Red Hat Enterprise Linux 6

ストレージ管理ガイド

Red Hat Enterprise Linux 6 における単一ノードストレージの導入と設定

Last Updated: 2021-09-13

Red Hat Enterprise Linux 6 ストレージ管理ガイド

Red Hat Enterprise Linux 6 における単一ノードストレージの導入と設定

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドでは、Red Hat Enterprise Linux 6 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。これは、Red Hat Enterprise Linux または Fedora に関する基本から中級の知識を持つシステム管理者向けのものです。

目次

前書き 9
1. ドキュメント規則 9
1.1. 誤字規則 9
1.2. 引用規則 10
1.3. 注記および警告 11
2. ヘルプの利用とフィードバック提供 11
2.1. ヘルプが必要ですか? 11
2.2. ご意見をお寄せください 12

第1章 概要 13
1.1. RED HAT ENTERPRISE LINUX 6 の新機能 13
ファイルシステムの暗号化（テクノロジープレビュー） 13
ファイルシステムキャッシング（テクノロジープレビュー） 13
Btrfs (テクノロジープレビュー) 13
I/O 制限処理 13
ext4 サポート 13
ネットワークブロックストレージ 14

パート I. ファイルシステム 15

第2章 ファイルシステム構造とメンテナンス 16
2.1. ファイルシステム階層標準 (FHS) の概要 16
2.1.1. FHS 組織 16
2.1.1.1. ファイルシステム情報の収集 16
2.1.1.2. /boot/ ディレクトリー 18
2.1.1.3. /dev/ ディレクトリー 18
2.1.1.4. /etc/ ディレクトリー 19
2.1.1.5. /lib/ ディレクトリー 19
2.1.1.6. /media/ ディレクトリー 19
2.1.1.7. /mnt/ ディレクトリー 19
2.1.1.8. /opt/ ディレクトリー 20
2.1.1.9. /proc/ ディレクトリー 20
2.1.1.10. /sbin/ Directory 20
2.1.1.11. /srv/ ディレクトリー 21
2.1.1.12. /sys/ ディレクトリー 21
2.1.1.13. /usr/ ディレクトリー 21
2.1.1.14. /var/ ディレクトリー 23
2.2. 特別な RED HAT ENTERPRISE LINUX ファイルの場所 24
2.3. /PROC 仮想ファイルシステム 25
2.4. 未使用ブロックの破棄 25

第3章 暗号化されたファイルシステム 27
3.1. ファイルシステムのマウント（暗号化済み） 27
3.2. 追加情報 28

第4章 BTRFS 29
Btrfs の機能 29

第5章 EXT3 ファイルシステム。 30
5.1. EXT3 ファイルシステムの作成 31
5.2. EXT3 ファイルシステムへの変換 31
5.3. EXT2 ファイルシステムへの復元 32

第6章 EXT4 ファイルシステム 34
6.1. EXT4 ファイルシステムの作成 35
6.2. EXT4 ファイルシステムのマウント 36
書き込みバリア 36
6.3. EXT4 ファイルシステムのサイズ変更 37
6.4. バックアップ EXT2/3/4 ファイルシステム 37
6.5. EXT2/3/4 ファイルシステムの復元 39
6.6. EXT4 ファイルシステムのその他のユーティリティー 40

第7章 GLOBAL FILE SYSTEM 2 42

第8章 XFS ファイルシステム 43
8.1. XFS ファイルシステムの作成 43
8.2. XFS ファイルシステムのマウント 44
書き込みバリア 45
8.3. XFS クォータ管理 45
プロジェクト制限の設定 47
8.4. XFS ファイルシステムのサイズの拡大 47
8.5. XFS ファイルシステムの修復 48
8.6. XFS ファイルシステムの一時停止 48
8.7. XFS ファイルシステムのバックアップおよび復元 49
xfsrestoreの単純なモード 50
xfsrestoreの累積モード 50
インタラクティブ操作 51
8.8. XFS ファイルシステムのその他のユーティリティー 51

第9章 NETWORK FILE SYSTEM (NFS) 53
9.1. NFS の仕組み 53
9.1.1. 必要なサービス 54
9.2. PNFS 55
9.3. NFS クライアント設定 56
9.3.1. /etc/fstab を使用した NFS ファイルシステムのマウント 57
9.4. AUTOFS 58
9.4.1. バージョン 4 と比較した autofs バージョン 5 の改善点 58
9.4.2. autofs の設定 59
9.4.3. サイト設定ファイルの上書きまたは拡張 61
9.4.4. LDAP を使用した自動マウント機能マップの格納 62
9.5. 一般的な NFS マウントオプション 64
9.6. NFS の起動および停止 65
9.7. NFS サーバーの設定 66
9.7.1. /etc/exports 設定ファイル 66
9.7.2. exportfs コマンド 69
9.7.2.1. NFSv4 で exportfs の使用 70
9.7.3. ファイアウォール背後での NFS の実行 70
9.7.3.1. NFS エクスポートの検出 71
9.7.4. ホスト名の形式 72
9.7.5. RDMA 経由の NFS 72
9.8. NFS のセキュア化 73
9.8.1. AUTH_SYS とエクスポート制御による NFS 保護 73
9.8.2. AUTH_GSSを使用した NFS セキュリティー 74
9.8.2.1. NFSv4 による NFS の保護 74
9.8.3. ファイル権限 75
9.9. NFS および RPCBIND 75
9.9.1. NFS と rpcbind のトラブルシューティング 75

9.10. リファレンス 76
インストールされているドキュメント 76
便利な Web サイト 77
関連書籍 77

第10章 FS-CACHE 78
10.1. パフォーマンスに関する保証 79
10.2. キャッシュの設定 79
10.3. NFS でのキャッシュの使用 80
10.3.1. キャッシュの共有 81
10.3.2. NFS でのキャッシュの制限 82
10.4. キャッシュカリング制限の設定 82
10.5. 統計情報 83
10.6. リファレンス 83

パート II. ストレージ管理 84

第11章 ストレージをインストールする際の注意点 85
11.1. インストール中のストレージ設定の更新 85
11.2. サポートされるファイルシステムの概要 85
11.3. 特に注意を要する事項について 86
/home、/opt、/usr/local には別々のパーティションを用意する 86
IBM System Z における DASD デバイスと zFCP デバイス 86
LUKS を使用したブロックデバイスの暗号化 87
古い BIOS RAID メタデータ 87
iSCSI の検出および設定 87
FCoE の検出および設定 87
DASD 87
DIF/DIX を有効にしているブロックデバイス 87

第12章 ファイルシステムの確認 89
12.1. FSCK のベストプラクティス 89
12.2. FSCK のファイルシステム固有の情報 90
12.2.1. ext2、ext3、および ext4 90
12.2.2. XFS 91
12.2.3. Btrfs 93

第13章 PARTITIONS 94
13.1. パーティションテーブルの表示 95
13.2. パーティションの作成 96
13.2.1. パーティションの作成 97
13.2.2. パーティションのフォーマットとラベル付け 98
13.2.3. /etc/fstab に追加 98
13.3. パーティションの削除 98
13.4. パーティションのサイズ変更 99

第14章 LVM (論理ボリュームマネージャー) 101
14.1. LVM2 とは 102
14.2. SYSTEM-CONFIG-LVM の使用 102
14.2.1. 初期化されていないエンタイトルメントの使用 105
14.2.2. ボリュームグループへの未割り当てのボリュームの追加 106
14.2.3. エクステントの移行 109
14.2.4. LVM を使用した新規ハードディスクの追加 111
14.2.5. 新規ボリュームグループの追加 112
14.2.6. ボリュームグループの拡張 114

14.2.7. 論理ボリュームの編集 115
14.3. LVM リファレンス 118
インストールされているドキュメント 118
便利な Web サイト 118

第15章 SWAP 領域 119
15.1. スワップ領域の追加 120
15.1.1. LVM2 論理ボリュームでのスワップ領域の拡張 120
15.1.2. スワップの LVM2 論理ボリュームの作成 121
15.1.3. スワップファイルの作成 121
15.2. スワップ領域の削除 122
15.2.1. LVM2 論理ボリュームでのスワップ領域の縮小 122
15.2.2. スワップの LVM2 論理ボリュームの削除 122
15.2.3. スワップファイルの削除 123
15.3. SWAP 領域の移動 123

第16章 ディスククォータ 124
16.1. ディスククォータの設定 124
16.1.1. クォータの有効化 124
16.1.2. ファイルシステムの再マウント 125
16.1.3. クォータデータベースファイルの作成 125
16.1.4. ユーザーごとのクォータ割り当て 126
16.1.5. グループごとのクォータ割り当て 127
16.1.6. ソフト制限の猶予期間の設定 128
16.2. ディスククォータの管理 128
16.2.1. 有効化と無効化 128
16.2.2. ディスククォータに関するレポート 128
16.2.3. クォータの精度維持 129
16.3. ディスククオータのリファレンス 130

第17章 RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS) 131
17.1. RAID のタイプ 131
ファームウェア RAID 131
ハードウェア RAID 131
ソフトウェア RAID 132
17.2. RAID レベルとリニアサポート 132
17.3. LINUX RAID サブシステム 134
Linux ハードウェア RAID のコントローラードライバー 134
mdraid 134
dmraid 134
17.4. インストーラーにおける RAID サポート 135
17.5. RAID セットの設定 135
mdadm 135
dmraid 135
17.6. RAID デバイスの詳細な作成 136

第18章 MOUNT コマンドの使い方 138
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 138
18.1.1. ファイルシステムタイプの指定 138
18.2. ファイルシステムのマウント 139
18.2.1. ファイルシステムタイプの指定 140
18.2.2. マウントオプションの指定 141
18.2.3. マウントの共有 142
18.2.4. マウントポイントの移動 146

18.3. ファイルシステムのアンマウント 146
18.4. MOUNT コマンドのリファレンス 147
18.4.1. man ページドキュメント 147
18.4.2. 便利な Web サイト 147

第19章 VOLUME_KEY 機能 148
19.1. コマンド 148
19.2. VOLUME_KEY を個別ユーザーとして使用する手順 149
19.3. 大規模な組織での VOLUME_KEY の使用 150
19.3.1. 暗号化キーを保存するための準備 150
19.3.2. 暗号化キーの保存 151
19.3.3. ボリュームへのアクセスの復元 151
19.3.4. 緊急パスフレーズの設定 152
19.4. VOLUME_KEY 参照 S 152

第20章 アクセス制御リスト 153
20.1. ファイルシステムのマウント 153
20.1.1. NFS 153
20.2. アクセス ACL の設定 153
20.3. デフォルト ACL の設定 155
20.4. ACL の取り込み 155
20.5. ACL が設定されているファイルシステムのアーカイブ作成 156
20.6. 旧システムとの互換性 157
20.7. ACL 参照情報 157

第21章 ソリッドステートディスクデプロイメントのガイドライン 158
21.1. デプロイメントに関する考慮事項 158
21.2. チューニングの留意事項 159
I/O スケジューラー 159
仮想メモリー 159
スワップ 159

第22章 書き込みバリア 160
22.1. WRITE BARRIERS の重要性 160
Write Barriers の仕組み 160
22.2. WRITE BARRIERS の有効化/無効化 160
22.3. アバージーの作成に関する留意事項 161
書き込みキャッシュの無効化 161
バッテリーバックグラウンド書き込みキャッシュ 161
high-End Arrays 162
NFS 162

第23章 ストレージ I/O アライメントとサイズ 163
23.1. ストレージアクセスのパラメーター 163
23.2. ユーザー空間アクセス 164
sysfs インターフェース 164
ブロックデバイス ioctls 164
23.3. STANDARDS（標準） 165
ATA 165
SCSI 165
23.4. スタッキング I/O パラメーター 165
23.5. 論理ボリュームマネージャー 166
23.6. パーティションおよびファイルシステムツール 166
util-linux-ng の libblkid および fdisk 166

parted および libparted 167
ファイルシステムツール 167

第24章 リモートディスクレスシステムの設定 168
24.1. ディスクレスクライアントの TFTP サービスの設定 168
24.2. ディスクレスクライアントの DHCP の設定 168
24.3. 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 169

第25章 デバイスマッパーマルチパスおよび仮想ストレージ 171
25.1. 仮想ストレージ 171
25.2. DM MULTIPATH 171

パート III. オンラインストレージ 173

第26章 ファイバーチャンネル 174
26.1. ファイバーチャネル API 174
26.2. ネイティブファイバーチャネルドライバーおよび機能 175

第27章 ISCSI ターゲットおよびイニシエーターの設定 176
27.1. ISCSI ターゲットの作成 176
27.2. ISCSI イニシエーターの作成 178

第28章 永続的な命名 180
28.1. WWID 180
28.2. UUID およびその他の永続識別子 181

第29章 ストレージデバイスの削除 183

第30章 ストレージデバイスへのパスの削除 185

第31章 ストレージデバイスまたはパスの追加 186

第32章 イーサネットインターフェース上でのファイバーチャネルの設定 188
32.1. FIBRE-CHANNEL OVER ETHERNET(FCOE)ターゲットのセットアップ 189

第33章 システムの起動時に FCOE インターフェースを自動マウントする設定 192

第34章 ストレージ INTERCONNECT のスキャン 194

第35章 ISCSI オフロードおよびインターフェースバインディングの設定 196
35.1. 利用可能な IFACE 設定の表示 196
35.2. ソフトウェア ISCSI のペースの設定 198
35.3. ISCSI オフロードの該当設定 198
35.4. ポータルへのペースのバインディング/選択解除 199

第36章 複数の LUN またはポータルを使用した ISCSI ターゲットのスキャン 201

第37章 オンライン論理ユニットのサイズ変更 203
37.1. ファイバーチャネル論理ユニットのサイズ変更 203
37.2. ISCSI 論理ユニットのサイズ変更 203
37.3. マルチパスデバイスのサイズの更新 204
37.4. オンライン論理ユニットの読み取りおよび書き込み状態の変更 205
37.4.1. 論理ユニットの再スキャン 206
37.4.2. マルチパスデバイスの R/W 状態の更新 206
37.4.3. 資料 207

第38章 論理ユニットの RESCAN-SCSI-BUS.SH の追加/削除 208
既知の問題(RESCAN-SCSI-BUS.SH) 208

第39章 リンクループ動作の変更 **209**
39.1. ファイバーチャンネル 209
39.2. DM-MULTIPATHを使用した ISCSI 設定 209
39.2.1. NOP-Out Interval/Timeout 210
SCSI エラーハンドラー 210
39.2.2. replacement_timeout 210
39.3. ISCSI ルート 211
特定のセッションのタイムアウトの設定 211

第40章 SCSI コマンドタイマーおよびデバイスステータスの制御 **213**
デバイスの状態 213
コマンドタイマー 213

第41章 オンラインストレージ設定のトラブルシューティング **214**

付録A 改訂履歴 **215**

前書き

1. ドキュメント規則

本ガイドでは、いくつかの規則を使用して特定の単語やフレーズを強調表示し、特定の情報への注意を促しています。

1.1. 誤字規則

特定の単語や句に注意を促すために 4 つの誤字規則を使用しています。これらの規則や、これらが適用される状況は以下のとおりです。

Mono-spaced Bold

シェルコマンド、ファイル名、パスなど、システム入力を強調表示するために使用されます。キーとキーの組み合わせを強調表示するためにも使用されます。以下に例を示します。

> 現在の作業ディレクトリーのファイル **my_next_bestselling_novel** の内容を表示するには、シェルプロンプトで **cat my_next_bestselling_novel** コマンドを入力し、**Enter** を押してコマンドを実行します。

上記には、ファイル名、シェルコマンドおよびキーが含まれます。これはすべて mono-spaced bold で示され、コンテキストにより区別可能なものになります。

キーの組み合わせは、キーの組み合わせの各パーツをつなげるプラス記号によって個別のキーと区別できます。以下に例を示します。

> **Enter** を押してコマンドを実行します。
>
> **Ctrl**+**Alt**+**F2** を押して、仮想端末に切り替えます。

最初の例では、押す特定のキーを強調表示しています。2 つ目の例は、同時に押す 3 つのキーのセットというキーの組み合わせを強調表示しています。

ソースコードについて記述では、クラス名、メソッド、関数、変数名、および段落内で記述された戻り値は、**mono-spaced bold**で示されます。以下に例を示します。

> ファイル関連クラスには、ファイルシステムの **filesystem**、ファイルの **file**、ディレクトリーの **dir** が含まれます。各クラスには、独自の関連付けられたパーミッションセットがあります。

Proportional Bold

これは、アプリケーション名、ダイアログボックステキスト、ラベルが付いたボタン、チェックボックスおよびラジオボタン、メニュータイトルおよびサブメニュータイトルなど、システムで発生した単語またはフレーズを示します。以下に例を示します。

> メインメニューバーから System → Preferences → Mouse を選択して、Mouse Preferences を起動します。**Buttons** タブで、**Left-handed mouse** チェックボックスを選択し、**Close** をクリックして、左から右に主要なマウスボタンを切り替えます (左側のマウスは適切なマウスになります)。
>
> 特殊文字を gedit ファイルに挿入するには、メインメニューバーから Applications → Accessories → Character Map を選択します。次に、Character Map メニューバーから Search → Find... を選択し、**Search** フィールドに文字の名前を入力して **Next** をクリックします。目的の文字は **Character Table** に強調表示されます。この強調表示し

た文字をダブルクリックして、**Text to copy** フィールドに配置し、**Copy** ボタンをクリックします。次に、ドキュメントに戻り、**gedit** メニューバーから **Edit → Paste** を選択します。

上記のテキストにはアプリケーション名、システム全体のメニュー名および項目、アプリケーション固有のメニュー名、GUI インターフェース内のボタンおよびテキストなどがあります。すべては proportional bold で示され、コンテキストと区別できます。

Mono-spaced Bold Italic または *Proportional Bold Italic*

mono-spaced bold または proportional bold のいずれでも、イタリックを追加すると、置換または変数テキストが表示されます。イタリックは、状況に応じて変化するテキストや、文字を入力しないテキストを表します。以下に例を示します。

ssh を使用してリモートマシンに接続するには、シェルプロンプトで **ssh *username@domain.name*** と入力します。リモートマシンが **example.com** で、そのマシンのユーザー名が john の場合は、ssh **john@example.com** を入力します。

mount -o remount *file-system* コマンドは、名前付きの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場合は、コマンドが **mount -o remount /home になります**。

現在インストールされているパッケージのバージョンを表示するには、**rpm -q *package*** コマンドを使用します。結果は以下のようになります： ***package-version-release***.

上記の太字のイタリック体の用語、username、domain.name、file-system、package、version、および release に注意してください。各単語はプレースホルダーで、コマンドの発行時に入力するテキストまたはシステムによって表示されるテキストのどちらかになります。

作業のタイトルを示す標準的な使用法のほかに、イタリックは新用語と重要な用語の最初の使用を示します。以下に例を示します。

Publican は *DocBook* 公開システムです。

1.2. 引用規則

端末の出力およびソースコードの一覧は、周りのテキストから視覚的に表示されます。

端末に送信される出力は **mono-spaced roman** にセットされて表現されます。

```
books       Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

ソースコードの一覧も **mono-spaced roman** にセットされますが、構文の強調表示が以下のように追加されます。

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                 struct kvm_assigned_pci_dev *assigned_dev)
{
        int r = 0;
        struct kvm_assigned_dev_kernel *match;

        mutex_lock(&kvm->lock);

        match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
```

```
                                assigned_dev->assigned_dev_id);
        if (!match) {
                printk(KERN_INFO "%s: device hasn't been assigned before, "
                  "so cannot be deassigned\n", __func__);
                r = -EINVAL;
                goto out;
        }

        kvm_deassign_device(kvm, match);

        kvm_free_assigned_device(kvm, match);

out:
        mutex_unlock(&kvm->lock);
        return r;
}
```

1.3. 注記および警告

最後に、3 つの視覚的スタイルを使用して、見落とす可能性のある情報に注意を促します。

注記

注記とは、タスクへのヒント、ショートカット、または代替アプローチです。注意を無視しても悪い結果を招くことはありませんが、便利なヒントを見逃してしまう可能性があります。

重要

見落としやすい詳細のある重要なボックス: 現行セッションにのみ適用される設定変更や、更新を適用する前に再起動が必要なサービスなどです。「Important」というラベルが付いたボックスを無視しても、データが失われることはありませんが、スムーズな操作が行えないことがあります。

警告

警告は無視すべきではありません。警告を無視すると、データが失われる可能性があります。

2. ヘルプの利用とフィードバック提供

2.1. ヘルプが必要ですか?

本ガイドで説明されている手順で問題が発生した場合は、Red Hat カスタマーポータル http://access.redhat.com にアクセスしてください。カスタマーポータルでは、以下を行うことができます。

- Red Hat 製品に関する技術サポート記事の知識ベースの検索または閲覧。

- Red Hat グローバルサポートサービス (GSS) へのサポートケースの送信。
- その他の製品ドキュメントへのアクセス。

Red Hat は、Red Hat のソフトウェアおよびテクノロジーについて、多くの電子メーリングリストも提供しています。一般に公開されているメーリングリストの一覧は、https://www.redhat.com/mailman/listinfoを参照してください。メーリングリストの名前をクリックして、その一覧をサブスクライブするか、またはメーリングリストのアーカイブにアクセスします。

2.2. ご意見をお寄せください

本ガイドで誤字脱字を発見されたり、このマニュアルを改善するための提案をお持ちの場合は、弊社までご連絡ください。Red_Hat_Enterprise_Linux 6 の製品に対して、http://bugzilla.redhat.com/ から Bugzilla レポートを送信してください。

バグレポートを送信する際には、マニュアル識別子(『doc-Storage_Admin_Guide』)を指定してください。

本ガイドを改善するためのご意見やご提案をお寄せいただく場合は、できるだけ具体的にご説明ください。エラーが見つかった場合は、簡単に確認できるように、セクション番号と前後のテキストを含めてください。

第1章 概要

『ストレージ管理ガイド』 には、Red Hat Enterprise Linux 6 でサポートされるファイルシステムおよびデータストレージ機能に関する広範な情報が含まれています。本書は、単一ノード (非クラスター化) ストレージソリューションを管理する管理者向けのクイックリファレンスになります。

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 つの部分に分割されます。

「ファイルシステム」のセクションでは、Red Hat Enterprise Linux 6 がサポートするさまざまなファイルシステムについて詳しく説明します。ファイルシステムについての説明のほか、それらを最大限に活用する方法についても説明します。

ストレージ管理の部分では、Red Hat Enterprise Linux 6 がサポートするさまざまなツールとストレージ管理タスクについて詳しく説明します。ファイルシステムについての説明のほか、それらを最大限に活用する方法についても説明します。

1.1. RED HAT ENTERPRISE LINUX 6 の新機能

Red Hat Enterprise Linux 6 におけるファイルシステムの機能拡張の特徴は、以下のとおりです。

ファイルシステムの暗号化（テクノロジープレビュー）

*eCryptfs*を使用して、マウント時にファイルシステムを暗号化できるようになりました。[1]、実際のファイルシステムに暗号化層を提供します。この「pseudo-file system」を使用すると、ファイル別およびファイル名の暗号化が可能となり、暗号化されたブロックデバイスよりも粒度の細かい暗号化が可能になります。ファイルシステムの暗号化の詳細については、3章*暗号化されたファイルシステム* を参照してください。

ファイルシステムキャッシング（テクノロジープレビュー）

*FS-Cache*の[1] により、ローカルストレージを使用して、ネットワーク経由で提供されるファイルシステムからデータをキャッシュすることができます（例： NFS を使用）。これは、ネットワークトラフィックを最小限に抑えますが、ネットワークを介したデータへのアクセスは高速化しません。FS-Cacheは、サーバー上のファイルシステムが、オーバーマウントされたファイルシステムを作成することなく、クライアントのローカルキャッシュと直接やりとりすることを可能にします。FS-Cache の詳細は、10章*FS-Cache* を参照してください。

Btrfs (テクノロジープレビュー)

Btrfs[1] は、現在利用可能なローカルファイルシステムです。これは、LVM 操作の統合を含む、パフォーマンスおよびスケーラビリティーを向上することにあります。Btrfs の詳細は 4章*Btrfs* を参照してください。

I/O 制限処理

Linux I/O スタックは、それを提供するデバイスの I/O 制限情報を処理できるようになりました。これにより、ストレージ管理ツールが一部のデバイスに I/O を最適化できるようになります。詳細は、23章*ストレージ I/O アライメントとサイズ* を参照してください。

ext4 サポート

本リリースでは、ext4 ファイルシステムに完全対応しています。Red Hat Enterprise Linux 6 のデフォルトのファイルシステムは、無制限のサブディレクトリーをサポートするようになりました。また、よ

り粒度の細かいタイムスタンプ、拡張属性のサポート、およびクォータジャーナリングも備えています。ext4 の詳細は、6章*Ext4 ファイルシステム* を参照してください。

ネットワークブロックストレージ

Fibre-channel over Ethernet に対応するようになりました。これにより、fibre-channel インターフェースは、fibre-channel プロトコルを維持しながら 10 Gigabit Ethernet ネットワークを使用できます。この設定方法については、32章*イーサネットインターフェース上でのファイバーチャネルの設定* を参照してください。

---

[1] この機能は、本リリースでは **テクノロジープレビューとして提供** されています。テクノロジープレビュー機能は、現在 Red Hat Enterprise Linux サブスクリプションサービスではサポートされていません。機能的に完全ではないことがあるため、通常は実稼働環境での使用には適していません。ただし、これらの機能はお客様の利便性として含まれ、より優れた公開機能を提供します。

また、完全にサポートされる前に、テクノロジープレビュー機能に関するフィードバックおよび機能についてのご提案をお寄せください。重大度の高いセキュリティー問題に対するエラータが提供されます。

パート I. ファイルシステム

File Systems セクションでは、ファイルシステムの構造と、eCryptfs および Btrfs の 2 つのテクノロジープレビューを説明します。この後、ファイルシステムは ext3、ext4、グローバルファイルシステム 2、XFS、NFS、および FS-Cache に完全に対応しています。

第2章 ファイルシステム構造とメンテナンス

ファイルシステムの構造は、オペレーティングシステムの組織の最も基本的なレベルです。オペレーティングシステムがユーザー、アプリケーション、およびセキュリティーモデルと対話する方法は、オペレーティングシステムがストレージデバイスでファイルを編成する方法にほぼ常に依存します。一般的なファイルシステムの構造を提供することで、ユーザーとプログラムがファイルにアクセスして書き込むことができます。

ファイルシステムは、ファイルを 2 つの論理カテゴリーに分類します。

- 共有可能ファイルと、共有できないファイル
- 変数と静的ファイル

共有可能な ファイルは、ローカルおよびリモートホストからアクセスできます。共有 *不可の* ファイルはローカルでのみ利用可能です。ログファイル *などの* 変数ファイルはいつでも変更できます。バイナリーなどの *静的ファイル* は、システム管理者からのアクションなしに変更しないでください。

