



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

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

Milan Navrátil
Red Hat Customer Content Services

Jacquelynn East
Red Hat Customer Content Services

Don Domingo
Red Hat Customer Content Services

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

法律上の通知

Copyright © 2017 Red Hat, Inc.

Server Development Kernel Storage

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Software Maintenance Engineering

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

msuchane@redhat.com

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

revers@redhat.com

オンラインストレージ

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

David Howells

Java ® is a registered trademark of Oracle and/or its affiliates.

Server Development Hardware Enablement

dhowells@redhat.com

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

FS Cache

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Base Operating System Installer

dlehman@redhat.com

Node.js ® is a trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

Jeff Moyer

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

Eric Sandeen

Server Development Kernel File System

esandeen@redhat.com

ext3、ext4、XFS、暗号化ファイルシステム

概要

Mike Snitzer

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ます。本ガイドは、Red Hat Enterprise Linux または Fedora について、基礎から中級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。

msnitzer@redhat.com

Red Hat Subject Matter Experts

Contributors

Contributors

Contributors

編集者

Marek Suchánek

Red Hat Customer Content Services

msuchane@redhat.com

目次

第1章 概要	6
1.1. RED HAT ENTERPRISE LINUX 7 の最新情報	6
パート I. ファイルシステム	8
第2章 ファイルシステムの構造およびメンテナンス	9
2.1. ファイルシステム階層標準 (FHS) の概要	9
2.2. 特殊な RED HAT ENTERPRISE LINUX ファイルの場所	17
2.3. /PROC 仮想ファイルシステム	17
2.4. 未使用ブロックの破棄	18
第3章 XFS ファイルシステム	19
3.1. XFS ファイルシステムの作成	20
3.2. XFS ファイルシステムのマウント	21
3.3. XFS クォータの管理	22
3.4. XFS ファイルシステムのサイズの拡大	24
3.5. XFS ファイルシステムの修復	24
3.6. XFS ファイルシステムの一時停止	25
3.7. XFS ファイルシステムのバックアップと復元	25
3.8. XFS ファイルシステムのその他のユーティリティー	28
3.9. EXT4 から XFS への移行	29
第4章 EXT3 ファイルシステム	32
4.1. EXT3 ファイルシステムの作成	33
4.2. EXT3 ファイルシステムへの変換	33
4.3. EXT2 ファイルシステムに戻る	34
第5章 EXT4 ファイルシステム	36
5.1. EXT4 ファイルシステムの作成	37
5.2. EXT4 ファイルシステムのマウント	38
5.3. EXT4 ファイルシステムのサイズ変更	39
5.4. EXT2、EXT3、EXT4 ファイルシステムのバックアップ	40
5.5. EXT2、EXT3、EXT4 ファイルシステムの復元	41
5.6. EXT4 ファイルシステムのその他のユーティリティー	43
第6章 BTRFS (テクノロジープレビュー)	45
6.1. BTRFS ファイルシステムの作成	45
6.2. BTRFS ファイルシステムのマウント	45
6.3. BTRFS ファイルシステムのサイズ変更	46
6.4. 複数デバイスの統合ボリューム管理	49
6.5. SSD の最適化	53
6.6. BTRFS の参考資料	53
第7章 GLOBAL FILE SYSTEM 2	55
第8章 NFS (NETWORK FILE SYSTEM)	56
8.1. NFS の仕組み	56
8.2. PNFS	59
8.3. NFS クライアントの設定	59
8.4. AUTOFS	61
8.5. 一般的な NFS マウントオプション	67
8.6. NFS の起動と停止	68
8.7. NFS サーバーの設定	70
8.8. NFS の保護	76

8.9. NFS および RPCBIND	79
8.10. 参照	80
第9章 FS-CACHE	82
9.1. 性能に関する保証	83
9.2. キャッシュの設定	83
9.3. NFS でのキャッシュの使用	84
9.4. キャッシュの間引き制限 (CACHE CULL) の設定	86
9.5. 統計情報	87
9.6. 参照	87
パート II. ストレージ管理	88
第10章 ストレージをインストールする際の注意点	89
10.1. 特に注意を要する事項について	89
第11章 ファイルシステムのチェック	92
11.1. FSCK のベストプラクティス	92
11.2. FSCK に関するファイルシステム固有の情報	93
第12章 パーティション	97
12.1. パーティションテーブルの表示	98
12.2. パーティションの作成	100
12.3. パーティションの削除	102
12.4. FDISK を使用したパーティションのサイズ変更	102
第13章 SNAPPER を用いたスナップショットの作成および維持	105
13.1. 初期の SNAPPER セットアップ	105
13.2. SNAPPER スナップショットの作成	106
13.3. SNAPPER スナップショット間の変更の追跡	109
13.4. スナップショット間の変更の取り消し	113
13.5. SNAPPER スナップショットの削除	114
第14章 SWAP領域	115
14.1. SWAP 領域の追加	116
14.2. SWAP 領域の削除	118
14.3. SWAP 領域の移動	119
第15章 SYSTEM STORAGE MANAGER (SSM)	120
15.1. SSM のバックエンド	120
15.2. 一般的な SSM タスク	122
15.3. SSM のリソース	129
第16章 ディスク割り当て	131
16.1. ディスククォータの設定	131
16.2. ディスククォータの管理	135
16.3. ディスククォータのリファレンス	138
第17章 RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS)	139
17.1. RAID のタイプ	139
17.2. RAID レベルとリニアサポート	140
17.3. LINUX RAID サブシステム	142
17.4. インストーラーでの RAID サポート	143
17.5. インストール後のルートディスクの RAID1 への変換	143
17.6. RAID セットの設定	143
17.7. 高度な RAID デバイスの作成	144

第18章 MOUNT コマンドの使い方	146
18.1. 現在マウントされているファイルシステムの一覧表示	146
18.2. ファイルシステムのマウント	147
18.3. ファイルシステムのアンマウント	157
18.4. MOUNT コマンドのリファレンス	157
第19章 VOLUME_KEY 機能	159
19.1. VOLUME_KEY コマンド	159
19.2. VOLUME_KEY の個人ユーザーとしての使用	160
19.3. 規模の大きな組織での VOLUME_KEY の使用	161
19.4. VOLUME_KEY のリファレンス	164
第20章 ソリッドステートディスクの導入ガイドライン	165
導入に関する考慮事項	165
パフォーマンスチューニングに関する注意点	167
第21章 書き込みバリア	168
21.1. 書き込みバリアの重要性	168
21.2. 書き込みバリアの有効化および無効化	169
21.3. 書き込みバリアに関する注意点	169
第22章 ストレージの I/O 調整とサイズ	171
22.1. ストレージアクセス用のパラメーター	171
22.2. ユーザー領域のアクセス	172
22.3. 標準	173
22.4. 入出力パラメーターのスタック	174
22.5. 論理ボリュームマネージャー	174
22.6. パーティションとファイルシステムのツール	175
第23章 リモートディスクレスシステムの設定	177
23.1. ディスクレスクライアントの TFTP サービスの設定	177
23.2. ディスクレスクライアントの DHCP の設定	178
23.3. ディスクレスクライアントのエクスポートしたファイルシステムの設定	179
第24章 オンラインストレージ管理	181
24.1. ターゲットの設定	181
24.2. iSCSI イニシエーターの作成	190
24.3. ファイバーチャネル	191
24.4. ファイバーチャネルオーバーイーサネットインターフェースの設定	193
24.5. 起動時に FCOE インターフェースを自動マウントする設定	194
24.6. iSCSI	196
24.7. 永続的な命名	197
24.8. ストレージデバイスの削除	201
24.9. ストレージデバイスへのパスの削除	202
24.10. ストレージデバイスまたはパスの追加	203
24.11. ストレージの相互接続のスキャン	205
24.12. iSCSI 検出の設定	206
24.13. 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	207
24.14. iSCSI 相互接続のスキャン	211
24.15. iSCSI ターゲットへのログイン	214
24.16. オンライン論理ユニットのサイズ変更	214
24.17. RESCAN-SCSI-BUS.SH による論理ユニットの追加と削除	218
24.18. リンク切れ動作の修正	219
24.19. SCSI コマンドタイマーとデバイス状態の制御	222
24.20. オンラインストレージ設定のトラブルシューティング	223

24.21. EH_DEADLINE を使用したエラーからの回復における最大時間の設定	224
第25章 DEVICE MAPPER マルチパス機能と仮想ストレージ	226
25.1. 仮想ストレージ	226
25.2. DM-MULTIPATH	226
第26章 外部アレイ管理 (LIBSTORAGEMGMT)	228
26.1. LIBSTORAGEMGMT とは	228
26.2. LIBSTORAGEMGMT の用語	229
26.3. インストール	231
26.4. LIBSTORAGEMGMT の使用	232
付録A ストレージ管理に関する RED HAT CUSTOMER PORTAL LABS	237
SCSI DECODER	237
FILE SYSTEM LAYOUT CALCULATOR	237
LVM RAID CALCULATOR	237
ISCSI HELPER	237
SAMBA CONFIGURATION HELPER	237
MULTIPATH HELPER	237
NFS HELPER	238
MULTIPATH CONFIGURATION VISUALIZER	238
RHEL BACKUP AND RESTORE ASSISTANT	238
付録B 改訂履歴	240
索引	241

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を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状態) でストレージソリューションを管理する場合に簡単に参照できるように構成されています。

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 つの分野で構成されています。

「ファイルシステム」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」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。詳細は、[15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 ファイルシステムの詳細は、[3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。詳細は、[13章 Snapper を用いたスナップショットの作成および維持](#) 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 の詳細は、[6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」セクションでは、ファイルシステムの構造および保守、Btrfs テクノリジープレビュー、および Red Hat が完全にサポートするファイルシステム (ext3、ext4、GFS2、XFS、NFS、および FS-Cache) に関する情報を提供します。

Red Hat Enterprise Linux のファイルシステムやストレージの制限に関する概要は、Red Hat ナレッジベース「[Red Hat Enterprise Linux テクノロジーの機能と制限](#)」を参照してください。

XFS は、Red Hat Enterprise Linux 7、および Red Hat におけるデフォルトのファイルシステムです。他のファイルシステムを使用する強い動機がある場合を除いて、XFS を使用することが推奨されます。一般的なファイルシステムとそのプロパティに関する一般的な情報は、Red Hat ナレッジベースの記事「[How to Choose your Red Hat Enterprise Linux File System](#)」を参照してください。

第2章 ファイルシステムの構造およびメンテナンス

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体系です。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編成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けます。

- 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できますが、**共有不可 (unsharable)** ファイルはローカルでしか利用できません。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、システム管理者が変更し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、もしくは各ユーザーがそのファイルに対してどのようなアクセスレベルを持つかが決定されます。この体系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。

2.1. 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。

2.1.1. 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例に過ぎません。

2.1.1.1. 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、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 **gdmoe-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。

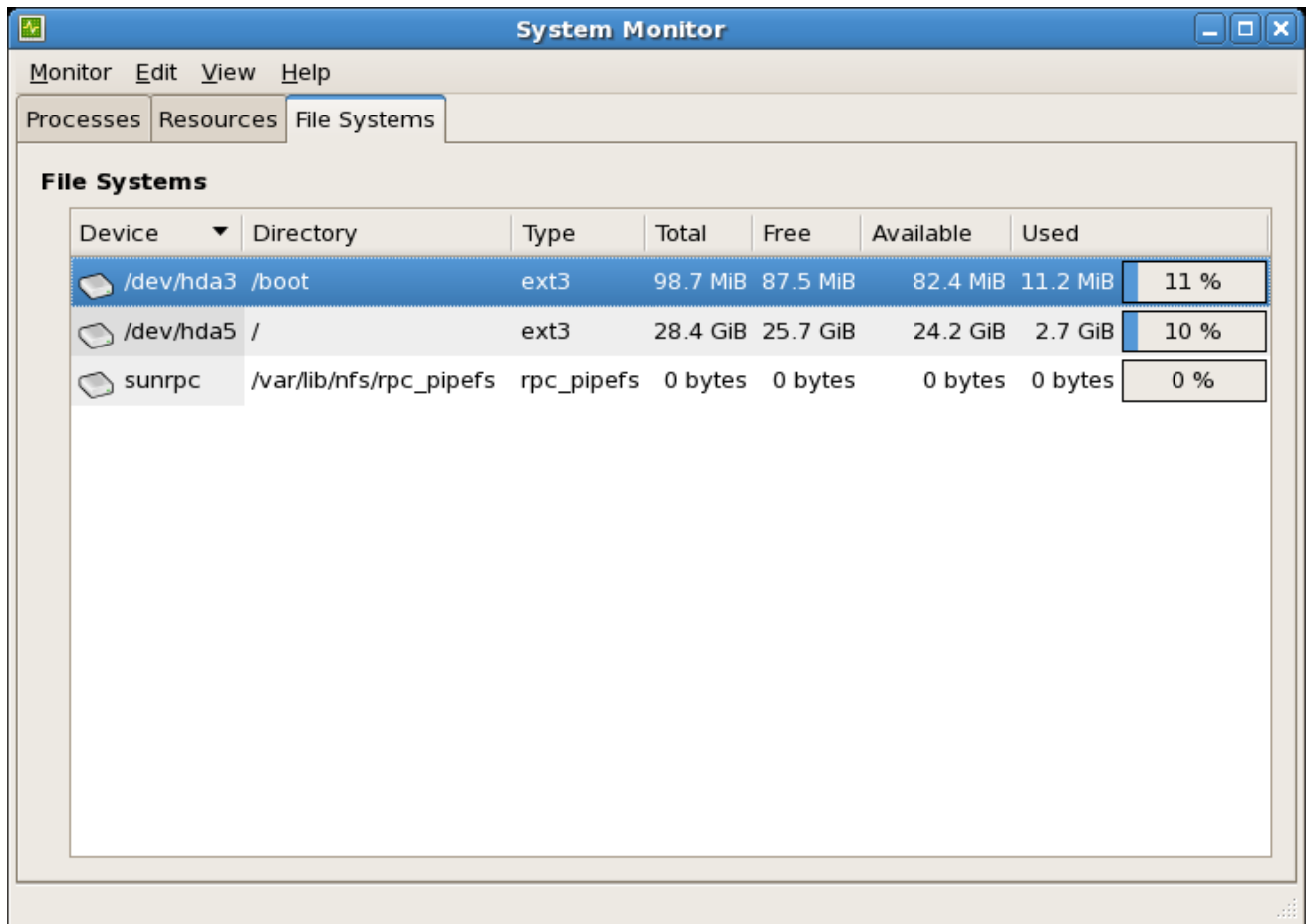


図2.1 GNOME システムモニターの「ファイルシステム」タブ

2.1.1.2. /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。

2.1.1.3. /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。**udev** 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時 (例: USB を使用) や挿入時 (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 チャンネル上のマスターデバイス
/dev/hdb	プライマリー IDE チャン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 チャンネル上の 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**。*sdb* はストレージデバイス名、*X* はパーティション番号になります。**/dev/sdbX** は、**/dev/disk/by-id/wwid** または **/dev/disk/by-uuid/UUID** になる場合があります (両方のオプションの詳細は「[永続的な命名](#)」を参照してください)。

2.1.1.4. /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。バイナリーをこのディレクトリーに格納してはならないため、バイナリーは **/usr/bin/** または **/usr/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。

2.1.1.5. /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。

2.1.1.6. `/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。

2.1.1.7. `/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。

2.1.1.8. `/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。

2.1.1.9. `/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。

2.1.1.10. `/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 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

Red Hat Enterprise Linux 7.0 では、**/sbin** は **/usr/sbin** に移動しました。このディレクトリーにはシステムの起動、復元、または修復に不可欠なバイナリーを含むシステム管理バイナリーがすべて含まれます。**/usr/sbin/** のバイナリーを使用するには root 権限が必要です。

/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、このサブディレクト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/** に保存し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。

2.1.1.11. **/var/** ディレクトリー

FHS では、Linux が **/usr/** を読み込み専用としてマウントすることを求めているため、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/`



重要

`/var/run/media/user` 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。以前は `/media/` ディレクトリーが使用されたことに注意してください。

`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/`

2.2. 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 の詳細は、`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** 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。

2.3. /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関する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から可能)。

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ある場合は、ここに含まれます。

/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 ファイルシステムの詳細は、Red Hat Enterprise Linux 7 『導入ガイド』を参照してください。

2.4. 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) を使って指定し、ユーザーの介入なしでリアルタイムで実行します。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を除き、バッチ破棄オペレーションを推奨します。

第3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にします。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 だけでなく XFS ファイルシステムにも影響を与えます。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関する詳細は、[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実行すると、ENOSPC で予期しない書き込みエラーが出て失敗する場合があります。これは XFS の設計がパフォーマンス志向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、各ファイルの、名前と値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なります。

クォータのジャーナリング

クラッシュ後に行なわれる時間が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。

デフォルトの `atime` の動作は `relatime`

XFS のデフォルトでは **Relatime** が有効になっています。**noatime** と比較すると **Relatime** のオーバーヘッドはほとんどなく、同じ **atime** の値を維持します。

3.1. 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3.1 **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 にも、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は、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数または 1 ストライプ内のストライプユニット数を指定します。

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
```

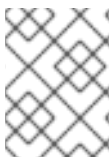
XFS ファイルシステムの作成方法は、**man mkfs.xfs** および『Red Hat Enterprise Linux パフォーマンスチューニングガイド』の『Basic Tuning for XFS』の章を参照してください。

3.2. 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にするため、XFS ではデフォルトで書き込みバリアを使用します。したがって、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
```

書き込みバリアの詳細は、[21章書き込みバリア](#)を参照してください。

Direct Access のテクノロジープレビュー

Red Hat Enterprise Linux 7.3 以降、**Direct Access** (DAX) では、アプリケーションがアドレス空間に永続メモリーを直接マッピングする手段を ext4 および XFS ファイルシステムでのテクノロジープレビューとして提供するようになりました。DAX を使用するには、永続メモリーがシステムに設定されている必要があります。永続メモリーは通常、1 つまたは複数の Non-Volatile Dual In-line Memory Module (NVDIMM) などの形式で提供され、DAX をサポートするファイルシステムは NVDIMM 上に作成する必要があります。また、ファイルシステムは **dax** のマウントオプションでマウントする必要があります。DAX でマウントしたファイルシステムにファイルを **mmap** すると、アプリケーションのアドレス空間にストレージが直接マッピングされます。

3.3. 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に排他的となります。

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一覧については、サブコマンドの **help** をご覧ください。

すべてのサブコマンドは、コマンドラインからも **-c** オプションを使って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3.2 サンプルのクォータレポートの表示

たとえば、`/home (/dev/blockdevice)` 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、プ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3.3 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。プロジェクト名を `/etc/projectid` に追加して、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ことができます。プロジェクトを `/etc/projects` に追加したら、次のコマンドを使用してそのプロジェクトディレクトリーを初期化します。

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

次のコマンドを実行して、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を設定します。

```
# 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。

3.4. 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 オプションを使用して、ファイルシステムをその **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と、**xfs_growfs** はデバイスで対応できる最大サイズにまで、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は `man xfs_growfs` を参照してください。

3.5. 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の内容を再現し、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現できない場合は **-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の詳細は、**man xfs_repair** を参照してください。

3.6. 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場合は、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のに、**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。

3.7. 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するには、**-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3.4 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。このダイアログで利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。

3.8. 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** で 1 つのファイルのみをデフラグできます。XFS はデフォルトで断片化の発生を防ぐため、ファイルシステム全体を定期的にデフラグしないことが推奨されます。システム全体でデフラグを行うと、空き領域が断片化する可能性があります。

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ツや、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の詳細は、各 **man** ページをご覧ください。

3.9. EXT4 から XFS への移行

Red Hat Enterprise Linux 7.0 より、デフォルトのファイルシステムが ext4 から XFS に変更になりました。本セクションでは、使用または管理時における XFS ファイルシステムの違いについて取り上げます。

ext4 ファイルシステムは Red Hat Enterprise Linux 7 でも完全にサポートされており、インストール時に ext4 を選択できます。ext4 から XFS への移行は可能ですが、必須ではありません。

3.9.1. XFS と ext3 および ext4 で使用されるコマンドの比較

以下の表は、ext3 および ext4 で使用される共通のコマンドと、XFS 固有のコマンドを比較しています。

表3.1 ext3 および ext4 の共通コマンドと XFS のコマンドの比較

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3.2 ext4 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

ここにリストされている多くの XFS コマンドの詳細は [3章XFS ファイルシステム](#) で確認できます。また、XFS 管理ツールの **man** ページで詳細情報を確認することもできます。

3.9.2. 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 は、起動時にユーザー空間で **e2fsck** を実行し、必要時にジャーナルを復元しますが、XFS は、マウント時にカーネル空間でジャーナルの復元を実行します。**fsck.xfs** シェルスクリプトは提供されますが、`initscript` の要件を満たすためだけに提供され、意味のあるアクションは実行しません。

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は、**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 **umount** 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の詳細は、[「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。

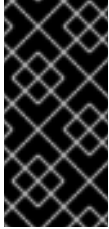
ファイルシステムのサイズ変更

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 inode があり 1 TB を超えるファイルシステムの場合や、512 バイトの inode があり 2 TB を超えるファイルシステムの場合、XFS の inode の数は 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` を呼び出す際に `Eoverflow` 戻り値が出力され失敗します。この問題は、4 つの割り当てグループを使用した非ストライプ化した、デフォルトの Red Hat Enterprise Linux 7 設定を使用していると発生する可能性があります。カスタム設定、たとえばファイルシステムを拡張、または XFS ファイルシステムのパラメーターを変更すると、挙動が異なる場合があります。

アプリケーションは、通常、このように数が大きい inode 番号を処理します。必要に応じて、**-o inode32** パラメーターを使用して XFS ファイルシステムをマウントして、inode の番号を強制的に 2 の 32 乗未満にします。**inode32** を使用しても、64 ビット番号で割り当ててある inode には影響しません。



重要

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。**inode32** オプションは割り当て動作を変更します。したがって、低レベルのディスクブロックに、inode の割り当てに利用できる領域がない場合は、ENOSPC エラーが発生する場合があります。

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、NFS サーバーでの同時ストリーミングの書き込みワークロードによって発生するファイル断片化を防ぎます。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と、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と、inode が解放 (reclaim) された時に事前割り当てが破棄されます。

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は、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 ファイルシステムで大きな問題となることはほとんどありません。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ツールは存在しますが、それを使用することは推奨されません。

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。コマンドが出力した内容を理解するには、多くの専門知識が必要になります。たとえば、断片化の割合が 75% というのは、ファイルごとの平均エクステンツ数が 4 しかないことを意味します。したがって、断片化の問題を調べるには、xfs_db フラグの出力だけでは十分とは言えず、より注意深く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。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ため、特に後者をデフラグすることは推奨されていません。

第4章 EXT3 ファイルシステム

ext3 ファイルシステムは、基本的に、ext2 ファイルシステムが拡張されたバージョンです。さまざまな改善点により、以下のような利点を提供されます。

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) が発生すると、マシンにマウントしている各 ext2 ファイルシステムは、**e2fsck** プログラムで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、システムの起動時間を著しく遅らせます。このプロセスの間、そのボリュームにあるデータは使用できません。

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、変更を加えることはできないため、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、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** を実行する方法も考えられます。

もしくは、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。保留中のメタデータ更新 (および書き込み) は、すべて再マウントの前にディスクへ強制的に入れられます。これにより、これ以前に破損がない限り、ファイルシステムは整合性を確保でき、**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が発生して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 の使用していても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、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は、ハードウェアのスピードにもよりますが、復元にかかる時間は 1 秒ほどになります。



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 よりもスループットが高くなります。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、モードの選択は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 設定を無効にすると提供され、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 パーティションを使用する場合は、以下の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、「[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

インストール後に、ext3 ファイルシステムを新たに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時に、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 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、その後に選択した UUID を **-U** オプションで指定し、**/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 に変換する場合は、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、可能な場合は、ext2 から ext3 に変換するのではなく、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 コマンドを実行して、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し、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
```

上記のコマンドの */mount/point* を、実際に使用する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続的に ext2 に変更したい場合は、忘れずに **/etc/fstab** ファイルを更新してください。更新せずにブートすると、変更した箇所が元に戻ります。

第5章 EXT4 ファイルシステム

ext4 ファイルシステムは、ext3 ファイルシステムに拡張を加えたファイルシステムです。Red Hat Enterprise Linux 7 では、ファイルサイズが最大 16 テラバイトまでサポートされます。また、Red Hat Enterprise Linux 6 でサポートされるファイルシステムは最大 16 テラバイトでしたが、Red Hat Enterprise Linux 7 では最大 50 テラバイトになりました。サポートされるサブディレクトリーの数は無制限 (ext3 ファイルシステムの場合は最大 32,000 までの対応) ですが、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** を実行する場合は、ext4 ボリュームのアンマウントが必要になります。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

(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異なり) ext4 はエクステントを使用し、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るため、ファイルシステムが大きくなるほどその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

遅延割り当てや他のパフォーマンスが最適化されるため、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** — クラッシュ発生後に、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 ファイルシステムの作成

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。出力結果には、ファイルシステムの配列や機能が表示されます。

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
```

ストライプ化されたブロックデバイス (RAID5 アレイ) の場合は、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改善されます。

LVM ボリュームや MD ボリュームにファイルシステムを作成する場合は、**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 の中でも、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または 1 ストライプ内のストライプユニット数を指定します。

どちらのサブオプションを使用する場合でも、**value** には、ファイルシステムのブロック単位で指定する必要があります。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) を指定する場合は、以下のようなコマンドになります。

```
# 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 を使用して、ext3 ファイルシステムで特定の ext4 機能を有効にすることができます。ただし、この方法で **tune2fs** を使用することは十分にテストされていないため、Red Hat Enterprise Linux 7 ではサポート対象外となります。したがって、Red Hat は、**tune2fs** で変換またはマウントした ext3 ファイルシステムに関しては、一貫性のあるパフォーマンスや、予測可能な動作を保証していません。

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 ファイルシステムのマウント

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、**data_err=abort** オプションを使用して、エラーがファイルデータに生じた場合にジャーナルを中止するようにできます。

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

tune2fs ユーティリティーを使用すると、管理者は、ファイルシステムのスーパーブロックにデフォルトのマウントオプションを設定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確保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。したがって、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、[21章書き込みバリア](#) を参照してください。

Direct Access のテクノロジープレビュー

Red Hat Enterprise Linux 7.3 以降、**Direct Access** (DAX) では、アプリケーションがアドレス空間に永続メモリーを直接マッピングする手段を、ext4 および XFS ファイルシステムでのテクノロジープレビューとして提供するようになりました。DAX を使用するには、永続メモリーがシステムに設定されている必要があります。永続メモリーは通常、1 つまたは複数の NVDIMM (Non-Volatile Dual In-line Memory Module) 形式で提供され、DAX をサポートするファイルシステムは NVDIMM に作成する必要があります。また、ファイルシステムは、**dax** のマウントオプションでマウントする必要があります。DAX でマウントしたファイルシステムにファイルを **mmap** すると、アプリケーションのアドレス空間にストレージが直接マッピングされます。

5.3. EXT4 ファイルシステムのサイズ変更

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** を使用して、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。

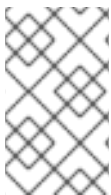
```
# resize2fs /mount/device size
```

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際に、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。特定の単位を指定するには、以下のサフィックスを使用します。

- **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 (多くの場合は不要です)。**resize2fs** は、論理ボリュームやパーティションなど、使用できるコンテナ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. EXT2、EXT3、EXT4 ファイルシステムのバックアップ

手順5.1 ext2、ext3、ext4 ファイルシステムのバックアップ例

1. なんらかの復元操作を実行する前に、すべてのデータをバックアップする必要があります。データは定期的にバックアップするようにしてください。データの他にも、**/etc/fstab** や **fdisk -l** の出力などの設定情報を保存する必要があります。sosreport/sysreport を実行し、設定情報を取得することが強く推奨されます。

```
# cat /etc/fstab
LABEL=/                /                    ext3    defaults    1 1
LABEL=/boot1           /boot               ext3    defaults    1 2
LABEL=/data            /data              ext3    defaults    0 0
tmpfs                  /dev/shm           tmpfs   defaults    0 0
devpts                 /dev/pts           devpts  gid=5,mode=620 0 0
sysfs                  /sys               sysfs   defaults    0 0
proc                   /proc              proc    defaults    0 0
LABEL=SWAP-sda5        swap               swap    defaults    0 0
/dev/sda6              /backup-files      ext3    defaults    0 0

# fdisk -l
   Device Boot      Start   End  Blocks  Id System
/dev/sda1 *          1      13   104391  83  Linux
/dev/sda2            14     1925  15358140  83  Linux
/dev/sda3           1926     3200   10241437+  83  Linux
/dev/sda4           3201     4864   13366080   5  Extended
```

/dev/sda5	3201	3391	1534176	82	Linux swap /
Solaris					
/dev/sda6	3392	4864	11831841	83	Linux

この例では、**/dev/sda6** パーティションにバックアップファイルを保存します。ここでは、**/backup-files** に **/dev/sda6** がマウントされていることを前提としています。

- バックアップしたパーティションが、オペレーティングシステムのパーティションである場合は、システムをシングルユーザーモードで起動してください。この手順は、通常のデータパーティションでは必要ありません。
- dump** を使用して、パーティションの内容をバックアップします。

注記

- システムが長時間稼働している場合は、バックアップを取得する前に、パーティションで **e2fsck** を実行することが推奨されます。
- 破損したバージョンのファイルがバックアップに含まれる可能性があるため、高負荷のマウントされたファイルシステムでは **dump** を使用しないでください。この問題は、dump.sourceforge.net に掲載されています。

重要

オペレーティングシステムのパーティションをバックアップする場合は、パーティションをアンマウントする必要があります。

通常のリデータパーティションは、マウントされている状態でバックアップすることも可能ですが、可能な場合はアンマウントしてバックアップすることが推奨されます。マウントされたデータパーティションをバックアップしようとすると、予想外の結果になる可能性があります。

```
# dump -0uf /backup-files/sda1.dump /dev/sda1
# dump -0uf /backup-files/sda2.dump /dev/sda2
# dump -0uf /backup-files/sda3.dump /dev/sda3
```

リモートでバックアップする場合は、ssh を使用するか、パスワードを使用しないログインを設定できます。

注記

標準のリダイレクションを使用する場合は、「-f」オプションを別に渡す必要があります。

```
# dump -0u -f - /dev/sda1 | ssh root@remoteserver.example.com dd
of=/tmp/sda1.dump
```

5.5. EXT2、EXT3、EXT4 ファイルシステムの復元

手順5.2 ext2、ext3、ext4 ファイルシステムの復元例

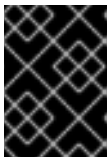
1. オペレーティングシステムのパーティションを復元する場合は、システムをレスキューモードで起動してください。この手順は、通常のデータパーティションでは必要ありません。
2. **fdisk** コマンドを使用して、sda1、sda2、sda3、sda4、および sda5 を再構築します。



注記

必要な場合は、復元したファイルシステムを格納するパーティションを作成します。作成するパーティションは、復元したデータを格納できる大きさである必要があります。開始および終了の番号が正しいことが重要になります。これらの番号はパーティションの開始および終了セクター番号になります。

