

```
$ lsmod | grep nfs_layout_nfsv41_files
```

pNFS の詳細情報は、http://www.pnfs.com を参照してください。

## 8.3. NFS クライアントの設定

**mount** コマンドの使用でクライアント側に NFS 共有をマウントします。その形式は次のようになります。

```
# mount -t nfs -o options host:/remote/export /local/directory
```

このコマンドは以下のような変数を使用します。

***options***

マウントオプションのカンマ区切りの一覧です。有効な NFS マウントオプションについては、「一般的な NFS マウントオプション」を参照してください。

***サーバー***

マウント予定のファイルシステムをエクスポートするサーバーのホスト名、IP アドレス、または完全修飾型ドメイン名

***/remote/export***

サーバーからエクスポートされるファイルシステム/ディレクトリー、つまり、マウントするディレクトリー

***/local/directory***

*/remote/export* をマウントする必要のあるクライアントの場所

Red Hat Enterprise Linux 7 で使用される NFS プロトコルのバージョンは、**mount** オプション

**nfsvers**、または**vers** で識別できます。デフォルトでは、**mount** は、**mount -t nfs** の形で NFSv4 を使用します。サーバーが NFSv4 をサポートしない場合は、サーバーでサポートされているバージョンにクライアントが自動的に格下げをします。**nfsvers**/**vers** オプションを使用して、サーバーでサポートされない特定のバージョンを渡すと、マウントは失敗します。また、ファイルシステムタイプ nfs4 も、レガシー用に使用可能です。これは、**mount -t nfs -o nfsvers=4 *host*:*/remote/export* */local/directory*** の実行と同じ意味を持ちます。

詳細情報については、**man mount** を参照してください。

NFS 共有が手動でマウントされると、その共有は次のブートでは自動的にマウントされません。Red Hat Enterprise Linux はブート時にリモートファイルシステムを自動的にマウントするために以下の 2 つの方法を提供します。**/etc/fstab** ファイルと **autofs** サービスです。詳細については、「**/etc/fstab** を使用した NFS ファイルシステムのマウント」 と 「**autofs**」 を参照してください。

### 8.3.1. /etc/fstab を使用した NFS ファイルシステムのマウント

別のマシンから NFS 共有をマウントする代替方法の 1 つとして、**/etc/fstab** ファイルに行を追加する方法があります。その行は、NFS サーバーのホスト名、サーバー上のエクスポートされるディレクトリー、および NFS 共有がマウントされるローカルマシン上のディレクトリーを記述している必要があります。**/etc/fstab** ファイルを修正するには root でなければなりません。

> **例8.1 構文の例**
>
> **/etc/fstab** 内に入れる行の一般的な構文は以下のようになります。
>
> ```
> server:/usr/local/pub    /pub    nfs    defaults 0 0
> ```

マウントポイントである **/pub** はこのコマンドを実行する前にクライアントマシン上に存在しなければなりません。クライアントシステムの **/etc/fstab** にこの行を追加した後は、コマンド**mount /pub** を使用すると、マウントポイント **/pub** がサーバーからマウントされます。

**/etc/fstab** ファイルは、ブート時に **netfs** サービスによって参照されます。そのため、 NFS 共有を参照する行は、ブートプロセス中に手動で **mount** コマンドを入力するのと同じ効果を発揮します。

NFS エクスポートをマウントするための有効な **/etc/fstab** エントリーには、以下の情報が含まれている必要があります。

```
server:/remote/export /local/directory nfs options 0 0
```

変数である、*server*、*/remote/export*、*/local/directory*、および *options* は手動で NFS 共有をマウントする際に使用するものと同じです。各変数の定義については、「NFS クライアントの設定」 を参照してください。

**注記**

マウントポイント */local/directory* は、**/etc/fstab** が読み込まれる前にクライアント上に存在しなければなりません。そうでないと、マウントは失敗します。

**/etc/fstab** の詳細情報については、**man fstab** を参照してください。

## 8.4. autofs

**/etc/fstab** を使用する場合、ユーザーが NFS でマウントしたファイルシステムにそれほど頻繁にはアクセスしなくても、システムにはその使用頻度に関係なくマウントしているファイルシステム専用のリソースを維持しなければならないという弱点があります。マウント数が 1 ~ 2 に限られている場合は問題になりませんが、1 度に数多くのシステムに対して複数のマウントを維持する場合にはシステム全体のパフォーマンスに影響を与える可能性があります。この **/etc/fstab** の代わりとなるものがカーネルベースの **automount** ユーティリティーです。自動マウント機能は次の 2 つのコンポーネントで構成されます。

- ファイルシステムを実装するカーネルモジュール
- 他のすべての機能を実行するユーザー領域デーモン

**automount** ユーティリティーでは NFS ファイルシステムの自動マウントおよび自動アンマウントが可能なため (オンデマンドによるマウント機能)、システムのリソースを節約することができます。このユーティリティーは、AFS、SMBFS、CIFS、およびローカルのファイルシステムなど他のファイルシステムをマウントする場合にも使用することができます。

**重要**

nfs-utils パッケージは「NFS ファイルサーバー」および「ネットワークファイルシステムクライアント」の両グループの一部になっているため、デフォルトではベースグループと一緒にインストールされなくなりました。このため、NFS 共有を自動マウントする前に、nfs-utils がシステムにインストールされていることを確認してください。

また、autofs も「ネットワークファイルシステムクライアント」グループの一部です。

**autofs** は、デフォルトの主要設定ファイルとして **/etc/auto.master** (マスターマップ) を使用します。これは、ネームサービススイッチ (NSS) のメカニズムと **autofs** 設定 (**/etc/sysconfig/autofs**) を使用して別のネットワークソースと名前を使用するように変更できます。**autofs** バージョン 4 デーモンのインスタンスはマスターマップ内に設定された各マウントポイントに対して実行されるため、任意のマウントポイントに対してコマンドラインから手動で実行することが可能でした。しかし、**autofs** バージョン 5 では、設定されたすべてのマウントポイントは 1 つのデーモンを使って管理されるため、これを実行することができなくなりました。すべての自動マウントはマスターマップ内で設定しなければなりません。これは業界標準となる他の自動マウント機能の一般的な要件と一致します。マウントポイント、ホスト名、エクスポートしたディレクトリー、および各種オプションは各ホストに対して手作業で設定するのではなく、すべて 1 つのファイルセット (またはサポートされている別のネットワークソース) 内に指定することができます。

### 8.4.1. autofs バージョン 5 の改善点 (バージョン 4 との比較)

**autofs** バージョン 5 をバージョン 4 と比較した場合の特長を以下に示します。

**ダイレクトマップサポート**

ファイルシステム階層内の任意のポイントでファイルシステムを自動マウントするメカニズムは **autofs** 内の複数のダイレクトマップで提供されます。ひとつのダイレクトマップはそのマスターマップ内の **/-** マウントポイントで表されます。 ひとつのダイレクトマップ内にある複数のエントリにはキーとして絶対パス名が含まれます (インダイレクトマップでは相対パス名が使用される)。

**レイジーマウントとアンマウントのサポート**

一つのキーの配下の複数マウントポイントから構成される階層はマルチマウントマップの複数エントリで表されます。 **-hosts** マップなどがその例で、 一般的には **/net/*host*** 配下の任意のホストにあるエクスポートをすべてマルチマウントマップのエントリとして自動マウントするために使用します。 **-hosts** マップを使用する場合、 **/net/*host*** に対して **ls** を行うと *host* に

ある各エクスポート用の *autofs* で起動するマウントをマウントします。 つまり、 アクセスが発生するとエクスポートがマウントされ、 一定時間アクセスがないとマウントの有効期限が切れます。 サーバーにアクセスする際に大量のエクスポートがある場合、 必要とされるアクティブなマウント数を大幅に減少させることができます。

**LDAP サポートの強化**

**autofs** の設定ファイル (**/etc/sysconfig/autofs**) により、 サイトが実装する**autofs** のスキーマを指定できる仕組みが提供されます。 このため、 アプリケーション自体で試行錯誤してスキーマを確定する必要がなくなります。 また、 認証済みの LDAP サーバーへのバインドにも対応するようになり、 一般的な LDAP サーバー実装でサポートしているほとんどのメカニズムを使用するようにます。 このため **/etc/autofs_ldap_auth.conf** という新しい設定ファイルが追加されています。 使用法はこの XML 形式を使用するデフォルト設定ファイル自体に記載されています。

**Name Service Switch (nsswitch) 設定を適切に処理**

特定の設定データをどこから取り込むのかを指定する場合に使用するのが Name Service Switch 設定ファイルです。 データのアクセスには同一のソフトウェアインターフェースを維持しながら、 管理者が使用するバックエンドのデータベースの選択には柔軟性を持たせることができるようになります。 バージョン 4 の自動マウント機能では NSS 設定の処理が徐々に向上されていましたが十分とは言えませんでした。 一方、 Autofs バージョン 5 ではニーズに対応できる完全実装となります。

このファイルでサポートされている構文については **man nsswitch.conf** を参照してください。すべての NSS データベースが有効なマップソースであるとは限らないため、無効なマップソースはパーサーによって拒否されることに注意してください。ファイル、**yp**、**nis**、**nisplus**、**ldap**、**hesiod** などが有効なソースになります。

**ひとつの autofs マウントポイントに対して複数のマスターマップのエントリ**

頻繁に使用されるのにまだ説明していないのがダイレクトマウントポイント **/-** に対する複数のマスターマップのエントリーの処理についてです。 各エントリのマップキーがマージされて 1つのマップとして動作します。

> **例8.2 ひとつの autofs マウントポイントに対して複数のマスターマップのエントリ**
>
> 以下のダイレクトマウント用の connectathon テストマップがその一例です。
>
> ```
> /- /tmp/auto_dcthon
> /- /tmp/auto_test3_direct
> /- /tmp/auto_test4_direct
> ```

## 8.4.2. autofs の設定

自動マウント機能の主要設定ファイルは **/etc/auto.master** になります。 これがマスターマップとも呼ばれるもので、 「autofs バージョン 5 の改善点 (バージョン 4 との比較)」 で説明しているように変更することも可能です。 マスターマップにはシステム上で **autofs** によって制御されているマウントポイントおよびその該当設定ファイルまたは自動マウントマップと呼ばれるネットワークソースが記載されています。 マスターマップの形式は以下のようになります。

```
mount-point map-name options
```

この形式で使用されている変数について以下に示します。

***mount-point***

**autofs** のマウントポイント、**/home** などになります。

***map-name***

マウントポイント一覧とマウントポイントがマウントされるファイルシステムの場所が記載されているマップソース名です。 マップエントリの構文を以下に示します。

***options***

オプションが与えられている場合、 該当マップ内のすべてのエントリにそのオプションが適用されます。 エントリ自体にはオプション指定を行いません。 オプションが累積されていた **autofs** バージョン 4 とは異なる動作になります。 混合環境の互換性を実装させるため変更が加えられています。

**例8.3 /etc/auto.master ファイル**

以下に **/etc/auto.master** ファイル内にある行の一例を示します (**cat /etc/auto.master** で表示)。

```
/home /etc/auto.misc
```

マップの一般的な形式はそのマスターマップと同じですが、 マスターマップではエントリの末尾に表示される「オプション (options)」がマウントポイント (mount-point) と場所 (location) の間に表示されます。

```
mount-point   [options]   location
```

この形式で使用されている変数について以下に示します。

***mount-point***

**autofs** のマウントポイントを参照しています。 これは 1 インダイレクトマウントの単一ディレクトリー名であっても、 複数のダイレクトマウント用のマウントポイントの完全パスであっても構いません。 ダイレクトマップとインダイレクトマップの各エントリーキー (上記の ***mount-point***) の後に空白で区切られたオフセットディレクトリー (「/」で始まるサブディレクトリー名) が記載されます。 これがマルチマウントエントリと呼ばれるものです。

***options***

オプションが与えられている場合、そのマップエントリー用のマウントオプションになります。 エントリー自体にはオプション指定を行いません。

***location***

ローカルファイルシステムのパス (Sun マップ形式のエスケープ文字「:」が先頭に付き、 マップ名が「/」で始める)、 NFS ファイルシステム、 他の有効なファイルシステムなどファイルシステムの場所になります。

以下は、マップファイルのサンプルコンテンツです (例: **/etc/auto.misc**)。

```
payroll -fstype=nfs personnel:/dev/hda3
sales -fstype=ext3 :/dev/hda4
```

マップファイルの先頭コラムは **autofs** マウントポイントを示しています (**sales** と **payroll**、 サーバー名が **personnel**)。 2 番目のコラムは **autofs** マウントのオプションを示し、 3 番目のコラムはマウントのソースを示しています。 上記の場合、 **/home/payroll** と **/home/sales** がautofs マウントポイントになります。 **-fstype=** は省略されることが多く、 一般的には正常な動作に特に必要とされません。

ディレクトリーが存在していない場合は自動マウント機能によりそのディレクトリーが作成されます。 自動マウント機能が起動する前からディレクトリーが存在している場合にはそのディレクトリーの削除は行われません。 次の 2 種類のコマンドいずれかを発行すると自動マウント機能のデーモンを起動または再起動させることができます。

- **service autofs start** (自動マウントのデーモンが停止している場合)
- **service autofs restart**

プロセスにより上記の設定を使用して **/home/payroll/2006/July.sxc** などのアンマウントされている **autofs** ディレクトリーへのアクセスが要求されると、 自動マウントのデーモンがそのディレクトリーを自動的にマウントすることになります。 タイムアウトが指定されている場合は、 タイムアウト期間中にそのディレクトリーへのアクセスがないと自動的にアンマウントされます。

次のコマンドを発行すると自動マウントのデーモンの状態を表示させることができます。

```
#  service autofs status
```

### 8.4.3. サイトの設定ファイルを無効化する／拡大する

クライアントシステム上の特定マウントポイントのサイトデフォルト値を無効にする場合に便利です。 たとえば、 次のような状況を想定してみます。

- 自動マウント機能のマップが NIS に格納され、 **/etc/nsswitch.conf** ファイルには次のようなディレクティブがあるとします。

  ```
  automount:    files nis
  ```

- **auto.master** ファイルには次が含まれます。

  ```
  +auto.master
  ```

- NIS の **auto.master** マップファイルには次が含まれます。

  ```
  /home auto.home
  ```

- NIS の **auto.home** マップには次が含まれます。

  ```
  beth        fileserver.example.com:/export/home/beth
  joe        fileserver.example.com:/export/home/joe
  *       fileserver.example.com:/export/home/&
  ```

- **/etc/auto.home** ファイルマップは存在しません。

このような状況でクライアントシステムが NIS マップの **auto.home** を無効にして、 ホームのディレクトリーを別のサーバーからマウントする必要があると仮定します。 クライアントは次のような **/etc/auto.master** マップを使用する必要があります。

```
/home /etc/auto.home
+auto.master
```

**/etc/auto.home** マップには次のエントリが含まれます。

```
*    labserver.example.com:/export/home/&
```

自動マウント機能で処理されるのは 1 番目に出現するマウントポイントのみになるため、**/home** には NIS **auto.home** マップではなく、**/etc/auto.home** のコンテンツが含まれます。

一方、サイト全体の **auto.home** マップにいくつかのエントリを加えて拡大させたい場合は、**/etc/auto.home** ファイルマップを作成して新しいエントリを組み込み、最後に NIS**auto.home** マップを組み込みます。**/etc/auto.home** ファイルマップは次のようになります。

```
mydir someserver:/export/mydir
+auto.home
```

**ls /home** を行うと、 上述の NIS **auto.home** マップの記載にしたがい以下のような出力になります。

```
beth joe mydir
```

**autofs** は読み込み中のファイルマップと同じ名前のファイルマップの内容を組み込まないため、上記の例は期待通りに動作します。このように **autofs** は **nsswitch** 設定内の次のマップソースに移動します。

### 8.4.4. 自動マウント機能のマップの格納に LDAP を使用する

LDAP から自動マウント機能のマップを取得するよう設定しているシステムにはすべて LDAP クライアントのライブラリをインストールしておかなければなりません。 Red Hat Enterprise Linux なら、**openldap** パッケージは **automounter** の依存パッケージとして自動的にインストールされるはずです。LDAP アクセスを設定する際は **/etc/openldap/ldap.conf** ファイルを編集します。 BASE、 URI、スキーマなどが使用するサイトに適した設定になっていることを確認してください。

自動マウント機能のマップを LDAP に格納するために既定された最新のスキーマが **rfc2307bis** に記載されています。 このスキーマを使用する場合は、 スキーマの定義のコメント文字を取り除き **autofs** 設定 (**/etc/sysconfig/autofs**) 内にセットする必要があります。

**例8.4 autofs 設定のセッティング**

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

設定内でコメントされていないスキーマエントリが上記だけであることを確認します。**automountKey** は **rfc2307bis** スキーマの **cn** 属性を置換します。**LDIF** のサンプル設定を以下に示します。

**例8.5 LDF の設定**

```
# extended LDIF
#
```

```
# LDAPv3
# base <> with scope subtree
# filter: (&(objectclass=automountMap)(automountMapName=auto.master))
# requesting: ALL
#

# auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto.master

# extended LDIF
#
# LDAPv3
# base <automountMapName=auto.master,dc=example,dc=com> with scope
subtree
# filter: (objectclass=automount)
# requesting: ALL
#

# /home, auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: automount
cn: /home

automountKey: /home
automountInformation: auto.home

# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (&(objectclass=automountMap)(automountMapName=auto.home))
# requesting: ALL
#

# auto.home, example.com
dn: automountMapName=auto.home,dc=example,dc=com
objectClass: automountMap
automountMapName: auto.home

# extended LDIF
#
# LDAPv3
# base <automountMapName=auto.home,dc=example,dc=com> with scope
subtree
# filter: (objectclass=automount)
# requesting: ALL
#

# foo, auto.home, example.com
dn: automountKey=foo,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: foo
automountInformation: filer.example.com:/export/foo
```

```
# /, auto.home, example.com
dn: automountKey=/,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: /
automountInformation: filer.example.com:/export/&
```

## 8.5. 一般的な NFS マウントオプション

リモートホストに NFS を使用してファイルシステムをマウントする以外にも、マウントした共有を簡単に使用できるようマウント時に指定できるオプションがあります。これらのオプションは、**mount** コマンド、**/etc/fstab** 設定、**autofs** などを手作業で実行する場合に使用できます。

以下に NFS マウントに一般的に使用されているオプションを示します。

**intr**

サーバーがダウンした場合やサーバーにアクセスできない場合に NFS 要求の割り込みを許可します。

**lookupcache=*mode***

特定マウントポイントのディレクトリーエントリのキャッシュをカーネルにどのように管理させるかを指定します。*mode* に使用できる引数は、**all**、**none**、 **pos**/**positive** になります。

**nfsvers=*version***

使用する NFS プロトコルのバージョンを指定します。*version* は 2、 3、 4 のいずれかになります。複数の NFS サーバーを実行するホスト群に便利です。バージョン指定がない場合、NFS はカーネルおよび **mount** コマンドで対応している最近のバージョンを使用します。

**vers** オプションは **nfsvers** と同一であり、互換性を持たせる目的で本リリースに含まれています。

**noacl**

ACP の処理をすべてオフにします。旧式の Red Hat Enterprise Linux、Red Hat Linux、Solaris などの古いバージョンと連動させる場合に必要となることがあります。こうした旧式のシステムには最新の ACL テクノロジーとの互換性がないためです。

**nolock**

ファイルのロック機能を無効にします。この設定は旧式の NFS サーバーに接続する場合に必要となることが時折あります。

**noexec**

マウントしたファイルシステムでバイナリーが実行されないようにします。互換性のないバイナリーを含む Linux 以外のファイルシステムをマウントしている場合に便利です。

**nosuid**

**set-user-identifier** または **set-group-identifier** ビットを無効にします。リモートユーザーが **setuid** プログラムを実行しても必要以上の特権を得られないようにします。

**port=*num***

**port=num** - NFS サーバーポートの数値を指定します。***num*** が **0** (デフォルト) の場合、**mount** は、使用するポート番号についてリモートホストの **rpcbind** サービスのクエリーを実行します。リモートホストの NFS デーモンがその **rpcbind** サービスに登録されていない場合は、標準の NFS ポート番号 TCP 2049 が代わりに使用されます。

**rsize=*num* および wsize=*num***

一度に転送するデータブロックサイズ (*num* はバイト単位) を大きめに設定することで NFS 通信の読み込み (**rsize**) と書き込み (**wsize**) の速度が上がります。旧式の Linux カーネルやネットワークカードの場合には、ブロックサイズが大きくなると正しく動作しなくなるものがあるため、これらの値を変更する際には注意してください。NFSv3 の場合、読み込みと書き込みのいずれのパラメーターもデフォルト値は 8192 に設定されます。NFSv4 の場合、デフォルト値はいずれも 32768 に設定されます。

**sec=*mode***

NFS 接続の認証時に利用するセキュリティータイプを指定します。デフォルトの設定は **sec=sys** になります。この設定は、ローカルの UNIX UID と GID を使用します。それらは NFS 操作の認証を行うために **AUTH_SYS** を使用します。

**sec=krb5** はユーザー認証にローカルの UNIX UID と GID ではなく Kerberos V5 を使用します。

**sec=krb5i** はユーザー認証に Kerberos V5 を使用し、データの改ざんを防ぐため安全なチェックサムを使って NFS 操作の整合性チェックを行います。

**sec=krb5p** はユーザー認証に Kerberos V5 を使用し、整合性チェックを実行し、トラフィックの傍受を防ぐため NFS トラフィックの暗号化を行います。これが最も安全な設定になりますが、パフォーマンスのオーバーヘッドも最も高くなります。

**tcp**

NFS マウントが TCP プロトコルを使用するよう指示します。

**udp**

NFS マウントが UDP プロトコルを使用するよう指示します。

オプションの完全一覧および各オプションの詳細情報は、**man mount** および **man nfs** を参照してください。

## 8.6. NFS の起動と停止

NFS サーバーを実行する場合は **rpcbind**[1] サービスを実行していなければなりません。**rpcbind** がアクティブであることを確認するには次のコマンドを使用します。

```
# service rpcbind status
```

**rpcbind** サービスが実行中の場合は **nfs** サービスを起動することができます。次のコマンドを実行して NFS サーバーを起動します。

```
# service nfs start
```

さらに、NFS クライアントおよびサーバーが正しく動作するよう **nfslock** も起動する必要があります。次のコマンドを使って NFS ロックを起動します。

```
# service nfslock start
```

NFS がブート時に起動するよう設定されている場合には、**nfslock** も起動するよう設定されているか確認してください。これは、**chkconfig --list nfslock** を実行して確認します。**nfslock** が **on** に設定されていない場合、コンピューターを起動するたびに **service nfslock start** を手動で実行しなければならないことに注意してください。ブート時に **nfslock** を自動的に起動させるには **chkconfig nfslock on** を使用します。

**nfslock** が必要なのは NFSv3 のみです。

サーバーを停止させるには、以下を使用します。

```
# service nfs stop
```

**restart** オプションは NFS をいったん停止させてから起動し直す手順を一度に行うことができる短縮オプションです。NFS の設定ファイルを変更した後にその変更を有効にする際の最も効率的な方法です。このサーバーの再起動を行うには、以下を使用します。

```
# service nfs restart
```

**condrestart** (*条件付きの再起動*) オプションでは、**nfs** が現在実行中の場合にのみ起動を行います。nfs が実行していない場合にはデーモンの起動を行わないため、このオプションはスクリプトなどで使用する場合に便利です。サーバーの条件付き再起動を行うには、以下を使用します。

```
# service nfs condrestart
```

サービスを再起動せずに NFS サーバー設定ファイルの再読み込みを実行するには、以下のように入力します。

```
# service nfs reload
```

## 8.7. NFS サーバーの設定

NFS サーバーの設定には 2 つの方法があります。

- NFS の設定ファイル、**/etc/exports** を手動で編集する方法。
- コマンド **exportfs** の使用により、コマンドラインを介して実行する方法。

### 8.7.1. /etc/exports 設定ファイル

**/etc/exports** ファイルは、リモートホストにエクスポートされるファイルシステムを制御して、オプションを指定します。以下の構文ルールに従います。

- 空白行は無視する。
- コメントを追加するには、その行をハッシュ記号 (#) で始める。
- 長い行はバックスラッシュ (\) を使って折り返す。
- エクスポートされるファイルシステムはそれぞれ独自の行に置く。
- エクスポートされるファイルシステムの後に配置される許可ホストの一覧はすべて、空白文字で分離する。

- 各ホストのオプションはホスト識別子の直後にある括弧内に配置し、ホストと最初の括弧の間に空白を入れない。

エクスポートされるファイルシステムの各エントリーには以下の構造があります。

```
export host(options)
```

前述の構造は以下の変数を使用します。

***export***

エクスポートされるディレクトリー

***host***

エクスポートを共有するホストまたはネットワーク

***options***

*host* に使用するオプション

各ホストに特定のオプションを付けて複数のホストを指定することができます。これを実行するにはホスト名の後にそれぞれのオプションを以下のように (括弧内に) 付けて、空白区切り行と同じ行にホストを一覧表示します。

```
export host1(options1) host2(options2) host3(options3)
```

ホスト名を指定するための異なるメソッドに関する詳細情報については、「ホスト名の形式」を参照してください。

最も簡素な形式では、**/etc/exports** ファイルはエクスポートしたディレクトリーとそれにアクセス許可のあるホストを指定するだけです。以下の例のようになります。

**例8.6 /etc/exports ファイル**

```
/exported/directory bob.example.com
```

ここで、**bob.example.com** は、 NFS サーバーから **/exported/directory/** をマウントできます。この例にはオプションが指定されていないため、NFS は **デフォルト** の設定を使用します。

デフォルトの設定は以下のようになります。

**ro**

エクスポートしたファイルシステムは読み込み専用です。リモートホストは、ファイルシステム上で共有されているデータの変更はできません。ファイルシステムに対して変更 (読み込み/書き込み) ができるようにするには、**rw** オプションを指定します。

**sync**

NFS サーバーは以前の要求で発生した変更がディスクに書き込まれるまでは、要求に応答しません。それに代わって非同期書き込みを有効にするには、オプション **async** を指定します。

**wdelay**

NFS サーバーは、別の書き込み要求が間近に来ていると判定すると、ディスクへの書き込みを遅らせます。別々の書き込みコマンドによるディスクへのアクセス回数を低減できるため、これが

書き込みのワークロードを低下させてパフォーマンスを向上します。これを無効にするには、**no_wdelay** を指定します。デフォルトの **sync** オプションも指定されている場合にのみ **no_wdelay** は利用可能になります。

**root_squash**

これは、(ローカルではなく) **リモート**で接続している root ユーザーが root 権限を持つことを阻止するものです。その代わりに、 NFS サーバーは、そのユーザーにユーザー ID **nfsnobody** を割り当てます。これが、効果的にリモートの root ユーザーの権力を最低のローカルユーザーレベルへと「押しつぶし」て、リモートサーバー上での無許可の書き込み可能性を阻止します。この root squashing を無効にするには、**no_root_squash** を指定します。

すべてのリモートユーザー (root を含む) を潰す (squash) には、**all_squash** を使用します。NFS サーバーが特定のホストからリモートユーザーに対して割り当てるべきユーザー ID とグループ ID を指定するには、**anonuid** と **anongid** のオプションをそれぞれ以下のように使用します。

```
export host(anonuid=uid,anongid=gid)
```

ここで、*uid* と *gid* はそれぞれ、ユーザー ID 番号およびグループ ID 番号です。**anonuid** と **anongid** のオプションは、リモート NFS ユーザーが共有するための特別なユーザーおよびグループアカウントの作成を可能にします。

デフォルトでは、*access control lists (アクセス制御リスト)* (*ACL*) は、Red Hat Enterprise Linux の下で NFS によってサポートされています。この機能を無効にするには、ファイルシステムをエクスポートする際に **no_acl** オプションを指定します。

すべてのエクスポートされたファイルシステムのデフォルトは明示的に上書きする必要があります。例えば、**rw** オプションを指定しないと、エクスポートしたファイルシステムは読み込み専用として共有されます。以下のサンプルは、**/etc/exports** からの行であり、これは 2 つのデフォルトオプションを上書きしています。

```
/another/exported/directory 192.168.0.3(rw,async)
```

この例では、**192.168.0.3** は **/another/exported/directory/** を読み込み/書き込みでマウントできて、ディスクへの書き込みはすべて非同期となります。エクスポートオプションの詳細については、**man exportfs** を参照してください。

さらに、デフォルト値が指定されていない他のオプションも利用可能です。これらのオプションとして、サブツリーチェックを無効にする機能、安全でないポートからのアクセスの許可、および安全でないファイルロックの許可 (一部の初期 NFS クライアント実装で必要)などがあります。頻繁には使用しないこれらのオプションの詳細については **man exports** を参照してください。

**重要**

**/etc/exports** ファイルの形式は、特に空白の使用に関しては非常に厳格です。常にホストとエクスポートされるファイルシステムの間、およびホスト同士の間を空白で隔離することを忘れないでください。しかし、コメント行以外では、ファイル内の他の場所には空白が存在すべきではありません。

例えば、以下の２つの行は同じ意味ではありません:

```
/home bob.example.com(rw)
/home bob.example.com (rw)
```

最初の行は **bob.example.com** からのユーザーにのみ **/home** ディレクトリーへの読み込み/書き込みアクセスを許可します。２番目の行は **bob.example.com** からのユーザーにディレクトリーを読み込みのみで(デフォルト)マウントを許可して、他の人々には読み込み/書き込みでマウントすることを許可します。

### 8.7.2. exportfs コマンド

NFS 経由でリモートユーザーにエクスポートされているすべてのファイルシステム、並びにそれらのファイルシステムのアクセスレベルは **/etc/exports** ファイル内に一覧表示してあります。**nfs** サービスが開始すると、**/usr/sbin/exportfs** コマンドが起動してこのファイルを読み込み、実際のマウントプロセスのために制御を **rpc.mountd** (NFSv2 または NFSv3の場合) とその後に**rpc.nfsd** に渡します。この時点でファイルシステムがリモートユーザーに使用可能になります。

**/usr/sbin/exportfs** コマンドを手動で発行すると、root ユーザーは NFS サービスを再開始せずにディレクトリーをエクスポートするか、しないかを選択できるようになります。適切なオプションが与えられると、**/usr/sbin/exportfs** コマンドはエクスポートしたファイルシステムを **/var/lib/nfs/xtab** に書き込みます。**rpc.mountd** はファイルシステムへのアクセス権限を決定する際に **xtab** ファイルを参照するため、エクスポートしたファイルシステム一覧への変更はすぐに反映されます。

**/usr/sbin/exportfs** で利用可能な一般的なオプションの一覧は以下のようになります。

**-r**

**/etc/exports** 内に一覧表示してあるすべてのディレクトリーから **/etc/lib/nfs/xtab** 内に新しいエクスポート一覧を構成することにより、それらのディレクトリーがエクスポートされることになります。結果的にこのオプションが **/etc/exports** 内のいずれかの変更でエクスポート一覧をリフレッシュすることになります。

**-a**

**/usr/sbin/exportfs** に渡される他のオプションに応じて、すべてのディレクトリーがエクスポートされるか、またはされないことになります。他のオプションが指定されない場合は、**/usr/sbin/exportfs** は、**/etc/exports** 内に指定してあるすべてのファイルシステムをエクスポートします。

**-o *file-systems***

**/etc/exports** 内に一覧表示されていないエクスポートされるディレクトリーを指定します。*file-systems* の部分を、エクスポートされるファイルシステムに置き換えます。これらのファイルシステムは、**/etc/exports** で指定されたものと同じフォーマットでなければなりません。このオプションは、多くの場合エクスポート用ファイルシステム一覧に永久追加する前

に、エクスポート予定のファイルシステムをテストするために使用されます。**/etc/exports** 構文の詳細情報については、「**/etc/exports** 設定ファイル」を参照してください。

**-i**

**/etc/exports** を無視します。コマンドラインから出されたオプションのみが、エクスポート用ファイルシステムの定義に使用されます。

**-u**

すべての共有ディレクトリーをエクスポートしません。コマンド **/usr/sbin/exportfs -ua** は、すべての NFS デーモンを稼働状態に維持しながら、NFS ファイル共有を保留します。NFS 共有を再度有効にするには、**exportfs -r** を使用します。

**-v**

詳細表示を意味します。エクスポート、または非エクスポートのファイルシステムが **exportfs** コマンドの実行の際により詳細に表示されます。

**exportfs** コマンドに対してオプションを渡さない場合、コマンドは現在エクスポートされているファイルシステムの一覧を表示します。**exportfs** コマンドの詳細については、**man exportfs** を参照してください。

#### 8.7.2.1. NFSv4 で exportfs の使用

Red Hat Enterprise Linux 7 では、提示されるファイルシステムは自動的に同じパスを使用して NFSv3 およびNFSv4 クライアントで利用可能になるため、NFSv4 のエクスポートを設定するための特別なステップは必要ありません。これは、以前のバージョンとの相違点です。

クライアントによる NFSv4 の使用を防止するには、**/etc/sysconfig/nfs** で **RPCNFSDARGS= -N 4** を選択することにより、それをオフにします。

### 8.7.3. ファイアウォール背後での NFS の実行

NFS は **rpcbind** を必要としますが、これは RPC サービス用のポートを動的に割り当てて、ファイアーウォールルールの設定中に問題を起こす可能性があります。クライアントがファイアウォール背後で NFS 共有にアクセスできるようにするには、**/etc/sysconfig/nfs** 設定ファイルを編集して必要な RPC サービスが実行するポートを制御するようにします。

**/etc/sysconfig/nfs** はデフォルトですべてのシステム上には存在していないかも知れません。存在しない場合は、作成して *port* の部分を未使用のポート番号で入れ替えることで、後に続く変数を追加します。(別の方法として、ファイルが存在する場合は、デフォルトのエントリをアンコメント化して必要に応じて変更します):

**MOUNTD_PORT=*port***

**mountd** (**rpc.mountd**) が使用する TCP と UDP ポートを制御します。

**STATD_PORT=*port***

状態 (**rpc.statd**) が使用する TCP と UDP ポートを制御します。

**LOCKD_TCPPORT=*port***

**nlockmgr** (**lockd**) が使用する TCP ポートを制御します。

**LOCKD_UDPPORT=*port***

**nlockmgr** (**lockd**) が使用する UDP ポートを制御します。

NFS が開始に失敗した場合、**/var/log/messages** をチェックします。通常は、すでに使用中のポート番号を指定した場合に NFS は開始に失敗します。**/etc/sysconfig/nfs** を編集した後に、**service nfs restart** を使用して NFS サービスを再開始します。**rpcinfo -p** コマンドを実行すると、その変化を確認できます。

NFS を許可するようにファイアウォールを設定するには、以下のステップを実行します。

**手順8.1 NFS を許可するためのファイアウォールの設定**

1. NFS 用に TCP と UDP ポート 2049 を許可します。
2. TCP と UDP ポート 111 (**rpcbind**/**sunrpc**) を許可します。
3. TCP と **MOUNTD_PORT="*port*"** で指定された UDP ポートを許可します。
4. TCP と **STATD_PORT="*port*"** で指定された UDP ポートを許可します。
5. **LOCKD_TCPPORT="*port*"** で指定された TCP ポートを許可します。
6. **LOCKD_UDPPORT="*port*"**で指定された UDP ポートを指定します。

**注記**

NFSv4.0 コールバックがファイアウォールを通過するように許可するには、**/proc/sys/fs/nfs/nfs_callback_tcpport** をセットして、サーバーがクライアント上のそのポートに接続できるようにします。

このプロセスは、NFSv4.1 またはそれ以降には必要ありません。そして **mountd**、**statd**、および **lockd** のための他のポート群は純粋な NFSv4 環境では必要ありません。

### 8.7.3.1. NFS エクスポートの発見

NFS サーバーがエクスポートするファイルシステムを発見する方法は２種類あります。

１つ目は、NFSv2 または NFSv3 をサポートするいずれかのサーバー上で、**showmount** コマンドの使用です。

```
$ showmount -e myserver
Export list for mysever
/exports/foo
/exports/bar
```

２つ目は、NFSv4 をサポートするサーバー上で、**/** をマウントして周囲を見て回ります。

```
# mount myserver:/ /mnt/
#cd /mnt/
exports
# ls exports
foo
bar
```

NFSv4 と更に NFSv2 か NFSv3 のどちらかの２種類をサポートするサーバー上では、上記の両方の方法が機能して同じ結果を出します。

**注記**

Red Hat Enterprise Linux 6 以前には、設定の仕方によって旧来の NFS サーバーは別々のパス経由で NFSv4 クライアントにファイルシステムをエクスポートすることがありました。それらのサーバーではデフォルトで NFSv4 を有効にしていないため、これは通常、問題にはなっていません。

### 8.7.4. ホスト名の形式

ホストは以下の形式にすることができます。

**単独マシン**

完全修飾型ドメイン名 (サーバーで解決可能な形式)、ホスト名 (サーバーで解決可能な形式)、あるいは IP アドレス

**ワイルドカードで指定された一連のマシン**

***** または **?** の文字を使用して文字列の一致を指定します。ワイルドカードは IP アドレスでは使用しないことになっていますが、逆引き DNS ルックアップが失敗した場合には偶然に機能するかも知れません。ワイルドカードを完全修飾型ドメイン名に指定する時は、ドット (**.**) はワイルドカードの一部にはなりません。例えば、***.example.com** は **one.example.com** を範囲に入れますが **one.two.example.com** はその範囲に入りません。

**IP ネットワーク**

*a.b.c.d/z* を使用します。ここで、*a.b.c.d* はネットワークであり、*z* はネットマスクのビット数です (例えば、 192.168.0.0/24)。別の使用可能形式は *a.b.c.d/netmask* となり、ここで *a.b.c.d* がネットワークで、*netmask* がネットマスクです (例えば、 192.168.100.8/255.255.255.0)。

**Netgroup**

形式 *@group-name* を使用します。ここで、*group-name* は NIS netgroup の名前です。

### 8.7.5. RDMA 上の NFS

Red Hat Enterprise Linux 7 では、RDMA 対応のハードウェアがマシンにある限り、RDMA サービスは自動化されます。そのため、以下の手順は RMDA パッケージがマシンのインストール時にインストールされなかった場合にのみ従う必要があります。

**手順8.2 クライアントから RDMA を有効化**

1. RDMA パッケージをインストールします。

```
# yum install rdma
```

2. **initramfs** を再作成します。

```
# dracut -f
```

3. 再起動します。

## 8.8. NFS の保護

ファイルシステム全体を既知の大量のホスト群で透過的に共有する場合に NFS は非常に適しています。ただし、使いやすさがある反面、安全上の各種の問題も付随します。サーバー上に NFS ファイルシステムをエクスポートする際や、クライアントにマウントする場合には次のセクションを考慮に入れるようにしてください。これにより、リスクを最小限に抑えサーバー上のデータをより効果的に保護することができます。

### 8.8.1. AUTH_SYS とエクスポート制御による NFS の保護

従来より、NFS ではエクスポートしたファイルへのアクセスを制御するために 2 種類のオプションを提供しています。

1 つ目は、IP アドレスまたはホスト名のいずれかによって、どのホストにいずれのファイルシステムのマウントを許可するかをサーバー側で制限するオプションです。

2 つ目は、ローカルユーザーと同じ方法で、サーバーが NFS クライアント上のユーザーに対してファイルシステムの権限を強制するオプションです。従来より、**AUTH_SYS** (**AUTH_UNIX** とも言う) を使って行われ、ユーザーの UID や GID の指定はクライアントに依存します。つまり、悪意のあるクライアントや誤って定されたクライアントがこれを誤用し、ファイルへのアクセスを許可すべきではないユーザーに対してファイルへのアクセスを簡単に与えてしまうことができるため注意が必要です。

こうしたリスクを抑えるため、管理者によって共通のユーザーおよびグループ ID へのユーザー権限が取り消されたり、読み取り専用のアクセスに制限されたりすることがよくあります。残念ながら、こうしたソリューションは NFS 共有が当初意図されていた方法で使用されることを制限することになります。

また、NFS ファイルシステムをエクスポートしているシステムで使用している DNSサーバーのコントロールが攻撃者に奪われると、特定のホスト名または完全修飾ドメイン名に関連付けられているシステムが未承認のマシンにポイントされる可能性があります。この時点で、NFS マウントにはこれ以上の安全確保を目的としたユーザー名やパスワード情報の交換が行われないため、この未承認のマシンが NFS 共有のマウントを **許可されたシステムになってしまいます**。

NFS 経由でディレクトリーのエクスポートを行う際にワイルドカードを使用する場合は慎重に行ってください。ワイルドカードの対象が予定よりも広い範囲のシステムを対象とする可能性があります。

また、TCP ラッパーで **rpcbind**[1] サービスへのアクセスを制限することも可能です。**iptables** でルールを作成しても **rpcbind**、**rpc.mountd**、**rpc.nfsd** などによって使用されるポートへのアクセスを制限することができます。

NFS および **rpcbind** に対する安全対策については**man iptables** を参照してください。

### 8.8.2. AUTH_GSS による NFS の保護

NFSv4 のリリースにより、RPCSEC_GSS および Kerberos バージョン 5 の GSS-API メカニズムの実装が義務付けられ、NFS セキュリティーに大きな変革がもたらされました。ただし、RPCSEC_GSS や Kerberos のメカニズムは NFS のいずれのバージョンでも利用可能です。

RPCSEC_GSS Kerberos メカニズムでは、AUTH_SYS の場合のようにファイルにアクセスしているユーザーを正確に表すためにサーバーがクライアントに依存するということがなくなります。代わりに暗号を使ってサーバーに対するユーザーの認証を行うため、悪意あるクライアントは kerberos 情報なしではなりすまし攻撃ができなくなります。

**注記**

NFSv4 サーバーの設定を行う前に、Kerberos チケットを発行するサーバー (KDC) が適切にインストールされ、設定されていることが前提となります。Kerberos は、対称暗号化および信頼できるサードパーティーの KDC を使用してクライアントとサーバーを相互に認証させることができるネットワーク認証システムになります。Kerberos についてさらに詳しくは、Red Hat の 『Identity Management Guide』 を参照してください。

RPCSEC_GSS のセットアップを行うには、次の手順を実行します。

**手順8.3 RPCSEC_GSS のセットアップ**

1. **nfs/client.*mydomain*@*MYREALM*** プリンシパルと **nfs/server.*mydomain*@*MYREALM*** プリンシパルを作成します。
2. クライアントとサーバーの各キータブに該当するキーを追加します。
3. サーバー側で **sec=krb5,krb5i,krb5p** をエクスポートに追加します。継続して AUTH_SYS を許可する場合は、**sec=sys,krb5,krb5i,krb5p** を代わりに追加します。
4. クライアント側で **sec=krb5** (セットアップによっては **sec=krb5i** または **sec=krb5p**) をマウントポイントに追加します。

**krb5**、**krb5i**、**krb5p** のそれぞれの違いなどの詳細については、**exports** および **nfs** の各 man ページを参照していただくか、または 「一般的な NFS マウントオプション」 をご覧ください。

**rpc.svcgssd** と **rpc.gssd** を同時に使用する方法などの**RPCSEC_GSS** フレームワークの詳細については、http://www.citi.umich.edu/projects/nfsv4/gssd/ を参照してください。

#### 8.8.2.1. NFSv4 による NFS の保護

NFSv4 には、旧式の機能や幅広い導入の経緯があるために、POSIX モデルではなく Microsoft Windows NT モデルをベースとした ACL サポートが含まれます。

NFSv4 のもう 1 つの重要なセキュリティー上の特長として、ファイルシステムをマウントする際に **MOUNT** プロトコルを使用する必要がなくなったという点があります。この**MOUNT** プロトコルには、ファイル処理の方法にセキュリティー上の欠点がある可能性のあることが指摘されています。

### 8.8.3. ファイル権限

リモートホストにおいて NFS ファイルシステムが読み取りと書き込みの権限でマウントされると、それぞれの共有ファイルを保護できるのはその権限のみになります。同じユーザー ID 値を共有している 2 ユーザーが同じ NFS ファイルシステムをマウントすると、それらのユーザーはファイルを相互に変更することができます。さらに、クライアントシステムに root としてログインするいずれのユーザーも、**su -** コマンドを使用して NFS 共有経由のファイルにすべてアクセスできます。

デフォルトでは、アクセス制御リスト (ACL) は Red Hat Enterprise Linux 環境下の NFS によってサポートされます。Red Hat はこの機能は有効な状態にしておくことを推奨しています。

NFS はファイルシステムをエクスポートする際に、デフォルトで *root squash* の機能を使用します。これにより、そのローカルマシン上で root ユーザーとして NFS 共有にアクセスするすべてのユーザーの ID は **nobody** に設定されます。Root squash 機能はデフォルトオプションの**root_squash** で制御されます。このオプションの詳細については、「**/etc/exports** 設定ファイル」 を参照してください。できる限りこの root squash 機能は無効にしないでください。

NFS 共有を読み取り専用でエクスポートする場合、**all_squash** オプションの使用を考慮してください。このオプションにより、エクスポートしたファイルシステムにアクセスするすべてのユーザーは **nfsnobody** ユーザーのユーザー ID を取得します。

## 8.9. NFS および rpcbind

**注記**

次のセクションは、後方互換用に **rpcbind** を必要とする NFSv3 の実装のみに適用されます。

**rpcbind**[1] ユーティリティーは、RPC サービスをそれらのサービスがリッスンするポートにマッピングします。RPC のプロセスが開始されるとその開始が **rpcbind** に通知され、そのプロセスがリッスンしているポートおよびそのプロセスが処理することが予想される RPC プログラム番号が登録されます。クライアントシステムによって特定の RPC プログラム番号を使ってサーバー上の **rpcbind** との通信が行われると、**rpcbind** サービスによりクライアントが適切なポート番号にリダイレクトされ、要求されたサービスと通信できるようになります。

RPC ベースのサービスは着信のクライアント要求のすべての接続を確立する際に **rpcbind** に依存します。このため、これらのサービスを起動する前に **rpcbind** を利用可能な状態にする必要があります。