この方法でファイルを分類すると、各ファイルの機能と、それらを保持するディレクトリーに割り当てられたパーミッションとを相互に関連付ける上で役立ちます。オペレーティングシステムとそのユーザーがファイルを操作する方法によって、ファイルが配置されるディレクトリー、そのディレクトリーが読み取り専用パーミッションまたは読み取りと書き込みのパーミッションでマウントされているかどうか、および各ユーザーがそのファイルにアクセスできるレベルが決まります。この組織のトップレベルは非常に重要です。基礎となるディレクトリーへのアクセスを制限できます。そうしないと、トップレベルから下のアクセスルールが厳密な構造に準拠していない場合に、セキュリティー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、*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やディレクトリーの名前、場所、およびパーミッションを定義します。

FHS ドキュメントは、すべての FHS 準拠のファイルシステムにとって信頼できるリファレンスですが、この標準では、多くの領域が未定義または拡張可能です。このセクションでは、標準の概要と、標準でカバーされていないファイルシステムの部分について説明します。

FHS コンプライアンスの 2 つの最も重要な要素は次のとおりです。

- 他の FHS 準拠システムとの互換性
- **/usr/** パーティションを読み取り専用としてマウントする機能。これは、**/usr/** には共通の実行ファイルが含まれており、ユーザーが変更できないため、特に重要です。さらに、**/usr/** は読み取り専用としてマウントされているため、CD-ROM ドライブまたは他のマシンから、読み取り専用 NFS マウント経由でマウントできる必要があります。

2.1.1. FHS 組織

ここで記載されているディレクトリーおよびファイルは、FHS ドキュメントで指定された小さなサブセットです。最も詳しい情報については、最新の FHS ドキュメントを参照してください http://www.pathname.com/fhs/。

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。

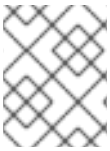
例2.1 df コマンドの出力

```
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
```

デフォルトでは、**df** はパーティションのサイズを 1 キロバイトブロック単位で表示し、使用中および利用可能なディスク領域の容量をキロバイト単位で表示します。メガバイトおよびギガバイトで情報を表示するには、**df -h** コマンドを使用します。**-h** 引数は「human-readable」の形式を表します。**df -h** の出力は以下のようになります。

例2.2 df -h コマンドの出力

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



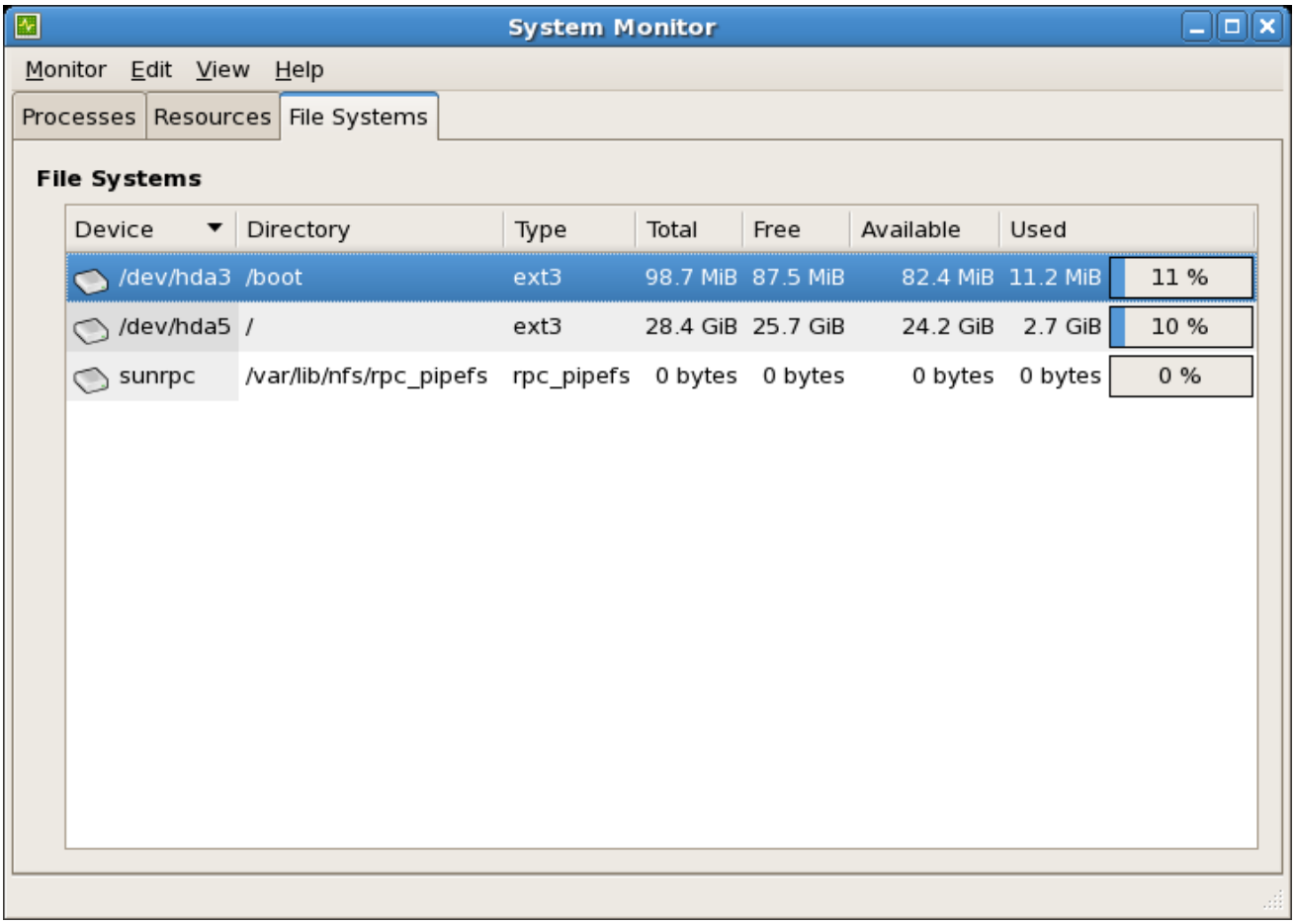
注記

上記の例では、マウントされたパーティション **/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が使用している推定領域を表示し、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。人間が判読可能な形式でディレクトリーの合計ディスク使用量のみを表示するには、du **-hs** を使用します。その他のオプションは、**man du を参照してください**。

グラフィカル形式で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表示するには、**Applications → System Tools → System Monitor** をクリックするか、**gnome-system-monitor** コマンドを使用して、Gnome **System Monitor** を使用します。**ファイルシステム** タブを選択して、システムのパーティションを表示します。以下の図は **File Systems** タブを示しています。

図2.1 GNOME システムモニターファイルシステムタブ



[D]

2.1.1.2. /boot/ ディレクトリー

/boot/ ディレクトリーには、Linux カーネルなど、システムの起動に必要な静的ファイルが含まれます。これらのファイルは、システムが正しく起動するためには不可欠です。

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、システムが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、以下のデバイス種別を表すデバイスノードが含まれます。

- システムに接続されているデバイス
- カーネルが提供する仮想デバイス

これらのデバイスノードは、システムが適切に機能するために不可欠です。**udevd** デーモンは、必要に応じて **/dev/** のデバイスノードを作成および削除します。

/dev/ ディレクトリーおよびサブディレクトリー内のデバイスは、**文字** (マウスやキーボードなどの入力と出力のシリアルストリームのみを提供) または**ブロック** (ハードドライブやフロッピードライブなどのランダムにアクセス可能) のいずれかとして定義されます。GNOME または KDE がインストールされている場合は、一部のストレージデバイスは (USBなどで) 接続時または (CDまたはDVDドライブなどの) 挿入時に自動的に検出され、内容を表示するポップアップウィンドウが表示されます。

表2.1 /dev ディレクトリーの一般的なファイルの例

ファイル	詳細
/dev/hda	プライマリー IDE チャネル上のマスターデバイス。
/dev/hdb	プライマリー IDE チャンネル上のスレーブデバイス。
/dev/tty0	最初の仮想コンソール。
/dev/tty1	2 番目の仮想コンソール
/dev/sda	プライマリー SCSI または SATA チャネル上の最初のデバイス。
/dev/lp0	最初の並列ポート。
/dev/ttyS0	シリアルポート。

2.1.1.4. /etc/ ディレクトリー

/etc/ ディレクトリーは、マシンのローカルとなる設定ファイル用に予約されています。バイナリーを含むことはできません。バイナリーは **/bin/ または /sbin/** に移動する必要があります。

たとえば、**/etc/skel/** ディレクトリーには、「スケルトン」ユーザーファイルが格納されています。これは、ユーザーの初回作成時にホームディレクトリーを設定するために使用されます。また、アプリケーションはこのディレクトリーに設定ファイルを保存し、実行時にそれらを参照する可能性があります。**/etc/exports** ファイルは、どのファイルシステムをリモートホストにエクスポートするかを制御します。

2.1.1.5. /lib/ ディレクトリー

/lib/ ディレクトリーには、**/ bin/ および /sbin/** でバイナリーの実行に必要なライブラリーのみが含まれている必要があります。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。

2.1.1.6. /media/ ディレクトリー

/media/ ディレクトリーには、USB ストレージメディア、DVD、CD-ROM 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。

2.1.1.7. /mnt/ ディレクトリー

/mnt/ ディレクトリーは、NFS ファイルシステムのマウントなどの、一時的にマウントされたファイルシステム用に予約されています。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。リムーバブルメディアは自動的に **/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は使用しないでください。

2.1.1.8. /opt/ ディレクトリー

/opt/ ディレクトリーは通常、デフォルトのインストールの一部ではないソフトウェアおよびアドオンパッケージ用に予約されています。**/opt/** にインストールするパッケージは、その名前を持つディレクトリーを作成します (例: **/opt/*packagename*/**)多くの場合、このようなパッケージは、予測可能なサブディレクトリー構造に従います。ほとんどの場合、バイナリーは **/opt/*packagename*/bin/** に保存され、**man** ページは **/opt/*packagename*/man/** に保存されます。

2.1.1.9. /proc/ ディレクトリー

/proc/ ディレクトリーには、カーネルから情報を抽出するか、カーネルに情報を送信する特別なファイルが含まれています。このような情報の例には、システムメモリー、CPU 情報、およびハードウェア設定が含まれます。**/proc/** の詳細は、「/proc 仮想ファイルシステム」 を参照してください。

2.1.1.10. /sbin/ Directory

/sbin/ ディレクトリーには、システムの起動、復元、復旧、または修復に不可欠なバイナリーが格納されています。**/sbin/** のバイナリーを使用するには、root 権限が必要です。また、**/sbin/** ディレクトリーがマウントされる **前に** システムが使用するバイナリーが含まれています。**/usr/** がマウントされた後に使用されるシステムユーティリティーが、通常は **/usr/ sbin/** に配置されます。

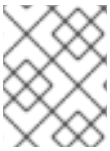
以下のプログラムは、少なくとも **/sbin/** に保存する必要があります。

- **arp**
- **クロック**
- **halt**
- **init**
- **fsck.***
- **grub**
- **ifconfig**
- **mingetty**
- **mkfs.***
- **mkswap**
- **reboot**
- **route**
- **shutdown**

- **swapoff**
- **swapon**

2.1.1.11. /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固有のデータが含まれます。このディレクトリーは、FTP、WWW、または CVS などの特定サービスのデータファイルの場所をユーザーに提供します。特定のユーザーにのみ関連するデータは、**/home/** ディレクトリー内になければなりません。



注記

デフォルトの httpd インストールは、提供されるコンテンツに **/var/www/html** を使用します。

2.1.1.12. /sys/ ディレクトリー

/sys/ ディレクトリーは、2.6 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。2.6 カーネルのホットプラグハードウェアデバイスのサポートが増えると、**/sys/** ディレクトリーには、**/proc/** が保持する情報と同様の情報が含まれますが、ホットプラグデバイスに固有のデバイス情報の階層ビューが表示されます。

2.1.1.13. /usr/ ディレクトリー

/usr/ ディレクトリーは、複数のマシンにまたがって共有できるファイル用です。多くの場合、**/usr/** ディレクトリーは独自のパーティションにあり、読み取り専用でマウントされます。**/usr/** ディレクトリーには、通常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に使用されます。

/usr/games

このディレクトリにはゲームが保存されています。

/usr/include

このディレクトリーは C ヘッダーファイルに使用されます。

/usr/kerberos

このディレクトリーは、Kerberos 関連のバイナリーおよびファイルに使用されます。

/usr/lib

このディレクトリーは、シェルスクリプトまたはユーザーが直接使用するように設計されていないオブジェクトファイルやライブラリーに使用されます。このディレクトリーは 32 ビットシステム用です。

/usr/lib64

このディレクトリーは、シェルスクリプトまたはユーザーが直接使用するように設計されていないオブジェクトファイルやライブラリーに使用されます。このディレクトリーは、64 ビットシステム用です。

/usr/libexec

このディレクトリーには、他のプログラムによって呼び出される小さなヘルパープログラムが含まれます。

/usr/sbin

このディレクトリーには、**/sbin/ に属さないシステム管理バイナリーが格納されます**。

/usr/share

このディレクトリーには、アーキテクチャー固有ではないファイルを保存します。

/usr/src

このディレクトリーには、ソースコードが保存されます。

/var/tmp にリンクされた /usr/tmp

このディレクトリーには、一時ファイルが保存されます。

/usr/ ディレクトリーには **/local/** サブディレクトリーも含まれる必要があります。FHS によると、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るので、システムの更新中に上書きされないようにする必要があります。**/usr/local** ディレクトリーの構造は **/usr/** と似ており、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 の **/usr/local/** の使用は、FHS と若干異なります。FHS は、 **/usr/local/** を使用して、システムソフトウェアをアップグレードする心配がないままのソフトウェアを保存するために使用する必要があることを記しています。**RPM Package Manager** 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/** に保存してファイルを保護する必要はありません。

代わりに、Red Hat Enterprise Linux はマシンにローカルなソフトウェアに **/usr/local/** を使用します。たとえば、**/usr/** ディレクトリーがリモートホストから読み取り専用の NFS 共有としてマウントされている場合でも、**/usr/local/** ディレクトリーの下にパッケージまたはプログラムをインストールすることができます。

2.1.1.14. /var/ ディレクトリー

FHS では、Linux が **/usr/** を読み取り専用としてマウントする必要があるため、ログファイルを書き込むプログラムや、**spool/** または **lock/** ディレクトリーが必要なプログラムは、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 によると、**/var/** は、spool ディレクトリーおよびファイル、ロギングデータ、一時的ファイルを含む変数データ用となります。

以下は、システムにインストールされている内容に応じて、**/var/** ディレクトリー内にあるディレクトリーの一部です。

- **/var/account/**
- **/var/arpwatch/**
- **/var/cache/**
- **/var/crash/**
- **/var/db/**
- **/var/empty/**
- **/var/ftp/**
- **/var/gdm/**
- **/var/kerberos/**
- **/var/lib/**
- **/var/local/**
- **/var/lock/**
- **/var/log/**
- **/var/spool/mail/** にリンクされた **/var/mail**
- **/var/mailman/**
- **/var/named/**
- **/var/nis/**
- **/var/opt/**
- **/var/preserve/**
- **/var/run/**
- **/var/spool/**
- **/var/tmp/**
- **/var/tux/**
- **/var/www/**

- **/var/yp/**

messages および **lastlog** などのシステムログファイルは、**/var/log/** ディレクトリーに移動します。**/var/lib/rpm/** ディレクトリーには、RPM システムデータベースが含まれます。ロックファイルは、通常はファイルを使用するプログラムのディレクトリーにある **/var/lock/** ディレクトリーに移動します。**/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格納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. 特別な RED HAT ENTERPRISE LINUX ファイルの場所

Red Hat Enterprise Linux は、特別なファイルに対応するために FHS 構造を若干拡張しています。

RPM に関連するほとんどのファイルは、**/var/lib/rpm/** ディレクトリーに保持されます。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 **Package Updater** が使用するファイルが含まれます。この場所は、システムの更新中にダウンロードされた RPM を一時的に保

存するためにも使用できます。Red Hat Network の詳細は、オンラインで https://rhn.redhat.com/ のドキュメントを参照してください。

Red Hat Enterprise Linux に固有の別の場所は **/etc/sysconfig/** ディレクトリーです。このディレクトリーには、さまざまな設定情報が格納されています。システムの起動時に実行されるスクリプトの多くは、このディレクトリー内の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 *仮想ファイルが含まれます*。したがって、**/proc** は通常仮想ファイルシステムと呼ばれます。これらの仮想ファイルは、大量の情報が含まれている場合でも、サイズは通常 0 バイトになります。

/proc ファイルシステムはストレージには使用されません。この主な目的は、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。リアルタイム情報は、対応する **/proc** ファイルを表示することで、多くのシステムコンポーネントで取得できます。**/proc** 内のファイルの一部は、カーネルを設定するために (ユーザーとアプリケーションの両方で) 操作することもできます。

以下の **/proc** ファイルは、システムストレージの管理および監視に関連しています。

/proc/devices
: 現在設定されているさまざまな文字およびブロックデバイスを表示します。

/proc/filesystems
: カーネルで現在対応しているすべてのファイルシステム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または RAID 設定に関する現在の情報が含まれます (存在する場合)。

/proc/mounts
: システムで現在使用しているマウントをすべて一覧表示します。

/proc/partitions
: パーティションブロック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 6 『デプロイメントガイド』 を参照してください。

2.4. 未使用ブロックの破棄

バッチ破棄とオンライン破棄操作は、ファイルシステムで使用されていないブロックを破棄するマウントされたファイルシステムの機能です。これは、ソリッドステートドライブおよび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。

バッチ破棄操作 は、ユーザーが **fstrim** コマンドを使用して明示的に実行します。このコマンドは、ファイルシステム内にある未使用のブロックで、管理者が指定した基準に一致するものをすべて破棄します。いずれの操作タイプも、ext4 ファイルシステムを Red Hat Enterprise Linux 6.2 以降として使用する場合にサポートされます。そのため、ファイルシステムの基礎となるブロックデバイスが物理的な

破棄操作に対応している限りです。これは、Red Hat Enterprise Linux 6.4 以降から XFS ファイルシステムにも該当します。**/sys/block/*デバイス*/queue/discard_max_bytes** の値がゼロではない場合は、物理的な破棄操作に対応しています。

オンライン破棄操作 は、マウント時に **-o discard** オプション（ **/etc/fstab** または **mount** コマンドの一部として）指定され、ユーザーの介入なしでリアルタイムで実行します。オンライン破棄操作は、使用中から空きに移行しているブロックのみを破棄します。オンライン破棄操作は、Red Hat Enterprise Linux 6.2 以降である ext4 ファイルシステム、および Red Hat Enterprise Linux 6.4 以降である 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を実行できない場合、またはパフォーマンスを維持するためにオンライン破棄操作が必要な場合を除いて、Red Hat はバッチ破棄操作を推奨します。

第3章 暗号化されたファイルシステム

Red Hat Enterprise Linux 6 は、*eCryptfs* のテクノロジープレビュー機能である「pseudo-file system」を提供します。これは、ファイルごとにデータおよびファイル名の暗号化を提供します。「pseudo-file system」という用語は、eCryptfs にオンディスクフォーマットがないため、実際のファイルシステム上にあるファイルシステム層です。eCryptfs レイヤーは暗号化機能を提供します。

ecryptfs は、基礎となる（暗号化）ファイルシステムに書き込むファイル操作を傍受することで、バインドマウントのように機能します。eCryptfs レイヤーは、基礎となるファイルシステムのファイルのメタデータにヘッダーを追加します。このメタデータは、このファイルの暗号化と、暗号化されたファイルシステムに渡される前にファイルデータを暗号化します。必要に応じて、eCryptfs もファイル名を暗号化できます。

ecryptfs はオンディスクファイルシステムではありません。したがって、**mkfs** などのツールで作成する必要はありません。代わりに、eCryptfs は特別な mount コマンドを実行して開始します。eCryptfs で保護されたファイルシステムを管理するには、最初に **ecryptfs-utils** パッケージをインストールする必要があります。

3.1. ファイルシステムのマウント（暗号化済み）

eCryptfs でファイルシステムを暗号化するには、以下のコマンドを実行します。

```
# mount -t ecryptfs /source /destination
```

*eCryptfs でディレクトリー階層（上記の例では/**source** ）を暗号化すると、これが eCryptfs（上記の例では/**destination** ）で暗号化されるマウントポイントにマウントします。/**destination** へのすべてのファイル操作は、基礎となる /**source ファイルシステムに対して** 暗号化されます。ただし、ファイル操作は eCryptfs レイヤーを通過せずに /**source** を直接変更する可能性があります。これにより、不整合になる可能性があります。*

*Red Hat では、ほとんどの環境ではこのような理由から、/**source** と / **destination** の両方の名前を同じようにすることを推奨します。以下に例を示します。*

```
# mount -t ecryptfs /home /home
```

*実質的には、ファイルシステムを暗号化することと、ファイルシステム **自体** にマウントすることを意味します。これを行うと、すべてのファイル操作を /home **に対して** /**home** が eCryptfs レイヤーを通過するのに役立ちます。*

*マウントおよび暗号化のプロセスで、**マウント** により、以下の設定が可能になります。*

暗号化キータイプ

***openssl**、**tspi**、または **passphrase**。**passphrase** を選択すると、**マウント** が要求されます。*

暗号

***aes**、**blowfish**、**des3_ede**、**cast6**、または **cast5**。*

キー bytesize

***16**、**32**、または **24**。*

plaintext passthrough

有効または無効化。

filename encryption

有効または無効化。

*対話型マウントの最後のステップの後に、**マウント** が行われ、マウントの実行がすべて表示されます。この出力は、選択した各設定と同等のコマンドラインオプションで構成されます。たとえば、**passphrase のキー** タイプの **aes** 暗号、**16** の鍵バイトサイズ **plaintext passthrough** を無効とすると、出力は以下のようになります。**filename encryption***

```
Attempting to mount with the following options:
  ecryptfs_unlink_sigs
  ecryptfs_key_bytes=16
  ecryptfs_cipher=aes
  ecryptfs_sig=c7fed37c0a341e19
Mounted eCryptfs
```

*このディスプレイのオプションは直接コマンドラインに渡して、同じ設定を使用してファイルシステムを暗号化およびマウントすることができます。これを行うには、mount の **-o** オプションへの引数として各 **オプションを使用します**。以下に例を示します。*

```
# mount -t ecryptfs /home /home -o ecryptfs_unlink_sigs \
 ecryptfs_key_bytes=16 ecryptfs_cipher=aes ecryptfs_sig=c7fed37c0a341e19[2]
```

3.2. 追加情報

*eCryptfs およびそのマウントオプションについての詳細は、**man ecryptfs（ecryptfs -utils** パッケージで提供される）を参照してください。以下のカーネルドキュメント（**kernel-doc パッケージで提供される**）も、eCryptfs に関する追加情報も提供します。*

/usr/share/doc/kernel-doc-version/Documentation/filesystems/ecryptfs.txt

---

[2] This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (\) — should be treated as one command, sans backslashes.