3. 以下のように、**mkfs** コマンドを使用して宛先パーティションをフォーマットします。



重要

上述の **/dev/sda6** はバックアップファイルを保存するのに使用するため、フォーマットはしないでください。

```
# mkfs.ext3 /dev/sda1
# mkfs.ext3 /dev/sda2
# mkfs.ext3 /dev/sda3
```

4. 新しいパーティションを作成する場合は、**fstab** ファイルと一致させるため、すべてのパーティションのラベルを変更してください。パーティションを再作成していない場合、この手順は必要ではありません。

```
# e2label /dev/sda1 /boot1
# e2label /dev/sda2 /
# e2label /dev/sda3 /data
# mkswap -L SWAP-sda5 /dev/sda5
```

5. 作業ディレクトリを準備します。

```
# mkdir /mnt/sda1
# mount -t ext3 /dev/sda1 /mnt/sda1
# mkdir /mnt/sda2
# mount -t ext3 /dev/sda2 /mnt/sda2
# mkdir /mnt/sda3
# mount -t ext3 /dev/sda3 /mnt/sda3
# mkdir /backup-files
# mount -t ext3 /dev/sda6 /backup-files
```

6. データを復元します。

```
# cd /mnt/sda1
# restore -rf /backup-files/sda1.dump
# cd /mnt/sda2
```

```
# restore -rf /backup-files/sda2.dump
# cd /mnt/sda3
# restore -rf /backup-files/sda3.dump
```

リモートホストから復元する場合や、リモートホストのバックアップファイルから復元する場合は、ssh または rsh を使用することができます。以下の例を実行するには、パスワードが不要なログインを設定する必要があります。

10.0.0.87 にログインし、ローカルの sda1.dump ファイルから sda1 を復元します。

```
# ssh 10.0.0.87 "cd /mnt/sda1 && cat /backup-files/sda1.dump |
restore -rf -"
```

10.0.0.87 にログインし、リモートの 10.66.0.124 sda1.dump ファイルから sda1 を復元します。

```
# ssh 10.0.0.87 "cd /mnt/sda1 && RSH=/usr/bin/ssh restore -r -f
10.66.0.124:/tmp/sda1.dump"
```

7. 再起動します。

5.6. EXT4 ファイルシステムのその他のユーティリティ

Red Hat Enterprise Linux 7 には、他にも ext4 ファイルシステム管理ユーティリティがあります。

e2fsck

ext4 ファイルシステムの修復時に使用します。ext4 でディスク構造が変更されたため、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 ファイルシステムで、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際、フリーズにはコマンド **# 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 ファイルシステムのマウント](#)」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、ext4 ファイルシステムのデバッグや分析を行う際には次のツールが役に立ちます。

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の詳細は、各 **man** ページをご覧ください。

第6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。

6.1. BTRFS ファイルシステムの作成

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詳細な方法は、「[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。

6.2. BTRFS ファイルシステムのマウント

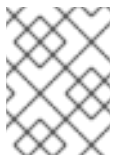
Btrfs ファイルシステムでデバイスをマウントするには、以下のコマンドを使用します。

```
# 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 ボリューム用に指定されたデバイスをスキャンします。このオプションは、Btrfs 以外のデバイスのマウントを試みるとマウントに失敗するため、確実にマウントを成功させるために使用します。



注記

これは、すべてのデバイスをスキャンすることを示しているだけで、すべてのデバイスがファイルシステムに追加されるわけではありません。

max_inline=number

このオプションは、メタデータ B ツリー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 ページの場合は、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

このオプションは、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

このオプションは、割り当てられるワーカースレッドの数を指定します。

discard

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

noacl

このオプションは、ACL の使用を無効にします。

space_cache

このオプションは、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

このオプションは、上記の **space_cache** を無効にします。

clear_cache

このオプションは、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。このオプションは安全ですが、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します。

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

このオプションは、マウント時の自動リカバリーを有効にします。

6.3. BTRFS ファイルシステムのサイズ変更

btrfs ファイルシステムのサイズを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は変更できます。使用しているデバイスが 1 つしかない場合は、ファイルシステムのサイズ変更と同様のことが行われます。使用しているデバイスが複数ある場合は、手動でのサイズ変更が必要になります。



注記

単位サイズは大文字と小文字を区別しないため、GiB には **G** と **g** の両方を使用できません。

このコマンドは、テラバイトを表す **t** や、ペタバイトを表す **p** を使用することはできません。使用できるのは、**k**、**m**、および **g** のみとなります。

Btrfs ファイルシステムの拡大

単一のデバイスでファイルシステムを拡大するには、以下のコマンドを使用します。

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

以下に例を示します。

```
# btrfs filesystem resize +200M /btrfsingle
Resize '/btrfsingle' of '+200M'
```

複数デバイスのファイルシステムを拡大するには、拡大するデバイスを指定する必要があります。最初に、特定のマウントポイントに btrfs ファイルシステムがあるデバイスをすべて表示します。

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

以下に例を示します。

```
# btrfs filesystem show /btrfstest
Label: none  uuid: 755b41b7-7a20-4a24-abb3-45fdbed1ab39
Total devices 4 FS bytes used 192.00KiB
devid    1 size 1.00GiB used 224.75MiB path /dev/vdc
devid    2 size 524.00MiB used 204.75MiB path /dev/vdd
devid    3 size 1.00GiB used 8.00MiB path /dev/vde
devid    4 size 1.00GiB used 8.00MiB path /dev/vdf

Btrfs v3.16.2
```

拡大するデバイスの **devid** を確認した後、以下のコマンドを使用します。

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

以下に例を示します。

```
# btrfs filesystem resize 2:+200M /btrfstest
Resize '/btrfstest/' of '2:+200M'
```



注記

amount には、特定の容量ではなく **max** を指定することでもできます。**max** にすると、デバイスの空き容量がすべて使用されます。

btrfs ファイルシステムの縮小

単一のデバイスでファイルシステムを縮小するには、以下のコマンドを使用します。

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

以下に例を示します。

```
# btrfs filesystem resize -200M /btrfssingle
Resize '/btrfssingle' of '-200M'
```

複数デバイスのファイルシステムを縮小するには、縮小するデバイスを指定する必要があります。最初に、特定のマウントポイントに btrfs ファイルシステムがあるデバイスをすべて表示します。

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

以下に例を示します。

```
# btrfs filesystem show /btrfstest
Label: none  uuid: 755b41b7-7a20-4a24-abb3-45fdbed1ab39
Total devices 4 FS bytes used 192.00KiB
```

```
devid    1 size 1.00GiB used 224.75MiB path /dev/vdc
devid    2 size 524.00MiB used 204.75MiB path /dev/vdd
devid    3 size 1.00GiB used 8.00MiB path /dev/vde
devid    4 size 1.00GiB used 8.00MiB path /dev/vdf
```

Btrfs v3.16.2

縮小するデバイスの **devid** を確認した後、以下のコマンドを使用します。

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

以下に例を示します。

```
# btrfs filesystem resize 2:-200M /btrfstest
Resize '/btrfstest' of '2:-200M'
```

ファイルシステムのサイズ設定

単一のデバイスでファイルシステムを特定のサイズに設定するには、以下のコマンドを使用します。

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

以下に例を示します。

```
# btrfs filesystem resize 700M /btrfssingle
Resize '/btrfssingle' of '700M'
```

複数デバイスのファイルシステムのサイズを設定するには、変更するデバイスを指定する必要があります。最初に、特定のマウントポイントに btrfs ファイルシステムがあるデバイスをすべて表示します。

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

以下に例を示します。

```
# btrfs filesystem show /btrfstest
Label: none  uuid: 755b41b7-7a20-4a24-abb3-45fdbed1ab39
Total devices 4 FS bytes used 192.00KiB
devid    1 size 1.00GiB used 224.75MiB path /dev/vdc
devid    2 size 724.00MiB used 204.75MiB path /dev/vdd
devid    3 size 1.00GiB used 8.00MiB path /dev/vde
devid    4 size 1.00GiB used 8.00MiB path /dev/vdf
```

Btrfs v3.16.2

変更するデバイスの **devid** を確認した後、以下のコマンドを使用します。

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

以下に例を示します。

```
# btrfs filesystem resize 2:300M /btrfstest
Resize '/btrfstest' of '2:300M'
```

6.4. 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、多くのデバイスの上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、デバイスが1つしかない場合、メタデータはそのデバイス上に複製されます。

6.4.1. 複数デバイスによるファイルシステムの作成

「[Btrfs ファイルシステムの作成](#)」で詳細に説明されているように、**mkfs.btrfs** コマンドではオプションとして、データ **-d** とメタデータ **-m** を指定できます。有効な仕様は以下の通りです。

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

-m single オプションは、メタデータの複製を行わ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、デバイスが少なくとも4つは必要です。

例6.1 raid10 btrfs ファイルシステムの作成

4つのデバイスにわたる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に、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 -d single /dev/device1 /dev/device2 /dev/device3
```

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。

```
# btrfs device add /dev/device1 /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。詳細は、「[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。

6.4.2. 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
```

1 つ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 /dev/device
```

6.4.3. 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、そのファイルシステムに含まれるデバイスを一覧表示します。

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ツ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上記のコマンド例は以下の通りです。

例6.2 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詳細な方法は「[Btrfs ファイルシステムの作成](#)」、btrfs ファイルシステムをマウントする詳細な方法は「[Btrfs ファイルシステムのマウント](#)」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 つ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。ファイルシステムにあるデータとメタデータをすべて読み取り、それを新規デバイス全体に再度書き込むためです。

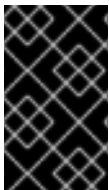
6.4.4. btrfs ファイルシステムの変換

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6.3 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。

6.4.5. 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ツをファイルシステムの他のデバイスに再配信します。

例6.4 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
```

6.4.6. 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が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

デバイスが見つからないこと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す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 レイアウトで、デバイスが失敗した場合には、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

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

initrd がない場合、または **btrfs** デバイススキャンを実行しない場合、ファイルシステムのすべてのデバイスを **mount** コマンドに明示的に渡すと、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6.5 /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=/dev/sdd,device=/dev/sde    0
```

UUID (universally unique identifier) を使用することもできます。UUID は、デバイスパスよりも安定性が高くなります。

6.5. SSD の最適化

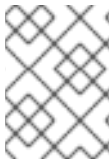
btrfs ファイルシステムを使用して SSD を最適化することができます。これは 2 つの方法で実行することができます。

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 경우가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があります。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。

6.6. 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のプロセスが 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. NFS の仕組み

現在、Red Hat Enterprise Linux には 2 つのメジャーバージョンの NFS が含まれています。NFS バージョン 3 (NFSv3) は安全な非同期書き込みに対応し、NFSv2 よりも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およびオフセットをサポートするため、クライアントは 2 GB を超えるファイルデータにアクセスできます。NFSv4 はファイアウォールを介して動作し、インターネット上でも動作します。さらに、**rpcbind** サービスが必要なく、ACL をサポートし、ステートフルな操作を利用します。

Red Hat Enterprise Linux 7 には NFS バージョン 4.1 (NFSv4.1) のサポートが追加され、Parallel NFS (pNFS) のクライアント側サポートを含むパフォーマンスおよびセキュリティの機能が複数強化されました。NFSv4.1 ではコールバックで個別の TCP 接続が不要になり、(NAT やファイアウォールが妨害する場合など) クライアントと通信できなくても、NFS サーバーは委任を許可できます。さらに、NFSv4.1 では exactly once セマンティックが提供されるようになり (再起動操作を除く)、返答が失われ、操作が 2 度送信された場合に、特定の操作が不正な結果を返す可能性のあるこれまでの問題が発生しないようになりました。

Red Hat Enterprise Linux 7 は NFSv3、NFSv4.0、および NFSv4.1 クライアントをサポートします。NFS クライアントは、デフォルトでは NFSv4.0 を使用してマウントを試行し、マウント操作に失敗すると NFSv3 にフォールバックします。



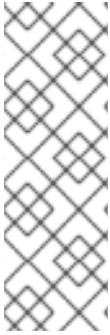
注記

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、(通常の状況では) 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。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になります。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のプロトコルが 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 では、デーモン **rpcbind**^[1]、**lockd**、**rpc.statd** とのやりとりが必要なくなります。**rpc.mountd** デーモンは、NFS サーバーでエクスポートをセットアップするのに必要ですが、送信オペレーションには関与しません。



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。

すべての RPC/NFS デーモンには '**-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、そのクライアントがエクスポート済みファイルシステムへのアクセスできるかどうかを確認します。アクセスが可能なが確認されると、そのユーザーは、ファイルおよびディレクトリーへの全操作を行えるようになります。



重要

ファイア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に、IPTables は、デフォルトの TCP ポート 2049 に設定してください。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のに、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のに使用されていました。このサービスは、Red Hat Enterprise Linux 7 でIPv6 に対応にするため、**rpcbind** に置き換えられています。

nfs

systemctl start nfs 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

systemctl start nfs-lock は、適切な RPC プロセスを起動する必須サービスをアクティベートし、NFS クライアントがサーバー上のファイルをロックできるようにします。

rpcbind

rpcbind で、ローカルの RPC サービスからポート予約を受け取ると、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

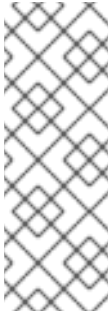
Network Status Monitor (NSM) RPC プロトコルを実装します。NFS サーバーが正常にシャットダウンされず再起動すると、NFS クライアントに通知します。**rpc.statd** は、**nfslock** サービスによって自動的に起動されるため、ユーザー設定を必要としません。このプロセスは NFSv4 では使用されません。

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。最低でも NFSv4 マッピングドメインを定義する「Domain」パラメーターを指定する必要があります。NFSv4 マッピングドメインが DNS ドメイン名と同じであると、このパラメーターをスキップできます。クライアントとサーバーが ID マッピングの NFSv4 マッピングドメインに一致しないと、適切に動作しません。

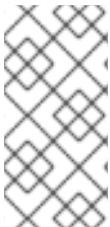


注記

Red Hat Enterprise Linux 7 では、NFSv4 サーバーのみが **rpc.idmapd** を使用します。NFSv4 はキーリングベースの idmapper の **nfsidmap** を使用します。**nfsidmap** は要求に応じてカーネルによって呼び出され、ID マッピングを実行するスタンドアロンプログラムで、デーモンではありません。**nfsidmap** に問題がある場合、クライアントは **rpc.idmapd** を使用してフォールバックします。**nfsidmap** の詳細は、**nfsidmap** の man ページを参照してください。

8.2. PNFS

Parallel NFS (pNFS) のサポートは、NFS v4.1 標準の一部として Red Hat Enterprise Linux 6.4 で提供されました。pNFS アーキテクチャーは NFS の拡張性を向上させるため、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 クライアントが対応しているレイアウトタイプはファイルだけで、オブジェクトとブロックはテクノロジープレビューになります。

この機能を有効にするには、pNFS が有効になっているサーバーのマウント時に、以下のマウントオプションを使用します。

```
-o v4.1
```

サーバーで pNFS を有効にすると、最初のマウント時に **nfs_layout_nfsv41_files** カーネルが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 server:/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に、**/etc/fstab** ファイルに行を追加する方法があります。その行には、NFS サーバーのホスト名、エクスポートされるサーバーディレクトリー、および NFS 共有がマウントされるローカルマシンディレクトリーを記述する必要があります。**/etc/fstab** ファイルを修正するには root 権限が必要です。

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。

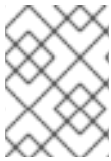
```
server:/usr/local/pub    /pub    nfs    defaults 0 0
```

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。

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。

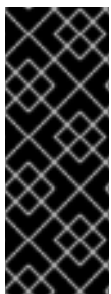
`/etc/fstab` の詳細情報については、`man fstab` を参照してください。

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。1 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。1 つのダイレクトマップ内にある複数のエントリー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ーで表されます。-**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できるようになり、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できるようになります。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。

1 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ー

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 1 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ー

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。マップエントリー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。エントリー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ー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ー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の無効化/拡大

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ーが含まれます。

```
*      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ー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ーを組み込み、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 を行くと、上述の NIS **auto.home** マップに従い、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. LDAP を使用した自動マウント機能のマップの格納

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ー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、手動で **mount** コマンド、**/etc/fstab** 設定、**autofs** などを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ー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。バージョンを指定しないと、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せん。

noacl

ACP の処理をすべてオフにします。古いバージョンの Red Hat Enterprise Linux、Red Hat Linux、Solaris と連動させる場合に必要となることがあります。こうした古いシステムには、最新の ACL テクノロジーに対する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、古いバージョンの NFS サーバーに接続する場合に必要な場合があります。

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 は、NFS サーバーポートの数値を指定します。**num** を **0** (デフォルト) にすると、**mount** は、使用するポート番号に関するリモートホストの **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) を大きくすると、NFS 通信の読み込み (**rsize**) と書き込み (**wsize**) の速度が上がります。Linux カーネルやネットワークカードが最新でないと、ブロックサイズを大きくしたときに正しく動作しなくなる場合があるため、これらの値を変更する際には注意が必要です。NFSv3 の場合、これらのパラメーターのデフォルト値はいずれも 8192 に設定されます。NFSv4 の場合は 32768 に設定されます。

sec=mode

デフォルト設定は **sec=sys** で、ローカルの UNIX UID および GID を使用します。**AUTH_SYS** を使用して NFS 操作を認証します。

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安全なチェックサムを使って、NFS 動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

NFSv4 だけを使用するように設定されていない NFS サーバーを実行するには、**rpcbind**^[1] サービスを実行している必要があります。**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。

```
# systemctl status rpcbind
```

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。

```
# systemctl start nfs
```

起動時に NFS を起動できるようにするには、以下のコマンドを使用します。

```
# systemctl enable nfs-server
```

注記

NFSv3 では、NFS が起動時に開始するよう設定されている場合は **nfs-lock** サービスを有効にする必要があります。Red Hat Enterprise Linux 7.1 以降では、必要時に **nfs-lock** が自動的に開始し、手動で有効にできません。Red Hat Enterprise Linux 7.0 では、**systemctl status nfs-lock** を実行して状態をチェックします。**nfs-lock** が有効になっていない場合は、**systemctl start nfs-lock** を実行します。**nfs-lock** が、Red Hat Enterprise Linux 7.0 の起動時に自動的に起動するよう設定するには、**systemctl enable nfs-lock** を実行してください。

サーバーを停止させるには、以下を使用します。

```
# systemctl stop nfs
```

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てから、その変更を有効にする際の最も効率的な方法です。このサーバーを再起動するには、以下のコマンドを使用します。

```
# systemctl restart nfs
```

/etc/sysconfig/nfs ファイルを編集したら、以下のコマンドを実行して **nfs-config** サービスを再起動し、新しい値が適用されるようにします。

```
# systemctl restart nfs-config
```

try-restart コマンドは、**nfs** が現在稼働中である場合に限り起動させます。このコマンドは、Red Hat init スクリプトの **condrestart** (*conditional restart*) と同様、NFS が実行されていないとデーモンを起動しないため便利です。

条件付きでサーバーを再起動するには、以下を入力します。

```
# systemctl try-restart nfs
```

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。

```
# systemctl reload nfs
```

8.7. NFS サーバーの設定

NFS サーバーのエクスポートを設定する方法は 2 つあります。

- NFS の設定ファイル **/etc/exports** を手動で編集する方法。
- コマンドラインで、コマンド **exportfs** を使用する方法。

8.7.1. /etc/exports 設定ファイル

/etc/exports ファイルは、リモートホストにどのファイルシステムをエクスポートするかを制御し、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。
- コメント行は、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。
- エクスポートするファイルシステムは、それぞれ 1 行で指定する。
- 許可するホストの一覧は、エクスポートするファイルシステムの後空白文字を追加し、その後追加する。
- 各ホストのオプションは、ホスト識別子の直後に括弧を追加し、その中に指定する。ホストと最初の括弧の間には空白を入れない。

エクスポートするファイルシステムのエントリーは、以下のように指定します。

```
export host(options)
```

ここでは、以下のような変数を使用しています。

export

エクスポートするディレクトリー

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

各ホストにそれぞれオプションを付けて、複数のホストを 1 行で指定することができます。この場合は、以下のように、各ホスト名の後に、そのホストに対するオプションを括弧を付けて追加します。ホストは空白文字で区切ります。

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

ホスト名を指定する、その他の方法については、「[ホスト名の形式](#)」を参照してください。

最も簡単な方法は、**/etc/exports** ファイルに、エクスポートするディレクトリーと、そのディレクトリーへのアクセスを許可す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
```

この例では、**bob.example.com** が、NFS サーバーの **/exported/directory/** をマウントできます。ここではオプションが指定されていないため、NFS では、**デフォルト** 設定が使用されます。

デフォルトの設定は以下のようになります。

ro

エクスポートする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、このファイルシステムで共有されているデータを変更することができません。このファイルシステムで変更 (読み込み/書き込み) を可能にするには、**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、要求に応答しません。代わりに非同期書き込みを有効にするには、**async** オプションを指定します。

wdelay

NFS サーバーは、別の書き込み要求が差し迫っていると判断すると、ディスクへの書き込みを遅らせます。これにより、複数の書き込みコマンドが同じディスクにアクセスする回数を減らすことができるため、書き込みのワークロードが低下し、パフォーマンスが向上します。これを無効にするには、**no_wdelay** を指定します。**no_wdelay** は、デフォルトの **sync** オプションが指定されている場合に限り利用可能になります。

root_squash

(ローカルからではなく) リモート から接続している root ユーザーが root 権限を持つことを阻止します。代わりに、そのユーザーには、NFS サーバーによってユーザー ID **nfsnobody** が割り当てられます。これにより、リモートの root ユーザーの権限を、最も低いローカルユーザーレベルにまで下げ (squash) て、権限を持たずにリモートサーバーに書き込むのを阻止します。**no_root_squash** と指定すると、この root squashing が無効になります。

(root を含む) すべてのリモートユーザーの権限を下げるには、**all_squash** を使用します。特定ホストのリモートユーザーに対して、NFS サーバーが割り当てるユーザー 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** のオプションを使用することで、リモート NFS ユーザーが共有する特別なユーザーおよびグループアカウントを作成できるようになります。

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 では 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。

エクスポートするファイルシステムのデフォルトはすべて、明示的に上書きする必要があります。たと

例えば、**rw** オプションを指定しないと、エクスポートするファイルシステムは読み込み専用として共有されます。以下は、**/etc/exports** の例になりますが、ここでは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)

このエントリーでは、**192.168.0.3** が **/another/exported/directory/** を読み込み/書き込みでマウントできるように設定しています。そして、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は、**man exports** 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オプションも利用できます。たとえば、サブツリーチェックを無効にする、安全でないポートからのアクセスの許可する、安全でないファイルロックを許可する(一部の初期 NFS クライアント実装で必要)などの機能があります。これらのオプションは頻繁には使用しませんが、その詳細については **man exports** を参照してください。

重要

/etc/exports ファイルでは、特に空白文字については、非常に厳しく扱われます。ホストとエクスポートされるファイルシステムの間、そしてホスト同士の間には、必ず空白文字を挿入してください。また、それ以外の場所(コメント行を除く)には、絶対に空白文字を追加しないでください。

たとえば、以下の2つの行は意味が異なります。

```
/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** からのユーザーには、ディレクトリーを読み込み専用(デフォルト)でマウントすることを許可しており、その他のユーザーには、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となります。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続的に追加する前に、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。**exportfs** コマンドを実行するときに表示されるエクスポート、または非エクスポートのファイルシステムの情報が、より詳細に表示されます。

exportfs コマンドにオプションを渡さない場合と、現在エクスポートされているファイルシステムの一覧が表示されます。**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。

クライアントが NFSv4 を使用しないようにするには、**/etc/sysconfig/nfs** に **RPCNFSDARGS= -N 4** を設定し、NFSv4 の使用を停止します。

8.7.3. ファイアウォール背後での NFS の実行

NFS は RPC サービスのポートを動的に割り当てる **rpcbind** を必要としますが、ファイアウォールルールの設定で問題が発生する可能性があります。ファイアウォールの背後でクライアントが NFS 共有にアクセスできるようにするには、**/etc/sysconfig/nfs** ファイルを編集し、RPC サービスが実行されるポートを設定します。

/etc/sysconfig/nfs ファイルは、デフォルトではすべてのシステムに存在しているわけではありません。**/etc/sysconfig/nfs** が存在しない場合は作成し、以下を指定します。

RPCMOUNTDOPTS="-p port"

これにより、`rpc.mount` コマンドラインに「-p port」が追加され、**rpc.mount -p port** になります。

nlockmgr サービスが使用するポートを指定するには、`/etc/modprobe.d/lockd.conf` ファイルの **nlm_tcpport** オプションと **nlm_udpport** オプションに、ポート番号を設定します。

NFS の起動に失敗すると、`/var/log/messages` がチェックされます。一般的に、すでに使用されているポート番号を指定した場合に NFS の起動に失敗します。`/etc/sysconfig/nfs` の編集後、以下のコマンドを実行して **nfs-config** サービスを再起動し、Red Hat Enterprise Linux 7.2 以下のバージョンで新しい値が反映されるようにします。

```
# systemctl restart nfs-config
```

NFS サーバーを再起動します。

```
# systemctl restart nfs-server
```

rpcinfo -p を実行し、変更が反映されたことを確認します。

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。

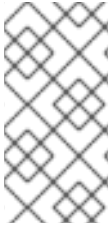
1 つ目は、NFSv2 または NFSv3 をサポートするいずれかのサーバー上で、**showmount** コマンドの使用です。

```
$ showmount -e myserver
Export list for myserver
/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 を有効にしていなかったため、これは通常、問題にはなっていません。

8.7.4. ファイアウォールからの **RPC** クォータを許可する

ディスククォータを使用するファイルシステムをエクスポートする場合は、RPC (Remote Procedure Call) サービスを使用して、NFS クライアントにディスククォータデータを提供できます。

手順8.1 ファイアウォールの背後で **RPC** クォータのアクセスを可能にする

1. **rpc-rquotad** サービスを有効にするには、以下を入力します。

```
# systemctl enable rpc-rquotad
```

2. **rpc-rquotad** サービスを起動するには、以下を入力します。

```
# systemctl start rpc-rquotad
```

rpc-rquotad が有効な場合は、**nfs-server** サービスが起動した後に自動的に起動します。

3. ファイアウォールの背後で、クォータ RPC サービスにアクセスできるようにするには、UDP ポートまたは TCP ポート **875** を開く必要があります。デフォルトのポート番号は **/etc/services** ファイルに指定します。

デフォルトのポート番号は、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-p port-number** を追加すると上書きできます。

4. **/etc/sysconfig/rpc-rquotad** ファイルへの変更を有効にするには、**rpc-rquotad** を再起動します。

```
# systemctl restart rpc-rquotad
```

リモートホストにクォータを設定する

デフォルトでは、リモートホストだけがクォーターを読み込めます。クォータを設定できるようにするには、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-S** オプションを追加します。

/etc/sysconfig/rpc-rquotad ファイルへの変更を有効にするには、**rpc-rquotad** を再起動します。

```
# systemctl restart rpc-rquotad
```

8.7.5. 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。

8.7.6. RDMA で NFS を有効にする (NFSv4.1 over RDMA)

Red Hat Enterprise Linux 7 では、RDMA に対応するハードウェアが存在すると、RDMA (remote direct memory access) サービスが自動的に有効になります。

rdma パッケージをインストールします。**/etc/rdma/rdma.conf** ファイルにはデフォルトで **XPRT_RDMA_LOAD=yes** が設定されており、NFSv4.1 over RDMA クライアントモジュールをロードする **rdma** サービスが必要になります。

NFSv4.1 over RDMA サーバー モジュールを自動的にロードするには、**/etc/rdma/rdma.conf** に **SVC_RDMA_LOAD=yes** 行を追加します。

/etc/sysconfig/nfs ファイルの **RPCNFSDARGS="--rdma=20049"** は、NFSv4.1 over RDMA サービスがクライアントをリッスンするポート番号 (20049) を指定します。『RFC 5667』では、RDMA で NFSv4 サービスを提供する場合は、サーバーがポート **20049** をリッスンすることが定められています。

/etc/rdma/rdma.conf ファイルを編集したら、**nfs** サービスを再起動します。

```
# service nfs restart
```

以前のカーネルバージョンで **/etc/rdma/rdma.conf** の変更を有効にするには、システムを再起動する必要があります。

8.8. NFS の保護

NFS は、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適しています。ただし、使いやすさがある反面、さまざまなセキュリティ問題を伴います。サーバーにおける NFS セキュリティリスクを最低限に抑え、データを保護するために、サーバー上の NFS ファイルシステムをエクスポートする場合や、クライアントにマウントする場合に、以下のセクションを考慮に入れるようにしてください。

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を使って、どのホストにど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** とも言います) 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設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。ただし、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、未承認のマシンに向かう可能性があります。この時、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 には、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できます。FIPS モードでは、FIPS が許可するアルゴリズムのみを使用できます。

AUTH_SYS とは異なり、RPCSEC_GSS Kerberos メカニズムでは、ファイルにアクセスしているユーザーを正確に表示することを、サーバーがクライアントに依存しなくなりました。代わりに、暗号を使用してサーバーに対してユーザー認証を行うため、Kerberos 情報を持たない悪意あるクライアントがなりすまし攻撃を行うことはできなくなります。RPCSEC_GSS Kerberos メカニズムでは、Kerberos の設定後、追加の変更を行わなくても動作するため、これがマウントをセキュアにする最も簡単な方法となります。

Kerberos の設定

NFSv4 Kerberos に対応するサーバーを設定する前に、KDC (Kerberos Key Distribution Centre) をインストールして設定する必要があります。Kerberos は、対象暗号化と信頼されているサードパーティー (KDC) を使用したネットワーク認証システムで、これによりクライアントとサーバーが互いに認証できるようになります。Red Hat は、Kerberos の設定に IdM (Identity Management) を使用することが推奨されます。

手順8.2 IdM が RPCSEC_GSS を使用するよう、NFS サーバーおよびクライアントを設定する

1. o NFS サーバーサイドに、**nfs/hostname.domain@REALM** プリンシパルを作成します。

- サーバーサイドとクライアントサイドに、**host/hostname.domain@REALM** プリンシパルを作成します。
- 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。

詳細は、『Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイド』の「[サービスエントリーおよび Keytab の追加と編集](#)」セクションと「[Kerberos 対応の NFS サーバーの設定](#)」セクションを参照してください。

2. サーバーサイドでは、**sec=** オプションを使用して、必要なセキュリティーフレーバーを有効にします。すべてのセキュリティーフレーバーと非暗号化マウントを有効にするには、次のコマンドを実行します。

```
/export *(sec=sys:krb5:krb5i:krb5p)
```

sec= オプションを使用するのに有効なセキュリティーフレーバーは、以下のようになります。

- **sys**: 非暗号化保護、デフォルト
 - **krb5**: 認証のみ
 - **krb5i**: 整合性保護
 - **krb5p**: プライバシー保護
3. クライアントサイドでは、**sec=krb5** (設定によっては **sec=krb5i** もしくは **sec=krb5p** となります) をマウントオプションに追加します。