**rpcbind** サービスはアクセス制御に TCP ラッパーを使用するため、**rpcbind** のアクセス制御ルールは RPC ベースの **すべての** サービスに影響します。代わりに、NFS RPC の各デーモンごとにアクセス制御ルールを指定することも可能です。こうしたルールの正確な構文に関しては **rpc.mountd** および **rpc.statd** の **man** ページに記載されている情報を参照してください。

### 8.9.1. NFS および rpcbind に関するトラブルシューティング

**rpcbind**[1] では通信に使用するポート番号と RPC サービス間の調整を行うため、トラブルシューティングを行う際は **rpcbind** を使って現在の RPC サービスの状態を表示させると便利です。**rpcinfo** コマンドを使用すると RPC ベースの各サービスとそのポート番号、RPC プログラム番号、バージョン番号、および IP プロトコルタイプ (TCP または UDP) が表示されます。

**rpcbind** に対して適切な RPC ベースの NFS サービスが有効になっていることを確認するには、次のコマンドを発行します。

```
# rpcinfo -p
```

**例8.7 rpcinfo -p コマンドの出力**

以下に上記コマンドの出力例を示します。

```
program vers proto   port service
      100021    1   udp  32774  nlockmgr
      100021    3   udp  32774  nlockmgr
      100021    4   udp  32774  nlockmgr
      100021    1   tcp  34437  nlockmgr
      100021    3   tcp  34437  nlockmgr
      100021    4   tcp  34437  nlockmgr
      100011    1   udp    819  rquotad
      100011    2   udp    819  rquotad
      100011    1   tcp    822  rquotad
```

```
    100011    2   tcp    822  rquotad
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100005    1   udp    836  mountd
    100005    1   tcp    839  mountd
    100005    2   udp    836  mountd
    100005    2   tcp    839  mountd
    100005    3   udp    836  mountd
    100005    3   tcp    839  mountd
```

NFS サービスが 1 つでも正しく開始されていないと、**rpcbind** は、そのサービスのクライアントから正しいポートに RPC 要求をマッピングできません。多くの場合、NFS が **rpcinfo** 出力にない場合、NFS を再起動すると **rpcbind** に正しく登録され、機能を開始します。

**rpcinfo** の詳細およびオプション一覧については**man** ページを参照してください。

## 8.10. 参照

NFS サーバーの管理は難しい課題となる場合があります。本章では言及していませんが、NFS 共有のエクスポートやマウントに利用できるオプションは多数あります。さらに詳しくは次のソースをご覧ください。

### インストールされているドキュメント

- **man mount** — NFS のサーバー設定およびクライアント設定の両方のマウントオプションに関して総合的に説明しています。
- **man fstab** — ブート時にファイルシステムをマウントするために使用する**/etc/fstab** ファイルの形式について記載しています。
- **man nfs** — NFS 固有のファイルシステムのエクスポートおよびマウントオプションについて詳細に説明しています。
- **man exports** — NFS ファイルシステムのエクスポート時に**/etc/exports** ファイル内で使用する一般的なオプションを表示します。

### 役に立つ Web サイト

- http://linux-nfs.org — プロジェクトの更新状況を確認できる開発者向けの最新サイトです。
- http://nfs.sourceforge.net/ — 開発者向けの旧ホームページですが、役に立つ情報が多数掲載されています。
- http://www.citi.umich.edu/projects/nfsv4/linux/ — Linux 2.6 カーネル用 NFSv4 のリソースです。
- http://www.vanemery.com/Linux/NFSv4/NFSv4-no-rpcsec.html — 2.6 カーネルが同梱されている Fedora Core 2 での NFSv4 について詳細に記載しています。
- http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.4086 — NFS バージョン 4 プロトコルの機能および拡張機能について記載しているホワイトペーパーです。

### 関連書籍

- 『Managing NFS and NIS』 (Hal Stern、Mike Eisler および Ricardo Labiaga 著、O'Reilly & Associates 出版): 各種の NFS エクスポートやマウントオプションについて記載している優れた参考ガイドです。
- 『NFS Illustrated』 (Brent Callaghan 著、Addison-Wesley Publishing Company 出版): NFS と他のネットワークファイルシステムとの比較、NFS 通信がどのように発生するかなどが詳細に紹介されています。

---

[1] Red Hat Enterprise Linux の以前のバージョンで RCP プログラム番号を IP アドレスのポート番号の組み合わせにマッピングする際に使用されていた **portmap** に代わり、**rpcbind** サービスが使用されます。さらに詳しくは、「必須サービス」 を参照してください。

# 第9章 FS-Cache

FS-Cache とはファイルシステムによって使用される永続的なローカルキャッシュのことです。ネットワーク経由で取得されたデータをローカルのディスクにキャッシングします。 ユーザーがネットワーク経由でマウントしているファイルシステムのデータにアクセスする場合にネットワークのトラフィックを最小限に抑えます (NFS など)。

以下に FS-Cache がどのように動作するのかを高次元で図解します。

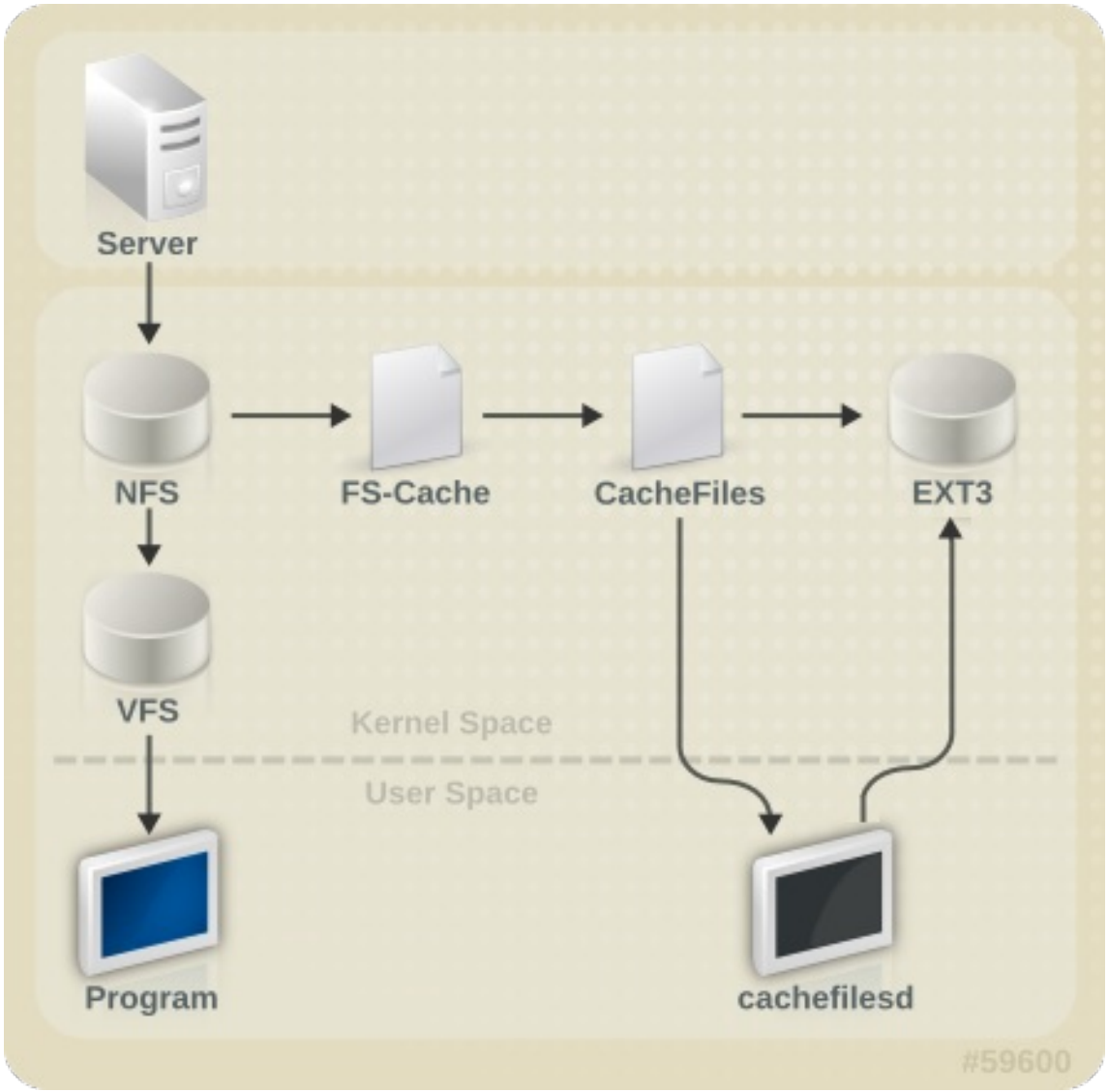



**図9.1 FS-Cache の概要**

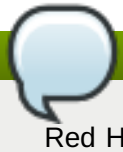
FS-Cache はシステムのユーザーおよび管理者に対してできるだけ透過的になるよう設計されています。Solaris の **cachefs** とは異なり、マウントしたファイルシステムに重ねてマウントすることなくサーバー上のファイルシステムがクライアントのローカルキャッシュと直接相互作用できるようになります。NFS の場合、マウントオプションを使って FS-Cache が有効になっている NFS 共有をマウントするようクライアントに指示します。

FS-Cache はネットワーク経由で動作するファイルシステムの基本的な動作を変更するものではありません。単にファイルシステムにデータをキャッシュできる永続的な場所を提供しているだけです。たとえば、クライアントは FS-Cache が有効になっているかいなかに関わらず NFS 共有をマウントすることができます。また、キャッシュされた NFS はキャッシュに収まらないファイル群を処理することができます (集約

的であるか個別であるかを問わない)。これはファイル群を部分的にキャッシュすることができ、前もって完全に読み込む必要性がないためです。また、FS-Cache はクライアントのファイルシステムドライバーのキャッシュで発生したすべての I/O エラーを非表示にします。

キャッシングのサービスを提供するには、*キャッシュバックエンド* が必要になります。キャッシュバックエンドとは、キャッシングサービスを提供するよう設定されたストレージのドライバーを指します (**cachefiles**)。この場合、FS-Cache には、キャッシュバックエンドとして拡張属性 (ext3 など) や **bmap** に対応するブロックベース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。

FS-Cache はネットワーク経由であるかその他の手段であるかを問わず、ファイルシステムを任意でキャッシュすることはできません。FS-Cache、データの格納および検索、メタデータのセットアップおよび検証などでの相互作用を可能にするよう共有ファイルシステムのドライバーを変更する必要があります。FS-Cache には永続性に対応するためキャッシュしたファイルシステムの *インデックスキー* および *コヒーレンスデータ* が必要になります。インデックスキーはファイルシステムのオブジェクトとキャッシュのオブジェクトを一致させ、コヒーレンスデータはキャッシュのオブジェクトがまだ有効であるかどうかを確定します。



**注記**

Red Hat Enterprise Linux 6.2 および旧バージョンでは、**cachefilesd** がデフォルトではインストールされていないため手作業でインストールを行う必要があります。

## 9.1. 性能に関する保証

FS-Cache では **パフォーマンスの向上は保証していません**。ただし、ネットワーク渋滞を避けることで一貫したパフォーマンスを実現します。キャッシュバックエンドを使用することによりパフォーマンスの低下が起こります。たとえば、キャッシュされた NFS 共有ではネットワーク全体を検索するためディスクへのアクセスが増加します。FS-Cache はできるだけ非同期にするよう試行しますが、これができない同期パス (読み込みなど) があります。

たとえば、2 台のコンピューター間で NFS 共有をキャッシュするために負荷のない GigE ネットワークを介して FS-Cache を使用しても、ファイルのアクセスにおけるパフォーマンスの向上はまったく見られません。むしろ、NFS 要求の場合はローカルディスクではなくサーバーメモリーから対応した方が速くなります。

したがって FS-Cache の使用には、各種の要素間での **妥協** が伴うと言えます。NFS トラフィックのキャッシュに FS-Cache を使用すると、クライアント側の速度は多少遅くなりますが、ネットワークの帯域幅を使用せずローカルに読み取り要求に対応することでネットワークおよびサーバー側の負荷を大幅に低減させることができます。

## 9.2. キャッシュを設定する

現在、Red Hat Enterprise Linux 7 で提供しているのは **cachefiles** キャッシングバックエンドのみになります。**cachefilesd** デーモンにより **cachefiles** が開始され、管理されます。**cachefiles** によるキャッシングサービスの提供方法については **/etc/cachefilesd.conf** ファイルで制御します。この種のキャッシュバックエンドを設定する場合は、**cachefilesd** パッケージをインストールしておく必要があります。

キャッシュバックエンドで最初に行うのはキャッシュとして使用するディレクトリーの設定です。次のパラメーターを使用して行います。

```
$ dir /path/to/cache
```

一般的に、キャッシュバックエンドのディレクトリーは以下のように **/etc/cachefilesd.conf** 内に **/var/cache/fscache** として設定されます。

```
$ dir /var/cache/fscache
```

FS-Cache は、***/path/to/cache*** をホストするファイルシステム内にキャッシュを格納します。ノートブックでは root ファイルシステム (**/**) をホストのファイルシステムとして使用することをお勧めします。ただし、デスクトップマシンの場合は、特にキャッシュ用にディスクパーティションをマウントするのは慎重に行ってください。

FS-Cache のキャッシュバックエンドで必要とされる機能に対応するファイルシステムには、次のような Red Hat Enterprise Linux 7 に実装されているファイルシステムが含まれます。

- ext3 (拡張属性が有効)
- ext4
- Btrfs
- XFS

ホストのファイルシステムはユーザー定義の拡張属性に対応する必要があります。FS-Cache はこれらの属性を使って一貫性を維持するための情報を格納します。ext3 ファイルシステム (***device***) のユーザー定義による属性を有効にするには、以下を使用します

```
# tune2fs -o user_xattr /dev/device
```

または、ファイルシステムの拡張属性をマウント時に以下のように有効にすることもできます。

```
# mount /dev/device /path/to/cache -o user_xattr
```

キャッシュバックエンドは、そのキャッシュを支えるパーティション上の特定の空き領域量を維持することで動作します。空き領域を使用する他の要素に応じてキャッシュは増大したり縮小したりして root ファイルシステム (ノートブックなど) で安全にキャッシュを使用できるようにしています。 FS-Cache ではデフォルトでこの動作が設定され、*キャッシュの間引き (cache cull) 制限* で設定を行うことができます。キャッシュの間引き制限を設定する方法については「キャッシュの間引き制限 (Cache Cull) を設定する」を参照してください。

設定ファイルの準備が整ったら **cachefilesd** デーモンを起動します。

```
# service cachefilesd start
```

起動時に **cachefilesd** が起動するよう設定するには次のコマンドを root で実行します。

```
# chkconfig cachefilesd on
```

## 9.3. NFS で Cache を使用する

NFS は明示的に指示しない限りキャッシュは使用しません。FS-Cache を使用するように NFS マウントを設定するには、**mount** コマンドに **-o fsc** オプションを組み込みます。

```
# mount nfs-share:/ /mount/point -o fsc
```

ファイルがダイレクト I/O や書き込みのために開かれていない限り、***/mount/point*** の下にあるファイル群へのアクセスはすべてキャッシュに移ります (「NFS でのキャッシュの制限」 を参照)。NFS インデックスは NFS の **ファイル名ではなくファイルハンドルを使って**コンテンツをキャッシュします。つまりハードリンクしたファイルはキャッシュを正しく共有できることになります。

NFS のバージョン 2、3、4 がキャッシュ機能に対応しています。ただし、各バージョンではキャッシングに異なるブランチを使用します。

### 9.3.1. キャッシュの共有

NFS キャッシュ共有を行う上でいくつか考慮すべき問題点があります。キャッシュは永続的であり、キャッシュ内のデータブロックは次の 4 つのキーの順序でインデックス化されます。

- レベル 1: サーバーの詳細
- レベル 2: いくつかのマウントオプション、セキュリティータイプ、FSID、識別子
- レベル 3: ファイルハンドル
- レベル 4: ファイル内のページ番号

スーパーブロック間での整合性の管理に関する問題を避けるには、データをキャッシュする NFS のスーパーブロックすべてに固有の レベル 2 キーを持たせます。通常、同じソースボリュームとオプションを持つ 2 つの NFS マウントは 1 つのスーパーブロックを共有するため、そのボリューム内に異なるディレクトリーをマウントする場合でもキャッシングを共有することになります。

> **例9.1 キャッシュの共有**
>
> 2 つの **mount** コマンドを例にあげます。
>
> ```
> mount home0:/disk0/fred /home/fred -o fsc
>
> mount home0:/disk0/jim /home/jim -o fsc
> ```
>
> **/home/fred** と **/home/jim** には同じオプションがあるので、スーパーブロックを共有する可能性が高くなります。とくに NFS サーバー上の同じボリュームやパーティションから作成されている場合は共有する可能性が高くなります (**home0**)。ここで、2 つの後続のマウントコマンドを示します。
>
> ```
> mount home0:/disk0/fred /home/fred -o fsc,rsize=230
>
> mount home0:/disk0/jim /home/jim -o fsc,rsize=231
> ```
>
> この場合、**/home/fred** と **/home/jim** は、レベル 2 の異なるネットワークアクセスパラメーターを持つため、スーパーブロックを共有しません。次のマウントコマンドも同様です。
>
> ```
> mount home0:/disk0/fred /home/fred1 -o fsc,rsize=230
>
> mount home0:/disk0/fred /home/fred2 -o fsc,rsize=231
> ```
>
> 上記の 2 つのサブツリー (**/home/fred1** と **/home/fred2**) は **2 回** キャッシュされます。
>
> スーパーブロックの共有を回避するもう 1 つの方法は **nosharecache** パラメーターで明示的に共有を避ける方法です。同じ例を示します。
>
> ```
> mount home0:/disk0/fred /home/fred -o nosharecache,fsc
>
> mount home0:/disk0/jim /home/jim -o nosharecache,fsc
> ```

この場合、レベル 2 キーの **home0:/disk0/fred** と **home0:/disk0/jim** を区別することができないため、1 つのスーパーブロックのみの使用が許可されます。これに対処するには、*固有の識別子* を少なくともどちらか 1 つのマウントに追加します (**fsc=*unique-identifier***)。

```
mount home0:/disk0/fred /home/fred -o nosharecache,fsc

mount home0:/disk0/jim /home/jim -o nosharecache,fsc=jim
```

**/home/jim** のキャッシュで使用されるレベル 2 キーに固有識別子の **jim** が追加されます。

### 9.3.2. NFS でのキャッシュの制限

ダイレクト I/O で共有ファイルシステムからファイルを開くと自動的にキャッシュが回避されます。この種のアクセスはサーバーに直接行なわれる必要があるためです。

書き込みで共有ファイルシステムからファイルを開いても NFS のバージョン 2 およびバージョン 3 では動作しません。これらのバージョンのプロトコルは、クライアントが別のクライアントからの同じファイルへの同時書き込みを検出するのに必要な整合性の管理に関する十分な情報を提供しません。

このように、ダイレクト I/O または書き込みのいずれかで共有ファイルシステムからファイルが開かれると、キャッシュされているファイルのコピーはフラッシュされます。ダイレクト I/O や書き込みの動作によってファイルが開かれなくなるまで、FS-Cache はそのファイルの再キャッシュを行いません。

また、FS-Cache の今回のリリースでは通常の NFS ファイルのみをキャッシュします。FS-Cache はディレクトリーやシンボリックリンク、デバイスファイル、FIFO、ソケットなどは **キャッシュしません**。

## 9.4. キャッシュの間引き制限 (Cache Cull) を設定する

**cachefilesd** デーモンは、共有ファイルシステムからのリモートデータをキャッシングすることで機能し、ディスク上の領域を解放します。ただし、これにより使用可能な空き領域をすべて消費してしまう可能性があり、ディスクに root パーティションも格納している場合には問題となる可能性があります。これを制御するために、**cachefilesd** は古いオブジェクト (最近のアクセスが少ないオブジェクト) をキャッシュから破棄することにより空き領域の一定量を維持します。

キャッシュの間引きは、基礎となるファイルシステムで利用可能なブロックとファイルのパーセンテージに基づいて実行されます。6 つの制限が **/etc/cachefilesd.conf** の設定で管理されます。

**brun *N*% (ブロックのパーセンテージ), frun *N*% (ファイルのパーセンテージ)**

キャッシュの空き領域と利用可能なファイルの数がこれらの制限を上回る場合、間引き動作がオフになります。

**bcull *N*% (ブロックのパーセンテージ), fcull *N*% (ファイルのパーセンテージ)**

キャッシュの空き領域と利用可能なファイルの数がこれらの制限のいずれかを下回る場合、間引き動作が開始されます。

**bstop *N*% (ブロックのパーセンテージ), fstop *N*% (ファイルのパーセンテージ)**

キャッシュの空き領域と利用可能なファイルの数がこれらの制限のいずれかを下回る場合、間引き動作によってこれらのレベルが再び制限を超えるまで、ディスク領域またはファイルの割り当ては行なわれません。

各設定の **N** のデフォルト値は以下の通りです。

- **brun**/**frun** - 10%

- **bcull**/**fcull** - 7%
- **bstop**/**fstop** - 3%

これらの設定を行う場合に、以下が正しく適用されていることを確認してください。

0 <= **bstop** < **bcull** < **brun** < 100

0 <= **fstop** < **fcull** < **frun** < 100

空き領域と利用可能なファイルのパーセンテージがそれぞれ表示されますが、**df** プログラムで表示されるような 100 % から差し引いたパーセンテージとしては表示されません。

**重要**

間引き動作は、b*xxx* と f*xxx* のペアに同時に依存します。これらを別個に処理することはできません。

## 9.5. 統計情報

FS-Cache では全般的な統計情報も追跡します。次のようにして表示させます。

```
cat /proc/fs/fscache/stats
```

FS-Cache の統計には決定ポイントとオブジェクトカウンターに関する情報が含まれます。FS-Cache で得られる統計情報の詳細については次のカーネルドキュメントを参照してください。

**/usr/share/doc/kernel-doc-*version*/Documentation/filesystems/caching/fscache.txt**

## 9.6. 参照

**cachefilesd** の詳細および設定方法については**man cachefilesd** と **man cachefilesd.conf** をご覧ください。 次のカーネルドキュメントにも **cachefilesd** に関する記載があります。

- **/usr/share/doc/cachefilesd-*version-number*/README**
- **/usr/share/man/man5/cachefilesd.conf.5.gz**
- **/usr/share/man/man8/cachefilesd.8.gz**

設計上の制約、利用できる統計情報および機能などの FS-Cache についての詳細は、次のカーネルドキュメントを参照してください。

**/usr/share/doc/kernel-doc-*version*/Documentation/filesystems/caching/fscache.txt**

# パート II. ストレージ管理

ストレージ管理のセクションは、Red Hat Enterprise Linux 7 のストレージに関する考察から始まり、パーティション、論理ボリューム管理、swap パーティションの説明が続きます。次に、ディスククォータ、RAID システムについて説明し、マウントコマンド、volume_key、acl などの機能説明がこれに続きます。その後に、SSD チューニング、書き込みバリア、I/O 制限、ディスクレスシステムについて説明し、オンラインストレージについての詳細な章の後に、Device Mapper のマルチパス化と仮想ストレージについて説明します。

以下の目次でこれらのストレージ管理タスクを確認してください。

# 第10章 ストレージをインストールする際の注意点

ストレージデバイスやファイルシステムの設定の多くはインストール時にしか実行することができません。ファイルシステムタイプなどの他の設定については、再フォーマットせずに変更できるのは特定の時点までになります。このようにストレージの設定については Red Hat Enterprise Linux 7 をインストールする前に慎重に計画する必要があります。

本章ではシステムのストレージ設定を計画する際の注意点について説明しています。実際のインストール方法については (インストール時のストレージ設定も含む)、Red Hat 提供の 『インストールガイド』 を参照してください。

Red Hat が正式にサポートするサイズおよびストレージの制限などの詳細情報については、記事の http://www.redhat.com/resourcelibrary/articles/articles-red-hat-enterprise-linux-6-technology-capabilities-and-limits を参照してください。

## 10.1. 特に注意を要する事項について

本セクションでは、ストレージの設定で特に注意を要する事項について記載しています。

### /home、 /opt、 /usr/local には別々のパーティションを用意する

将来的にシステムのアップグレードが想定される場合、**/home**、**/opt**、**/usr/local** は別々のデバイスに配置します。これによりユーザーやアプリケーションのデータを維持した状態で、オペレーティングシステムを含むデバイスまたはファイルシステムの再フォーマットが可能になります。

### DASD デバイスと zFCP デバイス - IBM System Z

IBM System Z のプラットフォームでは、DASD デバイスと zFCP デバイスは *Channel Command Word* (CCW) メカニズムで設定されます。CCW のパスをシステムに明示的に追加してからオンラインにする必要があります。DASD デバイスの場合、起動コマンドラインか、または CMS 設定ファイル内で **DASD=** パラメーターにデバイス番号 (またはデバイス番号の範囲) を記載します。

zFCP デバイスの場合は、デバイス番号、*論理ユニット番号* (LUN)、 *ワールドワイドポート名* (WWPN) を記載する必要があります。zFCP が初期化されると CCW パスにマッピングが行われます。起動コマンドライン (または CMS 設定ファイル内) の **FCP_x=** の行を使用して、インストーラーに対してこの情報を指定することができます。

### LUKS を使用してブロックデバイスを暗号化する

LUKS/**dm-crypt** を使ってブロックデバイスを暗号化するとデバイス上に存在しているフォーマットがすべて破棄されます。このため、まず暗号化するデバイス (ある場合) を選択してください。次に、新しいシステムのストレージ設定をインストールプロセスの一部としてアクティブにします。

### 古い BIOS RAID メタデータ

ファームウェア RAID 用に設定したシステムのディスクに残っている RAID メタデータを **削除しないまま** でそのディスクをシステムから移動すると、**Anaconda** がディスクを正常に検出できなくなる場合があります。

**警告**

ディスクから RAID メタデータを削除または消去すると、保存データがすべて破棄される可能性があります。そのため、これを実行する前に必ずバックアップを取っておくことをお勧めします。

ディスクの RAID メタデータを削除するには、次のコマンドを使用します。

```
dmraid -r -E /device/
```

RAID デバイスの管理については **man dmraid** および 18章*RAID (Redundant Array of Independent Disks)* を参照してください。

## iSCSI の検出と設定

iSCSI ドライブのプラグアンドプレイ検出の場合には、iBFT 起動が可能な *ネットワークインターフェースカード* (NIC) のファームウェアで設定を行ってください。インストール時の iSCSI ドライブの CHAP 認証がサポートされています。ただし、インストール時の iSNS 検出はサポートされていません。

## FCoE の検出と設定

*イーサネット経由のファイバーチャネル* (FCoE) のプラグアンドプレイ検出の場合には、EDD 起動が可能な NIC のファームウェアで設定を行ってください。

## DASD

インストール中に *ダイレクトアクセスストレージデバイス*(DASD) を追加したり設定したりすることはできません。このデバイスは CMS 設定ファイル内で指定してください。

## DIF/DIX を有効にしているブロックデバイス

DIF/DIX は、特定の SCSI ホストバスのアダプターおよびブロックデバイスで提供されているハードウェアチェックサムの機能です。DIF/DIX が有効になっていると、ブロックデバイスを汎用目的で使用した場合にエラーが発生します。DIF/DIX チェックサムの計算後はバッファーされたデータが上書きされないようにするためのインターロックがバッファー書き込みパス内にないため、バッファーされた入出力または **mmap(2)** ベースの入出力が正常に動作しなくなります。

これにより入出力がチェックサムのエラーで後で失敗することになります。すべてのブロックデバイス (またはファイルシステムベース) のバッファーされた入出力または **mmap(2)** 入出力に対する共通の問題となるため、上書きによるこれらのエラーを回避することはできません。

このため、DIF/DIX を有効にしているブロックデバイスは **O_DIRECT** を使用するアプリケーションでのみ使用するようにしてください。こうしたアプリケーションはローブロックデバイスを使用するはずです。また、XFS ファイルシステムを通して発行されるのが **O_DIRECT** 入出力のみである限り、DIF/DIX が有効になっているブロックデバイスで XFS ファイルシステムを使用しても安全です。特定の割り当て動作を行う際にバッファーされた入出力にフォールバックを行わないファイルシステムは XFS のみです。

DIF/DIX チェックサムの計算後に入出力データが変更されないようにするのは常にアプリケーションの役目となるため、DIF/DIX を使用できるアプリケーションを **O_DIRECT** 入出力および DIF/DIX ハードウェアでの使用を目的として設計されたアプリケーションに限ってください。

# 第11章 ファイルシステムのチェック

ファイルシステムについては、その整合性をチェックでき、オプションでファイルシステム固有のユーザースペースのツールを使って修復を実行することができます。これらのツールは、通常 **fsck** ツールと呼ばれます。この **fsck** は、*file system check* の省略版です。

**注記**

これらのファイルシステムのチェックは、ファイルシステム全体でのメタデータの整合性のみを保証します。これらは、ファイルシステムに含まれる実際のデータを認識しないため、データリカバリーツールではありません。

ファイルシステムの不整合はさまざまな理由によって生じる可能性があります。これらの理由には、ハードウェアのエラー、ストレージ管理のエラー、およびソフトウェアのバグを含みますが、これらに限定されません。

最新のメタデータジャーナリングファイルシステムが一般的になる前に、ファイルシステムのチェックは、システムがクラッシュしたり、電源が切れたりするたびに必要となっていました。これは、ファイルシステムの更新が中断し、不整合な状態が生じる可能性があったためです。結果として、ファイルシステムのチェックは、ブート時に、**/etc/fstab** にリストされた各ファイルシステムで行なわれてきました。ジャーナリングファイルシステムの場合、通常これは非常に短い操作で実行できます。ファイルシステムのメタデータジャーナリングにより、クラッシュが発生した後でも整合性が確保されるためです。

ただし、ジャーナリングファイルシステムの場合であっても、ファイルシステムの不整合や破損が生じることがあります。いったんこれが生じると、ファイルシステムのチェッカーを使用してファイルシステムを修復する必要があります。以下は、この手順を実行する際のベストプラクティスとその他の役立つ情報です。

## 11.1. fsck のベストプラクティス

一般的に、ファイルシステムのチェックおよび修復ツールを実行することにより、チェックによって見つかる不整合の一部を自動的に修復できることが予想されます。場合によっては、重度にダメージを受けた inode やディレクトリーは、修復できない場合に破棄されることがあります。ファイルシステムへの大幅な変更が発生する可能性があります。予想外の、または好ましくない変更が永続的に行なわれないようにするには、以下の予防的な手順を実行します。

**Dry run**

ファイルシステムのほとんどのチェッカーには、チェックを行うものの、ファイルシステムの修復は行なわない操作モードがあります。このモードでは、チェッカーは、発見したエラーと実行した可能性のあるアクションを出力しますが、ファイルシステムを実際に変更することはありません。

**注記**

整合性チェックの後のフェーズでは、修復モードで実行されていた場合に前のフェーズで修正されていた可能性のある不整合を発見し、追加のエラーを出力する可能性があります。

**ファイルシステムのイメージ上での初回操作**

ほとんどのファイルシステムは、メタデータのみを含むスパースコピーである *メタデータイメージ* の作成に対応しています。ファイルシステムのチェッカーは、メタデータ上でのみ動作するため、このようなイメージを使用して、実際のファイルシステムの修復の Dry Run を実行し、実際に加えられた可能性のある変更を評価することができます。変更が受け入れ可能なものである場合は、修復はファイルシステム自体で実行できます。

**注記**

ファイルシステムが大幅に損傷している場合、メタデータイメージの作成に関連して問題が発生する可能性があります。

**サポート調査のためのファイルシステムイメージの保存**

修復前のファイルシステムのメタデータイメージは、破損の原因がソフトウェアのバグの可能性がある場合のサポート調査を行う上で役に立つことがあります。修復前のイメージに見つかる破損のパターンは、根本原因の分析に役立つことがあります。

**アンマウントされたファイルシステム上のみの操作**

ファイルシステムの修復は、アンマウントされたファイルシステムのみで実行する必要があります。ツールには、ファイルシステムへの単独アクセスが必要であり、それがないと追加の損傷が発生する可能性があります。一部のファイルシステムはマウントされているファイルシステムでチェックのみのモードのみをサポートしますが、ほとんどのファイルシステムツールは、修復モードでこの要件を実行します。チェックのみのモードがマウントされているファイルシステム上で実行されている場合、アンマウントされていないファイルシステム上で実行される場合には見つからない正しくないエラーを見つける可能性があります。

**ディスクエラー**

ファイルシステムのチェックツールは、ハードウェアの問題を修復することはできません。修復を正常に機能させるには、ファイルシステムは完全に読み取り可能かつ書き込み可能である必要があります。ファイルシステムがハードウェアのエラーによって破損する場合、まずファイルシステムを **dd(8)** ユーティリティーなどを使って、良好なディスクに移行する必要があります。

## 11.2. fsck に関するファイルシステム固有の情報

### 11.2.1. ext2、ext3、および ext4

上記のすべてのファイルシステムは、ファイルシステムのチェックと修復を実行するために **e2fsck** バイナリーを使用します。ファイル名の **fsck.ext2**、**fsck.ext3**、および **fsck.ext4** は、この同じバイナリーのハードリンクです。これらのバイナリーはブート時に自動的に実行され、それらの動作はチェックされるファイルシステムやファイルシステムの状態に基づいて異なります。

完全なファイルシステムのチェックおよび修復が、メタデータジャーナリングファイルシステムではない ext2 や、ジャーナルのない ext4 ファイルシステムについて実行されます。

メタデータジャーナリングのある ext3 と ext4 ファイルシステムの場合、ジャーナルはユーザースペースで再生され、バイナリーが終了します。これは、ジャーナルの再生によりクラッシュ後のファイルシステムの整合性が確保するため、デフォルトのアクションになります。

これらのファイルシステムで、マウント中にメタデータの不整合が生じると、それらはその事実をファイルシステムのスーパーブロックに記録します。ファイルシステムにこのようなエラーのマークが付けられていることを **e2fsck** が発見すると、**e2fsck** はジャーナルの再生後にフルチェックを実行します (ある場合)。

**e2fsck** は、**-p** オプションが指定されていない場合、実行時にユーザー入力を求める場合があります。**-p** オプションは **e2fsck** に対して、安全に実行される可能性のあるすべての修復を自動的に実行するように指示します。ユーザーの介入が必要な場合、**e2fsck** はその出力の未解決の問題を示し、この状態を出口コードに反映させます。

共通して使用される **e2fsck** の実行時のオプションには以下が含まれます。

**-n**

非変更モードです。チェックのみの操作です。

**-b スーパーブロック**

プライマリーブロックが損傷している場合、別のスーパーブロックのブロック番号を指定します。

**-f**

スーパーブロックに記録されたエラーがない場合に、フルチェックを強制実行します。

**-j ジャーナルデバイス**

外部のジャーナルデバイス (ある場合) を指定します。

**-p**

ユーザー入力のないファイルシステムを自動的に修復または「preen (修復)」する

**-y**

すべての質問に「yes」の回答を想定する

**e2fsck** のすべてのオプションが**e2fsck(8)** man ページで指定されています。

以下の 5 つの基本フェーズが、実行中に **e2fsck** で実行されます。

1. Inode、ブロック、およびサイズのチェック。
2. ディレクトリー構造のチェック。
3. ディレクトリー接続のチェック。
4. 参照数のチェック。
5. グループサマリー情報のチェック。

**e2image(8)** ユーティリティーは、診断またはテスト目的で、修復前のメタデータイメージを作成するために使用できます。**-r** オプションは、テスト目的で、ファイルシステム自体と同じサイズのスパースファイルを作成するために使用する必要があります。その後 **e2fsck** は、結果として作成されるファイルで直接動作できます。イメージが診断目的でアーカイブまたは提供される場合に、**-Q** オプションを指定する必要があります。これにより、送信に適したよりコンパクトなファイル形式が作成されます。

### 11.2.2. XFS

ブート時に修復は自動的に行なわれません。ファイルシステムのチェックまたは修復を開始するには、**xfs_repair** ツールが使用されます。

### 注記

**fsck.xfs** バイナリーは *xfsprogs* パッケージにありますが、これは、ブート時に **fsck.*filesystem*** バイナリーを検索する initscript に対応するためにのみ存在します。**fsck.xfs** は、出口コードとして 0 を設定して終了します。

留意すべきもう 1 つの点として、古い *xfsprogs* パッケージには**xfs_check** ツールが含まれます。このツールは非常にスピードが遅く、大きなファイルシステムに対して十分な拡張性がありません。そのため、**xfs_repair -n** が優先的に選択され、これは非推奨になっています。

ファイルシステム上のクリーンログは **xfs_repair** が動作するために必要です。ファイルシステムがクリーンな状態でアンマウントされなかった場合、**xfs_repair** を使用する前に、マウントとアンマウントが実行される必要があります。ログが破損していて再生できない場合、**-L** オプションを使用してログをゼロ化できます。

### 重要

**-L** オプションは、ログを再生できない場合にのみ使用する必要があります。このオプションは、ログ内のすべてのメタデータの更新を破棄し、結果としてさらに不整合を生じさせます。

**-n** オプションを使用して、Dry Run で、チェックのみモードの**xfs_repair** を実行することができます。このオプションが指定されると、ファイルシステムに一切の変更は行なわれません。

**xfs_repair** で使用できるオプションは非常に限られています。共通に使用されるオプションには以下が含まれます。

**-n**

変更不可モードです。チェックのみの操作です。

**-L**

メタデータログがゼロになります。マウントによってログを再生できない場合にのみ使用します。

**-m maxmem**

最大 MB の実行時に使用されるメモリーを制限します。必要な最小メモリーの概算を出すために 0 を指定できます。

**-l logdev**

外部ログデバイス (ある場合) を指定します。

**xfs_repair** のすべてのオプションが**xfs_repair(8)** man ページで指定されています。

以下の 8 つの基本フェーズが、実行中に **xfs_repair** によって実施されます。

1. Inode および inode ブロックマップ (アドレス指定) のチェック。
2. Inode 割り当てマップのチェック。
3. Inode サイズのチェック。
4. ディレクトリーのチェック。

5. パス名のチェック。
6. リンク数のチェック。
7. フリーマップのチェック。
8. スーパーブロックのチェック。

これらのフェーズについては、操作時に出力されるメッセージと共に、**xfs_repair(8)** man ページに詳細にわたって説明されています。

**xfs_repair** はインタラクティブな操作ではありません。すべての操作は、ユーザーの入力なしに自動的に実行されます。

診断またはテスト目的で、修復前のメタデータイメージを作成する必要がある場合は、**xfs_metadump(8)** および **xfs_mdrestore(8)** ユーティリティーを使用することができます。

### 11.2.3. Btrfs

**btrfsck** ツールは btrfs ファイルシステムをチェックし、修復するために使用されます。このツールはまだ開発の初期段階にあり、ファイルシステムの破損のすべてのタイプを検出または修復できない可能性があります。

デフォルトで、**btrfsck** はファイルシステムを変更しません。つまり、デフォルトでチェックのみモードを実行します。修復が必要な場合は、**--repair** オプションを指定する必要があります。

以下の 3 つの基本フェーズを、実行中に **btrfsck** によって実行します。

1. エクステントのチェック。
2. ファイルシステムの root チェック。
3. ルートの参照数のチェック。

**btrfs-image(8)** ユーティリティーを使用して、診断またはテストの目的で、修復前のメタデータイメージを作成することができます。

# 第12章 パーティション

ユーティリティー **parted** の使用によりユーザーは以下を実行できます。

- 既存パーティションテーブルの表示
- 既存パーティションのサイズ変更
- 空き領域または追加のハードドライブからのパーティションの追加

**parted** パッケージは、Red Hat Enterprise Linux のインストール時にデフォルトで収納されています。**parted** を開始するには、root としてログインして、シェルプロンプトでコマンド**parted** ***/dev/sda*** を入力します (ここで***/dev/sda*** は設定するドライブのデバイス名です)。

パーティションのあるデバイスが使用中の場合は、そのパーティションの削除やサイズ変更を実行することができません。使用中のデバイスで新規パーティションを作成することはできますが、これは推奨されません。

使用が終了する予定のデバイスでは、そのデバイス上のパーティションのいずれもマウントすべきではなく、そのデバイスで swap 領域を有効にしないでください。

さらに、使用中の場合はパーティションテーブルを修正すべきではありません。カーネルが変更を正しく認識できない可能性があるためです。パーティションテーブルがマウント済みのパーティションの実際の状態と一致しないと、情報は誤ったパーティションに書き込まれる可能性があり、データの消失および上書きが発生する可能性があります。

これを実行する最も簡単な方法は、システムをレスキューモードで起動することです。ファイルシステムのマウントを求めるプロンプトが出された時点で **Skip** を選択します。

または、デバイスに使用中のパーティション (ファイルシステムを使用するか、またはアンマウント状態からロックするシステムのプロセス) が含まれない場合は、**umount** コマンドを使用してそれらのパーティションをアンマウントしてから、**swapoff** コマンドを使用してハードドライブ上のすべての swap 領域をオフにします。

表12.1「**parted** コマンド」には、一般的に使用される **parted** コマンドの一覧が含まれています。次に続くセクションでは、これらのコマンドおよび引数の一部をより詳しく説明します。

**表12.1 parted コマンド**

| コマンド | 詳細 |
|---|---|
| **check** ***minor-num*** | ファイルシステムに対して簡単なチェックを実行します。 |
| **cp** ***from to*** | ファイルシステムを１つのパーティションから別のパーティションにコピーします。*from* と *to* にはパーティションのマイナー番号が入ります。 |
| **help** | 利用可能なコマンドの一覧を表示します |
| **mklabel** ***label*** | パーティションテーブル用のディスクラベルを作成します |
| **mkfs** ***minor-num file-system-type*** | タイプ *file-system-type* のファイルシステムを作成します。 |
| **mkpart** ***part-type fs-type start-mb end-mb*** | 新しいファイルシステムを作成せずに、パーティションを作成します |
| **mkpartfs** ***part-type fs-type start-mb end-mb*** | パーティションを作成し、かつ指定されたファイルシステムを作成します |
| **move** ***minor-num start-mb end-mb*** | パーティションを移動します |

| コマンド | 詳細 |
| --- | --- |
| **name *minor-num name*** | Mac と PC98 のディスクラベル用のみのパーティションに名前を付けます |
| **print** | パーティションテーブルを表示します |
| **quit** | **parted** を終了します |
| **rescue** *start-mb end-mb* | *start-mb* から *end-mb* へ、消失したパーティションを復旧します。 |
| **resize *minor-num start-mb end-mb*** | *start-mb* から *end-mb* へパーティションのサイズを変更します |
| **rm *minor-num*** | パーティションを削除します |
| **select *device*** | 設定する別のデバイスを選択します |
| **set *minor-num flag state*** | パーティションにフラグを設定します。*state* はオンまたはオフのいずれかになります |
| **toggle [*NUMBER* [*FLAG*]** | パーティション *NUMBER* 上の *FLAG* の状態を切り替えます |
| **unit *UNIT*** | デフォルトのユニットを *UNIT* に設定します |

## 12.1. パーティションテーブルの表示

**parted** を開始した後に、コマンド **print** を使用して、パーティションテーブルを表示します。以下のようなテーブルが表示されます。

**例12.1 パーティションテーブル**

```
Model: ATA ST3160812AS (scsi)
Disk /dev/sda: 160GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      32.3kB  107MB   107MB   primary   ext3         boot
 2      107MB   105GB   105GB   primary   ext3
 3      105GB   107GB   2147MB  primary   linux-swap
 4      107GB   160GB   52.9GB  extended       root
 5      107GB   133GB   26.2GB  logical   ext3
 6      133GB   133GB   107MB   logical   ext3
 7      133GB   160GB   26.6GB  logical               lvm
```

最初の行には、ディスクタイプ、製造元、モデル番号とインターフェースが含まれ、2 行目にはディスクラベルのタイプが表示されます。4 行目以下の残りの出力でパーティションテーブルが表示されます。

パーティションテーブルでは、パーティション **number** が マイナー番号を示します。たとえば、マイナー番号が 1 のパーティションは **/dev/sda1** に相当します。**Start** と **End** の値は、メガバイト単位の値になります。有効な **Type** として、metadata(メタデータ)、free(空き領域)、primary(プライマリー)、extended(拡張)、logical(論理) を使用できます。**Filesystem** は、ファイルシステムのタイプであり、以下のいずれかになります。

- ext2
- ext3
- fat16

- fat32
- hfs
- jfs
- linux-swap
- ntfs
- reiserfs
- hp-ufs
- sun-ufs
- xfs

デバイスの **Filesystem** が値を表示していない場合、そのファイルシステムが不明なファイルシステムであることを示します。

**Flags** 列には、パーティションのフラグセットが一覧表示されます。利用可能なフラグには、boot、root、swap、hidden、raid、lvm または lba が含まれます。

**注記**

**parted** を再開始せずに、異なるデバイスを選択するには、**select** コマンドとその後にデバイス名 (例: **/dev/sda**) を使用します。そうすることで、そのデバイスのパーティションテーブルを表示したり、設定したりできるようになります。

## 12.2. パーティションの作成

**警告**

使用中のデバイスでパーティションを作成しないようにしてください。

**手順12.1 パーティションの作成**

1. パーティションを作成する前に、レスキューモードで起動します (または、デバイス上のパーティションをアンマウントして、デバイス上の swap 領域をすべてオフにします)。
2. **parted** を開始します。ここで、**/dev/*sda*** は、パーティションの作成先となるデバイスです。

```
# parted /dev/sda
```