第4章 BTRFS

*B-tree ファイルシステム(**Btrfs**)は、パフォーマンスとスケーラビリティーを向上させるローカルシステムです。Btrfs は、Red Hat Enterprise Linux 6 でテクノロジープレビューとして、AMD64 および Intel 64 のアーキテクチャーに導入されました。Btrfs テクノロジープレビューは Red Hat Enterprise Linux 6.6 で終了し、今後更新されることはありません。Btrfs は Red Hat Enterprise Linux 6 の将来のリリースに含まれますが、いかなる方法でもサポートは提供されません。*

Btrfs の機能

システム管理者の管理を容易にするために、Btrfs にいくつかのユーティリティーが組み込まれています。これらには以下が含まれます。

組み込みシステムのロールバック

ファイルシステムのスナップショットを使用すると、問題が発生した場合にシステムを以前の状態にロールバックすることができます。

組み込み圧縮

これにより、領域の節約が容易になります。

チェックサム機能

これにより、エラー検出が改善されます。

特定の機能には、以下のような統合 LVM 操作が含まれます。

- *新規ストレージデバイスの動的、オンラインの追加、または削除*
- *コンポーネントデバイス全体での RAID の内部サポート*
- *メタデータまたはユーザーデータに異なる RAID レベルを使用できる機能*
- *すべてのメタデータおよびユーザーデータの完全なチェックサム機能。*

第5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、以下のような利点が提供されます。

可用性

予期しない停電やシステムクラッシュ (クリーンでないシステムシャットダウン とも言われる) 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。

ライブファイルシステムで **fsck -n** を実行できます。ただし、変更は加えられず、部分的に書き込まれたメタデータが発生した場合に、誤解を招く結果が表示される可能性があります。

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、以前の破損がないため、ファイルシステムが一貫した状態になります。**fsck -n** を実行できるようになりました。

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ではなく、一貫性を維持するために使用される ジャーナル のサイズに依存します。デフォルトのジャーナルサイズは、ハードウェアの速度に応じて、復旧するのに約1 秒かかります

注記

Red Hat で対応している 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

一部のデータを複数回書き込みますが、ext3 のジャーナリングにより、ハードドライブのヘッドモーションが最適化されるため、ほとんどの場合、ext3 のスループットは ext2 よりも高くなります。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。

簡単なトランジション

ext2 から ext3 に簡単に移行でき、再フォーマットをせずに、堅牢なジャーナリングファイルシステムの恩恵を受けることができます。このタスクの実行方法は、「Ext3 ファイルシステムへの変換」を参照してください。

Red Hat Enterprise Linux 6 バージョンの ext3 では、以下の更新が行われています。

デフォルトの Inode サイズ変更

ディスク上の inode のデフォルトのサイズは、拡張属性（ACL、SELinux 属性など）のより効率的なストレージに増えました。この変更により、特定のサイズのファイルシステム上に作成されたデフォルトの inode の数が減少しました。inode サイズは、ske **2fs -I** オプションで選択することも、**/etc/mke2fs.conf** で選択したりして、システム全体のデフォルト for **mke2fs** を設定できます。

注記

ext3 ファイルシステムを維持する意図のある Red Hat Enterprise Linux 6 にアップグレードすると、ファイルシステムを再行う必要はありません。

新規のマウントオプション： data_err

新しいマウントオプションが追加されました： **data_err=abort**このオプションは、data **=ordered** モードで（メタデータではなく）ファイルデータでエラーが発生した場合に、ジャーナルを中断するように ext3 に指示します。このオプションはデフォルトで無効にされています（**data_err=ignore**に設定されています）。

より効率的なストレージの使用

ファイルシステム(**mkfs**)を作成すると、**mke2fs** は「discard」または「trim」ブロックを、ファイルシステムのメタデータで使用されていないブロックを試みます。これは、SSD またはシンプロビジョニングされたストレージを最適化するのに役立ちます。この動作を非表示にするには、withke **2fs -K** オプションを使用します。

以下のセクションでは、ext3 パーティションの作成およびチューニングについて説明します。ext2 パーティションの場合は、以下のパーティショニングとフォーマットセクションを省略し、「Ext3 ファイルシステムへの変換」 に直接移動します。

5.1. EXT3 ファイルシステムの作成

インストール後、ext3 ファイルシステムを新たに作成しないといけない場合があります。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次のとおりです。

手順5.1 ext3 ファイルシステムを作成する

1. **mkfs** で、ext3 ファイルシステムでパーティション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。

5.2. 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

Red Hat Enterprise Linux のデフォルトインストールは、すべてのファイルシステムに ext4 を使用します。ただし、ext2 を ext3 に変換するには、**tune2fs** を使用する前後に、常に **e2fsck** ユーティリティーを使用してファイルシステムを確認してください。ext2 から ext3 への変換を試みる前に、エラーが発生した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てください。

さらに、Red Hat は、可能な限り 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、データを移行することを推奨します。

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、ターミナルに以下のコマンドを入力します。

```
# tune2fs -j block_device
```

block_device には、変換する ext2 ファイルシステムが含まれます。

有効なブロックデバイスは、以下の 2 種類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の論理ボリューム (例: **/dev/mapper/VolGroup00-LogVol02**)。

静的デバイス

たとえば、**/dev/sdbX** (sdb はストレージデバイス名で X はパーティション番号) などの従来のストレージボリューム。

df コマンドを実行して、マウントされたファイルシステムを表示します。

5.3. EXT2 ファイルシステムへの復元

ext2 ファイルシステムに戻すには、以下の手順に従います。

分かりやすくするため、本セクションのコマンド例は、ブロックデバイスに以下の値を使用します。

/dev/mapper/VolGroup00-LogVol02

手順5.2 ext3 から ext2 に戻す

1. root としてログインし、パーティションをアンマウントして以下を入力します。

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

2. 以下のコマンドを入力して、ファイルシステムの種類を ext2 に変更します。

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

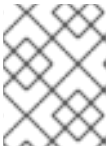
3. 以下のコマンドを入力して、パーティションでエラーの有無を確認します。

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

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

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

上記のコマンドでは、/mount/point を、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

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

パーティションを ext2 に永続的に変更するには、**/etc/fstab** ファイルを更新することを忘れないでください。そうしないと、起動後に元に戻されます。

第6章 EXT4 ファイルシステム

ext4 ファイルシステムは、ext3 ファイルシステムの拡張であり、Red Hat Enterprise Linux 5 のデフォルトのファイルシステムです。Ext4 は、Red Hat Enterprise Linux 6 のデフォルトのファイルシステムで、サイズが 16 テラバイトまでのファイルおよびファイルシステムをサポートできます。また、サブディレクトリーの数を無制限にサポートします (ext3 ファイルシステムは最大 32,000 までしかサポートしません)。ただし、リンク数が 65,000 を超えると 1 にリセットされ、増加しなくなります。

注記

ext3 と同様に、**fsck** を実行するには、ext4 ボリュームをアンマウントする必要があります。詳細は、5章Ext3 ファイルシステム。 を参照してください。

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への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()** を使用して確認しないというプログラムのバグが表面化しませんでした。一方、ext4 ファイルシステムは、ディスクへの変更書き込みの前に数秒間待機することが多く、書き込みを結合して再度順序付けを行うことにより、ext3 を上回るディスクパフォーマンスを実現しています。

警告

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

Ext4 のその他の機能

ext4 ファイルシステムでは次の機能にも対応しています。

- 拡張属性 (**xattr**) - これにより、システムは、ファイルごとに、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。
- クォータジャーナリング - 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。

注記

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

- サブセカンド (一秒未満) 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。

6.1. 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場合は、**mkfs.ext4** コマンドを使用します。一般的な用途では、デフォルトのオプションが最適です。

```
# 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6.1 mkfs.ext4 コマンドの出力

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

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

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

ストライプ化されたブロックデバイス (RAID5 アレイなど) の場合は、ファイルシステムの作成時にストライプジオメトリーを指定できます。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 の中にも、こうした最適な配列を選択するものがあります。*

*ストライプジオメトリーを指定するには、以下のサブオプションとともに **mkfs.ext4** の **-E** オプション (拡張ファイルシステムオプション) を使用します。*

stride=value

RAID チャンクサイズを指定します。

stripe-width=value

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。

*両方のサブオプションの場合、**value** 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 ブロックのファイルシステムで、64k ストライド (16 x 4096) のファイルシステムを作成する場合は、次のコマンドを使用します。*

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

*ファイルシステムの作成方法は、**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 ext4 機能の一部を有効にし、ext4 ドライバーを使用して ext3 ファイルシステムをマウントすることができます。ただし、これらのアクションは Red Hat Enterprise Linux 6 で完全にテストされていないため、**サポートされません**。このため、Red Hat は、この方法で変換またはマウントした ext3 ファイルシステムのパフォーマンスと予測可能な動作を保証できません。*

6.2. EXT4 ファイルシステムのマウント

ext4 ファイルシステムは、追加のオプションなしでマウントできます。以下に例を示します。

```
# 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
```

***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。詳細は、**man tune2fs** を参照してください。*

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** オプションを使用し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、22章書き込みバリア を参照してください。

6.3. 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選択してください。

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 node
```

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。以下のサフィックスは、特定の単位を示しています。

- **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任意です (多くの場合は必要ありません)。**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ーの全領域に渡って自動的に拡張を行います。

ext4 ファイルシステムのサイズ変更に関する詳細は、**man resize2fs** を参照してください。

6.4. バックアップEXT2/3/4 ファイルシステム

手順6.1 バックアップ ext2/3/4 ファイルシステムの例

1. 復元操作を行う前に、すべてのデータをバックアップする必要があります。データバックアップは定期的に行う必要があります。データの他に、**/etc/fstab** や **fdisk -l** の出力など、保存すべき設定情報があります。sosreport/sysreport を実行すると、この情報を取得し、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** パーティションを使用してバックアップファイルを保存します。また、**/dev/sda6** が **/backup-files にマウントされていることを前提とします。***

2. *バックアップされる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 Single User モードに起動してください。通常のデータパーティションには、このステップは必要ありません。*

3. ***ダンプ** 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

- *システムが長時間実行中の場合は、バックアップの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。*
- ***ダンプは**、破損したバージョンのファイルをバックアップする可能性があるため、負荷の高いファイルシステムやマウントされたファイルシステムでは使用しないでください。この問題は以下に記載されています dump.sourceforge.net。*

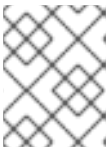
重要

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。

通常のデータパーティションがマウントされている間に、通常のデータパーティションを作成することは可能ですが、可能な場合はこれをアンマウント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測できなく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
```

リモートバックアップを実行する場合は、ssh の両方を使用するか、パスワード以外のログインを設定できます。



注記

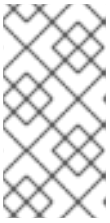
標準のリダイレクトを使用する場合は、「-f」オプションを別々に渡す必要があります。

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

6.5. EXT2/3/4 ファイルシステムの復元

手順6.2 ext2/3/4 ファイルシステムの例の復元

1. オペレーティングシステムパーティションで復元を行っている場合は、レスキューモードにシステムを起動します。このステップは、通常のデータパーティションには不要です。
2. **fdisk** コマンドを使用して、sda1/sda2/sda3/sda4/sda5 を再構築します。



注記

必要な場合は、復元されたファイルシステムを含むパーティションを作成します。新しいパーティションは、復元したデータを含めるのに十分な大きさである必要があります。開始番号と終了番号を常に取得することが重要になります。これらは、パーティションの開始点と終了セクター番号になります。

3. 以下に示すように **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

バックアップファイルを保存するため、上記の例では **/dev/sda6** 形式ではありません。

```
# 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、すべてのパーティションに **fstab** ファイルに一致するようにラベルを付けます。パーティションが再作成されていない場合は、この手順は必要ありません。

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

5. 作業ディレクトリーを準備します。

```
# 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したり、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 のいずれかを使用できます。以下の例に、パスワードを使用しない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。

6.6. EXT4 ファイルシステムのその他のユーティリティー

Red Hat Enterprise Linux 6 には、他にも ext4 ファイルシステム管理ユーティリティーがあります。

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** の使用に関する詳細は、**man quota** および「ディスククォータの設定」 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、または ext4 の重要なファイルシステムメタデータをファイルに保存します。

*これらのユーティリティーの詳細は、それぞれの **man** ページを参照してください。*

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 は分散メタデータと複数のジャーナルを採用します。

GFS2 は、理論的には 8 エクサバイトのファイルシステムに対応できる 64 ビットアーキテクチャーに基づいています。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 TB です。システムで 100 TB を超える GFS2 ファイルシステムが必要な場合は、Red Hat サービスの担当者にお問い合わせください。

ファイルシステムのサイズを決定する際には、その復旧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 **fsck** コマンドを実行すると、長い時間がかかり、大量のメモリーを消費する可能性があります。また、ディスクまたはディスクサブシステムに障害が発生した場合、復旧時間は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 ノードは Red Hat Cluster Suite 設定および管理ツールを使用して、設定および管理できます。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、異なるノードにあるプロセスで GFS2 ファイルを共有できるようになります。これは、同じノードにあるプロセスでローカルファイルシステム上のファイル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 に関する情報は、『Red Hat クラスターの『管理ガイド』』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論理ボリュームの管理が可能となり、クラスター内のすべてのノードで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の詳細は、Red Hat の『論理ボリュームマネージャーの管理』ガイドを参照してください。

gfs2.ko カーネルモジュールは GFS2 ファイルシステムを実装しており、GFS2 クラスターノードにロードされます。

クラスター化ストレージおよび非クラスター化ストレージでの GFS2 ファイルシステムの作成および設定に関する包括的な情報は、Red Hat の『グローバルファイルシステム 2』ガイドを参照してください。

第8章 XFS ファイルシステム

XFS は、元々は Silicon Graphics, Inc で設計された非常に拡張性の高い、高性能のファイルシステムです。これは、非常に大きなファイルシステム（最大 16 個）、ファイル（8 メガバイト）、およびディレクトリー構造（数百万のエントリー）をサポートするために作成されました。

主な特長

XFS は、メタデータジャーナリング をサポートしているため、クラッシュリカバリーを迅速に行うことができます。XFS ファイルシステムは、マウントされ、アクティブな状態でデフラグし、拡張できます。また、Red Hat Enterprise Linux 6 は XFS 固有のバックアップおよび復元ユーティリティーをサポートしています。

割り当て機能

XFS は、次の割り当てスキームを特長としています。

- 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

遅延割り当てや他のパフォーマンスの最適化は、ext4 と同じように XFS に影響を与えます。つまり、XFS ファイルシステムへのプログラムの書き込みは、プログラムが後で **fsync()** 呼び出しを発行しない限り、ディスク上にあることが保証されません。

ファイルシステムで遅延割り当ての影響についての詳細は、6章Ext4 ファイルシステム で 『割り当て機能』 を参照してください。XFS にディスクへの書き込みも適用できるようにする回避策です。

その他の XFS 機能

XFS ファイルシステムは、以下もサポートしています。

拡張属性 (xattr)

これにより、システムが、ファイルごとに、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。

クォータジャーナリング

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できます。

サブセカンド (一秒未満) のタイムスタンプ

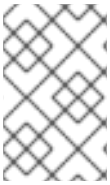
これにより、タイムスタンプはサブセカンド (一秒未満) になることができます。

8.1. 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場合は、**mkfs.xfs /dev/device** コマンドを使用します。通常、デフォルトのオプションは、一般的な使用に最適なものです。

既存のファイルシステムを含むブロックデバイスで **mkfs.xfs** を使用する場合は、- **f** オプションを使用してそのファイルシステムを強制的に上書きします。

> **例8.1 mkfs.xfs コマンドの出力**
>
> 以下は、**mkfs.xfs** コマンドの出力例です。
>
> ```
> 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、そのサイズを縮小することはできません。ただし、**xfs_growfs** コマンドを使用して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) の場合は、ファイルシステムの作成時にストライプジオメトリーを指定できます。適切なストライプジオメトリーを使用すると、XFS ファイルシステムのパフォーマンスが向上します。

LVM ボリュームまたは MD ボリュームにファイルシステムを作成する場合、**mkfs.xfs** は最適なジオメトリーを選択します。これは、ジオメトリー情報をオペレーティングシステムにエクスポートする一部のハードウェア RAID でも当てはまる場合があります。

ストライプジオメトリーを指定するには、以下の **mkfs.xfs** サブオプションを使用します。

su=value

ストライプユニットまたは RAID チャンクサイズを指定します。**value** は、バイトで指定する必要があります。オプションで、**k**、**m**、または **g** 接尾辞が付きます。

sw=value

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、4 つのストライプユニットを含む RAID デバイスで 64k のチャンクサイズを指定しています。

```
# 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** を参照してください。

8.2. XFS ファイルシステムのマウント

XFS ファイルシステムは、追加オプションなしでマウントできます。以下に例を示します。

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

XFS は、動作に影響を与える複数のマウントオプションもサポートします。

XFS は、デフォルトでディスク上のロケーションを反映するために inode を割り当てます。ただし、32 ビットのユーザー空間アプリケーションには、inode 番号が 2^{32} を超える inode との互換性がないため、XFS はディスクロケーションに inode をすべて割り当て、32 ビットの inode 番号になります。これにより、非常に大きなファイルシステム（2 テラバイトより大きい）のパフォーマンスが低下する可能性があります。

これに対処するには、**inode64** マウントオプションを使用します。このオプションを使用すると、ファイルシステム全体で inode とデータを割り当てるように XFS を設定します。これにより、パフォーマンスを向上できます。

```
# mount -o inode64 /dev/device /mount/point
```

書き込みバリア

デフォルトでは、XFS は書き込みバリアを使用して、書き込みキャッシュが有効なデバイスの電源が失われた場合でも、ファイルシステムの整合性を確保します。書き込みキャッシュがないデバイス、またはバッテリーでバックアップされた書き込みキャッシュがあるデバイスの場合は、**nobarrier** オプションを使用し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、22章書き込みバリア を参照してください。

8.3. XFS クォータ管理

XFS クォータサブシステムは、ディスク領域 (ブロック) およびファイル (inode) の使用量の制限を管理します。XFS クォータは、ユーザー、グループ、ディレクトリーレベル、またはプロジェクトレベルでこれらの項目の使用を制御または報告します。また、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クォータは個別に有効になりますが、グループとプロジェクトのクォータは相互に排他的であることに注意してください。

ディレクトリーまたはプロジェクトごとに管理する場合、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。そうすることで、XFS はプロジェクト間の組織間にまたがる「グループ」境界を認識します。これにより、ユーザーまたはグループのクォータを管理する際に使用できるレベルよりも、広いレベルの制御が提供されます。

XFS クォータは、特定のマウントオプションを指定して、マウント時に有効になります。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。これにより、制限を強制適用せずに使用状況の報告が可能になります。有効なクォータマウントオプションは以下の通りです。

- ***uquota*/*uqnoenforce***: ユーザークォータ
- ***gquota*/*gqnoenforce***: グループクォータ
- ***pquota*/*pqnoenforce***: プロジェクトクォータ

クォータを有効にすると、**xfs_quota** ツールを使用して制限を設定し、ディスク使用量を報告できます。**xfs_quota** は、デフォルトでは対話形式かつ basic mode で実行されます。基本モードのサブコマンドは使用量を報告するだけで、すべてのユーザーが使用できます。基本的な **xfs_quota** サブコマンドには以下が含まれます。

quota username/userID

*指定の **username** または数値の **userID** の使用状況および制限を表示します。*

df

ブロックおよびinode の空きおよび使用済みの数を表示します。

*これとは対照的に、**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は、制限を実際に設定することができるため、昇格した特権を持つユーザーのみが利用できます。エキスパートモードのサブコマンドを対話的に使用するには、**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報告します。

limit

クォータの制限を変更します。

*基本モードまたはエキスパートモードに関するサブコマンドの一覧は、サブコマンド **ヘルプ** を使用します。*

*すべてのサブコマンドは、**-c** オプションを使用してコマンドラインから直接実行することもできます。エキスパートサブコマンドの場合は **-x** を使用します。*

> ***例8.2 サンプルクォータレポートの表示***
>
> *たとえば、(**/dev/blockdevice** の) **/home** 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、以下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が **/home/john**) に対して、inode 数のソフト制限およびハード制限をそれぞれ500 と 700 に設定するには、以下のコマンドを使用します。*
>
> ```
> # xfs_quota -x -c 'limit isoft=500 ihard=700 /home/john'
> ```

*デフォルトでは、**limit** サブコマンドはターゲットをユーザーとして認識します。グループに制限を設定する場合は、(前出の例のように) **-g** オプションを使用します。同様に、プロジェクトには **-p** を使用します。*

*ソフトブロックおよびハードブロック制限は、**isoft** または **ihard** の代わりに **bsoft** または **bhard** を使用して設定することもできます。*

> ***例8.3 ソフトブロックおよびハードブロックの制限の設定***

> *たとえば、**/target/path** のファイルシステムで **accounting** をグループ化するために、ソフトおよびハードのブロック制限をそれぞれ 1000m と 1200m に設定するには、以下のコマンドを使用します。*
>
> ```
> # xfs_quota -x -c 'limit -g bsoft=1000m bhard=1200m accounting' /target/path
> ```

重要

*リアルタイムブロック (**rtbhard**/**rtbsoft**) は、クォータの設定時に有効な単位として **man xfs_quota** に記述されていますが、本リリースではリアルタイムサブボリュームが有効化されていません。そのため、**rtbhard** オプションおよび **rtbsoft** オプションは、適用されません。*

プロジェクト制限の設定

*プロジェクトが制御するディレクトリーに制限を設定する前に、最初に **/etc/projects** に追加します。プロジェクト名を **/etc/projectid** に追加して、プロジェクト ID をプロジェクト名にマップできます。プロジェクトを **/etc/projects** に追加したら、以下のコマンドを使用してプロジェクトディレクトリーを初期化します。*

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

初期化したディレクトリーを持つプロジェクトのクォータは、以下を使用して設定できます。

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

*汎用クォータ設定ツール (例: **quota**、**repquota**、および **edquota**) を使用して XFS クォータを操作することもできます。ただし、このツールは XFS プロジェクトクォータでは使用できません。*

*XFS クォータの設定に関する詳細は、**man xfs_quota** を参照してください。*

8.4. XFS ファイルシステムのサイズの拡大

***xfs_growfs** コマンドを使用して、マウントしたまま XFS ファイルシステムを大きくすることができます。*

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

***-D size** オプションは、指定した **size** にファイルシステムを拡張します (ファイルシステムブロックで表現)。**xfs_growfs** は、**-D size** のオプションなしでは、デバイスがサポートする最大サイズまでファイルシステムを拡張します。*

***-D size** で XFS ファイルシステムを拡張する前に、基礎となるブロックデバイスが、後でファイルシステムを保持するのに適したサイズであることを確認します。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選択してください。*

注記

XFS ファイルシステムは、マウント中にサイズを大きくすることができますが、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張する方法は、**man xfs_growfs** を参照してください。

8.5. 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非常にスケーラブルで、inode が多数ある非常に大きなファイルシステムを効率的に修復するように設計されています。**xfs_repair** は、他の Linux ファイルシステムとは異なり、XFS ファイルシステムが正しくアンマウントされていなくても起動時には実行されません。クリーンでないアンマウントが発生した場合、**xfs_repair** はマウント時にログを再生するだけで、一貫したファイルシステムが保証されます。

警告

xfs_repair ユーティリティーは、ダーティーログで XFS ファイルシステムを修復できません。ログをクリアするには、XFS ファイルシステムをマウントおよび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) オプションを使用してログの消去を行います (つまり **xfs_repair -L /dev/device**)。これにより、さらなる破損やデータの損失が生じる可能性があることに注意してください。

XFS ファイルシステムの修復の詳細は、**man xfs_repair** を参照してください。

8.6. 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または再開するには、**xfs_freeze** を使用します。書き込みアクティビティーを停止することで、ハードウェアベースのデバイスのスナップショットを使用して、一貫性のある状態でファイルシステムをキャプチャーできます。

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、最初に **xfs_freeze** を使用し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。

注記

***xfs_freeze** ユーティリティーを使用して、ext3、ext4、GFS2、XFS、およびBTRFS、ファイルシステムをフリーズまたはアン解放することもできます。これを実行するための構文は同じです。*

*XFS ファイルシステムのフリーズとアンフリーズに関する詳細は、**man xfs_freeze** を参照してください。*

8.7. XFS ファイルシステムのバックアップおよび復元

*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 つのユーティリティーが必要です。*

*XFS ファイルシステムのバックアップまたはダンプは、**xfsdump** ユーティリティーを使用します。Red Hat Enterprise Linux 6 は、テープドライブまたは通常のファイルイメージへのバックアップをサポートし、また複数のダンプを同じテープに書き込みすることもできます。**xfsdump** ユーティリティーでは、ダンプは複数のテープにまたがることもできますが、通常のファイルに書き込みを行うことができるのは 1 つだけです。さらに、**xfsdump** は増分バックアップをサポートし、フィルターするサイズ、サブツリー、または inode フラグを使用してバックアップからファイルを除外できます。*

*増分バックアップをサポートするために、**xfsdump** は ダンプレベル を使用して、特定のダンプの相対的なベースダンプを決定します。**-l** オプションは、ダンプレベル(0-9)**を指定します**。完全なバックアップを実行するには、以下のようにファイルシステム(**/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

***-f** オプションは、バックアップの宛先を指定します。たとえば、**/dev/st0** 宛先は通常テープドライブに使用されます。**xfsdump** 宛先は、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です。*

*これとは対照的に、増分バックアップは最後 **のレベル** 0 のダンプ以降に変更したダンプファイルのみになります。**レベル** 1 のダンプは、フルダンプ後の最初の増分ダンプです。次の増分バックアップダンプは **レベル** 2 であるため、**レベル** 9 **までになります**。したがって、テープドライブに **レベル** 1 のダンプを実行するには、次のコマンドを実行します。*

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

*一方、**xfsrestore** ユーティリティーは、**xfsdump** が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 **単純な** モードと **累積** モードの 2 つのモードがあります。特定のダンプは セッション ID またはセッション ラベル によって識別されます。したがって、ダンプを復元するには、対応するセッション ID またはラベルが必要です。すべてのダンプのセッション ID およびラベル（フル増分および増分の両方）を表示するには、- **I** オプションを使用します。*

```
# xfsrestore -I
```

これにより、以下のような出力が表示されます。

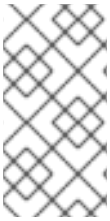
例8.4 すべてのダンプのセッション ID およびラベル

```
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の単純なモード

simple *モードでは、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できます。***レベル** 0 *のダンプのセッション ID（例：* ***session-ID***）*を特定した後に、以下を使用して* ***/path/to/destination*** *に完全に復元します。*

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



注記

-f *オプションはダンプの場所を指定します。-* ***S オプションまたは - L*** *オプションは、復元する特定のダンプを指定します。****-S*** *オプションはセッション ID を指定するために使用されますが、-* ***L*** *オプションがセッションラベルに使用されます。****-I*** *オプションは、各ダンプのセッションラベルと ID の両方を表示します。*

xfsrestoreの累積モード

xfsrestore *の* **累積** *モードを使用すると、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が可能になります（例：* **レベル** 1 *から* **レベル** 9）*。増分バックアップからファイルシステムを復元するには、-* ***r オプションを追加する*** *だけです。*

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

インタラクティブ操作

xfsrestore ユーティリティーは、ダンプからの特定のファイルを抽出、追加、または削除することもできます。**xfsrestore** をインタラクティブに使用するには、**-i** オプションを使用します。

xfsrestore -f /dev/st0 -i

インタラクティブダイアログは、**xfsrestore** が、指定したデバイスの読み込み中にエラーが発生しました。このダイアログで利用可能なコマンドには、cd 、 **ls**、**add**、**delete**、および **extract** などがあります。コマンドの全一覧については、**help** を使用してください。

XFS ファイルシステムのダンプおよび復元の詳細は、**man xfsdump および man xfsrestore を参照してください。**

8.8. 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 6 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、ユーザーは指定された時間にデフラグを一時停止し、後で中断したところから再開することもできます。

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。Red Hat は、通常保証されないため、ファイルシステム全体を定期的にデフラグすることをお勧めします。

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トや、該当するブロックがないファイルの領域 (ホール) を一覧表示します。

xfs_info

XFS ファイルシステムの情報を表示します。

xfs_admin

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

xfs_copy

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

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

xfs_metadump

XFS ファイルシステムのメタデータをファイルにコピーします。**xfs_metadump** ユーティリティーは、アンマウントされたファイルシステム、読み取り専用、またはフリーズしたファイルシステムをコピーする場合にのみ使用する必要があります。それ以外の場合は、生成されたダンプが破損したり、一貫性のない可能性があります。

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 ファイルシステムをデバッグします。

*これらのユーティリティーの詳細は、それぞれの **man** ページを参照してください。*

第9章 NETWORK FILE SYSTEM (NFS)

ネットワークファイルシステム (NFS) を利用すると、リモートのホストが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。

9.1. NFS の仕組み

現在、NFS のバージョンは 3 つあります。NFS バージョン 2(NFSv2) は、以前のバージョンで完全にサポートされています。NFS バージョン 3 (NFSv3) は安全な非同期書き込みに対応しており、NFSv2 よりもエラー処理において安定しています。64 ビットのファイルサイズとオフセットにも対応しているため、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。NFSv2 は、Red Hat Enterprise Linux 7 **でサポートされない** ことに注意してください。

NFS バージョン 4 (NFSv4) はファイアウォールやインターネットを介して動作し、**rpcbind** サービスを必要とせず、ACL に対応し、ステートフルな操作を利用します。Red Hat Enterprise Linux 6 は NFSv2、NFSv3、および NFSv4 クライアントに対応します。NFS でファイルシステムをマウントすると、サーバーがサポートする場合に、Red Hat Enterprise Linux はデフォルトで NFSv4 を使用します。

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。NFSv2 および 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 NFSv2 または 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。

NFSv4 プロトコルには、マウントとロックのプロトコルが組み込まれています。サーバーは、既知の TCP ポート 2049 もリッスンします。したがって、NFSv4 は **rpcbind**と対話する必要はありません。[3]、**lockd**、および **rpc.statd** デーモン。エクスポートを設定するには、rpc **.mountd** デーモンが NFS サーバーで必要になります。

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 が必要です。

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。アクセスが可能なことが確認されると、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

ファイアーウォールを有効にしている Red Hat Enterprise Linux のデフォルトインストールで NFS を正しく動作させるために、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。*

9.1.1. 必要なサービス

*Red Hat Enterprise Linux は、カーネルレベルのサポートとデーモンプロセスの組み合わせを使用して、NFS ファイル共有を提供します。すべての NFS バージョンは、クライアントとサーバー間の Remote Procedure Calls (RPC) に依存します。Red Hat Enterprise Linux 6 での RPC サービスは、 **rpcbind** サービスによって制御されます。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 6 で IPv6 に対応にするため、**rpcbind** に置き換えられています。*

nfs

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

nfslock

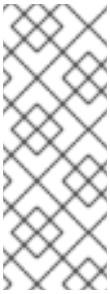
service nfslock は、NFS クライアントがサーバー上のファイルをロックできるように適切な RPC プロセスを開始する必須サービスをアクティブにします。

rpcbind

***rpcbind** は、ローカルの RPC サービスからのポート予約を受け入れます。その後、これらのポートは、対応するリモートの RPC サービスにより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 サービスへの接続を設定します。このプロセスは NFSv4 では使用されません。*

rpc.nfsd

***rpc.nfsd** は、サーバーが公開している明示的な NFS のバージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的な要求に対応するため、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。*



注記

Red Hat Enterprise Linux 6.3 では、NFSv4 サーバーのみが **rpc.idmapd** を使用します。NFSv4 クライアントは、キーリングベースの idmapper **nfsidmap** を使用します。**nfsidmap** は、ID マッピングを実行するためにオンデマンドでカーネルによって呼び出されるスタンドアロンプログラムです。**nfsidmap** に問題がある場合は、クライアントが **rpc.idmapd** の使用にフォールバックします。**nfsidmap** の詳細は、man ページの nfsidmap を参照してください。

以下の RPC プロセスは、NFS サービスを容易にします。

rpc.mountd

NFS サーバーは、このプロセスを使用して NFSv2 クライアントおよび 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 サーバーによりエクスポートされているか、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確認します。マウント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 共有の **File-Handle** を NFS クライアントに戻します。

lockd

lockd は、クライアントとサーバーの両方で実行するカーネルスレッドです。Network Lock Manager (NLM) プロトコルを実装し、これにより、NFSv2 および NFSv3 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なります。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

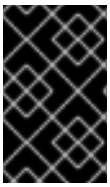
このプロセスは、Network Status Monitor (NSM) RPC プロトコルを実装します。NFS サーバーが正常にシャットダウンされずに再起動すると、NFS クライアントに通知します。**rpc.statd** は **nfslock** サービスが自動的に起動するため、ユーザー設定は必要ありません。このプロセスは NFSv4 では使用されません。

rpc.rquotad

このプロセスは、リモートユーザーのユーザークォーター情報を提供します。**rpc.rquotad** は **nfs** サービスにより自動的に起動するため、ユーザー設定は必要ありません。

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。少なくとも、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 ドメイン名と同じ場合は、このパラメーターは必要ありません。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合意しないと、適切に動作しません。



重要

NFSv4 の ID マッピングを Red Hat Enterprise Linux の特定のバージョンで処理される方法の詳細は、ナレッジベースの記事「RHEL: NFSv4 and ID mapping 」を参照してください。

9.2. PNFS

NFS v4.1 標準の一部として Parallel NFS(pNFS) のサポートは、Red Hat Enterprise Linux 6.4 以降で利用できます。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ソースを介してデータにアクセスできるようになります。

*この機能を有効にするには、pNFS 対応のサーバーからのマウントに **-o v4.1** マウントオプションを使用します。*

*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。モジュールが正常にロードされると、以下のメッセージが **/var/log/messages** ファイルに記録されます。*