```
# mount -o sec=krb5 server:/export /mnt
```

NFS クライアントの設定方法は、『Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイド』の「[Kerberos 対応の NFS クライアントの設定](#)」セクションを参照してください。

Red Hat は IdM を使用することを推奨していますが、AD (Active Directory) Kerberos サーバーもサポート対象となります。詳細は、Red Hat ナレッジベースの記事「[How to set up NFS using Kerberos authentication on RHEL 7 using SSSD and Active Directory](#)」を参照してください。

詳細は、`exports(5)` および `nfs(5)` の man ページと、「[一般的な NFS マウントオプション](#)」を参照してください。

gssproxy と **rpc.gssd** を同時に使用方法など、**RPCSEC_GSS** フレームワークの詳細は、「[GSSD flow description](#)」を参照してください。

8.8.2.1. NFSv4 による NFS の保護

NFSv4 には、Microsoft Windows NT モデルの機能や幅広い導入の経緯があるため、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 でセキュリティーに関する、もう 1 つの重要な特長は、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りました。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

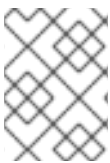
リモートホストにより、NFS ファイルシステムが読み取りと書き込みの権限でマウントされると、それぞれの共有ファイルを保護できるのはその権限のみになります。同じユーザー ID 値を共有している 2 ユーザーが同じ NFS ファイルシステムをマウントすると、それらのユーザーはファイルを相互に変更することができます。さらに、クライアントシステムに root としてログインするいずれのユーザーも、**su** - コマンドを使用して NFS 共有経由のファイルにすべてアクセスできます。

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、この機能を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 のプロセスが開始すると、その開始が **rpcbind** に通知され、そのプロセスがリ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が、特定の RPC プログラム番号を使って、サーバーの **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、クライアントの受信要求の接続を確立するのに、必ず **rpcbind** を使用します。したがって、RPC ベースのサービスが起動する前に、**rpcbind** が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
```

正しく開始していない NFS サービスが1つでもあると、**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、**rpcinfo** 出力に NFS がない場合、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。詳細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に使用する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/> — 開発者向けのホームページで、少し古いですが、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://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] **rpcbind** サービスは、旧バージョンの Red Hat Enterprise Linux で、各 RPC プログラムの番号を、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ために使用していた **portmap** に代わります。詳細は「[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。

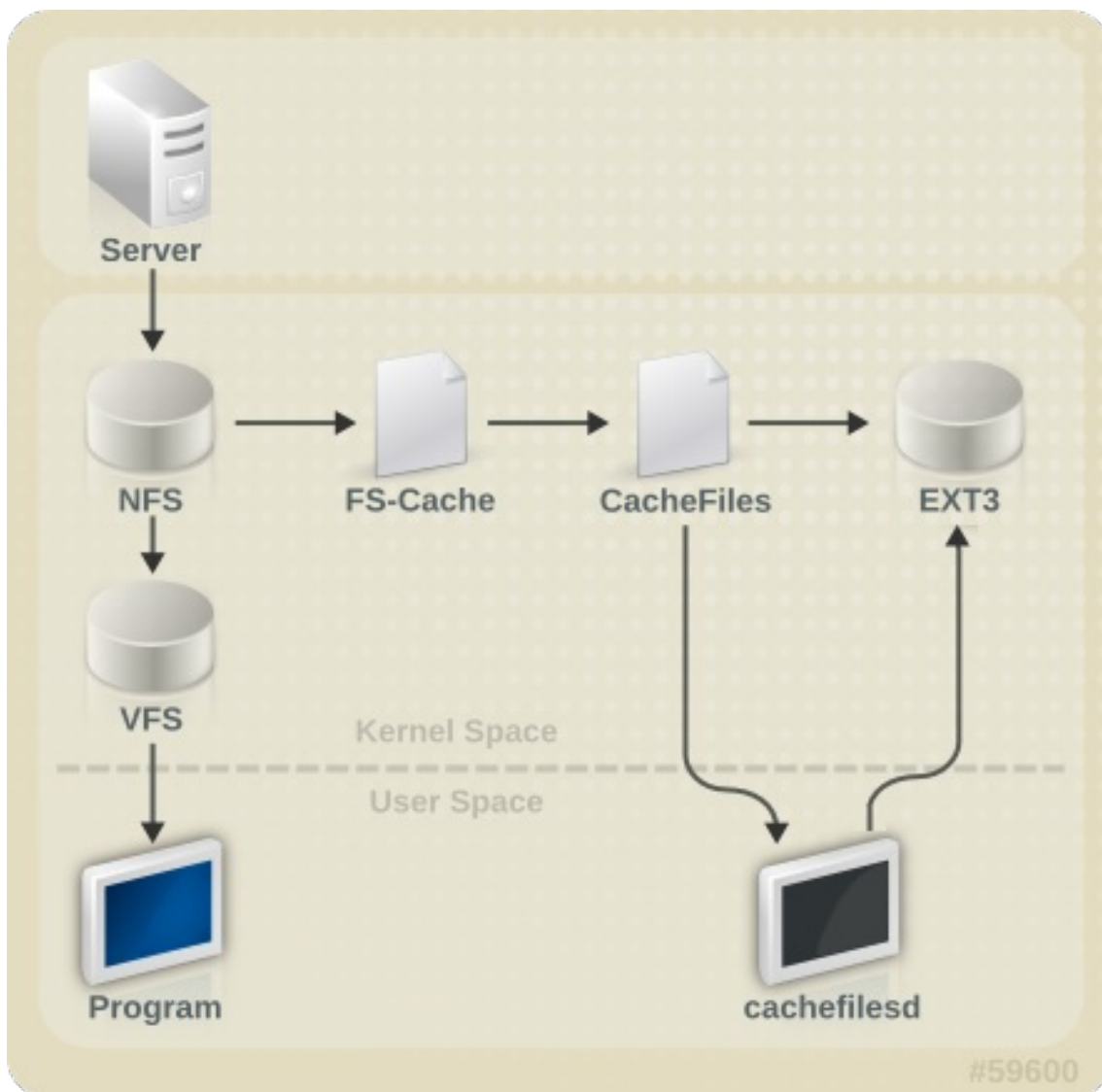


図9.1 FS-Cache の概要

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 7 には、**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. キャッシュの設定

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 \(Cache Cull\) の設定](#)」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 でのキャッシュの使用

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,rsiz=230
```

```
mount home0:/disk0/jim /home/jim -o fsc,rsiz=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,rsiz=230
```

```
mount home0:/disk0/fred /home/fred2 -o fsc,rsiz=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) の設定

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%

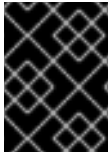
- **bstop/fstop - 3%**

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。

$0 \leq \mathbf{bstop} < \mathbf{bcull} < \mathbf{brun} < 100$

$0 \leq \mathbf{fstop} < \mathbf{fcull} < \mathbf{frun} < 100$

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、**bxxx** と **fxxx** 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」セクションでは、最初に Red Hat Enterprise Linux 7 のストレージに関する考察を説明し、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詳細に説明した後、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。

IBM System Z における DASD デバイスと zFCP デバイス

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、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 [17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

Fibre Channel over Ethernet (FCoE) ドライブのプラグアンドプレイ検出は、EDD で起動可能な 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っ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書き込みパス内にないため、バッファ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、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。



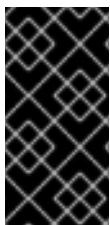
注記

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。



重要

マシンが起動しない場合、ファイルシステムが極めて大きい場合、またはファイルシステムがリモートストレージにある場合を除き、Red Hat はファイルシステムチェックを推奨しません。`/etc/fstab` の 6 番目のフィールドを 0 に設定すると起動時にファイルシステムチェックを無効にすることができます。

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でのみの操作

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。これは、ブート時に **fsck.filesystem** バイナリーを検索する `initrd` 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はありません。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. エクステンツのチェック。
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

parted ユーティリティを使用すると、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
- 空き領域または他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インストールされています。**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 **parted /dev/sda** コマンドを入力します (**/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。使用中のデバイスに新規パーティションを作成することはできますが、推奨はされません。

使用の終了が予定されるデバイス上のパーティションは、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、情報が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表示されたら、**Skip** を選択します。

また、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、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** コマンドが示されています。ここに記載されているコマンドおよび引数の一部については、その後のセクションで詳しく説明します。

表12.1 parted コマンド

コマンド	詳細
check <i>minor-num</i>	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 <i>from to</i>	パーティション間で、ファイルシステムをコピーします。 <i>from</i> と <i>to</i> 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 <i>label</i>	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

コマンド	詳細
mkfs <i>minor-num file-system-type</i>	タイプ <i>file-system-type</i> のファイルシステムを作成します。
mkpart <i>part-type fs-type start-mb end-mb</i>	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 <i>part-type fs-type start-mb end-mb</i>	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 <i>minor-num start-mb end-mb</i>	パーティションを移動します
name <i>minor-num name</i>	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 <i>start-mb end-mb</i>	<i>start-mb</i> から <i>end-mb</i> へ、消失したパーティションを復旧します。
resize <i>minor-num start-mb end-mb</i>	<i>start-mb</i> から <i>end-mb</i> へ、パーティションのサイズを変更します
rm <i>minor-num</i>	パーティションを削除します
select <i>device</i>	設定する別のデバイスを選択します
set <i>minor-num flag state</i>	パーティションにフラグを設定します。 <i>state</i> はオンまたはオフのいずれかになります
toggle [<i>NUMBER</i> [<i>FLAG</i>]]	パーティション <i>NUMBER</i> 上の <i>FLAG</i> の状態を切り替えます
unit <i>UNIT</i>	デフォルトのユニットを <i>UNIT</i> 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)、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を示し、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** に値が示されていない場合は、そのファイルシステムが不明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 を再度開始せずに、異なるデバイスを選択するには、**select** コマンドにデバイス名 (例: `/dev/sda`) を付けて実行すると、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

使用中のデバイスに、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. 現在のパーティションテーブルを表示し、十分な空き領域があるかどうか確認します。

```
# print
```

空き領域が十分でない場合は、現在のパーティションのサイズを変更できます。詳細については「[fdisk を使用した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. パーティションの作成

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 (この場合は拡張パーティションは含みません)。パーティションが 5 つ以上必要な場合は、プライマリーパーティションを 3 つ、拡張パーティションを 1 つにし、その拡張パーティションの中に複数の論理パーティションを追加します。ディスクパーティションの概要については、Red Hat Enterprise Linux 7 『インストールガイド』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 メガバイトに ext3 ファイルシステムの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

代わりに **mkpartfs** コマンドを使用すると、パーティションが作成されてからファイルシステムが作成されます。**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。

Enter を押すと変更が反映されるため、押す前に再度確認してください。

パーティションを作成したら、**print** コマンドを実行し、パーティションテーブルにパーティションタ

イプ、ファイルシステムタイプ、およびサイズが正しく設定されてい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` を終了したら `cat /proc/partitions` を実行し、カーネルが新規のパーティションを認識しているを確認します。

`parted` では、最大で 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得られるエリアは、128 個のパーティションに制限されます。

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。

```
# /usr/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現存するすべてのデータが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` への追加

`root` 権限で `/etc/fstab` ファイルを編集し、パーティションの UUID を使用してパーティションを追加します。すべてのパーティションの UUID を一覧を表示するには `blkid -o list` コマンドを使用し、個別のデバイスの詳細を表示するには `blkid device` を使用します。

最初の列では、**UUID=** 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、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) が記載されます。フォーマットの詳細は、`man fstab` コマンドを実行し、`man` ページをご覧ください。

4 つ目の列に **defaults** と指定されていると、パーティションはシステムの起動時にマウントされます。システムを再起動せずにパーティションをマウントするには、`root` 権限で以下のコマンドを実行します。

```
mount /work
```

12.3. パーティションの削除



警告

パーティションが設定されているデバイスが使用中の場合は、削除しないでください。

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確認します。

```
# print
```

4. **rm** コマンドでパーティションを削除します。例えば、マイナー番号 3 のパーティションを削除するのは以下のコマンドです。

```
# rm 3
```

変更は **Enter** を押すと変更が反映されるため、押す前にコマンドを再度確認してください。

5. パーティションを削除したら、**print** コマンドを実行して、そのパーティション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. fdisk を使用したパーティションのサイズ変更

fdisk ユーティリティを使用して、GPT、MBR、Sun、SGI または BSD のパーティションテーブルを作成して操作できます。**fdisk** の GUID パーティションテーブル (GPT) サポートは、実験的フェー

ズにあるため、GPT を使用しているディスクには、**parted** ユーティリティーを使用することが推奨されます。

fdisk を使用してパーティションサイズを変更する場合は、パーティションを削除して再作成するため、パーティションのサイズを変更する前に、ファイルシステムに保存されているデータを**バックアップ**してから、手順をテストします。



重要

サイズを変更するパーティションは、そのディスクの最後のパーティションである必要があります。

Red Hat は、LVM パーティションの拡張およびサイズ変更だけをサポートします。

手順12.4 パーティションのサイズ変更

以下の手順は、参照用としてのみ提供しています。**fdisk** を使用してパーティションのサイズを変更します。

1. デバイスをアンマウントします。

```
~]# umount /dev/vda
```

2. 以下の例のように、**fdisk disk_name** を実行します。

```
~]# fdisk /dev/vda
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

3. **p** オプションで、削除するパーティションのライン番号を確認します。

```
Command (m for help): p
Disk /dev/vda: 16.1 GB, 16106127360 bytes, 31457280 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0006d09a

   Device   Boot      Start         End      Blocks   Id  System
/dev/vda1   *          2048     1026047       512000   83   Linux
/dev/vda2             1026048     31457279     15215616   8e   Linux LVM
```

4. **d** オプションで、パーティションを削除します。パーティションが2つ以上ある場合は、削除するパーティションの番号を指定します。

```
Command (m for help): d
Partition number (1,2, default 2): 2
Partition 2 is deleted
```

5. **n** オプションで新しいパーティションを作成し、指示に従います。今後のサイズ変更に必要な容量を十分に確保するようにしてください。(Enter を押したときの) **fdisk** のデフォルトの動作は、デバイスの領域をすべて使用することです。パーティションの末尾は、セクターで指定するか、人が判読できる値を **+<size><suffix>** の形式 (例: +500M、+10G) で指定できます。

fdisk は、パーティションの末尾を物理セクターに合わせて調整するため、空き領域をすべて使用したくない場合は、人が判読できるサイズ設定を選択することが推奨されます。(セクター単位で) 実際の数値を指定すると、**fdisk** がパーティションの調整を行うことはなくなります。

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): *Enter*
Using default response p
Partition number (2-4, default 2): *Enter*
First sector (1026048-31457279, default 1026048): *Enter*
Using default value 1026048
Last sector, +sectors or +size{K,M,G} (1026048-31457279, default
31457279): +500M
Partition 2 of type Linux and of size 500 MiB is set
```

6. パーティションのタイプを LVM に設定します。

```
Command (m for help): t
Partition number (1,2, default 2): *Enter*
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'
```

7. 変更が正しければ、**w** を押して変更を書き込みます。書き込みにエラーがあると、選択したパーティションが不安定になる可能性があります。
8. デバイスで **e2fsck** を実行し、整合性を確認します。

```
~]# e2fsck /dev/vda
e2fsck 1.41.12 (17-May-2010)
Pass 1:Checking inodes, blocks, and sizes
Pass 2:Checking directory structure
Pass 3:Checking directory connectivity
Pass 4:Checking reference counts
Pass 5:Checking group summary information
ext4-1:11/131072 files (0.0% non-contiguous),27050/524128 blocks
```

9. デバイスをマウントします。

```
~]# mount /dev/vda
```

詳細は、**fdisk(8)** の man ページを参照してください。

第13章 SNAPPER を用いたスナップショットの作成および維持

スナップショットボリュームはターゲットボリュームの指定時のコピーで、ファイルシステムを以前の状態に戻す方法を提供します。Snapper は、Btrfs およびシンプロビジョニング対応 LVM ファイルシステムのスナップショットを作成および維持するコマンドラインツールです。

13.1. 初期の SNAPPER セットアップ

Snapper には、Snapper が操作を行う各ボリュームにそれぞれ設定ファイルが必要です。設定ファイルは手動で設定する必要があります。デフォルトでは、root ユーザーだけが snapper コマンドを実行できます。

Red Hat は、Red Hat Enterprise Linux 7 で Snapper と ext4 ファイルシステムを使用することを推奨します。障害の原因となる容量不足の問題を防ぐため、プールの空き容量を監視する場合のみ lvm-thin ボリュームで XFS ファイルシステムを使用するようにしてください。

Btrfs ツールおよびファイルシステムはテクノロジープレビューとして提供されるため、実稼働システムでの使用には適していないことに注意してください。

root 以外のユーザーまたはグループが特定の Snapper コマンドを実行できるようにすることは可能ですが、特権のないユーザーやグループのパーティションは昇格しないことが推奨されます。特権を昇格すると SELinux をすり抜けることとなり、セキュリティのリスクを伴う可能性があります。**sudo** インフラストラクチャーの使用を検討することを、セキュリティ部門に相談することが推奨されます。

手順13.1 Snapper 設定ファイルの作成

1. 以下のいずれかを作成または選択します。
 - Red Hat がサポートするファイルシステムが構築されたシンプロビジョニング対応の論理ボリューム
 - Btrfs サブボリューム
2. ファイルシステムをマウントします。
3. このボリュームを定義する設定ファイルを作成します。

LVM2 の場合

```
# snapper -c config_name create-config -f "lvm(fs_type)" /mount-point
```

Btrfs の場合

```
# snapper -c config_name create-config -f btrfs /mount-point
```

- **-c config_name** オプションは設定ファイルの名前を指定します。
- **create-config** は、設定ファイルの作成を Snapper に指示します。
- **-f file_system** は、使用するファイルシステムを Snapper に指示します。省略すると、Snapper はファイルシステムの検出を試みます。
- **/mount-point** は、サブボリュームまたはシンプロビジョニング対応 LVM2 ファイルシステムがマウントされた場所を指定します。

たとえば、ext4 ファイルシステムが構築され、`/lvm_mount` にマウントされた LVM2 サブボリュームに `lvm_config` という名前の設定ファイルを作成するには、以下を使用します。

```
# snapper -c lvm_config create-config -f "lvm(ext4)" /lvm_mount
```

`/btrfs_mount` にマウントされた Btrfs サブボリュームに `btrfs_config` という名前の設定ファイルを作成するには、以下を使用します。

```
# snapper -c btrfs_config create-config -f btrfs /btrfs_mount
```

設定ファイルは `/etc/snapper/configs/` ディレクトリーに保存されます。

13.2. SNAPPER スナップショットの作成

Snapper は、以下の種類のスナップショットを作成できます。

プリスナップショット

プリスナップショットは、ポストスナップショットの起点となります。プリスナップショットおよびポストスナップショットは密接に関係し、この間のファイルシステムの変更を追跡します。プリスナップショットはポストスナップショットの前に作成する必要があります。

ポストスナップショット

ポストスナップショットは、プリスナップショットの終点となります。結合されたプリスナップショットおよびポストスナップショットは比較の範囲を定義します。デフォルトでは、関連するポストスナップショットが正常に作成された後、バックグラウンド比較を作成するために新しい Snapper ボリュームが設定されます。

シングルスナップショット

シングルスナップショットは、特定時に作成されるスタンドアロンのスナップショットです。シングルスナップショットは変更のタイムラインを追跡するために使用され、後で戻る点があります。

13.2.1. プリスナップショットおよびポストスナップショットのペアの作成

13.2.1.1. Snapper を用いたプリスナップショットの作成

プリスナップショットを作成するには、以下を実行します。

```
# snapper -c config_name create -t pre
```

`-c config_name` オプションは、名前付き設定ファイルの仕様に従ってスナップショットを作成します。設定ファイルが存在しない場合は「[初期の Snapper セットアップ](#)」を参照してください。

`create -t` オプションは、作成するスナップショットの種類を指定します。許可されるエントリーは `pre`、`post`、または `single` です。

たとえば、「[初期の Snapper セットアップ](#)」で作成された `lvm_config` 設定ファイルを使用してプリスナップショットを作成するには、以下を実行します。

```
# snapper -c SnapperExample create -t pre -p
1
```

-p オプションは、作成されたスナップショットの数を出力する任意のオプションです。

13.2.1.2. Snapper を用いたポストスナップショットの作成

ポストスナップショットは、スナップショットの終点で、[「Snapper を用いたプリスナップショットの作成」](#)の手順に従って親のプリスナップショットを作成してから、作成する必要があります。

手順13.2 ポストスナップショットの作成

1. プリスナップショットの番号を確認します。

```
# snapper -c config_name list
```

たとえば、**lvm_config** 設定ファイルを使用して作成されたスナップショットの一覧を表示するには、以下を実行します。

```
# snapper -c lvm_config list
Type      | # | Pre # | Date                | User | Cleanup |
Description | Userdata
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
single | 0 |      |                      | root |          | current
|
pre    | 1 |      | Mon 06<...>        | root |          |
|
```

上記の出力は、プリスナップショットの番号が 1 であることを表しています。

2. 以前作成されたプリスナップショットにリンクされるポストスナップショットを作成します。

```
# snapper -c config_file create -t post --pre-num
pre_snapshot_number
```

- **-t post** オプションは、ポストスナップショットの作成を指定します。
- **--pre-num** オプションは、対応するプリスナップショットを指定します。

たとえば、**lvm_config** 設定ファイルを使用してポストスナップショットを作成し、番号が 1 のプリスナップショットにリンクするには、以下を実行します。

```
# snapper -c lvm_config create -t post --pre-num 1 -p
2
```

-p オプションは、作成されたスナップショットの数を出力する任意のオプションです。

3. 番号が 2 のポストスナップショットが作成され、番号が 1 のプリスナップショットとペアになります。**list** コマンドで確認します。

```
# snapper -c lvm_config list
Type      | # | Pre # | Date                | User | Cleanup |
Description | Userdata
-----+-----+-----+-----+-----+-----+-----+
-----+-----+
single | 0 |      |                      | root |          | current
```

```
|
| pre      | 1 |          | Mon 06<...>          | root |          |
|
| post     | 2 | 1        | Mon 06<...>          | root |          |
|
```

13.2.1.3. プリまたはポストスナップショットでのコマンドのラッピング

プリスナップショットおよびポストスナップショット内でコマンドをラッピングすることも可能です。これはテスト時に便利です。簡単な手順は、[手順13.3「プリまたはポストスナップショットでのコマンドのラッピング」](#)を参照してください。

1. **snapper create pre snapshot** コマンドを実行します。
2. ファイルシステムの内容に影響する可能性があるアクションを実行するため、1 つまたは複数のコマンドを実行します。
3. **snapper create post snapshot** コマンドを実行します。

手順13.3 プリまたはポストスナップショットでのコマンドのラッピング

1. プリスナップショットおよびポストスナップショットでコマンドをラッピングするには、以下を行います。

```
# snapper -c lvm_config create --command "command_to_be_tracked"
```

たとえば、**/lvm_mount/hello_file** ファイルの作成を追跡するには、以下を実行します。

```
# snapper -c lvm_config create --command "echo Hello >
/lvm_mount/hello_file"
```

2. これを確認するには、**status** コマンドを実行します。

```
# snapper -c config_file status
first_snapshot_number..second_snapshot_number
```

たとえば、最初のステップで変更した内容を追跡するには、以下を実行します。

```
# snapper -c lvm_config status 3..4
+..... /lvm_mount/hello_file
```

必要な場合は **list** コマンドを実行し、スナップショットの番号を確認します。

status コマンドの詳細は、「[Snapper スナップショット間の変更の追跡](#)」を参照してください。

スナップショットが上記の例のコマンドのみをキャプチャーするとは限らないことに注意してください。Snapper はユーザーによる変更だけでなく、システムによる変更もすべて記録します。

13.2.2. シングル Snapper スナップショットの作成

シングル Snapper スナップショットの作成は、プリスナップショットまたはポストスナップショットの作成と似ており、**create -t** オプションのみがシングルスナップショットを指定します。シングルス

ナップショットは、他のスナップショットと関連せずに単一のスナップショットを作成するために使用されます。しかし、自動的に比較を生成したり、追加情報をリストする必要がなく、LVM2 シンボリックボリュームのスナップショットを作成する明確な方法が必要な場合は、Snapper の代わりに System Storage Manager を使用することが推奨されます (「[スナップショット](#)」 参照)。

シングルスナップショットを作成するには、以下を実行します。

```
# snapper -c config_name create -t single
```

たとえば、以下のコマンドは **lvm_config** 設定ファイルを使用してシングルスナップショットを作成します。

```
# snapper -c lvm_config create -t single
```

シングルスナップショットは、変更の追跡用に特別に設計されたものではありませんが、**snapper diff** コマンド、**xadiff** コマンド、および **status** コマンドを使用すると、2つのスナップショットを比較できます。各コマンドの詳細は「[Snapper スナップショット間の変更の追跡](#)」を参照してください。

13.2.3. スナップショットを自動的に取得するよう Snapper を設定

スナップショットの自動取得は、Snapper の主な機能の 1 つです。デフォルトでは、ボリュームに対して Snapper を設定すると、Snapper はそのボリュームのスナップショットを 1 時間ごとに取得します。

デフォルトの設定では、Snapper は以下の周期でスナップショットを取得します。

- 10 時間周期のスナップショット。最後のスナップショットは Daily スナップショットとして保存されます。
- 10 日周期のスナップショット。月最後のスナップショットは Monthly スナップショットとして保存されます。
- 10 カ月周期のスナップショット。最後のスナップショットは Yearly スナップショットとして保存されます。
- 10 年周期のスナップショット。

Snapper はデフォルトで 50 個以下のスナップショットを保持しますが、デフォルトでは作成後 1,800 秒未満のスナップショットはすべて保持されます。

デフォルト設定は、**/etc/snapper/config-templates/default** ファイルに指定されます。**snapper create-config** コマンドを実行して設定を作成すると、未指定の値はすべてデフォルト設定を基に設定されます。定義されたボリュームの設定は、**/etc/snapper/configs/config_name** ファイルで編集できます。

13.3. SNAPPER スナップショット間の変更の追跡

status、**diff**、および **xadiff** コマンドを使用して、スナップショット間におけるサブボリュームの変更を追跡します。

status

status コマンドは、2つのスナップショットの間で作成、編集、または削除されたファイルおよびディレクトリを一覧表示します (2つのスナップショットの間で発生した変更の包括的なリス

ト)。システム管理者は、このコマンドを使用して、過剰な情報が含まれない、適度な変更の概要を取得できます。

詳細は「[status コマンドによる変更の比較](#)」を参照してください。

diff

diff コマンドは、1 つ以上の変更が検出された場合に **status** コマンドが出力する 2 つのスナップショットの間に変更されたファイルおよびディレクトリーの差異を表示します。

詳細は「[diff コマンドによる変更の比較](#)」を参照してください。

xadiff

xadiff コマンドは、2 つのスナップショットの間にファイルまたはディレクトリーの拡張属性がどのように変更されたかを比較します。

詳細は「[xadiff コマンドによる変更の比較](#)」を参照してください。

13.3.1. status コマンドによる変更の比較

status コマンドは、2 つのスナップショットの間で作成、編集、または削除されたファイルおよびディレクトリーを表示します。

2 つのスナップショット間のファイル状態を表示するには、以下を実行します。

```
# snapper -c config_file status
first_snapshot_number..second_snapshot_number
```

必要な場合は **list** コマンドを実行し、スナップショットの番号を確認します。

たとえば、以下のコマンドは **lvm_config** 設定ファイルを使用して、スナップショット 1 と 2 との間に加えられた変更を表示します。

```
snapper -c lvm_config status 1..2
tp.... /lvm_mount/dir1
-..... /lvm_mount/dir1/file_a
c.ug.. /lvm_mount/file2
+..... /lvm_mount/file3
....x. /lvm_mount/file4
cp..xa /lvm_mount/file5
```

出力の最初の部分にある文字と点 (ピリオド) は列とします。

```
+..... /lvm_mount/file3
|||||
123456
```

列 1 はファイル (ディレクトリーエントリー) タイプの変更を示します。可能な値は以下のとおりです。

列 1

出力	意味
.	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 は、ファイル権限の変更を示しています。使用できる値は次のとおりです。

列 2

出力	意味
.	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 は、ユーザー所有権の変更を示しています。使用できる値は以下のとおりです。

列 3

出力	意味
.	いずれのユーザー所有権も変更されていません。
u	ユーザーの所有権が変更されました。

列 4 は、グループ所有権の変更を示しています。使用できる値は以下のとおりです。

列 4

出力	意味
.	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 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とおりです。

列 5

出力	意味
.	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 は、アクセス制御リスト (ACL) の変更を示しています。使用できる値は以下のとおりです。

列 6

出力	意味
.	ACL は変更されませんでした。
a	ACL は変更されました。

13.3.2. diff コマンドによる変更の比較

diff コマンドは、2 つのスナップショットの間に編集されたファイルおよびディレクトリーの変更箇所を表示します。

```
# snapper -c config_name diff
first_snapshot_number..second_snapshot_number
```

必要な場合は **list** コマンドを実行し、スナップショットの番号を確認します。

たとえば、**lvm_config** 設定ファイルを使用してスナップショット 1 と 2 の間で変更されたファイルを比較するには、以下を使用します。

```
# snapper -c lvm_config diff 1..2
--- /lvm_mount/.snapshots/13/snapshot/file4 19<...>
+++ /lvm_mount/.snapshots/14/snapshot/file4 20<...>
@@ -0,0 +1 @@
+words
```

上記の出力は、**file4** に「words」をファイルに追加して編集されたことを示しています。

13.3.3. xadiff コマンドによる変更の比較

xadiff コマンドは、2 つのスナップショットの間にファイルまたはディレクトリーの拡張属性がどのように変更されたかを比較します。

```
# snapper -c config_name xadiff
first_snapshot_number..second_snapshot_number
```

必要な場合は **list** コマンドを実行し、スナップショットの番号を確認します。

たとえば、**lvm_config** 設定ファイルを使用してスナップショット 1 と 2 との間の xadiff の出力を表示するには、以下を実行します。

```
# snapper -c lvm_config xadiff 1..2
```

13.4. スナップショット間の変更の取り消し

既存の 2 つの Snapper スナップショット間の変更を元に戻すには、**undochange** コマンドを、以下の形式で実行します。**1** は最初のスナップショットで、**2** は 2 つ目のスナップショットになります。