3. 十分な空き領域があるかどうか判別するために現在のパーティションテーブルを表示します。

```
# print
```

十分な空き領域がない場合、現在のパーティションのサイズを変更することができます。詳細については「パーティションのサイズ変更」を参照してください。

### 12.2.1. パーティション作成

パーティションテーブルから、新しいパーティションの開始点と終了点、およびパーティションのタイプを決定します。プライマリーパーティションは 1 つのデバイス上に 4 つまで保有できます (拡張パーティションは含まない)。5 つ以上のパーティションが必要な場合は、3 つのプライマリーと 1 つの拡張にして、拡張の中に複数の論理パーティションを含めることができます。ディスクパーティションの概要については、Red Hat Enterprise Linux 7 『インスト－ルガイド』 内の付録 『ディスクパーティションの概要』 を参照してください。

たとえば、ハードドライブの 1024 メガバイトから 2048 メガバイトまでに ext3 ファイルシステムを持つプライマリーパーティションを作成するには、以下のコマンドを入力します。

```
# mkpart primary ext3 1024 2048
```

**注記**

**mkpartfs** コマンドを代わりに使用する場合、パーティションが作成された後にファイルシステムが作成されます。しかし、**parted** は ext3 ファイルシステムの作成をサポートしないため、ext3 ファイルシステムを作成する場合には、**mkpart** を使用してから、後述のように**mkfs** コマンドを使ってファイルシステムを作成します。

変更は **Enter** を押した時点に反映されます。そのため、それを実行する前に再確認する必要があります。

パーティションを作成した後には、**print** コマンドを使用して、それが正しいパーティションタイプ、ファイルシステムタイプ、およびサイズが設定された状態でパーティションテーブル内にあることを確認します。また、新規パーティションのマイナー番号を確認し、その上のファイルシステムにラベル付けできるようにしてください。また、parted を終了した後に **cat /proc/partitions** の出力を表示して、カーネルが新規のパーティションを認識することを確認します。

parted では、最大 128 のパーティションが作成されます。*GPT (GUID Partition Table)* の仕様により、パーティションテーブル用に確保するエリアを拡大することでさらに多くのパーティションを作成することができますが、parted で用いられる一般的な方法では、128 パーティション用として十分なエリアに制限されます。

### 12.2.2. パーティションのフォーマットとラベル付け

パーティションをフォーマットしてラベルを付けるには、以下の手順を使用します。

**手順12.2 パーティションのフォーマットとラベル付け**

1. パーティションにはまだファイルシステムがありません。ファイルシステムを作成するには、以下のコマンドを使用します。

   ```
   # /sbin/mkfs -t ext3 /dev/sda6
   ```

**警告**

パーティションをフォーマットすると、そのパーティションに存在する現行のデータがすべて永久に抹消されます。

2. 次に、パーティション上のファイルシステムにラベルを付与します。たとえば、新規パーティションのファイルシステムが **/dev/sda6** であり、それに **/work** のラベルを付ける場合は、以下を使用します。

```
# e2label /dev/sda6 /work
```

デフォルトでは、インストールプログラムはパーティションのマウントポイントをラベルとして使用して、ラベルが固有なものとなるようにします。ユーザーは使用するラベルを選択できます。

その後に、root としてマウントポイント (例、**/work**) を作成します

### 12.2.3. /etc/fstab に追加します

root として、パーティションの UUID を使用して新規のパーティションを含めるように **/etc/fstab** ファイルを編集します。パーティションの UUID の完全な一覧を確認するには、コマンド **blkid -o list** を使用し、個別のデバイスの詳細については**blkid device** を使用します。

最初の列には、**UUID=** が含まれ、その後にファイルシステムの UUID が続きます。2 つ目の列には、新規パーティションのマウントポイントが含まれる必要があり、その次の列にはファイルシステムのタイプ (ext3 または swap など) が含まれる必要があります。フォーマットについてさらに詳しくは、コマンド **man fstab** を使用してその man ページをご覧ください。

4 つ目の列に **defaults** の語が入る場合は、パーティションは起動時にマウントされます。再起動を必要とせずにパーティションをマウントするには、root として以下のコマンドを入力します。

```
mount /work
```

## 12.3. パーティションの削除

**警告**

使用中のデバイス上にあるパーティションは削除しないようにしてください。

**手順12.3 パーティションの削除**

1. パーティションを削除する前に、レスキューモードで起動します (または、デバイス上のパーティションのすべてをアンマウントして、そのデバイスの swap 領域をすべてオフにします)。
2. **parted** を開始します。ここで、**/dev/*sda*** はパーティションの削除元となるデバイスです。

```
# parted /dev/sda
```

3. 現在のパーティションテーブルを表示して、削除するパーティションのマイナー番号を判別します。

```
# print
```

4. コマンド **rm** を使用してパーティションを削除します。たとえば、マイナー番号が 3 のパーティションを削除するには、以下を実行します。

```
# rm 3
```

変更は **Enter** を押すとすぐに反映されます。そのため、これを押す前にコマンドを再確認してください。

5. パーティションを削除した後には、**print** コマンドを使用してそれがパーティションテーブルから除かれていることを確認します。**/proc/partitions** の出力も表示して、パーティションが削除されたことをカーネルが認識していることを確認します。

```
# cat /proc/partitions
```

6. 最後のステップは、パーティションを **/etc/fstab** ファイルから削除することです。削除されているパーティションを宣言している行を見つけ、その行をファイルから削除します。

## 12.4. パーティションのサイズ変更

**警告**

使用中のデバイスにあるパーティションのサイズを変更しないようにしてください。

**手順12.4 パーティションのサイズ変更**

1. パーティションのサイズを変更をする前に、レスキューモードで起動します (または、デバイス上のすべてのパーティションをアンマウントして、デバイス上のすべての swap 領域をオフにします)。
2. **parted** を開始します。ここで、**/dev/*sda*** は、パーティションのサイズを変更するデバイスです。

```
# parted /dev/sda
```

3. 現在のパーティションテーブルを表示して、サイズ変更するパーティションのマイナー番号と、そのパーティションの開始点および終了点を判別します。

```
# print
```

4. パーティションのサイズを変更するには、**resize** コマンドを、その後にパーティションのマイナー番号、メガバイトで開始の場所、およびメガバイトで終了の場所を付けて使用します。

**例12.2 パーティションのサイズ変更**

例:

```
resize 3 1024 2048
```

**警告**

パーティションは、デバイス上で利用可能な領域よりも大きくすることはできません。

5. パーティションのサイズを変更した後には、**print** コマンドを使用して、パーティションが正しくサイズ変更されたこと、またパーティションタイプが正しいこと、およびファイルシステムタイプが正しいことを確認します。

6. 通常モードでシステムを再起動した後に、コマンド **df** を使用してパーティションがマウントされていることや、新しいサイズで認識されていることを確認します。

# 第13章 LVM (論理ボリュームマネージャー)

LVM/LVM2 について扱う本章では LVM の GUI 管理ツール (**system-config-lvm**) の使い方に焦点を当てます。クラスター化したストレージとクラスター化していないストレージでの LVM パーティションの作成と設定についての詳細については、Red Hat が提供する 『論理ボリュームマネージャの管理』 ガイドを参照してください。

さらに、Red Hat Enterprise Linux 7 の 『インストールガイド』 では、インストール時に LVM 論理ボリュームを作成し、これを設定する方法についても説明しています。

Red Hat Enterprise Linux 7 では、md RAID は LVM のターゲットとして利用できます。これは、md RAID 4、5、6、10、書き込み予定のビットマップ (write intent bitmap)、オンラインの再シェーピング (on-line reshaping)、およびライトビハインド (write behind) 機能がすべて利用可能になったことを意味します。さらに詳しくは、Red Hat の 『論理ボリュームマネージャの管理』 を参照してください。

LVM とは、ディスクの割り当て、ストライピング、ミラーリングおよび論理ボリュームのサイズ変更など、論理ボリュームの管理を行うツールです。

LVM を使って 1 つのハードドライブまたは複数のハードドライブのセットを 1 つまたは複数の *物理ボリューム* に割り当てます。

**/boot/** パーティションを除き、物理ボリュームは組み合わせて *論理ボリューム* にすることができます。ブートローダーが論理ボリュームグループを読み込むことができないため、**/boot/** パーティションを論理ボリュームグループ上に配置することはできません。root (**/**) パーティションが論理ボリュームにある場合は、**/boot/** パーティションをボリュームグループ以外の場所に別途作成するようにしてください。

1 つの物理ボリュームを複数のドライブにまたがせることはできないため、複数のドライブにまたがせるには、LVM を使用してドライブごとに 1 つまたは複数の物理ボリュームを作成します。

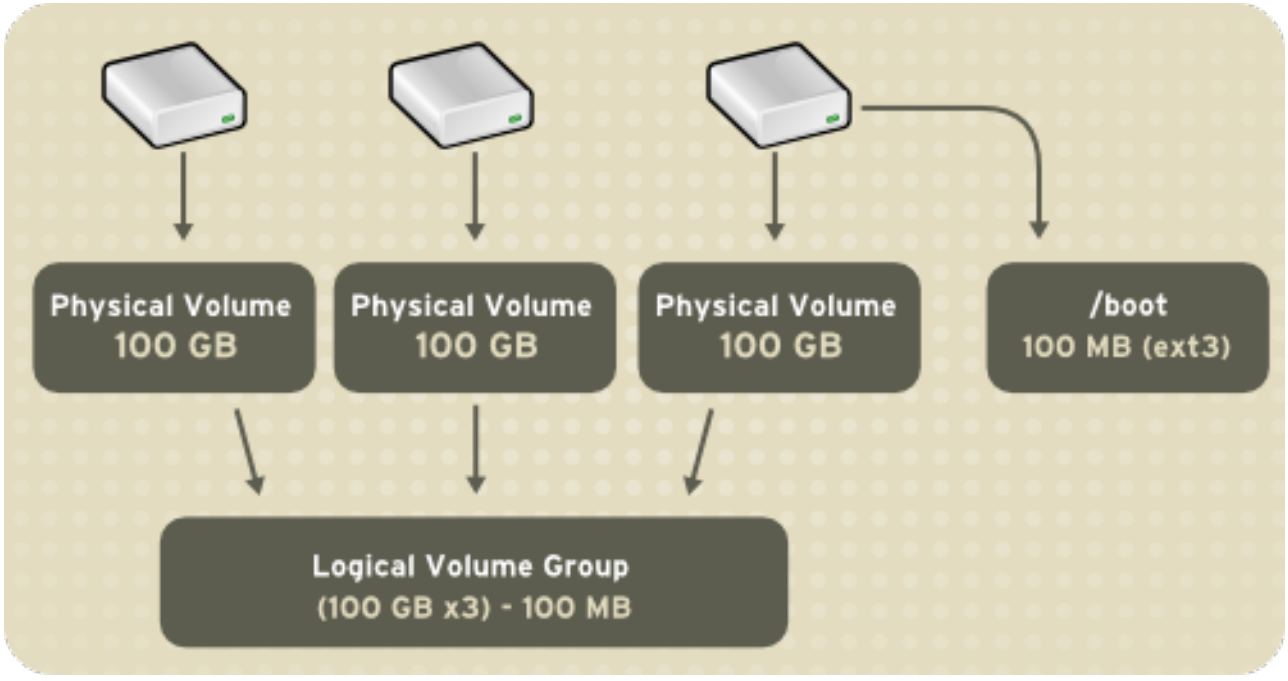



**図13.1 論理ボリューム**

ボリュームグループは、**/home** や **/** などのマウントポイント、および ext2 や ext3 などのファイルシステムのタイプが割り当てられる複数の *論理ボリューム* に区分けできます。「パーティション」の容量が一杯になった場合、ボリュームグループの空き領域を論理ボリュームに追加してパーティションのサイズを大きくすることができます。新しいハードドライブをシステムに追加する場合、ボリュームグループに追加することで論理ボリュームとなるパーティションのサイズを拡大できます。

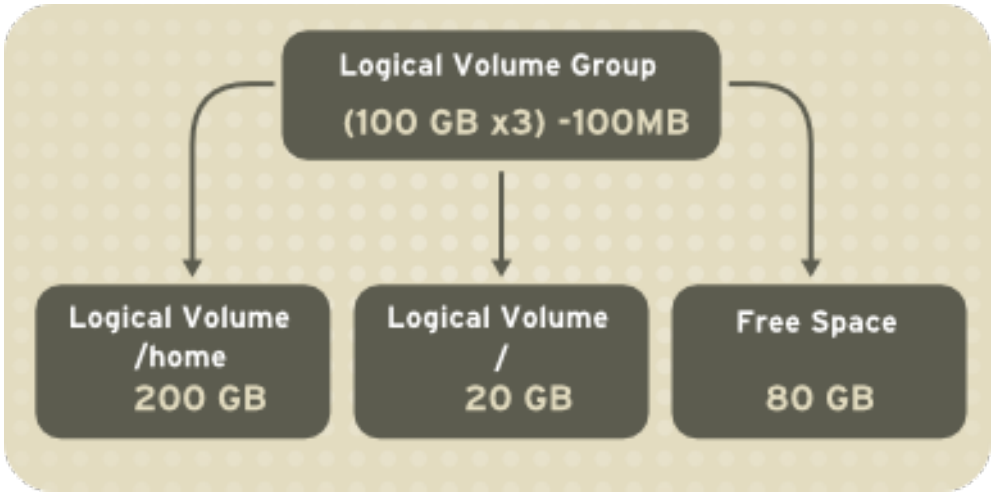



**図13.2 論理ボリューム**

一方、ext3 ファイルシステムでシステムのパーティションを設定している場合、ハードドライブは指定サイズの複数パーティションに区分けされます。1 つのパーティションが一杯になってからは、そのパーティションのサイズを拡大するのは簡単ではありません。パーティションを別のハードドライブに移動させたとしても、元のハードドライブ領域を別のパーティションに再度割り当てるか、またはその領域を全く使用しないかのオプションを選択しなければなりません。

## 13.1. LVM2 とは

Red Hat Enterprise Linux 5 のデフォルトは LVM バージョン 2 (LVM2) です。LVM2 は 2.6 カーネルに含まれるデバイスマッパードライバーを使用します。LVM2 は、2.4 カーネルを実行する Red Hat Enterprise Linux バージョンからのアップグレードが可能です。

## 13.2. 参照

LVM の詳細については次の情報ソースをご覧ください。

### インストールされているドキュメント

- **`rpm -qd lvm2`** — このコマンドは man ページなどの**`lvm`** パッケージ内にあるすべてのドキュメントを表示します。
- **`lvm help`** — このコマンドは使用できるすべての LVM コマンドを表示します。

### 役に立つ Web サイト

- http://sources.redhat.com/lvm2 — LVM2 の Web ページです。LVM 2 の概要、メーリングリストのリンク、およびその他の役に立つ情報が掲載されています。
- http://tldp.org/HOWTO/LVM-HOWTO/ — Linux Documentation Project が提供している LVM HOWTO (LVM 入門書)です。

# 第14章 Snapper

Red Hat Enterprise Linux 7 では、Snapper ユーティリティーを導入しています。Snapper には以下の機能が付属しています。

- マウントされた設定済みの Snapper ボリュームから読み取り専用のスナップショットを作成することは、未処理 (raw) の LVM2 コマンドまたは btrfs-progs コマンドを使用する場合よりも容易です。
- スナップショットを削除する機能です。
- Snapper 設定ですべてのスナップショットを一覧表示する機能です。
- Snapper は、btrfs ボリュームと LVM2 のシンプロビジョニングされたボリュームについて特定のファイルシステムツールと同じ機能を提供します。
- **status** コマンドは、各種スナップショット間の Snapper ボリュームにおけるすべての変更点のよく整理された概要を提供します。**status** コマンドの出力についての説明は、「Snapper **status** コマンド」を参照してください。
- **undochange** コマンドは、システム内の変更を元に戻すための限定的ながら、強力な手段になります。これは、設定の変更を元に戻すのに役立ち、別々の Snapper ボリュームをセットアップする際に最もよく機能します (たとえば、**/home**、**/usr**、**/usr/var/log** など)。
- **diff** コマンドは、ステータスレポートに基づいて、変更されたコンテンツを含むファイルの差分を出力します。
- **xadiff** コマンドは**diff** コマンドと同様の形態の変更された拡張属性のリストを提供します。さらに、このコマンドはステータスレポートの出力でも適用されます。

## 14.1. 初期の Snapper セットアップ

以下の例は、Snapper の **/home** をセットアップする方法を示しています。

**例14.1 初期の Snapper セットアップの例**

**/home** ディレクトリーを Snapper 設定に含める必要があります。これを実行するには、以下を true にする必要があります。

1. **/home** は以下のいずれかに置かれている必要があります。
   - サポートされるファイルシステムがその上部に設定された別個の論理ボリューム、または
   - Btrfs サブボリューム。
2. ファイルシステムをマウントする必要があります。
3. LVM:

```
# snapper -c home_volume create-config -f "lvm(xfs)" /home
```

Btrfs:

```
# snapper -c home_volume create-config -f btrfs /home
```

## 14.2. Snapper スナップショットの作成

## 14.2. Snapper スナップショットの作成

スナップショットの作成は、LVM と Btrfs で同じように機能します。**--type** または **-t** オプションが省略される場合、Snapper スナップショットの単一タイプが作成されます。

**例14.2 Snapper スナップショットサンプルの作成**

新規の単一 Snapper のスナップショットを、*lvm_with_xfs* という LVM Snapper ボリュームに作成するには、以下のコマンドを使用します。

```
# snapper -c lvm_with_xfs create -t pre -p
```

これにより、6 の出力が生成されることがあります。

ポストスナップショットをその親のプリスナップショットに関連付けて作成するには、以下のコマンドを使用します。

```
# snapper -c lvm_with_xfs create -t post --pre-num 6 -p
```

これにより、上記のプリスナップショットのコマンドと合わせて使用すると、7 の出力が生成されます。

これで、6 および 7 のスナップショットがプリ/ポストのペアとなり、Snapper ボリュームが正しく設定されていると、バックグラウンド比較が Snapper デーモンによって作成されます。

## 14.3. Snapper status コマンド

以下のセクションは、Snapper **status** コマンドの出力について説明しています。

**例14.3 Snapper status コマンドの例**

以下の例では、*home_vol* という単一 Snapper ボリュームで **status** コマンドを実行し、スナップショット 1 と 2 を比較します。

```
snapper -c home_vol 1..2
tp.... /home/user/dir1
-..... /home/user/dir1/file_a
c.ug.. /home/user/file2
+..... /home/user/file3
....x. /home/user/file4
cp..xa /home/user/file5
```

出力の最初の部分の文字と点は、列として読み込むことができます。

```
+..... /home/user/example_file
||||||
123456
```

**列 1**

これは、ファイルの (ディレクトリーエントリー) タイプについての変更を示しています。使用できる値は以下の通りです。

**.**

何も変更されていません。

**+**

ファイルが作成されました。

**-**

ファイルが削除されました。

**c**

コンテンツが変更されました。

**t**

ディレクトリーエントリーのタイプが変更されました (たとえば、以前のシンボリックリンクは同じファイル名を持つ通常ファイルに変更されました)。

**列 2**

これは、ファイルの権限における変更を示しています。使用できる値は以下の通りです。

**.**

いずれの権限も変更されていません。

**p**

権限が変更されました

**列 3**

これは、ユーザー所有権における変更を示しています。使用できる値は以下の通りです。

**.**

いずれのユーザー所有権も変更されていません。

**u**

ユーザー所有権が変更されました

**列 4**

これは、グループ所有権における変更を示しています。使用できる値は以下の通りです。

**.**

グループ所有権は変更されていません。

**g**

グループ所有権は変更されました。

**列 5**

これは拡張属性の変更を示しています。使用できる値は以下の通りです。

**.**

拡張属性は変更されていません。

**x**

拡張属性は変更されました。

**列 6**

これはアクセス制御リスト (ACL) における変更を示しています。使用できる値は以下の通りです。

**.**

ACL は変更されました。

**a**

ACL は変更されました。

## 14.4. 他の Snapper コマンド

以下のコマンド例は、「初期の Snapper セットアップ」と同じシナリオを使用しています。

スナップショットを削除するには、以下のコマンドを使用します。

```
# snapper -c home_vol delete snapshot_number
```

特定の Snapper ボリュームでスナップショットのリストを取得するには、以下のコマンドを使用します。

```
# snapper -c home_vol list
```

2 つの既存 Snapper スナップショット (たとえば、1 と 5) 間の変更ファイルのリストを取得するには、以下のコマンドを使用します。

```
# snapper -c home_vol status 1..5
```

2 つの既存 Snapper スナップショット (たとえば 1 と 5) 間のファイルの差分出力を取得するには、以下のコマンドを使用します。

```
# snapper -c home_vol diff 1..5
```

2 つの既存 Snapper スナップショット (たとえば 1 と 5) 間のファイルの xadiff 出力を取得するには、以下のコマンドを使用します。

```
# snapper -c home_vol xadiff 1..5
```

2 つの既存 Snapper スナップショット (たとえば 1 と 5) 間の変更を元に戻すには、以下のコマンドを使用します。

```
# snapper -c home_vol undochange 1..5
```

# 第15章 Swap 領域

## 15.1. Swap 領域とは？

Linux に於ける *Swap 領域* は、物理メモリー (RAM) の容量が満杯になった時点で使用されます。システムが更なるメモリーリソースを必要として、 RAM が満杯である場合、メモリー内の活動していないページが swap 領域に移動されます。Swap 領域は RAM の容量の一部としてマシンを支援しますが、より大容量の RAM の代用として考慮すべきではありません。swap 領域は、ハードディスク内に配置してありますが、そこでは物理メモリーよりもアクセスが遅くなります。

Swap 領域は専任の Swap パーティション (推奨)、Swap ファイル、あるいはその両方の組み合わせとして使用できます。

Swap は、物理 RAM サイズが 2 GB までは、物理 RAM の 2 倍として、2 GB 以上になると、 1 物理 RAM 毎に同量を追加します。しかし、決して 32 MB 以下にはしません。

> **例15.1 Swap RAM**
>
> 以下の場合:
>
> M = RAM 容量の GB 表示、S = swap サイズの GB 表示
>
> ```
> If M < 2
>  S = M *2
> Else
>  S = M + 2
> ```
>
> この数式を使うと、2 GB の物理 RAM を持つシステムは、4 GB の swap を持つことになり、3 GB の物理 RAM を持つシステムは 5 GB の swap を持つことになります。後日に RAM をアップグレードする予定の場合には、大きめの swap 領域のパーティションを作成すると、後で役に立ちます。

極めて大容量の RAM (32 GB以上) を持つシステムでは、多分上記基準より少なめの swap パーティション (物理 RAM の 1 倍、またはそれ以下)で充分でしょう。

**重要**

swap 領域として割り当てたファイルシステムおよび LVM2 ボリュームは使用中に修正しては**いけません**。システムプロセスやカーネルが swap 領域を使用している場合に swap への修正の試みはいずれも失敗します。swap の使用量とその使用場所を確認するには、**free** と **cat /proc/swaps** のコマンドを使用します。

Red Hat では、システムを **rescue** モードでブートして swap 領域を修正されるように推奨しています。**rescue** モードでのブート方法の案内については、『インスト－ルガイド』を参照してください。ファイルシステムのマウントを催促された時には、**Skip** を選択してください。

## 15.2. Swap 領域の追加

場合によっては、インスト－ルした後にさらに swap 領域の追加が必要になることもあります。例えば、システムの RAM の容量を 1 GB から 2 GB にアップグレードした時点に swap 領域が 2 GB のままかも知れません。メモリー集中型の操作や大容量のメモリーを必要とするアプリケーションを実行する場合には、

swap 領域を 4 GB に増加することが有効な対処法となります。

選択肢が 3 つあります: 新規の swap パーティションの作成、新規の swap ファイルの作成、あるいは、既存の LVM2 論理ボリューム上で swap の拡張。この中で既存論理ボリュームの拡張を推奨致します。

### 15.2.1. LVM2 の swap 領域を拡張する

Red Hat Enterprise Linux 7 は、デフォルトで、使用可能なすべての領域をインストール時に使用します。この設定を選択している場合、まず swap 領域として使用するボリュームグループに新しい物理ボリュームを追加しなければなりません。

swap 領域のボリュームグループにストレージを追加した後に、それを拡張することができるようになります。これを実行するには、次の手順に従います (**/dev/VolGroup00/LogVol01** ボリュームのサイズを 2 GB 拡張するとします)。

**手順15.1 LVM2 の swap 領域を拡張する**

1. 関連付けられている論理ボリュームの swap 機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. LVM2 論理ボリュームをサイズ変更して 2 GB 拡張します。

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3. 新しい swap 領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 拡張した論理ボリュームを有効にします。

```
# swapon -v /dev/VolGroup00/LogVol01
```

論理ボリュームが正常に拡張されたかどうか確認するには、**cat /proc/swaps** または **free** を使って swap 領域を確認します。

### 15.2.2. swap 用の LVM2 論理ボリュームを作成する

swap ボリュームグループを追加します (**/dev/VolGroup00/LogVol02** が追加する swap ボリュームであるとします)。

1. サイズが 2 GB の LVM2 論理ボリュームを作成します。

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. 新しい swap 領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol02
```

3. 次のエントリーを **/etc/fstab** ファイルに追加します。

```
# /dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

4. 拡張した論理ボリュームを有効にします。

```
# swapon -v /dev/VolGroup00/LogVol02
```

論理ボリュームが正常に作成されたかどうかテストするには、**cat /proc/swaps** または **free** を使って swap 領域を確認します。

### 15.2.3. swap ファイルを作成する

swap ファイルを追加します。

**手順15.2 swap ファイルの追加**

1. 新しい swap ファイルのサイズをメガバイト単位で決定してから 1024 をかけてブロック数を確定します。たとえば swap ファイルのサイズが 64 MB の場合、ブロック数は 65536 になります。
2. シェルに、以下のコマンドに必要なブロックサイズと等しい **count** を付けて入力します。

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

3. 次のコマンドで swap ファイルを設定します。

```
# mkswap /swapfile
```

4. swap ファイルをすぐに有効にします。ただし、起動時に自動的に有効にしません。

```
# swapon /swapfile
```

5. 起動時に有効にするには、次のエントリーを含めるように **/etc/fstab** を編集します。

   **/swapfile swap swap defaults 0 0**

   次にシステムが起動すると新しい swap ファイルが有効になります。

新しい swap ファイルが正常に作成されたかどうかをテストするには、**cat /proc/swaps** または **free** を使って swap 領域を確認します。

## 15.3. Swap 領域の削除

時には、インスト－ルの後に swap 領域を減量することが賢明な場合もあります。例えば、システムの RAM サイズを 1 GB から 512 MB にダウングレードした場合、swap 領域が 2 GB のままで残ることになります。この状況では、2 GB はディスク領域の無駄になりますので、swap 領域を 1 GB に減量した方がリソースの有効活用になります。

ここでも選択肢が 3 つあります: swap 用に使用していた LVM2 論理ボリューム全体を削除、swap ファイルの削除、あるいは既存の LVM2 論理ボリューム上の swap 領域の低減。

### 15.3.1. LVM2 論理ボリュームの swap 領域を縮小する

以下のようにして LVM2 の swap 論理ボリュームを縮小します (**/dev/VolGroup00/LogVol01** が縮小するボリュームであるとします)。

**手順15.3 LVM2 の swap 論理ボリュームを縮小する**

1. 関連付けられている論理ボリュームの swap 機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. LVM2 論理ボリュームをサイズ変更して 512 MB 分縮小します。

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. 新しい swap 領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 縮小した論理ボリュームを有効にします。

```
# swapon -v /dev/VolGroup00/LogVol01
```

swap 論理ボリュームサイズが正常に縮小されたことを確認するには、**cat /proc/swaps** または **free** を使って swap 領域を確認します。

### 15.3.2. swap 用の LVM2 論理ボリュームを削除する

swap ボリュームグループを削除します (**/dev/VolGroup00/LogVol02** が削除するボリュームであるとします)。

**手順15.4 swap ボリュームグループの削除**

1. 関連付けられている論理ボリュームの swap 機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol02
```

2. サイズが 512 MB の LVM2 論理ボリュームを削除します。

```
# lvremove /dev/VolGroup00/LogVol02
```

3. 次のエントリーを **/etc/fstab** ファイルから削除します。

```
/dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

論理ボリュームサイズが正常に削除されたことを確認するには、**cat /proc/swaps** または **free** を使って swap 領域を確認します。

### 15.3.3. swap ファイルを削除する

swap ファイルを削除します。

**手順15.5 swap ファイルの削除**

1. シェルプロンプトで次のコマンドを実行して swap ファイルを無効にします (swap ファイルの場所が**/swapfile** であるとします)。

```
# swapoff -v /swapfile
```

2. **/etc/fstab** ファイルから該当するエントリーを削除します。

3. 実際のファイルを削除します。

```
# rm /swapfile
```

## 15.4. Swap 領域の移動

swap 領域を 1 つの場所から別の場所に移動するには、swap 領域削除の手順に従います。それから swap 領域追加の手順に従います。

# 第16章 System Storage Manager (SSM)

*System Storage Manager* (SSM) は、さまざまなテクノロジーでストレージを管理するためのコマンドラインインターフェースを提供します。ストレージ環境が Device Mapper (dm)、論理ボリュームマネージャー (LVM)、および Multiple Devices (md) によって複雑になればなるほど、システムはユーザーにとって使いにくいものとなり、エラーや問題が生じやすくなります。SSM はこの問題に関する支援を提供します。

## 16.1. SSM のインストール

SSM をインストールするには、以下のコマンドを使用します。

```
# yum install system-storage-manager
```

関連パッケージがインストールされている場合にのみ、いくつかのバックエンドが有効にされます。

- LVM バックエンドには **lvm2 binaries** パッケージが必要です。
- Btrfs バックエンドには **btrfs progs** パッケージが必要です。
- Crypt バックエンドには、**dmsetup** および **cryptsetup** パッケージが必要です。

## 16.2. SSM コマンド

SSM には数多くのユーザー固有のコマンドがあります。さらに、それらのコマンドのいずれかと使用できる数多くのオプションがあります。

**-h, --help**

ヘルプのメッセージを表示して終了します。

**--version**

プログラムのバージョン番号を表示して終了します。

**-v, --verbose**

実行中に追加の情報を表示します。

**-f, --force**

SSM にいくつかの警告または質問がある場合に実行を強制します。

**-b *backend_type*, --backend *backend_type***

使用するバックエンドを選択します。現在の許可されるオプションは以下のとおりです。

- lvm
- btrfs
- crypt

**-n, --dry-run**

dry run (空実行) を実行します。コマンドラインのオプションを解析し、何も実行せずにシステム情報を収集します。これは、デバッグ目的で主に使用されますが、一部のチェックはバックエンドで処理されるため、すべてのチェックが実行する訳ではありません。

### 16.2.1. 一覧表示

**ssm list** は、システムで検出されるすべてのデバイス、プール、ボリュームおよびスナップショットについての情報を一覧表示します。**ssm list** を使用するだけで、利用可能なすべての情報を表示するか、または以下のオプションを使用して特定のセクションを要求することができます。

**volumes または vol**

このオプションは、システムにあるすべてのボリュームについての情報を一覧表示します。

**devices または dev**

このオプションは、システムにあるすべてのデバイスについての情報を一覧表示します。

**注記**

一部のデバイスは意図的に非表示になります。これには、ボリュームとして一覧表示される cdrom や DM/MD デバイスが含まれます。

**pools または pool**

このオプションは、システムにあるすべてのプールについての情報を一覧表示します。

**filesystems または fs**

このオプションは、システム上のファイルシステムを含むすべてのボリュームについての情報を一覧表示します。

**snapshots または snap**

このオプションは、システム上のすべてのスナップショットについての情報を一覧表示します。

**注記**

一部のバックエンドはスナップショットに対応せず、スナップショットと通常のボリュームを区別しないものもあります。この場合、SSM は、スナップショットを特定するためにボリューム名を認識しようとします。SSM の正規表現がスナップショットのパターンに一致しない場合、スナップショットは認識されません。

**-h または --help**

このオプションは、上記の情報を表示して終了します。

**例16.1 ssm list の例**

以下は、**ssm list** の表示例です。

```
# ssm list
----------------------------------------
Device           Total   Mount point
----------------------------------------
/dev/loop0     5.00 GB
/dev/loop1     5.00 GB
```

```
/dev/loop2     5.00 GB
/dev/loop3     5.00 GB
/dev/loop4     5.00 GB
/dev/sda     149.05 GB   PARTITIONED
/dev/sda1     19.53 GB   /
/dev/sda2     78.12 GB
/dev/sda3      1.95 GB   SWAP
/dev/sda4      1.00 KB
/dev/sda5     49.44 GB   /mnt/test
---------------------------------------
------------------------------------------------------------------------
--------
Volume     Pool     Volume size  FS     FS size      Free    Type
Mount point
------------------------------------------------------------------------
--------
/dev/dm-0  dm-crypt   78.12 GB   ext4  78.12 GB   45.01 GB  crypt
/home
/dev/sda1             19.53 GB   ext4  19.53 GB   12.67 GB  part   /
/dev/sda5             49.44 GB   ext4  49.44 GB   29.77 GB  part
/mnt/test
------------------------------------------------------------------------
--------
```

### 16.2.2. 作成

**ssm create** コマンドは、定義済みのパラメーターにより、新規ボリュームを作成します。以下のオプションを指定することでより具体的に設定することができます。

**-s *size* または --size *size***

このオプションは、新規の論理ボリュームに割り当てるサイズを指定します。以下のサフィックスは、2 のべき乗単位を定義するために許可されています。

- **K** または **k**
- **M** または **m**
- **G** または **g**
- **T** または **t**
- **P** または **p**
- **E** または **e**

デフォルトはキロバイトであり、サフィックスが指定されていない場合、SSM はこのデフォルトに設定されます。

このオプションが指定されていない場合、最大サイズが使用されます。

**-n *name* または --name *name***

このオプションは、新規の論理ボリュームに名前を指定します。このオプションが指定されない場合、対応するバックエンドで生成される名前が使用されます。バックエンドについてさらに詳しくは、「SSM のバックエンド」を参照してください。

**--fstype *filesystem_type***

このオプションは、ファイルシステムのタイプと新規の論理ボリュームを指定します。サポートされるファイルシステムは以下の通りです。

- ext3
- ext4
- xfs
- btrfs

このオプションが指定されない場合、ファイルシステムは作成されません。

**-r *level* または --raid *level***

このオプションは、新規ボリュームを作成する際に使用する RAID レベルを指定します。選択できる対応レベルは、以下の通りです。

- 0
- 1
- 10

**注記**

一部のバックエンドは、サポートされているすべての RAID レベルを実装する訳ではありません。

このオプションが指定されない場合、リニアボリュームが作成されます。

**-I *stripesize* または --stripesize *stripesize***

このオプションは、ストライプの粒度を表すキロバイト数を指定します。

**注記**

このオプションを使用するためには、RAID のレベルを指定する必要があります。

このオプションが指定されない場合、バックエンドのデフォルトが使用されます。バックエンドについてさらに詳しくは、「SSM のバックエンド」を参照してください。

**-i *stripes* または --stripes *stripes***

このオプションは、新規ボリュームのストライプ数を指定します。これは、論理ボリュームがまたがる物理ボリュームの数と等しくなります。

**注記**

このオプションを使用するためには、RAID のレベルを指定する必要があります。

このオプションが指定されない場合、ストライプのサイズが設定され、かつストライプが複数デバイスに指定されている場合、ストライプ数はデバイス数から自動的に判別されます。

**-p *pool* または --pool *pool***

このオプションは、新規ボリュームを作成するために使用するプールを指定します。プールが存在する場合、新規ボリュームはそのプールから作成されます (オプションで **device** をプールに追加します)。プールが存在しない場合、SSM は指定されている **device** で新規プールの作成を試行し、その後にこのプールから新規ボリュームを作成します。

**-backend** オプションが省略される場合、デフォルトの SSM バックエンドである LVM が使用されます。

**-e または --encrypt**

このオプションは、暗号化されたボリュームを作成します。使用できる拡張を指定できますが、許可されている拡張は以下の通りです。

- luks
- plain

**device *device***

デバイスが指定されている場合、デバイスは新規ボリュームを作成するために使用され、ボリュームの作成前にプールに追加されます (さらに詳しくは、「追加」を参照してください)。

指定されたデバイスが別のプールですでに使用されている場合、SSM は元のプールからこれを削除するかどうかを問い合わせます。これが拒否されるか、または削除が失敗するか、またはサイズが指定されていなかった場合などに、ボリューム作成は失敗します。サイズが指定されているものの、一部のデバイスをプールに追加できない場合、プールに十分な領域がある場合はボリューム作成は成功します。

**mount *mount_point***

マウントポイントが指定されている場合、SSM は作成後にボリュームのマウントを試行します。マウント可能なファイルシステムがボリュームにない場合は、マウントは失敗します。

**-h または --help**

このオプションにより、ヘルプのメッセージが表示され終了します。

**例16.2 ssm create の例**

以下は、**ssm create** の例です。これは、定義済みファイルシステム (ext4) が設定された、定義済みのサイズが (15G) のボリュームを作成します。デフォルトのバックエンドは LVM に設定され、LVM のデフォルトプール名は lvm_pool になります。

```
# ssm create --fs ext4 -s 15G /dev/loop0 /dev/loop1
```

### 16.2.3. 削除

**ssm remove** コマンドはシステムから項目を削除します。複数の項目を指定することができます。項目が何かの理由で削除されない場合はスキップされます。

使用できる項目には、以下が含まれます。

**device**

これは、デバイスをプールから削除します。デバイスがプールで使用されている場合にはこれを実行することができません。**-f** 引数がこの削除を強制実行します。デバイスがいずれのプールにも属さない場合は、スキップされます。

**pool**

これは、プールから作成されたすべてのボリュームを含め、システムからプールを削除します。

**volume**

これは、ボリュームをシステムから削除します。ボリュームがマウントされている場合、削除は失敗します。**-f** 引数を使って強制することはできません。

以下のオプションをコマンドに追加できます。

**-a または --all**

このオプションは、システム内のすべてのプールを削除します。

**-h または --help**

このオプションは、ヘルプのメッセージを表示して終了します。

**例16.3 ssm remove の例**

以下の例は、古いボリュームを削除するために **ssm remove** を使用する方法を示しています。

```
# ssm remove /dev/lvm_pool/lvol001
```

## 16.2.4. サイズ変更

**ssm resize** コマンドは、指定されたボリュームとファイルシステムのサイズを変更します。ファイルシステムがない場合、ボリュームのサイズのみが変更されます。

**注記**

デバイスは、ボリュームサイズが拡大する場合にのみプールに追加されます。

デバイスが別のプールですでに使用されている場合、SSM はこれを元のプールから削除するかどうかを問い合わせます。

場合によって、サイズの変更を行うためにファイルシステムをマウントする必要があることがあります。これは、ボリュームを一時的にマウントすることで SSM によって自動的に処理されます。

以下は、オプションの一覧です。

**-s *size* または --size *size***

このオプションは、ボリュームの新たなサイズを設定します。正 (+) または負 (-) の符号を新たな値に設定すると、ボリュームの実際のサイズから加算および減算が行なわれます。正または負を設定しないと、指定した値が新規ボリュームサイズになります。

サフィックスを追加して、値のサイズを指定します。許可されているサフィックスは以下の通りです。

- キロバイトの場合、**K** または **k**。
- メガバイトの場合、**M** または **m**。
- ギガバイトの場合、**G** または **g**。
- テラバイトの場合、**T** または **t**。
- ペタバイトの場合、**P** または **p**。

単位が指定されない場合、デフォルトサイズはキロバイトです。

**-h または --help**

このオプションは、ヘルプのメッセージを表示して終了します。

**例16.4 ssm resize の例**

以下は、**resize** コマンドの例です。これは、ボリューム**/dev/lvm_pool/lvol001** を 15 ギガバイトから 10 ギガバイトにサイズ変更します。

```
# ssm resize -s-5G /dev/lvm_pool/lvol001
```

以下は、ボリュームを 25 ギガバイトに増やす例で、ここではさらに多くのデバイスをプールに追加することが必要です。

```
# ssm resize -s 25G /dev/lvm_pool/lvol001 /dev/loop2
```

### 16.2.5. チェック

**ssm check** コマンドはボリューム上のファイルシステムの整合性をチェックします。チェックする複数のボリュームを指定することができます。ボリューム上にファイルシステムがない場合、ボリュームはスキップされます。

ファイルシステムをチェックするには、ファイルシステムをマウントしなければならない場合があります。これは、ボリュームを一時的にマウントすることで SSM によって自動的に処理されます。

以下のオプションは、**ssm check** に使用できます。

**-h または --help**

このオプションにより、ヘルプのメッセージが表示され終了します。

### 16.2.6. スナップショット

**ssm snapshot** コマンドは、既存ボリュームのスナップショットを取ります。

**注記**

この操作は、ボリュームが属するバックエンドがスナップショットをサポートしない場合には失敗します。

ボリュームのスナップショットを取るにはファイルシステムをマウントしなければならない場合があります。これは、ボリュームを一時的にマウントすることで SSM によって自動的に処理されます。

**注記**

***name*** と ***destination*** オプションは相互に排他的であるため、同時に指定することができません。

以下のオプションは、**ssm snapshot** コマンドに使用できます。

**-s *size* または --size *size***

これは、新規スナップボリュームに割り当てるサイズを指定します。許可されるサフィックスは以下の通りです。

- キロバイトの場合は、**K** または **k**
- メガバイトの場合は、**M** または **m**
- ギガバイトの場合は、**G** または **g**
- テラバイトの場合は、**T** または **t**
- ペタバイトの場合は、**P** または **p**
- エクサバイトの場合は、**E** または **e**

サイズのサフィックスは、2 のべき乗単位を定義するために使用できます。単位のサイズが指定されない場合、デフォルトはキロバイトです。このオプションが指定されない場合、サイズは自動的に決定されます。

**-d *destination* または --dest *destination***

このオプションは、新規のスナップショットに使用する絶対パスで指定される、スナップショットの宛先を指定します。このオプションが指定されない場合、デフォルトのバックエンドポリシーが実行されます。

**-n *name* または --name *name***

このオプションは、新規スナップショットの名前を指定します。これが指定されない場合、デフォルトのバックエンドポリシーが実行されます。

**-h または --help**

このオプションは、ヘルプのメッセージを表示して終了します。

**例16.5 ssm snapshot の例**

以下のコマンドは、Btrfs ボリューム *btrfs_pool:my_volume* のスナップショットを取ります。

```
# ssm snapshot btrfs:pool:my_volume
```

### 16.2.7. 追加

**ssm add** コマンドは、デバイス情報をプールに追加します。複数デバイスが指定されると、すべてがプールに追加されます。

**注記**

デバイスがすでに別のプールの一部である場合、コマンドは失敗します。ユーザーは、デバイスをそのプールから削除するかどうかを尋ねられます。

プールが指定されない場合、デフォルトのプールが選択されます。指定されるプールが存在しない場合、指定のデバイスを使って作成されます。

以下のオプションは、**ssm add** コマンドに使用できます。

**-p *pool* または --pol *pool***

このオプションは、デバイスが追加されるプールを指定します。これが指定されない場合、デフォルトのプールが使用されます。

**-h または --help**

このオプションは、ヘルプのメッセージを表示して終了します。

**例16.6 ssm add の例**

以下のコマンドは、デバイス **/dev/sda** および **/dev/sdb** を *btrfs_pool* プールに追加します。

```
# ssm add -p btrfs_pool /dev/sda /dev/sdb
```

### 16.2.8. マウント

**ssm mount** コマンドは、指定されたディレクトリーにボリュームをマウントしません。ボリュームは、**man mount(8)** に詳しく説明されているのと同じ方法で指定できます。さらに、ボリュームがコマンド **ssm list** を使って表示される表のフォーマットにボリュームを指定することもできます。たとえば、Btrfs サブボリューム **btrfs_pool:vol001** をマウントするために、このサブボリュームを構成するデバイスとサブボリューム ID を判別する代わりに、**ssm mount btrfs_pool:vol001 /mnt/test** を呼び出すことができます。

以下のオプションは、**ssm mount** に利用できます。

**-o *options* または --options *options***

オプションは、**-o** フラグと、それに続くコンマ区切りのストリングのオプションで指定されます。このオプションは、**man mount(8)** で指定されるオプションと同等のものです。

**-h または --help**

このオプションは、ヘルプのメッセージを表示して終了します。

**例16.7 `ssm mount` の例**

以下のコマンドは、BTRFS サブボリューム *btrfs_pool:vol001* を **`/mnt/test`** にマウントします。

```
# ssm mount btrfs_pool:vol001 /mnt/test
```

## 16.3. SSM のバックエンド

SSM は、Device Mapper (DM)、Btrfs ファイルシステム、および Multiple Devices (MD) などの各種テクノロジーの統一されたユーザーインターフェースを作成します。これを実行するために、**`ssmlib/main.py`** に核となる抽象化層が置かれます。この抽象化層は、基礎となるテクノロジーを全く認識しませんが、デバイス、プール、およびボリュームの抽象化に準拠します。

各種のバックエンドを **`ssmlib/main.py`** に登録して、ボリュームおよびプールの **`create`**、**`snapshot`**、または **`remove`** などのメソッドを実装する特定のストレージテクノロジーを処理します。その後、コアはこれらのメソッドを呼び出して、ストレージの下部に何があるかを認識する必要なく、ストレージを管理できます。現在、SSM で登録されたバックエンドは Btrfs、LVM、Crypt、および MD です。

### 16.3.1. Btrfs バックエンド

Btrfs は、ボリューム管理を含む多くの高度な機能を持つファイルシステムです。Btrfs は SSM の他のファイルシステムとは異なる方法で処理され、ボリューム管理のバックエンドとして使用されます。プール、ボリューム、およびスナップショットは Btrfs バックエンドを使って作成できます。以下は Btrfs の観点から見たこれらの定義になります。

**プール**

プールは、デバイスを追加することで拡張し、デバイスを削除することで縮小できるため、それ自体が BTRFS ファイルシステムです。サブボリュームおよびスナップショットも作成することができます。BTRFS プールが作成されると、SSM は BTRFS ファイルシステムを作成します。つまり、すべての新規 BTRFS プールは、プール全体を削除しないと削除できないプール自体と同じ名前の 1 つのボリュームを持つことになります。デフォルトの BTRFS プール名は **`btrfs_pool`** です。