```
kernel: nfs4filelayout_init: NFSv4 File Layout Driver Registering...
```

NFSv4.1 マウントが正常に行われたことを確認するには、以下を実行します。

```
$ mount | grep /proc/mounts
```

重要

*サーバーおよびクライアントが NFS v4.1 以降をネゴシエートすると、利用可能な場合は pNFS を活用します。クライアントおよびサーバーはどちらも同じ「layout タイプ」をサポートする必要があります。使用できるレイアウトタイプには、**ファイル**、**ブロック**、**オブジェクト**、柔軟性ファイル、 および **SCSI** が含まれます。*

*Red Hat Enterprise Linux 6.4 以降、クライアントは **ファイルレイアウトタイプのみをサポートし、サーバーが files レイアウト** タイプも対応する場合に限り pNFS を使用します。Red Hat は、Red Hat Enterprise Linux 6.6 以降でのみ、**ファイル** プロファイルを使用することを推奨します。*

pNFS の詳細は を参照してください http://www.pnfs.com。

9.3. NFS クライアント設定

***mount** コマンドは、クライアント側に NFS 共有をマウントします。形式は以下のようになります。*

```
# 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。

オプション

マウントオプションのカンマ区切りリスト。有効な NFS マウントオプションの詳細については、「一般的な NFS マウントオプション」 を参照してください。

server

マウントする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またはディレクトリー、つまり、マウントするディレクトリー

/local/directory

/remote/export がマウントされているクライアントの場所

*Red Hat Enterprise Linux 6 で使用される NFS プロトコルのバージョンは、**mount** オプションの **nfsvers** または **vers** で識別されます。デフォルトでは、**mount は mount -t nfs** で NFSv4 を使用します。サーバーが NFSv4 に対応していない場合、クライアントはサーバーがサポートしているバージョンに自動的にステップダウンします。サーバーでサポートされていない特定のバージョンを渡すために **nfsvers**/**vers** オプションを使用した場合は、マウントに失敗します。レガシーの理由により、ファイルシステムタイプ nfs4 も利用可能です。これは、**mount -t nfs -o nfsvers=4 host:/remote/export /local/directory** を実行することと同じになります。*

*詳細は **man mount** を参照してください。*

*NFS 共有が手動でマウントされた場合は、再起動時に共有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、**/etc/fstab** ファイルおよび **autofs** サービスの 2 つの方法を提供します。詳細は、「**/etc/fstab** を使用した NFS ファイルシステムのマウント」 および 「**autofs**」 を参照してください。*

9.3.1. /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別の方法は、**/etc/fstab** ファイルに行を追加することです。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** ファイルを変更するには、root でなければなりません。*

> ***例9.1 構文の例***
>
> ***/etc/fstab** の行の一般的な構文は以下の通りです。*
>
> ```
> server:/usr/local/pub    /pub    nfs    defaults 0 0
> ```

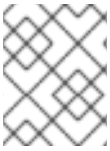
*このコマンドを実行する前に、マウントポイント **/pub** がクライアントマシンに存在している必要があります。この行をクライアントシステムの **/etc/fstab** に追加した後に、コマンド **mount /pub** を使用すると、マウントポイント **/pub** がサーバーからマウントされます。*

***/etc/fstab** ファイルは起動時に **netfs** サービスで参照されるため、NFS 共有を参照する行は、ブートプロセス中に手動で **mount** コマンドを入力するのと同じ効果があります。*

*NFS エクスポートをマウントする有効な **/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 server、/remote/export、/local/directory、および options は、NFS 共有を手動でマウントする際に使用されるものと同じです。各変数の定義については、「NFS クライアント設定」 を参照してください。



> ***注記***
>
> ***/etc/fstab** を読み取る前に、マウントポイント /local/directory はクライアントに存在している必要があります。そうしないと、マウントは失敗します。*

/etc/fstab の詳細は、man fstab を参照してください。

9.4. AUTOFS

/etc/fstab を使用する場合の欠点の1 つは、NFS マウントされたファイルシステムにユーザーがアクセスする頻度に関わらず、マウントされたファイルシステムを所定の場所で維持するために、システムがリソースを割り当てる必要があることです。これは1 つまたは2 つのマウントでは問題になりませんが、システムが一度に多くのシステムへのマウントを維持している場合、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。/etc/fstab に代わるのは、カーネルベースの automount ユーティリティーを使用することです。自動マウント機能は以下の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空間デーモン*

automount ユーティリティーは、NFS ファイルシステムを自動的にマウントおよびアンマウント (オンデマンドマウント) できるため、システムリソースを節約できます。これは、AFS、SMBFS、CIFS、およびローカルファイルシステムを含む他のファイルシステムをマウントするために使用できます。

重要

nfs-utils パッケージは、「NFS ファイルサーバー」と「ネットワークファイルシステムクライアント」グループの両方に含まれるようになりました。そのため、Base グループではデフォルトでインストールされなくなりました。NFS 共有の自動マウントを試みる前に、最初に 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。

autofs は、「ネットワークファイルシステムクライアント」グループに含まれます。

autofs は、デフォルトのプライマリー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 つのデーモンを使用して管理されるため、これを実行することができなくなりました。これは、他の業界標準の自動マウント機能の通常の要件と一致しています。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動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。

9.4.1. バージョン4 と比較した autofs バージョン5 の改善点

autofs バージョン5 では、バージョン4 と比較して、以下の機能拡張が行われています。

ダイレクトマップのサポート

autofs のダイレクトマップは、ファイルシステム階層の任意の時点でファイルシステムを自動的にマウントするメカニズムを提供します。ダイレクトマップは、マスターマップの /- のマウントポイントによって示されます。ダイレクトマップのエントリーには、(間接マップで使用される相対パス名の代わりに) 絶対パス名がキーとして含まれています。

レイジーマウントとアンマウントのサポート

マルチマウントマップエントリーは、単一のキーの下にあるマウントポイントの階層を記述します。この良い例として、-hosts マップがあります。これは通常、マルチマウントマップエントリーとして、/net/host の下のホストからのすべてのエクスポートを自動マウントするために使用されます。-hosts マップを使用する場合は、/net/host の ls が、host からの各エクスポートの autofs トリ

ガーマウントをマウントします。次に、これらはマウントされ、アクセスされると期限切れとなります。これにより、エクスポートが多数あるサーバーにアクセスする際に必要なアクティブなマウントの数を大幅に減らすことができます。

強化された LDAP サポート

autofs 設定ファイル (**/etc/sysconfig/autofs**) は、サイトが実装する **autofs** スキーマを指定するメカニズムを提供するため、アプリケーション自体で試行錯誤してこれを判断する必要がなくなります。さらに、共通の LDAP サーバー実装でサポートされるほとんどのメカニズムを使用して、LDAP サーバーへの認証済みバインドがサポートされるようになりました。このサポートには、新しい設定ファイル (**/etc/autofs_ldap_auth.conf**) が追加されました。デフォルトの設定ファイルは自己文書化されており、XML 形式を使用します。

Name Service Switch (nsswitch) 設定の適切な使用

Name Service Switch 設定ファイルは、特定の設定データがどこから来るのかを判別する手段を提供するために存在します。この設定の理由は、データにアクセスするための統一されたソフトウェアインターフェイスを維持しながら、管理者が最適なバックエンドデータベースを柔軟に使用できるようにするためです。バージョン 4 の自動マウント機能は、今まで以上に NSS 設定を処理できるようになっていますが、まだ完全ではありません。一方、autofs バージョン 5 は完全な実装です。

このファイルで対応している構文の詳細は、**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訳ではなく、パーサーは無効なデータベースを拒否します。有効なソースは、ファイル、**yp**、**nis**、**nisplus**、**ldap** および **hesiod** です。

autofs マウントポイントごとの複数のマスターマップエントリー

頻繁に使用されますが、まだ記述されていないのは、直接マウントポイント **/-** の複数のマスターマップエントリーの処理です。各エントリーのマップキーはマージされ、1 つのマップとして機能します。

> **例9.2 autofs マウントポイントごとの複数のマスターマップエントリー**
>
> 以下にダイレクトマウントの connectathon テストマップの例が表示されます。
>
> ```
> /- /tmp/auto_dcthon
> /- /tmp/auto_test3_direct
> /- /tmp/auto_test4_direct
> ```

9.4.2. 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** であり、マスターマップとも呼ばれます。マスターマップは、「バージョン 4 と比較した autofs バージョン 5 の改善点」 で説明されているとおり変更できます。マスターマップには、システム上の **autofs** 制御のマウントポイントと、それに対応する構成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一覧表示されます。マスターマップの形式は次のとおりです。

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

この形式で使用されている変数を以下に示します。

mount-point

autofs *マウントポイント (例: /**home**)*

map-name

マウントポイントの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は以下で説明します。

オプション

*指定されている場合、それら自体にオプションが指定されていない場合に、指定されたマップ内のすべてのエントリーに適用されます。この動作は、オプションが累積されていた **autofs** バージョン 4 とは異なります。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9.3 /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を以下に示します。

mount-point

*これは **autofs** マウントポイントを参照します。これは 1 つのインダイレクトマウント用の 1 つのディレクトリー名にすることも、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にすることもできます。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キー (上記の **mount-point**) の後には、スペースで区切られたオフセットディレクトリー (それぞれ「/」で始まるサブディレクトリー名) のリストが続き、マルチマウントエントリーと呼ばれるものになります。*

オプション

指定した場合は、これらは、独自のオプションを指定しないマップエントリーのマウントオプションになります。

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「/」で始まります)、NFS ファイルシステム、他の有効なファイルシステムの場所などのファイルシステムの場所を参照します。

*以下は、マップファイルのコンテンツの例になります (例: **/etc/auto.misc**)。*

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

*マップファイルの最初の列は、**autofs** マウントポイント (**personnel** と呼ばれるサーバーの **sales** と **payroll**) を示します。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。任意の設定に基づき、autofs マウントポイントは、**/home/payroll** と*

*/**home**/**sales** になります。**-fstype=** オプションは省略されることが多く、通常は正しい操作には必要ありません。*

ディレクトリーが存在しない場合、自動マウント機能はディレクトリーを作成します。ディレクトリーが存在している状況で自動マウント機能が起動した場合は、自動マウント機能の終了時にディレクトリーが削除されることはありません。以下のいずれかのコマンドを実行して、自動マウントデーモンを起動または再起動できます。

- ***service autofs の起動** （自動マウントデーモンが停止した場合）*
- ***service autofs restart***

*所定の設定を使用して **/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、自動デーモンは、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。タイムアウトを指定した場合は、タイムアウト期間中ディレクトリーにアクセスしないと、ディレクトリーが自動的にアンマウントされます。*

以下のコマンドを実行して、自動マウントデーモンのステータスを表示できます。

```
# service autofs status
```

9.4.3. サイト設定ファイルの上書きまたは拡張

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、次の条件を検討します。

- *自動マウント機能マップは NIS に保存され、**/etc/nsswitch.conf** ファイルには以下のディレクティブがあります。*

  ```
  automount:    files nis
  ```

- ***auto.master** ファイルには、*

  ```
  +auto.master
  ```

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

  ```
  /home auto.home
  ```

- *NIS の **auto.home** マップには以下が含まれています。*

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

- *ファイルマップ **/etc/auto.home** 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。この場合、クライアントは以下の **/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ます。最後に、NIS の auto.home マップを含めます。次に、/etc/auto.home ファイルマップは以下のようになります。

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

上記の NIS の auto.home マップがリストされると、ls /home が出力されます。

```
beth joe mydir
```

autofs は、読み取り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、上記の例は期待どおりに動作します。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。

9.4.4. LDAP を使用した自動マウント機能マップの格納

LDAP から自動マウント機能マップを取得するように設定されているすべてのシステムに、LDAP クライアントライブラリーをインストールする必要があります。Red Hat Enterprise Linux では、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には、スキーマ定義からコメント文字を削除して、autofs 設定 /etc/autofs.conf で設定する必要があります。

例9.4 autofs の設定

```
map_object_class = automountMap
entry_object_class = automount
map_attribute = automountMapName
entry_attribute = automountKey
value_attribute = automountInformation
```

注記

Red Hat Enterprise Linux 6.6 の時点で、LDAP autofs は以前のリリースの /etc/systemconfig/autofs ファイルではなく、/etc/autofs.conf ファイルで設定されます。

設定内でコメントされていないスキーマエントリー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。設定例の LDIF の説明は次のとおりです。

例9.5 LDIF 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/&
```

9.5. 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、手動の **mount** コマンド、**/etc/fstab** 設定、**autofs** と併用できます。

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

任意のマウントポイントに対して、カーネルがディレクトリーエントリーのキャッシュを管理する方法を指定します。mode の有効な引数は、**all**、**none**、または **pos**/**positive** です。

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、または 4 になります。これは、複数の NFS サーバーを実行するホストに役立ちます。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** と同じで、互換性のためにこのリリースに含まれています。

noacl

ACL 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 NFS サーバーへの接続時に必要になることがあります。

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

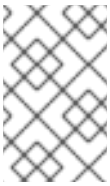
set-user-identifier または **set-group-identifier** ビットを無効にします。これにより、リモートユーザーは、**setuid** プログラムを実行してより高い権限を取得できなくなります。

port=num

port=num - NFS サーバーポートの数値を指定します。**num** を **0** (デフォルト値) 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

これらの設定は、大規模なデータブロックサイズ（数字、バイト単位）を設定して、 読み取り (rsize) および書き込み(**wsize**) の NFS 通信を 1 度に転送します。これらの値を変更する場合には注意が必要です。古い Linux カーネルとネットワークカードは、大きなブロックサイズで適切に機能しません。



注記

rsize の値が指定されていない場合や、指定した値が、クライアントまたはサーバーがサポートできる最大値よりも大きい場合、クライアントとサーバーは両方のサポートが可能な最大サイズサイズをネゴシエートします。

sec=mode

NFS 接続の認証時に使用するセキュリティーのタイプを指定します。デフォルトの設定は **sec=sys** で、NFS 操作を認証する **AUTH_SYS** を使用してローカルの UNIX UID および GID を使用します。

sec=krb5 は、ユーザー認証に、ローカルの UNIX の 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用して、NFS 操作の整合性チェックを行います。

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

tcp

NFS マウントが TCP プロトコルを使用するよう指示します。

udp

NFS マウントが UDP プロトコルを使用するよう指示します。

オプションの完全なリストと、各オプションの詳細は、**man mount および man nfs を参照してください。**

9.6. NFS の起動および停止

NFS サーバーを実行するには、**rpcbind**[3] 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、次のコマンドを実行します。

```
# service rpcbind status
```

rpcbind サービスを実行している場合は、**nfs** サービスを起動できます。NFS サーバーを起動するには、次のコマンドを使用します。

```
# service nfs start
```

-

NFS クライアントとサーバーの両方が適切に機能するには、**nfslock** も起動する必要があります。NFS ロックを開始するには、以下のコマンドを使用します。

```
# service nfslock start
```

NFS がシステムの起動時に起動するように設定されている場合は、chkconfig **--list nfslock を実行して、nfslock** も起動してください。**nfslock** が **on** に設定されていない場合には、コンピューターの起動時に毎回 **service nfslock を手動で実行** する必要があることを意味します。**nfslock** がシステムの起動時に自動的に起動するように設定するには、chkconfig **nfslock on** を使用します。

nfslock は NFSv2 および NFSv3 にのみ必要です。

サーバーを停止させるには、以下を使用します。

```
# service nfs stop
```

restart オプションは、停止して NFS を起動する簡単な方法です。これは、NFS の設定ファイルを編集した後に設定変更を行う最も効率的な方法です。サーバータイプを再起動するには、以下を実行します。

```
# service nfs restart
```

condrestart（条件付き再起動）オプションは、現在実行している場合にのみ **nfs** を起動します。このオプションは、デーモンが実行していない場合は起動しないため、スクリプトに役に立ちます。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。

```
# service nfs condrestart
```

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

```
# service nfs reload
```

9.7. NFS サーバーの設定

NFS サーバーの設定方法は 2 つあります。

- NFS の設定ファイル (つまり **/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。

9.7.1. /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 (\) で改行できる。

- 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に空白文字を追加し、その後に追加する。
- 各ホストのオプションは、ホストの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使用しない。

エクスポートするファイルシステムの各エントリーは、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

オプション
host に使用されるオプション

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別の方法は、「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9.6 /etc/exports ファイル

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

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

デフォルトの設定は以下のようになります。

ro
エクスポートするファイルシステムは読み取り専用です。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できません。このファイルシステムで変更 (つまり読み取り/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、書き込みのオーバーヘッドが低下し、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使用できます。

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 サーバーにより、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て (squash)、高い確率でリモートサーバーへの書き込む権限を与えないようにすることができます。root squashing を無効にするには、**no_root_squash** を指定します。

（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** オプションと **anongid** オプションを以下のように使用します。

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

uid と gid は、それぞれユーザー ID とグループ ID の番号になります。 **anonuid** オプションと **anongid** オプションにより、共有するリモート NFS ユーザー用に、特別なユーザーアカウントおよびグループアカウントを作成できます。

Red Hat Enterprise Linux の NFS では、デフォルトで アクセス制御リスト (ACLs) に対応しています。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。

エクスポートするすべてのファイルシステムの各デフォルトは、明示的に上書きする必要があります。たとえば、**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が読み取り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ます。

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

この例では、**192.168.0.3** は **/another/exported/directory/** の読み書きをマウントでき、ディスクへの書き込みはすべて非同期になります。エクスポートオプションの詳細は、**man exportfs を参照してください**。

さらに、デフォルト値が指定されていない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。あまり使用されないオプションの詳細については、**man エクスポート** を参照してください。

重要

/etc/exports ファイルの形式では、特に空白文字の使用が非常に厳しく扱われます。ホストからエクスポートす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、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 **/home** ディレクトリーへの読み取り/書き込みアクセスのみを許可します。2 番目の行では、**bob.example.com** からのユーザーにディレクトリーを読み取り専用 (デフォルト) でマウントすることを許可し、その他のユーザーに読み取り/書き込みでマウントすることを許可します。

9.7.2. 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 および NFSv3 の場合) に渡してから、**rpc.nfsd** に渡します。この時点でリモートユーザーがファイルシステムを使用できるようになります。

/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/etab** 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 **etab** 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。

以下は、**/usr/sbin/exportfs** で利用可能な一般的に使用されるオプションの一覧です。

-r

/etc/exports に記載されるすべてのディレクトリーから、**/etc/lib/nfs/etab** に新しいエクスポート一覧を作成して、すべてのディレクトリーをエクスポートします。このオプションにより、**/etc/exports** に変更が加えられると、エクスポート一覧が効果的に更新されます。

-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、**/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 内に記載されていない、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追加の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、エクスポートするファイルシステムのリストに永続的に追加する前に、エクスポートするファイルシステムをテストするためによく使用されます。**/etc/exports** 構文の詳細は、「**/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。コマンドラインで指定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なくなります。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

詳細な表示です。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 コマンドの詳細は、man exportfs を参照してください。

9.7.2.1. NFSv4 で exportfs の使用

Red Hat Enterprise Linux 6 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 NFSv2、NFSv3、NFSv4 クライアントで利用可能になるため、NFSv4 のエクスポートを設定するための特別なステップは必要ありません。これが、以前のバージョンとは異なります。

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。

9.7.3. 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。クライアントがファイアウォールの背後にある NFS 共有にアクセスできるようにするには、/etc/sysconfig/nfs 設定ファイルを編集して、必要な RPC サービスを実行するポートを制御します。

すべてのシステムでは、デフォルトで /etc/sysconfig/nfs が存在しない場合があります。存在しない場合は、これを作成し、以下の変数を追加して、port を未使用の ポート番号 で置き換えます（ファイルが存在する場合は、コメント解除して、デフォルトのエントリーを必要に応じて変更します）。

MOUNTD_PORT=port

rpc.mountd (rpc.mountd)が使用する TCP ポートおよび UDP ポートのマウント を制御します。

STATD_PORT=port

rpc.statd が使用する TCP および UDP ポートのステータス(rpc.statd) を制御します。

LOCKD_TCPPORT=port

どの TCP portnlock mgr (lockd)が使用するかを制御します。

LOCKD_UDPPORT=port

どの UDP ポートnlock mgr (lockd)が使用するかを制御します。

NFS が起動しない場合は、/var/log/messages を確認してください。通常、すでに使用中のポート番号を指定すると、NFS が起動しません。/etc/sysconfig/nfs を編集したら、service nfs restart を使用して NFS サービスを再起動します。rpcinfo -p コマンドを実行して変更を確認します。

NFS を許可するようにファイアウォールを設定するには、以下の手順を実行します。

手順9.1 NFS を許可するようにファイアウォールを設定

1. NFS 用に TCP ポート 2049 および UDP ポート 2049 を許可します。
2. TCP および UDP ポート 111(**rpcbind/sunrpc)を許可します。**
3. で指定する TCP ポートおよび UDP ポートを許可します。 **MOUNTD_PORT="port"**
4. で指定する TCP ポートおよび UDP ポートを許可します。 **STATD_PORT="port"**
5. で指定する TCP ポートを許可します。 **LOCKD_TCPPORT="port"**
6. で指定する UDP ポートを許可します。 **LOCKD_UDPPORT="port"**

注記

NFSv4.0 コールバックがファイアウォールを通過できるようにするには、**/proc/sys/fs/nfs/nfs_callback_tcpport** を設定して、サーバーがクライアント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 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。

ファイアウォールの背後での NFS の設定に関する詳細は、以下の Red Hat ナレッジベース記事を参照してください。

- システムを NFSv3 サーバーとして設定する方法。ファイアウォールの外部にある NFS クライアントを使用してファイアウォールの内側にありますか？
- システムを NFSv4 サーバーとして設定する方法。ファイアウォールの外部にある NFS クライアントを使用してファイアウォールの内側にありますか？

9.7.3.1. NFS エクスポートの検出

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

まず、NFSv2 または NFSv3 に対応しているサーバーで、show **mount** コマンドを使用します。

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

次に、NFSv4 に対応しているサーバーで、マウント / を確認し、確認します。

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

NFSv4 と、NFSv2 または NFSv3 のいずれか一方に対応するサーバーでは、両方の方法が機能し、同じ結果が得られます。

注記

Red Hat Enterprise Linux 6 以前には、設定の仕方によって以前の 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。これらのサーバーはデフォルトでは NFSv4 を有効にしないため、通常は問題になるべきではありません。

9.7.4. ホスト名の形式

ホストは以下の形式にすることができます。

単独のマシン

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

***** 文字または **?** 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誤って動作する可能性があります。完全修飾ドメイン名でワイルドカードを指定する場合、ドット (.) はワイルドカードに含まれません。たとえば、***.example.com** には **one.example.com** が含まれていますが、**include one.two.example.com** は含まれません。

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、z はネットマスクのビット数です (たとえば、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 (たとえば、192.168.100.8/255.255.255.0)。

Netgroups

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

9.7.5. RDMA 経由の NFS

NFSoRDMA(Remote Direct Memory Access) を介した NFS は、大量のデータを転送する必要がある CPU 集中型のワークロードに適しています。NFSoRDMA は通常 InfiniBand fiber で使用されます。これにより、レイテンシーが低く、パフォーマンスが向上します。RDMA で利用可能なデータ移動オフロード機能は、コピーされたデータ量を減らします。

手順9.2 NFS サーバーでの RDMA トランスポートの有効化

1. RDMA RPM がインストールされ、RDMA サービスが有効になっていることを確認します。