```
snapper -c config_name undochange 1..2
```

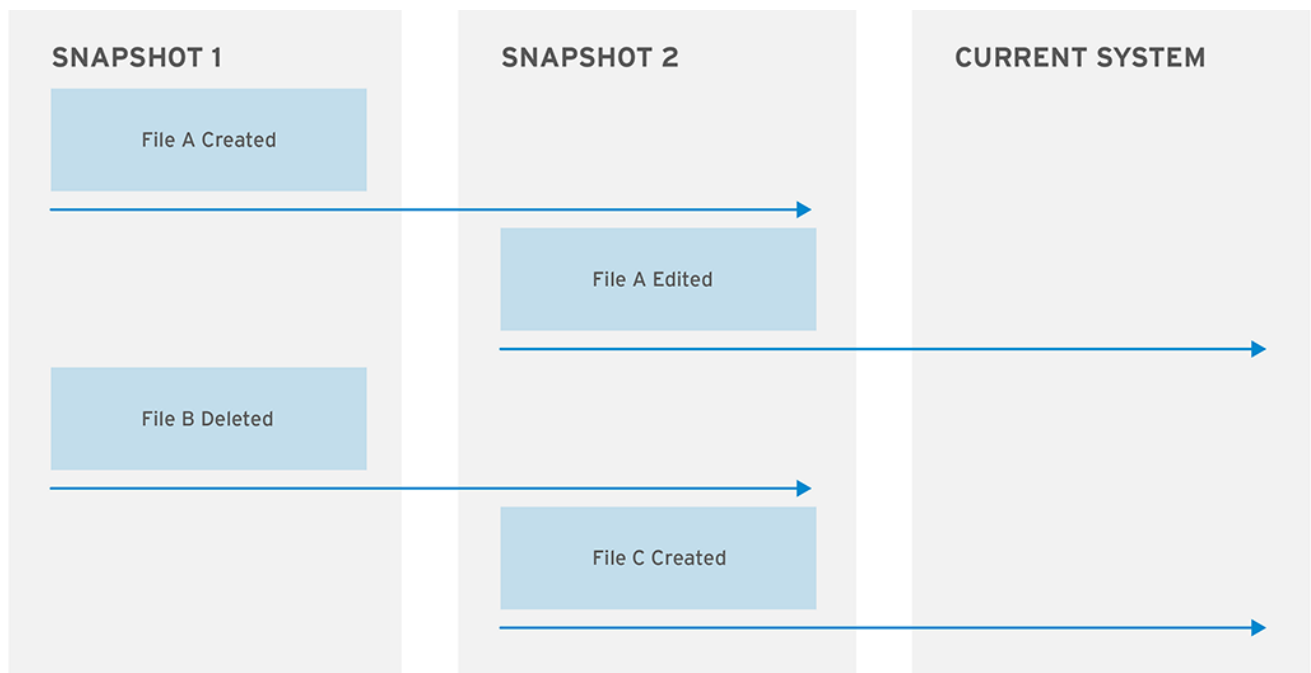
重要

undochange コマンドを使用しても、Snapper ボリュームは元の状態に戻らず、データの整合性は維持されません。指定範囲外で行われるファイルの編集 (例: スナップショット 2 以降) は、スナップショット 1 の状態に戻されても変更しません。たとえば、**undochange** を実行してユーザーが作成したものを元に戻す場合、そのユーザーが所有するファイルはそのままの状態を維持できます。

また、スナップショットの作成時にファイルの整合性を保証するメカニズムはないため、**undochange** コマンドを使用すると、すでに存在する不整合がスナップショットに戻される可能性があります。

undochange コマンドは失敗する可能性が高いため、ルートファイルシステムには Snapper の **undochange** コマンドを使用しないでください。

以下の図は、**undochange** コマンドの仕組みを表しています。



RHEL_387527_0125

図13.1 時間経過後の Snapper の状態

上記の図では、**snapshot_1** が作成されてから、**file_a** が作成され、**file_b** が削除されています。

その後 **Snapshot_2** が作成されてから、**file_a** が編集され、**file_c** が作成されています。これがシステムの現在の状態です。現在のシステムには編集後の **file_a** があり、**file_b** はなく、新規作成された **file_c** があります。

undochange コマンドが呼び出されると、最初のスナップショットと 2 つ目のスナップショットの間に編集されたファイルのリストが生成されます。この図の状態では、**snapper -c SnapperExample undochange 1..2** コマンドを実行すると、編集されたファイルのリスト (**file_a** が作成され、**file_b** が削除されてます) が作成され、現在のシステムに適用されます。したがって、以下のようになります。

- 現在のシステムには **file_a** がありません。**snapshot_1** が作成されてから作成されたからです。
- **file_b** は、**snapshot_1** から、現在のシステムにコピーされます。
- 指定期間外に作成されたため、**file_c** は存在します。

file_b と **file_c** が競合すると、システムが破損する可能性があります。

また、**snapper -c SnapperExample undochange 2..1** コマンドを使用することも可能です。この場合、現在のシステムにある **file_a** が、**snapshot_1** のものに置き換わるため、**snapshot_2** を作成した後に加えられた変更内容は元に戻ります。

mount および **unmount** コマンドを使用した変更の取り消し

undochange コマンドが、変更を元に戻す最適な方法ではないことがあります。**status** コマンドと **diff** コマンドを併用して、適切な判断を行ってください。さらには、Snapper ではなく、**mount** コマンドと **unmount** コマンドを使用します。**mount** コマンドと **unmount** コマンドは、Snapper ワークフロー外のコンテンツを閲覧する場合に限り便利です。

必要に応じて、**mount** コマンドで、マウント前に対応する LVM Snapper スナップショットをアクティベートします。スナップショットをマウントし、古いバージョンのファイルを複数抽出する場合は、**mount** コマンドおよび **unmount** コマンドを使用します。ファイルを元に戻す場合は、マウントしているスナップショットから現在のファイルシステムにコピーします。現在のファイルシステムであるスナップショット 0 は、[手順13.1「Snapper 設定ファイルの作成」](#) から作成したライブファイルシステムです。このファイルを、元の /mount-point のサブツリーにコピーします。

明示的なクライアント側のリクエストには、**mount** コマンドおよび **unmount** コマンドを使用します。**/etc/snapper/configs/config_name** ファイルには、ユーザーおよびグループを追加できる **ALLOW_USERS=** 変数および **ALLOW_GROUPS=** 変数が含まれます。**snapperd** を使用すると、追加されたユーザーおよびグループのマウント操作を実行できます。

13.5. SNAPPER スナップショットの削除

スナップショットの削除手順

```
# snapper -c config_name delete snapshot_number
```

list コマンドを実行して、スナップショットが正常に削除されたことを確認できます。

第14章 SWAP領域

Linux の *swap* 領域は、物理メモリー (RAM) の空き容量がなくなると使用されます。システムがより多くのメモリーリソースを必要とする場合、RAM に空き容量がないとメモリーの非アクティブなページが *swap* 領域に移されます。RAM の容量が小さいマシンでは *swap* 領域が役に立ちますが、容量の大きい RAM の代わりとしてはなりません。*swap* 領域は、物理メモリーよりもアクセスに時間がかかるハードドライブ上にあります。*swap* 領域は、専用のスワップパーティション (推奨)、スワップファイル、またはスワップパーティションとスワップファイルの組み合わせにすることができます。**Btrfs** は *swap* 領域をサポートしないため注意してください。

過去数年、推奨される *swap* 領域のサイズは、システムの RAM サイズとともに直線的に増加していました。しかし、最近のシステムのメモリーサイズは数百ギガバイトまで増加したため、現在推奨される *swap* 領域の容量は、システムメモリーではなくシステムメモリーのワークロードの関数であると考えられています。

表14.1「システムの推奨 *swap* 領域」は、ご使用のシステムの RAM 容量別に、システムがハイバネートするために十分なメモリーが不要な場合と必要な場合のスワップパーティションの推奨サイズをまとめています。推奨の *swap* 領域はインストール中に自動的に確定されますが、ハイバネーションも可能にするには、カスタムパーティション分割の段階で *swap* 領域の編集が必要となります。

表14.1「システムの推奨 *swap* 領域」の推奨情報は、メモリーの容量が小さいシステム (1 GB 以下) で特に重要になります。システムで十分な *swap* 領域を割り当てできないと、安定性の問題や、インストールされたシステムが起動できない問題などが発生する原因となることがあります。

表14.1 システムの推奨 *swap* 領域

システム内の RAM の容量	推奨される <i>swap</i> 領域	ハイバネートを許可する場合の推奨 <i>swap</i> 領域
≤ 2 GB	RAM 容量の 2 倍	RAM 容量の 3 倍
> 2 GB – 8 GB	RAM 容量と同じ	RAM 容量の 2 倍
> 8 GB – 64 GB	最低 4GB	RAM 容量の 1.5 倍
> 64 GB	最低 4GB	ハイバネートは推奨しません

ご使用のシステムが 表14.1「システムの推奨 *swap* 領域」に記載されている境界線上 (RAM が 2GB、8GB または 64GB) になる場合、*swap* サイズやハイバネートへの対応を決める際は慎重に行なってください。システムリソースに余裕がある場合は、*swap* 領域を大きくするとパフォーマンスが向上することがあります。140 個を超える論理プロセッサまたは 3 TB を超える RAM があるシステムには、最低でも 100 GB の *swap* 領域を確保することが推奨されます。

高速のドライブ、コントローラー、およびインターフェイスを搭載したシステムでは、複数のストレージデバイスに *swap* 領域を分散すると、*swap* 領域のパフォーマンスも向上します。



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変更してはいけません。システムプロセスやカーネルが swap 領域を使用しているときに swap を変更しようとしても変更に失敗します。swap の使用量とその使用場所を確認するには、**free** および **cat /proc/swaps** コマンドを使用します。

システムが **rescue** モードで起動している間に swap 領域を変更してください。詳細は『Red Hat Enterprise Linux 7 インストールガイド』の [レスキューモードでコンピューターを起動する](#) を参照してください。ファイルシステムのマウントを要求されたら **Skip** を選択してください。

14.1. SWAP 領域の追加

場合によっては、インストールした後にさらに 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で 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。

14.1.1. LVM2 論理ボリュームでの swap 領域の拡張

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ますようになります。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14.1 LVM2 論理ボリュームでの swap 領域の拡張

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。

14.1.2. swap の LVM2 論理ボリュームの作成

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ています)。

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。

14.1.3. swap ファイルを作成する

swap ファイルを追加します。

手順14.2 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 ファイルのセキュリティーを変更し、あらゆるユーザーが読み取りできないようにします。

```
# chmod 0600 /swapfile
```

5. 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
```

6. 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。

14.2. SWAP 領域の削除

時には、インストールの後に 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 のままだ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。

14.2.1. LVM2 論理ボリュームでの **swap** 領域の縮小

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て)。

手順14.3 LVM2 の swap 論理ボリュームの縮小

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。

14.2.2. swap の LVM2 論理ボリュームの削除

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て)。

手順14.4 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。

14.2.3. swap ファイルの削除

swap ファイルを削除します。

手順14.5 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** であるとして)。

```
# 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
```

14.3. 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15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。ストレージシステムは、Device Mappers (DM)、論理ボリュームマネージャー (LVM)、Multiple Devices (MD) の使用によってますます複雑になっています。これにより、システムはユーザーが使いにくいものとなり、エラーは問題が生じやすくなります。SSM は統一されたユーザーインターフェースを作成してこの問題を軽減します。このインターフェースによって、ユーザーは複雑なシステムを簡単に実行できるようになります。たとえば、SSM を使用せずに新しいファイルシステムを作成およびマウントするには、5 つのコマンドを使用する必要がありますが、SSM を使用すると 1 つのコマンドのみが必要になります。

本章では、SSM がさまざまなバックエンドと対話する仕組みを説明した後、一般的なユースケースを詳細に説明します。

15.1. SSM のバックエンド

SSM は、デバイス、プールおよびボリュームの抽象化に準拠する `ssmlib/main.py` のコア抽象化レイヤーを使用し、基盤のテクノロジーの詳細を無視します。ボリュームおよびプールの `create`、`snapshot`、または `remove` などの特定のストレージ技術を処理するため、バックエンドを `ssmlib/main.py` で登録できます。

SSM にはすでに複数のバックエンドが登録されています。以下のセクションでは、これらのバックエンドの基本情報を説明し、プール、ボリューム、スナップショット、およびデバイスの処理方法の定義についても説明します。

15.1.1. Btrfs バックエンド

BTRFS は多くの高度な機能を持つファイルシステムで、SSM ではボリューム管理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 バックエンドで作成できます。

15.1.1.1. BTRFS プール

BTRFS ファイルシステム自体はプールです。デバイスを追加して拡張したり、デバイスを削除して縮小することができます。SSM は BTRFS プールの作成時に BTRFS ファイルシステムを作成します。新規作成された各 BTRFS プールにはプールと同じ名前のボリュームが 1 つあり、プール全体を削除しないと削除できません。BTRFS プールのデフォルトの名前は `btrfs_pool` です。

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のない BTRFS ファイルシステムがすでに存在する場合は、BTRFS プールは、`btrfs_device_base_name` という形式で、内部使用向けの名前を生成します。

15.1.1.2. Btrfs ボリューム

プールの最初のボリュームの後に作成されたボリュームはサブボリュームと同じです。サブボリュームを作成するために BTRFS ファイルシステムがアンマウントされた場合、SSM は BTRFS ファイルシステムを一時的にマウントします。

ボリュームの名前は、BTRFS ファイルシステムのサブボリュームのパスとして使用されます。たとえば、サブボリュームは `/dev/lvm_pool/lvol001` のように表示されます。ボリュームの作成には、このパスのすべてのオブジェクトが存在する必要があります。ボリュームはこのマウントポイントで参照することもできます。

15.1.1.3. BTRFS スナップショット

SSM では、システムのすべての BTRFS ボリュームからスナップショットを作成できます。BTRFS はサブボリュームとスナップショットを区別しないことに注意してください。そのため、SSM は BTRFS スナップショットの宛先を認識できませんが、特別な名前の形式を認識しようとします。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。

15.1.1.4. BTRFS 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。

15.1.2. 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。

15.1.2.1. LVM 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。

15.1.2.2. LVM ボリューム

LVM ボリュームは通常の論理ボリュームと同じです。

15.1.2.3. LVM スナップショット

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。

15.1.2.4. LVM デバイス

SSM では、ユーザーに対して透過的な物理デバイス上に LVM バックエンドを作成する必要があります。

15.1.3. Crypt

SSM における Crypt バックエンドは、**cryptsetup** および **dm-crypt target** 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。

次のセクションは、crypt の観点からボリュームおよびスナップショットを定義します。

15.1.3.1. crypt ボリューム

crypt ボリュームは **dm-crypt** によって作成され、元の暗号化されたデバイス上のデータを暗号化されていない状態で表します。RAID またはデバイスの連結はサポートしません。

luks と plain の2つのモード (拡張) がサポートされます。デフォルトでは luks が使用されます。拡張の詳細は、**man cryptsetup** の出力を参照してください。

15.1.3.2. crypt 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。

15.1.4. 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。

15.2. 一般的な SSM タスク

次のセクションでは、基本的なユースケースを取り上げ、SSM のインストール方法や検出されたすべてのデバイス、プール、およびボリュームに関する情報の表示方法を説明します。次に、2つのボリュームと1つの XFS ファイルシステムでプールを作成します。ファイルシステムの一貫性をチェックし、ボリュームのサイズを増やした後、スナップショットを作成します。最後にボリュームの1つを削除します。

15.2.1. 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** 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 **device-mapper** および **cryptsetup** パッケージが必要です。

15.2.2. 検出された全デバイスの情報表示

list コマンドを使用すると、検出されたすべてのデバイス、プール、およびスナップショットの情報を表示できます。オプションを指定せずに **ssm list** を実行すると以下の出力が表示されます。

```
~]# ssm list
-----
Device          Free      Used      Total  Pool  Mount point
-----
/dev/sda                2.00 GB          PARTITIONED
/dev/sda1             47.83 MB          /test
/dev/vda               15.00 GB          PARTITIONED
/dev/vda1             500.00 MB          /boot
/dev/vda2  0.00 KB  14.51 GB  14.51 GB  rhel
-----
Pool  Type  Devices      Free      Used      Total
```

```

-----
rhel  lvm  1          0.00 KB  14.51 GB  14.51 GB
-----
-----
Volume          Pool  Volume size  FS      FS size      Free  Type
Mount point
-----
-----
/dev/rhel/root   rhel      13.53 GB  xfs     13.52 GB      9.64 GB  linear /
/dev/rhel/swap   rhel      1000.00 MB                linear
/dev/sda1        47.83 MB  xfs     44.50 MB      44.41 MB  part
/test
/dev/vda1        500.00 MB  xfs     496.67 MB     403.56 MB  part
/boot
-----
-----

```

表示内容を絞り込むには、引数を使用して表示する項目を指定します。**ssm list --help** コマンドを実行すると、使用できるオプションのリストが表示されます。

注記

指定した引数によっては、すべての情報が表示されないことがあります。

- **devices** または **dev** 引数を実行すると一部のデバイスが省略されます。CD ROM および DM/MD デバイスなどはボリュームとしてリストされ、意図的に表示されません。
- 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できません。このようなバックエンドに **snapshot** 引数を使用すると、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。
- メインの BTRFS ボリューム (ファイルシステム自体) 以外のアンマウントされた BTRFS ボリュームは表示されません。

15.2.3. 新規のプール、論理ボリューム、およびファイルシステムの作成

このセクションでは、デフォルトの名前で新しいプールを作成します。このプールには **/dev/vdb** および **/dev/vdc** デバイス、1 G の論理ボリューム、および XFS ファイルシステムが含まれます。

このようなプールを作成するには、**ssm create --fs xfs -s 1G /dev/vdb /dev/vdc** コマンドを使用します。以下のオプションが使用されます。

- **--fs** オプションは必要なファイルシステムの種類を指定します。現在サポートされるファイルシステムの種類は次のとおりです。
 - ext3
 - ext4
 - xfs

- btrfs
- **-s** は論理ボリュームのサイズを指定します。以下の接尾辞がユニットの定義でサポートされます。
 - 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**
- **/dev/vdb** および **/dev/vdc** の2つのデバイスを作成します。

```
~]# ssm create --fs xfs -s 1G /dev/vdb /dev/vdc
Physical volume "/dev/vdb" successfully created
Physical volume "/dev/vdc" successfully created
Volume group "lvm_pool" successfully created
Logical volume "lvol001" created
```

他にも **ssm command** には便利なコマンドが2つあります。その1つが **-p pool** コマンドです。このコマンドは、ボリュームが作成されるプールを指定します。プールが存在しない場合は、SSM によって作成されます。このコマンドは上記の例では省略されているため、SSM によってデフォルトの名前 **lvm_pool** が使用されます。しかし、既存の命名規則に合った特定の名前を使用する場合は **-p** オプションを使用します。

2つ目の便利なオプションは **-n name** コマンドです。このコマンドは新規作成された論理ボリュームに名前を付けます。**-p** と同様に、既存の命名規則に合った特定の名前を使用するために必要になります。

これらの2つのオプションを使用した例を以下に示します。

```
~]# ssm create --fs xfs -p new_pool -n XFS_Volume /dev/vdd
Volume group "new_pool" successfully created
Logical volume "XFS_Volume" created
```

1つのコマンドのみを使用して、2つの物理ボリューム、1つのプール、および1つの論理ボリュームが作成されます。

15.2.4. ファイルシステムの一整合性確認

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。

ボリューム **lvol001** のすべてのデバイスをチェックするには、コマンド **ssm check /dev/lvm_pool/lvol001** をチェックします。

```
~]# ssm check /dev/lvm_pool/lvol001
Checking xfs file system on '/dev/mapper/lvm_pool-lvol001'.
```



```

Phase 1 - find and verify superblock...
Phase 2 - using internal log
          - scan filesystem freespace and inode maps...
          - found root inode chunk
Phase 3 - for each AG...
          - scan (but don't clear) agi unlinked lists...
          - process known inodes and perform inode discovery...
          - agno = 0
          - agno = 1
          - agno = 2
          - agno = 3
          - agno = 4
          - agno = 5
          - agno = 6
          - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
          - setting up duplicate extent list...
          - check for inodes claiming duplicate blocks...
          - agno = 0
          - agno = 1
          - agno = 2
          - agno = 3
          - agno = 4
          - agno = 5
          - agno = 6
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
          - traversing filesystem ...
          - traversal finished ...
          - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.

```

15.2.5. ボリュームの容量追加

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、ボリュームの容量のみが変更されます。

この例では、現在 900MB の **lv1001** という論理ボリュームが 1 つ **/dev/vdb** にあります。

```

~]# ssm list
-----
Device                Free          Used          Total  Pool      Mount point
-----
/dev/vda                15.00 GB
/dev/vda1              500.00 MB      /boot
/dev/vda2    0.00 KB   14.51 GB   14.51 GB   rhel
/dev/vdb   120.00 MB  900.00 MB    1.00 GB   lvm_pool
/dev/vdc                1.00 GB
-----

Pool    Type  Devices      Free          Used          Total
-----
lvm_pool lvm    1          120.00 MB   900.00 MB  1020.00 MB
rhel     lvm    1           0.00 KB   14.51 GB   14.51 GB

```

```
-----
```

Volume	Pool	Volume size	FS	FS size	Free
Type	Mount point				

/dev/rhel/root	rhel	13.53 GB	xf	13.52 GB	9.64 GB
linear	/				
/dev/rhel/swap	rhel	1000.00 MB			
linear					
/dev/lvm_pool/lvol001	lvm_pool	900.00 MB	xf	896.67 MB	896.54 MB
linear					
/dev/vda1		500.00 MB	xf	496.67 MB	403.56 MB
part	/boot				

論理ボリュームをさらに 500 MB 増やす必要があります。これには、デバイスをプールに追加する必要があります。

```
~]# ssm resize -s +500M /dev/lvm_pool/lvol001 /dev/vdc
Physical volume "/dev/vdc" successfully created
Volume group "lvm_pool" successfully extended
Phase 1 - find and verify superblock...
Phase 2 - using internal log
          - scan filesystem freespace and inode maps...
          - found root inode chunk
Phase 3 - for each AG...
          - scan (but don't clear) agi unlinked lists...
          - process known inodes and perform inode discovery...
          - agno = 0
          - agno = 1
          - agno = 2
          - agno = 3
          - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
          - setting up duplicate extent list...
          - check for inodes claiming duplicate blocks...
          - agno = 0
          - agno = 1
          - agno = 2
          - agno = 3
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
          - traversing filesystem ...
          - traversal finished ...
          - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
Extending logical volume lvol001 to 1.37 GiB
Logical volume lvol001 successfully resized
meta-data=/dev/mapper/lvm_pool-lvol001 isize=256    agcount=4,
agsize=57600 blks
          =                               sectsz=512   attr=2, projid32bit=1
          =                               crc=0
```

```

data      =                               bsize=4096   blocks=230400, imaxpct=25
          =                               sunit=0       swidth=0 blks
naming    =version 2                     bsize=4096   ascii-ci=0 ftype=0
log       =internal                      bsize=4096   blocks=853, version=2
          =                               sectsz=512    sunit=0 blks, lazy-count=1
realtime  =none                          extsz=4096   blocks=0, rtextents=0
data blocks changed from 230400 to 358400

```

SSM はデバイス上でチェックを実行した後、指定した容量分ボリュームを拡張します。これは、**ssm list** コマンドで確認できます。

```
~]# ssm list
```

```

-----
Device                Free                Used                Total  Pool                Mount point
-----
/dev/vda              15.00 GB                PARTITIONED
/dev/vda1             500.00 MB                /boot
/dev/vda2      0.00 KB      14.51 GB      14.51 GB  rhel
/dev/vdb      0.00 KB      1020.00 MB      1.00 GB  lvm_pool
/dev/vdc      640.00 MB      380.00 MB      1.00 GB  lvm_pool
-----

Pool      Type  Devices      Free                Used                Total
-----
lvm_pool  lvm    2            640.00 MB      1.37 GB      1.99 GB
rhel      lvm    1            0.00 KB      14.51 GB      14.51 GB
-----

Volume                Pool      Volume size  FS      FS size
Free  Type      Mount point
-----
/dev/rhel/root                rhel      13.53 GB  xfs      13.52 GB      9.64 GB
linear /
/dev/rhel/swap                rhel      1000.00 MB
linear
/dev/lvm_pool/lvol001  lvm_pool      1.37 GB  xfs      1.36 GB      1.36 GB
linear
/dev/vda1                500.00 MB  xfs      496.67 MB      403.56 MB
part  /boot
-----
-----

```

注記

LVM ボリュームのみ容量を減らすことが可能です。他のボリュームタイプではサポートされません。容量を減らすには、+ の代わりに - を使用します。たとえば、LVM ボリュームの容量を 50M 減らすには、以下のコマンドを使用します。

```
~]# ssm resize -s-50M /dev/lvm_pool/lvol002
Rounding size to boundary between physical extents: 972.00 MiB
WARNING: Reducing active logical volume to 972.00 MiB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce lvol002? [y/n]: y
Reducing logical volume lvol002 to 972.00 MiB
Logical volume lvol002 successfully resized
```

+ または - を指定しないと、値は絶対値として認識されます。

15.2.6. スナップショット

既存ボリュームのスナップショットを作成するには、**ssm snapshot** コマンドを使用します。

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。

lvol001 のスナップショットを作成するには、以下のコマンドを使用します。

```
~]# ssm snapshot /dev/lvm_pool/lvol001
Logical volume "snap20150519T130900" created
```

ssm list を使用して、追加されたスナップショットセクションを確認します。

```
~]# ssm list
```

Device	Free	Used	Total	Pool	Mount point
/dev/vda			15.00 GB		PARTITIONED
/dev/vda1			500.00 MB		/boot
/dev/vda2	0.00 KB	14.51 GB	14.51 GB	rhel	
/dev/vdb	0.00 KB	1020.00 MB	1.00 GB	lvm_pool	
/dev/vdc			1.00 GB		

Pool	Type	Devices	Free	Used	Total
lvm_pool	lvm	1	0.00 KB	1020.00 MB	1020.00 MB
rhel	lvm	1	0.00 KB	14.51 GB	14.51 GB

Volume	Free	Type	Mount point	Pool	Volume size	FS	FS size
--------	------	------	-------------	------	-------------	----	---------

```

-----
/dev/rhel/root          rhel          13.53 GB  xfs      13.52 GB      9.64 GB
linear /
/dev/rhel/swap          rhel          1000.00 MB
linear
/dev/lvm_pool/lvol001   lvm_pool      900.00 MB  xfs      896.67 MB      896.54 MB
linear
/dev/vda1               500.00 MB  xfs      496.67 MB      403.56 MB
part    /boot
-----
-----
Snapshot                Origin   Pool      Volume size
Size  Type
-----
/dev/lvm_pool/snap20150519T130900  lvol001  lvm_pool    120.00 MB  0.00 KB
linear
-----
-----

```

15.2.7. 項目の削除

デバイス、プール、またはボリュームを削除するには **ssm remove** を使用します。



注記

削除時にデバイスがプールによって使用されていると、コマンドの実行に失敗します。コマンドの実行を強制するには、**-f** 引数を使用します。

削除時にボリュームがマウントされていると、コマンドの実行に失敗します。これは、**-f** 引数を使用して強制できません。

lvm_pool およびその内容をすべて削除するには、以下のコマンドを実行します。

```

~]# ssm remove lvm_pool
Do you really want to remove volume group "lvm_pool" containing 2 logical
volumes? [y/n]: y
Do you really want to remove active logical volume snap20150519T130900?
[y/n]: y
Logical volume "snap20150519T130900" successfully removed
Do you really want to remove active logical volume lvol001? [y/n]: y
Logical volume "lvol001" successfully removed
Volume group "lvm_pool" successfully removed

```

15.3. SSM のリソース

SSM の詳細は以下を参照してください。

- **man ssm** ページは説明や例を提供し、本書には記載されていないすべてのコマンドおよびオプションの詳細を提供します。

- SSM のローカルドキュメントは **doc/** ディレクトリーにあります。
- 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、開発者が情報交換を行う場です。現在、ユーザーのメーリングリストはないため、質問がある場合はこのメーリングリストに投稿してください。

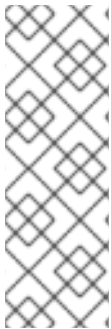
第16章 ディスク割り当て

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。

16.1. 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。

16.1.1. 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16.1 /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16.2 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。

16.1.2. 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も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。

16.1.3. 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16.3 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。

16.1.4. ユーザーごとのクォータ割り当て

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16.4 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
```

ユーザーのクォータが設定されていることを確認するには、以下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
```

16.1.5. グループごとのクォータ割り当て

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
```

このコマンドにより、グループの既存クォータがテキストエディターに表示されます。

```
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
```

16.1.6. 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。

16.2. 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には、若干の保守が必要となります — 大半は、クォータの超過監視および精度確認という形となります。

当然ながら、ユーザーが繰り返しクォータを超過したり、常にソフトリミットに達している場合には、ユーザーのタイプや、ユーザーの作業にディスク容量が及ぼす影響の度合に応じて、システム管理者には2つの選択肢があります。管理者は、ユーザーが使用するディスク領域を節約する方法をわかるようにするか、ユーザーのディスククォータを拡大するかのいずれかを行うことができます。

16.2.1. 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。

16.2.2. 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** ユーティリティの実行が必要になります。

例16.5 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
```

クォータが有効化されたすべてのファイルシステム (オプション **-a**) のディスク使用状況レポートを表示するには、以下のコマンドを使用します。

```
# repquota -a
```

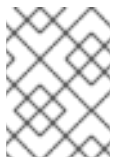
分かりやすいレポートが表示されますが、ここでいくつかのポイントを説明します。各ユーザーの後に表示される `--` はブロックまたは inode が超過しているかどうかを素早く判別するための手段です。どちらかのソフトリミットが超過していると、対応する `-` の位置に `+` が表示されます。最初の `-` はブロックの制限で、2 つ目が inode の制限を示します。

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。

16.2.3. 正確なクォータの維持

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** は定期的に行うことができます。**quotacheck** を定期的に行う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 の詳細は、**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 を安全に行う別の方法として、クォ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 の設定方法の詳細は、**man cron** を参照してください。

16.3. ディスククォータのリファレンス

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17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。

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。

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

17.1. 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、1 RAID アレイごとに 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。この拡大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

17.2. 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

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、各ディスクにミラー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 内でミラー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 内でミラー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 は稀にしか使用されないため、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。その結果、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 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ました。ただし、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り、全デバイスでの結合された集合データの転送速度が高速にな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、1 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 1 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。

17.3. 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ますようになります。

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 Mapper 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。

17.4. 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を提供しています。ただし、このユーティリティ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。

17.5. インストール後のルートディスクの RAID1 への変換

Red Hat Enterprise Linux 7 のインストール後に RAID の一部でないルートディスクを RAID1 ミラーに変換する必要がある場合は、Red Hat ナレッジベースの記事 [How do I convert my root disk to RAID1 after installation of Red Hat Enterprise Linux 7?](#) を参照してください。

PowerPC (PPC) アーキテクチャーでは、以下の追加ステップを行う必要があります。

1. PowerPC Reference Platform (PReP) ブートパーティションの内容を **/dev/sda1** から **/dev/sdb1** にコピーします。

```
# dd if=/dev/sda1 of=/dev/sdb1
```

2. 両方のディスクの最初のパーティション上で **prep** および **boot** フラグを更新します。

```
$ parted /dev/sda set 1 prep on
$ parted /dev/sda set 1 boot on

$ parted /dev/sdb set 1 prep on
$ parted /dev/sdb set 1 boot on
```

PowerPC マシンでは **grub2-install /dev/sda** コマンドを実行できず、エラーが返されますが、システムは想定どおり起動します。

17.6. RAID セットの設定

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 セットをいったん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。

17.7. 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17.1 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、マシン上の 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18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。

18.1. 現在マウントされているファイルシステムの一覧表示

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、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