新規の BTRFS プールを作成する際、プールの名前はファイルシステムのラベルとして使用されます。システム内にラベルなしの BTRFS ファイルシステムがすでにある場合には、BTRFS プール名は、**`btrfs_`*`device_base_name`*** の形式で、内部使用向けの名前を生成します。

BTRFS プールは、**`create`** または **`add`** コマンドのいずれかが指定されたデバイス、および存在していないプール名を設定して使用される場合に作成されます。

**volume**

BTRFS バックエンドのボリュームは、最初のボリュームが BTRFS プールで作成される場合を除くと BTRFS サブボリュームと同じです。サブボリュームは、マウント時に BTRFS ファイルシステム上でのみ作成できますが、SSM は、これを実行するためにファイルシステムを一時的に自動マウントします。

ボリューム名は、BTRFS ファイルシステム内のサブボリュームパスとして使用され、ボリュームを作成するためにはこのパスにあるすべてのオブジェクトが存在している必要があります。内部トラッキングまたはユーザーへの表示用に使用するボリューム名は、**`{pool_name}:{volume_name}`** の形式で生成されます。ボリュームはそのマウントポイントで参照することもできます。

主な BTRFS ボリューム (ファイルシステム自体) を除くと、BTRFS ボリュームは、ファイルシ

ステムのマウント時に **list** 出力にのみ表示されます。

新規の BTRFS ボリュームは **create** を使って作成できます。

**スナップショット**

BTRFS ファイルシステムは、サブボリュームのスナップショット作成をサポートします。つまり、スナップショットは、SSM によりシステムのいずれの BTRFS ボリュームからも作成することができます。ただし、スナップショットは実際には一部のブロックが別のサブボリュームと共有されているサブボリュームに過ぎないため、サブボリュームとスナップショットを区別しません。つまり、SSM は BTRFS スナップショットの宛先を認識しませんが、その代わりに BTRFS ボリュームの特殊な名前の形式を認識します。スナップショットの作成時に指定される名前が特定のパターンに一致しない場合、スナップショットは認識されず、その代わりに通常の BTRFS ボリュームとして一覧表示されます。

新規の BTRFS スナップショットは、**snapshot** コマンドを使用して作成できます。

**デバイス**

BTRFS では、特殊デバイスをその上に作成する必要がありません。

### 16.3.2. LVM バックエンド

プール、ボリューム、およびスナップショットは LVM を使って作成できます。以下は、LVM の観点から見た定義です。

**プール**

LVM プールは、LVM ボリュームグループと同じです。つまり、デバイスと新規論理ボリュームのグループ化は LVM プールから実行できます。デフォルトの LVM プール名は **lvm_pool** です。

LVM プールは、**create** または **add** コマンドのいずれかが指定されるデバイスや、すでに存在しなくなったプール名と共に使用される場合に作成されます。

**ボリューム**

LVM ボリュームは論理ボリュームと同じですが、**create** コマンドで作成できます。

**スナップショット**

LVM ボリュームのスナップショットを作成することができます。スナップショットが LVM ボリュームから作成される場合、新規の **snapshot** ボリュームが作成され、それは他の LVM ボリュームと全く同様の方法で処理されます。BTRFS とは異なり、LVM はスナップショットと通常のボリュームを区別することができるため、スナップショットの名前が特定のパターンに一致している必要はありません。

**デバイス**

LVM では、物理デバイスをデバイス上に作成する必要がありますが、SSM はこれをユーザーに対して透過的に表示します。

### 16.3.3. Crypt

SSM における Crypt バックエンドは、cryptsetup と dm-crypt ターゲットを使用して暗号化されたボリュームを管理します。Crypt バックエンドは、通常のブロックデバイス上 (または LVM または MD ボリュームなどの他のボリューム上) に暗号化されたボリュームを作成するための通常のバックエンドとして使用したり、単一の手順で暗号化された LVM ボリュームを作成するために使用したりできます。

ボリュームのみが crypt バックエンドを使って作成できます。プールはサポートされておらず、特殊なデバイスは不要です。

以下は、crypt バックエンドの観点から見た定義です。

**プール**

Crypt バックエンドはプールをサポートしません。crypt プールを作成することも、デバイスをプールに追加することもできません。

**ボリューム**

Crypt バックエンドのボリュームは、**dm-crypt** で作成されるボリュームです。これは、暗号化されていないフォームで元の暗号化デバイス上のデータを表します。Crypt ボリュームを作成するには 1 つのデバイスしか使用できません。これは、RAID またはその他のデバイスの連結をサポートしません。

luks および plain の 2 つのモードまたは拡張がサポートされます。Luks はデフォルトで使用されます。これらの拡張についてさらに詳しくは、**man cryptsetup** を参照してください。

**スナップショット**

Crypt バックエンドはスナップショットの作成をサポートしていません。ただし、暗号化されたボリュームが LVM ボリューム上に作成される場合、ボリューム自体のスナップショットを作成できます。その後、スナップショットは **cryptsetup** を使用して開くことができます。

**デバイス**

Crypt バックエンドでは、その上に特殊デバイスを作成する必要がありません。

### 16.3.4. Multiple Devices (MD)

MD バックエンドは、現時点で、システム内の MD ボリュームについての情報を収集することに制限されています。

## 16.4. SSM の参考文献

- SSM のローカルドキュメントは **doc/** に格納されます。

**注記**

ドキュメントは、重複を避けるために Sphinx ソフトウェアを使用して作成されます。従って、ドキュメントを変更する場合は、man ページ、HTML ページ、または **README** ディレクトリーを変更しないでください。その代わりに、**doc/*.rst** および **doc/src/*.rst** ファイルを随時使用して、変更がすべてのドキュメントに伝播するようにしてください。

- SSM wiki は、http://storagemanager.sourceforge.net/index.html からアクセスできます。
- メーリングリストは https://lists.sourceforge.net/lists/listinfo/storagemanager-devel からサブスクライブでき、メーリングリストのアーカイブは http://sourceforge.net/mailarchive/forum.php?forum_name=storagemanager- devel からアクセスできます。メーリングリストは、開発者がコミュニケーションに使う場所です。ユーザーのメーリングリストはありませんので、質問がある場合はここに自由に投稿することもできます。

# 第17章 ディスク割り当て

ディスク領域はディスククォータ (quota) によって制限できます。ディスククォータは、ユーザーが過度のディスク領域を消費するか、パーティションが満杯になる前にシステム管理者に警告をします。

ディスククォータは、ユーザーグループ用にも個別のユーザー用にも設定できます。これにより、ユーザーが参加しているプロジェクトに割り振られた領域 (プロジェクトごとに所有グループが存在すると想定) とは別に、ユーザー固有のファイル (電子メールなど) に割り振った領域を管理することが可能になります。

さらにクォータは、消費されるディスクブロックの数の制御だけでなく、inode (UNIX ファイルシステム内のファイルに関する情報を含むデータ構造) の数も制御するように設定できます。inode はファイル関連の情報を組み込むように使用されるため、これが作成されるファイルの数を制御することも可能にします。

ディスククォータを実装するには、**quota** RPM をインストールしておく必要があります。

**注記**

この章はすべてのファイルシステムを対象としていますが、一部のファイルシステムには独自のクォータ管理ツールが含まれています。詳細については、これらのファイルシステムの説明の該当する部分を参照してください。

XFS ファイルシステムについては、「XFS クォータの管理」を参照してください。

Btrfs にはディスククォータがないため、ここでは扱われません。

## 17.1. ディスククォータの設定

ディスククォータを実装するには、以下のステップを使用します。

1. **/etc/fstab** を修正することで、ファイルシステムごとのクォータを有効にします。
2. ファイルシステムを再マウントします。
3. クォータデータベースファイルを作成して、ディスク使用状況テーブルを生成します。
4. クォータポリシーを割り当てます。

これらの各ステップは次のセクションで詳しく説明されています。

### 17.1.1. クォータの有効化

root としてテキストエディターを使用し、**/etc/fstab** ファイルを編集します。

**例17.1 /etc/fstab の編集**

たとえば、テキストエディター **vim** を使用するには、以下を入力します。

```
# vim /etc/fstab
```

**usrquota** と **grpquota** のどちらかのオプション、またはそれら両方をクォータが必要となるファイルシステムに追加します。

**例17.2 クォータの追加**

```
/dev/VolGroup00/LogVol00 /          ext3    defaults         1 1
LABEL=/boot              /boot      ext3    defaults         1 2
none                     /dev/pts   devpts  gid=5,mode=620   0 0
none                     /dev/shm   tmpfs   defaults         0 0
none                     /proc      proc    defaults         0 0
none                     /sys       sysfs   defaults         0 0
/dev/VolGroup00/LogVol02 /home      ext3    defaults,usrquota,grpquota
1 2
/dev/VolGroup00/LogVol01 swap       swap    defaults         0 0 . . .
```

この例では、**/home** ファイルシステムがユーザーとグループの両方のクォータを有効にしています。

**注記**

以下の例では、Red Hat Enterprise Linux のインストール時に個別の **/home** パーティションが作成されていると想定しています。root (**/**) パーティションは **/etc/fstab** ファイル内でクォータポリシーを設定するために使用できます。

### 17.1.2. ファイルシステムの再マウント

**usrquota** と **grpquota** オプションのどちらか、またはそれら両方を追加した後には、**fstab** エントリーが修正されたそれぞれのファイルシステムを再マウントします。ファイルシステムがどのプロセスで使用されていない場合は、以下のメソッドのいずれかを使用します。

- **umount** コマンドを発行して、その後に**mount** コマンドを発行してファイルシステムを再マウントします。各種ファイルシステムのマウントとアンマウント用の特定の構文に関しては、**umount** と **mount** の両方の **man** ページを参照してください。
- **mount -o remount *file-system*** コマンド (ここで ***file-system*** はファイルシステムの名前) を発行してファイルシステムを再マウントします。たとえば、**/home** ファイルシステムを再マウントするために発行するコマンドは、**mount -o remount /home** です。

ファイルシステムが現在使用中の場合、そのファイルシステムを再マウントする最も簡単な方法は、システムを再起動することです。

### 17.1.3. クォータデータベースファイルの作成

クォータが有効にされたそれぞれのファイルシステムを再マウントした後は、**quotacheck** コマンドを実行します。

**quotacheck** コマンドは、クォータが有効にされているファイルシステムを検証し、現在のディスク使用状況テーブルをファイルシステムごとに作成します。このテーブルは、ディスク使用状況についてのオペレーティングシステム用コピーを更新するのに使用されます。また、ファイルシステムのディスククォータファイルが更新されます。

**注記**

**quotacheck** コマンドは、ディスク使用状況テーブルがマウント時に自動的に完成するため、XFS には全く影響を与えません。詳細情報については、**xfs_quota(8)** の man ページを参照してください。

クォータファイル (**aquota.user** と **aquota.group**) をファイルシステム上で作成するには、**quotacheck** コマンドで **-c** オプションを使用します。

**例17.3 クォータファイルの作成**

たとえば、ユーザーとグループ用のクォータが **/home** ファイルシステム上で有効になっている場合、以下のようにして **/home** ディレクトリー内にファイルを作成します。

```
# quotacheck -cug /home
```

**-c** オプションは、クォータが有効なそれぞれのファイルシステムにクォータファイルを作成すべきことを指定し、**-u** オプションは、ユーザークォータ用のチェックを指定し、**-g** オプションはグループクォータ用のチェックを指定します。

**-u** と **-g** のどちらのオプションも指定されていない場合は、ユーザークォータファイルのみが作成されます。 **-g** のみが指定されている場合には、グループクォータファイルのみが作成されます。

ファイル作成が完了した後は、以下のコマンドを実行してクォータが有効なファイルシステムごとの現在のディスク使用状況テーブルを生成します。

```
# quotacheck -avug
```

使用されているオプションは以下のようになります。

**a**

クォータが有効にされた、ローカルマウントのファイルシステムをすべてチェック

**v**

クォータチェックの進行状態について詳細情報を表示

**u**

ユーザーディスククォータの情報をチェック

**g**

グループディスククォータの情報をチェック

**quotacheck** の実行が終了すると、有効なクォータ (ユーザーまたはグループ、あるいは両方) に対応するクォータファイルには、**/home** などのクォータが有効なローカルマウントの各ファイルシステム用のデータが追加されます。

### 17.1.4. ユーザーごとのクォータの割り当て

最終ステップは、**edquota** コマンドを使用したディスククォータの割り当てです。

ユーザーにクォータを設定するには、シェルプロンプトで root として以下のコマンドを実行します。

```
# edquota username
```

クォータを必要とする各ユーザーに対してこのステップを実行します。クォータが **/etc/fstab** 内で **/home** パーティション (以下の例では **/dev/VolGroup00/LogVol02**) 用に有効になっていて、コマンド **edquota testuser** が実行されると、システムのデフォルトとして設定されているエディターに以下が表示されます。

```
Disk quotas for user testuser (uid 501):
Filesystem                   blocks     soft     hard    inodes   soft
hard
/dev/VolGroup00/LogVol02  440436        0        0     37418      0
0
```

**注記**

**EDITOR** 環境変数で定義されているテキストエディターが**edquota** によって使用されます。このエディターを変更するには、使用している **~/.bash_profile** ファイル内で **EDITOR** 環境変数を選択するエディターへの完全パスに設定します。

最初の列はクォータが有効になっているファイルシステムの名前です。2 つ目の列はユーザーが現在使用しているブロック数を示します。その次の 2 つの列はこのファイルシステムでのユーザーに対するソフトおよびハードブロックリミットを設定するために使用されます。**inodes** の列は、現在ユーザーが使用している inode の数を示します。最後の列は、ファイルシステム上のユーザーに対するソフトおよびハードの inode 制限を設定するために使用されます。

ブロックのハードリミットは、ユーザーまたはグループが使用できる絶対的な最大ディスク容量です。この上限に達すると、それ以上のディスク容量を使用できなくなります。

ブロックのソフトリミットは、使用可能なディスク容量の最大値を定義します。しかし、ハードリミットとは異なり、ソフトリミットは一定の期間だけ超過できるようになっています。この期間は*猶予期間* として知られています。猶予期間は秒、分、時間、日、週、または月で表されます。

いずれかの値がゼロに設定してある場合は、そのリミットは設定されていないことになります。テキストエディターで必要な制限に変更します。

**例17.4 必要な制限の変更**

例:

```
Disk quotas for user testuser (uid 501):
Filesystem                   blocks     soft     hard   inodes   soft
hard
/dev/VolGroup00/LogVol02  440436   500000   550000    37418      0
0
```

ユーザー用にクォータが設定されていることを確認するには、次のコマンドを使用します。

```
# quota username
Disk quotas for user username (uid 501):
```

```
    Filesystem   blocks    quota    limit    grace    files    quota    limit
grace
     /dev/sdb    1000*    1000     1000                0        0        0
```

### 17.1.5. グループごとのクォータの割り当て

クォータはグループごとにも割り当てることができます。たとえば、**devel** グループ (グループクォータ設定前にグループが存在していなければなりません) 用にグループクォータを設定するには、次のコマンドを使用します。

```
# edquota -g devel
```

このコマンドはグループの既存クォータをテキストエディター内に表示します。

```
Disk quotas for group devel (gid 505):
Filesystem                   blocks    soft      hard    inodes    soft
hard
/dev/VolGroup00/LogVol02   440400       0         0     37418       0
0
```

このリミットを変更し、ファイルを保存します。

グループクォータが設定されたことを確認するには、次のコマンドを使用します。

```
# quota -g devel
```

### 17.1.6. ソフトリミットの猶予期間の設定

所定のクォータがソフトリミットを持つ場合、その猶予期間 (ソフトリミットを超過してもよい期間の値) は以下のコマンドで編集できます。

```
# edquota -t
```

このコマンドはユーザーまたはグループのいずれかの inode またはブロックのクォータに対して機能します。

**重要**

他の **edquota** コマンドは特定のユーザーまたはグループのクォータで機能しますが、**-t** オプションはクォータが有効になっているすべてのファイルシステムで機能します。

## 17.2. ディスククォータの管理

クォータが実装されている場合は、その維持が必要になります — これはほとんどの場合、クォータを超過しているかどうかをチェックして、クォータが正確であることを確認する形で行われます。

当然のことながら、ユーザーが頻繁にそのクォータを超過したり、常にそのソフトリミットに達する場合には、システム管理者は、それらのユーザーのタイプとそのユーザーの作業に影響するディスク容量に応じていくつかの選択肢を持ちます。システム管理者はユーザーが使用ディスク容量を低減する方法を判断できるように支援するか、またはユーザーのディスククォータを増加させることができます。

### 17.2.1. 有効化と無効化

クォータはゼロに設定することなく、無効にすることができます。すべてのユーザーとグループのクォータをオフにするには、以下のコマンドを使用します。

```
# quotaoff -vaug
```

**-u** か **-g** のどちらも指定されていない場合、ユーザーのクォータのみが無効になります。**-g** のみが指定されている場合は、グループのクォータのみが無効になります。**-v** スイッチはコマンドが実行する際に状態の詳細情報を表示します。

クォータを再度有効にするには、同じオプションで **quotaon** コマンドを使用します。

たとえば、すべてのファイルシステムにユーザーとグループのクォータを有効にするには、以下のコマンドを使用します。

```
# quotaon -vaug
```

**/home** などの特定のファイルシステムにクォータを有効にするには、以下のコマンドを使用します。

```
# quotaon -vug /home
```

**-u** または **-g** のどちらも指定されていない場合、ユーザーのクォータのみが有効になります。**-g** のみが指定されている場合は、グループのクォータのみが有効になります。

**注記**

**quotaon** コマンドは、マウント時に自動的に実行されるため、XFS に常に必要となる訳ではありません。詳細情報については、**quotaon(8)** の man ページを参照してください。

### 17.2.2. ディスククォータに関するレポート

ディスク使用状況のレポートを作成するには、**repquota** ユーティリティーの実行が必要になります。

**例17.5 repquota コマンドの出力**

たとえば、コマンド **repquota /home** は以下の出力を生成します。

```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol02
Block grace time: 7days; Inode grace time: 7days
   Block limits   File limits
User  used soft hard grace used soft hard grace
----------------------------------------------------------------------
root      --      36       0       0              4     0     0
kristin   --     540       0       0            125     0     0
testuser  --  440400  500000  550000          37418     0     0
```

クォータが有効なすべてのファイルシステム (オプション **-a**) についてのディスクの使用状況を表示するには、次のコマンドを使用します。

```
# repquota -a
```

分かりやすいレポートが表示されますが、ここでいくつかのポイントを説明します。各ユーザーの後に表示される **--** はブロックまたは inode が超過しているかどうかを素早く判別するための手段です。どちらかのソフトリミットが超過していると、対応する **-** の位置に **+** が表示されます。最初の**-** はブロックの制限で、2 つ目が inode の制限を示します。

通常、**grace** 列は空白です。ソフトリミットが超過した場合、この列には猶予期間の残り時間の値に等しい時間指定が含まれます。猶予期間が超過した場合、この位置には **none** が表示されます。

### 17.2.3. 正確なクオータの維持

ファイルシステムが正常にアンマウントできなかった場合 (システムクラッシュが原因の場合など) に、**quotacheck** を実行する必要があります。しかし、システムがクラッシュしていない場合でも **quotacheck** は定期的に実行することができます。**quotacheck** を定期的に実行する際の安全な方法には以下が含まれます。

**次回の再起動時に quotacheck を確実に実行する**

**注記**

この方法は、定期的に再起動される (ビジーな) 複数ユーザーシステムに最も適しています。

root として、**touch /forcequotacheck** コマンドを含んだシェルスクリプトを、**/etc/cron.daily/** または **/etc/cron.weekly/** ディレクトリーに配置するか、あるいは **crontab -e** コマンドを使用して上記のコマンドを含むスクリプトをスケジュールします。このスクリプトは root ディレクトリーに空の **forcequotacheck** ファイルを作成するため、起動時にシステムの init スクリプトがそれを検索します。それが発見されると init スクリプトは **quotacheck** を実行します。その後、init スクリプトは**/forcequotacheck** ファイルを削除します。このように、**cron** でこのファイルが定期的に作成されるようにスケジュールすることにより、次回の再起動時に **quotacheck** を確実に実行することができます。

**cron** についてさらに詳しくは、**man cron** を参照してください。

**シングルユーザーモードで quotacheck を実行**

**quotacheck** を安全に実行する別の方法として、クオータファイルのデータ破損の可能性を回避するためにシングルユーザーモードでシステムを起動して、以下のコマンドを実行する方法があります。

```
# quotaoff -vug /file_system
```

```
# quotacheck -vug /file_system
```

```
# quotaon -vug /file_system
```

**実行中のシステム上で quotacheck を実行**

必要な場合には、いずれのユーザーもログインしておらず、チェックされているファイルシステムに開いているファイルがない状態のマシン上で **quotacheck** を実行することができます。**quotacheck -vaug *file_system*** コマンドを実行します。このコマンドは、**quotacheck** が指定の *file_system* を読み込み専用として再マウントできない場合に失敗し

ます。チェックの後には、ファイルシステムは読み込み/書き込みとして再マウントされることに注意してください。

**警告**

読み込み/書き込みでマウントされているライブのファイルシステム上での **quotacheck** の実行は、quota ファイルが破損する可能性があるため、推奨されません。

**cron** の設定方法についてさらに詳しくは、**man cron** を参照してください。

## 17.3. 参照

ディスククォータに関する詳細情報については、以下のコマンドの **man** ページを参照してください。

- **quotacheck**
- **edquota**
- **repquota**
- **quota**
- **quotaon**
- **quotaoff**

# 第18章 RAID (Redundant Array of Independent Disks)

RAID の登場した背景には、容量が小さく手頃なディスクドライブを複数集めてアレイに結合させ、容量が大きく高価なドライブに負けないパフォーマンスと冗長性を実現しようとする動きがありました。この複数のデバイスからなるアレイは、コンピューター上では単一の論理ストレージユニットまたはドライブとして表されます。

## 18.1. RAID とは

RAID では複数のディスクに情報を拡散させることができます。*ディスクのストライピング*(RAID レベル 0)、*ディスクのミラーリング*(RAID レベル 1)、*パリティーによるディスクのストライピング*(RAID レベル 5) などの技術を使用して冗長性を得ながら待ち時間を抑え、帯域幅を増幅させることでハードディスクがクラッシュした場合の復元力を最大限に引き出します。

データは、一貫して同じサイズの大きさのチャンク (通常は 256K または 512K、 ただし他の値も可) に分割され、 アレイ内の各ドライブに分散されます。 各チャンクは導入している RAID レベルに応じて RAID アレイ内のハードドライブに書き込まれます。 データが読み込まれるとこのプロセスが逆をたどります。その動作はアレイ内の複数のドライブがまるで一台の大容量ドライブであるかのように見えます。

## 18.2. 誰が RAID を必要とするか

システム管理者や大容量のデータを管理している方にとって RAID は利点の多いテクノロジーとなります。RAID を採用する主な利点を示します。

- 速度を高める
- 単一の仮想ディスクを使用してストレージ容量を増加させる
- ディスク障害によるデータ損失のリスクを最小限に抑える

## 18.3. RAID のタイプ

RAID には、 ファームウェア RAID、 ハードウェア RAID、 ソフトウェア RAID の 3 種類の RAID タイプがあります。

### ファームウェア RAID

*ファームウェア RAID* (ATARAID とも呼ばれる) とは、 ソフトウェア RAID の種類でファームウェアベースのメニューを使って RAID セットを設定することができます。 この種類の RAID で使用されるファームウェアは BIOS にも搭載され、 RAID セットから起動することができます。 RAID セットのメンバーをマークするオンディスクのメタデータ形式は製造元によって異なります。 Intel Matrix RAID がファームウェア RAID システムの一例となります。

### ハードウェア RAID

ハードウェアベースのアレイでは、 RAID サブシステムをホストとは別に管理します。 ホストに対して、RAID アレイごと 1 つのディスクを表します。

ハードウェア RAID デバイスはシステムの内部にあっても外部にあっても構いません。 内部デバイスは一般的には特殊なコントローラーカードで構成され、 RAID の作業はオペレーティングシステムに対して透過的に処理されます。 外部デバイスは一般的には SCSI、 ファイバーチャネル、 iSCSI、 InfiniBand、 他の高速ネットワークなどの相互接続でシステムに接続され、 システムには論理ボリュームとして表されます。

RAID コントローラーカードはオペレーティングシステムに対し SCSI コントローラーのように動作し、実際のドライブ通信をすべて処理します。 ドライブはユーザーによって RAID コントローラに接続されてから (通常の SCSI コントローラー同様に)、 RAID コントローラーの構成に追加されます。 オペレーティングシステムではこの違いを認識できません。

### ソフトウェア RAID

ソフトウェア RAID では、 カーネルディスク (ブロックデバイス) コード内に各種の RAID レベルを実装しています。 高価なディスクコントローラーカードやホットスワップ機能のシャーシ [2] などを必要としないため、 もっとも安価なソリューションとなります。 ソフトウェア RAID は安価な IDE ディスクでも SCSI ディスクでも動作します。 また、 最近の高速な CPU でもソフトウェア RAID は一般的にハードウェア RAID より優れたパフォーマンスを見せます。

Linux カーネルには *マルチディスク* (MD) ドライバーが含まれ、 これにより RAID ソリューションは完全にハードウェアに依存しなくてもよくなります。 ソフトウェアベースのアレイのパフォーマンスはサーバーの CPU 性能や負荷に左右されます。

Linux ソフトウェア RAID スタックの主な機能のいくつかを示します。

- マルチスレッド設計
- 再構成を行うことなく Linux マシン間でのアレイの移動が可能
- 待機状態のシステムリソースを使ってバックグラウンドでのアレイの再構成が可能
- ホットスワップ可能なドライブのサポート
- CPU の自動検出でストリーミング SIMD サポートなどの特定 CPU の機能を活用
- アレイ内の複数ディスク上にある不正セクターを自動修正
- RAID データの整合性を定期的にチェックしアレイの健全性を確保
- アレイの予防的なモニタリング、 重要なイベントが発生した際は指定アドレスへの警告メールを送信
- 書き込みを集中としたビットマップ、 アレイ全体を再同期させるのではなく再同期を必要とするディスク部分を正確にカーネルに認識させることで再同期イベントの速度を大幅に高速化
- 再同期のチェックポイント機能、 再同期中のコンピュータの再起動時に全体をすべてやり直すのではなく前回の停止時点から再同期を開始
- インストール後のアレイのパラメーター変更が可能、 新しいディスクを追加しディスク 4 台の RAID5 から 5 台の RAID5 に増大させることが可能。 作業はライブで行うことができ、 新しいアレイでの再インストールは不要

## 18.4. RAID レベルとリニアサポート

RAID ではレベル 0、 1、 4、 5、 6、 10、 リニアなどの各種設定に対応します。 RAID タイプは以下のように定義されます。

**レベル 0**

ストライピングとも呼ばれる RAID レベル 0 はストライプ化したデータをマッピングする技術でパフォーマンスが本位とされます。 つまり、 アレイに書き込まれたデータはストライプに分割されアレイのメンバーディスク群に分散して書き込まれます。レベル固有のオーバーヘッドも少なく I/O パフォーマンスに優れていますが冗長性はまったくありません。

RAID レベル 0 実装の多くがアレイ内のメンバーデバイス群にデータの分散を行う際、 最小サイ

ズのデバイスに合わせたサイズまでしかデータの分散を行いません。 つまり、 各デバイスの容量が異なる場合、 すべて最小サイズのドライブと同じサイズであるかのように扱われます。 したがってレベル 0 アレイの一般的なストレージ容量はハードウェア RAID 内の最小サイズのメンバーディスクの容量と同じになるか、 ソフトウェア RAID 内の最小サイズのメンバーパーティションにアレイ内のディスク数かパーティション数をかけたものと同じ容量になります。

**レベル 1**

ミラーリングとも呼ばれる RAID レベル 1 は RAID 形式の中では最も長く使用されているレベルになります。 同一データをアレイの各メンバーディスクに書き込むことで冗長性を提供し、 各ディスクにミラリングしたコピーを残します。 ミラーリングは簡素でありながら高いデータ可用性を提供するため現在でもよく使用されています。 レベル 1 は 2 台以上のディスクで動作、 データに関する信頼性が高く読み込み頻度の高いアプリケーションのパフォーマンスは向上されますが、相当のオーバーヘッドも必要とします。 [3]

レベル 1 アレイのストレージ容量は、 ハードウェア RAID 内でミラリングされている最小サイズのハードディスクと同じ容量か、 ソフトウェア RAID 内でミラリングされている最小のパーティションと同じ容量になります。 レベル 1 の冗長性が RAID タイプのなかでは最高となります。 アレイは単一ディスクのみで動作可能です。

**レベル 4**

レベル 4 ではデータ保護のため単一ディスクドライブに集中したパリティ [4] を使用します。 専用パリティディスクは RAID アレイへのすべての書き込みトランザクションでパリティ固有のボトルネックとなるため、 システム管理者が意図的にこのボトルネックを考慮に入れてソフトウェア RAID デバイスを設計している場合を除き (アレイにデータの移植が完了したら書き込みのトランザクションはほとんど発生しないようなアレイ)、 レベル 4 はライトバックのキャッシュ機能などのテクノロジーが付随されない限りめったに使用されません。 このように RAID レベル 4 はめったに使用されないため、 Anaconda のオプションとして提供されていません。 ただし、 ユーザーが必要とする場合には手動による作成が可能です。

ハードウェア RAID レベル 4 のストレージ容量は、 最小サイズのメンバーのパーティションにパーティション数から **1 を引いた** 数をかけたものと同じ容量になります。 RAID レベル 4 アレイのパフォーマンスは常に非対称となります。 つまり、 読み込みの方が書き込みより優れているということです。 パリティを生成する際に書き込みの方が CPU やメインメモリーを多く消費し、 また実際のデータをディスクに書き込み際にもデータだけではなくパリティも書き込むためバスの帯域幅を余計に消費します。 一方、 読み込みの場合はアレイが低下状態でない限りはデータを読み取るだけでパリティは関係ありません。 結果、 通常の動作状況で同じデータ転送量の場合、 読み込みの方がドライブやコンピュータのバス全体に対するトラフィックが少なくなります。

**レベル 5**

最も一般的なタイプの RAID です。 アレイのメンバーディスクドライブすべてにパリティを分散させることで、 RAID レベル 5 はレベル 4 で見られた書き込みに関するボトルネックを解消します。 パフォーマンス関連の唯一のボトルネックはパリティを計算するプロセス自体となります。 最近の CPU やソフトウェア RAID ではパリティを非常に早く生成できるようになってきたため、 これもボトルネックではなくなってきています。 ただし、 全デバイスに渡り結合された集合データ転送速度が十分に高速となるような、 ソフトウェア RAID5 アレイ内に非常に多数のメンバーデバイスがある場合にはこのボトルネックが問題となってくる可能性があります。

レベル 4 と同様、 レベル 5 のパフォーマンスも非対称となり、 読み込みの方が書き込みより大幅にパフォーマンスが高くなります。 RAID レベル 5 のストレージ容量はレベル 4 と同じです。

**レベル 6**

データの冗長性および維持がパフォーマンスより重要となると共に、 レベル 1 での領域使用に関する非効率性は認められないような場合、 一般的に採用される RAID レベルになります。 レベル 6 ではアレイ内の 2 台のドライブの損失からの復元が可能となる複合パリティスキームを使用し

ています。 この複合パリティスキームによりソフトウェア RAID デバイスにはかなり大きな負荷が CPU にかかることとなる他、 書き込みのトランザクション時の負荷も大きくなります。 このため、 レベル 6 はレベル 4 および 5 よりさらに非対称なパフォーマンスを見せます。

RAID レベル 6 アレイの合計容量は、 RAID レベル 5 および 4 の計算方法と同じですが、 デバイス数から (1 ではなく) 追加パリティストレージ領域用に2を引きます。

**レベル 10**

この RAID レベルではレベル 0 のパフォーマンス性とレベル 1 の冗長性の両方を取り入れることを目的としています。 また、 デバイスが 2 つ以上あるレベル 1 のアレイでは無駄になる領域を低減します。 レベル 10 では、 3 台のドライブアレイで格納するデータのコピーは 2 つのみとなるよう設定することが可能です。 これによりアレイの全体サイズを最小サイズのデバイスと同じサイズ (3 台のドライブのレベル 1 アレイと同様) ではなく、 最小サイズの 1.5 倍のサイズにすることができるようになります。

レベル 10 アレイを作成する場合、 使用可能なオプションが数多くあるためインストール時に作成するのは実用的とは言えません。 コマンドラインツールの **mdadm** を使用すると手作業で作成することができます。 オプションの詳細およびパフォーマンスに関するトレードオフについては **man md** を参照してください。

**リニア RAID**

より大きな仮想ドライブを作成するために複数ドライブをグループ化するのがリニア RAID です。 リニア RAID では、 一つのメンバードライブから順次チャンクを割り当て、 ドライブが完全に満杯になってから次のドライブに移動します。 メンバードライブ間での I/O 動作が分割される可能性はないため、 グループ化によってはパフォーマンスの向上は見られません。 また、 リニア RAID では冗長性も得られません。 メンバードライブのひとつに障害が発生した場合はアレイ全体が使用できないため、 実際には信頼性についても低下します。 容量は全メンバーディスクの合計になります。

## 18.5. Linux RAID サブシステム

Linux の RAID は以下のサブシステムから構成されます。

### Linux ハードウェア RAID のコントローラードライバー

Linux ではハードウェア RAID コントローラーに特定の RAID サブシステムはありません。 特殊な RAID チップセットを使用するため、 ハードウェア RAID のコントローラーにはそれ自体のドライブが同梱されています。 このドライバーによりシステムは RAID セットを通常のディスクとして検出することができるようになります。

### mdraid

**mdraid** サブシステムは Linux 向けのソフトウェア RAID ソリューションとして設計され、 また Linux 環境のソフトウェア RAID に適したソリューションとなります。 このサブシステムでは独自のメタデータ形式を使用します。 一般的にはネイティブの **mdraid** メタデータと呼ばれます。

**mdraid** では外部のメタデータとして知られる他のメタデータ形式にも対応しています。Red Hat Enterprise Linux 7 では外部のメタデータで **mdraid** を使用し ISW / IMSM (Intel のファームウェア RAID) セットにアクセスします。**mdraid** セットは **mdadm** ユーティリティーで設定および制御を行います。

### dmraid

*Device-mapper RAID* や **dmraid** はディスクをひとつの RAID セットにまとめるメカニズムを提供するデバイスマッパーのカーネルコードを参照します。 同じこのカーネルでは RAID 設定のメカニズムは提供し

ていません。

**dmraid** は完全にユーザー領域で設定され、各種のオンディスクメタデータ形式への対応を容易にしています。このため、**dmraid** は幅広いファームウェア RAID 実装で使用されています。また、**dmraid** は Intel のファームウェア RAID にも対応しますが、Red Hat Enterprise Linux 7 では **mdraid** を使って Intel ファームウェア RAID セットにアクセスします。

## 18.6. インストーラーでの RAID サポート

システム上のハードウェアおよびファームウェア RAID セットはすべて **Anaconda** インストーラーによって自動的に検出され、インストールが行えるようになります。また、**Anaconda** は **mdraid** を使用してソフトウェア RAID に対応しているため、既存の **mdraid** セットを認識することができます。

**Anaconda** ではインストール時に RAID セットを作成するユーティリティーを提供しています。ただし、このユーティリティーでは新しいセットのメンバーにできるのはパーティションのみになります (ディスク全体とは対照的)。セットにディスク全体を使用するには、ディスク全体にまたがる 1 つのパーティションを作成し、そのパーティションを RAID セットのメンバーとして使用します。

RAID セットが root ファイルシステムによって使用される場合、**Anaconda** により特殊なカーネルコマンドラインオプションがブートローダーの設定に渡され、root ファイルシステムを検索する前に RAID セットをアクティブにするよう **initrd** に指示します。

インストール時に RAID を設定する方法については Red Hat Enterprise Linux 7 の『インストールガイド』を参照してください。

## 18.7. RAID セットを設定する

一般的には、ほとんどの RAID セットがその作成時にファームウェアメニューやインストーラーを使って設定されます。システムのインストール後、できればマシンを再起動したりファームウェアメニューに入らずに RAID セットの作成や変更を行う必要が生じることがあります。

RAID セットを簡単に設定したり、ディスクの追加後でも新たなセットを定義したりすることができるハードウェア RAID コントローラーがあります。これらのコントローラーには標準の API がないためドライバー固有のユーティリティーを使用する必要があります。詳細についてはご使用のハードウェア RAID コントローラーの説明書などを参照してください。

### mdadm

Linux では **mdadm** コマンドラインツールを使ってソフトウェア RAID の管理を行います (**mdraid**)。**mdadm** の各種のモードおよびオプションについては **man mdadm** を参照してください。**man** にはソフトウェア RAID アレイの作成や監視、組み立てなど一般的な作業についても役に立つ事例が記載されています。

### dmraid

その名の通り **dmraid** はデバイスマッパー RAID セットの管理に使用されます。**dmraid** ツールは各種の形式に対応している複数のメタデータ形式のハンドラを使用して ATARAID デバイスの検索を行います。対応している形式の一覧を表示させるには、**dmraid -l** を実行します。

「Linux RAID サブシステム」で説明している通り、RAID セットを一旦作成するとその後は **dmraid** ツールによる設定を行うことはできません。**dmraid** の使い方については **man dmraid** を参照してください。

## 18.8. 高度な RAID デバイスの作成

インストール完了後では作成できないアレイ上にオペレーティングシステムをインストールしたい場合があります。 一般的には **/boot** や root ファイルシステムを複雑な RAID デバイス上にセットアップする場合などです。 このような場合、 **Anaconda** ではサポートしていないアレイオプションを使わなければならない場合があります。 これを回避する策として次の手順を行います。

**手順18.1 高度な RAID デバイスの作成**

1. 通常通りにインストールディスクを挿入します。
2. 最初に起動した時点で、 **インストール**や **アップグレード** ではなく**レスキューモード** を選択します。 **レスキューモード** でシステムが完全に起動すると、 コマンドラインターミナルが表示されます。
3. このターミナルで **parted** を使用し、RAID パーティションを目的のハードドライブ上に作成します。次に、**mdadm** を使用し、使用できるすべての設定およびオプションを使ってこれらのパーティションから RAID アレイを手作業で作成します。実行方法の詳細については、12章パーティション、**man parted**、および **man mdadm** を参照してください。
4. アレイを作成したら、オプションでアレイ上にファイルシステムを作成することもできます。
5. コンピューターを再起動して、今度は **インストール**か **アップグレード** を選択し通常通りにインストールを行います。 **Anaconda** によってシステム内のディスクが検索され、 すでに存在している RAID デバイスが検出されます。
6. システム内のディスクの使い方に関しては、**カスタムレイアウト** を選択して **次へ** をクリックします。デバイスの一覧にすでに存在している MD RAID デバイス群が表示されます。
7. RAID デバイスを選択し、 **編集** をクリックしてそのマウントポイントと (オプションで) 使用するファイルシステムのタイプを設定し **完了** をクリックします。**Anaconda** によりすでに存在しているこの RAID デバイスへのインストールが行われ、 **レスキューモード** で作成したときに選択したカスタムオプションが維持されます。

**注記**

制約のあるインストーラーの **レスキューモード** に **man** ページは含まれません。**man mdadm** および **man md** にはいずれもカスタム RAID アレイを作成する場合に役立つ情報が記載されているため、回避策を講じている間に必要となる場合があります。このような場合には、 **man** ページを表示させたマシンにアクセスをしておくか、**レスキューモード** で起動してカスタムアレイを作成する前に man ページを印刷しておくと便利です。

[2] ホットスワップ機能のシャーシを使用するとシステムの電源を落とさずにハードドライブを取り除くことができます。

[3] RAID レベル 1 では同じ情報がアレイ内の全ディスクに書き込まれることになるためデータの信頼性は高くなりますが、 レベル 5 などのパリティベースの RAID レベルに比べ領域使用の効率性は低くなります。 ただし、 この効率性性の低さによりパフォーマンスが向上されます。 パリティベースの RAID レベルではパリティを生成するためにかなりの CPU を消費します。 一方、 RAID レベル 1 では単純に同じデータを複数の RAID メンバーに複数回書き込むだけのため、 CPU のオーバーヘッドは非常に少なくなります。 RAID レベル 1 はソフトウェア RAID を採用しているマシン上ではパリティベースの RAID レベルより優れたパフォーマンスを見せます。 また、 マシンの CPU リソースには RAID アクティビティ以外の動作による負担が常にかかります。

[4] パリティ情報はアレイ内の残りのメンバーディスクのコンテンツに応じて計算されます。 この情報はアレイ内のいずれかのディスクに障害が発生した場合に行われるデータの再構成に使用されます。 再構成されたデータは、 置換される前に障害が発生したディスクへの I/O 要求に応えるため使用され、 また置換後に障害が発生したディスクへの移植にも使用されます。

# 第19章 mount コマンドの使い方

Linux や UNIX、また同様のオペレーティングシステムでは、パーティションやリムーバブルデバイス (CD、 DVD、 USB フラッシュドライブなど) 上にあるファイルシステムをディレクトリーツリー内の特定のポイント (*マウントポイント*) に接続したり取り外したりすることができます。ファイルシステムの接続や取り外しを行う場合は、**mount** コマンドと **umount** コマンドをそれぞれ使用します。本章では、これらのコマンドの基本的な使い方や、マウントポイントの移動や共有サブツリーの作成などの高度なテクニックについてもいくつか扱います。

## 19.1. 現在マウントしているファイルシステムを表示させる

現在接続している全ファイルシステムを表示させる場合は、 **mount** コマンドを実行します。いずれの引数も付けません。

```
mount
```

上記のコマンドで既知のマウントポイントの一覧が表示されます。行ごとにデバイス名、ファイルシステムのタイプ、マウントしているディレクトリー、およびマウントオプションなどの情報が以下のような形で表示されます。

*device* on *directory* type *type* (*options*)

Red Hat Enterprise Linux 6.1 からは **findmnt** ユーティリティーも使用できるようになりました。このユーティリティーを使うとマウントしているファイルシステムをツリー形式で表示させることができます。現在接続している全ファイルシステムを表示させるには、**findmnt** コマンドを実行します。いずれの引数も付けません。

```
findmnt
```

### 19.1.1. ファイルシステムのタイプを指定する

**mount** コマンドの出力には、デフォルトで**sysfs** や **tmpfs** など各種の仮想ファイルシステムが含まれます。特定のファイルシステムタイプのデバイスのみを表示するには、コマンドラインで **-t** オプションを指定します。

```
mount -t type
```

**findmnt** コマンドを使用して、特定のファイルシステムタイプのデバイスを表示させる場合も同様です。

```
findmnt -t type
```

一般的なファイルシステムタイプの一覧については 表19.1「一般的なファイルシステムのタイプ」 をご覧ください。使用例については 例19.1「現在マウントしている**ext4** ファイルシステムを表示させる」 を参照してください。

> **例19.1 現在マウントしている ext4 ファイルシステムを表示させる**
>
> 通常、**/** パーティションと **/boot** パーティションはいずれも **ext4** を使用するようフォーマットされます。このファイルシステムを使用しているマウントポイントだけを表示する場合は、以下をシェルプロンプトに入力します。

```
~]$ mount -t ext4
/dev/sda2 on / type ext4 (rw)
/dev/sda1 on /boot type ext4 (rw)
```

**findmnt** コマンド使用してマウントポイントを一覧表示するには、以下を入力します。

```
~]$ findmnt -t ext4
TARGET SOURCE    FSTYPE OPTIONS
/      /dev/sda2 ext4   rw,realtime,seclabel,barrier=1,data=ordered
/boot  /dev/sda1 ext4   rw,realtime,seclabel,barrier=1,data=ordered
```

## 19.2. ファイルシステムをマウントする

特定のファイルシステムを接続するには、以下のような形式で **mount** コマンドを使用します。

```
mount [option…] device directory
```

*device* は *ブロックデバイス* への完全パス (「/dev/sda3」 など)、*普遍的な固有識別子* (UUID; 「UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb」 など)、または *ボリュームラベル* (「LABEL=home」 など) のいずれかで指定することができます。ファイルシステムをマウントしている間は *directory* の元のコンテンツにはアクセスできないことに注意してください。

**重要**

Linux では、すでにファイルシステムが接続されているディレクトリーに対してファイルシステムをマウントする動作が阻止されることはありません。特定のディレクトリーがマウントポイントとして使用されているかどうかを確認するには、そのディレクトリーを引数として **findmnt**ユーティリティーを実行し、終了コードを確認します。

```
findmnt directory; echo $?
```

ディレクトリーにいずれのファイルシステムも接続されていない場合は **1** を返します。

**mount** コマンドに必要なすべての情報 (つまりデバイス名、目的のディレクトリー、ファイルシステムタイプなどの情報) を全く指定せずに実行すると、**/etc/fstab** 設定ファイルの内容を読み込み、特定のファイルシステムの記載がないか確認します。このファイルには、選択したファイルシステムをマウントするデバイス名やディレクトリーの一覧が記載されているほか、ファイルシステムのタイプやマウントポイントも記載されています。このため、このファイルで指定されているファイルシステムをマウントする場合は次のコマンドのいずれかを使用できます。

```
mount [option…] directory
mount [option…] device
```

**root** でコマンドを実行しない限り、ファイルシステムのマウントには権限が必要であることに注意してください (「マウントオプションを指定する」 を参照)。

**注記**

特定のデバイスの UUID やラベル (デバイスがラベルを使用している場合) を確認するには、次のようにして **blkid** コマンドを使用します。

```
blkid device
```

たとえば、**/dev/sda3** の情報を表示させるには次のように入力します。

```
~]# blkid /dev/sda3
/dev/sda3: LABEL="home" UUID="34795a28-ca6d-4fd8-a347-73671d0c19cb"
TYPE="ext3"
```