```
# yum install rdma; chkconfig --level 2345 rdma on
```

2. **nfs-rdma サービスを提供する** パッケージがインストールされ、サービスが有効であることを確認します。

```
# yum install rdma; chkconfig --level 345 nfs-rdma on
```

3. RDMA ポートが推奨されるポートに設定されていることを確認します（Red Hat Enterprise Linux 6 のデフォルトは 2050）: **/etc/rdma/rdma.conf ファイルを編集して、NFSoRDMA_LOAD=yes** および **NFSoRDMA_PORT** を必要なポートに設定します。

4. NFS マウントでは、エクスポートしたファイルシステムを通常通りセットアップします。

手順9.3 クライアントからの RDMA の有効化

1. RDMA RPM がインストールされ、RDMA サービスが有効になっていることを確認します。

```
# yum install rdma; chkconfig --level 2345 rdma on
```

2. マウント呼び出しで RDMA オプションを使用して、NFS でエクスポートされたパーティションをマウントします。port オプションは、任意で呼び出しに追加できます。

```
# mount -t nfs -o rdma,port=port_number
```

以下の Red Hat ナレッジベースの記事 は、NFSoRDMA でサポートされるカーネルモジュールを使用するカードの概要を説明しています： What RDMA ハードウェアは、Red Hat Enterprise Linux でサポートされますか？

9.8. NFS のセキュア化

NFS は、透過的な方法で多数の既知のホストとファイルシステム全体を共有するのに適しています。ただし、使いやすさがある反面、さまざまなセキュリティー問題を伴います。サーバーで 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際に、以下のセクションを検討してください。これにより、NFS セキュリティーリスクが最小限に抑えられ、サーバーのデータを保護します。

9.8.1. AUTH_SYS とエクスポート制御による NFS 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用して、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** とも呼ぶ) 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、このソリューションにより、NFS 共有が元々想定されている方法では使用されなくなります。

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**[3] 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** のセキュリティー保護の詳細は、**man iptables** を参照してください。

9.8.2. AUTH_GSS を使用した NFS セキュリティー

NFSv4 のリリースでは、RPCSEC_GSS および Kerberos バージョン 5 GSS-API メカニズムの実装を調整することで、NFS セキュリティーへの再電子が発生します。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。

RPCSEC_GSS Kerberos メカニズムでは、サーバーは AUTH_SYS のケースのように、どのユーザーがファイルにアクセスするかを正確に表すため、クライアントに依存しなくなりました。代わりに、暗号化を使用してサーバーにユーザーを認証し、悪意のあるクライアントがそのユーザーの Kerberos 認証情報を持たずにユーザーを偽装しないようにします。

注記

NFSv4 サーバーを設定する前に、Kerberos チケット保証サーバー(KDC)がインストールされ、正しく設定されていることを前提とします。Kerberos はネットワーク認証システムであり、対称暗号化と信頼できるサードパーティーである KDC を使用して、クライアントとサーバーが相互に認証できるようにします。Kerberos の詳細は、『Red Hat 『Identity Management ガイド』を参照してください』。

RPCSEC_GSS を設定するには、以下の手順に従います。

手順9.4 RPCSEC_GSS のセットアップ

1. **nfs/client.mydomain@MYREALM** および **nfs/server.mydomain@MYREALM** プリンシパルを作成します。
2. クライアントとサーバーのキータブに、対応する鍵を追加します。
3. サーバーで、sec **=krb5,krb5i,krb5p** をエクスポートに追加します。AUTH_SYS を引き続き許可するには、sec **=sys**、**krb5**、**krb5i**、**krb5p** を追加します。
4. クライアントで、**sec=krb5** (もしくは設定に応じて **sec=krb5i** または **sec=krb5p**) をマウントオプションに追加します。

krb5、**krb5i**、**krb5 p** の相違点などの詳細は、man ページの **exports** および **nfs** を参照してください。または、「一般的な NFS マウントオプション」 を参照してください。

rpc **.svcgssd および rpc.gssd** の相互運用方法など、**RPCSEC_GSS** フレームワークの詳細は、http://www.citi.umich.edu/projects/nfsv4/gssd/ を参照してください。

9.8.2.1. NFSv4 による NFS の保護

NFSv4 には、以前の機能や幅広いデプロイメントが原因で、POSIX モデルではなく、Microsoft Windows NT モデルをベースとした ACL サポートが含まれています。

NFSv4 のもう 1 つの重要なセキュリティー機能は、ファイルシステムのマウントに **MOUNT** プロトコルを使用しなくなることです。このプロトコルは、ファイルハンドルが処理する方法により、セキュリティーの不安定になります。

9.8.3. ファイル権限

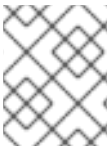
NFS ファイルシステムがリモートホストにより読み取り/書き込みをマウントすると、各共有ファイルに対する唯一の保護がパーミッションになります。同じユーザー ID 値を共有している 2 つのユーザーが同じ NFS ファイルシステムをマウントすると、それらのユーザーはファイルを相互に変更することができます。また、クライアントシステムで root としてログインした場合は、**su -** コマンドを使用して、NFS 共有が設定されたファイルにアクセスできます。

デフォルトでは、アクセス制御リスト (ACL) は、Red Hat Enterprise Linux では NFS が対応しています。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。

デフォルトでは、NFS がファイルシステムをエクスポートする際に、root squashing を使用します。これにより、NFS 共有にアクセスするユーザーのユーザー ID が、ローカルマシンの root ユーザーとして **nobody** に設定されます。root squashing は、デフォルトのオプション **root_squash** で制御されます。このオプションの詳細は、「**/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としてエクスポートする場合は、**all_squash** オプションの使用を検討してください。このオプションでは、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が、**nfsnobody** ユーザーのユーザー ID を取得します。

9.9. NFS および RPCBIND



備考

次のセクションは、後方互換用に **rpcbind** サービスを必要とする NFSv2 または NFSv3 の実装のみに適用されます。

rpcbind[3] ユーティリティーは、RPC サービスを、それらのサービスがリッスンするポートにマッピングします。RPC プロセスが開始すると、その開始が **rpcbind** に通知され、そのプロセスがリッスンしているポートと、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は、特定の RPC プログラム番号でサーバーの **rpcbind** と通信します。**rpcbind** サービスは、クライアントを適切なポート番号にリダイレクトし、要求されたサービスと通信できるようにします。

RPC ベースのサービスは、**rpcbind** を使用して、クライアントの受信要求ですべての接続を確立します。したがって、RPC ベースのサービスが起動する前に、**rpcbind** を利用可能な状態にする必要があります。

rpcbind サービスはアクセス制御に TCP ラッパーを使用するため、**rpcbind** のアクセス制御ルールは RPC ベースの **すべての** サービスに影響します。あるいは、NFS RPC デーモンごとにアクセス制御ルールを指定することもできます。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。

9.9.1. NFS と rpcbind のトラブルシューティング

rpcbind[3] 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** を使用して現在の 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、次のコマンドを実行します。

```
# rpcinfo -p
```

例 9.7 rpcinfo -p コマンドの出力

以下に、上記コマンドの出力例を示します。

```
program vers proto   port service
    100021    1   udp  32774  nlockmgr
    100021    3   udp  32774  nlockmgr
    100021    4   udp  32774  nlockmgr
    100021    1   tcp  34437  nlockmgr
    100021    3   tcp  34437  nlockmgr
    100021    4   tcp  34437  nlockmgr
    100011    1   udp    819  rquotad
    100011    2   udp    819  rquotad
    100011    1   tcp    822  rquotad
    100011    2   tcp    822  rquotad
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100005    1   udp    836  mountd
    100005    1   tcp    839  mountd
    100005    2   udp    836  mountd
    100005    2   tcp    839  mountd
    100005    3   udp    836  mountd
    100005    3   tcp    839  mountd
```

NFS サービスの 1 つが正しく起動しないと、**rpcbind** は、そのサービスに対するクライアントからの RPC 要求を、正しいポートにマッピングできません。多くの場合は、NFS が **rpcinfo** の出力に表示されていない時に NFS を再起動すると、サービスが **rpcbind** に正しく登録され、動作を開始します。

rpcinfo の詳細情報と、その **man ページを参照してください**。

9.10. リファレンス

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、以下の資料を参照してください。

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**: 起動時にファイルシステムのマウントに使用される **/etc/fstab** ファイルのフォーマットの詳細を指定します。

- ***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。
- ***man 8 nfsidmap*** - ***nfsidmap*** cammand に従い、一般的なオプションを一覧表示します。

便利な 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://www.vanemery.com/Linux/NFSv4/NFSv4-no-rpcsec.html: Fedora Core 2 での NFSv4 の詳細を説明します。これには 2.6 カーネルが含まれます。
- http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.4086 – NFS バージョン 4 プロトコルの機能および拡張機能について記載しているホワイトペーパーです。

関連書籍

- 『Managing NFS and NIS』(Hal Stern、Mike Eisler および Ricardo Labiaga 著、O'Reilly & Associates 出版) – 2001 年の時点で利用可能な各種の 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。

---

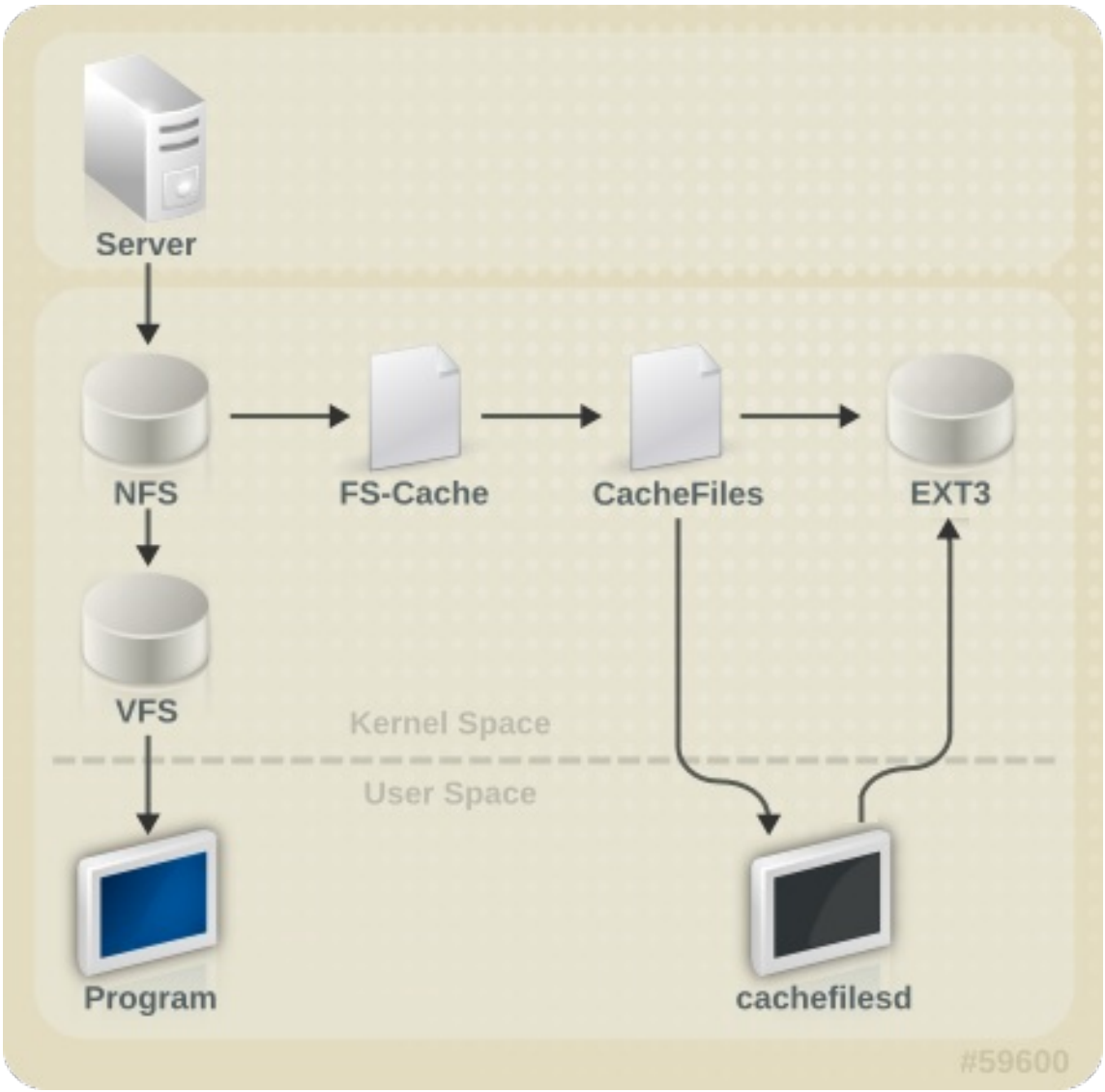
[3] **rpcbind** サービスは、**以前のバージョンの Red Hat Enterprise Linux で使用されていたポート** マップから、RPC プログラム番号の組み合わせを IP アドレスポート番号をマッピングします。詳細は、「必要なサービス」 を参照してください。

第10章 FS-CACHE

FS-Cacheは、ファイルシステムがネットワーク経由で取得したデータを取得し、ローカルディスクにキャッシュするために使用できる永続的なローカルキャッシュです。これは、ネットワーク経由でマウントされたファイルシステムからデータにアクセスするユーザーのネットワークトラフィックを最小限に抑えます(例: NFS)。

以下の図は、FS-Cache の仕組みの概要を示しています。

図10.1 FS-Cache の概要



[D]

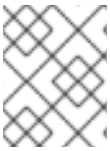
FS-Cache は、システムのユーザーおよび管理者が可能な限り透過的になるように設計されています。Solaris では **cachefs** とは異なり、サーバー上のファイルシステムは、オーバーマウントしたファイルシステムを作成せずに、クライアントのローカルキャッシュと直接対話できます。NFS では、マウントオプションにより、FS-cache が有効になっている NFS 共有をマウントするようにクライアントに指示します。

FS-Cache はネットワーク上で機能するファイルシステムの基本操作を変更せず、単にデータをキャッシュできる永続的な場所でファイルシステムを提供するだけです。たとえば、クライアントは FS-Cache が有効になっているかどうかに関わらず、NFS 共有をマウントできます。さらに、キャッシュさ

れた NFS は、ファイルを部分的にキャッシュでき、事前に完全に読み取る必要がないため、(個別にまたは集合的に) キャッシュに収まらないファイルを処理できます。また、FS-Cache は、クライアントファイルシステムドライバーからキャッシュで発生するすべての I/O エラーも非表示にします。

キャッシングサービスを提供するには、キャッシュバックエンド が必要です。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 では、キャッシュバックエンドとして **bmap** および拡張属性 (ext3 など) 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する必要があります。

FS-Cache は、ネットワークを介するかどうかに関係なく、ファイルシステムを任意にキャッシュすることはできません。共有ファイルシステムのドライバーを変更して、FS-Cache、データストレージ/検索、メタデータのセットアップと検証を操作できるようにする必要があります。FS-Cache では、永続性に対応するためにキャッシュされたファイルシステムの インデックスキー と 一貫性データ 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オブジェクトをキャッシュオブジェクトに一致させ、一貫性データを使用してキャッシュオブジェクトが有効のままかどうかを判断します。



注記

Red Hat Enterprise Linux 6.2 以降、**cachefilesd** はデフォルトでインストールされず、手動でインストールする必要があります。

10.1. パフォーマンスに関する保証

FS-Cache は、パフォーマンスの向上を保証 **しません** が、ネットワークの輻輳を回避することで一貫したパフォーマンスを保証します。キャッシュバックエンドを使用するとパフォーマンスが低下します。たとえば、キャッシュされた NFS 共有では、ネットワーク間のルックアップにディスクアクセスが追加されます。FS-Cache は可能な限り非同期となりますが、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、FS-Cache を使用して、他の方法では負荷のない GigE ネットワークを介して 2 台のコンピューター間の NFS 共有をキャッシュしても、ファイルアクセスのパフォーマンスが向上することはありません。代わりに、NFS 要求はローカルディスクからではなく、サーバーメモリーより早く満たされます。

したがって、FS-Cacheの使用は、さまざまな要因における **妥協** です。たとえば、NFS トラフィックのキャッシュにFS-Cache を使用すると、クライアントは多少遅くなりますが、ネットワークの帯域幅を消費せずにローカルに読み取り要求を満たすことでネットワークおよびサーバーの読み込み負荷が大幅に削減されます。

10.2. キャッシュの設定

現在、Red Hat Enterprise Linux 6 は **cachefiles** キャッシュバックエンドのみを提供します。**cachefilesd** デーモンは **cachefiles** を開始し、管理します。**/etc/cachefilesd.conf** ファイルは、**cachefiles** によるキャッシュサービスの提供方法を制御します。このタイプのキャッシュバックエンドを設定するには、**cachefilesd** パッケージがインストールされている必要があります。

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。これは、次のパラメーターを使用して設定します。

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

一般的に、キャッシュバックエンドディレクトリーは、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にキャッシュを保存します。ラップトップでは、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が推奨されますが、デスクトップマシンの場合は、キャッシュ専用のディスクパーティションをマウントするより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、以下のファイルシステムの Red Hat Enterprise Linux 6 実装が含まれます。

- ext3 (拡張属性が有効)
- ext4
- BTRFS
- XFS

ホスト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 はこの属性を使用して、整合性のメンテナンス情報を保存します。ext3 ファイルシステム (つまり **device**) のユーザー定義の拡張属性を有効にするには、次のコマンドを実行します。

```
# 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、キャッシュをホストしているパーティション上の一定の空き領域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を増大および縮小し、root ファイルシステム (ラップトップなど) で安全に使用できるようにします。FS-Cache ではこの動作でデフォルト値を設定します。これは、キャッシュカリング制限 で設定できます。キャッシュカリング制限の設定に関する詳細は、「キャッシュカリング制限の設定」 を参照してください。

設定ファイルを置いたら、**cachefilesd** サービスを起動します。

```
# service cachefilesd start
```

起動時に **cachefilesd** が起動するように設定するには、root で次のコマンドを実行します。

```
# chkconfig cachefilesd on
```

10.3. NFS でのキャッシュの使用

明示的に指示されない限り、NFS はキャッシュを使用しません。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いていない限り**、**/mount/point** の下にあるファイルへのアクセスはすべてキャッシュを通過します（詳細は、「NFS でのキャッシュの制限」 を参照してください）。NFS インデックスは NFS ファイルハンドルを使用してコンテンツをキャッシュします。ファイル名 **ではなく**、ハードリンクされたファイルはキャッシュを正しく共有します。

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

10.3.1. キャッシュの共有

NFS キャッシュの共有には潜在的な問題がいくつかあります。キャッシュは永続的であるため、キャッシュ内のデータブロックは 4 つのキーのシーケンス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: 一部のマウントオプション、セキュリティータイプ、FSID、識別子
- レベル3: ファイルハンドル
- レベル4: ファイル内のページ番号

スーパーブロック間での整合性の管理に関する問題を回避するには、データをキャッシュするすべての NFS のスーパーブロックに固有のレベル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 マウントはスーパーブロックを共有しているため、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ュを共有することになります。

> ***例10.1 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*** および ***/home/jim*** には同じオプションがあるため、スーパーブロックを共有する可能性が高くなります。特に NFS サーバー上の同じボリュームやパーティションから作成されている場合は共有する可能性が高くなります (***home0***)。ここで、2 つの後続のマウントコマンドを示します。
>
> ***mount home0:/disk0/fred /home/fred -o fsc,rsize=230***
>
> ***mount home0:/disk0/jim /home/jim -o fsc,rsize=231***
>
> この場合、***/home/fred*** と ***/home/jim*** は、レベル2 キー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。次のマウントシーケンスについても同じことが言えます。
>
> ***mount home0:/disk0/fred /home/fred1 -o fsc,rsize=230***
>
> ***mount home0:/disk0/fred /home/fred2 -o fsc,rsize=231***
>
> 上記の2 つのサブツリー (***/home/fred1*** と ***/home/fred2***) は **2 回** キャッシュされます。
>
> スーパーブロックの共有を回避するもう1 つの方法は、***nosharecache*** パラメーターで明示的に共有を回避することです。同じ例を使用します。
>
> ***mount home0:/disk0/fred /home/fred -o nosharecache,fsc***
>
> ***mount home0:/disk0/jim /home/jim -o nosharecache,fsc***
>
> ただし、この場合、***home0:/disk0/fred*** および ***home0:/disk0/jim*** のレベル2 キーを区別するものがないため、スーパーブロックの1 つだけがキャッシュの使用を許可されます。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***)。以

> 下に例を示します。
>
> **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。

10.3.2. 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を回避します。これは、この種のアクセスが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。

したがって、ダイレクト I/O または書き込みのいずれかに対して共有ファイルシステムからファイルを開くと、キャッシュされたファイルのコピーがフラッシュされます。ダイレクト I/O や書き込みのためにファイルが開かれなくなるまで、FS-Cache はファイルを再キャッシュしません。

さらに、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、シンボリックリンク、デバイスファイル、FIFO、ソケットを**キャッシュしません**。

10.4. キャッシュカリング制限の設定

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ュして、ディスクの領域を解放することで機能します。これにより、利用可能な空き領域がすべて消費される可能性があり、ディスクがルートパーティションも格納している場合は問題に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 (つまり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して、一定量の空き領域を維持しようとします。この動作は キャッシュカリング と呼ばれます。

キャッシュカリングは、基盤となるファイルシステムで使用可能なブロックのパーセンテージとファイルのパーセンテージに基づいて行われます。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と、カリングはオフになります。

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と、カリング動作が開始します。

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)

キャッシュ内の使用可能な領域または使用可能なファイルの数がこの制限のいずれかを下回ると、カリングによってこれらの制限を超える状態になるまで、ディスク領域またはファイルのそれ以上の割り当ては許可されません。

各設定の **N** のデフォルト値は以下の通りです。

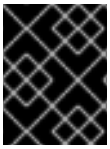
- ***brun/frun*** - 10%
- ***bcull/fcull*** - 7%
- ***bstop/fstop*** - 3%

この設定を行う場合は、以下の条件を満たす必要があります。

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

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

*これは、空き領域と利用可能なファイルの割合であり、100 から、**df** プログラムで表示される割合を引いたものではありません。*



重要

カリングは、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。

10.5. 統計情報

FS-Cache は一般的な統計情報も追跡します。この情報を表示するには、次を使用します。

cat /proc/fs/fscache/stats

FS-Cache の統計にはディシジョン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 によって提供される統計の詳細は、以下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

10.6. リファレンス

***cachefilesd** の詳細および設定方法は、**man cachefilesd** および **man cachefilesd.conf** を参照してください。その他にも、以下のカーネルドキュメントを参照してください。*

- ***/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、利用可能な統計、機能など、FS-Cache に関する一般的な情報は、以下のカーネルドキュメントを参照してください。

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

パート II. ストレージ管理

「ストレージ管理」のセクションは、Red Hat Enterprise Linux 6 におけるストレージに関する考慮事項から始まります。パーティション、論理ボリューム管理、およびスワップパーティションに関する手順は次のとおりです。ディスククォータ、RAID システムの横には、mount コマンド、volume_key、および acls の関数が続きます。SSD のチューニング、書き込みバリア、I/O 制限およびディスクレスシステムは、これに従います。Online Storage の大きな章は次に行われ、最後にデバイスマッパーのマルチパスおよび仮想ストレージを終了させます。

以下のコンテンツ表を使用して、これらのストレージ管理タスクを確認します。

第11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 6 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。Red Hat が提供する『『インストールガイド』』 を参照してください（インストール時のストレージ設定を含む）。

11.1. インストール中のストレージ設定の更新

Red Hat Enterprise Linux 6 では、以下の設定/デバイスのインストール設定が更新されました。

イーサネット上ファイバーチャネル (FCoE)

Anaconda で、インストール時に FCoE ストレージデバイスを設定できるようになりました。

ストレージデバイスのフィルタリングインターフェース

Anaconda で、インストール時に使用されるストレージデバイスの制御が改善されました。システムストレージに実際に使用されているデバイスや、インストーラーに利用可能なデバイス、または表示可能なデバイスを制御できるようになりました。デバイスのフィルタリングには、以下の 2 つのパスがあります。

基本パス

ローカルで割り当てられたディスクおよびファームウェアの RAID アレイのみをストレージデバイスとして使用するシステムの場合は、

高度なパス

SAN（マルチパス、iSCSI、FCoE）デバイスを使用するシステムの場合は、

自動パーティション設定および /home

自動パーティション設定では、LVM 物理ボリュームの割り当てに 50 GB 以上が利用可能な場合に、**/home** ファイルシステム用に別の論理ボリュームが作成されます。ルートファイルシステム(/)は、別の **/home** 論理ボリュームを作成する際に最大 50 GB に制限されますが、**/home** 論理ボリュームは、ボリュームグループの残りの領域をすべて占有します。

11.2. サポートされるファイルシステムの概要

本セクションでは、Red Hat Enterprise Linux 6 でサポートされる各ファイルシステムの基本的な技術情報を説明します。

表11.1 サポートされるファイルシステムの技術仕様

ファイルシステム	サポートされる最大サイズ	最大ファイルオフセット	最大サブディレクトリー（ディレクトリーごと）	Max Depth of Symbolic Links	ACL サポート	詳細
Ext2	8TB	2TB	32,000	8	はい	該当なし

ファイルシステム	サポートされる最大サイズ	最大ファイルオフセット	最大サブディレクトリー（ディレクトリーごと）	Max Depth of Symbolic Links	ACL サポート	詳細
Ext3	16TB	2TB	32,000	8	はい	5章Ext3 ファイルシステム。
Ext4	16TB	16TB[a]	無制限[b]	8	はい	6章Ext4 ファイルシステム
XFS	100TB	100TB[c]	無制限	8	はい	8章XFS ファイルシステム

[a] この最大ファイルサイズは、64 ビットのマシンに基づいています。32 ビットマシンでは、最大ファイルサイズは 8TB です。

[b] リンク数が 65,000 を超える場合は、リセットされ 1 にリセットされ、増加しなくなりました。

[c] この最大ファイルサイズは、64 ビットのマシンのみになります。Red Hat Enterprise Linux は、32 ビットマシンで XFS に対応していません。

注記

一覧表示されたファイルおよびファイルシステムサイズは、Red Hat がテストおよびサポートするものになります。理論上の上限は考慮に入れられません。

サポートされる最大サイズと最大ファイルオフセット列は、4k ブロックを想定しています。

11.3. 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**、および **/usr/local** は別々のデバイスに配置します。これにより、ユーザーおよびアプリケーションデータを保存しながら、オペレーティングシステムを含むデバイス/ファイルシステムを再フォーマットできます。

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、これは単純に、起動コマンドラインか、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) を記載することを意味します。

zFCP デバイスの場合は、デバイス番号、論理ユニット番号 (LUN)、および 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 を使用したブロックデバイスの暗号化