18.1.1. ファイルシステムタイプの指定

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18.1「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18.1「現在マウントされている ext4 ファイルシステムの一覧表示」](#) を参照してください。

例18.1 現在マウントされている ext4 ファイルシステムの一覧表示

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 コマンドを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
```

18.2. ファイルシステムのマウント

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。

デバイス名、目的のディレクトリー、またはファイルシステムタイプなど、必要な情報をすべて指定せずに **mount** コマンド実行すると、**mount** コマンドは **/etc/fstab** ファイルの内容を読み取り、指定のファイルシステムが記載されているか確認します。**/etc/fstab** ファイルには、選択したファイルシステムがマウントされるデバイス名およびディレクトリーのリスト、ファイルシステムタイプ、およびマウントオプションが含まれます。そのため、**/etc/fstab** で指定されたファイルシステムをマウントするときに、以下のオプションの 1 つを選択できます。

```
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の指定」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"
```

18.2.1. ファイルシステムタイプの指定

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18.1「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18.1 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
xf	xf 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。

タイプ	詳細
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18.2「USB フラッシュドライブのマウント」](#) を参照してください。

例18.2 USB フラッシュドライブのマウント

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
```

18.2.2. マウントオプションの指定

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18.2「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18.2 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。

オプション	詳細
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18.3「ISO イメージのマウント」](#) を参照してください。

例18.3 ISO イメージのマウント

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/** とい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。

18.2.3. マウントポイントの共有

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18.4「共有マウントポイントの作成」](#)を参照してください。

例18.4 共有マウントポイントの作成

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。マウントポイントの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、シェルスクリ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18.5「スレーブマウントポイントの作成」](#) を参照してください。

例18.5 スレーブマウントポイントの作成

`/media` ディレクトリの内容が `/mnt` ディレクトリでも表示されるようにしながら、`/mnt` ディレクトリ内のマウントは `/media` ディレクトリ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` ディレクトリの複製を `/mnt` ディレクトリに作成して、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18.6「プライベートマウントポイントの作成」](#) を参照してください。

例18.6 プライベートマウントポイントの作成

[例18.4「共有マウントポイントの作成」](#) 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
```

バインド不可能なマウント

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、次のようにシェルスクリプトに入力します。

```
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18.7「バインド不可能なマウントポイントの作成」](#) を参照してください。

例18.7 バインド不可能なマウントポイントの作成

/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、シェルスクリ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
```

18.2.4. マウントポイントの移動

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18.8「既存の NFS マウントポイントの移動」](#) を参照してください。

例18.8 既存の NFS マウントポイントの移動

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
```

18.2.5. ルートへの読み取り専用パーミッションの設定

場合によっては、ルートファイルシステムを読み取り専用パーミッションでマウントする必要があることがあります。ユースケースの例には、システムの予期せぬ電源切断後に行うセキュリティの向上またはデータ整合性の保持が含まれます。

18.2.5.1. 起動時に読み取り専用パーミッションでマウントするようルートを設定

1. `/etc/sysconfig/readonly-root` ファイルで **READONLY** を **yes** に変更します。

```
# Set to 'yes' to mount the system file systems read-only.
READONLY=yes[出力は省略されています]
```

2. `/etc/fstab` ファイルでルートエントリー (`/`) の **defaults** を **ro** に変更します。

```
/dev/mapper/luks-c376919e... / ext4 ro,x-systemd.device-timeout=0 1
1
```

3. **ro** を `/etc/default/grub` ファイルの **GRUB_CMDLINE_LINUX** ディレクティブに追加し、**rw** が含まれないようにします。

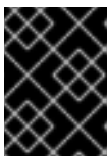
```
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet ro"
```

4. GRUB2 設定ファイルを再作成します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

5. **tmpfs** ファイルシステムで書き込みパーミッションでマウントされるファイルおよびディレクトリーを追加する必要がある場合は、`/etc/rwtab.d/` ディレクトリーでテキストファイルを作成し、そのファイルに設定を追加します。たとえば、`/etc/example/file` を書き込みパーミッションでマウントするには、以下の行を `/etc/rwtab.d/example` ファイルに追加します。

```
files /etc/example/file
```



重要

tmpfs のファイルおよびディレクトリーの変更内容は、再起動後に永続されません。

この手順の詳細は、「[書き込みパーミッションを保持するファイルおよびディレクトリー](#)」を参照してください。

6. システムを再起動します。

18.2.5.2. ルートを即時に再マウント

システムの起動時にルート (/) が読み取り専用パーミッションでマウントされた場合、書き込みパーミッションで再マウントできます。

```
~]# mount -o remount,rw /
```

これは、誤って / が読み取り専用パーミッションでマウントされた場合に便利です。

/ を読み取り専用パーミッションで再マウントする場合は、以下を実行します。

```
~]# mount -o remount,ro /
```



注記

このコマンドは、/ 全体を読み取り専用パーミッションでマウントします。「[起動時に読み取り専用パーミッションでマウントするようルートを設定](#)」に説明があるように、RAM にコピーして特定ファイルおよびディレクトリーの書き込みパーミッションを保持する方がよいでしょう。

18.2.5.3. 書き込みパーミッションを保持するファイルおよびディレクトリー

システムが適切に機能するためには、一部のファイルやディレクトリーに書き込みパーミッションが必要になります。ルートが読み取り専用モードである場合、これらのファイルやディレクトリーは **tmpfs** 一時ファイルシステムの RAM にマウントされます。このようなファイルおよびディレクトリーのデフォルトのセットは、以下が含まれる **/etc/rwtab** ファイルから読み取りされます。

```
dirs /var/cache/man
dirs /var/gdm[出力は省略されています]
empty /tmp
empty /var/cache/foomatic[出力は省略されています]
files /etc/adjtime
files /etc/ntp.conf[出力は省略されています]
```

/etc/rwtab ファイルのエントリーは以下の形式にしたがいます。

ファイルまたはディレクトリーが **tmpfs** にコピーされる方法
トリーへのパス

ファイルまたはディレク

ファイルまたはディレクトリーは 3 つの方法で **tmpfs** にコピーされるため、エントリーの種類は 3 つあります。

- **empty path**: 空のパスが **tmpfs** にコピーされます。例: **empty /tmp**
- **dirs path**: ディレクトリーツリーは空の状態では **tmpfs** にコピーされます。例: **dirs /var/run**
- **files path**: ファイルまたはディレクトリーツリーはそのまま **tmpfs** にコピーされます。例: **files /etc/resolv.conf**

カスタムパスを **/etc/rwtab.d/** に追加する場合も同じ形式が適用されます。

18.3. ファイルシステムのアンマウント

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の指定」を参照)。使用例については 例18.9「CD のアンマウント」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18.9 CD のアンマウント

/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
```

18.4. MOUNT コマンドのリファレンス

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。

18.4.1. 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。

18.4.2. 役立つ **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19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ーになります。**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。

19.1. VOLUME_KEY 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* からキーとパス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。

19.2. VOLUME_KEY の個人ユーザーとしての使用

volume_key を個人ユーザーとして使用すると、以下の手順に従って暗号キーを保存することができます。



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が。

手順19.1 volume_key のスタンドアロンとしての使用

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19.2 エスクローパケットを用いたデータアクセスの復元

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フレーズ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。

19.3. 規模の大きな組織での VOLUME_KEY の使用

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。

19.3.1. 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19.3 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。

- 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。
- 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。

- 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。

19.3.2. 暗号キーの保存

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ますようになります。



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が。

手順19.4 暗号キーの保存

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。

19.3.3. ボリュームへのアクセスの復元

暗号キーの保存 (「[暗号キーを保存するための準備](#)」 および 「[暗号キーの保存](#)」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19.5 ボリュームへのアクセスの復元

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。

19.3.4. 緊急時のパスフレーズの設定

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。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。

19.4. `VOLUME_KEY` のリファレンス

`volume_key` の詳細は、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20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、回転する円盤状の磁気記憶媒体にデータを格納します。SSD では、論理ブロックアドレス (LBA) 全体における、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では、広範囲のアドレスにまたがるデータにアクセスするため時間がかかります。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) では、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発行します (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)。

以下の 2 つの条件を満たす場合は、**discard** のサポートを有効にすると大変便利です。

- ファイルシステムに空き容量がある場合。
- 基盤のストレージデバイスの論理ブロックが、ほぼ書き込み済みである場合。

TRIM の詳細は、『Data Set Management T13 Specifications』 (以下のリンク) 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、『SCSI Block Commands 3 T10 Specification』 (以下のリンク) 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デバイスのすべてが **discard** に対応している訳ではありません。ソリッドステートデバイスが **discard** に対応しているかを判別するには、**/sys/block/sda/queue/discard_granularity** を確認します。

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** 場合は、1 番目のパーティションを 1MB の境界に作成することが推奨されます。

LVM が使用する論理ボリュームマネージャー (LVM)、device-mapper (DM) ターゲット、および MD (ソフトウェア RAID) は discard 要求をサポートします。discard 要求に対応しない DM ターゲットは dm-snapshot、dm-crypt、dm-raid45 です。dm-mirror では、Red Hat Enterprise Linux 6.1 で discard 要求に対応するようになり、MD は、バージョン 7.0 以降で discard 要求に対応するようになりました。

SSD で RAID レベル 5 を使用する場合は、SSD で **discard** が適切に処理されないとパフォーマンスが低下します。discard は、**raid456.conf** ファイルまたは GRUB2 設定にします。手順については、以下を参照してください。

手順20.1 raid456.conf に discard の設定

devices_handle_discard_safely モジュールパラメーターが **raid456** モジュールに設定されています。**raid456.conf** ファイルで discard を有効にするには、以下の手順を行います。

1. ハードウェアが discard をサポートするかどうかを確認するには、以下のコマンドを実行します。

```
# cat /sys/block/disk-name/queue/discard_zeroes_data
```

戻り値が **1** の場合は discard が有効です。**0** が返った場合は、RAID コードがディスクを取り除く必要があるため、さらに時間がかかります。

2. **/etc/modprobe.d/raid456.conf** ファイルを作成し、次の行を追加します。

```
options raid456 devices_handle_discard_safely=Y
```

3. **dracut -f** コマンドを実行して、初期 ramdisk (**initrd**) を再構築します。
4. システムを再起動して、変更を有効にします。

手順20.2 GRUB2 設定に discard を設定します。

devices_handle_discard_safely モジュールパラメーターは **raid456** モジュールに設定されています。GRUB2 設定で discard を有効にするには、以下を行います。

1. ハードウェアが discard をサポートするかどうかを確認するには、以下のコマンドを実行します。

```
# cat /sys/block/disk-name/queue/discard_zeroes_data
```

戻り値が **1** の場合は discard が有効です。**0** が返った場合は、RAID コードがディスクを取り除く必要があるため、さらに時間がかかります。

2. **/etc/default/grub** ファイルに次の行を追加します。

```
raid456.devices_handle_discard_safely=Y
```

3. GRUB2 設定ファイルの場所は、システムが BIOS ファームウェアか UEFI を使っているかで異なります。次のいずれかのコマンドを実行し、GRUB2 設定ファイルを再作成します。

- BIOS ファームウェアを使用している場合は、以下のコマンドを使用します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ファームウェアを使用している場合は、以下のコマンドを使用します。

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

4. システムを再起動して、変更を有効にします。



注記

Red Hat Enterprise Linux 7 で `discard` に完全に対応しているファイルシステムは、`ext4` と `XFS` だけです。

Red Hat Enterprise Linux 6.3 以前では、`ext4` ファイルシステムだけが `discard` に完全に対応しています。Red Hat Enterprise Linux 6.4 以降では、`ext4` ファイルシステムと `XFS` ファイルシステムの両方が `discard` を完全にサポートします。デバイスで `discard` コマンドを有効にするには、`mount` コマンドの **`discard`** オプションを使用します。たとえば、`discard` を有効にして `/dev/sda2` を `/mnt` にマウントするには、以下のコマンドを実行します。

```
# 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デバイスで問題が発生するのを回避するためです。Linux の **`swap`** コードは、`discard` を有効にしたデバイスに対して **`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。

パフォーマンスチューニングに関する注意点

ソリッドステートディスクでのパフォーマンスチューニングに関する詳細は、『Red Hat Enterprise Linux 7 パフォーマンスチューニングガイド』の「[ソリッドステートディスク](#)」セクションを参照してください。

第21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。

21.1. 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ます、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

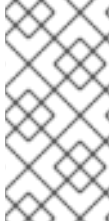
Linux カーネルでの書き込みバリアは、I/O の前後にストレージの書き込みキャッシュをフラッシュすることで実施されます。これは **順序が非常に重要** になります。トランザクションが書き込まれた後、ストレージキャッシュがフラッシュされてコミットブロックの書き込みが行われます。その後、再びキャッシュがフラッシュされます。これにより以下が確実に行われることになります。

- ディスクにすべてのデータが含まれる
- 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。

21.2. 書き込みバリアの有効化および無効化

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ものであり、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 21.1 「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21.1 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

21.3. 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の無効化

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

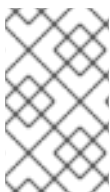
バッテリー駆動の書き込みキャッシュを持つハードウェア RAID コントローラーを使用している場合にも書き込みバリアは不要になります。このようなコントローラーを装備したシステムでそのコンポーネントとなるドライブが搭載している書き込みキャッシュが無効になっている場合、コントローラー自体がライトスルーにしていることがわかります。このため、電力供給停止時でも書き込みキャッシュのデータは維持されることがカーネルで認識されます。

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 -LA11 -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 -La11 -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22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。

22.1. 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する

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。

22.2. 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`

または

`/sys/block/disk/partition/alignment_offset`



注記

ファイルの場所は、ディスクが物理ディスク (ローカルディスク、ローカル RAID、またはマルチパス LUN) または仮想ディスクであるかによって異なります。最初のファイルの場所は物理ディスクに適用でき、2 つ目のファイルの場所は仮想ディスクに適用できます。これは、`virtio-blk` は常にパーティションのアライメント値を報告するためです。物理ディスクはアライメント値を報告する場合と報告しない場合があります。

- `/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、以下ようになります。

例22.1 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 `ioctl`s

- `BLKALIGNOFF`: `alignment_offset`
- `BLKPBSZGET`: `physical_block_size`
- `BLKSSZGET`: `logical_block_size`
- `BLKIOMIN`: `minimum_io_size`
- `BLKIOOPT`: `optimal_io_size`

22.3. 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` ユーティリティを与えます。このユーティリティを使って **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
```

22.4. 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。

22.5. 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。

22.6. 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マティックな 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23章 リモートディスクレスシステムの設定

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**

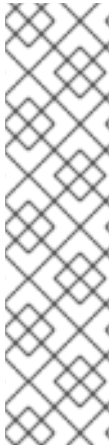


注記

dracut-network パッケージをインストールした後、以下の行を **/etc/dracut.conf** に追加します。

```
add_dracutmodules+="nfs"
```

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 ロードャ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。



注記

SELinux は NFSv4.2 上でのみサポートされます。SELinux を使用するには、以下の行を **/etc/sysconfig/nfs** に追加して NFS を明示的に有効にする必要があります。

```
RPCNFSDARGS="-V 4.2"
```

/var/lib/tftpboot/pxelinux.cfg/default で **root=nfs:server-ip:/exported/root/directory** を **root=nfs:server-ip:/exported/root/directory,vers=4.2** に変更します。

最後に、NFS サーバーを再起動します。

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。



重要

一部の RPM パッケージでは、ファイル機能 (**setcap** や **getcap** など) の使用が開始されました。しかし、現在 NFS はファイル機能をサポートしないため、ファイル機能を使用するパッケージのインストールや更新に失敗します。

23.1. ディスクレスクライアントの TFTP サービスの設定

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23.1 tftp の設定

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ください。

ディスククライアントの **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「[ディスククライアントの DHCP の設定](#)」および「[ディスククライアントのエクスポートしたファイルシステムの設定](#)」を参照してください。

23.2. ディスククライアントの DHCP の設定

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、「[ディスククライアントのエクスポートしたファイルシステムの設定](#)」を参照してください。



注記

libvirt 仮想マシンがディスククライアントとして使用されると、**libvirt** によって DHCP サービスが提供され、スタンドアロン DHCP サーバーは使用されません。このような場合、**libvirt** ネットワーク設定の `virsh net-edit` で `bootp file='filename'` オプションを使用してネットワークブートを有効にする必要があります。

23.3. ディスクレスクライアントのエクスポートしたファイルシステムの設定

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 --
releasever=
```

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23.2 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 (**vmlinux-kernel-version**)、**tftp** の boot ディレクトリーにコピーします。

```
# cp /boot/vmlinux-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
```

4. **initrd** のファイルパーミッションを 600 に変更する必要があります。変更しないと「file not found」(ファイルが見つかりません) というエラーが発生し、**pxelinux.0** ブートローダーが動作しません。以下のコマンドを実行してファイルパーミッションを変更します。

```
# chmod go-r initramfs-kernel-version.img
```

5. 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。
6. **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24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働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働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。

また、オンラインストレージの再設定を行う前にすべてのデータのバックアップを取ることを推奨します。

24.1. ターゲットの設定

Red Hat Enterprise Linux 7 では、**targetcli** シェルを、Linux-IO ターゲットの設定を表示、編集、および保存するためのフロントエンドとして使用するため、カーネルターゲットの設定ファイルを直接操作する必要はありません。**targetcli** ツールはコマンドラインインターフェースで、管理者はこれを使用してファイル、ボリューム、ローカル SCSI デバイス、または RAM ディスクに関連付けられたローカルストレージリソースをリモートにエクスポートできます。**targetcli** ツールははツリーベースのレイアウトと、組み込みのタブ補完が含まれ、完全な自動補完サポートとインラインドキュメントを提供します。

targetcli の階層は、カーネルのインターフェースに常に一致するとは限りません。これは、**targetcli** が可能な限り簡素化されているためです。



重要

targetcli の変更内容を永続化するには、ターゲットサービスを開始し、有効にします。

```
# systemctl start target
# systemctl enable target
```

24.1.1. targetcli のインストールおよび実行

targetcli をインストールするには、以下を実行します。

```
# yum install targetcli
```

target を開始するには、以下を実行します。

```
# systemctl start target
```

target サービスがブート時に起動するように設定するには、以下を実行します。

```
# systemctl enable target
```

targetcli コマンドを実行してから **ls** を実行し、ツリーインターフェイスのレイアウトを取得します。

```
# targetcli
:
/> ls
o- /.....[...]
  o- backstores.....[...]
    | o- block.....[Storage Objects: 0]
    | o- fileio.....[Storage Objects: 0]
    | o- pscsi.....[Storage Objects: 0]
    | o- ramdisk.....[Storage Objects: 0]
  o- iscsi.....[Targets: 0]
  o- loopback.....[Targets: 0]
```



注記

Red Hat Enterprise Linux 7.0 では、Bash から **targetcli** コマンド (例: **targetcli iscsi/ create**) を使用できず、エラーも出力されませんでした。Red Hat Enterprise Linux 7.1 からは、エラーステータスコードが出力されるようになったため、shell スクリプトで **targetcli** が使いやすくなりました。

24.1.2. バックストアの作成

バックストアは、ローカルマシンにエクスポートされた LUN のデータを保存するさまざまな方法に対応します。ストレージオブジェクトを作成して、バックストアが使用するリソースを定義します。



注記

Red Hat Enterprise Linux 6 では、作成されたマッピングを「バックングストア」と呼びました。他の意味で使われる「バックングストア」と混合しないように、Red Hat Enterprise Linux 7 では、作成されたマッピングを「ストレージオブジェクト」と呼び、別のバックングデバイスを示す場合は「バックストア」と呼んでいます。

LIO がサポートするバックストアデバイスは次のとおりです。

FILEIO (Linux ファイルと関連付けられたストレージ)

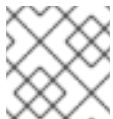
FILEIO ストレージオブジェクトは、**write_back** または **write_thru** 操作のいずれかをサポートします。**write_back** はローカルファイルシステムキャッシュを有効にします。これにより、パフォーマンスが向上しますが、データ損失のリスクが大きくなります。**write_back=false** を使用して **write_back** を無効にし、**write_thru** を利用することが推奨されます。

fileio ストレージオブジェクトを作成するには、以下の例のようにコマンド **/backstores/fileio create file_name file_location file_size write_back=false** を実行します。

```
/> /backstores/fileio create file1 /tmp/disk1.img 200M write_back=false
Created fileio file1 with size 209715200
```

BLOCK (Linux BLOCK デバイス)

ブロックドライバーは、**/sys/block** に指定されているブロックデバイスを LIO で使用できるようにします。このようなブロックデバイスには、物理デバイス (HDD、SSD、CD、DVD など) および論理デバイス (ソフトウェアまたはハードウェア RAID ボリューム、LVM ボリュームなど) が含まれます。



注記

通常、BLOCK バックストアは最良のパフォーマンスを提供します。

/dev/sdb ブロックデバイスを使用して BLOCK バックストアを作成するには、以下のコマンドを実行します。

```
/> /backstores/block create name=block_backend dev=/dev/sdb
Generating a wwn serial.
Created block storage object block_backend using /dev/sdb.
```

PSCSI (Linux パススルー SCSI デバイス)

SCSI エミュレーションなしで SCSI コマンドの直接パススルーをサポートするストレージオブジェクト、および **/proc/scsi/scsi** に **lsscsi** とともに表示される基盤の SCSI デバイスで SCSI コマンドの直接パススルーをサポートするストレージオブジェクトは、バックストアとして設定できます。SCSI-3 以上がこのサブシステムによってサポートされます。



警告

PSCSI は上級ユーザーのみが使用するようになっています。非対称論理ユニット割り当て (ALUA) や永続予約 (VMware ESX や vSphere によって使用される永続予約など) は、通常はデバイスのファームウェアに実装されず、誤作動やクラッシュが発生する原因となることがあります。確信が持てない場合、実稼働の設定には BLOCK を使用してください。

この例の `/dev/sr0` を使用して物理 SCSI デバイスである **TYPE_ROM** デバイスの PSCSI バックストアを作成するには、以下を使用します。

```
> backstores/pscsi/ create name=pscsi_backend dev=/dev/sr0
Generating a wwn serial.
Created pscsi storage object pscsi_backend using /dev/sr0
```

メモリーコピー RAM ディスク (Linux RAMDISK_MCP)

メモリーコピー RAM ディスク (**ramdisk**) は、完全な SCSI エミュレーションと、イニシエーターのメモリーコピーを使用した個別のメモリーマッピングが含まれる RAM ディスクを提供します。メモリーコピー RAM ディスクはマルチセッションの機能を提供し、実稼働を目的とした高速で不安定なマスタストレージでは特に便利です。

1 GB の RAM ディスクバックストアを作成するには、以下のコマンドを実行します。

```
> backstores/ramdisk/ create name=rd_backend size=1GB
Generating a wwn serial.
Created rd_mcp ramdisk rd_backend with size 1GB.
```

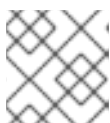
24.1.3. iSCSI ターゲットの作成

iSCSI ターゲットを作成するには、以下の手順に従います。

手順24.1 iSCSI ターゲットの作成

1. **targetcli** を実行します。
2. iSCSI の設定パスに移動します。

```
> iscsi/
```



注記

cd コマンドでもディレクトリの変更ができ、移動先へのパスも表示します。

3. デフォルトのターゲット名を使用して iSCSI ターゲットを作成します。

```
/iscsi> create
```

```
Created target
iqn.2003-01.org.linux-iscsi.hostname.x8664:sn.78b473f296ff
Created TPG1
```

または、特定の名前を使用して iSCSI ターゲットを作成します。

```
/iscsi > create iqn.2006-04.com.example:444
Created target iqn.2006-04.com.example:444
Created TPG1
```

4. **ls** でターゲットをリストし、新規作成されたターゲットが表示されることを確認します。

```
/iscsi > ls
o- iscsi.....[1 Target]
  o- iqn.2006-04.com.example:444.....[1 TPG]
    o- tpg1.....[enabled, auth]
      o- acls.....[0 ACL]
      o- luns.....[0 LUN]
      o- portals.....[0 Portal]
```



注記

Red Hat Enterprise Linux 7.1 では、ターゲットが作成されるとデフォルトのポータルも作成されます。

24.1.4. iSCSI ポータルの設定

iSCSI ポータルを設定するには、最初に iSCSI ターゲットを作成し、TPG と関連付ける必要があります。この手順は、[「iSCSI ターゲットの作成」](#) を参照してください。



注記

Red Hat Enterprise Linux 7.1 では、iSCSI ターゲットが作成されるとデフォルトのポータルも作成されます。このポータルは、デフォルトのポート番号 (0.0.0.0:3260) ですべての IP アドレスをリッスンするよう設定されます。このポータルを削除し、特定のポータルのみを追加するには、**/iscsi/iqn-name/tpg1/portals delete**

ip_address=0.0.0.0 ip_port=3260 を使用し、必要な情報で新しいポータルを作成します。

手順24.2 iSCSI ポータルの作成

1. TPG に移動します。

```
/iscsi> iqn.2006-04.example:444/tpg1/
```

2. ポータルを作成する方法には、デフォルトポータルを作成する方法と、リッスンする IP アドレスを指定してポータルを作成する方法の 2 つがあります。

デフォルトポータルを作成する方法では、デフォルトの iSCSI ポート 3260 を使用し、ターゲットがそのポートのすべての IP アドレスをリッスンできるようにします。

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create
```

```
Using default IP port 3260
Binding to INADDR_Any (0.0.0.0)
Created network portal 0.0.0.0:3260
```

リッスンする IP アドレスを指定してポータルを作成するには、以下のコマンドを使用します。

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create 192.168.122.137
Using default IP port 3260
Created network portal 192.168.122.137:3260
```

3. **ls** コマンドを実行して新規作成されたポータルが表示されることを確認します。

```
/iscsi/iqn.20...mple:444/tpg1> ls
o- tpg..... [enambled, auth]
  o- acls .....[0 ACL]
  o- luns .....[0 LUN]
  o- portals .....[1 Portal]
    o- 192.168.122.137:3260.....[OK]
```

24.1.5. LUN の設定

LUN を設定するには、最初にストレージオブジェクトを作成します。詳細は「[バックストアの作成](#)」を参照してください。

手順24.3 LUN の設定

1. 作成済みのストレージオブジェクトの LUN を作成します。

```
/iscsi/iqn.20...mple:444/tpg1> luns/ create
/backstores/ramdisk/ramdisk1
Created LUN 0.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/block/block1
Created LUN 1.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/fileio/file1
Created LUN 2.
```

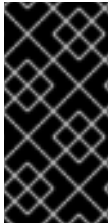
2. 変更内容を表示します。

```
/iscsi/iqn.20...mple:444/tpg1> ls
o- tpg..... [enambled, auth]
  o- acls .....[0 ACL]
  o- luns .....[3 LUNS]
    | o- lun0.....[ramdisk/ramdisk1]
    | o- lun1.....[block/block1 (/dev/vdb1)]
    | o- lun2.....[fileio/file1 (/foo.img)]
  o- portals .....[1 Portal]
    o- 192.168.122.137:3260.....[OK]
```



注記

デフォルトの LUN 名は 0 で始まることに注意してください。Red Hat Enterprise Linux 6 の **tgt** を使用した場合は 1 で始まっていました。



重要

デフォルトでは、読み書きパーミッションを持つ LUN が作成されます。ACL が作成された後に新しい LUN が追加された場合、その LUN は利用可能なすべての ACL に自動的にマップされます。これはセキュリティ上のリスクとなります。以下の手順に従って、読み取り専用の LUN を作成します。

手順24.4 読み取り専用の LUN の作成

1. 読み取り専用パーミッションを持つ LUN を作成するには、最初に以下のコマンドを使用します。

```
/> set global auto_add_mapped_luns=false
Parameter auto_add_mapped_luns is now 'false'.
```

これにより、LUN が既存の ACL へ自動的にマッピングされないようになり、LUN を手動でマッピングできるようになります。

2. 次に、コマンド **iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name/ create mapped_lun=next_sequential_LUN_number tpg_lun_or_backstore=backstore write_protect=1** を実行し、LUN を手動作成します。

```
/> iscsi/iqn.2015-06.com.redhat:target/tpg1/acls/iqn.2015-06.com.redhat:initiator/ create mapped_lun=1
tpg_lun_or_backstore=/backstores/block/block2 write_protect=1
Created LUN 1.
Created Mapped LUN 1.
/> ls
o- / ..... [....]
  o- backstores ..... [....]
    <snip>
  o- iscsi ..... [Targets: 1]
    | o- iqn.2015-06.com.redhat:target ..... [TPGs: 1]
      | o- tpg1 ..... [no-gen-acls, no-auth]
        | o- acls ..... [ACLs: 2]
          | o- iqn.2015-06.com.redhat:initiator .. [Mapped LUNs: 2]
            | | o- mapped_lun0 ..... [lun0 block/disk1 (rw)]
            | | o- mapped_lun1 ..... [lun1 block/disk2 (ro)]
          o- luns ..... [LUNs: 2]
            | o- lun0 ..... [block/disk1 (/dev/vdb)]
            | o- lun1 ..... [block/disk2 (/dev/vdc)]
          <snip>
```

(mapped_lun0 の (rw) とは異なり) mapped_lun1 行の最後に (ro) が表示されますが、これは、読み取り専用であることを表しています。

24.1.6. ACL の設定

接続している各イニシエーターに対して ACL を作成します。これにより、イニシエーターの接続時に

認証が強制され、LUN のみが各イニシエーターに公開されます。通常、各イニシエーターは LUN へ排他的にアクセスできます。ターゲットおよびイニシエーターは一意的識別名を持ちます。ACL を設定するには、イニシエーターの一意的名を知っている必要があります。open-iscsi イニシエーターの場合は、`/etc/iscsi/initiatorname.iscsi` で一意名を確認できます。

手順24.5 ACL の設定

1. `acls` ディレクトリーへ移動します。

```
/iscsi/iqn.20...mple:444/tpg1> acls/
```

2. ACL を作成します。イニシエーターの `/etc/iscsi/initiatorname.iscsi` にあるイニシエーター名を使用するか、簡単に覚えられる名前を使用している場合は「[iSCSI イニシエーターの作成](#)」を参照して ACL がイニシエーターと一致するようにします。例を以下に示します。

```
/iscsi/iqn.20...444/tpg1/acls> create iqn.2006-04.com.example.foo:888
Created Node ACL for iqn.2006-04.com.example.foo:888
Created mapped LUN 2.
Created mapped LUN 1.
Created mapped LUN 0.
```



注記

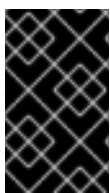
上記の例の挙動は、使用する設定によって異なります。この場合、グローバル設定の `auto_add_mapped_luns` が使用されます。これは、LUN を作成された ACL に自動的にマップします。

3. 変更内容を表示します。

```
/iscsi/iqn.20...444/tpg1/acls> ls
o- acls .....[1 ACL]
  o- iqn.2006-04.com.example.foo:888 ....[3 Mapped LUNs, auth]
    o- mapped_lun0 .....[lun0 ramdisk/ramdisk1 (rw)]
    o- mapped_lun1 .....[lun1 block/block1 (rw)]
    o- mapped_lun2 .....[lun2 fileio/file1 (rw)]
```