### 19.2.1. ファイルシステムのタイプを指定する

ほとんどの場合、**mount** によって自動的にファイルシステムが検出されます。ただし、**NFS** (Network File System) や **CIFS** (Common Internet File System) などの認識できないファイルシステムがあるため、こうしたファイルシステムの場合は手作業で指定しなければなりません。ファイルシステムのタイプを指定するには次のように **mount** コマンドを使用します。

```
mount -t type device directory
```

表19.1「一般的なファイルシステムのタイプ」は、**mount** コマンドで使用できる一般的なファイルシステムのタイプの一覧を提供します。利用可能なファイルシステムのタイプについての詳細の一覧については「man ページ」に記載のそれぞれの man ページをご覧ください。

**表19.1 一般的なファイルシステムのタイプ**

| タイプ | 詳細 |
|---|---|
| **ext2** | **ext2** ファイルシステム |
| **ext3** | **ext3** ファイルシステム |
| **ext4** | **ext4** ファイルシステム |
| **btrfs** | **btrfs** ファイルシステム |
| **xfs** | **xfs** ファイルシステム |
| **iso9660** | **ISO 9660** ファイルシステム、通常は CD などの光学メディアで使用されます。 |
| **jfs** | **JFS** ファイルシステムは IBM によって開発されました。 |
| **nfs** | **NFS** ファイルシステム、ネットワーク経由でファイルにアクセスする場合に一般的に使用されます。 |
| **nfs4** | **NFSv4** ファイルシステム、ネットワーク経由でファイルにアクセスする場合に一般的に使用されます。 |
| **ntfs** | **NTFS** ファイルシステム、 Windows オペレーティングシステムを稼動しているマシンで一般的に使用されます。 |
| **udf** | **UDF** ファイルシステム、DVD などの光学メディアで一般的に使用されます。 |
| **vfat** | **FAT** ファイルシステム、Windows オペレーティングシステムを稼動しているマシンや特定のデジタルメディア (USB フラッシュドライブ、フロッピーディスクなど) 上で一般的に使用されます。 |

使用例については 例19.2「USB フラッシュドライブをマウントする」を参照してください。

**例19.2 USB フラッシュドライブをマウントする**

旧式の USB フラッシュドライブは FAT ファイルシステムを使用していることがよくあります。このようなドライブが **/dev/sdc1** デバイスを使用しているとします。また**/media/flashdisk/** というディレクトリーが存在すると仮定します。このデバイスを **/media/flashdisk/** ディレクトリーにマウントするには、 **root** で次のようにシェルプロンプトに入力します。

```
~]# mount -t vfat /dev/sdc1 /media/flashdisk
```

## 19.2.2. マウントオプションを指定する

マウントの追加オプションを指定する場合は、次のような形式のコマンドを使用します。

```
mount -o options device directory
```

複数のオプションを使う場合は、コンマの後に空白を入れないようにしてください。空白を入れてしまうと、**mount** は空白の後の値を追加のパラメーターとして解釈してしまいます。

一般的なマウントオプションの一覧を 表19.2「一般的なマウントオプション」 に示します。使用できる全オプションの一覧については 「man ページ」 セクションに記載している該当の man ページをご覧ください。

**表19.2 一般的なマウントオプション**

| オプション | 詳細 |
|---|---|
| **async** | ファイルシステム上での非同期の入/出力を許可します。 |
| **auto** | **mount -a** コマンドを使ったファイルシステムの自動マウントを許可します。 |
| **defaults** | **async,auto,dev,exec,nouser,rw,suid** のエイリアスを指定します。 |
| **exec** | 特定のファイルシステムでのバイナリーファイルの実行を許可します。 |
| **loop** | イメージをループデバイスとしてマウントします。 |
| **noauto** | **mount -a** コマンドを使ったファイルシステムの自動マウントをデフォルトの動作として拒否します。 |
| **noexec** | 特定のファイルシステムでのバイナリーファイルの実行を拒否します。 |
| **nouser** | 普通のユーザー (つまり **root** 以外のユーザー) によるファイルシステムのマウントおよびアンマウントを拒否します。 |
| **remount** | ファイルシステムがすでにマウントされている場合は再度マウントを行います。 |
| **ro** | 読み取り専用でファイルシステムをマウントします。 |
| **rw** | ファイルシステムを読み取りと書き込み両方でマウントします。 |
| **user** | 普通のユーザー (つまり **root** 以外のユーザー) によるファイルシステムのマウントおよびアンマウントを許可します。 |

使用例については 例19.3「ISO イメージをマウントする」 を参照してください。

**例19.3 ISO イメージをマウントする**

ISO イメージ (または一般的にはディスクイメージ) はループデバイスを使用することでマウントすることができます。Fedora 14 インストールディスクの ISO イメージが現在の作業ディレクトリーにあると仮定します。また、**/media/cdrom/** と言うディレクトリーが存在するとします。このイメージを **/media/cdrom/** ディレクトリーにマウントするには**root** で次のコマンドを実行します。

```
~]# mount -o ro,loop Fedora-14-x86_64-Live-Desktop.iso /media/cdrom
```

> ISO9660 は設計上、読み取り専用のファイルシステムになっていることに注意してください。

### 19.2.3. マウントポイントを共有する

システム管理作業の中には、同じファイルシステムにディレクトリーツリー内の複数の場所からのアクセスする必要がある場合があります (chroot 環境を準備する場合など)。Linux では同じファイルシステムを複数のディレクトリーに必要なだけマウントすることが可能です。また、**mount** コマンドは重複したマウントポイントを持たせることができる **--bind** オプションを実装します。以下のように使用します。

```
mount --bind old_directory new_directory
```

上記のコマンドにより、ユーザーはいずれの場所からでもファイルシステムにアクセスできるようになりますが、これは元のディレクトリー内にマウントされているファイルシステムには適用されません。これらのマウントも含めるには、次を実行します。

```
mount --rbind old_directory new_directory
```

さらに Red Hat Enterprise Linux 7 では、可能な限り柔軟性を持たせるために、*共有サブツリー*と呼ばれる機能を実装しています。次の 4 種類のマウントを使用することができます。

**共有マウント**

共有マウントにより、任意のマウントポイントと全く同一の複製マウントポイントを作成することができます。マウントポイントを共有マウントしてマークすると、元のマウントポイント内のあらゆるマウントが反映されます。マウントポイントのタイプを共有マウントに変更するには、シェルプロンプトで以下を入力します。

```
mount --make-shared mount_point
```

代わりに、選択したマウントポイントとその下にあるすべてのマウントポイントのマウントタイプを変更するには、以下を入力します。

```
mount --make-rshared mount_point
```

使用例については、例19.4「共有マウントポイントを作成する」を参照してください。

**例19.4 共有マウントポイントを作成する**

他のファイルシステムがマウントされる一般的な場所が 2 箇所あります。リムーバルメディア用の **/media** ディレクトリーと一時的にファイルシステムをマウントする場合の**/mnt** 所です。共有マウントを使用することで、これら 2 種類のディレクトリーが同じコンテンツを共有できるようになります。これを実行するには、**root** になり、**/media** ディレクトリーを「shared」としてマークします。

```
~]# mount --bind /media /media
~]# mount --make-shared /media
```

次に、以下のコマンドを使用して、複製を **/mnt** ディレクトリーに作成します。

```
~]# mount --bind /media /mnt
```

これで **/media** 内のマウントが **/mnt** 内にも表示されることが確認できます。たとえば、CD ROM ドライブに何らかのコンテンツを持つメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は次のコマンドを実行します。

```
~]# mount /dev/cdrom /media/cdrom
~]# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
~]# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

同様に、**/mnt** ディレクトリー内にマウントしているファイルシステムが**/media** 内に反映されていることを確認できます。たとえば、**/mnt/flashdisk/** というディレクトリーが存在し、また何らかのコンテンツを持つ USB フラッシュドライブが **/dev/sdc1** デバイスを使用するとした場合、この USB をプラグインしてから次を入力します。

```
~]# mount /dev/sdc1 /mnt/flashdisk
~]# ls /media/flashdisk
en-US  publican.cfg
~]# ls /mnt/flashdisk
en-US  publican.cfg
```

### スレーブマウント

スレーブマウントにより、所定のマウントポイントの複製を作成する際に制限を課すことができます。マウントポイントをスレーブマウントとしてマークすると、元のマウントポイント内のすべてのマウントがそれに反映されますが、スレーブマウント内のマウントはオリジナルには反映されません。マウントポイントのタイプをスレーブマウントに変更するには、シェルプロンプトで次を入力します。

```
mount --make-slave mount_point
```

選択したマウントポイントとその下にあるすべてのマウントポイントのマウントタイプを変更することも可能です。次のように入力します。

```
mount --make-rslave mount_point
```

使用例については 例19.5「スレーブのマウントポイントを作成する」を参照してください。

**例19.5 スレーブのマウントポイントを作成する**

**/media** ディレクトリーの内容が **/mnt** ディレクトリーでも表示されるようにしながら、**/mnt** ディレクトリー内のマウントは**/media** ディレクトリーには反映させない方法を以下に示します。**root** になり、まず **/media** ディレクトリーに「shared」 のマークを付けます。

```
~]# mount --bind /media /media
~]# mount --make-shared /media
```

次に **/media** ディレクトリーの複製を **/mnt** ディレクトリーに作成して、今度は「slave」のマークを付けます。

```
~]# mount --bind /media /mnt
~]# mount --make-slave /mnt
```

**/media** 内のマウントが **/mnt** でも表示されるかを確認します。たとえば、CD-ROM ドライブに何らかの内容を持つメディアがあり、**/media/cdrom/** というディレクトリーが存在するとします。次のコマンドを実行します。

```
~]# mount /dev/cdrom /media/cdrom
~]# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
~]# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

また、**/mnt** ディレクトリー内にマウントされているファイルシステムが**/media** に反映されていることを確認します。たとえば、**/dev/sdc1** デバイスを使用する何らかのコンテンツを含む USB フラッシュドライブをプラグインしており、かつ **/mnt/flashdisk/** ディレクトリーが存在している場合に以下を入力します。

```
~]# mount /dev/sdc1 /mnt/flashdisk
~]# ls /media/flashdisk
~]# ls /mnt/flashdisk
en-US  publican.cfg
```

### プライベートマウント

プライベートマウントはマウントのデフォルトタイプであり、共有マウントやスレーブマウントと異なり、伝播イベントの受信や転送は一切行いません。マウントポイントを明示的にプライベートマウントにするには、シェルプロンプトで以下を入力します。

```
mount --make-private mount_point
```

または、選択したマウントポイントとその下にあるすべてのマウントポイントを変更することもできます。

```
mount --make-rprivate mount_point
```

使用例については 例19.6「プライベートマウントポイントを作成する」を参照してください。

**例19.6 プライベートマウントポイントを作成する**

例19.4「共有マウントポイントを作成する」の状況を考慮に入れ、共有マウントポイントが次のコマンドを使って **root** で以前に作成されていると仮定します。

```
~]# mount --bind /media /media
~]# mount --make-shared /media
~]# mount --bind /media /mnt
```

**/mnt** ディレクトリーに「private」のマークを付けるには次のように入力します。

```
~]# mount --make-private /mnt
```

これで **/media** 内のマウントはいずれも **/mnt** 内では表示されないことを確認できるようになります。たとえば、CD-ROM デバイスに何らかのコンテンツを含むメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合に、次のコマンドを実行します。

```
~]# mount /dev/cdrom /media/cdrom
```

```
~]# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
~]# ls /mnt/cdrom
~]#
```

また、**/mnt** ディレクトリー内にマウントしているファイルシステムは**/media** ディレクトリーには反映されないことを確認することもできます。たとえば、**/dev/sdc1** デバイスを使用し、何らかのコンテンツを含む USB フラッシュドライブをプラグインして、**/mnt/flashdisk/** ディレクトリーが存在する場合に以下を入力します。

```
~]# mount /dev/sdc1 /mnt/flashdisk
~]# ls /media/flashdisk
~]# ls /mnt/flashdisk
en-US  publican.cfg
```

**バインド不能のマウント**

任意のマウントポイントに対して一切複製が行われないようにするには、バインド不能のマウントを使用します。マウントポイントのタイプをバインド不能のマウントに変更するには、次のようにシェルプロンプトに入力します。

```
mount --make-unbindable mount_point
```

または、選択したマウントポイントとその下にあるすべてのマウントポイントを変更することもできます。

```
mount --make-runbindable mount_point
```

使用例については 例19.7「バインド不能のマウントポイントを作成する」を参照してください。

**例19.7 バインド不能のマウントポイントを作成する**

**/media** ディレクトリーが共有されないようにする場合は、**root** として、シェルプロンプトに以下を入力します。

```
~]# mount --bind /media /media
~]# mount --make-unbindable /media
```

これにより、これ以降にこのマウントの複製を作成しようとするとエラーが出て失敗します。

```
~]# mount --bind /media /mnt
mount: wrong fs type, bad option, bad superblock on /media,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail  or so
```

### 19.2.4. マウントポイントを移動する

ファイルシステムがマウントされているディレクトリーを変更するには、次のコマンドを使用します。

```
mount --move old_directory new_directory
```

使用例については 例19.8「既存の NFS マウントポイントを移動する」 を参照してください。

**例19.8 既存の NFS マウントポイントを移動する**

NFS ストレージにはユーザーのディレクトリーが含まれ、すでに **/mnt/userdirs/** にマウントされています。**root** として、次のコマンドを使用してこのマウントポイントを**/home** に移動します。

```
~]# mount --move /mnt/userdirs /home
```

マウントポイントが正しく移動したことを確認するため、両方のディレクトリーのコンテンツを表示させます。

```
~]# ls /mnt/userdirs
~]# ls /home
jill  joe
```

## 19.3. ファイルシステムをアンマウントする

以前にマウントしていたファイルシステムを切り離す場合、以下のいずれかの **umount** コマンドを使用します。

```
umount directory
umount device
```

ファイルシステムのアンマウントを **root** でログインしている間に行わない場合は、アンマウントに適切な権限が必要です (「マウントオプションを指定する」 を参照)。 使用例については 例19.9「CD をアンマウントする」 を参照してください。

**重要**

ファイルシステムを使用中に (このファイルシステム上でプロセスが読み取りを行っている場合や、カーネルによって使用中の場合など)、**umount** コマンドを実行するとエラーを出して失敗します。次のように **fuser** コマンドを使ってファイルシステムにアクセスしているプロセスを判別します。

```
fuser -m directory
```

**/media/cdrom/** ディレクトリーにマウントしているファイルシステムにアクセスしているプロセスを表示させる場合は、以下を入力します。

```
~]$ fuser -m /media/cdrom
/media/cdrom:         1793  2013  2022  2435 10532c 10672c
```

**例19.9 CD をアンマウントする**

**/media/cdrom/** ディレクトリーに以前にマウントしていた CD をアンマウントする場合は、シェルプロンプトで以下を入力します。

```
~]$ umount /media/cdrom
```

## 19.4. ドキュメンテーション

コマンドなどの詳細については、以下のドキュメントをご覧ください。

### 19.4.1. man ページ

- **man 8 mount** — **mount** コマンドの man ページです。使い方などに関する詳細が記載されています。
- **man 8 umount** — **umount** コマンドの man ページです。使い方などに関する詳細が記載されています。
- **man 8 findmnt** — **findmnt** コマンドの man ページです。使い方などに関する詳細が記載されています。
- **man 5 fstab** — **/etc/fstab** ファイル形式に関する詳細が記載されている man ページです。

### 19.4.2. 役に立つ Web サイト

- 『Shared subtrees』 — 共有サブツリーの概念について解説されている LWN の記事です。

# 第20章 volume_key 機能

volume_key 機能では libvolume_key と **volume_key** の 2 種類のツールを提供しています。libvolume_key はストレージボリュームの暗号キーを操作したりボリュームとは別に格納したりするためのライブラリになります。 **volume_key** は暗号化されたハードドライブへのアクセスを取り戻すためにキーとパスフレーズを抽出する関連コマンドラインツールになります。

第一ユーザーがキーやパスワードを忘れてしまった、 ユーザーが突然退職してしまった、 ハードウェアまたはソフトウェアの障害で暗号化していたボリュームのヘッダーが破損したためデータを抽出する必要がある、 などといった場合にこの機能は便利です。 企業などの場合、 エンドユーザーにコンピュータを手渡す前に IT ヘルプデスクによって **volume_key** を使用した暗号キーのバックアップをとっておくことが可能です。

現在、 **volume_key** で対応しているのは LUKS ボリュームの暗号形式のみです。

**注記**

**volume_key** は Red Hat Enterprise Linux 7 サーバーの標準インストールには含まれません。**volume_key** のインストールについては、http://fedoraproject.org/wiki/Disk_encryption_key_escrow_use_casesを参照してください。

## 20.1. コマンド

**volume_key** の形式は次のようになります。

```
volume_key [OPTION]... OPERAND
```

**volume_key** の動作のモードとオペランド (演算対象) は以下のいずれかのオプションを指定して確定します。

**--save**

このコマンドはオペランド *volume* [*packet*] を予期します。*packet* を指定すると、**volume_key** はその *packet* からキーとパスフレーズを抽出します。*packet* を指定しない場合は、*volume* からキーとパフフレーズを抽出します。必要に応じてユーザー入力を求めます。キーとパスフレーズは 1 つまたは複数の出力パケットに格納されます。

**--restore**

このコマンドはオペランド *volume packet* を予期します。*volume* を開き、*packet* 内にあるキーとパスフレーズを使って *volume* に再びアクセスできるようにします。新しいパスフレーズの入力など、必要に応じてユーザー入力を求めます。

**--setup-volume**

このコマンドはオペランド *volume packet name* を予期します。*volume* を開き、*packet* 内のキーとパスフレーズを使って *volume* を *name* という名前に設定して解読したデータ用に使用できるようにします。

*Name* は dm-crypt ボリュームの名前です。この操作により解読したボリュームは **/dev/mapper/*name*** として使用できるようになります。

新しいパスフレーズを追加するなど、このコマンドでの操作は *volume* を永続的に変更することはありません。ユーザーは解読されたボリュームにアクセスして変更を行うことができ、処理中に *volume* を変更することができます。

**--reencrypt、 --secrets、 --dump**

この 3 種類のコマンドは同じような機能ですが出力方法が異なります。それぞれにオペランド *packet* が必要です。各コマンドは*packet* を開いて必要に応じて解読します。**--reencrypt** はその情報を 1 つまたは複数の新しい出力パケットに格納します。**--secrets** は *packet* に含まれているキーとパスフレーズを出力します。**--dump** は *packet* のコンテンツを出力しますが、キーとパスフレーズはデフォルトでは出力しません。これは **--with-secrets** をコマンドに追加することで変更できます。また、**--unencrypted** コマンドを使ってパケットの暗号化されていない部分だけをダンプすることも可能です。これには、パスフレーズやプライベートキーは必要ありません。

上記のコマンドのそれぞれには、次のオプションを付けることができます。

**-o、--output *packet***

このコマンドでデフォルトのキーやパスフレーズを *packet* に書き込みます。デフォルトのキーまたはパスフレーズはボリューム形式によって異なります。期限切れにならないようものを選択してください。また、**--restore** を使ってボリュームへのアクセスを取り戻すことができることを確認します。

**--output-format *format***

このコマンドはすべての出力パケットに対して指定した *format* を使用します。現在使用できる *format* は以下のいずれかになります。

- **asymmetric**: CMS を使ってパケット全体を暗号化します。証明書が必要です。
- **asymmetric_wrap_secret_only**: 機密情報またはキーとパスフレーズのみをラップします。証明書が必要です。
- **passphrase**: GPG を使ってパケット全体を暗号化します。パスフレーズが必要です。

**--create-random-passphrase *packet***

英数字のランダムなパスフレーズを生成して *volume* に追加 (他のパスフレーズには影響しません) した後に、このパスフレーズを *packet* に格納します。

## 20.2. volume_key を 1 ユーザーとして使用する

**volume_key** を 1 ユーザーとして使用して暗号キーを保存することができます。以下の手順を実行します。

**注記**

このファイル内の全サンプルで ***/path/to/volume*** は LUKS デバイスになり、その中に含まれるプレーンテキストデバイスにはなりません。**blkid -s *type* */path/to/volume*** を使用すると ***type*="crypto_LUKS"** が表示されるはずです。

**手順20.1 volume_key をスタンドアロンで使用する**

1. 次を実行します。

```
volume_key --save /path/to/volume -o escrow-packet
```

キーを保護するためのエスクローパケットのパスフレーズの入力を求めるプロンプトが表示されます。

2. 生成された **escrow-packet** ファイルを保存し、パスフレーズを忘れないようにしてください。

ボリュームのパスフレーズを忘れてしまった場合は、保存したエスクローパケットを使ってデータへのアクセスを復元します。

**手順20.2 エスクローパケットでデータへのアクセスを取り戻す**

1. **volume_key** を実行することができ、エスクローパケットが使用できる環境でシステムを起動します (レスキューモードなど)。
2. 次を実行します。

```
volume_key --restore /path/to/volume escrow-packet
```

エスクローパケットの作成時に使用したエスクローパケットのパスフレーズの入力を求めるプロンプト、次にボリュームの新しいパスワードの入力を求めるプロンプトが表示されます。

3. 選択したパスフレーズを使ってこのボリュームをマウントします。

暗号化したボリュームの LUKS ヘッダー内にあるパスフレーズスロットを解放するには、忘れてしまった古いパスフレーズを **cryptsetup luksKillSlot** コマンドを使って削除します。

## 20.3. 規模の大きな組織で volume_key を使用する

大規模な組織では、システム管理者の誰もが知っている単一のパスワードを使ってシステムごとに異なるパスワードを管理していくのは非現実的であるばかりでなく、セキュリティー上の危険も伴います。こうした状況に対応するため、**volume_key** では非対称暗号を使用します。これにより、コンピューター上の暗号化されたデータへのアクセスに必要なパスワードを知り得る人の人数を最小限に抑えることができます。

本セクションでは、暗号キーを保存する前の準備として必要な手順や、暗号キーを保存する方法、ボリュームへのアクセスを取り戻す方法、および緊急時のパスフレーズを設定する方法について説明します。

### 20.3.1. 暗号キーを保存するための準備

暗号キーを保存する前に、準備しておく必要のあることがいくつかあります。

**手順20.3 準備**

1. X509 証明書とプライベートキーのペアを作成します。
2. プライベートキーを他人に漏らしたりしない信頼できるユーザーを指定します。これらのユーザーはエスクローパケットを解読できるようになります。
3. エスクローパケットの解読に使用するシステムを選択します。これらのシステムでプライベートキーを含む NSS データベースのセットアップを行います。

   プライベートキーが NSS データベースに作成されていない場合は、次の手順に従います。

   A. 証明書とプライベートキーを **PKCS#12** ファイルに保存します。

   B. 次を実行します。

```
certutil -d /the/nss/directory -N
```

これで NSS データベースのパスワードを選択できるようになりました。各 NSS データベースには別々のパスワードを持たせることができるため、指定したユーザーがそれぞれ別々の NSS データベースを使用する場合はユーザー間で 1 つのパスワードを共有する必要はありません。

C. 次を実行します。

```
pk12util -d /the/nss/directory -i the-pkcs12-file
```

4. システムをインストールしているユーザーか、または既存のシステムにキーを保存しているユーザー全員に証明書を配信します。

5. 保存したプライベートキー用にマシンおよびボリュームからそのキーの検索が可能なストレージを用意します。たとえば、マシン 1 台に対して 1 つのサブディレクトリーを持つ単純なディレクトリーでも、他のシステム管理タスクにも使用されるデータベースであっても構いません。

### 20.3.2. 暗号キーを保存する

必要な準備が整ったら (「暗号キーを保存するための準備」 を参照)、次の手順で暗号キーを保存することができるようになります。

**注記**

このファイル内の全サンプルで ***/path/to/volume*** は LUKS デバイスになり、その中に含まれるプレーンテキストデバイスにはなりません。**blkid -s *type* */path/to/volume*** を使用すると ***type*****="crypto_LUKS"** が表示されるはずです。

**手順20.4 暗号キーを保存する**

1. 次を実行します。

```
volume_key --save /path/to/volume -c /path/to/cert escrow-packet
```

2. 生成した **escrow-packet** ファイルを準備したストレージに保存し、システムおよびボリュームに関連付けます。

この手順は手作業で行うこともできますが システムインストールの一部としてスクリプト化して実行することもできます。

### 20.3.3. ボリュームへのアクセスを取り戻す

暗号キーの保存 (「暗号キーを保存するための準備」 および 「暗号キーを保存する」 を参照) が完了したら、必要に応じてドライバーへのアクセスを取り戻すことができます。

**手順20.5 ボリュームへのアクセスを取り戻す**

1. パケットのストレージからそのボリュームのエスクローパケットを取得して、指定ユーザーの 1 人が解読できるようにそのエスクローパケットを送信します。

2. 指定ユーザーは次を実行します。

```
volume_key --reencrypt -d /the/nss/directory escrow-packet-in -o
escrow-packet-out
```

NSS データベースのパスワードを入力した後に、指定ユーザーは暗号化する **escrow-packet-out** のパスフレーズを選択します。パスフレーズは毎回異なるものになっても構いません。このパスフレーズは、暗号キーが指定ユーザーから目的のシステムに移動する間のみ暗号キーを保護します。

3. 指定ユーザーから **escrow-packet-out** ファイルとパスフレーズを受け取ります。
4. レスキューモードなど、**volume_key** の実行が可能で、**escrow-packet-out** ファイルが利用可能な環境で目的のシステムを起動します。
5. 次を実行します。

```
volume_key --restore /path/to/volume escrow-packet-out
```

指定ユーザーによって選択されたパケットのパスフレーズの入力と、ボリューム用の新しいパスフレーズの入力を求めるプロンプトが表示されます。

6. 選択したボリュームパスフレーズでボリュームをマウントします。

忘れてしまった古いパスフレーズは **cryptsetup luksKillSlot** で削除し、暗号化しているボリュームの LUKS ヘッダー内のパスフレーズスロットを解放することができます。これは、**cryptsetup luksKillSlot *device key-slot*** を使って実行します。詳細とサンプルについては、**cryptsetup --help** でご覧ください。

### 20.3.4. 緊急時のパスフレーズを設定する

状況によって (たとえば、出張中などの場合)、システム管理者が影響を受けたシステムを直接操作できないことがありますが、そのような場合でもユーザーはデータにアクセスしなければならないことがあります。その場合、**volume_key** をパスフレーズや暗号キーで動作させることができます。

システムのインストール時に次を実行します。

```
volume_key --save /path/to/volume -c /path/to/ert --create-random-
passphrase passphrase-packet
```

ランダムなパスフレーズが生成され、指定したボリュームに追加されてから ***passphrase-packet*** に保存されます。**--create-random-passphrase** と **-o** オプションを組み合わせるとパケットを同時に生成することが可能です。

ユーザーがパスワードを忘れてしまった場合、指定ユーザーは次を実行します。

```
volume_key --secrets -d /your/nss/directory passphrase-packet
```

ランダムなパスフレーズを表示します。このパスフレーズをエンドユーザーに渡します。

## 20.4. ドキュメント

**volume_key** にについてさらに詳しくは、以下を参照してください。

- **/usr/share/doc/volume_key-*/README** にある readme ファイル
- **volume_key** の man ページ (**man volume_key** で表示)

- オンラインのドキュメント (http://fedoraproject.org/wiki/Disk_encryption_key_escrow_use_cases)

# 第21章 アクセス制御リスト

ファイルとディレクトリーには、ファイルの所有者、そのファイルに関連したグループおよびシステムを使用する他のすべてのユーザーの権限セットが設定されます。しかし、これらの権限には制限があります。たとえば、異なるユーザーごとに別々の異なる権限を設定することはできません。そのため *アクセス制御リスト* (ACL) が実装されています。

Red Hat Enterprise Linux カーネルは ext3 ファイルシステムと NFS でエクスポートしたファイルシステムに対して ACL サポートを提供します。ACL は、Samba 経由でアクセスされる ext3 ファイルシステム上でも認識されます。

カーネルでのサポートと共に、**acl** パッケージが ACL の実装に必要になります。このパッケージには、ACL 情報の追加、修正、削除および取得のためのユーティリティーが収納されています。

**cp** コマンドと **mv** コマンドは、ファイルとディレクトリーに関連したすべての ACL のコピーまたは移動を実行します。

## 21.1. ファイルシステムのマウント

ファイルやディレクトリー用に ACL を使用する前に、そのファイルまたはディレクトリーのパーティションを ACL サポートでマウントする必要があります。ローカルの ext3 ファイルシステムの場合は、以下のコマンドでマウントすることができます。

**mount -t ext3 -o acl *device-name partition***

例:

**mount -t ext3 -o acl /dev/VolGroup00/LogVol02 /work**

別の方法として、パーティションが **/etc/fstab** ファイルにリストされている場合は、パーティションのエントリーに **acl** オプションを含むことができます。

```
LABEL=/work      /work       ext3    acl         1 2
```

ext3 ファイルシステムが Samba 経由でアクセスされ、ACL がそのアクセス用に有効になっている場合は、ACL は認識されます。これは、Samba が **--with-acl-support** オプションでコンパイルされているためです。Samba 共有のアクセス時またはマウント時に特別なフラグは必要ありません。

### 21.1.1. NFS

デフォルトでは、NFS サーバーでエクスポートされているファイルシステムが ACL をサポートしており、NFS クライアントが ACL を読み込める場合、ACL はクライアントシステムによって使用されます。

サーバーを設定する際に NFS 共有上の ACL を無効にするには、**/etc/exports** ファイル内に **no_acl** オプションを組み込みます。クライアントに NFS 共有をマウントする際に ACL を無効にするには、コマンドライン経由か、または **/etc/fstab** ファイルで **no_acl** オプションを使用してマウントします。

## 21.2. アクセス ACL の設定

ACL には、*アクセス ACL* と *デフォルト ACL* と 2 つのタイプがあります。アクセス ACL とは、特定のファイルまたはディレクトリー用のアクセス制御リストです。デフォルト ACL は、ディレクトリーにのみ関連付けられます。ディレクトリー内のファイルがアクセス ACL を持たない場合は、そのディレクトリーにはデフォルト ACL のルールが適用されます。デフォルト ACL はオプションです。

ACL は以下のように設定できます。

1. ユーザーごと
2. グループごと
3. 実効権 (effective rights) マスクの使用
4. ファイルのユーザーグループに属しないユーザーに対して

**setfacl** ユーティリティーは、ファイルとディレクトリー用の ACL を設定します。**-m** オプションを使用すると、ファイルまたはディレクトリーの ACL の追加または修正を実行できます。

```
# setfacl -m rules files
```

ルール (*rules*) は、以下の形式で指定しなければなりません。複数のルールがカンマで区切られている場合は、それらのルールを同じコマンドに指定することができます。

**u:*uid*:*perms***

ユーザー用のアクセス ACL を設定します。ユーザー名または UID を指定できます。ユーザーにはシステム上の任意の有効なユーザーを指定できます。

**g:*gid*:*perms***

グループ用のアクセス ACL を設定します。グループ名または GID を指定できます。グループにはシステム上の任意の有効なグループを指定できます。

**m:*perms***

実効権マスクを設定します。このマスクは所有グループのすべての権限とユーザーおよびグループのエントリーすべてを結合したものです。

**o:*perms***

ファイル所有グループ内のユーザー以外のユーザー用にアクセス ACL を設定します。

権限 (*perms*) は、読み込み、書き込みおよび実行を表す**r**、**w**、および **x** の文字の組み合わせで表示されます。

ファイルまたはディレクトリーにすでに ACL があり、**setfacl** コマンドが使用されている場合、追加のルールが既存の ACL に追加されるか、または既存のルールが修正されます。

**例21.1 読み込みと書き込みの権限を付与する**

たとえば、ユーザー「andrius」に読み込みと書き込みの権限を付与するには以下を実行します。

```
# setfacl -m u:andrius:rw /project/somefile
```

ユーザー、グループまたはその他からすべての権限を削除するには、**-x** オプションを使用して、いずれの権限も指定しないようにします。

```
# setfacl -x rules files
```

**例21.2 すべての権限を削除する**

たとえば、UID 500 のユーザーからすべての権限を削除するには以下を実行します。

```
# setfacl -x u:500 /project/somefile
```

## 21.3. デフォルト ACL の設定

デフォルト ACL を設定するには、ルールの前に **d:** をルールの前に追加してから、ファイル名の代わりにディレクトリーを指定します。

**例21.3 デフォルト ACL の設定**

たとえば、**/share/** ディレクトリーのデフォルト ACL を設定して、ユーザーグループに属さないユーザー用の読み込みと実行を設定するには以下を実行します (個別ファイルのアクセス ACL はこれを上書きできます)。

```
# setfacl -m d:o:rx /share
```

## 21.4. ACL の取り込み

ファイルまたはディレクトリー用の既存の ACL を判別するには、**getfacl** コマンドを使用します。以下の例では、**getfacl** がファイルの既存 ACL を判別するために使用されています。

**例21.4 ACL の取り込み**

```
# getfacl home/john/picture.png
```

上記のコマンドは次のような出力を返します。

```
# file: home/john/picture.png
# owner: john
# group: john
user::rw-
group::r--
other::r--
```

デフォルト ACL の設定されたディレクトリーが指定されている場合、デフォルト ACL も以下のように表示されます。たとえば、**getfacl home/sales/** により以下のような出力が表示されます。

```
# file: home/sales/
# owner: john
# group: john
user::rw-
user:barryg:r--
group::r--
mask::r--
other::r--
default:user::rwx
```

```
default:user:john:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

## 21.5. ACL を持つファイルシステムのアーカイブ作成

デフォルトでは、バックアップ操作時に **dump** コマンドが ACL を保存します。**tar** を使用してファイルまたはファイルシステムをアーカイブする場合は、**--acls** オプションを使用して ACL を保存します。同様に**cp** を使用して ACL を持つファイルをコピーするには、**--preserve=mode** オプションを入れて ACL が全体に確実にコピーされるようにします。さらに、**cp** の **-a** オプション (**-dR --preserve=all** と同等) もバックアップ時にタイムスタンプ、SELinux コンテキストなどの情報と一緒に ACL を保存します。**dump**、**tar**、または **cp** についてさらに詳しくは、それぞれの**man** ページを参照してください。

**star** ユーティリティーは、ファイルのアーカイブ生成に使用される点で**tar** ユーティリティーと似ています。しかし、一部のオプションは異なります。最も一般的に使用されるオプションの一覧については 表 21.1「**star** のコマンドラインオプション」 を参照してください。すべての利用可能なオプションについては、**man star** を参照してください。このユーティリティーを使用するには**star** パッケージが必要になります。

**表21.1 star のコマンドラインオプション**

| オプション | 詳細 |
|---|---|
| **-c** | アーカイブファイルを作成します |
| **-n** | ファイルを抽出しません。**-x** と併用すると、ファイルが行う抽出を表示します。 |
| **-r** | アーカイブ内のファイルを入れ替えます。ファイルはアーカイブファイルの末尾に書き込まれて、同じパスとファイル名を持つファイルを入れ替えます。 |
| **-t** | アーカイブファイルのコンテンツを表示します。 |
| **-u** | アーカイブファイルを更新します。ファイルは、アーカイブにファイルが存在しない場合や、アーカイブ内にある同名のファイルよりも新しい場合はアーカイブの末尾に書き込まれます。このオプションは、アーカイブがファイルか、またはバックスペース可能な非ブロックテープの場合にのみ機能します。 |
| **-x** | アーカイブからファイルを抽出します。**-U** との併用で、アーカイブ内のファイルがファイルシステム上の対応するファイルよりも古い場合、ファイルは抽出されません。 |
| **-help** | 最も重要なオプションを表示します。 |
| **-xhelp** | 最も重要ではないオプションを表示します。 |
| **-/** | アーカイブからファイルを抽出する際に、ファイル名から先頭のスラッシュを取り除きません。デフォルトでは、ファイルが抽出される際に先頭のスラッシュは取り除かれます。 |
| **-acl** | 作成または抽出時に、ファイルとディレクトリーに関連付けられているすべての ACL をアーカイブするか、または復元します。 |

## 21.6. 旧システムとの互換性

ACL が所定のファイルシステム上のいずれかのファイルに設定されている場合、そのファイルシステムは **ext_attr** 属性を持ちます。この属性は、以下のコマンドを使用することがあります。

```
# tune2fs -l filesystem-device
```

**ext_attr** の属性を取得したファイルシステムは、古いカーネルでマウントができますが、それらのカーネルは設定されている ACL のいずれも強制しません。

バージョン 1.22 以降の **e2fsprogs** パッケージ (Red Hat Enterprise Linux 2.1 および 4 内のバージョンも含む) に含まれている **e2fsck** ユーティリティーのバージョンは、**ext_attr** 属性を使用してファイルシステムをチェックすることができます。古いバージョンはこのチェックを拒否します。

## 21.7. 参照

より詳しい情報については以下の man ページを参照してください。

- **man acl** — ACL の説明
- **man getfacl** — ファイルアクセス制御リストの取得方法について説明しています
- **man setfacl** — ファイルアクセス制御リストの設定方法について説明しています
- **man star** — **star** ユーティリティーとその数多くのオプションについて詳しく説明しています

# 第22章 ソリッドステートディスクの導入ガイドライン

*ソリッドステートディスク*(SSD) とは永続的なデータの格納に NAND フラッシュチップを使用するストレージデバイスを指します。今までのディスクとは大きく異なり、データを回転する円盤状の磁気記憶媒体に格納します。SSD では、論理ブロックアドレス (LBA) 全体でのデータへのアクセス時間は一定になります。一方、回転媒体を使用するこれまでのディスクの場合、広範囲のアドレスにまたがるデータにアクセスするパターンでは時間がかかります。このように、SSD デバイスの方が待ち時間やスループットに優れています。

使用中のブロック数がディスクの最大容量に近づくにつれパフォーマンスが低下してきます。パフォーマンス低下の度合いはデバイスのベンダーごとに大きく異なりますが、いずれのデバイスにもある程度のパフォーマンス低下が見られます。

パフォーマンス低下の問題に対応するため、ホストシステム (Linux カーネルなど) は、特定のブロック範囲が使用されなくなっていることをストレージに知らせる discard 要求を使用できます。SSD はこの情報に基づいて領域を内部で解放し、解放した空きブロックをウェアレベリングに使用することができます。discard 要求が実行できるのは、そのストレージがストレージプロトコル (ATA または SCSI) に対応している場合のみです。discard の要求は、ストレージプロトコル固有のネゴシエート済みの discard コマンドによってストレージに対して実行されます (ATA の場合は **TRIM** コマンド、SCSI の場合は **WRITE SAME** (**UNMAP** を設定)、または **UNMAP** コマンドになります)。

**discard** サポートを有効にすると、以下の 2 点が「true」である場合に最も役立ちます。1 つ目として、ファイルシステムに使用可能な空き領域が依然としてある場合、2 つ目として、基礎となるストレージデバイスの論理ブロックのほとんどがすでに書き込まれている場合です。**TRIM** の詳細については、次のリンクにある 『Data Set Management T13 Specifications』 を参照してください。

http://t13.org/Documents/UploadedDocuments/docs2008/e07154r6-Data_Set_Management_Proposal_for_ATA-ACS2.doc

**UNMAP** の詳細については、次のリンクにある 『SCSI Block Commands 3 T10 Specification』 のセクション 4.7.3.4 を参照してください。

http://www.t10.org/cgi-bin/ac.pl?t=f&f=sbc3r26.pdf

**注記**

市場で販売されているソリッドステートのデバイスすべてに **discard** サポートがある訳ではありません。ソリッドステートのデバイスに **discard** サポートがあるかどうかを判別するには、**/sys/block/sda/queue/discard_granularity** を確認します。

## 22.1. 導入に関する考慮事項

SSD の操作および内部のレイアウト上、デバイスのパーティション設定は内部の *消去ブロックの境界* で行うのが最適です。SSD によりトポロジー情報がエクスポートされると、Red Hat Enterprise Linux 7 のパーティション設定ユーティリティーによって適切なデフォルト設定が選択されます。

ただし、デバイスがトポロジー情報を **エクスポートしない** 場合には、1 番目のパーティションを 1MB の境界で作成することをお勧めします。

さらに、**MD** (ソフトウェア RAID) は discard 要求に対応しないことに注意してください。これとは対照的に、論理ボリュームマネージャー (LVM) や LVM が使用する device-mapper (DM) ターゲットは discard 要求に対応しています。discard 要求に対応しない DM ターゲットは dm-snapshot、dm-crypt、dm-raid45 のみなります。Red Hat Enterprise Linux 6.1 では dm-mirror の discard 要求サポートが追加されました。

また、Red Hat では SSD 上でのソフトウェア RAID レベル 1、4、5、6 の使用は推奨していないことに注意してください。このような RAID レベルの初期化段階で、チェックサムの正常な動作確認のために RAID 管理ユーティリティー (**mdadm** など) によってストレージデバイス上の**全ブロック** に書き込みが行われることがあります。この書き込みが実行されると、SSD のパフォーマンスが急速に低下してしまいます。

Red Hat Enterprise Linux 6.4 では、**discard** に完全に対応しているファイルシステムは ext4 と XFS のみになります。Red Hat Enterprise Linux 6 のそれ以前のバージョンでは、**discard** に完全に対応しているのは ext4 のみでした。デバイスで **discard** コマンドを有効にするには、**mount** オプションの **discard** を使用します。たとえば、**discard** を有効にして **/dev/sda2** を **/mnt** にマウントする場合は、以下を実行します。

```
# mount -t ext4 -o discard /dev/sda2 /mnt
```

ext4 はデフォルトでは **discard** コマンドを発行しません。**discard** コマンドを正しく実装しない可能性があるデバイス上での問題を回避するためです。Linux **swap** コードは **discard** が有効になっているデバイスに対して **discard** コマンドを発行するため、この動作を制御するオプションはありません。

## 22.2. チューニングに関する注意点

本セクションでは、SSD のパフォーマンスに影響を及ぼす可能性のある設定を行う場合の注意すべきいくつかの点について説明します。

### I/O スケジューラー

いずれの I/O スケジューラーでもほとんどの SSD で正しく動作するはずです。ただし、他のストレージタイプと同様に、特定のワークロードに対する最適な設定を決定するにはベンチマーク評価を行うことを推奨します。

SSD を使用する際は、I/O スケジューラーの変更は特定のワークロードのベンチマーク評価を行う場合に限ることをお勧めしています。I/O スケジューラーの各種タイプについては 『I/O Tuning Guide』 (Red Hat 提供) を参照してください。また、以下のカーネル関連のドキュメントにも I/O スケジューラーの切り替え方法についての記載があります。

**/usr/share/doc/kernel-*version*/Documentation/block/switching-sched.txt**

Red Hat Enterprise Linux 7.0 では、デフォルトの I/O スケジューラーが、SATA ドライブの場合を除き Deadline になりました。SATA ドライブの場合、CFQ がデフォルトの I/O スケジューラーです。高速のストレージの場合、Deadline のパフォーマンスは CFQ を上回り、特別なチューニングなしにパフォーマンスが強化されます。ただし、このデフォルトは一部のディスク (SAS の回転ディスクなど) に適さない場合があります。その場合、I/O スケジューラーは CFQ に変更する必要があります。

### 仮想メモリー

I/O スケジューラーと同様に、仮想メモリー (VM) サブシステムにも特別なチューニングは必要ありません。たとえば SSD での I/O が高速であることを考慮すると、書き込みアクティビティーが増加してもディスク上の他の動作の待ち時間にネガティブな影響は与えないはずですので、**vm_dirty_background_ratio** や **vm_dirty_ratio** の設定を低くすることができるでしょう。ただし、**より全般的な I/O** を生成する場合があるため、ワークロードに固有のテストを行わずに使用するのはお勧めできません。

### Swap

SSD は swap デバイスとしても使用でき、ページアウトやページインのパフォーマンスに優れている場合が多くあります。

# 第23章 書き込みバリア

*書き込みバリア*とはカーネルのメカニズムで、電力供給の停止が揮発性の書き込みキャッシュを持つストレージデバイスに対して発生した場合でもファイルシステムのメタデータは永続的なストレージに正しい順序で書き込まれるようにします。また、書き込みバリアが有効になっているファイルシステムでは、 電力供給の停止が発生しても **fsync()** で転送されるデータの永続性を維持します。

書き込みバリアを有効にすると相当のパフォーマンス低下を招くアプリケーションがあります。特に、**fsync()** をかなり頻繁に使用するアプリケーションや小さなファイルの作成、削除を繰り返すアプリケーションの場合、実行速度がかなり遅くなる可能性が高くなります。

## 23.1. 書き込みバリアの重要性

ファイルシステムは、整合性が維持されるようメタデータの安全な更新を慎重に行います。ジャーナリングされたファイルシステムによりメタデータの更新がトランザクションにバンドルされて次のように恒久ストレージに送信されます。

1. まず、トランザクションのボディーがストレージデバイスに送信されます。
2. 次にコミットブロックが送信されます。
3. トランザクションとそのトランザクションのコミットブロックがディスクに書き込まれると、ファイルシステムは、そのトランザクションが電力供給の停止にも耐え得るとみなします。

ただし、キャッシュの容量が大きいストレージデバイスの場合、電力供給が停止している間のファイルシステムの整合性の維持はより複雑になります。ローカルの S-ATA ドライブや SAS ドライブのようなストレージターゲットのデバイスには 32 MB から 64 MB の書き込みキャッシュがある場合があります (最近のドライブ)。ハードウェア RAID コントローラーには内部書き込みキャッシュがあるものがよくあります。さらに、NetApp、IBM、Hitachi、EMC (その他多数) などのハイエンドアレイにも大容量のキャッシュがあります。

書き込みキャッシュのあるストレージデバイスは、データがキャッシュに入った時点で I/O は「complete」(完了) と報告します。キャッシュの電力供給が停止した場合にはデータも失われます。さらに悪いことに、永続的なストレージへのキャッシュのデステージにより、元のメタデータの順序が変わってしまう場合もあります。これが発生すると、関連付けられている完全なトランザクションがないままコミットブロックがディスクに現れる可能性があります。その結果、電力復旧後にジャーナルは初期化されていないトランザクションブロックをファイルシステムに再生する場合があり、これがデータの不整合や破損を招くことになります。

### 書き込みバリアの動作

Linux カーネルでの書き込みバリアは、I/O の前後にストレージの書き込みキャッシュをフラッシュすることで実施されます。これは **順序が非常に重要** になります。トランザクションが書き込まれた後、ストレージキャッシュがフラッシュされてコミットブロックの書き込みが行われます。その後、再びキャッシュがフラッシュされます。これにより以下が確実に行われることになります。

- ディスクにすべてのデータが含める
- 再度の順序付けは行わない

バリアを有効にすると、**fsync()** 呼び出しによってストレージキャッシュのフラッシュも実行されます。これにより **fsync()** が返された直後に電力供給の停止が発生した場合でも、ファイルのデータは必ずディスク上で永続化します。

## 23.2. 書き込みバリアを有効または無効にする

電源供給の停止が発生した場合のデータ破損のリスクを軽減するため、バッテリー駆動の書き込みキャッシュを使用するストレージデバイスがあります。一般的にはハイエンドのアレイや数種のハードウェアコントローラではバッテリー駆動の書き込みキャッシュを使用しています。ただし、キャッシュの揮発性がカーネルには見えないため、Red Hat Enterprise Linux 7 ではデフォルトで、対応している全ジャーナリングファイルシステム上の書き込みバリアを有効にしています。

**注記**

書き込みキャッシュは I/O のパフォーマンス向上を目的として設計されています。ただし、書き込みバリアを有効にするということは、これらのキャッシュを継続的にフラッシュするということであり、これにより大幅なパフォーマンス低下が生じる可能性があります。

揮発性ではないバッテリー駆動の書き込みキャッシュを持つデバイスや、書き込みキャッシュ機能を無効にしているデバイスに対しては、**mount** に **-o nobarrier** オプションを使ってマウント時に書き込みバリアを安全に無効にすることができます。ただし、書き込みバリアに対応していないデバイスがあります。こうしたデバイスの場合、エラーメッセージが **/var/log/messages** に記録されます (表23.1「ファイルシステムごとの書き込みバリアエラーメッセージ」 を参照)。

**表23.1 ファイルシステムごとの書き込みバリアエラーメッセージ**

| ファイルシステム | エラーメッセージ |
|---|---|
| ext3/ext4 | **JBD: barrier-based sync failed on *device* - disabling barriers** |
| XFS | **Filesystem *device* - Disabling barriers, trial barrier write failed** |
| btrfs | **btrfs: disabling barriers on dev *device*** |

## 23.3. 書き込みバリアに関する注意点

データ保護に書き込みバリアを必要としないシステム設定があります。こうしたシステム設定ではほとんどの場合、書き込みバリアを有効にすることが大幅なパフォーマンス低下を招く要因となるため、書き込みバリア以外の方法を選択するのが得策と言えます。

### 書き込みキャッシュを無効にする

データ整合性の問題を回避するもう 1 つの方法は、電力供給の停止が発生した場合に書き込みキャッシュによるデータが損失しないようにすることです。可能な場合は、書き込みキャッシュを単純に無効にしてしまうのが最適な方法です。1 つまたは複数の SATA ドライブ (ローカル SATA Controller Intel AHCI 部分と区別) を搭載している簡単なサーバーやデスクトップでは、次のようにして該当の SATA ドライブの書き込みキャッシュを **hdparm** コマンドで無効にすることができます。