LUKS/**dm-crypt** を使用し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が存在する場合はそのデバイスを選択してください。次に、新しいシステムのストレージ設定をインストールプロセスの一部としてアクティブにします。

古い BIOS RAID メタデータ

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

警告

ディスクから RAID メタデータを削除または消去すると、保存データがすべて破棄される可能性があります。Red Hat では、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。

ディスクから RAID メタデータを削除するには、以下のコマンドを使用します。

dmraid -r -E /device/

RAID デバイスの管理の詳細は、**man dmraid** および 17章RAID (Redundant Array of Independent Disks) を参照してください。

iSCSI の検出および設定

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 ターゲットの 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 の検出および設定

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 を有効にすると、ブロックデバイスが汎用のブロックデバイスとして使用されているとエラーが発生します。DIF/DIX チェックサムの計算後はバッファーされたデータが上書きされないようにするためのインターロックがバッファー書き込みパス内にないため、バッファーされた入出力または **mmap(2)** ベースの入出力が正常に動作しなくなります。*

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

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

*DIF/DIX チェックサムの計算後に入出力データが変更し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12章 ファイルシステムの確認

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用して修復を実行することができます。このツールは、通常 **fsck** ツールと呼ばれることが多く、**fsck** は、ファイルシステムチェック を短くした名前になります。

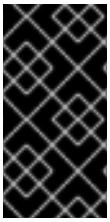
注記

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

ファイルシステムの不整合は、ハードウェアエラー、ストレージ管理エラー、ソフトウェアバグなどのさまざまな理由で発生する可能性があります。

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。その結果、ファイルシステムの確認は、通常、起動時に **/etc/fstab** に記載されている各ファイルシステムで実行されます。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。この場合は、ファイルシステム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にベストプラクティスおよび他の有用な情報を提供します。



重要

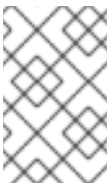
マシンが起動しない場合、ファイルシステムが極めて大きい場合、またはファイルシステムがリモートストレージにある場合を除き、Red Hat はファイルシステムチェックの無効化を推奨しません。**/etc/fstab** の 6 番目のフィールドを 0 に設定すると、システムの起動時にファイルシステムの確認を無効にできます。

12.1. FSCK のベストプラクティス

通常、ファイルシステムの確認および修復のツールを実行すると、検出された不整合の少なくとも一部が自動的に修復されることが期待できます。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きな変更が加えられ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

ほとんどのファイルシステムチェッカーは、ファイルシステムを修復しない操作モードを持ちます。このモードでは、チェッカーは、実際にファイルシステムを変更せずに、作成したアクションを見つけたエラーおよび操作を出力します。

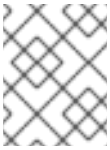


注記

整合性の確認後のフェーズでは、修復モードで実行していた場合に前のフェーズで修正されていた不整合が検出される可能性があるため、追加のエラーが出力される場合があります。

ファイルシステムイメージで最初に操作

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、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は、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にファイルシステムイメージを保存します。

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でのみ操作

ファイルシステムの修復は、マウント解除されたファイルシステムでのみ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は、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確認ツールは、ハードウェアの問題を修復できません。修復を正常に動作させるには、ファイルシステムが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。ハードウェアエラーが原因でファイルシステムが破損した場合は、まず **dd(8)** ユーティリティーなどを使用して、ファイルシステムを適切なディスクに移動する必要があります。*

12.2. FSCK のファイルシステム固有の情報

12.2.1. ext2、ext3、および ext4

*これらのエラーはすべて、**e2fsck** バイナリーを使用して、ファイルシステムの確認と修復を実行します。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、システムの起動時に自動的に実行し、その動作は確認されるファイルシステムと、そのファイルシステムの状態によって異なります。*

完全なファイルシステムの確認および修復は、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対して呼び出されます。

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。ジャーナルの再生により、クラッシュ後の一貫したファイルシステムが保証されるため、これはデフォルトのアクションです。

*マウント中にこれらのファイルシステムがメタデータの不整合が発生した場合には、このファクトがファイルシステムのスーパーブロックに記録されます。**e2fsck** が、このようなエラーでファイルシステムがマークされていることを検出すると、ジャーナル（ある場合）を再生した後、**e2fsck** が完全な*

チェックを実行します。

e2fsck は、**-p** オプションが指定されていないと、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。チェックのみの操作。

-b スーパーブロック

プライマリースーパーブロックが損傷している場合は、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くても、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。ファイルシステム自体と同じサイズのスパースファイルを作成するには、- **r** オプションは、テスト目的に使用する必要があります。その後、**e2fsck** は作成されたファイルで直接操作できます。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。

12.2.2. XFS

ブート時に修復が自動的に行なわれません。ファイルシステムの確認または修復を開始するには、**xfs_repair** ツールを使用します。

注記

*xfsprogs パッケージには **fsck.xfs** バイナリーが存在しますが、これは、システムの起動時に **fsck.filesystem** バイナリーを検索する initscripts を満たすためにのみ存在します。**fsck.xfs** は、すぐに終了コード 0 で終了します。*

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

*ファイルシステム上のクリーンログは **xfs_repair** が動作するために必要です。ファイルシステムがクリーンな状態でアンマウントされなかった場合は、**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更新を破棄し、さらに不整合が生じます。*

***-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。このオプションを指定すると、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

非変更モードです。チェックのみの操作。

-L

ゼロメタデータログ。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. スーパーブロックチェック。

これらのフェーズ、および操作中に出力されるメッセージは、man **ページの xfs_repair(8)** の深度に文書化されています。

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。

12.2.3. Btrfs

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

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

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

1. エクステントのチェック。

2. ファイルシステムの root チェック

3. ルートの参照数のチェック。

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

第13章 PARTITIONS

ユーティリティー **parted** を使用すると、ユーザーは以下を行うことができます。

- 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

デフォルトでは、Red Hat Enterprise Linux のインストール時に **parted** パッケージが含まれています。**parted** を起動するには、root としてログインし、シェルプロンプトで **parted /dev/sda** コマンドを実行します（**/dev/sda** は設定するドライブのデバイス名です）。

パーティションの削除またはサイズ変更を行う場合は、パーティションが存在するデバイスが使用中でない必要があります。使用中のデバイスに新しいパーティションを作成することは推奨されませんが、推奨されません。

デバイスを使用しないようにするには、デバイスのパーティションを何もマウントできません。また、デバイスの swap 領域は有効にしないでください。

また、カーネルが変更を適切に認識しないため、使用中の間にパーティションテーブルは変更しないでください。パーティションテーブルがマウントされたパーティションの実際の状態と一致しない場合は、情報を誤ったパーティションに書き込む可能性があります。その結果、データが損失され、上書きされる可能性があります。

これにより、レスキューモードでシステムを起動するのが最も簡単な方法です。ファイルシステムをマウントするように指示されたら、**スキップ** を選択します。

または、ドライブに使用中のパーティション（ファイルシステムのマウント解除されないようにファイルシステムを使用するシステムプロセスまたはロック解除）が含まれていない場合は、**umount** コマンドを使用してアンマウントし、**swapoff** コマンドを使用して、ハードドライブ上のスワップ領域をすべてオフにできます。

表13.1「**parted** コマンド」 一般的に使用される **parted** コマンドの一覧が含まれています。これ以降のセクションでは、これらのコマンドおよび引数について詳しく説明します。

表13.1 parted コマンド

コマンド	詳細
check minor-num	ファイルシステムの簡単なチェックを実行する
cp from to	ファイルシステムをあるパーティションから別のパーティションにコピーします。from と to はパーティションのマイナー番号です。
ヘルプ	利用可能なコマンドの一覧を表示します
mklabel label	パーティションテーブル用のディスクラベルを作成します
mkfs minor-num file-system-type	タイプの file-system-type のファイルシステムを作成します。

コマンド	詳細
mkpart part-type fs-type start-mb end-mb	新しいファイルシステムを作成せずに、パーティションを作成します
mkpartfs part-type fs-type start-mb end-mb	パーティションを作成し、指定したファイルシステムを作成します。
move minor-num start-mb end-mb	パーティションを移動します
name minor-num name	Mac と PC98 のディスクラベル用のみのパーティションに名前を付けます
print	パーティションテーブルを表示します
quit	**parted** を終了します
rescue start-mb end-mb	start-mb から end-mb へ、消失したパーティションを復旧します。
resize minor-num start-mb end-mb	パーティションを start-mb から end-mb に変更します。
rm minor-num	パーティションを削除
select device	設定する別のデバイスを選択します
set minor-num flag state	パーティションにフラグを設定します。state はオンまたはオフのいずれかになります
toggle [NUMBER [FLAG]	パーティション NUMBER 上の FLAG の状態を切り替えます
unit UNIT	デフォルトのユニットを UNIT に設定します

13.1. パーティションテーブルの表示

parted の起動後、コマンドが **印刷して** パーティションテーブルを表示します。以下のような表が表示されます。

> **例13.1 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、モデル番号およびインターフェースが含まれ、2 番目の行はディスクラベルの種類を表示します。4 行目の下の残りの出力には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の マイナー 番号は、パーティション **number** です。たとえば、マイナー番号1 のパーティションは、/**dev**/**sda1** に対応します。**Start** および **End** の値はメガバイトです。有効な **Type** はメタデータ、フリー、プライマリー、拡張、または論理です。**Filesystem** はファイルシステムのタイプで、次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** に値が表示されない場合は、そのファイルシステムタイプが不明であることを示します。*

***フラグ** は、パーティションに設定したフラグを一覧表示しています。利用可能なフラグは、boot、root、swap、hidden、raid、lvm、または lba です。*

注記

***parted** を再起動することなく別のデバイスを選択するには、**select** コマンドの後にデバイス名（例：/**dev**/**sda**）を指定します。そうすることで、デバイスのパーティションテーブルの表示と設定を行うことができます。*

13.2. 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13.1 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** を起動します。**/dev/sda** は、パーティションを作成するデバイスに置き換えます。

```
# 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を確認します。

```
# print
```

十分な空き容量がない場合は、既存のパーティションのサイズを変更できます。詳細は、「パーティションのサイズ変更」 を参照してください。

13.2.1. 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は、Red Hat Enterprise Linux 6 『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。ただし、**parted** は ext3 ファイルシステムの作成に対応していません。したがって、ext3 ファイルシステムを作成する場合は、m **kpart を使用して、**後述した **mkfs** コマンドで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** コマンドを使用して、正しいパーティションタイプ、ファイルシステムタイプ、およびサイズで、パーティションテーブルにあることを確認します。新しいパーティションのマイナー番号も覚えておいてください。そのパーティション上のファイルシステムにラベルを付けることができます。また、parted が閉じられた後に **cat /proc/partitions** の出力も表示し、カーネルが新しいパーティションを認識するようにする必要があります。

parted が作成するパーティションの最大数は 128 です。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。

13.2.2. 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13.2 パーティションのフォーマットとラベル付け

1. このパーティションには、引き続きファイルシステムがありません。以下のコマンドを実行してこれを作成します。

   # /sbin/mkfs -t ext3 /dev/sda6

警告

パーティションをフォーマットすると、そのパーティションに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割り当てます。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、root でマウントポイント（例： **/work**）を作成します。

13.2.3. /etc/fstab に追加

root として、**/etc/fstab** ファイルを編集して、パーティションの UUID で新規パーティションを含めます。パーティションの UUID の完全なリストを表示するには、**blkid -o list** コマンドを使用し、個別のデバイス詳細を表示するには、**blkid device** コマンドを使用します。

最初の列には、**UUID=** の後にファイルシステムの UUID が含まれている必要があります。2 列には新しいパーティションのマウントポイントが含まれ、次の列はファイルシステムのタイプである必要があります（ext3 や swap など）。形式の詳細情報が必要な場合は、man コマンド man **fstab で man ページを参照してください。**

4 番目のコラムが **defaults** という単語で、パーティションは起動時にマウントされます。再起動せずにパーティションをマウントするには、root で以下のコマンドを入力します。

mount /work

13.3. 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13.3 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** を起動します。**/dev/sda** は、パーティションを削除するデバイスに置き換えます。

```
# 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** コマンドを使用して、パーティションテーブルから削除されていることを確認します。また、パーティションが削除されるように、**/proc/partitions** の出力も表示する必要があります。

```
# cat /proc/partitions
```

6. 最後の手順では、**/etc/fstab** ファイルから削除します。削除したパーティションを宣言している行を見つけ、ファイルから削除します。

13.4. パーティションのサイズ変更

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、サイズを変更しないでください。

手順13.4 パーティションのサイズ

1. パーティションのサイズを変更する前に、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** を起動します。**/dev/sda** は、パーティションのサイズを変更するデバイスです。

```
# parted /dev/sda
```

3. 現在のパーティションテーブルを表示して、サイズを変更するパーティションのマイナー番号と、パーティションの開始点と終了点を確認します。

```
# print
```

4. パーティションのサイズ変更には、**サイズ変更** コマンドの後にパーティションのマイナー番号、メガバイトの開始場所、および終了場所（メガバイト単位）を使用します。

例13.2 パーティションのサイズ

以下に例を示します。

```
resize 3 1024 2048
```

警告

パーティションをデバイスで利用可能な領域よりも大きくすることはできません。

5. パーティションのサイズ変更後に、**print** コマンドを使用して、パーティションのサイズが正しく変更されており、正しいパーティションタイプで、ファイルシステムのタイプが正しいことを確認します。

6. システムを通常モードに再起動したら、**df** コマンドを使用して、パーティションがマウントされ、新しいサイズで認識されるようにします。

第14章 LVM (論理ボリュームマネージャー)

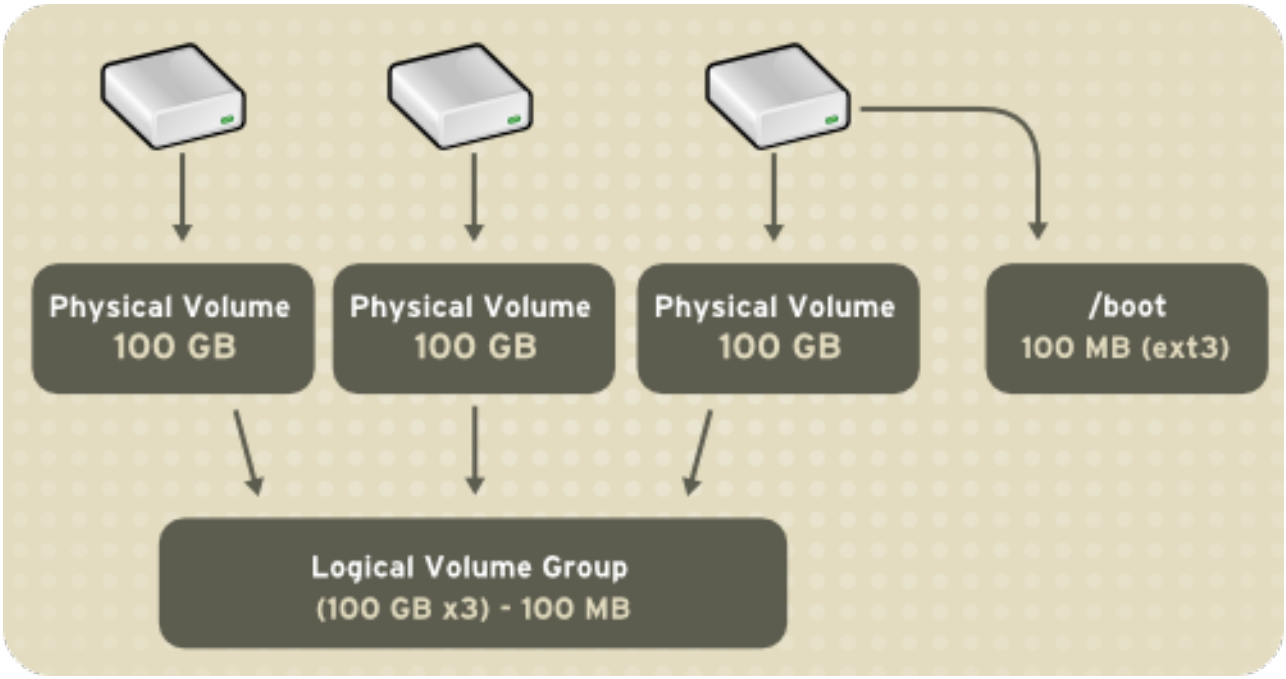
LVM は、論理ボリュームの割り当て、削除、ミラーリング、およびサイズ変更を含む論理ボリューム管理用のツールです。

LVM では、ハードドライブまたはハードドライブセットが1 つ以上の 物理ボリューム に割り当てられます。LVM 物理ボリュームは、2 つ以上のディスクにまたがる可能性のある他のブロックデバイスに配置できます。

物理ボリュームは、**/boot/** パーティションを除き、論理ボリューム に統合されます。ブートローダーが読み取れないため、**/boot/** パーティションは論理ボリューム上に存在できません。root(/)**パーティションが**論理ボリュームにある場合は、ボリュームグループの一部ではない別の **/boot/** パーティションを作成します。

物理ボリュームは複数のドライブを経由できないため、複数のドライブにまたがるには、ドライブごとに1 つまたは複数の物理ボリュームを作成してください。

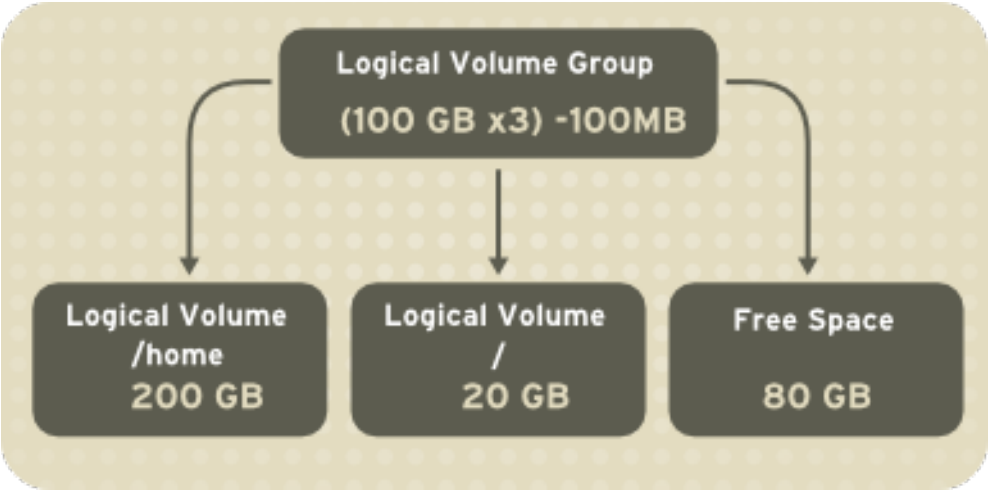
図14.1 論理ボリューム



[D]

ボリュームグループは 論理ボリューム に分割できます。論理ボリュームには、**/home**、/、**ファイルシステムタイプ**（ext2、ext3 など）などのマウントポイントが割り当てられます。「パーティション」がフル容量に達すると、ボリュームグループの空き領域を論理ボリュームに追加して、パーティションのサイズを大きくすることができます。新しいハードドライブがシステムに追加されると、そのドライブをボリュームグループに追加できます。また、論理ボリュームであるパーティションのサイズは増やすことができます。

図14.2 論理ボリューム



[D]

一方、ext3 ファイルシステムでシステムパーティションを作成すると、ハードドライブは定義されたサイズのパーティションに分割されます。パーティションが満杯になると、パーティションのサイズを拡張できません。パーティションが別のハードドライブに移動された場合でも、元のハードドライブの領域を別のパーティションとして再割り当てするか、使用されていないようにする必要があります。

重要

*LVM/LVM2 の本章は、LVM GUI 管理ツール（**system-config-lvm** など）の使用に焦点を当てています。クラスターおよび非クラスターストレージでの LVM パーティションの作成および設定に関する包括的な情報は、Red Hat が提供する 『*Logical Volume Manager Administration*』 ガイドを参照してください。*

また、Red Hat Enterprise Linux 6 の 『インストールガイド』 では、インストール時に LVM 論理ボリュームを作成および設定する方法が説明されています。詳細は、Red Hat Enterprise Linux 6 の 『『インストールガイド』の「』 『LVM 論理ボリュームの作成』 」セクションを参照してください。

14.1. LVM2 とは

LVM バージョン 2 または LVM2 は、Red Hat Enterprise Linux 5 ではデフォルトで、2.6 カーネルに含まれるデバイスマッパードライバーを使用します。LVM2 は、2.4 カーネルを実行している Red Hat Enterprise Linux のバージョンからアップグレードできます。

14.2. SYSTEM-CONFIG-LVM の使用

LVM ユーティリティーを使用すると、X ウィンドウ内で論理ボリュームを管理したり、グラフィカルに管理できます。これは事前にインストールされているわけではないため、最初に実行するようにインストールします。

```
# yum install system-config-lvm
```

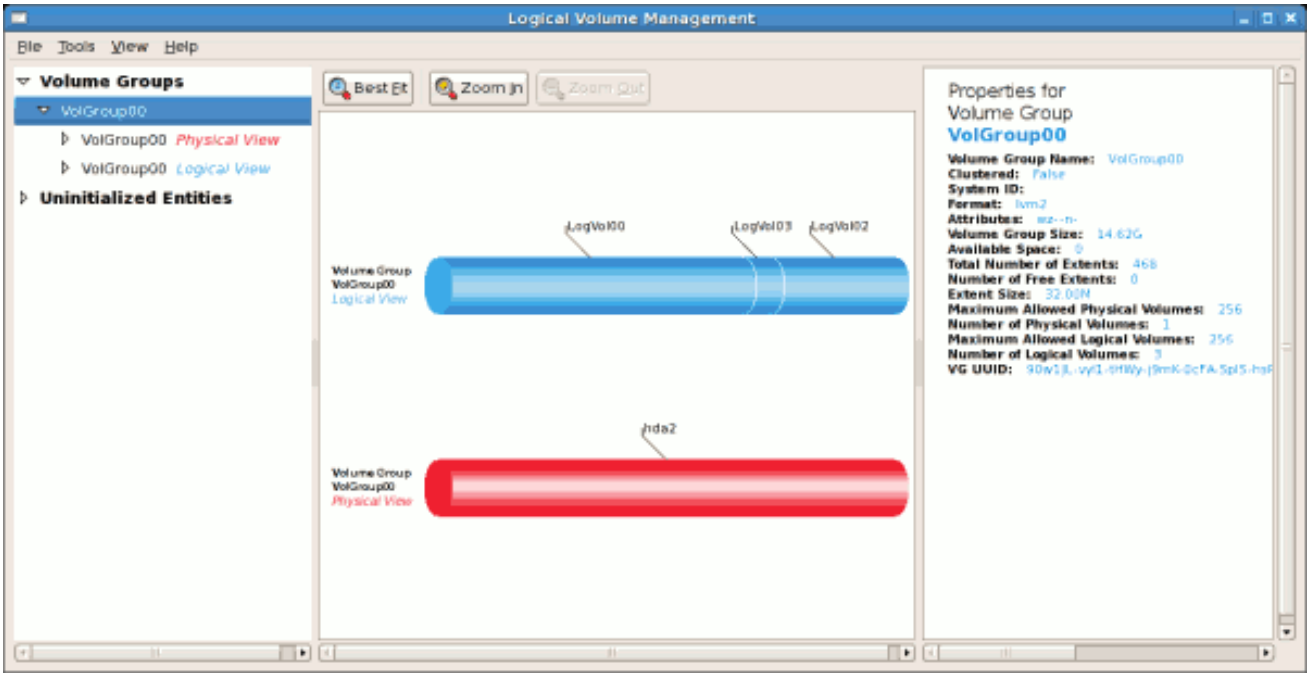
*次に、メニューパネル(**System → Administration → Logical Volume Management**)から選択すると、アプリケーションにアクセスできます。別の方法として、論理ボリューム管理ユーティリティーを開始するには、端末から **system-config-lvm** を入力します。*

このセクションの例では、インストール中に作成されたボリュームグループの詳細を以下に示します。

> **例14.1 インストール時のボリュームグループの作成**
>
> > /boot - (Ext3) file system. Displayed under 'Uninitialized Entities'. (DO NOT initialize this partition).
> > LogVol00 - (LVM) contains the (/) directory (312 extents).
> > LogVol02 - (LVM) contains the (/home) directory (128 extents).
> > LogVol03 - (LVM) swap (28 extents).