24.1.7. ファイバーチャネルオーバーイーサネット (FCoE) ターゲットの設定

`targetcli` を使用すると、「[ファイバーチャネルオーバーイーサネットインターフェースの設定](#)」で説明した FCoE における LUN のマウントの他に、FCoE の他のマシンへ LUN をエクスポートすることもサポートされます。



重要

次に進む前に、「[ファイバーチャネルオーバーイーサネットインターフェースの設定](#)」を参照して、基本的な FCoE 設定が完了しており、`fcoeadm -i` を実行すると、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24.6 FCoE ターゲットの設定

1. FCoE ターゲットを設定するには、**targetcli** 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。**targetcli** の基本および設定に関する詳細は、「[ターゲットの設定](#)」を参照してください。
2. 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が表示されることを確認してください。

3. 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24.1 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

```
/> tcm_fc/00:11:22:33:44:55:66:77
```

```
/> luns/ create /backstores/fileio/example2
```

4. 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。

5. 再起動後も変更を維持するには、**saveconfig** コマンドを使用し、入力が必要されたら **yes** を入力します。この作業を行わないと、再起動後に設定が失われます。
6. **exit** とタイプするか、**ctrl+D** を押して **targetcli** を終了します。

24.1.8. targetcli を使用したオブジェクトの削除

バックストアを削除するには、以下のコマンドを使用します。

```
/> /backstores/backstore-type/backstore-name
```

ACL などの iSCSI ターゲットの一部を削除するには、以下のコマンドを使用します。

```
/> /iscsi/iqn-name/tpg/acls/ delete iqn-name
```

(ACL、LUN、およびポータルのすべてを含む) ターゲット全体を削除するには、以下のコマンドを使用します。

```
/> /iscsi delete iqn-name
```

24.1.9. targetcli のリファレンス

targetcli の詳細については、次のリソースを参照してください。

man targetcli

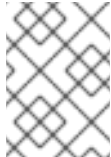
targetcli の man ページ。ウォークスルーの例が含まれています。

Linux SCSI Target Wiki

<http://linux-iscsi.org/wiki/Targetcli>

Andy Grover によるスクリーンキャスト

<https://www.youtube.com/watch?v=BkBGTBadOO8>



注記

この動画は 2012 年 2 月 28 日にアップロードされたため、サービス名が **targetcli** から **target** に変更されています。

24.2. iSCSI イニシエーターの作成

「[ターゲットの設定](#)」に従って **targetcli** でターゲットを作成したら、**iscsiadm** ユーティリティーを使用してイニシエーターを設定します。

Red Hat Enterprise Linux 7 のデフォルトでは、iSCSI サービスの起動に時間がかかります。したがって、**iscsiadm** コマンドが実行してから、このサービスが起動します。

手順24.7 iSCSI イニシエーターの作成

1. **iscsi-initiator-utils** をインストールします。

```
# yum install iscsi-initiator-utils
```

2. 「[ACL の設定](#)」で ACL にカスタム名を付けた場合は、それに合わせて **/etc/iscsi/initiatorname.iscsi** ファイルを修正します。

```
# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2006-04.com.example.node1
```

3. ターゲットを検出します。

```
# iscsiadm -m discovery -t st -p target-ip-address
10.64.24.179:3260,1 iqn.2006-04.com.example:3260
```

4. ターゲットサーバーの TPG ノードに、作成した ACL を設定します。

```
/iscsi/iqn.20...scsi:444/tpg1> set attribute generate_node_acls=1
```

次に、**exit** を実行します。

5. 手順 3 で検出したターゲット IQN で、ターゲットにログインします。

```
# iscsiadm -m node -T iqn.2006-04.com.example:3260 -l
Logging in to [iface: default, target: iqn.2006-04.com.example:3260,
```

```
portal: 10.64.24.179,3260] (multiple)
Login to [iface: default, target: iqn.2006-04.com.example:3260,
portal: 10.64.24.179,3260] successful.
```

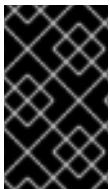
「ACL の設定」の説明どおりにイニシエーター名を ACL に追加しさえすれば、この手順に従って、同じ LUN に接続されるイニシエーターをいくつでも作成できます。

24.3. 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。

24.3.1. 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/targetH: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/hostH/`

- `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。

24.3.2. 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 には、以下のネイティブファイバーチャネルドライバーが同梱されます。

- `lpfc`
- `qla2xxx`
- `zfcp`
- `bfa`



重要

`qla2xxx` ドライバーは、デフォルトでイニシエーターモードで実行します。Linux-IO で `qla2xxx` を使用するには、対応する *qlini_mode* モジュールパラメーターでファイバーチャネルのターゲットモードを有効にします。

最初に、qla デバイス用のファームウェアパッケージ (`ql2200-firmware` など) がインストールされていることを確認します。

ターゲットモードを有効にするには、`/usr/lib/modprobe.d/qla2xxx.conf` `qla2xxx` モジュール設定ファイルに以下のパラメーターを追加します。

```
options qla2xxx qlini_mode=disabled
```

次に、`dracut -f` コマンドを実行して初期 ramdisk (`initrd`) を再構築し、システムを再起動して変更を有効にします。

表24.1 「ファイバーチャネル API の機能」 では、Red Hat Enterprise Linux 7 の各ネイティブドライバーにおけ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24.1 ファイバーチャネル API の機能

	lpfc	qla2xxx	zfcp	bfa
トランスポート <code>port_id</code>	X	X	X	X
トランスポート <code>node_name</code>	X	X	X	X

	lpfc	qla2xxx	zfcp	bfa
トランスポート port_name	X	X	X	X
リモートポート dev_loss_tmo	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
ホスト issue_lip	X	X		X
[a] Red Hat Enterprise Linux 5.4 以降でサポート [b] Red Hat Enterprise Linux 6.0 以降でサポート				

24.4. ファイバーチャネルオーバーイーサネットインターフェースの設定

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をインストールしたら、以下の手順を実行して、仮想 LAN (VLAN) で FCoE を有効にします。

手順24.8 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. 新規の VLAN を設定するには、既存のネットワークスクリプト (例: **/etc/fcoe/cfg-eth0**) のコピーを作成し、FCoE をサポートするイーサネットデバイスの名前を変更します。これは、デフォルトの設定ファイルになります。FCoE デバイスが **ethX** の場合は、以下を実行します。

```
# cp /etc/fcoe/cfg-ethx /etc/fcoe/cfg-ethX
```

必要に応じて **cfg-ethX** の内容を編集します。ネットワーキングインターフェースが、ハードウェアの Data Center Bridging Exchange (DCBX) プロトコルクライアントを実装場合は、**DCB_REQUIRED** を **no** に設定します。

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、そのデバイスの **/etc/sysconfig/network-scripts/ifcfg-ethX** ファイルに、**ONBOOT=yes** と設定します。たとえば、FCoE デバイスが eth2 の場合は、**/etc/sysconfig/network-scripts/ifcfg-eth2** ファイルに設定します。

3. 以下を実行し、データセンタブリッジングデーモン (**dcbd**) を起動します。

```
# systemctl start lldpad
```

4. ネットワーキングインターフェースがハードウェア DCBX クライアントを実装する場合は、このステップを省略します。

インターフェースでソフトウェア DCBX クライアントが必要な場合は、以下のコマンドを実行して、イーサネットインターフェースでデータセンタ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で FCoE を有効にします。

```
# dcbtool sc ethX app:fcoe e:1
```

これらのコマンドは、イーサネットインターフェースの **dcbd** の設定を変更していない場合に限り有効になります。

5. 以下を使用して、FCoE デバイスをロードします。

```
# ip link set dev ethX up
```

6. 次のコマンドで FCoE を開始します。

```
# systemctl start fcoe
```

他のすべての fabric 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、システムの起動時に FCoE と **lldpad** が実行するように設定することが推奨されます。これを実行するには、以下のように **chkconfig** を使用します。

```
# systemctl enable lldpad
```

```
# systemctl enable fcoe
```



注記

systemctl stop fcoe コマンドを実行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リセットされません。リセットするには、# **systemctl -s SIGHUP kill fcoe** コマンドを実行します。

Red Hat Enterprise Linux 7 では、DCB 対応のイーサネットインターフェースの DCB 設定をクエリおよび設定に、ネットワークマネージャーを使用できます。

24.5. 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24.2 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
    done
}
```

```

    # mount any newly discovered disk
    mount -a 2>/dev/null
}

```

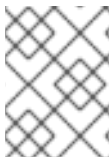
mount_fcoe_disks_from_fstab 関数は、**fcoe** サービススクリプトが **fcoemon** デーモンを開始した後に呼び出される必要があります。この関数が、**/etc/fstab** 内の以下のパスで指定されている FCoE ディスクをマウントします。

```

/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は、**mount_fcoe_disks_from_fstab** 関数が FCoE ディスクマウントエントリを識別できるようにします。**/etc/fstab** のエントリの詳細は、**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。

24.6. iSCSI

このセクションでは、iSCSI API および、**iscsiadm** ユーティリティについて説明します。**iscsiadm** ユーティリティを使用するには、**yum install iscsi-initiator-utils** を実行して **iscsi-initiator-utils** パッケージをインストールしてください。

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 iSCSI 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、ディスクバリーコマンド **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。

24.6.1. 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24.3 iscsiadm -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の詳細は、`/usr/share/doc/iscsi-initiator-utils-version/README` を参照してください。

24.7. 永続的な命名

Red Hat Enterprise Linux では、ストレージデバイスを識別する方法が複数あります。特にドライブへのインストール時やドライブの再フォーマット時に誤ったデバイスにアクセスしないようにするため、適切なオプションを使用して各デバイスを識別することが重要になります。

24.7.1. /dev/sd* およびメジャーおよびマイナー番号

sd ドライバーによって管理されるストレージデバイスは、メジャーデバイス番号とそれらに関連するマイナー番号によって内部で識別されます。この目的で使用されるメジャーデバイス番号は、連続した範囲にありません。各ストレージデバイスは、デバイス全体またはデバイス内のパーティションの識別に使用されるメジャー番号とマイナー番号によって表されます。**sd<letter(s)><optional number(s)>** の形式でデバイスおよび番号に割り当てられるメジャー番号およびマイナー番号の間には直接的な関係があります。**sd** ドライバーによって新しいデバイスが検出されると、利用可能なメジャー番号およびマイナー番号の範囲が割り当てられます。デバイスがオペレーティングシステムから削除されると、メジャー番号とマイナー番号の範囲が解放され、後で再使用されます。

デバイスが検出されると、メジャーおよびマイナー番号の範囲と、関連する **sd** 名がそのデバイスに割り当てられます。そのため、デバイス検出の順番が変わった場合、メジャーおよびマイナー番号の範囲と、関連する **sd** 名との関係が変更される可能性があります。これは、一部のハードウェア設定ではまれですが (シャーシ内の物理的な場所によって SCSI ターゲット ID が割り当てられる内部 SCSI コントローラーおよびディスクなどの場合)、以下の例のような状況で発生する可能性があります。

- ディスクが起動しなかったり、SCSI コントローラーに応答しなかった場合。この場合、通常のデバイスプロンプトによって検出されません。ディスクはシステムにアクセスできなくなり、後続のデバイスは関連する次の **sd** 名が含まれる、メジャーおよびマイナー番号の範囲を持ちます。たとえば、通常 **sdb** と呼ばれるディスクが検出されないと、**sdc** と呼ばれるディスクが **sdb** として表示されます。
- SCSI コントローラー (ホストバスアダプターまたは HBA) が初期化に失敗し、その HBA に接続されているディスクすべてが検出されなかった場合。後続のプロンプトされた HBA に接続しているディスクは、異なるメジャーおよびマイナー番号の範囲および異なる関連する **sd** 名を割り当てられます。

- システムに異なるタイプの HBA が存在する場合、ドライバー初期化の順序が変更される可能性があります。これにより、これらの HBA に接続されているディスクが異なる順序で検出される可能性があります。また、HBA がシステムの他の PCI スロットに移動された場合でも発生する可能性があります。
- ストレージアレイや干渉するスイッチの電源が切れた場合など、ストレージデバイスがプローブされたときに、ファイバーチャネル、iSCSI、または FCoE アダプターを持つシステムに接続されたディスクがアクセスできなくなる可能性があります。システムが起動するまでの時間よりもストレージアレイがオンラインになるまでの時間の方が長く、電源の障害後にシステムが再起動すると、この問題が発生する可能性があります。一部のファイバーチャネルドライバーは WWPN マッピングへの永続 SCSI ターゲット ID を指定するメカニズムをサポートしますが、メジャーおよびマイナー番号の範囲や関連する **sd** 名は予約されず、一貫性のある SCSI ターゲット ID 番号のみが提供されます。

そのため、**/etc/fstab** ファイルなどにあるデバイスを参照するときにメジャーおよびマイナー番号の範囲や関連する **sd** 名を使用することは望ましくありません。誤ったデバイスがマウントされ、データが破損する可能性があります。

しかし、場合によっては他のメカニズムが使用される場合でも **sd** 名を参照する必要があることがあります (デバイスによってエラーが報告される場合など)。これは、Linux カーネルはデバイスに関するカーネルメッセージで **sd** 名 (および SCSI ホスト、チャネル、ターゲット、LUN タプル) を使用するためです。

24.7.2. 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 は、SCSI 標準がすべての SCSI デバイスに必要とするシステムに依存しない永続的な ID です。各ストレージデバイスの WWID は一意となることが保証され、デバイスのアクセスに使用され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24.4 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 つのシステムから 1 つの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

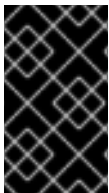
`/dev/mapper/wwid` 内に示します。

`multipath -l` コマンドは、永続化されていない識別子である **Host:Channel:Target:LUN**、`/dev/sd` 名、および **major:minor** 番号へのマッピングを表示します。

```
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 は、各 WWID ベースのデバイス名からシステムの対応する `/dev/sd` 名の適切なマッピングを自動的に維持します。これらの名前は、パスの変更後も永続され、他のシステムからデバイスにアクセスしたときに一貫性を保持します。

user_friendly_names 機能を使用すると (**device-mapper-multipath** の機能)、WWID は `/dev/mapper/mpathn` 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。

システムによって提供される永続的な名前他に、**udev** ルールを使用して独自の永続的な名前を実装し、ストレージの WWID にマップすることもできます。

24.7.3. udev メカニズム (`/dev/disk/by-*`) によって管理されるデバイス名

udev のメカニズムは、主な 3 つのコンポーネントで構成されます。

カーネル

デバイスの追加、削除、または変更時にユーザー領域へ送信されるイベントを生成します。

udev デーモン

イベントを受信します。

udev ルール

udev デーモンがカーネルイベントを受信したときに実行するアクションを指定します。

このメカニズムはストレージデバイスのみでなく、Linux の全種類のデバイスに使用されます。ストレージデバイスの場合、Red Hat Enterprise Linux には `/dev/disk/` ディレクトリーにシンボリックリンクを作成する **udev** ルールが含まれ、ストレージデバイスのコンテンツ、一意な識別子、シリアル番号、またはデバイスのアクセスに使用されるハードウェアパスを用いてストレージデバイスを参照できます。

/dev/disk/by-label

このディレクトリーのエントリーは、デバイスに保存されたコンテンツ (データ) のラベルを用いてストレージデバイスを参照するシンボリック名を提供します。**blkid** プログラムは、デバイスからデータを読み取り、そのデバイスの名前 (ラベル) を判断するために使用されます。例を以下に示します。

```
/dev/disk/by-label/Boot
```



注記

情報はデバイス上のコンテンツ (データ) から取得されるため、コンテンツが別のデバイスにコピーされても、ラベルは同じになります。

また、以下の構文どおりにラベルを使用すると **/etc/fstab** のデバイスを参照することもできます。

```
LABEL=Boot
```

/dev/disk/by-uuid

このディレクトリーのエントリーは、デバイスに保存されたコンテンツ (データ) の一意な識別子を用いてストレージデバイスを参照するシンボリック名を提供します。**blkid** プログラムは、デバイスからデータを読み取り、デバイスの一意な識別子 (uuid) を取得するために使用されます。例を以下に示します。

```
UUID=3e6be9de-8139-11d1-9106-a43f08d823a6
```

/dev/disk/by-id

このディレクトリーのエントリーは、一意な識別子 (他のストレージデバイスとは異なる) を用いてストレージデバイスを参照するシンボリック名を提供します。識別子はデバイスのプロパティですが、デバイス上のコンテンツ (データ) には保存されません。例を以下に示します。

```
/dev/disk/by-id/scsi-3600508e000000000ce506dc50ab0ad05
```

```
/dev/disk/by-id/wwn-0x600508e0000000000ce506dc50ab0ad05
```

ID は、デバイスのワールドワイド ID またはデバイスのシリアル番号から取得されます。**/dev/disk/by-id** エントリーにパーティション番号が含まれることもあります。例を以下に示します。

```
/dev/disk/by-id/scsi-3600508e000000000ce506dc50ab0ad05-part1
```

```
/dev/disk/by-id/wwn-0x600508e0000000000ce506dc50ab0ad05-part1
```

/dev/disk/by-path

このディレクトリーのエントリーは、デバイスのアクセスに使用されるハードウェアパスを用いてストレージデバイスを参照するシンボリック名を提供します。PCI 階層のストレージコントローラーへの参照で始まり、SCSI ホスト、チャネル、ターゲット、および LUN 番号が含まれ、任意でパーティション番号が含まれます。メジャーおよびマイナー番号または **sd** 名のよりも、これらの名前の使用が推奨されますが、ファイバーチャネル SAN 環境でターゲット番号が変更しないようにし

(永続バインディングなどを使用)、ホストアダプターを別の PCI ストックに移す場合は名前の使用を更新するよう注意する必要があります。さらに、HBA のプローブに失敗した場合、ドライバーが異なる順序でロードされた場合、または新しい HBA がシステムにインストールされた場合など、SCSI ホスト番号が変更する可能性があります。バイパスリストの例を以下に示します。

```
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:0
```

`/dev/disk/by-path` エントリーに以下のようなパーティション番号が含まれることもあります。

```
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:0-part1
```

24.7.3.1. udev デバイス命名規則の制約

udev 命名規則の制約の一部は次のとおりです。

- **udev** イベントに対して **udev** ルールが処理されるときに **udev** メカニズムはストレージデバイスをクエリーする機能に依存する可能性があるため、クエリーの実行時にデバイスにアクセスできない場合がある可能性があります。これは、デバイスがサーバーシャーシにない場合にファイバーチャネル、iSCSI、または FCoF ストレージデバイスで発生する可能性が高くなります。
- カーネルも **udev** イベントをいつでも送信する可能性があるため、デバイスにアクセスできない場合に `/dev/disk/by-*` リンクが削除される可能性があります。
- **udev** イベントが生成される時点とそのイベントが処理される時点の間で遅延が発生することがあります (大量のデバイスが検出され、ユーザー領域の **udev** デモンによる各デバイスのルールの処理に時間がかかる場合など)。これにより、カーネルがデバイスを検出する時点と `/dev/disk/by-*` の名前が利用できる時点の間で遅延が発生することがあります。
- ルールによって呼び出される **blkid** などの外部プログラムによってデバイスが短期間開放され、他の目的でデバイスにアクセスできなくなる可能性があります。

24.8. 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24.9 デバイスの完全削除

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、**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24.9「デバイスの完全削除」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のスキャン](#)」) 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のスキャン](#)」を参照)。

24.9. ストレージデバイスへのパスの削除

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24.10 ストレージデバイスへのパスの削除

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24.9「[デバイスの完全削除](#)」の記載を参照)。

手順24.10「[ストレージデバイスへのパスの削除](#)」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のスキャン](#)」を参照)。

24.10. 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24.11 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のスキャン](#)」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24.5 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 チャンネル、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 チャンネル、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。

24.11. ストレージの相互接続のスキャン

コマンドによっては、1 つまたは複数の相互接続のリセットやスキャン (または両方) を実行でき、1 つの操作で複数のデバイスを追加および削除する可能性があります。このようなスキャンは、I/O 操作のタイムアウト中に遅延が発生する原因となり、デバイスが予期せず削除される可能性があるため、大きな混乱が生じる可能性があります。Red Hat は、**必要な場合のみ**相互接続のスキャンを使用することを推奨します。ストレージの相互接続をスキャンするときは以下の制約に注意してください。

- 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。
- 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 件中 10 件を超えるサンプルでメモリー空き領域がメモリーの合計の 5% 未満である場合は、相互接続のスキャンは推奨されません。また、スワップがアクティブである場合 (**vmstat** 出力の **si** と **so** の欄がゼロ以外の値である場合)、相互接続のスキャンは推奨されません。**free** コマンドを使用してメモリーの合計容量を表示することもできます。

以下のコマンドを使用すると、ストレージの相互接続をスキャンできます。

```
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、デバイスの追加や削除が行われます。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ために必要になります。

issue_lip は非同期の操作であ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することがあります。**/var/log/messages** を監視して **issue_lip** の完了を判断する必要があります。

lpfc、**qla2xxx**、および **bnx2fc** ドライバーは **issue_lip** をサポートします。Red Hat Enterprise Linux の各ドライバーによってサポートされる API 機能については [表24.1「ファイバーチャネル API の機能」](#) を参照してください。

```
/usr/bin/rescan-scsi-bus.sh
```

`/usr/bin/rescan-scsi-bus.sh` スクリプトは Red Hat Enterprise Linux 5.4 で導入されました。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。既知の問題を含むこのスクリプトの詳細は「[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/hosth/scan
```

このコマンドは、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です。ただし、この場合にはチャンネル番号、SCSI ターゲット ID、および LUN の値がワイルドカードになります。識別子とワイルドカードのあらゆる組み合わせが可能であるため、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。

```
modprobe --remove driver-name,modprobe driver-name
```

`modprobe --remove driver-name` コマンドの実行後に `modprobe driver-name` コマンドを実行すると、ドライバーによって制御されるすべての相互接続の状態が完全に再初期化されます。これは強硬手段ではありますが、場合によっては適切な処置であることがあります。たとえば、このコマンドを使用すると異なるモジュールパラメーターの値でドライバーを再起動することができます。

24.12. 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 相互接続のスキャン](#)」を参照してください。

iSCSI ターゲットの検出を設定する詳細な方法は、**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。

24.13. 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に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せん。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。

24.13.1. 利用可能な **iface** 設定の表示

iSCSI オフロードとインターフェースバインディングは、以下の iSCSI イニシエーター実装でサポートされます。

ソフトウェア iSCSI

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

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 (**ethX**) 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24.2 **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24.6 **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24.7 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24.8 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
```

24.13.2. 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24.9 iface0 の MAC アドレスの設定

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。

24.13.3. iSCSI Offload 用 **iface** の設定

デフォルトでは、**iscsiadm** は各ポートの **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 で使用する同じコマンドを使用します (例: **iscsiadm -m iface**)。

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るべき 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デバイスの場合、IP アドレスは BIOS のセットアップ画面で設定されます。その他のデバイスすべてで **iface** の IP アドレスを設定するには、以下を実行します。

```
# iscsiadm -m iface -I iface_name -o update -n iface.ipaddress -v
target_IP
```

例24.10 Chelsio カードの **iface** IP アドレスの設定

たとえば、**iface** 名が **cxgb3i.00:07:43:05:97:07** のカードを使用するときに **iface** の IP アドレスを **20.15.0.66** 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
```

24.13.4. **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、**iface** 構成には、検出されたポータルに手作業で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。

24.14. iSCSI 相互接続のスキャン

iSCSI では、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。

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24.11 iscsiadm を使用した sendtargets コマンドの発行

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24.12 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。

デバイスモデルによっては、1つのターゲットに複数の論理ユニットやポータルがあるものがあります。この場合、最初に **sendtargets** コマンドをホストに発行して、ターゲット上で新しいポータルを探します。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** コマンドをホストに実行し、**各ターゲット**の新しいポータルを探す必要があります。**--rescan** オプションを使用して既存のセッションを再スキャンし、既存セッションの新しい論理ユニットを検索します。

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24.13 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-d1585-03-1
```

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24.14 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-d1585-03-1 \ --portal 10.16.41.155:3260,0 --login[8]
```

24.15. iSCSI ターゲットへのログイン

「[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 **/mnt/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](#)」と「[iSCSI 相互接続のスキャン](#)」を参照してください。

24.16. オンライン論理ユニットのサイズ変更

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。

24.16.1. ファイバーチャネル論理ユニットのサイズ変更

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ます。どのデバイスがマルチパス論理ユニットのパスであるかを判断するには、**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ーを探します。エントリーの WWID を参照すると、サイズ変更した論理ユニットに一致するエントリーを簡単に見つけることができます。

24.16.2. iSCSI 論理ユニットのサイズ変更

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 相互接続のスキャン](#)」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したときに使用したコマンドと同じです。

24.16.3. マルチパスデバイスのサイズ更新

マルチパス機能がシステムで有効になっている場合は、論理ユニットサイズでの変更を論理ユニットの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)。**`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`** を使用して **`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ーになります。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。

- **`mpathX`**: ***X*** は使用デバイスの該当エントリーになります (**`mpath0`** など)
- **`WWID: 3600508b400105e210000900000490000`** 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ーの確認には **`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一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。

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。

24.16.4. 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、以下のようになります。

```
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24.12 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** のエントリーになります。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。

24.16.4.1. 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。たとえば、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 -ll** を使用して、変更される論理ユニットと同じエントリーを検索します。

例24.15 multipath -ll コマンドの使用

たとえば、上記の **multipath -ll** は WWID 36001438005deb47100005000000640000 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
```

24.16.4.2. 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 -ll** コマンドを使用して変更を確定します。

24.16.4.3. ドキュメンテーション

追加の情報は、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。

24.17. 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。

24.18. リンク切れ動作の修正

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。

24.18.1. 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24.13 リモートポートの状態判断

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24.14 dev_loss_tmo の変更

- **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 の詳細は、「[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 **sdM** 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/sdx** は、**/dev/sdx** 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、**sdx** は復元されます。

24.18.2. 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。

24.18.2.1. 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、SCSI レイヤーに対し、可能な時点で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、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** の詳細は、「**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
```

24.18.2.2. 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。

24.18.3. iSCSI のルート

iSCSI ディスクから直接ルート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、コマンドが即座に SCSI レイヤーのキューに置かれてはなりません。これは **dm-multipath** を実装する場合に行う作業とは逆になります。

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。

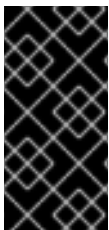
```
node.session.timeo.replacement_timeout = replacement_timeout
```

`/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 相互接続のスクリーン」](#) 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

ルート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。

24.19. 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、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はプロセスが異なります。この場合、ドライバーはエラーハンドラーをアクティベートする前に **rport** が再度オンラインになるまで数秒間待ちます。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らないようにします。

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** ファイルを編集します。

```
# echo value > /sys/block/device-name/device/timeout
```

このコマンド上の **value** は実装するタイムアウト値 (秒単位) に置き換えてください。

24.20. オンラインストレージ設定の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24.15 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24.16 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。

- 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24.17 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
```

24.21. EH_DEADLINE を使用したエラーからの回復における最大時間の設定

重要

ほとんどのシナリオでは、**eh_deadline** パラメーターを有効にする必要はありませんが、ファイバーチャネルスイッチとターゲットポートとの間でリンクが切れていたり、HBA (Host Bus Adapter) が、RSCN (Registered State Change Notification) を受け取らないなどの一部のシナリオでは、**eh_deadline** パラメーターを使用すると便利です。このとき、I/O 要求とエラー回復のコマンドではエラーが発生せず、すべてタイムアウトになります。したがって、この環境に **eh_deadline** を設定すると、回復時間に上限が設定されるため、失敗した I/O を、マルチパスで利用可能な別のパスで再試行できます。

ただし、RSCN を有効にすると、HBA がリンクを登録しなくなるか、もしくは利用できなくなります。I/O とエラー回復のコマンドがすぐに失敗し、それによりマルチパスの再試行が可能になるため、**eh_deadline** 機能が提供する利点が利用できなくなります。

SCSI ホストオブジェクトの **eh_deadline** パラメーターを使用すると、SCSI エラー処理がエラー回復を実行する最大時間 (HBA を停止してリセットするまでの時間) を設定することができます。

eh_deadline パラメーターの値は秒単位で指定し、デフォルト値は **off** です。**off** にすると、時間制限が無効になり、エラー回復がすべて行われます。**sysfs** とともに **scsi_mod.eh_deadline** カーネルパラメーターを使用すると、すべての SCSI HBA にデフォルト値を設定できます。

eh_deadline の有効期限が切れると HBA がリセットされるため、失敗したパスだけでなく、この HBA にあるすべてのターゲットパスが影響を受けます。その結果、何らかの理由で冗長パスの一部が利用できないと、I/O エラーが発生します。したがって、すべてのターゲットに、完全に冗長なマルチ

パス設定が設定されていない場合は、***eh_deadline*** を有効にしないでください。

[5] ***target_IP*** 変数および ***port*** 変数は、ターゲット/ポータルの IP アドレスおよびポートの組み合わせをそれぞれ参照します。詳細は [「iSCSI API」](#) および [「iSCSI 相互接続のスキャン」](#) を参照してください。

[6] ***proper_target_name*** の詳細は [「iSCSI 相互接続のスキャン」](#) を参照してください。

[7] セッションの SID 値を取得する方法の詳細は、[「iSCSI API」](#) を参照してください。

[8] このコマンドは、改行がない 1 行のコマンドになりますが、印刷版または PDF 版では複数の行で表されることがあります。行の前にバックスラッシュ「\」が付いている場合は、バックスラッシュを除いた 1 つのコマンドを示していることに注意してください。

[9] Red Hat Enterprise Linux 5.4 より前のバージョンでは、デフォルトの NOP-Out 要求のタイムアウトは 15 秒でした。

第25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。

25.1. 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。

25.2. 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。

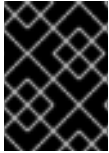
DM-Multipath は主に次のような理由で使用されます。

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26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。

26.1. 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、そ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、よりレベルが高い管理ツール向け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、ストレージ管理タスクをスクリプトで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、ならびに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 や相互接続帯域幅などのサーバーリソースは使用されません。

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の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ド上にあるすべてのデータを破棄する機能があります。論理エラーを削除するため、本番稼働システムを使用して作業する前に、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が推奨されます。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。

Red Hat Enterprise Linux 7 の **libStorageMgmt** パッケージは、REPORTED LUNS DATA HAS CHANGED という unit attention に対処する、デフォルトの udev ルールを追加します。

ストレージの設定が変更すると、Unit Attention ASC/ASCQ コードの 1 つが変更を報告します。その後 uevent が生成され、**sysfs** によって自動的に再スキャンされます。

/lib/udev/rules.d/90-scsi-ua.rules ファイルには、カーネルが生成できる他のイベントを列挙するサンプルルールが含まれます。

libStorageMgmt ライブラリーでは、プラグインのアーキテクチャーを使用してストレージアレイドの相違点に対応します。**libStorageMgmt** プラグインとその作成方法については、Red Hat の『開発者ガイド』を参照してください。

26.2. LIBSTORAGEMGMT の用語

アレイドベンダーおよびストレージ標準は複数存在し、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。このライブラリーでは、以下の用語を使用します。

ストレージアレイド

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) ストレージアレイドは、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。**multipath[2]** が有効になっている場合、1 つのボリュームは、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にアクセスできるのは、指定のイニシエーターに限られます。

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、割り当て時に、次に利用可能な LUN を選択します。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を使用して、IP ネットワーク経由で OS をホストする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 用語では DiskDrive として知られています。

イニシエーター

イニシエーターは、ファイバーチャネル (FC) またはファイバーチャネルオーバーイーサネット (FCoE) では、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、iSCSI 修飾名 (IQN) です。NFS または CIFS では、ホスト名またはホストの IP アドレスです。

子の依存関係

一部のアレイでは、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) の間に、暗黙的な関係があります。たとえば、親に依存する子が1つ以上ある場合は、その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。

26.3. インストール

コマンドラインや、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** と、任意で **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、適切なプラグインパッケージを1つ以上選択します。