```
# hdparm -W0 /device/
```

### バッテリー駆動の書き込みキャッシュ

バッテリー駆動の書き込みキャッシュを持つハードウェア RAID コントローラーを使用している場合にも書き込みバリアは不要になります。このようなコントローラーを装備したシステムでそのコンポーネントとなるドライブが搭載している書き込みキャッシュが無効になっている場合、コントローラー自体がライトスルーにしていることがわかります。このため、 電力供給停止時でも書き込みキャッシュのデータは維持されることがカーネルで認識されます。

ほとんどの場合、コントローラーは該当ドライブへの問い合わせや操作に製造元固有のツールを使用します。たとえば、LSI Megaraid SAS コントローラーの場合はバッテリー駆動の書き込みキャッシュを使用し、この種のコントローラーには該当ドライブの管理に **MegaCli64** ツールが必要になります。 LSI Megaraid SAS のすべてのバックエンドドライブの状態を表示する場合は以下のようにします。

```
# MegaCli64 -LDGetProp  -DskCache  -LAll -aALL
```

LSI Megaraid SAS の全バックエンドドライブの書き込みキャッシュを無効にする場合は次のようにします。

```
# MegaCli64 -LDSetProp -DisDskCache -Lall -aALL
```

**注記**

ハードウェア RAID カードはシステムの稼働中にバッテリー充電を行います。一定時間以上システムの電源がオフになるとバッテリーの充電がなくなるため、電力供給停止時に保存したデータが不安定になります。

### ハイエンドのアレイ

ハイエンドのアレイには、電力供給停止時におけるデータ保護に関してさまざまな方法があるため、外付け RAID ストレージ内の内部ドライブの状態を確認する必要はありません。

### NFS

データの整合性は NFS サーバー側で処理されるため、NFS クライアント側では書き込みバリアを有効にする必要はありません。電源供給の停止中、データの永続性が確保されるよう NFS サーバーの設定を行ってください (書き込みバリアまたは別の方法のいずれかによる)。

# 第24章 ストレージの I/O 調整とサイズ

最近の SCSI および ATA 標準への強化により、ストレージデバイスが推奨の (また、場合によっては必須の) *I/O 調整* と *I/O サイズ* を示すようになりました。この情報は特に物理的なセクターサイズを 512 バイトから 4 キロバイトに増加させている新しいディスクドライブで役に立ちます。また、チャンクサイズやストライプのサイズがパフォーマンスに影響を与える可能性がある RAID デバイスに対しても役に立ちます。

ベンダー提供の I/O 調整と I/O サイズの情報を処理するために Linux I/O スタックが強化され、ストレージ管理ツール (**parted**、**lvm**、**mkfs.***  など) によるデータ配置とアクセスの最適化が可能になります。レガシーデバイスが I/O 調整や I/O サイズなどのデータをエクスポートしない場合、Red Hat Enterprise Linux 7 のストレージ管理ツールは安全のため 4k (または 4k より大きい 2 の累乗) の境界で I/O を調整します。これにより、4k セクターのデバイスが必須または推奨の I/O 調整やサイズを表示しない場合であっても正しく動作するようになります。

デバイスから取得したオペレーティングシステムの情報を確定する方法については 「ユーザー領域のアクセス」 を参照してください。このデータはこの後にデータの配置を確定するためにストレージ管理ツールによって使用されます。

IO スケジューラーは Red Hat Enterprise Linux 7 から変更されました。デフォルトの IO スケジューラーは、SATA ドライブの場合を除き、*Deadline* です。CFQ は SATA ドライブのデフォルト IO スケジューラーです。高速なストレージの場合、Deadline のパフォーマンスは CFQ を上回り、Deadline の使用時には、特別なチューニングなしにパフォーマンスが強化されます。

一部のディスク (SAS 回転ディスクなど) に適したデフォルトが設定されていない場合は、IO スケジューラーを CFQ に変更してください。これはワークロードによって異なります。

## 24.1. ストレージアクセス用のパラメーター

オペレーティングシステムは、次の情報を使って I/O の調整とサイズを確定します。

**physical_block_size**

デバイスが動作できる最小の内部ユニット

**logical_block_size**

デバイス上の場所指定に外部で使用される

**alignment_offset**

基礎となる物理的なアライメントのオフセットとなる Linux ブロックデバイス (パーティション／MD／LVM デバイス) の先頭部分のバイト数

**minimum_io_size**

ランダムな I/O に対して推奨のデバイス最小ユニット

**optimal_io_size**

ストリーミング I/O に対するデバイスの推奨ユニット

たとえば、特定の 4K セクターのデバイスでは、内部では **physical_block_size** に 4K を使用しているのに、Linux に対してはより小さい 512 バイトの **logical_block_size** を公開している場合があります。この違いが I/O の調整ミスを招くことがあります。これに対処するため、ブロックデバイスの先頭が基礎となる物理的なアライメントのオフセットとなる場合、Red Hat Enterprise Linux 7 の I/O スタックは必ず alignment_offset に十分なサイズとなるよう必然的に調整される境界 (**physical_block_size**) 上ですべてのデータエリアを開始しようとします。

ストレージの製造元では、デバイスのランダムな I/O (**minimum_io_size**) およびストリーミングの I/O (**optimal_io_size**) に対して推奨となる最小ユニットに関する *I/O hints* も提供しています。たとえば、**minimum_io_size** と **optimal_io_size** は RAID デバイスのチャンクサイズとストライプサイズにそれぞれ該当します。

## 24.2. ユーザー領域のアクセス

常に正しく調整された正しいサイズの入出力を使用するよう注意してください。とくに、ダイレクトな入出力アクセスの場合には重要となります。ダイレクトな入出力は **logical_block_size** の境界上で、**logical_block_size** の倍数単位で調整してください。

ネイティブの 4K デバイス (つまり、**logical_block_size** が 4K という意味) では、アプリケーションがデバイスの **logical_block_size** の倍数単位でダイレクトな入出力を行うことが重要となってきます。つまり、4k の調整した入出力ではなく 512 バイトの調整した入出力を行うネイティブな 4k デバイスではアプリケーションの実行は失敗することになります。

これを回避するには、正しい入出力調整とサイズを使用していることを確認するためアプリケーションにデバイスの入出力パラメーターの問い合わせを行わせる必要があります。前述のように入出力のパラメーターは **sysfs** とブロックデバイス **ioctl** の両方のインターフェースを介して公開されます。

詳細は **man libblkid** をご覧ください。**man** ページは **libblkid-devel** パッケージで提供しています。

### sysfs インターフェース

- /sys/block/***disk***/alignment_offset
- /sys/block/***disk***/***partition***/alignment_offset
- /sys/block/***disk***/queue/physical_block_size
- /sys/block/***disk***/queue/logical_block_size
- /sys/block/***disk***/queue/minimum_io_size
- /sys/block/***disk***/queue/optimal_io_size

カーネルは、入出力のパラメーター情報を提供しないレガシーなデバイス用にこれらの **sysfs** 属性をエクスポートします。たとえば、以下のようになります。

**例24.1 sysfs インターフェース**

```
alignment_offset:    0
physical_block_size: 512
logical_block_size:  512
minimum_io_size:     512
optimal_io_size:     0
```

### ブロックデバイス ioctls

- **BLKALIGNOFF**: **alignment_offset**
- **BLKPBSZGET**: **physical_block_size**

- **BLKSSZGET**: **logical_block_size**
- **BLKIOMIN**: **minimum_io_size**
- **BLKIOOPT**: **optimal_io_size**

## 24.3. 標準

本セクションでは ATA デバイスおよび SCSI デバイスによって使用される入出力の標準について説明します。

### ATA

ATA デバイスは **IDENTIFY DEVICE** コマンドで適切な情報を報告する必要があります。ATA デバイスが報告する入出力パラメーターは、**physical_block_size**、**logical_block_size**、**alignment_offset** のみになります。その他の I/O hints は ATA コマンドセットの範囲外となります。

### SCSI

Red Hat Enterprise Linux 7 の入出力パラメーターのサポートでは、少なくとも *SCSI プライマリーコマンドプロトコル*の**バージョン 3** (SPC-3) が必要になります。カーネルが SPC-3 の準拠を求めるデバイスに対して送信するのは、*拡張した問い合わせ*(**BLOCK LIMITS VPD** ページへのアクセスを取得するため) と **READ CAPACITY(16)** コマンドのみになります。

**READ CAPACITY(16)** コマンドでブロックサイズと調整オフセットを与えます。

- **LOGICAL BLOCK LENGTH IN BYTES** は、**/sys/block/*disk*/queue/physical_block_size** の取得に使用します
- **LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT** は、**/sys/block/*disk*/queue/logical_block_size** の取得に使用します
- **LOWEST ALIGNED LOGICAL BLOCK ADDRESS** は、 以下の取得に使用します
  - **/sys/block/*disk*/alignment_offset**
  - **/sys/block/*disk*/*partition*/alignment_offset**

I/O hints は **BLOCK LIMITS VPD** ページ (**0xb0**) にあります。また、このページは以下を取得するため **OPTIMAL TRANSFER LENGTH GRANULARITY** や **OPTIMAL TRANSFER LENGTH** も使用します。

- **/sys/block/*disk*/queue/minimum_io_size**
- **/sys/block/*disk*/queue/optimal_io_size**

**sg3_utils** パッケージで **sg_inq** ユーティリティーを与えます。このユーティリティーを使って **BLOCK LIMITS VPD** ページにアクセスします。次を実行します。

```
# sg_inq -p 0xb0 disk
```

## 24.4. 入出力パラメーターのスタック

Linux 入出力スタックの層はすべて各種の入出力パラメーターがスタックまで伝播するよう設計されています。任意の層が属性を消費したり多くのデバイスを集約する場合、その層は適切な入出力パラメーターを公開しなければなりません。これにより上位の層のデバイスまたはツールは変換後のストレージを正確に認識することができるようになります。例をいくつか示します。

- ゼロ以外の **alignment_offset** の調整は入出力スタック内の 1 つの層に限ってください。この層によって調整が完了すると **alignment_offset** がゼロのデバイスがエクスポートされます。
- LVM で作成したストライプ化した Device Mapper (DM) によって、そのストライプ数 (ディスク数) とユーザー入力のチャンクサイズに応じた **minimum_io_size** と **optimal_io_size** がエクスポートされなければなりません。

Red Hat Enterprise Linux 7 では、Device Mapper と Software Raid (MD) デバイスのドライバーを使用することで異なる入出力パラメーターを持つデバイスの自由な組み合わせを実現しています。カーネルのブロック層によって各デバイスの入出力パラメーターを合理的に組み合わせるよう試行されます。カーネル自体は異種のデバイスの組み合わせを避けることはありませんが、異種のデバイスの組み合わせによって生じるリスクについては注意が必要になります。

たとえば、512 バイトのデバイスと 4K のデバイスを一つの論理的な DM デバイスに組み合わせ、**logical_block_size** を 4K にすることができます。このようなハイブリッドデバイス上で層を構成するファイルシステムは 4K がアトミックに書き込みされるとみなしますが、実際には 512 バイトのデバイスの場合には 8 つの論理ブロックアドレスに広がることになります。4K の **logical_block_size** を上位レベルの DM デバイスに使用すると、システムのクラッシュが発生した場合に 512 バイトのデバイスには不完全な書き込みが起こる可能性が高くなります。

複数のデバイスの入出力パラメーターを組み合わせると競合が起きる場合には、デバイスには不完全な書き込みの可能性があり、誤調整されていることを示す警告がブロック層によって発行される場合があります。

## 24.5. 論理ボリュームマネージャー

LVM は、カーネルの DM デバイス管理に使用するユーザー領域ツールを提供します。LVM は LVM 管理のデバイスに関連付けられているゼロ以外の **alignment_offset** に十分なサイズを確保するためデータエリア (任意の DM デバイスによって使用される) の開始点をずらします。つまり、論理ボリュームが正しく調整されることになります (**alignment_offset=0**)。

デフォルトでは LVM はいずれの **alignment_offset** の調整も行いますが、**/etc/lvm/lvm.conf** 内の **data_alignment_offset_detection** を **0** に設定することでこの動作を無効にすることができます。 この動作の無効化はお勧めしません。

また、 LVM はデバイスの I/O hints も検出します。 デバイスのデータエリアの開始は sysfs で公開される **minimum_io_size** や **optimal_io_size** の倍数になります。 **optimal_io_size** が定義されていない場合は (つまり **0** になっている場合)、 **minimum_io_size** が使用されます。

デフォルトではこのような I/O hints は LVM によって自動的に確定されますが、 **/etc/lvm/lvm.conf** 内の **data_alignment_detection** を **0** に設定することでこの動作を無効にすることができます。 この動作の無効化はお勧めしません。

## 24.6. パーティションとファイルシステムのツール

本セクションでは、 さまざまなパーティションツールやファイルシステム管理ツールがどのようにしてデバイスの入出力パラメータと相互作用するのかについて説明しています。

### util-linux-ng の libblkid と fdisk

**util-linux-ng** パッケージに入っている **libblkid** ライブラリにはデバイスの入出力パラメータへアクセスするためのプログラム的な API が含まれています。 **libblkid** によってアプリケーション、 特に

ダイレクト入出力を使用するアプリケーションがその入出力要求を正しく区分できるようになります。**util-linux-ng** に入っている **fdisk** ユーティリティーは **libblkid** を使ってデバイスの入出力パラメータを全パーティションで最適となる配置に定義します。 **fdisk** ユーティリティーによって 1MB の境界ですべてのパーティション調整が行われます。

## parted と libparted

**parted** の **libparted** ライブラリも **libblkid** の入出力パラメータ API を使用します。Red Hat Enterprise Linux 7 のインストーラー (**Anaconda**) では **libparted** が使用されます。つまり、インストーラまたは **parted** のいずれかで作成されるパーティションはすべて正しく調整されることになります。入出力パラメータを提供しないようなデバイスで作成されるパーティションの場合、デフォルトのアライメントは 1MB になります。

経験的法則による **parted** では次を使用します。

- 1 番目のプライマリパーティションの開始のオフセットには常に報告された **alignment_offset** を使用します。
- **optimal_io_size** を指定すると (つまり **0** 以外を指定)、 **optimal_io_size** の境界にあるパーティションすべての調整を行います。
- **optimal_io_size** を指定しないと (つまり **0**)、 **alignment_offset** は **0** になります。 また、 **minimum_io_size** は 2 の累乗になり 1MB のデフォルトアライメントを使用します。

  これが I/O hints を提供しないようなレガシーなデバイスの汎用になります。 このようにデフォルトでは全パーティションが 1MB の境界で調整されます。

**注記**

Red Hat Enterprise Linux 7 では、I/O hints を提供するデバイスとしないデバイスとを **alignment_offset=0** や **optimal_io_size=0** で区別することはできません。 この様なデバイスには単一の SAS 4K デバイスなども含まれます。最悪の場合、ディスクの先頭にある 1MB の領域を失うことになります。

## ファイルシステムのツール

各種の **mkfs.*filesystem*** ユーティリティーも拡張されデバイスの入出力パラメータを使用するようになっています。 こうしたユーティリティーでは、 基礎となるストレージデバイスの **logical_block_size** より小さいブロックサイズを使用したファイルシステムのフォーマットは行えません。

**mkfs.gfs2** の場合を除き、 他の **mkfs.*filesystem*** ユーティリティーもすべて I/O hints を使って基礎となるストレージデバイスの **minimum_io_size** と **optimal_io_size** に応じたオンディスクデータ構造とデータエリアをレイアウトします。 これにより、 ファイルシステムを各種の RAID (ストライプ化) レイアウトに応じて最適にフォーマットできるようになります。

# 第25章 リモートディスクレスシステムを設定する

PXE 経由で起動する基本的なディスクレスシステムを設定する場合、次のようなパッケージが必要になります。

- **tftp-server**
- **xinetd**
- **dhcp**
- **syslinux**
- **dracut-network**

リモートディスクレスシステムを起動するには、**tftp** サービス (**tftp-server** 提供) と DHCP サービス (**dhcp** 提供) の両方が必要になります。**tftp** サービスは、PXE ローダーを使ってネットワーク経由でカーネルのイメージと **initrd** を取得する際に使用されます。

次のセクションでは、ネットワーク環境にリモートディスクレスシステム群を導入する場合に必要な手順について簡単に説明します。

## 25.1. ディスクレスクライアントの tftp サービスを設定する

**tftp** はデフォルトでは無効になっています。tftp を有効にしてネットワーク経由による PXE の起動を許可するには、**/etc/xinetd.d/tftp** の **Disabled** オプションを **no** に設定します。**tftp** の設定は次の手順で行います。

**手順25.1 tftp を設定するには**

1. **tftp** root ディレクトリー (**chroot**) は **/var/lib/tftpboot** に置かれます。以下のようにして **/usr/share/syslinux/pxelinux.0** を **/var/lib/tftpboot/** にコピーします。

   ```
   cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/
   ```

2. **tftp** root ディレクトリー内に **pxelinux.cfg** ディレクトリーを作成します。

   ```
   mkdir -p /var/lib/tftpboot/pxelinux.cfg/
   ```

また、**tftp** のトラフィックを許可するためファイアウォールのルールを適切に設定する必要があります。**tftp** は TCP ラッパーに対応しているため、**/etc/hosts.allow** を使って **tftp** へのホストのアクセスを設定することができます。TCP ラッパーの設定方法および **/etc/hosts.allow** 設定ファイルについての詳細は、Red Hat Enterprise Linux 7 『セキュリティガイド』 を参照してください。また、**man hosts_access** でも **/etc/hosts.allow** に関する記載をご覧になれます。

ディスクレスクライアントの **tftp** を設定した後に、DHCP、NFS およびエクスポートしたファイルシステムの設定を適宜行います。これらの設定方法については 「ディスクレスクライアントの DHCP を設定する」 および 「ディスクレスクライアント用にエクスポートしたファイルシステムの設定を行う」を参照してください。

## 25.2. ディスクレスクライアントの DHCP を設定する

**tftp** サーバーを設定した後に、DHCP サーバーを同じホストマシン上に設定する必要があります。DHCP サーバーの設定方法については、Red Hat Enterprise Linux 7 『導入ガイド』 を参照してください。また、DHCP サーバーで PXE の起動を有効にしてください。PXE の起動を有効にするには、次の設定を **/etc/dhcp/dhcp.conf** に追加します。

```
allow booting;
allow bootp;
class "pxeclients" {
   match if substring(option vendor-class-identifier, 0, 9) =
"PXEClient";
   next-server server-ip;
   filename "pxelinux.0";
}
```

***server-ip*** は **tftp** サービスと DHCP サービスがあるホストマシンの IP アドレスに置き換えてください。これで **tftp** と DHCP が設定されるので、残りは NFS とエクスポートしたファイルシステムの設定のみが必要になります。これらの設定方法については、「ディスクレスクライアント用にエクスポートしたファイルシステムの設定を行う」を参照してください。

## 25.3. ディスクレスクライアント用にエクスポートしたファイルシステムの設定を行う

エクスポートしたファイルシステム (ネットワーク内でディスクレスのクライアントが使用) の root ディレクトリーを NFS 経由で共有します。**/etc/exports** に root ディレクトリーを追加してディレクトリーをエクスポートするように NFS サービスを設定します。実行方法の詳細については、「**/etc/exports** 設定ファイル」を参照してください。

ディスクレスのクライアントに完全に対応できるようにするため、root ディレクトリーには Red Hat Enterprise Linux の完全なインストールを組み込む必要があります。以下のように、**rsync** で実行中のシステムと同期することができます。

```
# rsync -a -e ssh --exclude='/proc/*' --exclude='/sys/*' hostname.com:/
/exported/root/directory
```

***hostname.com*** には **rsync** で同期する実行中のシステムのホスト名を入れます。***/exported/root/directory*** はエクスポートしたファイルシステムへのパスになります。

または、**yum** に **--installroot** オプションを指定し、Red Hat Enterprise Linux を特定の場所にインストールすることもできます。たとえば、以下のようになります。

```
yum groupinstall Base --installroot=/exported/root/directory
```

エクスポートしたファイルシステムをディスクレスクライアントが使用できるようにする前に行っておかなければならない設定があります。次の手順に従ってください。

**手順25.2 ファイルシステムの設定**

1. エクスポートしたファイルシステムの **/etc/fstab** を編集して (少なくとも) 次の設定を組み込みます。

   ```
   none  /tmp  tmpfs defaults 0 0
   tmpfs  /dev/shm tmpfs defaults 0 0
   sysfs  /sys  sysfs defaults 0 0
   proc  /proc  proc  defaults 0 0
   ```

2. ディスクレスのクライアントが使用するカーネルを選択し (**vmlinuz-*kernel-version***)、**tftp** の boot ディレクトリーにコピーします。

```
# cp /boot/vmlinuz-kernel-version /var/lib/tftpboot/
```

3. ネットワークサポートで **initrd** (**initramfs-*kernel-version*.img**) を作成します。

```
# dracut initramfs-kernel-version.img kernel-version
```

作成した **initramfs-*kernel-version*.img** を **tftp** boot ディレクトリーにもコピーします。

4. **initrd** と **/var/lib/tftpboot** 内のカーネルを使用するようにデフォルトの起動設定を編集します。この設定によりディスクレスクライアントの root には、エクスポートしたファイルシステム (***/exported/root/directory***) を読み込みと書き込みの両方の権限でマウントするよう指示されます。次のように **/var/lib/tftpboot/pxelinux.cfg/default** を設定します。

```
default rhel7

label rhel7
  kernel vmlinuz-kernel-version
  append initrd=initramfs-kernel-version.img root=nfs:server-
ip:/exported/root/directory rw
```

***server-ip*** には **tftp** サービスと DHCP サービスがあるホストマシンの IP アドレスを入力します。

これで NFS 共有をディスクレスのクライアントにエクスポートする準備が整いました。これらのディスクレスのクライアントは PXE のネットワーク経由で起動できるようになります。

# 第26章 オンラインストレージ管理

オペレーティングシステムを稼動させたまま再起動せずにストレージデバイスの追加、削除、またはサイズ変更を実行したい場合がよくあります。本章では、 Red Hat Enterprise Linux 7 のホストシステムを稼動させたままでシステム上のストレージデバイスを再設定する手順について簡単に説明しています。さらに、本章では iSCSI およびファイバーチャネルのストレージ相互接続について扱います。他のタイプについての詳細は今後追加していく予定です。

本章では、ストレージデバイスの追加、削除、変更、モニターを中心に説明します。ファイバーチャネルおよび iSCSI プロトコルについては詳細に説明しません。プロトコルについては他のドキュメントを参照してください。

さまざまな **sysfs** オブジェクトを参照しますが、このオブジェクトの名前やディレクトリー構成は Red Hat Enterprise Linux の主要なリリースごとに変更されます。これは、アップストリームとなる Linux カーネルで安定した内部 API が提供されていないためです。移行が可能な方法で **sysfs** オブジェクトを参照する方法についてのガイドラインは、カーネルソースツリーにある **/usr/share/doc/kernel-doc-*version*/Documentation/sysfs-rules.txt** を参照してください。

**警告**

オンラインでのストレージの再設定は慎重に行ってください。プロセス中のシステム障害や中断は予期しない結果を招く恐れがあります。変更の操作を行っている間は、エクステントが最大となるようシステム負荷をできるだけ軽減させることをお勧めします。これにより、設定変更の途中で入出力エラーやメモリー不足によるエラーなどが発生する可能性が低減します。次のセクションでは、これに関してさらに詳しく説明します。

In addition, Red Hat recommends that you back up all data before reconfiguring online storage.

## 26.1. ファイバーチャネル

このセクションでは、ファイバーチャネル API、ネイティブの Red Hat Enterprise Linux 7 ファイバーチャネルドライバー、およびこれらのドライバーのファイバーチャネル機能について説明します。

### 26.1.1. ファイバーチャネル API

以下は、ユーザースペース API の提供に使用されるファイルを含む **/sys/class/** ディレクトリーの一覧です。それぞれの項目で、ホスト番号は ***H***、バス番号は ***B***、ターゲットは ***T***、論理ユニット番号 (LUN) は ***L***、およびリモートポート番号は***R*** で表示されています。

**重要**

マルチパスソフトウェアを使用されている場合は、このセクションで記載する値のいずれかを変更する前に、ご使用のハードウェアのベンダーにお問い合わせになることをお勧めします。

**トランスポート:/sys/class/fc_transport/target*H*:*B*:*T*/**

- **port_id** — 24-bit ポート ID/アドレス
- **node_name** — 64-bit ノード名

- **port_name** — 64-bit ポート名

**リモートポート:/sys/class/fc_remote_ports/rport-*H*:*B*-*R*/**

- **port_id**
- **node_name**
- **port_name**
- **dev_loss_tmo** — リンクが「不良」とマークされるまでの待ち時間を示す秒数です。リンクが「不良」とマークされると、該当するパスで実行している I/O (およびそのパス上のすべての新規 I/O) は失敗します。

  デフォルトの **dev_loss_tmo** 値は使用されるドライバー/デバイスによって異なります。Qlogic アダプターが使用される場合は 35 秒で、Emulex アダプターが使用されると、30 秒となります。**dev_loss_tmo** の値は、**scsi_transport_fc** モジュールパラメーターの **dev_loss_tmo** で変更できますが、ドライバーはこのタイムアウト値を上書きできます。

  **dev_loss_tmo** の最大値は 600 です。**dev_loss_tmo** がゼロか、または 600 より大きな値に設定されている場合は、ドライバーの内部タイムアウトが代わりに使用されます。
- **fast_io_fail_tmo** — リンク問題が検出された時に実行された I/O が失敗するまでの待ち時間の長さです。ドライバーに達する I/O は失敗します。I/O がブロックされたキューにある場合は、**dev_loss_tmo** が期限切れになり、キューがブロック解除になるまでは失敗しません。

**ホスト:/sys/class/fc_host/host*H*/**

- **port_id**
- **issue_lip** — ドライバーがリモートポートを再検出するように指示します。

### 26.1.2. ネイティブファイバーチャネルのドライバーと機能

Red Hat Enterprise Linux 7 は以下のネイティブファイバーチャネルドライバーと共に提供されます。

- **lpfc**
- **qla2xxx**
- **zfcp**
- **mptfc**
- **bfa**

表26.1「ファイバーチャネルの API 機能」では、Red Hat Enterprise Linux 7 の各ネイティブドライバーの異なるファイバーチャネル API 機能を説明しています。X は該当する機能のサポートがあることを示します。

**表26.1 ファイバーチャネルの API 機能**

| | lpfc | qla2xxx | zfcp | mptfc | bfa |
|---|---|---|---|---|---|
| トランスポート **port_id** | X | X | X | X | X |
| トランスポート **node_name** | X | X | X | X | X |

| | lpfc | qla2xxx | zfcp | mptfc | bfa |
|---|---|---|---|---|---|
| トランスポート **port_name** | X | X | X | X | X |
| リモートポート **dev_loss_tmo** | X | X | X | X | X |
| リモートポート **fast_io_fail_tmo** | X | X [a] | X [b] | | X |
| ホスト **port_id** | X | X | X | X | X |
| ホスト **issue_lip** | X | X | | | X |

[a] Red Hat Enterprise Linux 5.4 時点でサポート
[b] Red Hat Enterprise Linux 6.0 時点でサポート

## 26.2. iSCSI

このセクションでは、iSCSI API および、**iscsiadm** ユーティリティーについて説明します。**iscsiadm** ユーティリティーを使用するには、**yum install iscsi-initiator-utils** を実行して **iscsi-initiator-utils** パッケージをインスト－ルしてください。

Red Hat Enterprise Linux 7 では、iSCSI サービスはデフォルトでレイジーに開始されます。root が iSCS デバイスになかったり、**node.startup = automatic** でマークされたノードがない場合は、iscsid または iscsi カーネルモジュールの起動を要求をする **iscsiadm** コマンドが実行されるまでは、iSCSI サービスは開始されません。たとえば、ディスカバリーコマンド **iscsiadm -m discovery -t st -p ip:port** を実行すると、iscsiadmin が iSCSI サービスを開始します。

iscsid デーモンを実行して、iSCSI カーネルモジュールのロードを強制するには、**service iscsid force-start** コマンドを実行します。

### 26.2.1. iSCSI API

実行中のセッションに関する情報を得るには、以下を実行します。

```
# iscsiadm -m session -P 3
```

このコマンドは、セッション/デバイスの状態、セッション ID (sid)、いくつかのネゴシエートしたパラメーター、およびセッション経由でアクセス可能な SCSI デバイスを表示します。

より短い出力 (たとえば、sid とノード間のマッピングのみの表示) には、以下を実行します。

```
# iscsiadm -m session -P 0
```

または

```
# iscsiadm -m session
```

これらのコマンドは、以下の形式で実行中のセッションの一覧を表示します。

```
driver [sid] target_ip:port,target_portal_group_tag proper_target_name
```

**例26.1 `iscisadm -m session` コマンドの出力**

例:

```
# iscsiadm -m session

tcp [2] 10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
tcp [3] 10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

iSCSI API についてさらに詳しくは、**/usr/share/doc/iscsi-initiator-utils-*version*/README** を参照してください。

### 26.2.2. iSCSI ターゲットの設定

#### 新しいターゲットの追加

新しい iSCSI ターゲットを追加する場合は **/etc/tgt/targets.conf** 設定ファイルを編集します。このファイルにはいろいろな設定オプションのサンプルが含まれています。サンプルはすべてコメントアウトされています。

基本的なターゲットは次のように定義することができます。

**例26.2 基本的なターゲット**

```
<target iqn.2008-09.com.example:server.target1>
    backing-store /srv/images/iscsi-share.img
    direct-store /dev/sdd
</target>
```

上記の例では LUN を 2 つ持っている単一ターゲットを定義しています。LUN については *バッキングストア* か *ダイレクトストア* のいずれかに記載されています。バッキングストアとはファイルまたはブロックデバイスを指し、ダイレクトストアはローカルの SCSI デバイスになります。シリアル番号やベンダー名などのデバイスパラメーターが新しい iSCSI LUN に渡されます。

#### tgtd サービスを開始する

tgtd サービスを開始するには次を実行します。

```
service tgtd start
```

#### tgtd サービスを停止する

tgtd サービスを停止するには次を実行します。

```
service tgtd stop
```

開かれている接続がある場合は次を使用します。

```
service tgtd force-stop
```

**警告**

このコマンドを使うことによりすべてのターゲットアレイが終了します。

## 26.3. 永続的な命名

オペレーティングシステムはストレージデバイスへのアクセスに使用するパスを参照してそのデバイスへの I/O を発行します。 SCSI デバイスの場合、 パスは次から構成されます。

- ホストのバスアダプターの PCI 識別子 (HBA)
- その HBA 上のチャネル番号
- リモートの SCSI ターゲットのアドレス
- 論理ユニット番号 (LUN)

このパスに基づいたアドレスは永続的ではありません。 システムが再構成されると (本ガイドに記載されているようなオンラインによるシステムの再構成や、 シャットダウン - 再構成 - 再起動の順序で行われる再構成など) アドレスは変更になる可能性があります。 また、 物理的な再構成が行われなかった場合でも、 システムの起動やバスの再スキャンでの検出プロセスで発生するタイミングの変化によりパスの識別子が変更される可能性があります。

オペレーティングシステムでは、 ストレージデバイスへのアクセスパスを表すのに永続的ではない名前がいくつか提供されます。 **/dev/sd** 名や **major:minor** 番号などがこれに該当します。 また、 **/dev/disk/by-path/** ディレクトリー内で管理されているシンボリックリンクも永続的ではありません。 シンボリックリンクはパス識別子から現在の **/dev/sd** 名へのマッピングを行います。 たとえば、 ファイバーチャネルデバイスなら、 PCI 情報と ***Host*:*BusTarget*:*LUN*** 情報は次のように表示されます。

```
pci-0000:02:0e.0-scsi-0:0:0:0 -> ../../sda
```

iSCSI デバイスの場合は、 **by-path/** の名前によりターゲット名とポータル情報から **sd** 名へのマッピングが行われます。

一般的には、 アプリケーションにこうしたパスベースの名前を使用させるのは**適切とは言えません**。 パスが参照するストレージデバイスは変更される可能性があり、 間違ったデータがデバイスに書き込まれる恐れがあるためです。 パスが名前になっているような名前はマルチパスデバイスなどにも適していません。 複数あるストレージデバイスの名前を間違えてしまうことにより、 一貫性のないアクセスで意図しないデータの変更を招いてしまう可能性があるためです。

また、 パスベースの名前はシステム固有となります。 このため、 クラスター内などでデバイスへのアクセスが複数のシステムから発生した場合、 意図しないデータ変更が行われる可能性があります。

こうした理由から、デバイスの識別を目的とした永続的でシステムに依存しない方法が開発されました。 次のセクションで詳しく説明していきます。

### 26.3.1. WWID

*World Wide Identifier* (WWID) を使用するとデバイスを正確に識別することが可能です。 WWID 識別子は、 SCSI 標準で全 SCSI デバイスに必要とされるような永続的でシステムに依存しない ID となります。 すべてのストレージデバイスに対して必ず固有となり、 デバイスのアクセスに使用するパスに依存しません。

この識別子は、 *Device Identification Vital Product Data* (ページ **0x83**) または *Unit Serial Number* (ページ **0x80**) を取得するための SCSI Inquiry を発行することで取得することができます。 WWID から現在の **/dev/sd** 名へのマッピングは **/dev/disk/by-id/** ディレクトリー内で管理されているシンボリックリンクで確認できます。

**例26.3 WWID**

たとえば、 ページ **0x83** の識別子を持つデバイスには次があります。

```
scsi-3600508b400105e210000900000490000 -> ../../sda
```

ページ **0x80** の識別子を持つデバイスには次があります。

```
scsi-SSEAGATE_ST373453LW_3HW1RHM6 -> ../../sda
```

Red Hat Enterprise Linux では、 WWID ベースのデバイス名からそのシステム上の現在の **/dev/sd** 名への正しいマッピングを自動的に維持します。 デバイスへのパスが変更したり、 別のシステムからそのデバイスへのアクセスがあった場合にも、 アプリケーションはディスク上のデータ参照に **/dev/disk/by-id/** を使用することができます。

1 システムから 1 デバイスへのパスが複数ある場合、 **device-mapper-multipath** は WWID を使って検出を行います。 次に **Device-mapper-multipath** は **/dev/mapper/3600508b400105df70000e00000ac0000** などの単一の「擬似デバイス」 を **/dev/mapper/wwid** 内に提示します。

**multipath -l** コマンドでは ***Host*:*Channel*:*Target*:*LUN***、 **/dev/sd** 名、 **major:minor** 番号などの永続的ではない識別子が表示されます。

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=0][active]
 \_ 5:0:1:1 sdc 8:32  [active][undef]
 \_ 6:0:1:1 sdg 8:96  [active][undef]
\_ round-robin 0 [prio=0][enabled]
 \_ 5:0:0:1 sdb 8:16  [active][undef]
 \_ 6:0:0:1 sdf 8:80  [active][undef]
```

**Device-mapper-multipath** では、 WWID に基づく各デバイス名からシステム上の該当**/dev/sd** 名への正しいマッピングが自動的に維持されます。 こうした名前はパスの変更後も永続的となるため、 別のシステムからそのデバイスへのアクセスが行われた場合でも一貫性が保たれます。

**user_friendly_names** 機能を使用すると (**device-mapper-multipath** の機能)、 WWID は **/dev/mapper/mpath*n*** の形式の名前にマッピングされます。 デフォルトでは、 このマッピングは **/etc/multipath/bindings** ファイル内で維持されます。 このファイルが維持されている限り **mpathn** の名前は永続的となります。

**重要**

**user_friendly_names** を使用する場合、クラスターで一貫した名前を取得するために追加の手順を行う必要があります。『DM Multipath 設定管理』 の「クラスター内で一貫したマルチデバイス名」の項を参照してください。

上記のようにシステムで提供される永続的な名前の他にも、 ストレージの WWID にマッピングされる独自の永続的な名前の実装を **udev** ルールを使っても行うことができます。 詳細については http://kbase.redhat.com/faq/docs/DOC-7319 を参照してください。

### 26.3.2. UUID とその他の永続的となる識別子

ストレージデバイスにファイルシステムが含まれる場合、 そのファイルシステムでは以下のいずれかまたはすべてが提供されている場合があります。

- *Universally Unique Identifier* (UUID)
- ファイルシステムラベル

このような識別子は永続的であり、 特定のアプリケーションによってデバイスに書き込まれるメタデータに基づいています。 また、 **/dev/disk/by-label/** ディレクトリー (**boot -> ../../sda1**) と **/dev/disk/by-uuid/** ディレクトリー (**f8bf09e3-4c16-4d91-bd5e-6f62da165c08 -> ../../sda1**) 内でオペレーティングシステムにより維持されるシンボリックリンクを使いデバイスにアクセスする場合にもこの識別子を使用することができます。

**md** および LVM はストレージデバイスにメタデータを書き込み、 デバイスのスキャンをする時にこのデータの読み込みを行います。 いずれの場合にも、 メタデータには UUID が含まれるためストレージデバイスへのアクセスにパス (またはシステム) を使用するかしないかに関係なくそのデバイスを識別することができます。 結果、 メタデータが変更されない限りこの機能で表されるデバイス名は永続的となります。

## 26.4. ストレージデバイスの削除

ストレージデバイス自体へのアクセスを削除する前に、デバイスのデータのバックアップを取ることをお勧めします。次に、I/O をフラッシュしてオペレーティングシステムのデバイスへのすべての参照を削除します (以下を参照)。デバイスでマルチパス機能を使用している場合は、マルチパスの「擬似デバイス」 (「WWID」) とそのデバイスへのパスを表す各識別子に対してこれを行います。マルチパスデバイスへの 1 つのパスだけを削除してその他のパスをそのまま残しておく場合は「ストレージデバイスまたはパスの追加」 に記載されているように、手順はより単純になります。

システムのメモリーが不足している際に I/O をフラッシュするとさらに負荷がかかるため、メモリーが不足している状態でのストレージデバイスの削除は推奨しません。メモリーレベルを判別するには、**vmstat 1 100** コマンドを実行します。以下の場合にはデバイスの削除は推奨されません。

- 100 中 10 を超えるサンプルで空きメモリーがメモリー合計の 5% を下回っている場合 (**free** を使用してメモリー合計を表示することもできます)
- swap 機能がアクティブになっている場合 (**vmstat** 出力で **si** と **so** の欄がゼロ以外の値になっている場合はアクティブです)

デバイスへのすべてのアクセスを削除する一般的な手順は以下になります。

**手順26.1 デバイスを完全に削除する**

1. デバイスのユーザーをすべて終了させ、必要に応じてデータのバックアップを取ります。
2. **umount** を使ってデバイスにマウントしているファイルシステムをすべてアンマウントします。
3. デバイスを使用している **md** および LVM ボリュームからそのデバイスを削除します。デバイスが LVM ボリュームグループのメンバーである場合、**pvmove** コマンドを使ってデータをデバイスから移動させる必要がある場合があります。次に **vgreduce** コマンドを使って物理ボリュームを削除し、(オプションで) **pvremove** コマンドを使ってディスクから LVM のメタデータを削除します。

4. デバイスがマルチパス機能を使用している場合、**multipath -l** を実行してデバイスへのすべてのパスをメモに取ります。次に **multipath -f device** を使ってマルチパスのデバイスを削除します。

5. **blockdev --flushbufs *device*** を実行して、残りの I/O をデバイスへのすべてのパスにフラッシュします。これはとくに **umount** や **vgreduce** オペレーションが I/O フラッシュを行わないローデバイスに重要です。

6. **/dev/sd**、**/dev/disk/by-path**、または **major:minor** 番号など、システム上のアプリケーション、スクリプトおよびユーティリティーのデバイスのパスに基づく名前への参照をすべて削除します。これは将来別のデバイスを追加する場合に、それらのデバイスが現在のデバイスと間違われないようにするのに重要です。

7. 最後に SCSI サブシステムからデバイスへの各パスを削除します。これを実行するには、**echo 1 > /sys/block/*device-name*/device/delete** コマンドを使用します。ここで***device-name*** は **sde** などになります。

   **echo 1 > /sys/class/scsi_device/*h*:*c*:*t*:*l*/device/delete** は上記の変形です。***h*** は HBA 番号、***c*** は HBA 上のチャネル、***t*** は SCSI のターゲット ID、***l*** は LUN になります。

**注記**

これらのコマンドの古い形式である **echo "scsi remove-single-device 0 0 0 0" > /proc/scsi/scsi** は廃止予定になっています。

デバイスの ***device-name***、HBA 番号、HBA チャネル、SCSI のターゲット ID、LUN などは、**lsscsi**、**scsi_id**、**multipath -l**、**ls -l /dev/disk/by-*** など各種のコマンドで確認することができます。

手順26.1「デバイスを完全に削除する」の実行後は、実行中のシステムからデバイスを物理的にかつ安全に取り除くことができるようになります。実行中に他のデバイスに対する I/O を停止する必要はありません。

物理的なデバイスの除去とその後に SCSI バスの再スキャン (「ストレージの相互接続をスキャンする」) を実行して物理的なオペレーティングシステムの状態を更新し、変更を反映するなどの他の手順は推奨されません。これにより、I/O タイムアウトによる遅延が生じ、デバイスが予期せず削除される可能性があります。相互接続の再スキャンを行う必要がある場合には I/O を一時停止してから行ってください (「ストレージの相互接続をスキャンする」 を参照)。

## 26.5. ストレージデバイスへのパスを削除する

マルチパスを使用するデバイスの 1 つのパスを削除する一般的な手順を以下に示します (デバイスへの他のパスは変更しません)。

**手順26.2 ストレージデバイスへのパスを削除する**

1. **/dev/sd**、**/dev/disk/by-path**、**major:minor** 番号など、システム上のアプリケーション、スクリプトおよびユーティリティーのデバイスのパスに基づく名前へのすべての参照を削除します。これは、将来別のデバイスを追加する場合に、これらのデバイスが現在のデバイスと間違われないようにするのに重要です。

2. **echo offline > /sys/block/sda/device/state** を使ってパスをオフラインにします。

   オフラインにすると、このパスのデバイスに送信される後続の I/O がすぐに失敗し始めます。**Device-mapper-multipath** は、このデバイスへの別のパスを継続して使用します。

3. SCSI サブシステムからパスを削除します。**echo 1 > /sys/block/*device-name*/device/delete** コマンドを使用します。ここで、***device-name*** は **sde** などになります (手順26.1「デバイスを完全に削除する」の記載を参照)。

手順26.2「ストレージデバイスへのパスを削除する」の実行後に、実行中のシステムからパスを安全に削除することができます。これを実行している間に I/O を停止する必要はありません。**device-mapper-multipath** は、設定済みのパスのグループ化とフェールオーバーポリシーに応じて、残りのパスへの I/O の再ルーティングを行います。

物理的なデバイスの除去とその後に SCSI バスの再スキャンを実行して物理的なオペレーティングシステムの状態を更新し、変更を反映するなどの他の手順は推奨されません。これにより、I/O タイムアウトによる遅延が生じ、デバイスが予期せず削除される可能性があります。相互接続の再スキャンを行う必要がある場合には I/O を一時停止してから行ってください (「ストレージの相互接続をスキャンする」を参照)。

## 26.6. ストレージデバイスまたはパスの追加

デバイスを追加する際には、システムが新規デバイスに割り当てるパスベースのデバイス名 (例えば、**/dev/sd** 名、**major:minor** 番号、および **/dev/disk/by-path** 名など) がすでに削除されているデバイスで以前に使用されていた可能性があることに注意してください。これまでに使用されたパスベースのデバイス名がある場合、これに対する古い参照がすべて削除されていることを確認してください。そうしないと、新規デバイスが古いデバイスとして誤って認識されてしまう可能性があります。

**手順26.3 ストレージデバイスまたはパスの追加**

1. ストレージデバイスまたはパスを追加する際の最初のステップは、その新規デバイスへのアクセス、または既存デバイスまでの新規パスへのアクセスを物理的に有効にすることです。これは、ファイバーチャネルまたは iSCSI ストレージサーバーでベンダー固有のコマンドを使用して実行します。これを行う際には、使用するホストに表示される新規ストレージ用の LUN の値に注意してください。ストレージサーバーがファイバーチャネルの場合は、そのストレージサーバーの *World Wide Node Name* (WWNN) を記録し、ストレージサーバー上のすべてのポートに単一の WWNN が使用されるかどうかを判別します。そうでない場合は、新規の LUN へのアクセスには、各ポートの *World Wide Port Name* (WWPN) が使用されることに注意してください。

2. 次に、オペレーティングシステムに新規のストレージデバイス、または既存デバイスへのパスを認識させるようにします。使用するコマンドは以下のとおりです。