上記の論理ボリュームは、ディスクエンティティー **/dev/hda2 に作成されており、/ boot が / dev/hda1** に作成されました。システムは、例14.2「初期化されていないエントリー」 で説明されている「初期化のないエンティティー」で構成されています。以下の図は、LVM ユーティリティーの主なウィンドウを示しています。上記の設定の論理および物理ビューを以下に示します。3 つの論理ボリュームが同じ物理ボリューム(hda2)に存在します。

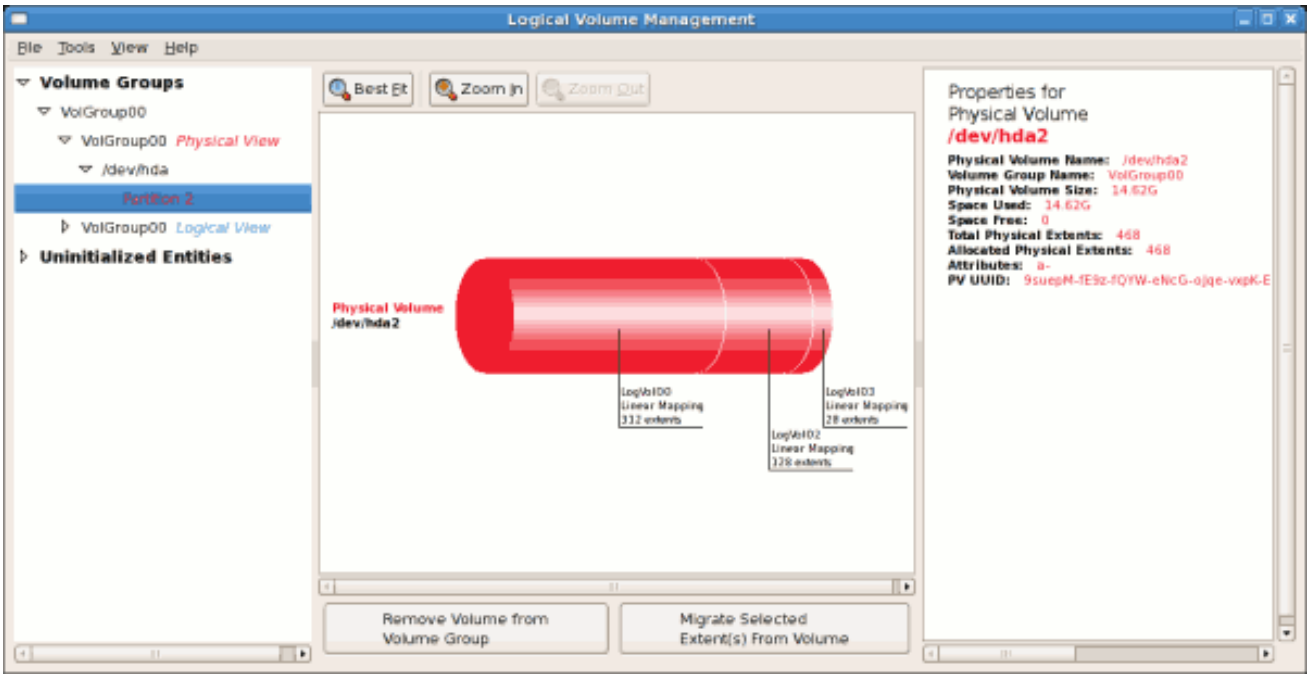
図14.3 メインのLVM ウインドウ



[D]

以下の図は、ボリュームの物理ビューを示しています。このウィンドウでは、ボリュームグループからボリュームを選択し、削除したり、ボリュームから別のボリュームグループにエクステントを移行することもできます。エクステントを移行する手順については、図14.10「エクステントの移行」 を参照してください。

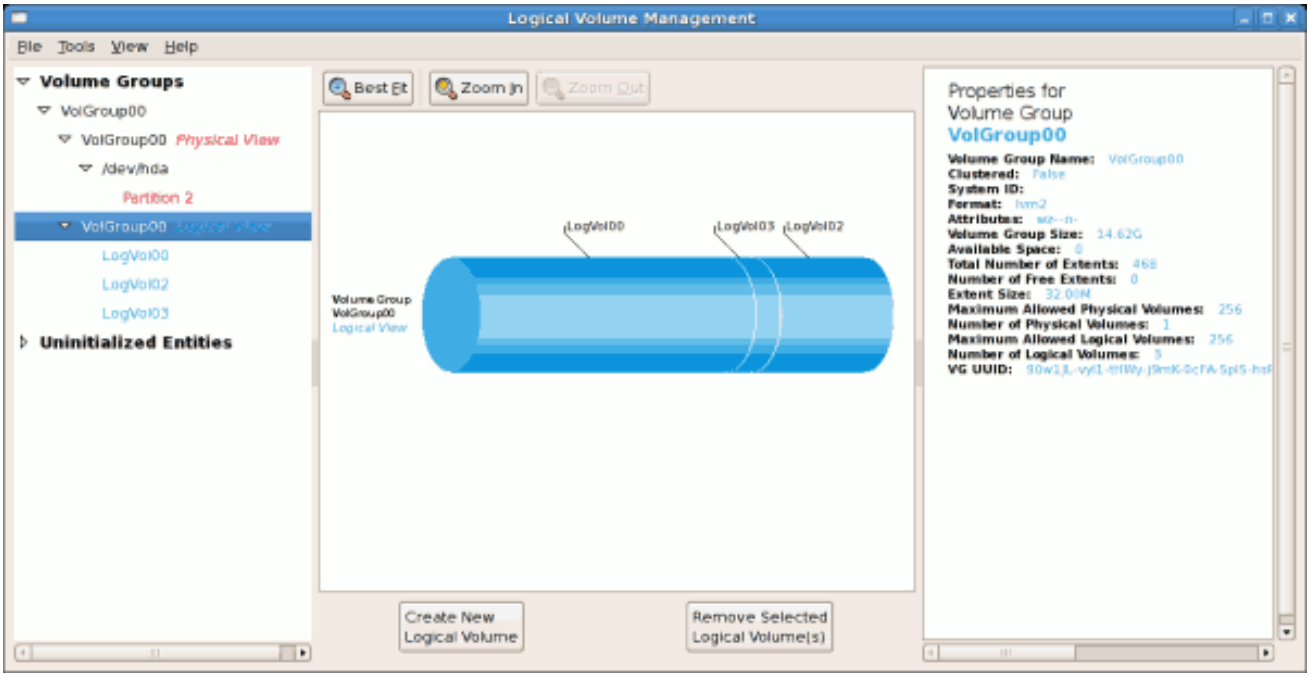
図14.4 物理ビューのウィンドウ



[D]

以下の図は、選択したボリュームグループの論理ビューを示しています。各論理ボリュームのサイズも示されています。

図14.5 論理表示ウィンドウ

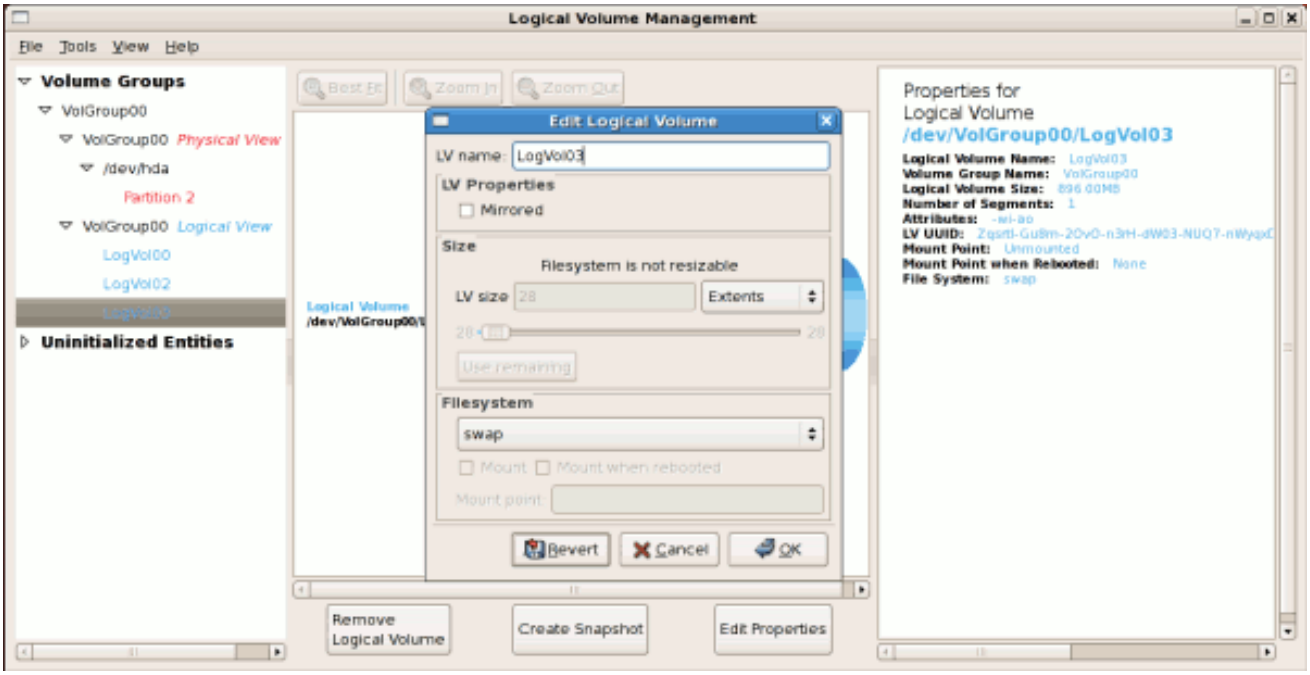


[D]

*左側の列で、ボリュームグループで個別の論理ボリュームを選択して、それぞれの詳細を表示できます。この例では、目的は 'LogVol03' の論理ボリューム名の名前を 'Swap' に変更することです。この操作を実行するには、（イメージではなく）各論理ボリュームを選択して、**プロパティーの編集** ボタンをクリックします。論理ボリュームの編集 ウィンドウが表示され、論理ボリューム名、サイズ（エクステント、ギガバイト、メガバイト、またはキロバイト）を変更し、論理ボリュームグループで利用可能な残りの領域も使用できます。以下の図はこの図を示しています。*

ボリュームグループに現在空き領域がないため、この論理ボリュームはサイズで変更できません。残りの領域がある場合は、このオプションが有効になります（ 図14.17「論理ボリュームの編集」 を参照してください）。**OK** ボタンをクリックして変更を保存します（これによりボリュームが再マウントされます）。変更をキャンセルするには、**Cancel** ボタンをクリックします。最後のスナップショット設定に戻すには、Rever **t** ボタンをクリックします。スナップショットは、LVM ユーティリティーウィンドウの **スナップショットの作成** ボタンをクリックして作成できます。選択した論理ボリュームがシステムで使用されている場合、たとえば、このタスクはアンマウントできないため、このタスクは成功しません。

図14.6 論理ボリュームの編集

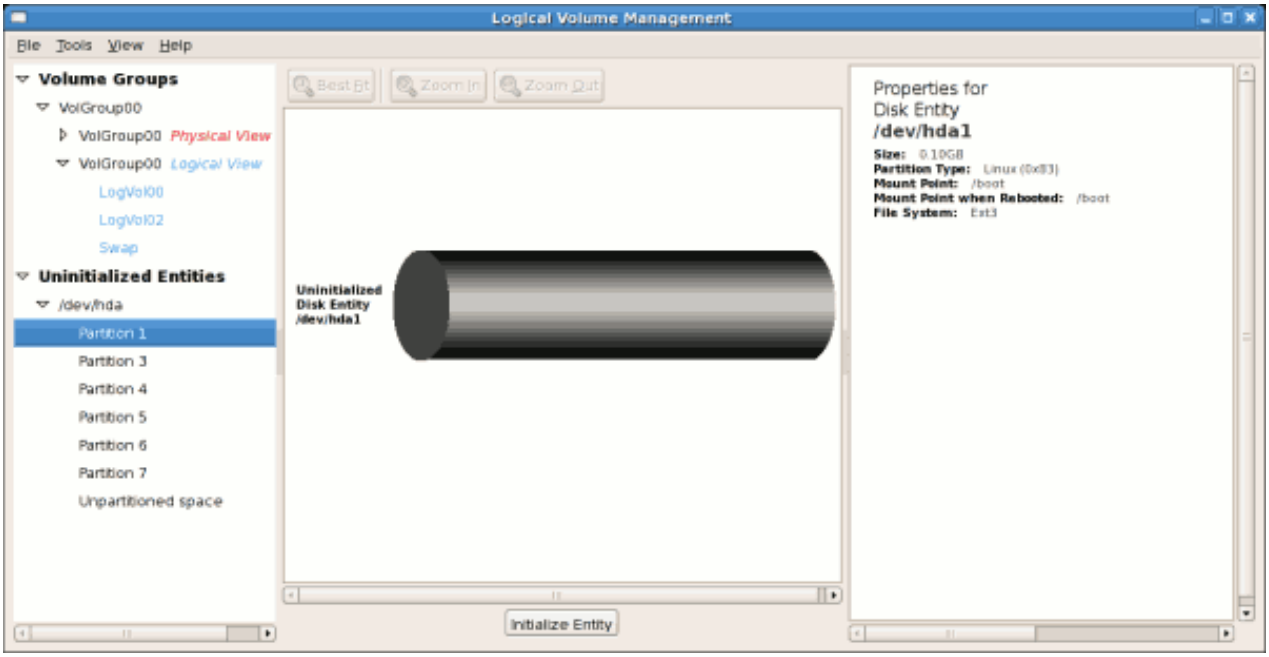


[D]

14.2.1. 初期化されていないエンタイトルメントの使用

「uninitialized Entities」は、パーティション分割されていない領域と LVM ファイルシステム以外で構成されます。この例では、インストール時にパーティション 3、4、5、6、および 7 が作成され、パーティションの分割されていない領域がハードディスクに残されています。各パーティションを表示し、ウィンドウの右側のコラムで「Properties for Disk Entity」を読み取り、重要なデータを削除しないようにします。この例では、パーティション 1 は **/boot** であるため初期化できません。初期化されていないエンティティーを以下に示します。

例14.2 初期化されていないエントリー



[D]

この例では、パーティション3 が初期化され、既存のボリュームグループに追加されます。パーティションまたは分割されていない領域を初期化するには、パーティションを選択して ***Initialize Entity*** *ボタンをクリックします。初期化されると、ボリュームは「未割り当てのボリューム」一覧に表示されます。*

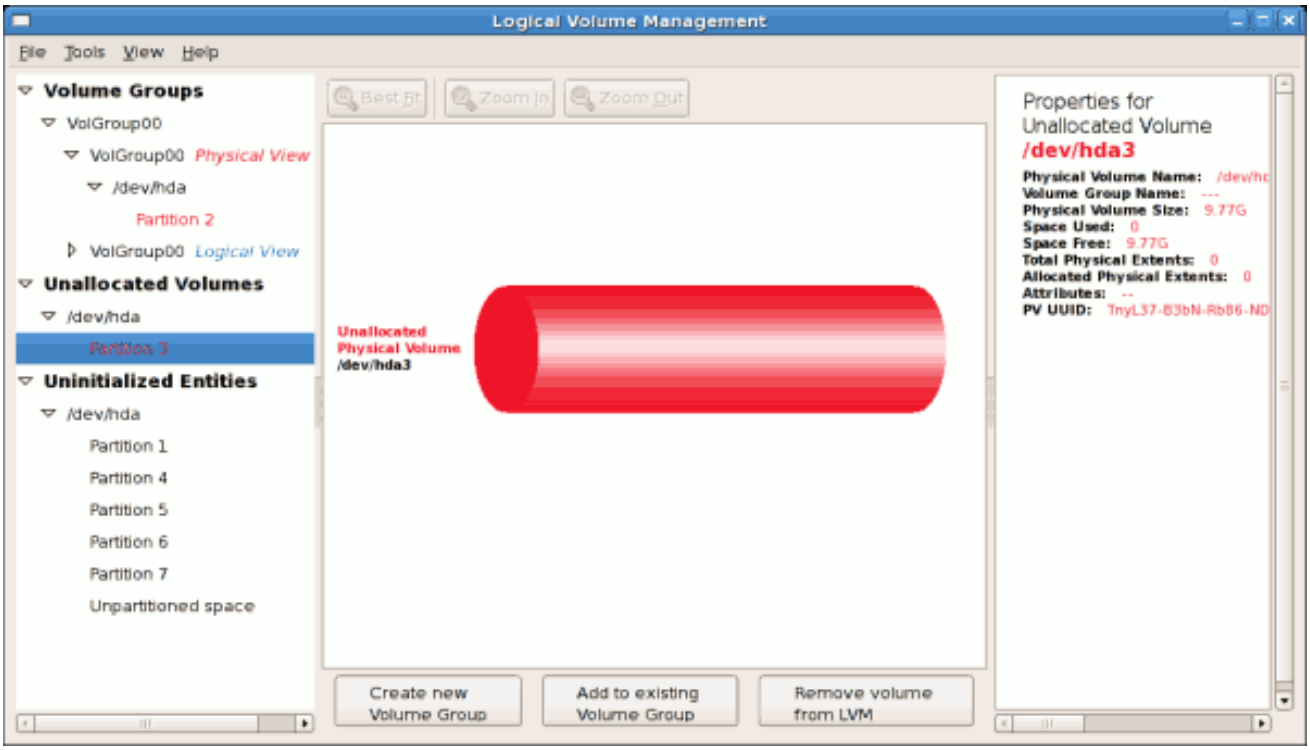
14.2.2. ボリュームグループへの未割り当てのボリュームの追加

初期化されると、ボリュームは「未割り当てのボリューム」一覧に表示されます。以下の図は、未割り当てのパーティションを示しています(Partition 3)。ウィンドウの下部にある各ボタンでは、以下を行うことができます。

- *新規ボリュームグループの作成*
- *既存のボリュームグループに未割り当てのボリュームを追加します。*
- *LVM からボリュームを削除します。*

既存のボリュームグループにボリュームを追加するには、既存のボリュームグループ ***への追加 ボタンをクリックします。***

図14.7 未割り当てのボリューム

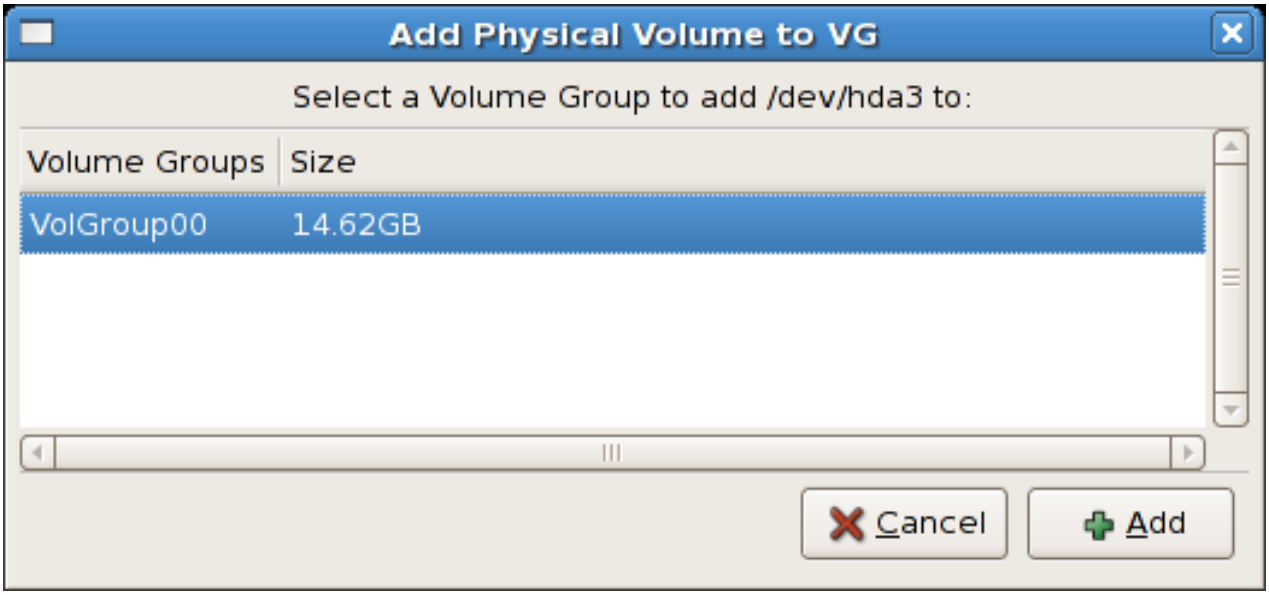


[D]

Add to Existing Volume Group *ボタンをクリックすると、初期化する物理ボリュームを追加する既存のボリュームグループが表示されているポップアップウィンドウが表示されます。ボリュームグループは、1 つ以上のハードディスクにまたがる可能性があります。*

例14.3 ボリュームグループへの物理ボリュームの追加

この例では、以下のように1 つのボリュームグループのみが存在します。



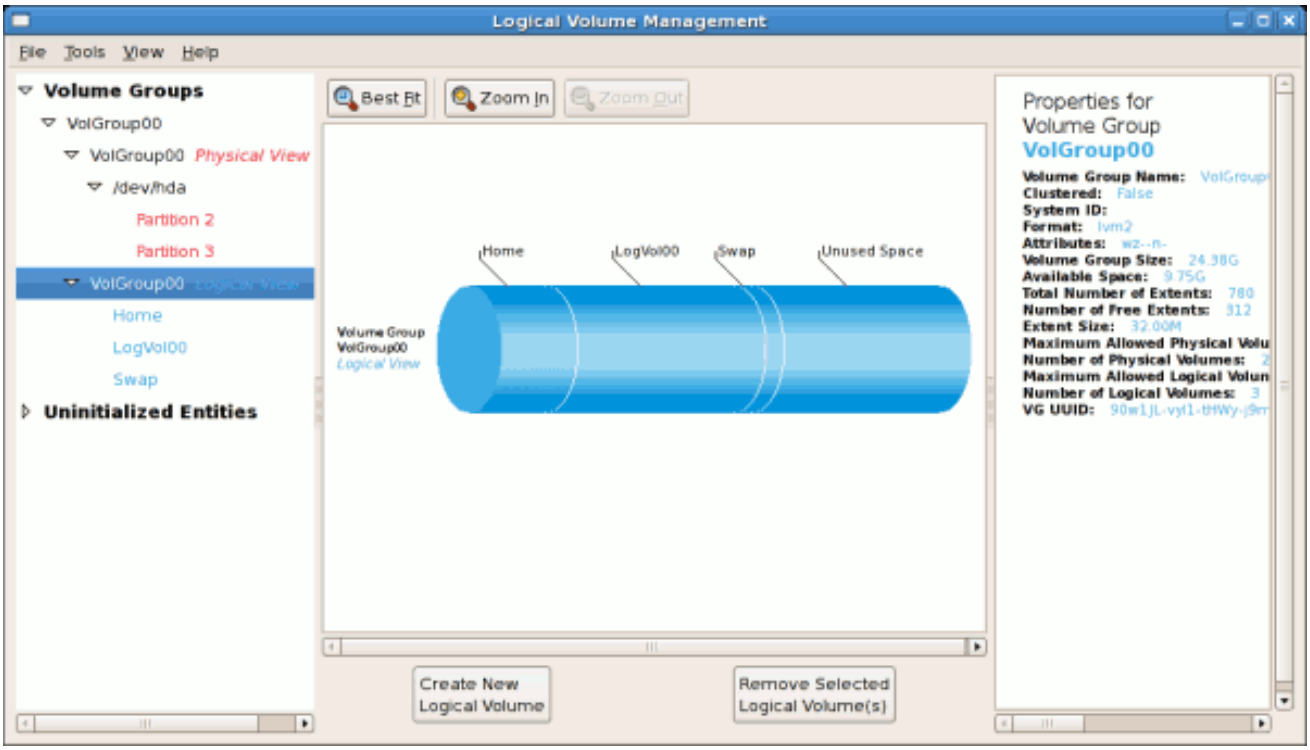
[D]

既存のボリュームグループに追加すると、新しい論理ボリュームが、選択したボリュームグループの未使用の領域に自動的に追加されます。未使用の領域を使用して以下を行うことができます。

- *新しい論理ボリュームを作成する（**新規論理ボリュームの作成 ボタンをクリックします**）*
- *既存の論理ボリュームのいずれかを選択し、エクステントを増やします（「ボリュームグループの拡張」を参照）。*
- ***選択した論理ボリュームの削除 ボタンをクリックして、既存の論理ボリュームを選択し、ボリュームグループからこれを削除します。**この操作を実行するために未使用の領域を選択することはできません。*

以下の図は、新規ボリュームグループの追加後の「VolGroup00」の論理ビューを示しています。

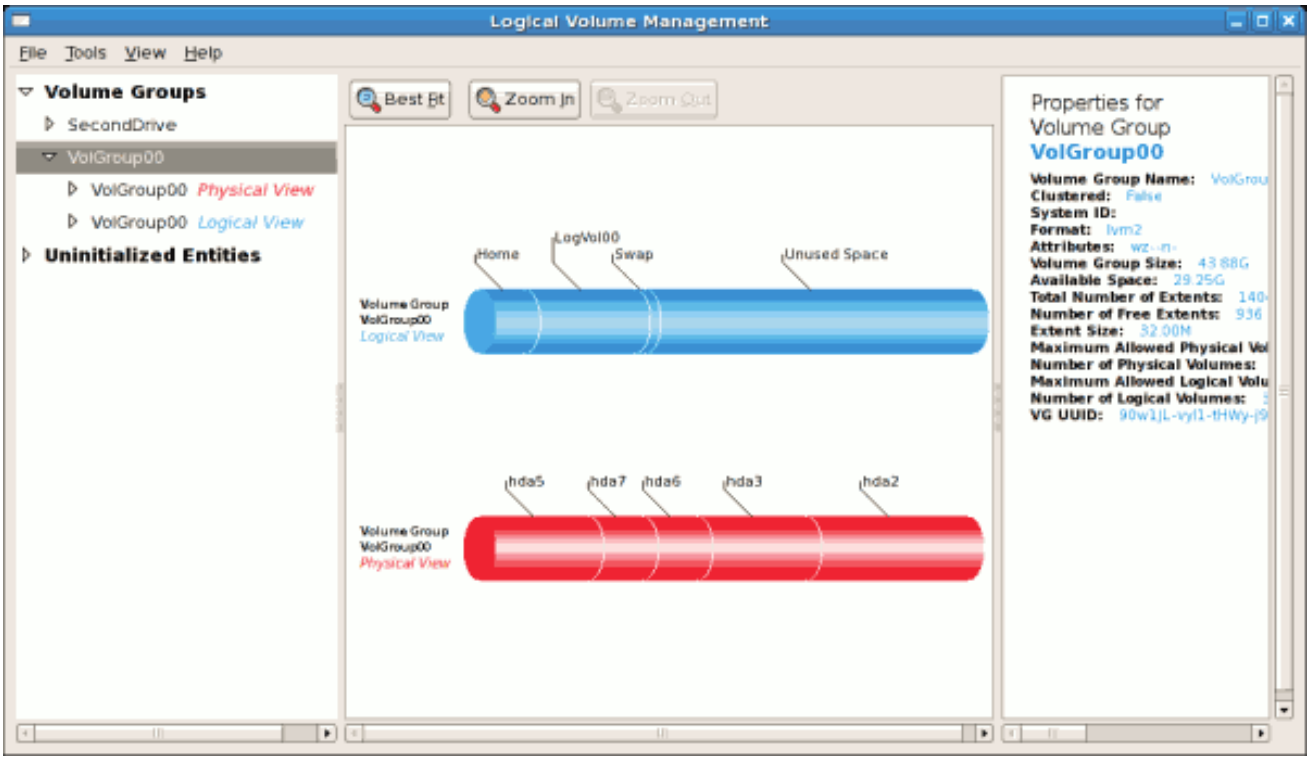
図14.8 ボリュームグループの論理ビュー



[D]

以下の図は、初期化されていないエンティティー（パーティション3、5、6、および7）が 'VolGroup00' に追加されました。

図14.9 ボリュームグループの論理ビュー

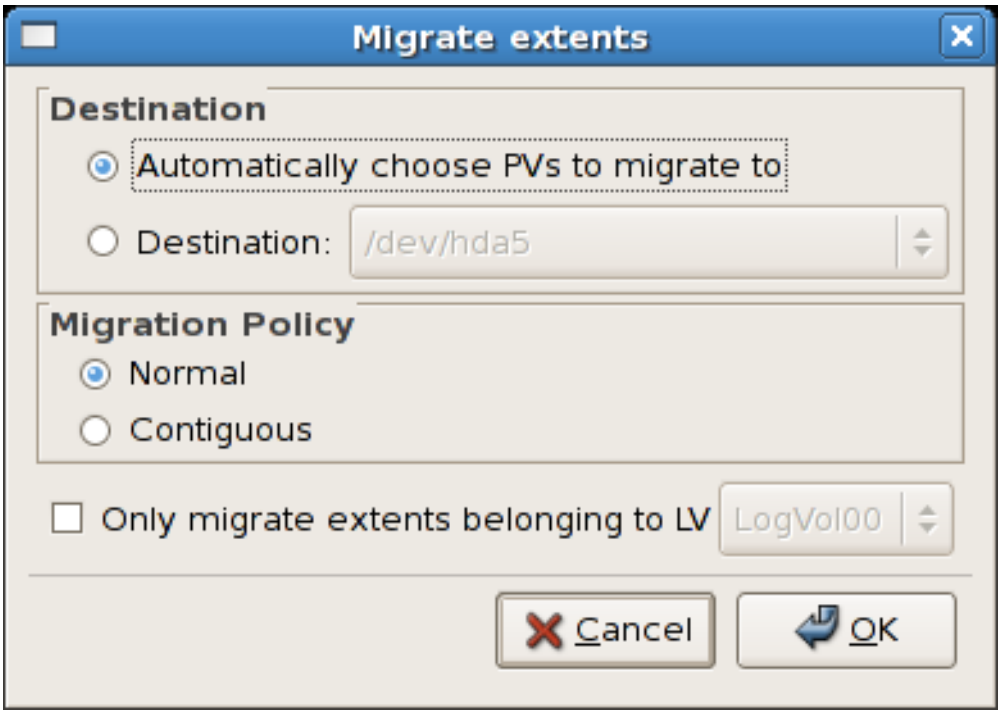


[D]