```
$ 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、システムの起動時にデーモンが実行されるように、インストール、設定されますが、システムが次に再起動するまではこれが行なわれません。システムを再起動せずにデーモンを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、多くの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。その他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には、継続的に変更と改善が追加されています。

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
```

サービスを停止するには、以下を使用します。

```
$ sudo systemctl stop libstoragemgmt
```

サービスを開始するには、以下を使用します。

```
$ sudo systemctl start libstoragemgmt
```

26.4. 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 ツールを実行するには、以下の 2 つが必要になります。

- アレイと、アレイに必要な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)。
- アレイで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

必要な要件は、各プラグインで異なります。

例26.1 プラグインによって要件が異なる例

ユーザー名またはパスワードを必要としない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

URI を使用方法としては、以下の 3 つがあります。

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. `~/lsmcli` ファイルに URI を指定します。ここでは、名前と値のペアを「=」で区切ります。現在サポートされる設定は `uri` のみです。

使用する URI を指定するのは、この順番で行う必要があります。3 つの方法がすべて使われる場合は、コマンドラインで一番最初に実行されたものが適用されます。

コマンドラインに `-P` を追加するか、または環境変数の `LSMCLI_PASSWORD` に設定して、パスワードを指定します。

例26.2 lsmcli の例

以下の例は、コマンドラインを使用して新規ボリュームを作成し、これをイニシエーターに表示する方法を示します。

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
-----+-----+-----+-----
+-----+
P002 | Pool 2        | 18446744073709551616 | 18446744073709551616 |
sim-01
P003 | Pool 3        | 18446744073709551616 | 18446744073709551616 |
sim-01
P001 | Pool 1        | 18446744073709551616 | 18446744073709551616 |
sim-01
P004 | 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 P001 -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して、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の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がなくなります。さらに、プラグインの作成者が、独自に選択した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と、クライアントライブラリーが URI 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渡されるファイ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。クライアントがそのプラグインに対してライブラリーをオープンにしている間に、プラグインプロセスが実行します。1 つまたは複数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。

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。したがって、要求される操作によっては、数時間待機する可能性があります。コマンドラインで **-b** オプションを使用すれば、通常の使用に戻ることができます。終了コードが 0 の場合は、コマンドが完了しています。終了コードが 7 の場合はコマンドが進行中で、標準出力にはジョブ ID が書き込まれます。その後、必要に応じてそのジョブ ID を使用し、ユーザー自身またはスクリプトで **lsmcli --jobstatus JobID** を実行すれば、コマンドのステータスを照会できます。ジョブが完了すると、終了値は 0 となり、結果が標準出力に出力されます。コマンドが依然として進行中の場合は、戻り値が 7 となり、完成度 (パーセント) が標準出力に出力されます。

例26.3 非同期の例

ボリュームを作成します。コマンドがすぐに返されるように、**-b** オプションを渡します。

```
$ lsmcli volume-create --name async_created --size 20G --pool P001 -b
JOB_3
```

終了値を確認します。7 は、ジョブが依然として進行中であることを示しています。

```
$ 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
```

終了値を確認します。ジョブが依然として進行中であることを示す 7 が示されていたことに注意してください。ここでは、完成度のパーセントが標準出力として表示されます (上記の画面の場合では 33%)。

```
$ echo $?
7
```

少し待って、再びチェックします。成功を示す終了値 0 が出力され、標準出力には新規のボリュームが表示されます。

```
$ lsmcli job-status --job JOB_3
ID      | Name                               | vpd83                               | Block Size
| ...
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
--+-
Vol2 | async_created | 855C9BA51991B0CC122A3791996F6B15 | 512          |
...
```

スクリプトの場合は、**-t *SeparatorCharacters*** オプションを渡してください。これにより、出力の解析がより簡単になります。

例26.4 スクリプトの例

```
$ lsmcli list --type volumes -t#
Vol1#volume_name#049167B5D09EC0A173E92A63F6C3EA2A#512#41943040#214748364
80#OK#sim-01#P001
Vol2#async_created#3E771A2E807F68A32FA5E15C235B60CC#512#41943040#2147483
6480#OK#sim-01#P001
```

```
$ lsmcli list --type volumes -t " | "
Vol1 | volume_name | 049167B5D09EC0A173E92A63F6C3EA2A | 512 | 41943040 |
21474836480 | OK | 21474836480 | sim-01 | P001
Vol2 | async_created | 3E771A2E807F68A32FA5E15C235B60CC | 512 | 41943040
| 21474836480 | OK | sim-01 | P001
```