```
$ echo "c t l" >  /sys/class/scsi_host/hosth/scan
```

上記のコマンドで、***h*** は　HBA 番号を、***c*** は HBA 上のチャネルを示し、***t*** は SCSI のターゲット ID を、***l*** は LUN を示します。

**注記**

このコマンドの古い形式である、**echo "scsi add-single-device 0 0 0 0" > /proc/scsi/scsi** は廃止予定になっています。

a. ファイバーチャネルハードウェアの中には、RAID アレイに新たに作成される LUN が *Loop Initialization Protocol* (LIP) オペレーションが実行されるまでオペレーティングシステムから確認できないものもあります。確認する方法については「ストレージの相互接続をスキャンする」を参照してください。

**重要**

LIP が必要な場合、このオペレーションを実行している間は I/O を停止する必要があります。

b. 新しい LUN が RAID アレイに追加されているのにもかかわらず、オペレーティングシステムで設定されていない場合、*sg3_utils* パッケージに含まれている **sg_luns** コマンドを使用して、LUN のリストがアレイによってエクスポートされていることを確認してください。これにより、RAID アレイに対して **SCSI REPORT LUNS** コマンドが実行され、現在ある LUN のリストが返されます。

すべてのポートに単一の WWNN を実装するファイバーチャネルストレージサーバーでは、**sysfs** で WWNN を検索することにより、正しい値の ***h***、***c***、および***t*** (HBA 番号、HBA チャネルおよび SCSI ターゲット ID) を判別できます。

**例26.4 h、c、およびt の正しい値を判別**

たとえば、ストレージサーバーの WWNN が **0x5006016090203181** の場合、以下を使用します。

```
$ grep 5006016090203181 /sys/class/fc_transport/*/node_name
```

これにより、以下のような出力が表示されます。

```
/sys/class/fc_transport/target5:0:2/node_name:0x5006016090203181
/sys/class/fc_transport/target5:0:3/node_name:0x5006016090203181
/sys/class/fc_transport/target6:0:2/node_name:0x5006016090203181
/sys/class/fc_transport/target6:0:3/node_name:0x5006016090203181
```

上記は、このターゲットに対して 4 つのファイバーチャネルのルート (2 つの単一チャネル HBA のそれぞれが 2 つのストレージポートへのアクセスが設定されている) があることを示しています。LUN の値が **56** であると想定すると、以下のコマンドで最初のパスが設定されます。

```
$ echo "0 2 56" >  /sys/class/scsi_host/host5/scan
```

新規デバイスへの各パスに対してこれを実行する必要があります。

すべてのポートに対する単一の WWNN を実装しないファイバーチャネルのストレージサーバーでは、**sysfs** 内でそれぞれの WWPN を検索することによって、正しい HBA 番号、HBA チャネル、および SCSI ターゲット ID を判別することができます。

HBA 番号、HBA チャネル、および SCSI ターゲット ID を判別するための別の方法として、新規デバイスと同じパス上にすでに設定してある別のデバイスを参照する方法があります。これは、**lsscsi**、**scsi_id**、**multipath -l**、および **ls -l /dev/disk/by-*** などの様々なコマンドで達成できます。この情報、および新規デバイスの LUN 番号は、上記に示してあるように新規デバイスへのパスの探索とその設定に使用することができます。

3. デバイスへすべての SCSI パスを追加した後には、**multipath** コマンドを実行して、デバイスが正しく設定されているかどうかチェックします。この時点で、デバイスは、**md**、LVM、**mkfs**、または **mount** などに追加することができます。

上述のステップに従うと、デバイスは実行中のシステムに安全に追加することができます。これを実行する際に他のデバイスへの I/O を停止する必要はありません。SCSI バスの再スキャン (またはリセット) を伴う他の手順は、これによりオペレーティングシステムが現在のデバイスの接続状態を反映するように状態の更新を行うため、ストレージの I/O が実行中の場合は推奨されません。

## 26.7. イーサネットインターフェース上のファイバーチャネルの設定

ファイバーチャネルオーバーイーサネット (FCoE) インターフェースのセットアップと導入には、2 つのパッケージが必要です。

- **fcoe-utils**
- **lldpad**

これらのパッケージがインストールされた後は、以下の手順を実行して仮想 LAN (VLAN) 上で FCoE を有効にします。

**手順26.4 FCoE を使用するためのイーサネットインターフェースを設定**

1. FCoE をサポートするイーサネットデバイスの名前に、既存のネットワークスクリプト (例:**/etc/fcoe/cfg-eth0**) をコピーして、新規の VLAN を設定します。これにより、設定するデフォルトのファイルが提供されます。FCoE デバイスが**eth*X*** である場合、以下を実行します。

   ```
   # cp /etc/fcoe/cfg-eth0  /etc/fcoe/cfg-ethX
   ```

   必要に応じて、**cfg-eth*X*** の内容を変更してください。ハードウェア DCBX クライアントを実装するネットワーキングインターフェースの場合、**DCB_REQUIRED** は **no** に設定する必要があることに注意してください。

2. 起動中にデバイスが自動的にロードするように設定するには、対応する **/etc/sysconfig/network-scripts/ifcfg-eth*X*** ファイル内で**ONBOOT=yes** を設定します。たとえば、FCoE デバイスが eth2 の場合、**/etc/sysconfig/network-scripts/ifcfg-eth2** と編集します。

3. 以下のコマンドを使用して、データセンターブリッジングデーモン (**dcbd**) を開始します。

   ```
   # /etc/init.d/lldpad start
   ```

4. ハードウェア DCBX クライアントを実装するネットワーキングインターフェースの場合は、このステップを省略し、次に進んでください。

   ソフトウェア DCBX クライアントが必要なインターフェースの場合は、以下のコマンドを使用して、イーサネットインターフェース上のデータセンターブリッジングを有効にします。

   ```
   # dcbtool sc ethX dcb on
   ```

   次に、以下を実行してイーサネットインターフェース上で FCoE を有効にします。

   ```
   # dcbtool sc ethX app:fcoe e:1
   ```

**注記**

これらのコマンドは、イーサネットインターフェース用の **dcbd** の設定が変更されていなかった場合にのみ機能します。

5. 以下を使用して、FCoE デバイスをロードします。

```
# ifconfig ethX up
```

6. 次のコマンドで FCoE を開始します。

```
# service fcoe start
```

構成中の他のすべての設定が正しければ、まもなく FCoE デバイスが表示されます。設定した FCoE デバイスを表示するには、以下を実行します。

```
# fcoeadm -i
```

FCoE を使用できるようにイーサネットインターフェースを正しく設定したら、スタートアップ時に FCoE と **lldpad** が実行されるように設定することを推奨します。これを実行するには、以下のように **chkconfig** を使用します。

```
# chkconfig lldpad on
```

```
# chkconfig fcoe on
```

**警告**

ソフトウェアベースの DCB または LLDP は、DCB を実装する CNA 上では実行しないでください。

一部の CNA (Combined Network Adapters 結合ネットワークアダプター) はファームウェア内でデータセンターブリッジング (DCB) プロトコルを実装します。DCB プロトコルは、特定のネットワークリンク上に DCB のオリジネーターが 1 つだけ存在すると想定します。このことは、DCB または、Link Layer Discovery Protocol (LLDP) のより高いレベルのソフトウェア実装が、DCB を実装する CNA 上では無効にされる必要があることを意味します。

**注記**

**systemctl stop fcoe.service** を使用すると、デーモンは停止しますが、FCoE インターフェースの設定は解除されません。これを実行するには、スーパーユーザーが **systemctl -s SIGHUP kill fcoe.service** コマンドを使用する必要があります。

### 26.7.1. ファイバーチャネルオーバーイーサネット (FCoE) のターゲット設定

「イーサネットインターフェース上のファイバーチャネルの設定」に示されるように、FCoE での LUN のマウントだけでなく、他のマシンへの FCoE での LUN エクスポートもサポートされています。

**重要**

次に進む前に、「イーサネットインターフェース上のファイバーチャネルの設定」を参照して、基本的な FCoE 設定が完了しており、**fcoeadm -i** により、設定した FCoE インターフェースが表示されることを確認してください。

**手順26.5 FCoE ターゲットの設定**

1. FCoE ターゲットの設定には、**fcoe-target-utils** パッケージとその依存関係をインストールする必要があります。

```
# yum install fcoe-target-utils
```

2. FCoE ターゲットサポートは、LIO カーネルターゲットをベースにしており、userspace デーモンを必要としません。しかし、fcoe-target サービスで必要なカーネルモジュールをロードできるようにし、再起動時にも設定が保持されるようにする必要は依然としてあります。

```
# service fcoe-target start
```

```
# chkconfig fcoe-target on
```

3. FCoE ターゲットの設定は、**.conf** を編集する一般的な方法ではなく、**targetcli** ユーティリティーを使用して行います。設定を保存して、システムを再起動した場合に復元できるようにします。

```
# targetcli
```

**targetcli** は階層構成のシェルで、シェルのノード間を移動するには**cd** を使用します。また、**ls** は現在の設定ノードまたはそれ以下の内容を表示します。その他のオプションについては、コマンド **help** を利用することもできます。

4. ファイル、ブロックデバイスまたはパススルー SCSI デバイスを定義してバックストアとしてエクスポートします。

**例26.5 デバイス定義の例 1**

```
/> backstores/block create example1 /dev/sda4
```

これで、**/dev/sda4** ブロックデバイスにマッピングする**example1** というバックストアが作成されます。

**例26.6 デバイス定義の例 2**

```
/> backstores/fileio create example2 /srv/example2.img 100M
```

これで、指定のファイルにマッピングする **example2** というバックストアが作成されます。このファイルが存在しない場合は作成されます。ファイルサイズには K、M、G の略語が使用され、バッキングファイルが存在しない場合にのみファイルサイズが必要です。

**注記**

グローバルの **auto_cd_after_create** オプションはデフォルトでオンになっており、Create コマンドを実行すると現在の設定ノードを新たに作成されたオブジェクトに変更します。これは、**set global auto_cd_after_create=false** とすることで無効にできます。root ノードに戻るには **cd /** を使用してください。

5. FCoE インターフェースで FCoE ターゲットインスタンスを作成します。

```
/> tcm_fc/ create 00:11:22:33:44:55:66:77
```

FCoE インターフェースがシステム上にある場合、**create** の後にタブ補完を行うと、使用可能なインターフェースが表示されます。このインターフェースがない場合は、**fcoeadm -i** でアクティブなインターフェースを表示するようにしてください。

6. バックストアをターゲットインスタンスにマッピングします。

**例26.7 バックストアのターゲットインスタンスへのマッピング例**

```
/> cd tcm_fc/00:11:22:33:44:55:66:77
```

```
/> luns/ create /backstores/fileio/example2
```

7. FCoE イニシエーターからの LUN へのアクセスを許可します。

```
/> acls/ create 00:99:88:77:66:55:44:33
```

LUN が指定のイニシエーターにアクセスできるようになりました。

8. **exit** とタイプするか、**ctrl+D** を入力して **targetcli** を終了します。

**targetcli** を終了すると、デフォルトで設定は保存されます。しかし、**saveconfig** コマンドを使って明示的に保存することもできます。

さらに詳しくは、**targetcli** の man ページを参照してください。

## 26.8. 起動時に FCoE インターフェースを自動マウントする設定

**注記**

このセクションの情報は、Red Hat Enterprise Linux 6.1 の時点では **/usr/share/doc/fcoe-utils-*version*/README** でご覧いただけます。複数のマイナーリリースで生じる可能性のある変更についてはこの README を参照してください。

新しく検出されたディスクは、**udev** ルール、**autofs**、および他の類似の方法でマウントできます。しかし、特定のサービスが FCoE ディスクの起動時のマウントを要求することがあります。そのような場合には、FCoE ディスクは **fcoe** サービスの実行後**すぐに**、かつ FCoE ディスクを要求するサービスの開始**前に** マウントされる必要があります。

FCoE ディスクを起動時に自動マウントするように設定するには、正式な FCoE マウントコードを **fcoe** サービスのスタートアップスクリプトに追加します。**fcoe** のスタートアップスクリプトは、**/etc/init.d/fcoe** です。

FCoE マウントコードは、単純フォーマットの FCoE ディスク、LVM、またはマルチパスデバイスのノードを使用しているかどうかなどのシステム設定ごとに異なります。

**例26.8 FCoE マウントコード**

以下は、**/etc/fstab** 内のワイルドカードで指定されたファイルシステムをマウントするための FCoE マウントコードのサンプルです。

```
mount_fcoe_disks_from_fstab()
 {
     local timeout=20
     local done=1
     local fcoe_disks=($(egrep 'by-path\/fc-.*_netdev' /etc/fstab | cut
-d ' ' -f1))

     test -z $fcoe_disks && return 0

     echo -n "Waiting for fcoe disks . "
     while [ $timeout -gt 0 ]; do
  for disk in ${fcoe_disks[*]}; do
   if ! test -b $disk; then
    done=0
    break
   fi
  done

  test $done -eq 1 && break;
  sleep 1
  echo -n ". "
  done=1
  let timeout--
     done

     if test $timeout -eq 0; then
  echo "timeout!"
     else
  echo "done!"
     fi

     # mount any newly discovered disk
     mount -a 2>/dev/null
 }
```

**mount_fcoe_disks_from_fstab** 関数は、**fcoe** サービススクリプトが**fcoemon** デーモンを開始した**後に** 呼び出される必要があります。この関数が、**/etc/fstab** 内の以下のパスで指定されている FCoE ディスクをマウントします。

```
/dev/disk/by-path/fc-0xXX:0xXX /mnt/fcoe-disk1 ext3  defaults,_netdev
0 0
/dev/disk/by-path/fc-0xYY:0xYY /mnt/fcoe-disk2 ext3  defaults,_netdev
0 0
```

**fc-** サブストリングと **_netdev** サブストリングを持つエントリーは、**mount_fcoe_disks_from_fstab** 関数が FCoE ディスクマウントエントリリーを識別できるようにします。**/etc/fstab** のエントリーについてさらに詳しくは、**man 5 fstab** を参照してください。

**注記**

**fcoe** サービスは FCoE ディスク検出のタイムアウトを実装しません。そのため、FCoE マウントコードでは独自のタイムアウト期間を実装する必要があります。

## 26.9. ストレージの相互接続をスキャンする

1 つまたは複数の相互接続のリセットやスキャンを実行できるコマンドがいくつかあり、1 回の動作で複数のデバイスを追加したり削除したりできる可能性があります。この種のスキャンは I/O 動作のタイムアウトによる遅延の原因になり、予期せぬデバイスの削除を招く恐れがあるため破壊的な影響を与える可能性があります。このため、スキャンの使用は **必要な場合のみ** とすることを推奨します。さらに、ストレージの相互接続をスキャンする場合には次の制約に注意してください。

1. 手順を実行する前に、対象となる相互接続上の I/O はすべて一時停止してフラッシュする必要があります。また、スキャン結果の確認は I/O を再開する前に行ってください。

2. デバイスの削除と同様、システムがメモリー不足になっている状態での相互接続スキャンは推奨されません。メモリーのレベルを確認するには **vmstat 1 100** コマンドを実行します。100 中 10 を超えるサンプルでメモリー空き領域が合計メモリーの 5% を下回る場合には相互接続のスキャンは推奨されません。また、swap 機能がアクティブになっている (**vmstat** 出力内の **si** と **so** の欄がゼロ以外の値) 場合のスキャンも推奨されません。**free** コマンドもメモリー合計を表示することができます。

ストレージの相互接続をスキャンするには、次のコマンドを使用できます。

**echo "1" > /sys/class/fc_host/host/issue_lip**

*Loop Initialization Protocol* (*LIP*) を実行してから相互接続をスキャンし、現在バス上にあるデバイスを反映するよう SCSI 層を更新します。LIP は基本的にはバスのリセットであり、デバイスを追加したり削除したりすることになります。 この手順は、ファイバーチャネルの相互接続で新しい SCSI ターゲットを設定する場合に必要になります。

**issue_lip** は非同期の動作となることに注意してください。このコマンドはスキャン全体が終了する前に完了してしまうことがあります。**/var/log/messages** を監視して終了時点を確認する必要があります。

**issue_lip** に対応しているのは **lpfc** ドライバー、**qla2xxx** ドライバー、および **bnx2fc** ドライバーです。Red Hat Enterprise Linux 内の各ドライバーで対応している API 機能については、表26.1「ファイバーチャネルの API 機能」 を参照してください。

**/usr/bin/rescan-scsi-bus.sh**

Red Hat Enterprise Linux 5.4 よりこのスクリプトが同梱されています。デフォルトではシステム上のすべての SCSI バスがこのスクリプトによってスキャンされ、バス上の新しいデバイスを反映するよう SCSI レイヤーが更新されます。このスクリプトは、デバイスの削除や LIP の実行

を可能にする追加のオプションも提供します。このスクリプトについてさらに詳しくは (既知の問題も含む) 「rescan-scsi-bus.sh による論理ユニットの追加と削除」を参照してください。

**echo "- - -" > /sys/class/scsi_host/host*h*/scan**

ストレージのデバイスまたはパスを追加する方法として「ストレージデバイスまたはパスの追加」で説明しているコマンドと同じものになります。ただし、この場合にはチャネル番号、SCSI ターゲット ID、および LUN などの値がワイルドカードになります。あらゆる識別子とワイルドカードの組み合わせが可能で、必要に応じて対象を絞り込むことも広げることも可能です。この手順では LUN を追加しますが、その削除は行いません。

**rmmod driver-name** or **modprobe driver-name**

ドライバーによって管理されているすべての相互接続の状態を完全に再初期化します。極端な手段になりますが、特定の状況に適していると言えます。たとえば、異なるモジュールパラメーター値でドライバーを再起動する場合などに使用することができます。

## 26.10. iSCSI 検出の設定

デフォルトの iSCSI 設定ファイルは **/etc/iscsi/iscsid.conf** です。このファイルには **iscsid** と **iscsiadm** によって使用される iSCSI の設定が含まれます。

ターゲットの検出時に、**iscsiadm** ツールは **/etc/iscsi/iscsid.conf** 内の設定を使用して 2 種類の記録を作成します。

**ノードの記録 - /var/lib/iscsi/nodes**

ターゲットへのログイン時に **iscsiadm** によってこのファイル内の設定が使用されます。

**検出の記録 - /var/lib/iscsi/*discovery_type***

同じターゲットに対して検出を行う場合、**iscsiadm** によってこのファイル内の設定が使用されます。

検出に別の設定を使用する場合は、まず現在の検出記録を削除します (例: **/var/lib/iscsi/*discovery_type***)。これを実行するには、次のコマンドを使います。

# iscsiadm -m discovery -t *discovery_type* -p *target_IP*:*port* -o delete [5]

***discovery_type*** は **sendtargets**、**isns**、**fw** のいずれかになります。

検出の各種タイプについては **man iscsiadm** の 『DISCOVERY TYPES』 セクションを参照してください。

検出記録の設定を再設定する方法は 2 通りあります。

- 検出を行う前に **/etc/iscsi/iscsid.conf** ファイルを直接編集します。検出の設定は、プレフィックス **discovery** を使用します。それらを表示するには、以下を実行します。

```
# iscsiadm -m discovery -t discovery_type -p target_IP:port
```

- または、**iscsiadm** を使って以下のように検出記録の設定を直接変更することもできます。

```
# iscsiadm -m discovery -t discovery_type -p target_IP:port -o update
-n setting -v %value
```

利用できる各 ***setting*** および有効な ***value*** に関しては **man iscsiadm** を参照してください。

検出の設定を行うと、その後のターゲット検出の試行からは新しい設定が使用されるようになります。新しい iSCSI ターゲットのスキャン方法については、「iSCSI 相互接続をスキャンする」を参照してください。

iSCSI ターゲットの検出を設定する方法についてさらに詳しくは、**iscsiadm** と **iscsid** の各 **man** ページを参照してください。**/etc/iscsi/iscsid.conf** ファイルには適切な設定構文に関するサンプルも複数含まれています。

## 26.11. iSCSI オフロードとインターフェースバインディングの設定

本章では、ソフトウェア iSCSI を使用する際にセッションを NIC ポートに結合させるための iSCSI インターフェースを設定する方法について説明します。また、オフロード機能に対応する Chelsio、Broadcom、 ServerEngines などのネットワークデバイスでインターフェースを使用できるよう設定を行う方法についても説明しています。

iSCSI が結合に使用するパスや NIC を指定できるようネットワークサブシステムを設定することができます。たとえば、ポータルと NIC が別々のサブネットにセットアップされている場合、結合用の iSCSI インターフェースを手作業で設定する必要はありません。

結合用の iSCSI インターフェースを設定する前に、まず次のコマンドを実行します。

```
$ ping -I ethX target_IP
```

**ping** が失敗する場合はセッションを NIC に結合することはできません。このような場合にはネットワーク設定をまず確認してください。

### 26.11.1. 利用可能な iface の設定を表示する

Red Hat Enterprise Linux 5.5 から、以下のような iSCSI イニシエーターの実装に対する iSCSI オフロードとインターフェースのバインディングに対応しています。

- *ソフトウェア iSCSI* — **scsi_tcp** モジュールや **ib_iser** モジュールと同様、このスタックはセッションごとに iSCSI ホストインスタンス (**scsi_host**) を割り当てます。接続は 1 セッションに 1 接続です。結果として **/sys/class_scsi_host** と **/proc/scsi** は、ログインしている各接続/セッションに対して 1 つの **scsi_host** を報告します。
- *オフロード iSCSI* — Chelsio **cxgb3i**、Broadcom **bnx2i**、ServerEngines **be2iscsi** モジュールなどと同様に、このスタックは各 PCI デバイスに対して **scsi_host** を割り当てます。このため、ホストのバスアダプター上の各ポートは、HBA ポートごとに別々の **scsi_host** を持つ異なった PCI デバイスとして表示されます。

いずれのタイプのイニシエーター実装を管理する場合も **iscsiadm** は **iface** 構造を使用します。この構造では、複数のセッションの結合に使用する各 HBA ポート、ソフトウェア iSCSI、またはネットワークデバイス (**eth*X***) などの **/var/lib/iscsi/ifaces** に **iface** の設定を記載しておく必要があります。

利用可能な **iface** の設定を表示するには、**iscsiadm -m iface** を実行します。次のような形式で **iface** の情報が表示されます。

```
iface_name
transport_name,hardware_address,ip_address,net_ifacename,initiator_name
```

それぞれの値または設定については次の表を参照してください。

**表26.2 iface の設定**

| 設定 | 詳細 |
| --- | --- |
| **iface_name** | **iface** の設定名 |
| **transport_name** | ドライバー名 |
| **hardware_address** | MAC アドレス |
| **ip_address** | このポートに使用する IP アドレス |
| **net_iface_name** | ソフトウェア iSCSI セッションの **vlan** またはエイリアス結合に使用する名前 (iSCSI オフロードの場合、再起動するとこの値は維持されないため **net_iface_name** は **<empty>** になります) |
| **initiator_name** | **/etc/iscsi/initiatorname.iscsi** で定義されているイニシエーターのデフォルト名を無効にする場合に使用 |

**例26.9 iscsiadm -m iface コマンドの出力例**

以下に **iscsiadm -m iface** コマンドの出力例を示します。

```
iface0 qla4xxx,00:c0:dd:08:63:e8,20.15.0.7,default,iqn.2005-
06.com.redhat:madmax
iface1 qla4xxx,00:c0:dd:08:63:ea,20.15.0.9,default,iqn.2005-
06.com.redhat:madmax
```

ソフトウェア iSCSI の場合、**iface** 設定には固有の名前を持たせなければなりません (65 文字未満)。オフロード機能に対応するネットワークデバイスの **iface_name** は ***transport_name.hardware_name*** の形式で表示されます。

**例26.10 Chelsio ネットワークカードを使用する場合の iscsiadm -m iface の出力**

たとえば、Chelsio ネットワークカードを使用しているシステムで **iscsiadm -m iface** コマンドを実行した場合の出力例は以下のようになります。

```
default tcp,<empty>,<empty>,<empty>,<empty>
iser iser,<empty>,<empty>,<empty>,<empty>
cxgb3i.00:07:43:05:97:07 cxgb3i,00:07:43:05:97:07,<empty>,<empty>,
<empty>
```

特定の **iface** 設定をさらにわかりやすい方法で表示することもできます。これを実行するには、**-I *iface_name*** オプションを使用します。これにより、次のような形式で設定が表示されます。

```
iface.setting = value
```

**例26.11 Chelsio 集中型ネットワークアダプターによる iface 設定を使用**

前述の例と同じ Chelsio 統合型ネットワークアダプターによる **iface** 設定は次のように表示されます (**iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07**) 。

```
# BEGIN RECORD 2.0-871
iface.iscsi_ifacename = cxgb3i.00:07:43:05:97:07
iface.net_ifacename = <empty>
iface.ipaddress = <empty>
```

```
iface.hwaddress = 00:07:43:05:97:07
iface.transport_name = cxgb3i
iface.initiatorname = <empty>
# END RECORD
```

### 26.11.2. ソフトウェア iSCSI 用 iface の設定

前述のように、セッションをバインドするために使用される各ネットワークオブジェクトには**iface** の設定が必要になります。

その前に

ソフトウェア iSCSI 用の **iface** 設定を作成するには、以下のコマンドを実行します。

```
# iscsiadm -m iface -I iface_name --op=new
```

これは、指定された ***iface_name*** を使用して、新規の**空の iface** 設定を作成します。既存の **iface** 設定が、すでに同じ ***iface_name*** を持っている場合は、それが新規の空の設定で上書きされます。

**iface** 構成の特定の設定を行うには、以下のコマンドを使用します。

```
# iscsiadm -m iface -I iface_name --op=update -n iface.setting -v
hw_address
```

**例26.12 iface0 の MAC アドレスを設定します**

たとえば、**iface0** の MAC アドレス (**hardware_address**) を **00:0F:1F:92:6B:BF** に設定するには、以下を実行します。

```
# iscsiadm -m iface -I iface0 --op=update -n iface.hwaddress -v
00:0F:1F:92:6B:BF
```

**警告**

**default** または **iser** を **iface** の名前として使用しないでください。これらのストリングは後方互換性について **iscsiadm** で使用される特別な値です。**default** または **iser** という名前の手動で作成された **iface** 構成は、いずれも後方互換性を無効にします。

### 26.11.3. iSCSI Offload 用 iface の設定

デフォルトでは、**iscsiadm** は、Chelsio、Broadcom、および ServerEngines ポートのそれぞれに対して **iface** 設定を作成します。利用可能な **iface** 設定を表示するには、ソフトウェア iSCSI 用に使うコマンドと同様に **iscsiadm -m iface** を使用します。

iSCSI オフロード用ネットワークカードの **iface** を使用する前に、まずデバイスが使用すべき IP アドレス (***target_IP***[5]) を設定します。**be2iscsi** ドライバーを使用する ServerEngines デバイス (つまり ServerEngines iSCSI HBA) の場合、IP アドレスは ServerEngines BIOS のセットアップ画面で設定されます。

Chelsio と Broadcom デバイスの場合、IP アドレスを設定する手順は、その他の **iface** 設定と同じです。**iface** の IP アドレスを設定するには、以下を使用します。

```
# iscsiadm -m iface -I iface_name -o update -n iface.ipaddress -v
target_IP
```

**例26.13 Chelsio カードの iface IP アドレスの設定**

たとえば、Chelsio カード (**iface** 名 **cxgb3i.00:07:43:05:97:07**) の **iface** IP アドレスを **20.15.0.66** に設定するには、以下を使用します。

```
# iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07 -o update -n
iface.ipaddress -v 20.15.0.66
```

### 26.11.4. iface のポータルに対する結合/結合解除

相互接続のスキャンに **iscsiadm** を使用すると、**iscsiadm** ユーティリティーは **/var/lib/iscsi/ifaces** 内の各 **iface** 構成の **iface.transport** 設定をまず最初にチェックします。次にこの **iface.transport** 設定が **tcp** になっている **iface** 構成に検出したポータルを結合します。

この動作は互換性を目的として実装されました。この動作を無効にする場合は、次のように **-I *iface_name*** を使って **iface** に結合させるポータルを指定します。

```
# iscsiadm -m discovery -t st -p target_IP:port -I iface_name -P 1
[5]
```

デフォルトでは、オフロード機能を使用する **iface** 構成にはポータルの自動結合は行われません。オフロード機能を使用する **iface** 構成の場合、**iface.transport** に **tcp** が設定されることはないためです。このため、Chelsio ポート、Broadcom ポート、ServerEngines ポートなどの **iface** 構成には、手作業で検出されたポータルに結合する必要があります。

既存の **iface** にポータルを一切結合しないようにすることもできます。以下のように *iface_name* に **default** を使用します。

```
# iscsiadm -m discovery -t st -p IP:port -I default -P 1
```

ターゲットと **iface** 間の結合を削除する場合は次を使用します。

```
# iscsiadm -m node -targetname proper_target_name -I iface0 --op=delete
      [6]
```

特定の **iface** の結合をすべて削除する場合は次を使用します。

```
# iscsiadm -m node -I iface_name --op=delete
```

特定のポータルに対する結合を削除する場合は次を使用します。

```
# iscsiadm -m node -p IP:port -I iface_name --op=delete
```

**注記**

**/var/lib/iscsi/iface** 内に **iface** 構成が定義されておらず、**-I** オプションが使用されていない場合は、**iscsiadm** により、ネットワークサブシステムが特定ポータルが使用するデバイスを決定できることになります。

## 26.12. iSCSI 相互接続をスキャンする

iSCSI の場合、新しいストレージが追加されたことを示す iSCSI 非同期イベントがターゲットによって送信されると、スキャンが自動的に行われます。Cisco MDS™ および EMC Celerra™ はこの機能に対応しています。

ただし、ターゲットが iSCSI 非同期イベントを送信しない場合は **iscsiadm** ユーティリティーを使って手作業でスキャンを行う必要があります。スキャンを行う前に、適切な **--targetname --portal** の値をまず取得する必要があります。ご使用のデバイスモデルが 1 ターゲットに 1 論理ユニットとポータルのみをサポートしている場合は、以下のように **iscsiadm** を使ってホストに対し **sendtargets** コマンドを発行します。

```
# iscsiadm -m discovery -t sendtargets -p target_IP:port
[5]
```

出力は以下のような形式になります。

```
target_IP:port,target_portal_group_tag proper_target_name
```

**例26.14 iscsiadm を使って sendtargets コマンドを発行する**

たとえば、***proper_target_name*** が **iqn.1992-08.com.netapp:sn.33615311** で ***target_IP:port*** が **10.15.85.19:3260** になるターゲットでは出力は次のようになります。

```
10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

上記の例では、 ターゲットにはポータルが 2 つあり、 ***target_ip:port*** にはそれぞれ **10.15.84.19:3260** と **10.15.85.19:3260** を使用しています。

各セッションに使用される **iface** 構成を表示させる場合は **-P 1** オプションを追加します。このオプションにより、セッション情報が以下のようにツリー形式で表示されます。

```
    Target: proper_target_name
        Portal: target_IP:port,target_portal_group_tag
           Iface Name: iface_name
```

**例26.15 iface 構成の表示**

たとえば、**iscsiadm -m discovery -t sendtargets -p 10.15.85.19:3260 -P 1** の場合、出力は以下のようになります。

```
Target: iqn.1992-08.com.netapp:sn.33615311
    Portal: 10.15.84.19:3260,2
       Iface Name: iface2
    Portal: 10.15.85.19:3260,3
       Iface Name: iface2
```

上記から、ターゲットの **iqn.1992-08.com.netapp:sn.33615311** はその **iface** に **iface2** を使用していることがわかります。

しかし、デバイスモデルの中には 1 つのターゲットで複数の論理ユニットやポータルに対応できるものがあります (EMC および Netapp など)。この場合、まず **sendtargets** コマンドをホストに発行して、ターゲット上の新しいポータルを検索します。次に既存のセッションを再スキャンするのに以下を使用します。

```
# iscsiadm -m session --rescan
```

セッションの ***SID*** 値を以下のようにして指定することで特定セッションの再スキャンを実行することもできます。

```
# iscsiadm -m session -r SID --rescan
      [7]
```

複数のターゲットに対応するデバイスの場合、**sendtargets** コマンドをその複数ホストに対して発行し、**各ターゲットごとに** 新しいポータルを探す必要があります。次に既存の複数セッションを再スキャンして、それらのセッションにある新しい複数の論理ユニットを探します (**--rescan** オプションを使用)。

**重要**

**--targetname** と **--portal** の値の取得に使用する **sendtargets** コマンドにより **/var/lib/iscsi/nodes** データベースのコンテンツが上書きされます。このデータベースは **/etc/iscsi/iscsid.conf** 内の設定を使って再設定されることになります。ただし、セッションが現在ログイン中で使用されている場合は再設定は行われません。

新しいターゲットやポータルの追加や古いターゲットやポータルの削除を安全に行うには、**-o new** オプションと **-o delete** オプションをそれぞれ使用します。たとえば、新しいターゲットやポータルを **/var/lib/iscsi/nodes** を上書きせずに追加する場合は次のコマンドを使用します。

```
iscsiadm -m discovery -t st -p target_IP -o new
```

検索中にターゲットが表示しなかった **/var/lib/iscsi/nodes** のエントリーを削除するには、次を使用します。

```
iscsiadm -m discovery -t st -p target_IP -o delete
```

次のようにして、両方のタスクを同時に行うこともできます。

```
iscsiadm -m discovery -t st -p target_IP -o delete -o new
```

**sendtargets** コマンドにより次のような出力が生成されます。

```
ip:port,target_portal_group_tag proper_target_name
```

**例26.16 sendtargets コマンドの出力**

たとえば、ターゲット、論理ユニット、およびポータルが 1 つで ***target_name*** が **equallogic-iscsi1** になるデバイスの場合、出力は次のようになります。

```
10.16.41.155:3260,0 iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-
63aff113e344a4a2-dl585-03-1
```

***proper_target_name*** と ***ip:port,target_portal_group_tag*** は「iSCSI API」にある同じ名前の値と同一になることに注意してください。

これで iSCSI デバイスの手作業によるスキャンを行うために必要な **--targetname** と **--portal** の適切な値を確認できました。次のコマンドを実行します。

```
# iscsiadm --mode node --targetname proper_target_name --portal
ip:port,target_portal_group_tag \ --login

       [8]
```

**例26.17 iscsiadm コマンド**

前述の例を使用すると (***proper_target_name*** は **equallogic-iscsi1**)、このコマンドは以下のようになります。

```
# iscsiadm --mode node --targetname  \ iqn.2001-05.com.equallogic:6-
8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1  \ --portal
10.16.41.155:3260,0 --login[8]
```

## 26.13. iSCSI ターゲットにログインする

「iSCSI」に記載されているように、ターゲットの検出やログインを行うには iSCSI サービスを実行していなければなりません。iSCSI サービスを起動するには次を実行します。

```
# service iscsi start
```

このコマンドを実行すると、iSCSI の **init** スクリプトは **node.startup** の設定が **automatic** に設定されているターゲットに自動的にログインします。**automatic** は、すべてのターゲットの **node.startup** 設定のデフォルト値になります。

ターゲットへの自動ログインを行わないようにするには、**node.startup** の設定を **manual** にします。これを実行するには、次のコマンドを実行します。

```
# iscsiadm -m node --targetname proper_target_name -p target_IP:port -o
update -n node.startup -v manual
```

記録全体を削除することによっても自動ログインを回避できます。これを実行するには、次を実行します。

```
# iscsiadm -m node --targetname proper_target_name -p target_IP:port -o
delete
```

ネットワーク上の iSCSI デバイスからファイルシステムを自動的にマウントするには、**/etc/fstab** にマウント用のパーティションエントリーを追加して **_netdev** オプションを付けます。たとえば、起動時に iSCSI デバイスの **sdb** を **/mount/iscsi** に自動的にマウントする場合は、次の行を**/etc/fstab** に追加します。

```
/dev/sdb /mnt/iscsi ext3 _netdev 0 0
```

iSCSI ターゲットに手動でログインする場合は次のコマンドを使用します。

```
# iscsiadm -m node --targetname proper_target_name -p target_IP:port -l
```

**注記**

***proper_target_name*** と ***target_IP:port*** はターゲットのフルネームと IP アドレス/ポートの組み合わせを参照します。詳細については、「iSCSI API」 および 「iSCSI 相互接続をスキャンする」 を参照してください。

## 26.14. オンラインの論理ユニットのサイズを変更する

ほとんどの場合、 オンラインの *論理ユニット* のサイズを完全に変更するには 2 つの作業が必要になります。 論理ユニット自体のサイズを変更する作業、 そしてそのサイズ変更を該当するマルチパスデバイスに反映させる作業です (システムでマルチパス機能を有効にしている場合)。

オンラインの論理ユニットのサイズ変更は、 ストレージデバイスのアレイ管理インターフェースから論理ユニットのサイズを変更するところから始めます。 手順はアレイによって異なるため、 詳細についてはストレージアレイの製造元が提供しているドキュメントを参照してください。

**注記**

オンラインのファイルシステムのサイズを変更する場合には、 パーティション設定しているデバイス上にはそのファイルシステムを配置させないようにしてください。

### 26.14.1. ファイバーチャネルの論理ユニットのサイズを変更する

オンライン論理ユニットのサイズを変更したら、 システムによって最新のサイズが検出されるよう論理ユニットの再スキャンを行います。 ファイバーチャネルの論理ユニットの場合なら次のコマンドを使用します。

```
$ echo 1 > /sys/block/sdX/device/rescan
```

**重要**

マルチパス機能を使用しているシステムのファイバーチャネル論理ユニットを再スキャンする場合は、 マルチパス設定されている論理ユニットのパスを表す各 sd デバイス (**sd1**、 **sd2** など) に対して前述のコマンドを実行していきます。 マルチパス設定されている論理ユニットのパスのデバイスを確認するには **multipath -ll** を使用して、 サイズ変更した論理ユニットに一致するエントリを探します。 サイズ変更した論理ユニットに一致するエントリを見つける場合、 エントリの WWID を参照すると簡単です。

### 26.14.2. iSCSI の論理ユニットのサイズを変更する

オンライン論理ユニットのサイズを変更したら、 システムによって最新のサイズが検出されるよう論理ユニットの再スキャンを行います。 iSCSI デバイスの論理ユニットの場合なら次のコマンドを使用します。

```
# iscsiadm -m node --targetname target_name -R
[5]
```

***target_name*** はデバイスがある場所のターゲット名にします。

**注記**

次のコマンドを使っても iSCSI 論理ユニットを再スキャンすることができます。

```
# iscsiadm -m node -R -I interface
```

***interface*** はサイズ変更した論理ユニットに該当するインターフェース名です (**iface0** など)。

- 上記のコマンドは、**echo "- - -" > /sys/class/scsi_host/*host*/scan** コマンドと同じ動作で新しいデバイスのスキャンを行います (「iSCSI 相互接続をスキャンする」 を参照)。
- 上記のコマンドは、 **echo 1 > /sys/block/sdX/device/rescan** コマンドと同じ動作で新しい論理ユニットまたは変更した論理ユニットの再スキャンを行います。 ファイバーチャネルの論理ユニットを再スキャンする場合に使用したものと同じコマンドになります。

### 26.14.3. マルチパスデバイスのサイズを更新する

システムでマルチパス機能を有効にしている場合は、 論理ユニットサイズでの変更を該当するマルチパスデバイスに対しても反映させる必要があります (**論理ユニットのサイズ変更後**)。 Red Hat Enterprise Linux 5.3 (またはそれ以降) の場合なら **multipathd** を使って変更を反映させることができます。 まず、 **service multipathd status** で **multipathd** が実行中であることを確認します。**multipathd** が正常に動作していることを確認したら、 次のコマンドを実行します。

```
# multipathd -k"resize map multipath_device"
```

***multipath_device*** の変数は **/dev/mapper** 内にある該当デバイスのマルチパスエントリになります。 システム上でマルチパス機能がどのように設定されているかによって ***multipath_device*** の形式は次の 2 種類のいずれかになります。

- **mpath*X***: ***X*** は使用デバイスの該当エントリになります (**mpath0** など)

- WWID: **3600508b400105e21000900000490000** などです

サイズ変更した論理ユニットに該当するマルチパスエントリの確認には **multipath -ll** を実行します。システム内に既存しているマルチパスの全エントリー覧および該当デバイスのメジャー番号とマイナー番号が表示されます。

**重要**

***multipath_device*** に対してキュー待ちしているコマンドがある場合には、**multipathd -k"resize map *multipath_device*"** を使用しないでください。 つまり、**no_path_retry** パラメーター (**/etc/multipath.conf** 内) が **"queue"** に設定されていて、 デバイスへのアクティブなパスがない場合には、 このコマンドを使用しないでください。

システムで Red Hat Enterprise Linux 5.0 - 5.2 を使用している場合、 **multipathd** デーモンに論理ユニットに加えた変更を認識させるため (また調整を行わせるため) 次の手順を行う必要があります。

**手順26.6 該当のマルチパスデバイスのサイズを変更する (Red Hat Enterprise Linux 5.0 - 5.2 の場合に必須)**

1. 次のようにしてマルチパス設定しているデバイスのデバイスマッパーテーブルをダンプします。

   **dmsetup table *multipath_device***

2. ダンプしたデバイスマッパーテーブルを ***table_name*** という名前を付けて保存します。 このテーブルは後ほど再読み込みをしてから編集を行います。

3. デバイスマッパーテーブルの内容を確認します。 各行の先頭にある 2 つの番号はディスクの開始セクターと終了セクターを示しています。

4. デバイスマッパーターゲットを一時停止にします。

   **dmsetup suspend *multipath_device***

5. 先ほど保存したデバイスマッパーテーブル (***table_name***) を開きます。 2 番目の番号 (ディスクの終了セクター) を変更して 512 バイトセクターの新しい番号を反映させます。 たとえば、 新しいディスクサイズが 2GB なら 2 番目の番号を 4194304 に変更します。

6. 変更したデバイスマッパーテーブルを再読み込みします。

   **dmsetup reload *multipath_device table_name***

7. デバイスマッパーターゲットを再開します。

   **dmsetup resume *multipath_device***

マルチパス機能の詳細については 『Red Hat Enterprise Linux 7 DM Multipath』 ガイドを参照してください。

## 26.14.4. オンライン論理ユニットの Read/Write ステータスの変更

ストレージデバイスによっては、ユーザーが Read/Write (R/W) から Read-Only (RO) へ、RO から R/W へデバイスのステータスを変更できるものもあります。通常これは、ストレージデバイスの管理インターフェースから行います。オペレーティングシステムは、変更があっても、デバイスのステータスの表示を自動更新しません。本章に説明されている手順に従い、オペレーティングシステムがこの変更を認識するようにします。

以下のコマンドを実行して、デバイスの R/W ステータスに関するオペレーティングシステムの現在の

ビューを確認します。この際 XYZ は任意のデバイス識別子に置き換えてください。

```
# blockdev --getro /dev/sdXYZ
```

以下のコマンドは、Red Hat Enterprise Linux 7 でも利用できます。

```
# cat /sys/block/sdXYZ/ro 1 = read-only 0 = read-write
```

マルチパスを使用する場合は、**multipath -ll** コマンドからの出力の2行目にある**ro** または **rw**フィールドを参照してください。例えば、

```
36001438005deb4710000500000640000 dm-8 GZ,GZ500
[size=20G][features=0][hwhandler=0][ro]
\_ round-robin 0 [prio=200][active]
 \_ 6:0:4:1  sdax 67:16  [active][ready]
 \_ 6:0:5:1  sday 67:32  [active][ready]
\_ round-robin 0 [prio=40][enabled]
 \_ 6:0:6:1  sdaz 67:48  [active][ready]
 \_ 6:0:7:1  sdba 67:64  [active][ready]
```

R/W ステータスを変更するには、以下の手順に従います。

**手順26.7 R/W ステータスの変更**

1. RO から R/W へデバイスを移動するには、手順 2 を参照してください。

   R/W から RO へデバイスを移動するには、これ以上書き込みがされないようにします。アプリケーションを停止するか、適切なアプリケーション固有のアクションを使用して、書き込みが行われないように設定します。