```
$ 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   | P001
-----
ID        | Vol2
Name      | async_created
VPD83     | 3E771A2E807F68A32FA5E15C235B60CC
Block Size | 512
#blocks   | 41943040
Size      | 21474836480
Status    | OK
System ID | sim-01
Pool ID   | P001
-----
```

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が推奨されます。

lsmcli の詳細は、man ページまたはコマンド **lsmcli --help** を参照してください。

付録A ストレージ管理に関する RED HAT CUSTOMER PORTAL LABS

Red Hat Customer Portal Labs のツールは、パフォーマンスの向上、問題のトラブルシューティング、セキュリティ問題の特定、設定の最適化に役立てていただくために開発しました。この付録では、ストレージ管理に関連する Red Hat Customer Portal Labs ツールの概要を説明します。Red Hat Customer Portal Labs は <https://access.redhat.com/labs/> からご利用いただくことができます。

SCSI DECODER

[SCSI decoder](#) は、SCSI エラーメッセージの理解を容易にするため、SCSI エラーメッセージを `/log/*` ファイルまたはログファイルのスニペットにデコードするように作られています。

SCSI decoder は、各 SCSI エラーメッセージを個別に診断し、問題を効果的に解決するためのソリューションを提示します。

FILE SYSTEM LAYOUT CALCULATOR

[File System Layout Calculator](#) は、現在使用しているストレージ、または使用が計画されているストレージに関する情報を入力すると、`ext3`、`ext4`、および `xfs` の各ファイルシステムを作成するのに最適なパラメーターを提示します。クエスチョンマーク ("?") にマウスポインターを置くと、各フィールドの簡単な説明が表示されます。また、画面の下の方にスクロールすると、すべてのオプションの説明が確認できます。

File System Layout Calculator で、指定した RAID ストレージに提供されているパラメーターを使用して、ファイルシステムを作成するコマンドを生成します。作成されたコマンドをコピーして、**root** でそのコマンドを実行して、必要なファイルシステムを作成します。

LVM RAID CALCULATOR

[LVM RAID Calculator](#) は、ストレージ情報を入力すると、RAID に論理ボリューム (LVM) を作成するのに最適なパラメーターを提示するために作られました。クエスチョンマーク ("?") にマウスポインターを置くと、各フィールドの簡単な説明が表示されます。また、画面の下の方にスクロールすると、すべてのオプションの説明が確認できます。

LVM RAID Calculator で、指定した RAID ストレージに LVM を作成するコマンドを生成します。作成されたコマンドをコピーして、**root** でそのコマンドを実行して、必要な LVM を作成します。

ISCSI HELPER

[iSCSI Helper](#) は、インターネットプロトコル (IP) ネットワーク越しにブロックレベルのストレージを提供し、サーバーの仮想環境でストレージプールを使用できるようにします。

iSCSI Helper では、入力した設定に基づいて、iSCSI ターゲット (サーバー) または iSCSI イニシエーター (クライアント) のロールに合わせてシステムを構成するスクリプトを生成します。

SAMBA CONFIGURATION HELPER

[Samba Configuration Helper](#) は、基本的なファイルと、Samba を通してプリンター共有を提供する設定を作成します。

- **Server** をクリックして、基本的なサーバー設定を表示します。
- **Shares** をクリックして、共有するディレクトリーを追加します。
- **Server** をクリックして、割り当てるプリンターを個別に追加します。

MULTIPATH HELPER

[Multipath Helper](#) は、Red Hat Enterprise Linux 5、6、または 7 のマルチパスデバイスに最適な設定を作成します。手順に従っていけば、カスタムのエイリアスやデバイスのブラックリストなど、高度なマルチパス設定を作成できます。

また、Multipath Helper は、確認に使用する **multipath.conf** ファイルも提供します。必要な設定が終了したら、インストールスクリプトをダウンロードして、サーバーで実行します。

NFS HELPER

[NFS Helper](#) を使えば、新しい NFS サーバーやクライアントの設定が簡単に行えます。手順に従ってエクスポートまたはマウントの情報を指定すると、NFS 設定スクリプトがダウンロードできるようになります。

MULTIPATH CONFIGURATION VISUALIZER

[Multipath Configuration Visualizer](#) は sosreport のファイルを解析してダイアグラムを提供し、マルチパス環境を視覚化します。**Multipath Configuration Visualizer** を使用して、以下を表示します。

- サーバーの HBA (Host Bus Adapters)、ローカルデバイス、および iSCSI デバイスを含むホストコンポーネント
- ストレージのストレージコンポーネント
- サーバーとストレージとの間のファブリック、またはイーサネットのコンポーネント
- 上記の全コンポーネントのパス

.xz、.gz、または .bz2 形式で圧縮した sosreport をアップロードするか、クライアントの解析用に選択したディレクトリーに sosreport を展開します。

RHEL BACKUP AND RESTORE ASSISTANT

[RHEL Backup and Restore Assistant](#) は、バックアップと復元のツールと、Linux における一般的なシナリオに関する情報を提供します。

対象ツール:

- **dump** および **restore**: ext2、ext3、または ext4 のファイルシステムのバックアップ
- **tar** および **cpio**: テープドライブをバックアップする際に使用される、ファイルおよびフォルダーのアーカイブまたは復元
- **rsync**: バックアップ操作の実行と、ファイルとディレクトリーの同期
- **dd**: 関連するファイルシステムやオペレーティングシステムとは無関係に、ブロックごとにファイルをコピー

対象シナリオ:

- 障害回復
- ハードウェアの移行
- パーティションテーブルのバックアップ
- 重要なフォルダーのバックアップ
- 増分バックアップ

- 差分バックアップ

付録B 改訂履歴

改訂 3-80.2 翻訳ファイルを XML ソースバージョン 3-80 と同期	Mon Feb 12 2018	Terry Chuang
改訂 3-80.1 翻訳ファイルを XML ソースバージョン 3-80 と同期	Sun Sep 24 2017	Terry Chuang
改訂 3-80 7.4 GA 公開用バージョン。	Thu Jul 27 2017	Milan Navratil
改訂 3-77 非同期のアップデート。	Wed May 24 2017	Milan Navratil
改訂 3-68 7.3 GA 公開用バージョン。	Fri Oct 21 2016	Milan Navratil
改訂 3-67 非同期のアップデート。	Fri Jun 17 2016	Milan Navratil
改訂 3-64 7.2 GA リリース向けのバージョン	Wed Nov 11 2015	Jana Heves
改訂 3-33 7.1 GA 向けバージョン	Wed Feb 18 2015	Jacquelynn East
改訂 3-26 Ceph の概要を追加	Wed Jan 21 2015	Jacquelynn East
改訂 3-22 7.1 Beta	Thu Dec 4 2014	Jacquelynn East
改訂 3-4 targetcli に関する新しい章を追加	Thu Jul 17 2014	Jacquelynn East
改訂 3-1 7.0 GA リリース向けバージョン	Tue Jun 3 2014	Jacquelynn East

索引

シンボル

[/boot/ ディレクトリー](#), [/boot/ ディレクトリー](#)

[/dev/shm](#), [ファイルシステム情報の収集](#)

[/etc/fstab](#), [Ext3 ファイルシステムへの変換](#), [/etc/fstab を使用した NFS ファイルシステムのマウント](#), [ファイルシステムのマウント](#)

[/etc/fstab ファイル](#)

[ディスククォータを有効化](#), [クォータの有効化](#)

[/local/directory](#) (クライアントの設定、マウント)

[NFS](#), [NFS クライアントの設定](#)

[/proc](#)

[/proc/devices](#), [/proc 仮想ファイルシステム](#)

[/proc/filesystems](#), [/proc 仮想ファイルシステム](#)

[/proc/mdstat](#), [/proc 仮想ファイルシステム](#)

[/proc/mounts](#), [/proc 仮想ファイルシステム](#)

[/proc/mounts/](#), [/proc 仮想ファイルシステム](#)

[/proc/partitions](#), [/proc 仮想ファイルシステム](#)

[/proc/devices](#)

[仮想ファイルシステム \(/proc\)](#), [/proc 仮想ファイルシステム](#)

[/proc/filesystems](#)

[仮想ファイルシステム \(/proc\)](#), [/proc 仮想ファイルシステム](#)

[/proc/mdstat](#)

[仮想ファイルシステム \(/proc\)](#), [/proc 仮想ファイルシステム](#)

[/proc/mounts](#)

[仮想ファイルシステム \(/proc\)](#), [/proc 仮想ファイルシステム](#)

[/proc/mounts/](#)

[仮想ファイルシステム \(/proc\)](#), [/proc 仮想ファイルシステム](#)

[/proc/partitions](#)

[仮想ファイルシステム \(/proc\)](#), [/proc 仮想ファイルシステム](#)

[/remote/export](#) (クライアントの設定、マウント)

[NFS](#), [NFS クライアントの設定](#)

[1 ユーザー](#)

[volume_key](#), [volume_key の個人ユーザーとしての使用](#)

[はじめに](#), [概要](#)

アンマウント, [ファイルシステムのアンマウント](#)

イニシエーターの実装

オフロードとインターフェースのバインディング

iSCSI, [利用可能な iface 設定の表示](#)

インストーラーのサポート

RAID, [インストーラーでの RAID サポート](#)

インストール時のストレージ設定

channel command word (CCW), [IBM System Z における DASD デバイスと zFCP デバイス](#)

DASD デバイスと zFCP デバイス - IBM System z, [IBM System Z における DASD デバイスと zFCP デバイス](#)

DIF/DIX を有効化にしているブロックデバイス, [DIF/DIX を有効にしているブロックデバイス](#)

iSCSI の検出と設定, [iSCSI の検出と設定](#)

LUKS/dm-crypt、ブロックデバイスの暗号化, [LUKS を使用してブロックデバイスを暗号化する](#)

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

古い BIOS RAID メタデータ, [古い BIOS RAID メタデータ](#)

更新, [ストレージをインストールする際の注意点](#)

最新情報, [ストレージをインストールする際の注意点](#)

インタラクティブな操作 (xfsrestore)

XFS, [インタラクティブな操作](#)

インデックスキー

FS-Cache, [FS-Cache](#)

エキスパートモード (xfs_quota)

XFS, [XFS クォータの管理](#)

エクスポートしたファイルシステム

ディスクレスのシステム, [ディスクレスクライアントのエクスポートしたファイルシステムの設定](#)

エラーメッセージ

書き込みバリア, [書き込みバリアの有効化および無効化](#)

オフラインの状態

Linux SCSI レイヤー, [SCSI コマンドタイマーとデバイス状態の制御](#)

オフロードとインターフェースバインディング

iSCSI, [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](#)

オプション (クライアントの設定、マウント)

NFS, [NFS クライアントの設定](#)

オンラインストレージ

トラブルシューティング, [オンラインストレージ設定のトラブルシューティング](#)

ファイバーチャネル, [ファイバーチャネル](#)

概要, [オンラインストレージ管理](#)

[sysfs](#), [オンラインストレージ管理](#)

オンライン論理ユニット

[read/write ステータスの変更](#), [オンライン論理ユニットの Read/Write ステータスの変更](#)

キャッシュの共有

[FS-Cache](#), [キャッシュの共有](#)

キャッシュの設定

[FS-Cache](#), [キャッシュの設定](#)

キャッシュの間引き制限

[FS-Cache](#), [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

キャッシュを設定する

[FS-Cache](#), [キャッシュの設定](#)

キャッシュバックエンド

[FS-Cache](#), [FS-Cache](#)

クォータの管理

[XFS](#), [XFS クォータの管理](#)

コヒーレンスデータ

[FS-Cache](#), [FS-Cache](#)

コマンド

[volume_key](#), [volume_key コマンド](#)

コマンドタイマー (SCSI)

[Linux SCSI レイヤー](#), [コマンドタイマー](#)

サイズ変更

[ext4](#), [ext4 ファイルシステムのサイズ変更](#)

サイズ変更済みの論理ユニット、サイズ変更, [オンライン論理ユニットのサイズ変更](#)

サイズ変更済み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変更](#)

サイト設定ファイルの無効化と拡大 (autofs)

[NFS](#), [autofs の設定](#)

サーバー (クライアントの設定、マウント)

[NFS](#), [NFS クライアントの設定](#)

システム情報

ファイルシステム, [ファイルシステム情報の収集](#)

[/dev/shm](#), [ファイルシステム情報の収集](#)

シンプルモード ([xfsrestore](#))

XFS, [xfsrestore](#) のシンプルモード

ストライピング

RAID, [RAID レベルとリニアサポート](#)

RAID の基本, [RAID \(Redundant Array of Independent Disks\)](#)

ストライプ配列

[ext4](#), [ext4 ファイルシステムの作成](#)

ストレージの相互接続、スキャン, [ストレージの相互接続のスキャン](#)

ストレージの相互接続をスキャンする, [ストレージの相互接続のスキャン](#)

ストレージをインストールする際の注意点

CCW (channel command word), [IBM System Z における DASD デバイスと zFCP デバイス](#)

DASD デバイスと zFCP デバイス - [IBM System z](#), [IBM System Z における DASD デバイスと zFCP デバイス](#)

DIF/DIX を有効にしているブロックデバイス, [DIF/DIX を有効にしているブロックデバイス](#)

iSCSI の検出と設定, [iSCSI の検出と設定](#)

LUKS/dm-crypt、ブロックデバイスの暗号化, [LUKS を使用してブロックデバイスを暗号化する](#)

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

古い BIOS RAID メタデータ, [古い BIOS RAID メタデータ](#)

更新, [ストレージをインストールする際の注意点](#)

最新情報, [ストレージをインストールする際の注意点](#)

ストレージアクセスのパラメーター

I/O の調整とサイズ, [ストレージアクセス用のパラメーター](#)

ストレージアクセス用のパラメーター

I/O の調整とサイズ, [ストレージアクセス用のパラメーター](#)

ストレージデバイスにパスを追加, [ストレージデバイスまたはパスの追加](#)

ストレージデバイスへのパス、削除, [ストレージデバイスへのパスの削除](#)

ストレージデバイスへのパス、追加, [ストレージデバイスまたはパスの追加](#)

ストレージデバイスへのパスを削除する, [ストレージデバイスへのパスの削除](#)

スループットクラス

[ソリッドステートディスク](#), [ソリッドステートディスクの導入ガイドライン](#)

スレーブマウント, [マウントポイントの共有](#)

ソフトウェア iSCSI

iSCSI, [ソフトウェア iSCSI 用 iface の設定](#)

[オフロードおよびインターフェースのバインディング](#)

iSCSI, ソフトウェア iSCSI 用 iface の設定

ソフトウェア iSCSI 用 iface

オフロードおよびインターフェースのバインディング

iSCSI, ソフトウェア iSCSI 用 iface の設定

ソフトウェア RAID (参照 RAID)

ソリッドステートディスク

SSD, ソリッドステートディスクの導入ガイドライン

TRIM コマンド, ソリッドステートディスクの導入ガイドライン

スループットクラス, ソリッドステートディスクの導入ガイドライン

デプロイメント, ソリッドステートディスクの導入ガイドライン

導入ガイドライン, ソリッドステートディスクの導入ガイドライン

ターゲット

iSCSI, iSCSI ターゲットへのログイン

ダンプのレベル

XFS, XFS ファイルシステムのバックアップと復元

ダーティーログ (XFS ファイルシステムの修復)

XFS, XFS ファイルシステムの修復

ダーティーログを持つ XFS ファイルシステムの修復

XFS, XFS ファイルシステムの修復

ツール (パーティション設定および他のファイルシステムの機能用)

I/O の調整とサイズ, パーティションとファイルシステムのツール

ディスククォータ, ディスク割り当て

その他のリソース, ディスククォータのリファレンス

の管理, ディスククォータの管理

レポート, ディスククォータに関するレポート

グループごとの割り当て, グループごとのクォータ割り当て

ソフトリミット, ユーザーごとのクォータ割り当て

ハードリミット, ユーザーごとのクォータ割り当て

ファイルシステムごとの割り当て, ソフトリミットの猶予期間の設定

ユーザーごとの割り当て, ユーザーごとのクォータ割り当て

有効化, ディスククォータの設定, 有効化と無効化

/etc/fstab の修正, クォータの有効化

quotacheck の実行, クォータデータベースファイルの作成

クォータファイルの作成, クォータデータベースファイルの作成

無効化, 有効化と無効化

猶予期間, [ユーザーごとのクォータ割り当て](#)

ディスククォータ
の管理

[quotacheck](#) コマンドを使用したチェック, [正確なクォータの維持](#)

ディスクストレージ (参照 [ディスククォータ](#))
[parted](#) (参照 [parted](#))

ディスクレスのシステム

DHCP、設定, [ディスクレスクライアントの DHCP の設定](#)

tftp サービス、設定, [ディスクレスクライアントの tftp サービスの設定](#)

エクスポートしたファイルシステム, [ディスクレスクライアントのエクスポートしたファイルシステムの設定](#)

ネットワーク起動サービス, [リモートディスクレスシステムの設定](#)

リモートディスクレスシステム, [リモートディスクレスシステムの設定](#)

必要なパッケージ, [リモートディスクレスシステムの設定](#)

ディスクレスクライアントの DHCP を設定する

ディスクレスのシステム, [ディスクレスクライアントの DHCP の設定](#)

ディスクレスクライアントの tftp サービスを設定する

ディスクレスのシステム, [ディスクレスクライアントの tftp サービスの設定](#)

ディレクトリー

[/boot/](#), [/boot/](#) ディレクトリー

[/dev/](#), [/dev/](#) ディレクトリー

[/etc/](#), [/etc/](#) ディレクトリー

[/mnt/](#), [/mnt/](#) ディレクトリー

[/opt/](#), [/opt/](#) ディレクトリー

[/proc/](#), [/proc/](#) ディレクトリー

[/srv/](#), [/srv/](#) ディレクトリー

[/sys/](#), [/sys/](#) ディレクトリー

[/usr/](#), [/usr/](#) ディレクトリー

[/var/](#), [/var/](#) ディレクトリー

デバイス、削除, [ストレージデバイスの削除](#)

デバイスがブロックされているか確認する

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

デバイスの削除, [ストレージデバイスの削除](#)

デバイスの状態

Linux SCSI レイヤー, [デバイスの状態](#)

デプロイメント

ソリッドステートディスク, [ソリッドステートディスクの導入ガイドライン](#)

トラブルシューティング

オンラインストレージ, [オンラインストレージ設定のトラブルシューティング](#)

トランスポート

ファイバーチャネル API, [ファイバーチャネル API](#)

ドライバー (ネイティブ)、ファイバーチャネル, [ネイティブファイバーチャネルのドライバーと機能](#)

ネイティブファイバーチャネルドライバー, [ネイティブファイバーチャネルのドライバーと機能](#)

ネットワークファイルシステム (参照 [NFS](#))

ネットワーク起動サービス

ディスクレスのシステム, [リモートディスクレスシステムの設定](#)

ハイエンドのアレイ

書き込みバリア, [ハイエンドのアレイ](#)

ハードウェア RAID (参照 [RAID](#))

ハードウェア RAID のコントローラードライバー

[RAID](#), [Linux ハードウェア RAID のコントローラードライバー](#)

バインド不可能なマウント, [マウントポイントの共有](#)

バックアップと復元

[XFS](#), [XFS ファイルシステムのバックアップと復元](#)

バックアップの復元

[XFS](#), [XFS ファイルシステムのバックアップと復元](#)

バッテリー駆動の書き込みキャッシュ

書き込みバリア, [バッテリー駆動の書き込みキャッシュ](#)

パリティ

[RAID](#), [RAID レベルとリニアサポート](#)

パーティション

サイズ変更, [fdisk を使用したパーティションのサイズ変更](#)

フォーマット

[mkfs](#), [パーティションのフォーマットとラベル付け](#)

一覧の表示, [パーティションテーブルの表示](#)

作成, [パーティションの作成](#)

[mkpart](#), [パーティションの作成](#)

削除, [パーティションの削除](#)

パーティションテーブル

表示, [パーティションテーブルの表示](#)

ファイバーチャネル

オンラインストレージ, [ファイバーチャネル](#)

ファイバーチャネル API, [ファイバーチャネル API](#)

ファイバーチャネルオーバーイーサネット

FCoE, [ファイバーチャネルオーバーイーサネットインターフェースの設定](#)

ファイバーチャネルドライバ (ネイティブ), [ネイティブファイバーチャネルのドライバと機能](#)

ファイルシステム, [ファイルシステム情報の収集](#)

Btrfs, [Btrfs \(テクノロジーレビュー\)](#)

ext2 (参照 ext2)

ext3 (参照 ext3)

FHS 標準, [FHS の組織](#)

構造, [ファイルシステムの構造およびメンテナンス](#)

組織, [FHS の組織](#)

階層, [ファイルシステム階層標準 \(FHS\) の概要](#)

ファイルシステムのその他のユーティリティー

ext4, [ext4 ファイルシステムのその他のユーティリティー](#)

ファイルシステムのサイズの拡大

XFS, [XFS ファイルシステムのサイズの拡大](#)

ファイルシステムのタイプ

ext4, [Ext4 ファイルシステム](#)

GFS2, [Global File System 2](#)

ファイルシステムの修復

XFS, [XFS ファイルシステムの修復](#)

ファイルシステムタイプ

XFS, [XFS ファイルシステム](#)

ブロックされたデバイス、確認

ファイバーチャネル

[リンク切れの動作を修正する](#), [ファイバーチャネル](#)

ブロックデバイス ioctls (ユーザー領域のアクセス)

I/O の調整とサイズ, [ブロックデバイス ioctls](#)

プライベートマウント, [マウントポイントの共有](#)

プロジェクト制限 (の設定)

XFS, [プロジェクト制限の設定](#)

ホスト

ファイバーチャネル API, [ファイバーチャネル API](#)

ポートの状態 (リモート)、判別

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

マウント

XFS, [XFS ファイルシステムのマウント](#)

マウント (クライアントの設定)

NFS, [NFS クライアントの設定](#)

マウントする, ファイルシステムのマウント

ext4, [ext4 ファイルシステムのマウント](#)

マウントポイントを移動する, マウントポイントの移動

ミラーリング

RAID, [RAID レベルとリニアサポート](#)

ユーザースペース API ファイル

ファイバーチャネル API, [ファイバーチャネル API](#)

ユーザー領域のアクセス

I/O の調整とサイズ, [ユーザー領域のアクセス](#)

リニア RAID

RAID, [RAID レベルとリニアサポート](#)

リモートディスクレスシステム

ディスクレスのシステム, [リモートディスクレスシステムの設定](#)

リモートポート

ファイバーチャネル API, [ファイバーチャネル API](#)

リモートポートの状態、判別

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

リモートポートの状態を判別する

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

リンク切れの動作を修正する, リンク切れ動作の修正

ファイバーチャネル, [ファイバーチャネル](#)

レベル

RAID, [RAID レベル](#)とリニアサポート

ログインする

iSCSI ターゲット, [iSCSI ターゲットへのログイン](#)

一時停止

XFS, [XFS ファイルシステム](#)の一時停止

主な特長

ext4, [Ext4 ファイルシステム](#)

XFS, [XFS ファイルシステム](#)

仮想ストレージ, [仮想ストレージ](#)

仮想ファイルシステム (/proc)

/proc/devices, [/proc 仮想ファイルシステム](#)

/proc/filesystems, [/proc 仮想ファイルシステム](#)

/proc/mdstat, [/proc 仮想ファイルシステム](#)

/proc/mounts, [/proc 仮想ファイルシステム](#)

/proc/mounts/, [/proc 仮想ファイルシステム](#)

/proc/partitions, [/proc 仮想ファイルシステム](#)

作成

ext4, [ext4 ファイルシステムの作成](#)

XFS, [XFS ファイルシステムの作成](#)

入出力パラメータのスタック

I/O の調整とサイズ, [入出力パラメーターのスタック](#)

共有サブツリー, [マウントポイントの共有](#)

スレーブマウント, [マウントポイントの共有](#)

バインド不可能なマウント, [マウントポイントの共有](#)

プライベートマウント, [マウントポイントの共有](#)

共有マウント, [マウントポイントの共有](#)

共有マウント, [マウントポイントの共有](#)

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

ストレージをインストールする際の注意点, [/home、/opt、/usr/local には別々のパーティションを用意する](#)

利用可能な iface の設定を表示する

オフロードとインターフェースのバインディング

iSCSI, [利用可能な iface 設定の表示](#)

割り当て機能

ext4, [Ext4 ファイルシステム](#)

XFS, [XFS ファイルシステム](#)

古い BIOS RAID メタデータ

ストレージをインストールする際の注意点, [古い BIOS RAID メタデータ](#)

実行中のセッション、その情報の取り込み

iSCSI API, [iSCSI API](#)

実行中の状態

Linux SCSI レイヤー, [SCSI コマンドタイマーとデバイス状態の制御](#)

導入ガイドライン

ソリッドステートディスク, [ソリッドステートディスクの導入ガイドライン](#)

必要なパッケージ

FCoE, [ファイバーチャネルオーバーイーサネットインターフェースの設定](#)

ディスクレスのシステム, [リモートディスクレスシステムの設定](#)

追加と削除

LUN (論理ユニット番号), [rescan-scsi-bus.sh](#) による論理ユニットの追加と削除

性能に関する保証

FS-Cache, [性能に関する保証](#)

既知の問題

追加と削除

LUN (論理ユニット番号), [rescan-scsi-bus.sh](#) に関する既知の問題

更新

ストレージをインストールする際の注意点, [ストレージをインストールする際の注意点](#)

書き込みのバリア

XFS, [書き込みバリア](#)

書き込みキャッシュ、無効にする

書き込みバリア, [書き込みキャッシュの無効化](#)

書き込みキャッシュを無効にする

書き込みバリア, [書き込みキャッシュの無効化](#)

書き込みバリア

ext4, [ext4 ファイルシステムのマウント](#)

NFS, [NFS](#)

エラーメッセージ, [書き込みバリアの有効化および無効化](#)

ハイエンドのアレイ, [ハイエンドのアレイ](#)

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](#)

定義, [書き込みバリア](#)

書き込みキャッシュを無効にする, [書き込みキャッシュの無効化](#)

書き込みバリアの動作, [書き込みバリアの動作](#)

書き込みバリアの重要性, [書き込みバリアの重要性](#)

有効または無効にする, [書き込みバリアの有効化および無効化](#)

書き込みバリアの動作

書き込みバリア, [書き込みバリアの動作](#)

書き込みバリアの重要性

書き込みバリア, [書き込みバリアの重要性](#)

最大サイズ

GFS2, [Global File System 2](#)

最大サイズ, [GFS2 ファイルシステム](#), [Global File System 2](#)

最新情報

ストレージをインストールする際の注意点, [ストレージをインストールする際の注意点](#)

有効または無効にする

書き込みバリア, [書き込みバリアの有効化および無効化](#)

検出

iSCSI, [iSCSI 検出の設定](#)

概要, [概要](#)

オンラインストレージ, [オンラインストレージ管理](#)

永続的な命名, [永続的な命名](#)

特定セッションのタイムアウト、設定

iSCSI 設定, [特定セッションのタイムアウトの設定](#)

相互接続 (スキャンする)

iSCSI, [iSCSI 相互接続のスキャン](#)

相互接続をスキャンする

iSCSI, [iSCSI 相互接続のスキャン](#)

累積モード (xfsrestore)

XFS, [xfsrestore の累積モード](#)

統計情報 (追跡)

FS-Cache, [統計情報](#)

統計情報を追跡する

FS-Cache, [統計情報](#)

自動マウント機能のマップを格納、格納に LDAP を使用 (autofs)

NFS, [サイトの設定ファイルの無効化/拡大](#)

記録の種類

検出

iSCSI, [iSCSI 検出の設定](#)

設定

検出

iSCSI, [iSCSI 検出の設定](#)

追加と削除

LUN (論理ユニット番号), [rescan-scsi-bus.sh](#) による論理ユニットの追加と削除

階層、ファイルシステム, [ファイルシステム階層標準 \(FHS\) の概要](#)

高度な RAID デバイス作成法

RAID, [高度な RAID デバイスの作成](#)

A

Anaconda サポート

RAID, [インストーラーでの RAID サポート](#)

API、iSCSI, [iSCSI API](#)

API、ファイバーチャネル, [ファイバーチャネル API](#)

ATA 標準

I/O の調整とサイズ, [ATA](#)

autofs , [autofs](#), [autofs](#) の設定

(参照 NFS)

autofs バージョン 5

NFS, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

B

bcull (キャッシュ間引き制限の設定)

FS-Cache, [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

brun (キャッシュ間引き制限の設定)

FS-Cache, [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

bstop (キャッシュ間引き制限の設定)

FS-Cache, [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

Btrfs

ファイルシステム, [Btrfs \(テクノロジーレビュー\)](#)

C

cachefiles

FS-Cache, [FS-Cache](#)

cachefilesd

FS-Cache, [キャッシュの設定](#)

CCW、channel command word

ストレージをインストールする際の注意点, [IBM System Z における DASD デバイスと zFCP デバイス](#)

channel command word (CCW)

ストレージをインストールする際の注意点, [IBM System Z における DASD デバイスと zFCP デバイス](#)

D

DASD デバイスと zFCP デバイス - IBM System z

ストレージをインストールする際の注意点, [IBM System Z における DASD デバイスと zFCP デバイス](#)

debugfs (ext4 ファイルシステムのその他のユーティリティー)

ext4, [ext4 ファイルシステムのその他のユーティリティー](#)

dev ディレクトリー, [/dev/ ディレクトリー](#)

device-mapper マルチパス機能, [DM-Multipath](#)

dev_loss_tmo

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

ファイバーチャネル API, [ファイバーチャネル API](#)

dev_loss_tmo を変更する

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

dev_loss_tmo、変更

ファイバーチャネル

リンク切れの動作を修正する, [ファイバーチャネル](#)

df, [ファイルシステム情報の収集](#)

DHCP、設定

ディスクレスのシステム, [ディスクレスクライアントの DHCP の設定](#)

DIF/DIX を有効にしているブロックデバイス

ストレージをインストールする際の注意点, [DIF/DIX を有効にしているブロックデバイス](#)

direct map support (autofs version 5)

NFS, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

dm-multipath

iSCSI 設定, [dm-multipath を実装している iSCSI の設定](#)

dmraid

RAID, [dmraid](#)

dmraid (RAID セットを設定する)

RAID, [dmraid](#)

du, [ファイルシステム情報の収集](#)

E

e2fsck, [Ext2 ファイルシステムに戻す](#)

e2image (ext4 ファイルシステムのその他のユーティリティー)

ext4, [ext4 ファイルシステムのその他のユーティリティー](#)

e2label

ext4, [ext4 ファイルシステムのその他のユーティリティー](#)

e2label (ext4 ファイルシステムのその他のユーティリティー)

ext4, [ext4 ファイルシステムのその他のユーティリティー](#)

enhanced LDAP support (autofs version 5)

NFS, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

etc ディレクトリー, [/etc/ ディレクトリー](#)

ext2

ext3 から戻す, [Ext2 ファイルシステムに戻す](#)

ext3

ext2 から変換, [Ext3 ファイルシステムへの変換](#)

作成, [Ext3 ファイルシステムの作成](#)

特徴, [Ext3 ファイルシステム](#)

ext4

debugfs (ext4 ファイルシステムのその他のユーティリティー), [ext4 ファイルシステムのその他のユーティリティー](#)

e2image (ext4 ファイルシステムのその他のユーティリティー), [ext4 ファイルシステムのその他のユーティリティー](#)

e2label, [ext4 ファイルシステムのその他のユーティリティー](#)

e2label ([ext4 ファイルシステムのその他のユーティリティー](#)), [ext4 ファイルシステムのその他のユーティリティー](#)

fsync(), [Ext4 ファイルシステム](#)

mkfs.ext4, [ext4 ファイルシステムの作成](#)

nobarrier マウントオプション, [ext4 ファイルシステムのマウント](#)

quota ([ext4 ファイルシステムのその他のユーティリティー](#)), [ext4 ファイルシステムのその他のユーティリティー](#)

resize2fs ([ext4 のサイズ変更](#)), [ext4 ファイルシステムのサイズ変更](#)

stride ([ストライプ配列を指定](#)), [ext4 ファイルシステムの作成](#)

stripe-width ([ストライプ配列を指定](#)), [ext4 ファイルシステムの作成](#)

tune2fs ([マウントする](#)), [ext4 ファイルシステムのマウント](#)

[サイズ変更](#), [ext4 ファイルシステムのサイズ変更](#)

[ストライプ配列](#), [ext4 ファイルシステムの作成](#)

[ファイルシステムのその他のユーティリティー](#), [ext4 ファイルシステムのその他のユーティリティー](#)

[ファイルシステムのタイプ](#), [Ext4 ファイルシステム](#)

[マウントする](#), [ext4 ファイルシステムのマウント](#)

[主な特長](#), [Ext4 ファイルシステム](#)

[作成](#), [ext4 ファイルシステムの作成](#)

[割り当て機能](#), [Ext4 ファイルシステム](#)

[書き込みバリア](#), [ext4 ファイルシステムのマウント](#)

F

fast_io_fail_tmo

[ファイバーチャネル API](#), [ファイバーチャネル API](#)

FCoE

[FCoE を使用するためにイーサネットインターフェースを設定](#), [ファイバーチャネルオーバーイーサネットインターフェースの設定](#)

[ファイバーチャネルオーバーイーサネット](#), [ファイバーチャネルオーバーイーサネットインターフェースの設定](#)

[必要なパッケージ](#), [ファイバーチャネルオーバーイーサネットインターフェースの設定](#)

[FCoE を使用するためにイーサネットインターフェースを設定](#)

[FCoE](#), [ファイバーチャネルオーバーイーサネットインターフェースの設定](#)

FHS, [ファイルシステム階層標準 \(FHS\) の概要](#), [FHS の組織](#)

([参照 ファイルシステム](#))

findmnt (コマンド)

[マウントのリスト](#), [現在マウントされているファイルシステムの一覧表示](#)

FS-Cache

[bcull](#) ([キャッシュ間引き制限の設定](#)), [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

[brun](#) ([キャッシュ間引き制限の設定](#)), [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

bstop (キャッシュ間引き制限の設定), [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

cachefiles, [FS-Cache](#)

cachefilesd, [キャッシュの設定](#)

NFS (で使用する), [NFS でのキャッシュの使用](#)

NFS (キャッシュの制限), [NFS でのキャッシュの制限](#)

tune2fs (キャッシュを設定する), [キャッシュの設定](#)

インデックスキー, [FS-Cache](#)

キャッシュの共有, [キャッシュの共有](#)

キャッシュの間引き制限, [キャッシュの間引き制限 \(Cache Cull\) の設定](#)

キャッシュを設定する, [キャッシュの設定](#)

キャッシュバックエンド, [FS-Cache](#)

コヒーレンスデータ, [FS-Cache](#)

性能に関する保証, [性能に関する保証](#)

統計情報 (追跡), [統計情報](#)

fsync()

ext4, [Ext4 ファイルシステム](#)

XFS, [XFS ファイルシステム](#)

G

GFS2

gfs2.ko, [Global File System 2](#)

ファイルシステムのタイプ, [Global File System 2](#)

最大サイズ, [Global File System 2](#)

GFS2 ファイルシステムの最大サイズ, [Global File System 2](#)

gfs2.ko

GFS2, [Global File System 2](#)

Global File System 2

gfs2.ko, [Global File System 2](#)

ファイルシステムのタイプ, [Global File System 2](#)

最大サイズ, [Global File System 2](#)

gquota/gqnoenforce

XFS, [XFS クォータの管理](#)

I

I/O の調整とサイズ, [ストレージの I/O 調整とサイズ](#)

ATA 標準, [ATA](#)

Linux I/O スタック, [ストレージの I/O 調整とサイズ](#)

logical_block_size, [ユーザー領域のアクセス](#)

LVM, [論理ボリュームマネージャー](#)

READ CAPACITY(16), [SCSI](#)

SCSI 標準, [SCSI](#)

sysfs インターフェース (ユーザー領域のアクセス), [sysfs インターフェース](#)

ストレージアクセスのパラメーター, [ストレージアクセス用のパラメーター](#)

ツール (パーティション設定および他のファイルシステムの機能用), [パーティションとファイルシステムのツール](#)

ブロックデバイス ioctl (ユーザー領域のアクセス), [ブロックデバイス ioctl](#)

ユーザー領域のアクセス, [ユーザー領域のアクセス](#)

入出力パラメーターのスタック, [入出力パラメーターのスタック](#)

iface (iSCSI オフロード用の設定)

オフロードとインターフェースバインディング

iSCSI, [iSCSI Offload 用 iface の設定](#)

iface のポータルに対する結合/結合解除

オフロードとインターフェースのバインディング

iSCSI, [iface のポータルに対する結合/結合解除](#)

iface の設定

オフロードとインターフェースのバインディング

iSCSI, [利用可能な iface 設定の表示](#)

iface の設定、表示

オフロードとインターフェースのバインディング

iSCSI, [利用可能な iface 設定の表示](#)

iface を結合/結合を解除する

オフロードとインターフェースのバインディング

iSCSI, [iface のポータルに対する結合/結合解除](#)

iSCSI

オフロードおよびインターフェースのバインディング

ソフトウェア iSCSI, [ソフトウェア iSCSI 用 iface の設定](#)

ソフトウェア iSCSI 用 iface, [ソフトウェア iSCSI 用 iface の設定](#)

オフロードとインターフェースのバインディング

iface のポータルに対する結合/結合解除, [iface のポータルに対する結合/結合解除](#)

iface の設定, [利用可能な iface 設定の表示](#)

iface の設定、表示, [利用可能な iface 設定の表示](#)

イニシエーターの実装, [利用可能な iface 設定の表示](#)

利用可能な iface の設定を表示する, [利用可能な iface 設定の表示](#)

オフロードとインターフェースバインディング, [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](#)

iface (iSCSI オフロード用の設定), [iSCSI Offload 用 iface の設定](#)

ソフトウェア iSCSI, [ソフトウェア iSCSI 用 iface の設定](#)

ターゲット, [iSCSI ターゲットへのログイン](#)

ログインする, [iSCSI ターゲットへのログイン](#)

検出, [iSCSI 検出の設定](#)

記録の種類, [iSCSI 検出の設定](#)

設定, [iSCSI 検出の設定](#)

相互接続をスキャンする, [iSCSI 相互接続のスキャン](#)

iSCSI API, [iSCSI API](#)

iSCSI のルート

iSCSI 設定, [iSCSI のルート](#)

iSCSI の検出と設定

ストレージをインストールする際の注意点, [iSCSI の検出と設定](#)

iSCSI の論理ユニット、サイズ変更, [iSCSI 論理ユニットのサイズ変更](#)

iSCSI の論理ユニットのサイズを変更する, [iSCSI 論理ユニットのサイズ変更](#)

issue_lip

ファイバーチャネル API, [ファイバーチャネル API](#)

L

lazy mount/unmount support (autofs version 5)

NFS, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

limit (xfs_quota エキスパートモード)

XFS, [XFS クォータの管理](#)

Linux I/O スタック

I/O の調整とサイズ, [ストレージの I/O 調整とサイズ](#)

logical_block_size

I/O の調整とサイズ, [ユーザー領域のアクセス](#)

LUKS/dm-crypt、ブロックデバイスの暗号化

ストレージをインストールする際の注意点, [LUKS を使用してブロックデバイスを暗号化する](#)

LUN (論理ユニット番号)

追加と削除, [rescan-scsi-bus.sh](#) による論理ユニットの追加と削除

[rescan-scsi-bus.sh](#), [rescan-scsi-bus.sh](#) による論理ユニットの追加と削除

必要なパッケージ, [rescan-scsi-bus.sh](#) による論理ユニットの追加と削除

既知の問題, [rescan-scsi-bus.sh](#) に関する既知の問題

LVM

I/O の調整とサイズ, [論理ボリュームマネージャー](#)

M

mdadm (RAID セットを設定する)

RAID, [mdadm](#)

mdraid

RAID, [mdraid](#)

mkfs , [パーティションのフォーマットとラベル付け](#)

mkfs.ext4

ext4, [ext4 ファイルシステムの作成](#)

mkfs.xfs

XFS, [XFS ファイルシステムの作成](#)

mkpart , [パーティションの作成](#)

mnt ディレクトリー, [/mnt/ ディレクトリー](#)

mount (コマンド), [mount コマンドの使い方](#)

オプション, [マウントオプションの指定](#)

ファイルシステムをマウントする, [ファイルシステムのマウント](#)

マウントポイントの表示, [現在マウントされているファイルシステムの一覧表示](#)

マウントポイントを移動する, [マウントポイントの移動](#)

共有サブツリー, [マウントポイントの共有](#)

スレーブマウント, [マウントポイントの共有](#)

バインド不可能なマウント, [マウントポイントの共有](#)

プライベートマウント, [マウントポイントの共有](#)

共有マウント, [マウントポイントの共有](#)

multiple master map entries per autofs mount point (autofs version 5)

NFS, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

N

NFS

[/etc/fstab](#) , [/etc/fstab](#) を使用した NFS ファイルシステムのマウント

[/local/directory](#) (クライアントの設定、マウント), [NFS クライアントの設定](#)

[/remote/export](#) (クライアントの設定、マウント), [NFS クライアントの設定](#)

autofs

configuration, [autofs の設定](#)

LDAP, [LDAP](#) を使用した自動マウント機能のマップの格納

拡大, [サイトの設定ファイルの無効化/拡大](#)

autofs バージョン 5, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

condrestart, [NFS の起動と停止](#)

direct map support (autofs version 5), [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

enhanced LDAP support (autofs version 5), [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

FS-Cache, [NFS でのキャッシュの使用](#)

lazy mount/unmount support (autofs version 5), [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

multiple master map entries per autofs mount point (autofs version 5), [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

NFS および rpcbind に関するトラブルシューティング, [NFS および rpcbind に関するトラブルシューティング](#)

proper nsswitch configuration (autofs version 5), use of, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

RDMA, [RDMA で NFS を有効にする \(NFSv4.1\)](#)

rfc2307bis (autofs), [LDAP を使用した自動マウント機能のマップの格納](#)

rpcbind , [NFS および rpcbind](#)

TCP, [NFS の仕組み](#)

UDP, [NFS の仕組み](#)

その他のリソース, [参照](#)

[インストールされているドキュメント](#), [インストールされているドキュメント](#)

[役に立つ Web サイト](#), [役に立つ Web サイト](#)

[関連書籍](#), [関連書籍](#)

はじめに, [NFS \(Network File System\)](#)

オプション (クライアントの設定、マウント), [NFS クライアントの設定](#)

クライアント

[autofs](#) , [autofs](#)

[マウントオプション](#), [一般的な NFS マウントオプション](#)

[設定](#), [NFS クライアントの設定](#)

[サイトの設定ファイルが無効化する／拡大する \(autofs\)](#), [autofs の設定](#)

[サーバー \(クライアントの設定、マウント\)](#), [NFS クライアントの設定](#)

[サーバーの設定](#), [NFS サーバーの設定](#)

[/etc/exports](#) , [/etc/exports 設定ファイル](#)

[exportfs コマンド](#), [exportfs コマンド](#)

[NFSv4で exportfs コマンド](#), [NFSv4 で exportfs の使用](#)

セキュリティー, [NFS の保護](#)

[NFSv3 ホストのアクセス](#), [AUTH_SYS とエクスポート制御による NFS の保護](#)

[NFSv4 ホストのアクセス](#), [AUTH_GSS による NFS の保護](#)

[ファイル権限](#), [ファイル権限](#)

ファイアウォールでの設定, [ファイアウォール背後での NFS の実行](#)

ホスト名の形式, [ホスト名の形式](#)

マウント (クライアントの設定), [NFS クライアントの設定](#)

停止, [NFS の起動と停止](#)

再読み込み, [NFS の起動と停止](#)

再起動, [NFS の起動と停止](#)

動作について, [NFS の仕組み](#)

必須サービス, [必須サービス](#)

書き込みバリア, [NFS](#)

状態, [NFS の起動と停止](#)

自動マウント機能のマップを格納、格納に LDAP を使用 (autofs), [サイトの設定ファイルの無効化/拡大](#)

起動, [NFS の起動と停止](#)

NFS (で使用する)

FS-Cache, [NFS でのキャッシュの使用](#)

NFS (キャッシュの制限)

FS-Cache, [NFS でのキャッシュの制限](#)

NFS および rpcbind に関するトラブルシューティング

NFS, [NFS および rpcbind に関するトラブルシューティング](#)

NFS でのキャッシュの制限

FS-Cache, [NFS でのキャッシュの制限](#)

nobarrier マウントオプション

ext4, [ext4 ファイルシステムのマウント](#)

XFS, [書き込みバリア](#)

NOP-Out (無効化)

iSCSI 設定, [iSCSI のルート](#)

NOP-Out を無効にする

iSCSI 設定, [iSCSI のルート](#)

NOP-Out 要求

リンク損失の修正

iSCSI 設定, [NOP-Out インターバル/タイムアウト](#)

O

opt ディレクトリー, [/opt/ ディレクトリー](#)

P

Parallel NFS

pNFS, [pNFS](#)

parted , [パーティション](#)

コマンドの表, [パーティション](#)

デバイスの選択, [パーティションテーブルの表示](#)

パーティションのサイズ変更, [fdisk を使用したパーティションのサイズ変更](#)

パーティションの作成, [パーティションの作成](#)

パーティションの削除, [パーティションの削除](#)

パーティションテーブルの表示, [パーティションテーブルの表示](#)

概要, [パーティション](#)

pNFS

Parallel NFS, [pNFS](#)

pquota/pqnoenforce

XFS, [XFS クォータの管理](#)

proc ディレクトリー, [/proc/ ディレクトリー](#)

proper nsswitch configuration (autofs version 5), use of

NFS, [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

Q

queue_if_no_path

iSCSI 設定, [dm-multipath を実装している iSCSI の設定](#)

リンク損失の修正

iSCSI 設定, [replacement_timeout](#)

quota (ext4 ファイルシステムのその他のユーティリティー)

ext4, [ext4 ファイルシステムのその他のユーティリティー](#)

quotacheck , [クォータデータベースファイルの作成](#)

quotacheck コマンド

でクォータの正確度をチェック, [正確なクォータの維持](#)

quotaoff , [有効化と無効化](#)

quotaon , [有効化と無効化](#)

R

RAID

Anaconda サポート, [インストーラーでの RAID サポート](#)

dmraid, [dmraid](#)

dmraid (RAID セットを設定する), [dmraid](#)

mdadm (RAID セットを設定する), [mdadm](#)

mdraid, [mdraid](#)

RAID のサブシステム, [Linux RAID サブシステム](#)

RAID セットを設定する, [RAID セットの設定](#)

インストーラーのサポート, [インストーラーでの RAID サポート](#)

ストライピング, [RAID レベルとリニアサポート](#)

ソフトウェア RAID, [RAID のタイプ](#)

ハードウェア RAID, [RAID のタイプ](#)

ハードウェア RAID のコントローラードライバー, [Linux ハードウェア RAID のコントローラードライバー](#)

パリティ, [RAID レベルとリニアサポート](#)

ミラーリング, [RAID レベルとリニアサポート](#)

リニア RAID, [RAID レベルとリニアサポート](#)

レベル, [RAID レベルとリニアサポート](#)

レベル 0, [RAID レベルとリニアサポート](#)

レベル 1, [RAID レベルとリニアサポート](#)

レベル 4, [RAID レベルとリニアサポート](#)

レベル 5, [RAID レベルとリニアサポート](#)

採用する利点, [RAID \(Redundant Array of Independent Disks\)](#)

高度な RAID デバイス作成法, [高度な RAID デバイスの作成](#)

RAID のサブシステム

RAID, [Linux RAID サブシステム](#)

RAID セットを設定する

RAID, [RAID セットの設定](#)

RDMA

NFS, [RDMA で NFS を有効にする \(NFSv4.1\)](#)

READ CAPACITY(16)

I/O の調整とサイズ, [SCSI](#)

read/write ステータスの変更

オンライン論理ユニット, [オンライン論理ユニットの Read/Write ステータスの変更](#)

Red Hat Enterprise Linux 固有のファイルの場所

/etc/sysconfig/, [特殊な Red Hat Enterprise Linux ファイルの場所](#)
(参照 [sysconfig ディレクトリー](#))

/var/cache/yum/, [特殊な Red Hat Enterprise Linux ファイルの場所](#)

/var/lib/rpm/, [特殊な Red Hat Enterprise Linux ファイルの場所](#)

replacement_timeout

リンク損失の修正

iSCSI 設定, [SCSI エラーハンドラー](#), [replacement_timeout](#)

replacement_timeoutM

iSCSI 設定, [iSCSI のルート](#)

report (xfs_quota エキスパートモード)

XFS, [XFS クォータの管理](#)

rescan-scsi-bus.sh

追加と削除

LUN (論理ユニット番号), [rescan-scsi-bus.sh](#) による論理ユニットの追加と削除

resize2fs, [Ext2 ファイルシステムに戻す](#)

resize2fs (ext4 のサイズ変更)

ext4, [ext4 ファイルシステムのサイズ変更](#)

rfc2307bis (autofs)

NFS, [LDAP を使用した自動マウント機能のマップの格納](#)

rpcbind , [NFS および rpcbind](#)

(参照 NFS)

NFS, [NFS および rpcbind に関するトラブルシューティング](#)

rpcinfo , [NFS および rpcbind に関するトラブルシューティング](#)

状態, [NFS の起動と停止](#)

rpcinfo , [NFS および rpcbind に関するトラブルシューティング](#)

S

SCSI エラーハンドラー

リング損失の修正

iSCSI 設定, [SCSI エラーハンドラー](#)

SCSI コマンドタイマー

Linux SCSI レイヤー, [コマンドタイマー](#)

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, [SCSI コマンドタイマーとデバイス状態の制御](#)

SCSI 標準

I/O の調整とサイズ, [SCSI](#)

srv ディレクトリー, [/srv/ ディレクトリー](#)

SSD

ソリッドステートディスク, [ソリッドステートディスクの導入ガイドライン](#)

SSM

System Storage Manager, [System Storage Manager \(SSM\)](#)

list コマンド, [検出された全デバイスの情報表示](#)

resize コマンド, [ボリュームの容量追加](#)

snapshot コマンド, [スナップショット](#)

インストール, [SSM のインストール](#)

バックエンド, [SSM のバックエンド](#)

stride (ストライプ配列を指定)

ext4, [ext4 ファイルシステムの作成](#)

stripe-width (ストライプ配列を指定)

ext4, [ext4 ファイルシステムの作成](#)

su (mkfs.xfs のサブオプション)

XFS, [XFS ファイルシステムの作成](#)

sw (mkfs.xfs のサブオプション)

XFS, [XFS ファイルシステムの作成](#)

swap 領域, [swap領域](#)

LVM2

作成, [swap の LVM2 論理ボリュームの作成](#)

削除, [swap の LVM2 論理ボリュームの削除](#)

拡張, [LVM2 論理ボリュームでの swap 領域の拡張](#)

縮小, [LVM2 論理ボリュームでの swap 領域の縮小](#)

ファイル

作成, [swap ファイルを作成する](#), [swap ファイルの削除](#)

作成, [Swap 領域の追加](#)

削除, [Swap 領域の削除](#)

拡張, [Swap 領域の追加](#)

推奨サイズ, [swap領域](#)

移動, [Swap 領域の移動](#)

sys ディレクトリー, [/sys/ ディレクトリー](#)

sysconfig ディレクトリー, [特殊な Red Hat Enterprise Linux ファイルの場所](#)

sysfs

概要

[オンラインストレージ](#), [オンラインストレージ管理](#)

sysfs インターフェース (ユーザー領域のアクセス)

I/O の調整とサイズ, [sysfs インターフェース](#)

System Storage Manager

SSM, [System Storage Manager \(SSM\)](#)

list コマンド, [検出された全デバイスの情報表示](#)

resize コマンド, [ボリュームの容量追加](#)

snapshot コマンド, [スナップショット](#)

インストール, [SSM のインストール](#)

バックエンド, [SSM のバックエンド](#)

T

tftp サービス、設定

[ディスクレスのシステム](#), [ディスクレスクライアントの tftp サービスの設定](#)

TRIM コマンド

[ソリッドステートディスク](#), [ソリッドステートディスクの導入ガイドライン](#)

tune2fs

で ext2 に戻す, [Ext2 ファイルシステムに戻す](#)

で ext3 に変換, [Ext3 ファイルシステムへの変換](#)

tune2fs (キャッシュを設定する)

[FS-Cache](#), [キャッシュの設定](#)

tune2fs (マウントする)

ext4, [ext4 ファイルシステムのマウント](#)

U

udev rule (timeout)

[command timer \(SCSI\)](#), [コマンドタイマー](#)

umount, [ファイルシステムのアンマウント](#)

uquota/uqnoenforce

[XFS](#), [XFS クォータの管理](#)

usr ディレクトリー, [/usr/ ディレクトリー](#)

V

var ディレクトリー, [/var/ ディレクトリー](#)

[var/lib/rpm/ ディレクトリー](#), [特殊な Red Hat Enterprise Linux ファイルの場所](#)

[var/spool/up2date/ ディレクトリー](#), [特殊な Red Hat Enterprise Linux ファイルの場所](#)

version

what is new

[autofs](#), [autofs バージョン 5 の改善点 \(バージョン 4 との比較\)](#)

volume_key

1 ユーザー, [volume_key の個人ユーザーとしての使用](#)

コマンド, [volume_key](#) コマンド

W

World Wide Identifier (WWID)

永続的な命名, [WWID](#)

WWID

永続的な命名, [WWID](#)

X

XFS

`fsync()`, [XFS ファイルシステム](#)

`gquota/gqnoenforce`, [XFS クォータの管理](#)

`limit (xfs_quota エキスパートモード)`, [XFS クォータの管理](#)

`mkfs.xfs`, [XFS ファイルシステムの作成](#)

`nobarrier` マウントオプション, [書き込みバリア](#)

`pquota/pqnoenforce`, [XFS クォータの管理](#)

`report (xfs_quota エキスパートモード)`, [XFS クォータの管理](#)

`su (mkfs.xfs のサブオプション)`, [XFS ファイルシステムの作成](#)

`sw (mkfs.xfs のサブオプション)`, [XFS ファイルシステムの作成](#)

`uquota/uqnoenforce`, [XFS クォータの管理](#)

`xfsdump`, [XFS ファイルシステムのバックアップと復元](#)

`xfsprogs`, [XFS ファイルシステムの一時停止](#)

`xfsrestore`, [XFS ファイルシステムのバックアップと復元](#)

`xfs_admin`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_bmap`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_copy`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_db`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_freeze`, [XFS ファイルシステムの一時停止](#)

`xfs_fsr`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_growfs`, [XFS ファイルシステムのサイズの拡大](#)

`xfs_info`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_mdrestore`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_metadump`, [XFS ファイルシステムのその他のユーティリティー](#)

`xfs_quota`, [XFS クォータの管理](#)

`xfs_repair`, [XFS ファイルシステムの修復](#)

インタラクティブな操作 (`xfsrestore`), [インタラクティブな操作](#)

エキスパートモード (`xfs_quota`), [XFS クォータの管理](#)

クォータの管理, [XFS クォータの管理](#)

シンプルモード (`xfsrestore`), [xfsrestore のシンプルモード](#)

ダンプのレベル, [XFS ファイルシステムのバックアップと復元](#)

ダーティーログを持つ XFS ファイルシステムの修復, [XFS ファイルシステムの修復](#)
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](#)
ファイルシステムのサイズの拡大, [XFS ファイルシステムのサイズの拡大](#)
ファイルシステムの修復, [XFS ファイルシステムの修復](#)
ファイルシステムタイプ, [XFS ファイルシステム](#)
プロジェクト制限 (の設定), [プロジェクト制限の設定](#)
マウント, [XFS ファイルシステムのマウント](#)
一時停止, [XFS ファイルシステムの一時的停止](#)
主な特長, [XFS ファイルシステム](#)
作成, [XFS ファイルシステムの作成](#)
割り当て機能, [XFS ファイルシステム](#)
書き込みバリア, [書き込みバリア](#)
累積モード (xfsrestore), [xfsrestore の累積モード](#)

xfsdump

XFS, [XFS ファイルシステムのバックアップと復元](#)

xfsprogs

XFS, [XFS ファイルシステムの一時的停止](#)

xfsrestore

XFS, [XFS ファイルシステムのバックアップと復元](#)

xfs_admin

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_bmap

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_copy

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_db

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_freeze

XFS, [XFS ファイルシステムの一時的停止](#)

xfs_fsr

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_growfs

XFS, [XFS ファイルシステムのサイズの拡大](#)

xfs_info

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_mdrestore

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_metadump

XFS, [XFS ファイルシステムのその他のユーティリティー](#)

xfs_quota

XFS, [XFS クォータの管理](#)

xfs_repair

XFS, [XFS ファイルシステムの修復](#)