   見処理の書き込み I/O がすべて完了しているか、以下のコマンドで確認します。

   ```
   # blockdev --flushbufs /dev/device
   ```

   任意の識別子で *device* を置き換えます。デバイスマッパーマルチパスは、これが**dev/mapper** のエントリとなります。例えば、**/dev/mapper/*mpath3*** などです。

2. ストレージデバイスの管理インターフェースを使用して、R/W から RO、または RO から R/W へステータスを変更します。この手順は、アレイごとに異なります。詳細情報は、該当のストレージアレイのベンダーが出している文書を参照してください。

3. デバイスを再度スキャンして、デバイスの R/W ステータスに関するオペレーティングシステムのビューを更新します。デバイスマッパーマルチパスを使用している場合、デバイスへのパスごとにこの再スキャンを行ってから、デバイスマップをリロードするようマルチパスに指示を出すコマンドを発行します。

   このプロセスについては、「論理ユニットの再スキャン」で詳しく説明されています。

### 26.14.4.1. 論理ユニットの再スキャン

「オンライン論理ユニットの Read/Write ステータスの変更」の記載の通り、オンライン論理ユニットの Read/Write ステータスを変更後、以下のコマンドを使用して、ステータスの更新をシステムが検出できるように論理ユニットを再スキャンします。

```
# echo 1 > /sys/block/sdX/device/rescan
```

マルチパス機能を使用するシステムで論理ユニットを再スキャンするには、マルチパス化された論理ユニットに対するパスを指す sd デバイスごとに上記のコマンドを実行してください。例えば、sd1、sd2、その他の sd デバイスにコマンドを実行します。どのデバイスがマルチパスユニットへのパスとなっているか判断するには、**multipath -11** を使用して、変更される論理ユニットと同じエントリを検索します。

**例26.18 multipath -11 コマンドの使用**

例えば、上記の **multipath -11** は WWID 36001438005deb4710000500000640000 の LUN のパスを表示します。この場合、以下を入力してください。

```
# echo 1 > /sys/block/sdax/device/rescan
# echo 1 > /sys/block/sday/device/rescan
# echo 1 > /sys/block/sdaz/device/rescan
# echo 1 > /sys/block/sdba/device/rescan
```

#### 26.14.4.2. マルチパスデバイスの R/W ステータスの更新

マルチパス機能が有効な場合、論理ユニットを再スキャンした後、ステータスの変更を論理ユニットに該当するマルチパスデバイスで反映させる必要があります。以下のコマンドで、マルチパスデバイスマップをリロードして、変更を反映します。

```
# multipath -r
```

次に、**multipath -11** コマンドを使用して変更を確定します。

#### 26.14.4.3. ドキュメンテーション

さらなる情報は、Red Hat ナレッジベースで確認いただけます。ナレッジベースへアクセスするには、https://www.redhat.com/wapps/sso/login.html?redirect=https://access.redhat.com/knowledge/ へ移動してログインします。次に、https://access.redhat.com/kb/docs/DOC-32850 の記事にアクセスしてください。

## 26.15. rescan-scsi-bus.sh による論理ユニットの追加と削除

**sg3_utils** パッケージは、必要に応じてホストの論理ユニット設定を自動的に更新する**rescan-scsi-bus.sh** スクリプトを提供します (デバイスがシステムに追加された後)。**rescan-scsi-bus.sh** スクリプトはまた、サポートされているデバイス上で **issue_lip** も実行できます。このスクリプトの使用法の詳細については、**rescan-scsi-bus.sh --help** を参照してください。

**sg3_utils** パッケージをインストールするには、**yum install sg3_utils** を実行します。

### rescan-scsi-bus.sh に関する既知の問題

**rescan-scsi-bus.sh** スクリプトを使用する際には、以下の既知の問題に注意してください。

- **rescan-scsi-bus.sh** が正常に機能するためには、最初にマップされる論理ユニットは**LUN0** でなければなりません。最初にマップされた論理ユニットが **LUN0** である場合にのみ、**rescan-scsi-bus.sh** はこれを検出できます。**--nooptscan** オプションを使用した場合でも、**rescan-scsi-bus.sh** は、最初にマップされた論理ユニットを検出しない限り、他の論理ユニットをスキャンできません。

- 論理ユニットが初めてマップされる場合には、競合状態により、**rescan-scsi-bus.sh** を 2 回実行することが必要になります。最初のスキャンでは、**rescan-scsi-bus.sh** は単に **LUN0** のみを追加するだけです。他のすべての論理ユニットは 2 回目のスキャンで追加されます。
- **--remove** オプションが使用されると、**rescan-scsi-bus.sh** スクリプト内のバグが、論理ユニット内の変化を認識するための機能を誤った方法で実行します。
- **rescan-scsi-bus.sh** スクリプトは、ISCSI の論理ユニットの削除を認識しません。

## 26.16. リンク切れの動作を修正する

本セクションでは、ファイバーチャネルまたは iSCSI のプロトコルのいずれかを使用しているデバイスに関するリンク切れの動作を修正する方法について説明します。

### 26.16.1. ファイバーチャネル

ドライバーがトランスポートの **dev_loss_tmo** コールバックを実装している場合、トランスポートの問題が検出されるとリンクを経由したデバイスへのアクセス試行がブロックされます。デバイスがブロックされているかどうかを確認するには次のコマンドを実行します。

```
$ cat /sys/block/device/device/state
```

デバイスがブロックされている場合は **blocked** が返されます。通常通りにデバイスが稼働している場合は **running** が返されます。

**手順26.8 リモートポートの状態を判別する**

1. リモートポートの状態を判別するには、次のコマンドを実行します。

   ```
   $ cat
   /sys/class/fc_remote_port/rport-H:B:R/port_state
   ```

2. リモートポート (およびこのポートを介してアクセスされるデバイス) がブロックされている場合は、このコマンドにより **Blocked** が返されます。通常通りにリモートポートが稼働している場合は **Online** が返されます。
3. **dev_loss_tmo** 秒以内に問題が解決されない場合、リモートポートおよびデバイスのブロックが解除され、そのデバイス上で実行されているすべての I/O (およびそのデバイスに送信される新規 I/O すべて) が失敗します。

**手順26.9 dev_loss_tmo を変更する**

- **dev_loss_tmo** の値を変更する場合は、ファイルに対して必要な値の**echo** を実行します。たとえば、**dev_loss_tmo** を 30 秒に設定するには、以下を実行します。

  ```
  $ echo 30 >
  /sys/class/fc_remote_port/rport-H:B:R/dev_loss_tmo
  ```

**dev_loss_tmo** についてさらに詳しくは、「ファイバーチャネル API」を参照してください。

リンク切れが **dev_loss_tmo** の値を超える場合、**scsi_device** および **sd*N*** デバイスは削除されます。通常、ファイバーチャネルのクラスはデバイスをそのままの状態にします。すなわち、**/dev/sd*x*** は、**/dev/sd*x*** のままになります。これは、ターゲットバインディングがファイバーチャネルドライバーによって保存されるためであり、ターゲットポートが返されると、SCSI アドレスが忠実に再作成されま

す。ただし、これは常に保証される訳ではなく、LUN のストレージ内のボックス設定に追加の変更が行なわれない場合にのみ、**sd*x*** は復元されます。

### 26.16.2. dm-multipath を実装している iSCSI の設定

**dm-multipath** を実装している場合、コマンドをマルチパスレイヤーにただちに委ねるよう iSCSI タイマーを設定することをお勧めします。次の行を **/etc/multipath.conf** ファイル内にある **device {** の下にネストさせます。

```
features  "1 queue_if_no_path"
```

これにより、**dm-multipath** 層ですべてのパスが失敗すると I/O エラーが再試行され、キューに入れられます。

SAN に問題がないことを監視するため iSCSI タイマーをさらに調整する必要性が生じる場合があります。設定できる iSCSI タイマーには *NOP-Out の間隔*、*NOP-Out のタイムアウト*および **replacement_timeout** などがあります。これらについては次のセクションで説明します。

#### 26.16.2.1. NOP-Out インターバル/タイムアウト

SAN に関する問題の監視を支援するために、iSCSI レイヤーは各ターゲットに NOP-Out 要求を送信します。NOP-Out 要求がタイムアウトになる場合には、iSCSI レイヤーは実行中のコマンドを停止して、SCS レイヤーに対し、可能な時点にそれらのコマンドを再度キューに入れるように指示することで対応します。

**dm-multipath** が使用されている場合、SCSI レイヤーが実行中のコマンドを停止して、それらのコマンドをマルチパスレイヤーに委ねます。次に、マルチパスレイヤーは別のパス上でこれらのコマンドを再試行します。**dm-multipath** が使用されて**いない** 場合には、これらのコマンドは 5 回まで再試行されてから完全に停止します。

NOP-Out 要求のインターバル (間隔) はデフォルトで 10秒です。これを調節するには、**/etc/iscsi/iscsid.conf** を開いて以下の行を編集します。

```
node.conn[0].timeo.noop_out_interval = [interval value]
```

インターバルが設定されると、iSCSI レイヤーはその *[interval value]* 秒ごとに各ターゲットに対して NOP-Out 要求を送信します。

デフォルトでは、NOP-Out 要求は 10秒 [9] でタイムアウトになります。これを調節するには、**/etc/iscsi/iscsid.conf** を開いて以下の行を編集します。

```
node.conn[0].timeo.noop_out_timeout = [timeout value]
```

これは、iSCSI レイヤーが *[timeout value]* 秒後に NOP-Out 要求をタイムアウトさせるように設定します。

#### SCSI エラーハンドラー

SCSI エラーハンドラーが実行中の場合は、あるパス上で実行中のコマンドは、そのパス上で NOP-Out 要求がタイムアウトになってもすぐには停止しません。その代わりに、これらのコマンドは **replacement_timeout** 秒の **経過後** に停止します。**replacement_timeout** についてさらに詳しくは、「**replacement_timeout**」 を参照してください。

SCSI エラーハンドラーが実行中であるかどうかを確認するには、以下を実行します。

```
# iscsiadm -m session -P 3
```

#### 26.16.2.2. replacement_timeout

**replacement_timeout** は、iSCSI レイヤーが実行中のコマンドを停止する前にタイムアウトになっているパス/セッションが再確立するまで待機する時間の長さを制御します。デフォルトの **replacement_timeout** 値は 120 秒です。

**replacement_timeout** を調整するには、**/etc/iscsi/iscsid.conf** を開いて以下の行を編集します。

```
node.session.timeo.replacement_timeout = [replacement_timeout]
```

**/etc/multipath.conf** 内の **1 queue_if_no_path** オプションは、コマンドをマルチパスレイヤーにすぐに委ねるように iSCSI タイマーを設定します (「**dm-multipath** を実装している iSCSI の設定」を参照)。この設定は、I/O エラーがアプリケーションに伝搬することを防ぐので、**replacement_timeout** は 15 秒から 20 秒に設定するのが適切でしょう。

**replacement_timeout** を低めに設定すると、(NOP-Out タイムアウト状態で) I/O は新しいパスにすぐに送信され、実行されます。その際に iSCSI レイヤーは停止したパス/セッションを再確立するよう試行します。すべてのパスがタイムアウトになるとマルチパスとデバイスマッパーレイヤーは、**/etc/iscsi/iscsid.conf** ではなく、**/etc/multipath.conf** の設定に基づいて内部的に I/O をキューに入れます。

**重要**

検討しているポイントがフェイルオーバーのスピードかセキュリティーかなどを問わず、**replacement_timeout** の推奨される値は、他の要素によって決まります。これらの要素には、ネットワーク、ターゲット、およびシステムのワークロードなどが含まれます。そのため、**replacements_timeout** に対する新たな設定を加える場合、これを徹底的にテストしてからミッションクリティカルなシステムに適用することが推奨されます。

### 26.16.3. iSCSI の root

iSCSI ディスクから直接 root パーティションにアクセスする場合は、iSCSI タイマーを設定し、iSCSI レイヤーからパスまたはセッションの再確立が複数回試行されるようにします。また、コマンドが SCSI レイヤーのキューにすぐに再度入れられないようにする必要があります。これは **dm-multipath** を実装する場合に行う作業と反対になります。

まず NOP-Out を無効にします。NOP-Out の間隔とタイムアウトをいずれもゼロにすると NOP-Out は無効になります。これを設定するには、**/etc/iscsi/iscsid.conf** ファイルを開いて次のように編集します。

```
node.conn[0].timeo.noop_out_interval = 0
node.conn[0].timeo.noop_out_timeout = 0
```

上記に合わせて、**replacement_timeout** には大きめの数値を設定します。これによりシステムのパスやセッションの再確立に要する待機時間が長くなります。**replacement_timeout** を調整するには、**/etc/iscsi/iscsid.conf** ファイルを開いて次のように編集します。

```
node.session.timeo.replacement_timeout = replacement_timeout
```

**/etc/iscsi/iscsid.conf** の設定が終了したら、影響を受けるストレージの再検出を行う必要があります。再検出によってシステムは **/etc/iscsi/iscsid.conf** で設定した新しい値を読み込み、その値を使用できるようになります。iSCSI デバイスの検出方法については、「iSCSI 相互接続をスキャンする」を参照してください。

### 特定セッションのタイムアウトの設定

特定セッションに対してタイムアウトを設定し、それを非永続的にすることができます (**/etc/iscsi/iscsid.conf** ファイルを使用しない方法)。これを実行するには、次のコマンドを実行します (状況に応じて変数を変更してください)。

```
# iscsiadm -m node -T target_name -p target_IP:port -o update -n
node.session.timeo.replacement_timeout -v $timeout_value
```

**重要**

root パーティションへのアクセスを必要とする iSCSI セッションには、本セクションに記載されている設定を行うことを推奨します。他のタイプのストレージへのアクセスを必要とする iSCSI セッション (つまり **dm-multipath** を使用するシステムの場合) については、「**dm-multipath** を実装している iSCSI の設定」を参照してください。

## 26.17. SCSI コマンドタイマーとデバイス状態の制御

Linux SCSI レイヤーは、各コマンドでタイマーをセットします。このタイマーが時間切れになると、SCSI レイヤーは *host bus adapter* (HBA) を休止して、残りのコマンドすべてがタイムアウトになるか、完了するのを待ちます。その後に、SCSI レイヤーはドライバーのエラーハンドラーを始動します。

エラーハンドラーがトリガーされると、以下の操作を順番に試行します (そのいずれかが正しく実行されるまで試行が続きます)。

1. コマンドの中止
2. デバイスのリセット
3. バスのリセット
4. ホストのリセット

これらの操作がすべて失敗する場合、デバイスは **offline** 状態にセットされます。この状況が発生すると、そのデバイスへのすべての I/O は、問題が修正されてユーザーがデバイスを **running** に設定しない限り、失敗します。

デバイスがファイバーチャネルプロトコルを使用し、**rport** がブロックされている場合にはプロセスは異なります。この場合、ドライバーはエラーハンドラーを始動する前に **rport** が再度オンラインになるのを数秒間待ちます。これにより、一時的なトランスポートの問題でデバイスがオフラインになることを防ぎます。

### デバイスの状態

デバイスの状態を表示するには、以下を使用します。

```
$ cat /sys/block/device-name/device/state
```

デバイスを **running** 状態に設定するには、以下を使用します。

```
$ echo running > /sys/block/device-name/device/state
```

### コマンドタイマー

コマンドタイマーを制御するには、**/sys/block/*device-name*/device/timeout** に書き込みをします。これを行うには以下を実行します。

**echo *value* /sys/block/*device-name*/device/timeout**

ここで、***value*** は実装する予定のタイムアウト値 (秒単位) です。

## 26.18. トラブルシューティング

このセクションでは、ユーザーがオンラインストレージの再設定で直面する一般的な問題へのソリューションを提供します。

**論理ユニットの削除状況がホスト上に反映されません。**

設定済みのファイラーで論理ユニットが削除されても、その変更はホスト上には反映されません。そのような場合、論理ユニットは*陳腐化*しているため、**dm-multipath** が使用されると、**lvm** コマンドは無限にハングします。

これを回避するには、以下の手順を実行します。

**手順26.10 陳腐化した論理ユニットの回避策**

1. **/etc/lvm/cache/.cache** 内のどの **mpath** リンクエントリーが陳腐化した論理ユニットに固有のものであるかを判別します。これを実行するには、以下のコマンドを実行します。

   ```
   $ ls -l /dev/mpath | grep stale-logical-unit
   ```

   **例26.19 特定の mpath リンクエントリーを判別**

   たとえば、***stale-logical-unit*** が 3600d0230003414f30000203a7bc41a00 である場合、以下の結果が出る可能性があります。

   ```
   lrwxrwxrwx 1 root root 7 Aug  2 10:33
   /3600d0230003414f30000203a7bc41a00 -> ../dm-4
   lrwxrwxrwx 1 root root 7 Aug  2 10:33
   /3600d0230003414f30000203a7bc41a00p1 -> ../dm-5
   ```

   これは、3600d0230003414f30000203a7bc41a00 が **dm-4** と **dm-5** の 2 つの **mpath** リンクにマップされていることを示しています。

2. 次に、**/etc/lvm/cache/.cache** を開いて、***stale-logical-unit*** および ***stale-logical-unit*** がマップされる **mpath** リンクを含むすべての行を削除します。

   **例26.20 関連した行の削除**

直前のステップと同じ例を使用すると、削除する必要のある行は以下になります。

```
/dev/dm-4
/dev/dm-5
/dev/mapper/3600d0230003414f30000203a7bc41a00
/dev/mapper/3600d0230003414f30000203a7bc41a00p1
/dev/mpath/3600d0230003414f30000203a7bc41a00
/dev/mpath/3600d0230003414f30000203a7bc41a00p1
```

[5] ***target_IP*** と ***port*** の変数は、ターゲット/ポータルの IP アドレスとポートの組み合わせを参照します。詳細については、「iSCSI API」 および 「iSCSI 相互接続をスキャンする」を参照してください。

[6] Refer to 「iSCSI 相互接続をスキャンする」 for information on ***proper_target_name***.

[7] セッションの SID 値を取得する方法についてさらに詳しくは、「iSCSI API」 を参照してください。

[8] This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (\) — should be treated as one command, sans backslashes.

[9] Red Hat Enterprise Linux 5.4 以前は、デフォルトの NOP-Out 要求のタイムアウトは 15 秒でした。

# 第27章 Device Mapper マルチパス機能と仮想ストレージ

Red Hat Enterprise Linux 7 では *DM-Multipath* と *仮想ストレージ* に対応しています。いずれの機能についても Red Hat 提供の 『DM Multipath』 および 『仮想管理ガイド』 で詳しく説明されています。

## 27.1. 仮想ストレージ

Red Hat Enterprise Linux 7 では、仮想ストレージ向けに以下のようなファイルシステムとオンラインストレージのメソッドに対応しています。

- ファイバーチャネル
- iSCSI
- NFS
- GFS2

Red Hat Enterprise Linux 7 の仮想化では、仮想インスタンスの管理に **`libvirt`** を使用しています。**`libvirt`** ユーティリティーでは仮想化したゲスト用のストレージ管理に*ストレージプール*という概念を採用しています。ストレージプールとは複数の小さなボリュームに区分けしたり、ゲストに直接割り当てたりすることのできるストレージを指します。ストレージプールのボリュームは仮想化したゲストに割り当てることができます。ストレージプールには利用可能な 2 種類のカテゴリーがあります。

**ローカルストレージのプール**

ローカルストレージとは、直接ホストに接続されるストレージデバイス、ファイル、ディレクトリーなどが該当します。ローカルストレージにはディスクに直接接続されたローカルのディレクトリーや LVM ボリュームグループが含まれます。

**ネットワーク (共有) ストレージプール**

ネットワークストレージとは、標準プロトコルを使ってネットワーク経由で共有されるストレージデバイスが該当します。ファイバーチャネル、iSCSI、NFS、GFS2、SCSI RDMA などのプロトコルを使った共有ストレージデバイスが含まれ、ホスト間での仮想化ゲストの移動には必須となります。

**重要**

ご使用の環境に仮想ストレージのインスタンスを導入し、設定する方法の詳細については、Red Hat 提供の 『Virtualization Deployment and Administration Guide』 を参照してください。

## 27.2. DM-Multipath

Device Mapper マルチパス機能 (DM-Multipath) は、サーバーノード群とストレージアレイ群の間の複数の入出力パスを 1 つのデバイスに設定することができる機能になります。こうした入出力パスは物理的な SAN 接続であり、別々のケーブル、スイッチ、コントローラーなどが含まれる可能性があります。マルチパス機能により入出力パスを集約し 集約されたパスで構成される新しいデバイスを作成できます。

DM-Multipath は主に次のような理由で使用されます。

**冗長性**

DM-Multipath では、アクティブ/パッシブな設定でフェールオーバーを行うことができます。アクティブ/パッシブな設定では、入出力に対して常にパスの半分しか使用されません。入出力パス

の要素 (ケーブル、スイッチ、コントローラーなど) のいずれかに障害が発生すると、DM-Multipath によって代替パスに切り替えられます。

**パフォーマンスの向上**

DM-Multipath は、アクティブ/アクティブモードでの設定が可能です。このモードでは、入出力はラウンドロビン式で複数のパス群に分散されます。一部の設定では、DM-Multipath は入出力パス上の負荷を検出することができるため、負荷のバランスを動的に再調整できます。

**重要**

ご使用の環境に DM-Multipath を導入し、設定する方法の詳細については Red Hat 提供の 『Using DM-Multipath』 ガイドを参照してください。

# 第28章 外部アレイ管理 (libStorageMgmt)

Red Hat Enterprise Linux 7 には、**libStorageMgmt** という新たな外部アレイ管理パッケージが標準装備されています。

## 28.1. libStorageMgmt とは

**libStorageMgmt** パッケージは、ストレージアレイから独立したアプリケーションプログラミングインターフェース (API) です。この API は安定して一貫性があるため、開発者はプログラミングを用いて異なるストレージアレイを管理し、指定のハードウェアで加速化された機能を利用できます。

このライブラリーは、より高いレベルの管理ツール向けのビルディングブロックとして使用されます。さらにエンドシステムの管理者は、ストレージを手動で管理し、スクリプトの使用によりストレージ管理タスクを自動化するためのツールとしてこのライブラリーを使用することもできます。

**libStorageMgmt** パッケージにより、以下のような操作が可能になります。

- ストレージプール、ボリューム、アクセスグループ、またはファイルシステムの一覧表示。
- ボリューム、アクセスグループ、ファイルシステム、または NFS エクスポートの作成および削除。
- ボリューム、アクセスグループ、またはイニシエーターへのアクセスの付与および削除。
- スナップショット、クローン、およびコピーによるボリュームの複製。
- アクセスグループの作成および削除、およびグループのメンバーの編集。

すべての操作がアレイ上で行なわれるため、CPU および相互接続帯域幅などのサーバーリソースは利用されません。

パッケージに含まれているもの:

- クライアントアプリケーションおよびプラグイン開発者向けの安定した C および Python API。
- ライブラリーを使用するコマンドラインインターフェース (**lsmcli**)。
- プラグインを実行するデーモン (**lsmd**)。
- クライアントアプリケーションのテストを許可するシミュレーターのプラグイン (**sim**)。
- アレイとのインターフェース用のプラグインのアーキテクチャー。

**警告**

このライブラリーとその関連ツールには、それが管理するアレイ上にあるすべてのデータを破棄する機能があります。本番稼働システムを使用して作業する前に、論理エラーを削除するため、ストレージシミュレーターのプラグインに対してアプリケーションとスクリプトの開発とテストを行うことを推奨します。また、本番への導入前に実際の非本番稼働ハードウェアでアプリケーションとスクリプトのテストを行うことも、可能な場合には強く推奨されます。

**libStorageMgmt** ライブラリーは、ストレージアレイの相違点に対応するためにプログインのアーキテクチャーを使用します。**libStorageMgmt** プラグインとそれらの作成方法についてさらに詳しくは、Red Hat の『開発者ガイド』 を参照してください。

## 28.2. libStorageMgmt の用語

### 28.2.1. libStorageMgmt の用語

複数のアレイベンダーおよびストレージ標準により、同様の機能に言及する場合でも異なる用語が使用されています。このライブラリーは、以下の用語を使用しています。

**ストレージアレイ**

Network Attached Storage (NAS) を使用してブロックアクセス (FC、FCoE、iSCSI) またはファイルアクセスを提供するストレージシステム。

**ボリューム**

Storage Area Network (SAN) ストレージアレイは、FC、iSCSI、または FCoE などの異なるトランスポート上でホストバスアダプター (HBA) にボリュームを公開できます。ホスト OS は、これをブロックデバイスとして処理します。1 つのボリュームは、**multipath[2]** が有効にされている場合、多くのディスクに公開されます)。

また、これは論理ユニット番号 (LUN)、SNIA 用語では StorageVolume、または仮想ディスクとしても知られています。

**プール**

ストレージスペースのグループです。ファイルシステムまたはボリュームはプールから作成できます。プールは、ディスク、ボリューム、および他のツールから作成できます。プールは、RAID 設定またはプロビジョニング設定を保持することもできます。

これは、SNIA 用語では StoragePool としても知られています。

**スナップショット**

任意の時点の、読み取り専用のスペース効率のよいデータのコピーです。

これは、読み取り専用のスナップショットとしても知られています。

**クローン**

任意の時点の、読み取りおよび書き込み可能な、スペース効率のよいデータのコピーです。

これは、読み取りおよび書き込み可能なスナップショットとしても知られています。

**コピー**

データの完全なビット単位のコピーです。フルスペースを占有します。

**鏡面印刷**

継続的に更新されるコピー (同期および非同期) です。

**アクセスグループ**

1 つまたは複数のストレージボリュームへのアクセスが付与される iSCSI、FC、および FCoE イニシエーターの集合です。これにより、ストレージボリュームのみが指定のイニシエーターによってアクセスされるようになります。

これは、イニシエーターグループとしても知られています。

**アクセス許可**

ボリュームを、指定のアクセスグループまたはイニシエーターに公開します。現在、**libStorageMgmt** ライブラリーは、特定の論理ユニット番号を選択する機能のある LUN マッピングをサポートしていません。**libStorageMgmt** ライブラリーでは、ストレージアレイは、次に利用可能な LUN を割り当て用に選択することができます。SAN からブートを設定する

か、または 256 を超えるボリュームをマスクするには、OS、ストレージ、または HBA 資料を必ず参照してください。

アクセス許可は LUN マスクとしても知られています。

**システム**

ストレージアレイまたは直接接続のストレージ RAID を表します。

**ファイルシステム**

Network Attached Storage (NAS) ストレージアレイは、NFS または CIFS プロトコルのいずれかを使用して、IP ネットワーク経由で OS をホストするためにファイルシステムを公開できます。ホスト OS は、クライアントのオペレーティングシステムによって、これをマウントポイントか、またはファイルが含まれるフォルダーとして処理します。

**ディスク**

データを保持する物理ディスクです。これは、通常 RAID 設定を使ってプールを作成する場合に使用されます。

これは、SNIA 用語を使用すると DiskDrive としても知られています。

**イニシエーター**

ファイバーチャネル (FC) またはイーサネット経由のファイバーチャネル (FCoE) では、イニシエーターはワールドワイドポートネーム (WWPN) またはワールドワイドノード名 (WWNN) です。iSCSI では、イニシエーターは iSCSI 修飾名 (IQN) です。NFS または CIFS では、イニシエーターはホスト名またはホストの IP アドレスです。

**子の依存関係**

一部のアレイには、元になるもの (親ボリュームまたはファイルシステム) と子 (スナップショットまたはクローンなど) 間の暗黙的な関係があります。たとえば、親に 1 つまたは複数の依存する子がある場合、親を削除することはできません。API は、このような関係が存在するかどうかを判別するためのメソッドと、必要なブロックを複製することで依存関係を削除するメソッドを提供します。

## 28.3. インストール

コマンドライン、必要なランタイムライブラリーおよびシミュレータープラグインを使用するために **libStorageMgmt** をインストールするには、以下のコマンドを使用します。

```
$ sudo yum install libstoragemgmt libstoragemgmt-python
```

ライブラリーを使用する C アプリケーションを開発するには、以下のコマンドを使って **libstoragemgmt-devel** や、オプションで **libstorage-debuginfo** パッケージをインストールします。

```
$ sudo yum install libstoragemgmt-devel libstoragemgmt-debuginfo
```

ハードウェアアレイと共に使用するために **libStorageMgmt** をインストールするには、以下のコマンドを使って、1 つ以上の適切なプラグインパッケージを選択します。

```
$ sudo yum install libstoragemgmt-name-plugin
```

利用可能なプラグインには、以下が含まれます。

**libstoragemgmt-smis-plugin**

汎用 SMI-S アレイサポート。

**libstoragemgmt-netapp-plugin**

NetApp ファイルの特殊サポート。

**libstoragemgmt-nstor-plugin**

NexentaStor の特殊サポート。

**libstoragemgmt-targetd-plugin**

targetd の特殊サポート。

次に、デーモンが起動時に実行されるようにインストールされ、設定されますが、次の再起動まではこれか行なわれません。デーモンを再起動せずにすぐに使用するには、デーモンを手動で開始します。

アレイを管理するには、プラグインによるサポートが必要です。ベースインストールパッケージには、数多くの異なるベンダー用のオープンソースのプラグインが含まれます。追加のプラグインパッケージは、アレイのサポートが改善されると別途利用可能になります。現在、サポートされているハードウェアは継続的に変化し、改善されています。プロジェクトの web ページには、サポートされているアレイと各アレイの特定の機能についての最新情報が記載されており、このページは http://libstoragemgmt.sourceforge.net/supported_hardware.html からアクセスできます。

libStorageMgmt デーモン (**lsmd**) は、システムの任意の標準サービスのように動作します。

libStorageMgmt の状態をチェックするには、以下を使用します。

```
$ sudo systemctl status libstoragemgmt.service
```

サービスを停止するには、以下を使用します。

```
$ sudo systemctl stop libstoragemgmt.service
```

サービスを開始するには、以下を使用します。

```
$ sudo systemctl start libstoragemgmt.service
```

## 28.4. libStorageMgmt の使用

**libStorageMgmt** を対話形式で使用するには、**lsmcli** コマンドを使用します。

**lsmcli** ツールでは、2 つのことが実行される必要があります。

- アレイおよびアレイが必要とする設定可能なオプションに接続するためのプラグインを特定するために使用される URI (Uniform Resource Identifier)。
- アレイについての有効なユーザー名およびパスワード。

URI は以下の形式になります。

plugin+*optional-transport*://*username*@*host*:*port*/?*query-string-parameters*

各プラグインには、必要なことについての異なる要件が含まれます。

**例28.1 異なるプラグイン要件の例**

**ユーザー名またはパスワードを必要とするシミュレーターのプラグイン**

```
sim://
```

**ユーザー名の root を使用する SSL 経由の NetApp プラグイン**

```
ontap+ssl://root@filer.company.com/
```

**EMC アレイ用の SSL 経由の SMI-S プラグイン**

```
smis+ssl://admin@provider.com:5989/?namespace=root/emc
```

追加の例については、https://sourceforge.net/p/libstoragemgmt/wiki/URISyntax/を参照してください。

URI を使用するには 3 つのオプションがあります。

1. コマンドの一部として URI を渡します。

```
$ lsmcli -u sim://...
```

2. URI を環境変数に保存します。

```
$ export LSMCLI_URI=sim:// && lsmcli ...
```

3. ファイル **~/.lsmcli** に URI を置きます。これには、"=" で区切られた名前と値のペアが含まれます。現在サポートされる唯一の設定は 'uri' です。

どの URI を使用するかを決定するには、以下の順序で行う必要があります。以下の 3 つがすべて指定される場合、コマンドライン上の最初のものだけが使用されます。

コマンドラインに -P を指定するか、または環境変数の LSMCLI_PASSWORD にこれを置くことによってパスワードを指定します。

**例28.2 lsmcli の例**

新規ボリュームを作成するためにコマンドラインを使用し、これをイニシエーターに表示する方法の例。

この接続によって処理されるアレイをリストします。

```
$ lsmcli list --type SYSTEMS
ID     | Name                              | Status
-------+-----------------------------------+--------
sim-01 | LSM simulated storage plug-in | OK
```

ストレージプールをリストします。

```
$ lsmcli list --type POOLS -H
ID   | Name          | Total space          | Free space           |
System ID
-----+---------------+----------------------+----------------------+--
---------
POO2 | Pool 2        | 18446744073709551616 | 18446744073709551616 |
```

```
sim-01
POO3 | Pool 3        | 18446744073709551616 | 18446744073709551616 |
sim-01
POO1 | Pool 1        | 18446744073709551616 | 18446744073709551616 |
sim-01
POO4 | lsm_test_aggr | 18446744073709551616 | 18446744073709551616 |
sim-01
```

ボリュームを作成します。

```
$ lsmcli volume-create --name volume_name --size 20G --pool POO1 -H
ID   | Name        | vpd83                            | bs  | #blocks
| status | ...
-----+-------------+----------------------------------+-----+---------
-+--------+----
Vol1 | volume_name | F7DDF7CA945C66238F593BC38137BD2F | 512 | 41943040
| OK     | ...
```

iSCSI イニシエーターを含むアクセスグループを作成します。

```
$ lsmcli --create-access-group example_ag --id iqn.1994-
05.com.domain:01.89bd01 --type ISCSI --system sim-01
ID                                | Name       | Initiator ID
|SystemID
----------------------------------+------------+----------------------
-----------+--------
782d00c8ac63819d6cca7069282e03a0 | example_ag | iqn.1994-
05.com.domain:01.89bd01 |sim-01
```

iSCSI イニシエーターを含むアクセスグループを作成します。

```
$ lsmcli access-group-create --name example_ag --init iqn.1994-
05.com.domain:01.89bd01 --init-type ISCSI --sys sim-01
ID                                | Name       | Initiator IDs
| System ID
----------------------------------+------------+----------------------
-----------+-----------
782d00c8ac63819d6cca7069282e03a0 | example_ag | iqn.1994-
05.com.domain:01.89bd01 | sim-01
```

新たに作成されたボリュームに対するアクセスグループの表示を許可します。

```
$ lsmcli access-group-grant --ag 782d00c8ac63819d6cca7069282e03a0 --vol
Vol1 --access RW
```

ライブラリーの設計上、*inter-process communication* (IPC) によりクライアントとプラグイン間のプロセスが分離します。これにより、プラグインのバグが原因で、クライアントアプリケーションがクラッシュすることを防ぎます。さらに、プラグインの作成者が独自に選択するライセンスを使ってプラグインを作成する手段も提供します。クライアントが URI を渡すライブラリーを開く際に、クライアントライブラリーは UR を参照し、どちらのプラグインを使用する必要があるのかを決定します。

プラグインは、実際はスタンドアロンのアプリケーションですが、コマンドライン上でそれらに渡すファールディスクリプターを持つように設計されています。クライアントライブラリーは、デーモンにプラグインをフォークし、実行させる適切な Unix ドメインソケットを開きます。これにより、クライアントライブラ

リーには、プラグインによるポイントツーポイントの通信チャンネルが提供されます。デーモンは、既存クライアントに影響を与えることなく再起動できます。クライアントがライブラリーをそのプラグインに対してオープンにしている間に、プラグインプロセスは実行されます。1 つ以上のコマンドが送信されると、プラグインは閉じられ、プラグインプロセスはクリーンアップしてから終了します。

このプロセスのシーケンスダイアグラムについては、https://sourceforge.net/p/libstoragemgmt/wiki/Architecture/を参照してください。

lsmcli のデフォルトの動作は、操作が完了するまで待機します。要求される操作によって、多くの時間がかかる場合があります。通常の使用に戻すために、コマンドライン上で **-b** オプションを使用することができます。出口コードが 0 の場合、コマンドが終了します。出口コードが 7 の場合、コマンドは進行中となり、ジョブ ID は標準出力に書き込まれます。ユーザーまたはスクリプトは、その後にジョブ ID を取り、**lsmcli --jobstatus *JobID*** を使用することにより、必要に応じてコマンドのステータスを照会します。ジョブが完了すると、出口の値は 0 となり、結果は標準出力に表示されます。コマンドが依然として進行中の場合、戻り値は 7 となり、完成度 (パーセント) が標準出力に表示されます。

**例28.3 非同期の例**

コマンドがすぐに返されるように、**-b** オプションを渡すボリュームを作成します。

```
$ lsmcli volume-create --name async_created --size 20G --pool POO1 -b
JOB_3
```

出口の値を確認するためにチェックします。7 はジョブが依然として進行中であることを示すことに注意してください。

```
$ echo $?
7
```

ジョブが完了しているかどうかをチェックします。

```
$ lsmcli job-status --job JOB_3
33
```

出口の値を確認するためにチェックします。7 はジョブが依然として進行中であることを示すため、標準出力には完成度 (パーセンテージ)、上記の画面では 33% と表示されます。

```
$ echo $?
7
```

少し待機し、再びチェックします。出口 0 は成功を示し、標準出力は新規のボリュームを表示することに注意してください。

```
$ lsmcli job-status --job JOB_3
ID   | Name          | vpd83                            | Block Size
| ...
-----+---------------+----------------------------------+-------------
+-----
Vol2 | async_created | 855C9BA51991B0CC122A3791996F6B15 | 512         |
...
```

スクリプトの場合、**-t *SeparatorCharacters*** オプションを渡します。これにより、出力の解析がより簡単になります。

**例28.4 スクリプトの例**

```
$ lsmcli list --type volumes -t#
Vol1#volume_name#049167B5D09EC0A173E92A63F6C3EA2A#512#41943040#21474836
480#OK#sim-01#POO1
Vol2#async_created#3E771A2E807F68A32FA5E15C235B60CC#512#41943040#214748
36480#OK#sim-01#POO1
```

```
$ lsmcli list --type volumes -t " | "
Vol1 | volume_name | 049167B5D09EC0A173E92A63F6C3EA2A | 512 | 41943040
| 21474836480 | OK | 21474836480 | sim-01 | POO1
Vol2 | async_created | 3E771A2E807F68A32FA5E15C235B60CC | 512 |
41943040 | 21474836480 | OK | sim-01 | POO1
```

```
$ lsmcli list --type volumes -s
-------------------------------------------
ID          | Vol1
Name        | volume_name
VPD83       | 049167B5D09EC0A173E92A63F6C3EA2A
Block Size  | 512
#blocks     | 41943040
Size        | 21474836480
Status      | OK
System ID   | sim-01
Pool ID     | POO1
-------------------------------------------
ID          | Vol2
Name        | async_created
VPD83       | 3E771A2E807F68A32FA5E15C235B60CC
Block Size  | 512
#blocks     | 41943040
Size        | 21474836480
Status      | OK
System ID   | sim-01
Pool ID     | POO1
-------------------------------------------
```

一般的なスクリプトには Python ライブラリーを使用することを推奨します。

**lsmcli** についてさらに詳しくは、man ページまたはコマンド**lsmcli --help** を参照してください。

## 28.5. libStorageMgmt の資料

さらに詳しくは、次の Web サイトを参照してください。

- 公式 Web サイト: https://sourceforge.net/projects/libstoragemgmt/
- プロジェクト Wiki: https://sourceforge.net/p/libstoragemgmt/wiki/Home/

# 付録A 改訂履歴

**改訂 3-4.1** **Mon Nov 23 2015** **Aiko Sasaki**
翻訳ファイルを XML ソースバージョン 2-53 と同期

**改訂 2-53** **Tue May 13 2014** **Jacquelynn East**
スタイル変更のための再ビルド

**改訂 2-52** **Tue Apr 8 2014** **Jacquelynn East**
ベータを更新

**改訂 2-51** **Tue Apr 8 2014** **Jacquelynn East**
BZ#1025572: ベータ版から Btrfs 警告を更新

**改訂 2-50** **Mon Apr 7 2014** **Jacquelynn East**
BZ#952417: md RAID が LVM ターゲットとして利用可能であることを追記
BZ#959504: Snapper に関する新しい章を追加
BZ#978110、BZ#1078802: NFSoRDMA サーバーに関する指示を削除、およびクライアントに関する指示を更新
BZ#795284: 新規の統一された extN ドライバーについての段落を追加
BZ#1071147: quota コマンドを修正
BZ#1084303: ファイルシステム構造についての若干の編集
BZ#1050839: bigalloc がサポートされていないことを明記
BZ#1070567: 仮想化ガイドへの誤った参照を修正
BZ#1032615: GCC および DTS についてのセクションを更新
BZ#1009827: GCC パワーデフォルトを追加

**改訂 2-48** **Thu Apr 2 2014** **Jacquelynn East**
BZ#795222: libStorageMgmt のプラグインの作成に関する開発者ガイドの参照を追加

**改訂 2-47** **Tue Apr 1 2014** **Jacquelynn East**
BZ#998954: NFSv2 への参照を削除
BZ#1064035: libStorageMgmt の例を更新

**改訂 2-45** **Fri Mar 28 2014** **Jacquelynn East**
BZ#795249、BZ#795250: pNFS の章を更新

**改訂 2-43** **Thu Mar 27 2014** **Jacquelynn East**
BZ#1056152: eCrypt は Red Hat Enerprise Linux 7 ではサポートされないため eCrypt の章を削除
BZ#1044055: Red Hat Enterprise Linux 7 の新機能を含む概要の章を更新。Red Hat Enterprise Linux 6 の最新ではない参考資料を削除。

**改訂 2-42** **Thu Mar 27 2014** **Jacquelynn East**
BZ#1044059: デフォルトのファイルシステムの参照を ext4 から xfs に変更。

**改訂 2-41** **Tue Mar 25 2014** **Jacquelynn East**
BZ#952423: SSM の章へのいくつかの変更。

**改訂 2-39** **Wed Mar 5 2014** **Jacquelynn East**
BZ#1068993: xfs_metadump 記述への若干の編集。

**改訂 2-34** **Tue Mar 4 2014** **Jacquelynn East**

BZ#952423: SSM の章を追加。

**改訂 2-33** **Fri Feb 7 2014** **Jacquelynn East**

BZ#1058529: xfs の章への若干の編集。

**改訂 2-32** **Wed Feb 5 2014** **Jacquelynn East**

BZ#1016367: ext3 セクションでのいくつかの編集。

**改訂 2-31** **Tue Jan 21 2014** **Jacquelynn East**

BZ#1055547: system-config-lvm の章の削除。

**改訂 2-30** **Mon Jan 20 2014** **Jacquelynn East**

BZ#795220: libStorageMgmt の章を追加。

**改訂 2-29** **Mon Jan 6 2014** **Jacquelynn East**

ファイルシステムのサポート制限へのリンクを追加 (Red Hat Enterprise Linux 6 バージョン)。

**改訂 2-27** **Fri Dec 13 2013** **Jacquelynn East**

BZ#1014902: いくつかの若干の編集
BZ#1006628: いくつかの若干の編集
BZ#795261: いくつかの若干の編集

**改訂 2-25** **Mon Dec 9 2013** **Jacquelynn East**

BZ#795292: I/O スケジューラーのパフォーマンスの増大についての情報を追加

**改訂 2-23** **Thu Dec 5 2013** **Jacquelynn East**

BZ#1006628: ext4 から xfs への移行に関するセクションを追加

**改訂 2-22** **Wed Dec 4 2013** **Jacquelynn East**

BZ#1036919: 若干の編集

**改訂 2-21** **Tue Dec 3 2013** **Jacquelynn East**

BZ#1028825: 若干の編集
BZ#1028820: 若干の編集
BZ#1028821: 若干の編集
BZ#1028823: 若干の編集

**改訂 2-20** **Wed Nov 27 2013** **Jacquelynn East**

ベータ前のビルド

**改訂 2-19** **Fri Nov 22 2013** **Jacquelynn East**

BZ#1007665: ベータ後に削除される Btrfs に関する重要なユースケースを追加

**改訂 2-18** **Tue Nov 19 2013** **Jacquelynn East**

BZ#1014902: xfs_quota に関する重要な注記を追加

**改訂 2-17** **Fri Nov 15 2013** **Jacquelynn East**

BZ#860097: systemctl stop fcoe.service に関する注記を追加

**改訂 2-16** **Thu Nov 14 2013** **Jacquelynn East**

スナップショットのビルド

**改訂 2-15** **Tue Nov 5 2013** **Jacquelynn East**
BZ#1015936: ファイルシステムについてのサポートの詳細を削除し、代わりに公式の記事にリンク

**改訂 2-14** **Mon Nov 4 2013** **Jacquelynn East**
BZ#1016367: Ext3 および Ext4 の章についての情報および修正を追加

**改訂 2-13** **Wed Oct 30 2013** **Jacquelynn East**
BZ#795261: Btrfs のマルチデバイスのセクションを追加
BZ#1007620: タイプミスを修正

**改訂 2-11** **Thu Oct 17 2013** **Jacquelynn East**
BZ#795268: Btrfs SDD の最適化セクションを追加

**改訂 2-10** **Mon Oct 14 2013** **Jacquelynn East**
BZ#836074: 以下の Btrfs セクションを追加。Btrfs とは、Btrfs ファイルシステムの作成、Btrfs ファイルシステムのマウント、Btrfs ファイルシステムのサイズ変更、Btrfs の参考情報

**改訂 2-7** **Wed Oct 9 2013** **Jacquelynn East**
BZ#1015936: xattr をデフォルトで有効にするという文章を追加、inode64 のマウントがデフォルトになったため、これに関する情報を削除、**`mkfs.xfs`** が存在しないことを説明、xfsrestore コマンドを修正。

**改訂 2-6** **Thu Sep 26 2013** **Jacquelynn East**
BZ#10007620: XFS および Btrfs を対象として扱うため、ディスククォータの章に対する注記を追加

**改訂 2-5** **Wed Sep 25 2013** **Jacquelynn East**
BZ#795282: ext4　についてのサポート情報を更新

**改訂 2-3** **Wed Sep 18 2013** **Jacquelynn East**
Red Hat Enterprise Linux 6 バージョンに一致する章を再編成。
BZ#911466: ENOSPC 書き込み障害エラーについての注記を追加。

**改訂 2-1** **Tue Jun 19 2013** **Jacquelynn East**
Red Hat Enterprise Linux 7 のブランチ作成

# 索引

## シンボル

## F

## G



## I

## U



## V



## W



## X