14.2.3. エクステントの移行

*物理ボリュームからエクステントを移行するには、左側のペインのリストからボリュームを選択し、中央ウィンドウで必要なエクステントを強調表示し、**選択した Extent(s)From Volume** ボタンをクリックします。ボリュームグループ内のエクステントを移行するには、十分な数の空きエクステントが必要です。空きエクステントが十分にない場合には、エラーメッセージが表示されます。この問題を解決するには、ボリュームグループを拡張します（「ボリュームグループの拡張」を参照）。ボリュームグループで十分な空きエクステントが検出された場合は、エクステントの宛先を選択するか、LVM が物理ボリューム(PV)を選択できる物理ボリューム(PV)を選択できるポップアップウィンドウが表示されます。以下で説明します。*

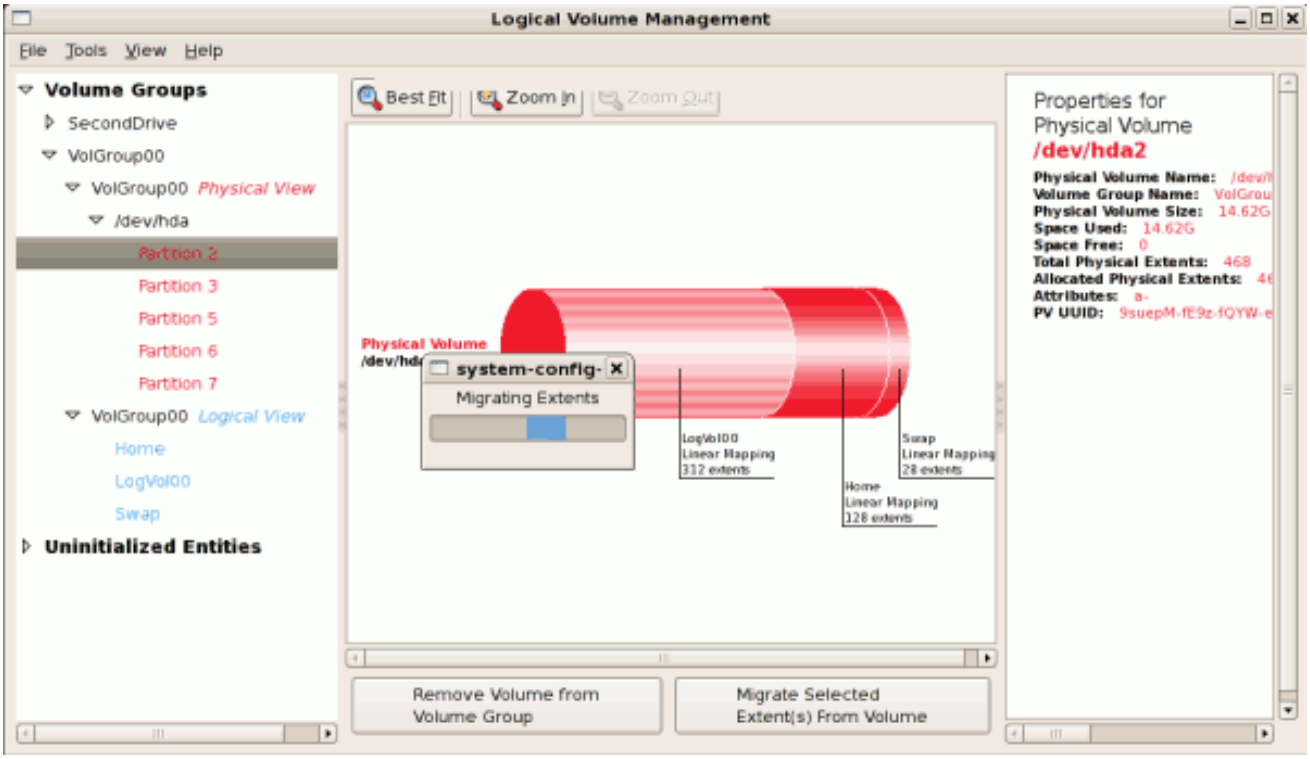
図14.10 エクステントの移行



[D]

以下の図は、進行中のエクステントの移行を示しています。この例では、エクステントは「Partition 3」に移行されました。

図14.11 エクステントの移行中

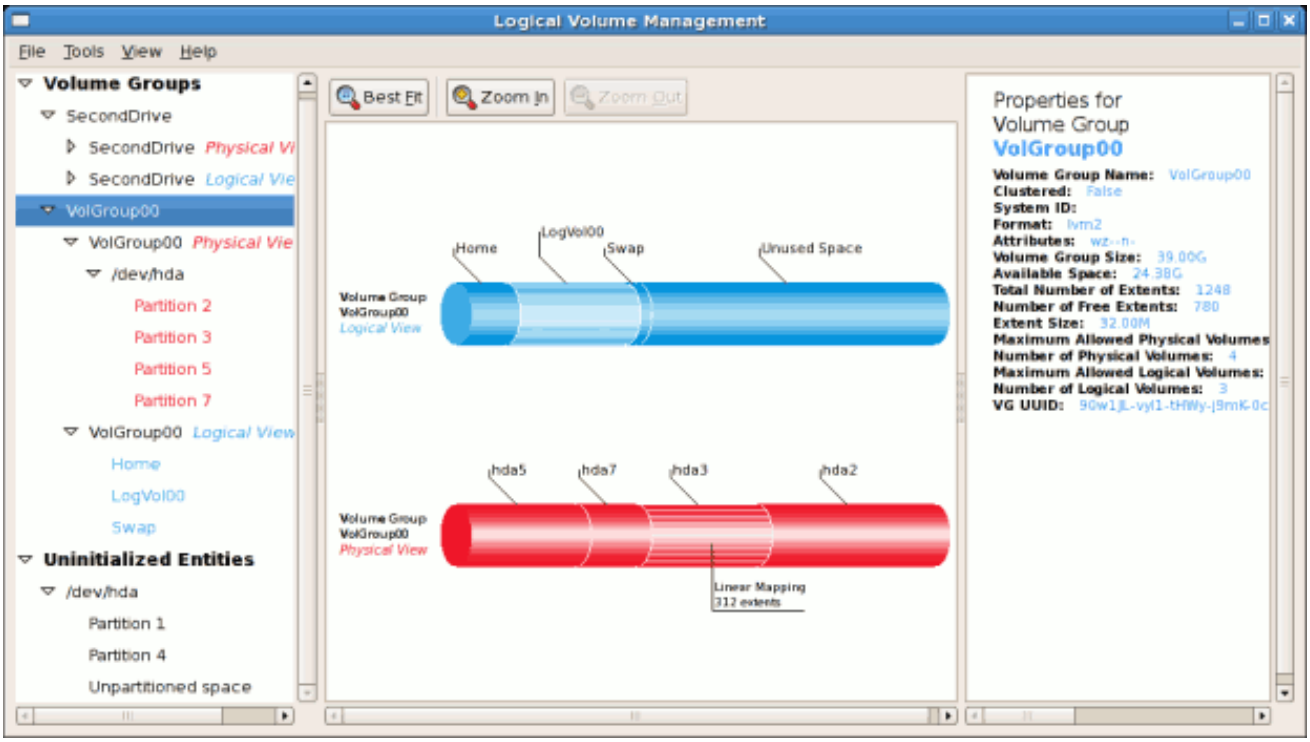


[D]

エクステントを移行すると、その物理ボリュームに未使用の領域が残されます。以下の図は、ボリュームグループの物理および論理ビューを示しています。hda2 の最初の LogVol00 のエクステントは hda3 にあります。エクステントを移行すると、ハードディスクのアップグレードやディスク領域の管理によ

り、論理ボリュームを移動できます。

図14.12 ボリュームグループの論理および物理ビュー

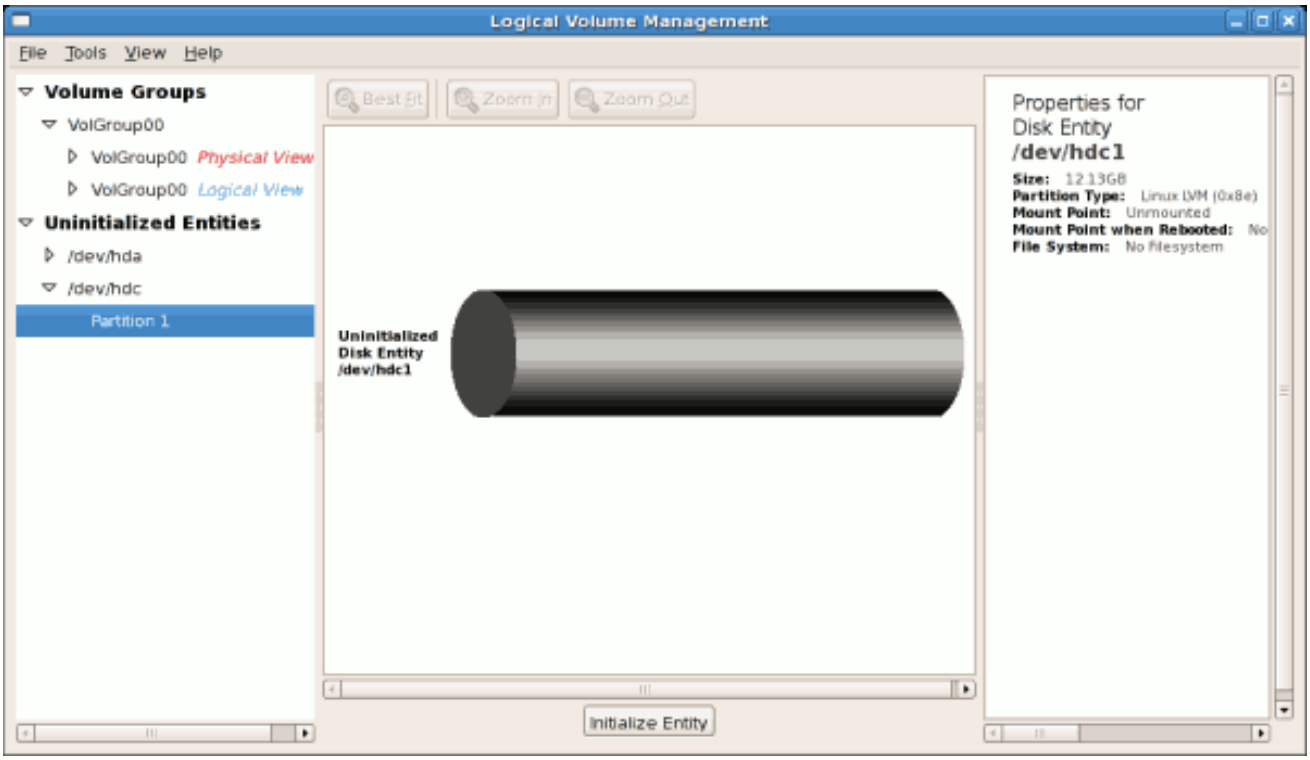


[D]

14.2.4. LVM を使用した新規ハードディスクの追加

この例では、新しい IDE ハードディスクが追加されました。以下の図は、新規ハードディスクの詳細を示しています。以下の図では、ディスクは初期化されず、マウントされません。パーティションを初期化するには、**Entity** ボタンをクリックします。詳細は、「初期化されていないエンタイトルメントの使用」 を参照してください。初期化されると、LVM は、例14.4「新規ボリュームグループの作成」 にあるように、未割り当てのボリューム一覧に新規ボリュームを追加します。

図14.13 未初期化のハードディスク

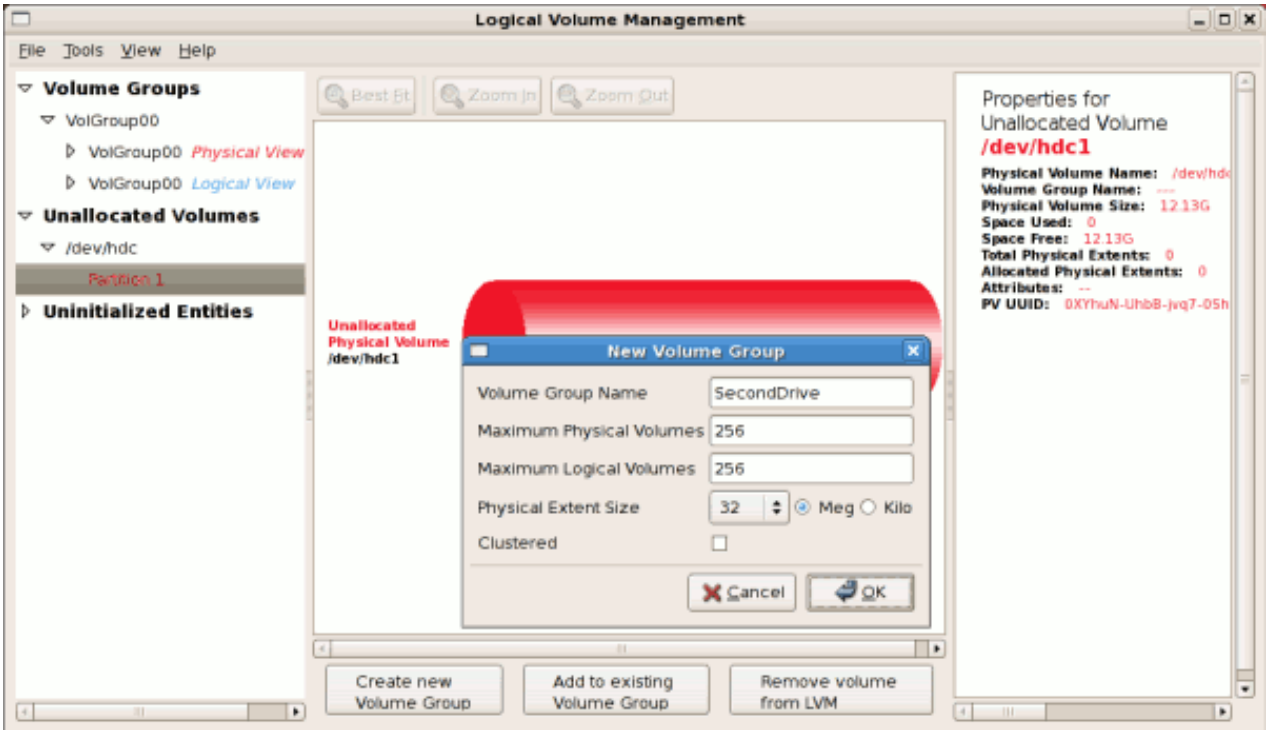


[D]

14.2.5. 新規ボリュームグループの追加

初期化されると、LVM は、新規ボリュームを、既存のボリュームグループに追加したり、新規ボリュームグループを作成したりできる、未割り当てのボリュームの一覧に追加します。LVM からボリュームを削除することもできます。ボリュームがLVM から削除される場合、図14.13「未初期化のハードディスク」 に記載されているように、これは「初期化されていないエンタイトルメント」一覧に追加されます。

> ***例14.4 新規ボリュームグループの作成***
>
> この例では、以下のように新しいボリュームグループが作成されています。

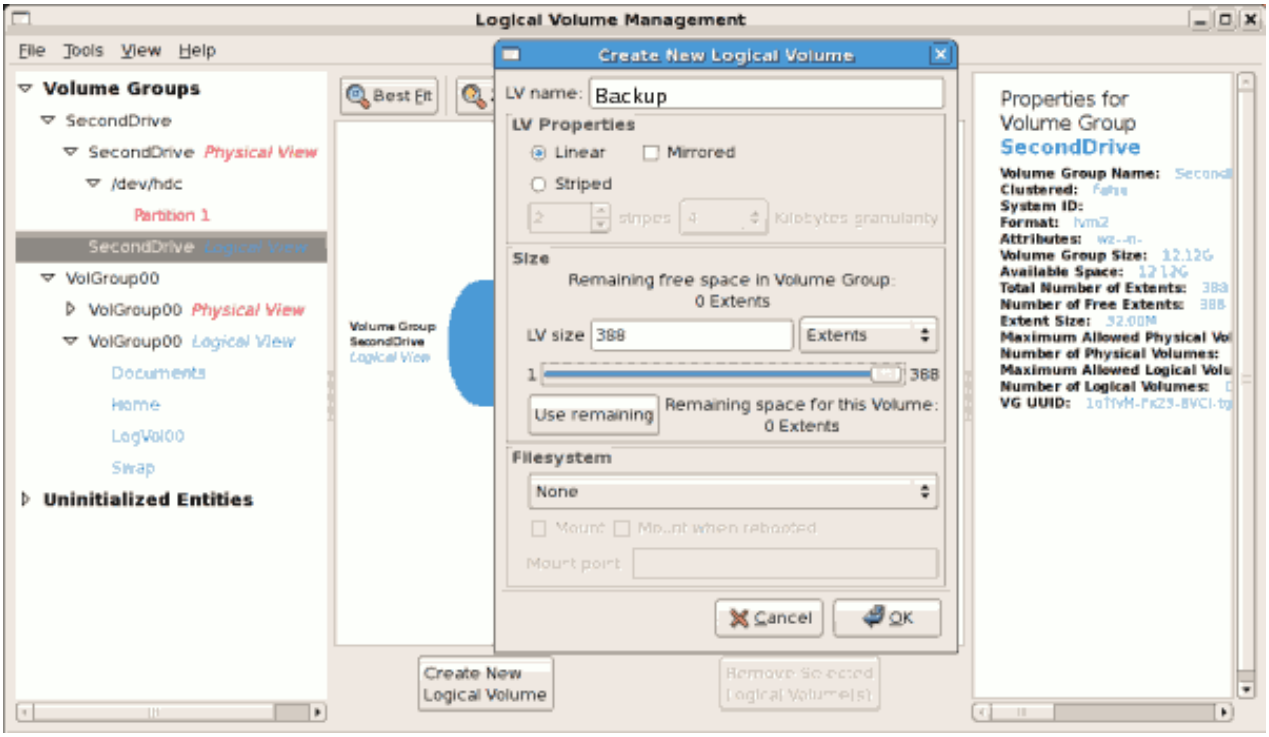


[D]

*新規ボリュームグループを作成すると、以下のように既存のボリュームグループが表示されます。論理ボリュームが作成されていないと、未使用の領域を持つ新しいボリュームグループが表示されます。論理ボリュームを作成するには、ボリュームグループを選択し、**以下のように新規論理ボリュームの作成**ボタンをクリックします。ボリュームグループで使用するエクステントを選択します。*

例4.5 エクステントの選択

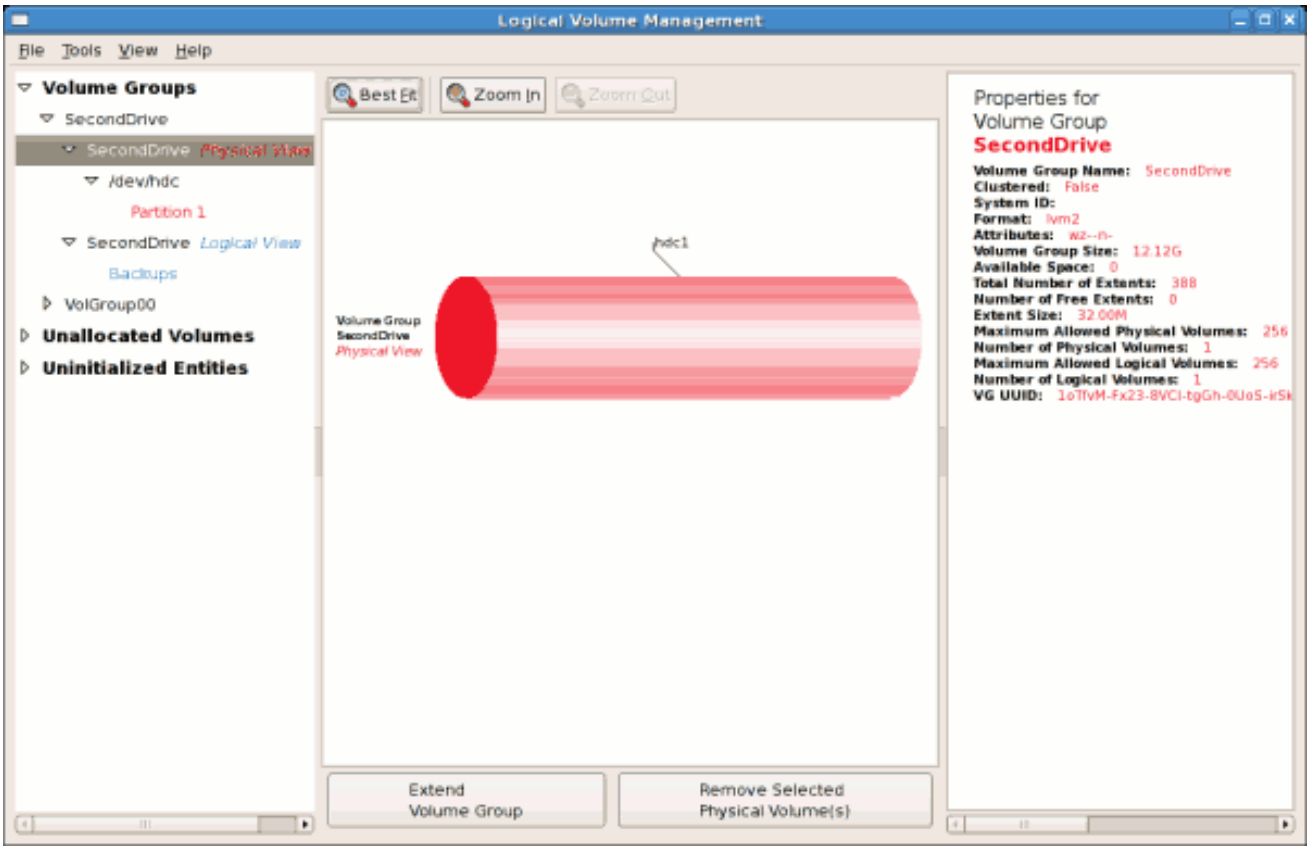
この例では、ボリュームグループのすべてのエクステントを使用して、新しい論理ボリュームを作成しました。



[D]

以下の図は、新規ボリュームグループの物理ビューを示しています。このボリュームグループに「Backups」という名前の新しい論理ボリュームも一覧表示されます。

図14.14 新規ボリュームグループの物理ビュー

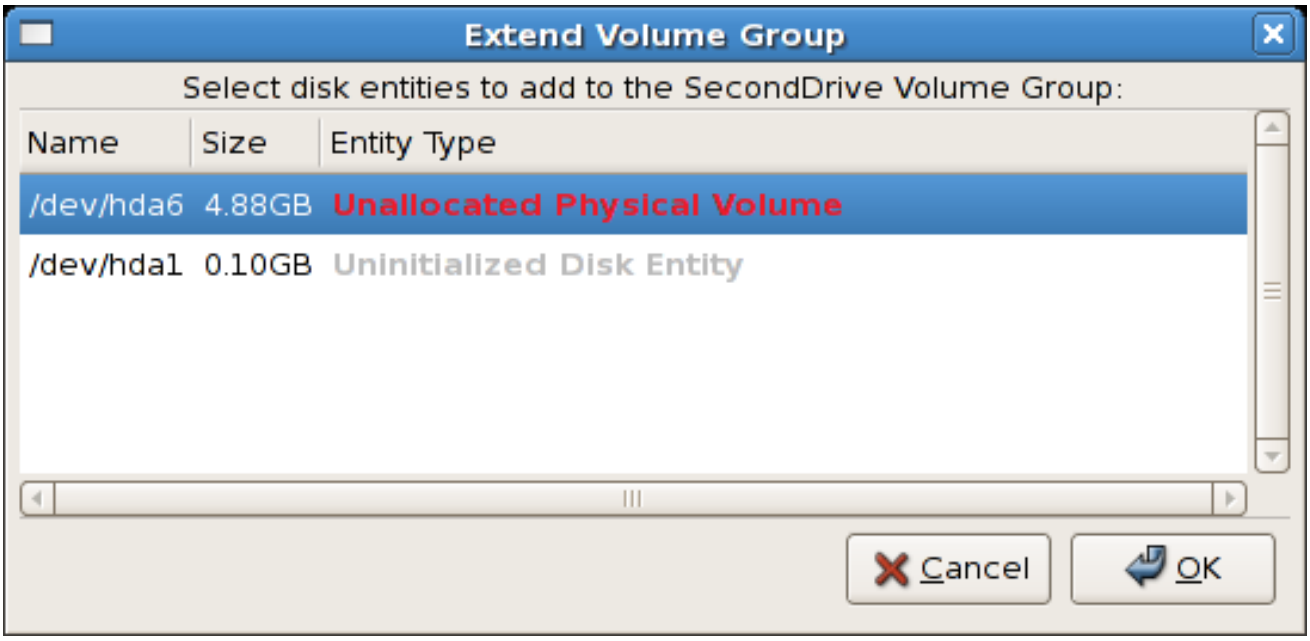


[D]

14.2.6. ボリュームグループの拡張

この例では、初期化されていないエンティティー（パーティション）を組み込むように新規ボリュームグループを拡張します。これにより、ボリュームグループのエクステントのサイズまたは数が増えます。ボリュームグループを拡張するには、左側のペインで、任意のボリュームグループ内で Physical View オプションが選択されていることを確認します。次に、ボリュームグループ ***の拡張 ボタンをクリックします。*** *これにより、以下のように「ボリュームグループの拡張」ウィンドウが表示されます。「ボリュームグループの拡張」ウィンドウで、ディスクエンティティー（パーティション）を選択してボリュームグループに追加できます。重要なデータが削除されないように、「未初期化ディスクエンタイトルメント」の内容（パーティション）の内容を確認してください（図14.13「未初期化のハードディスク」を参照してください）。この例では、以下のようにディスクエンティティー（パーティション）* ***/dev/hda6*** *が選択されています。*

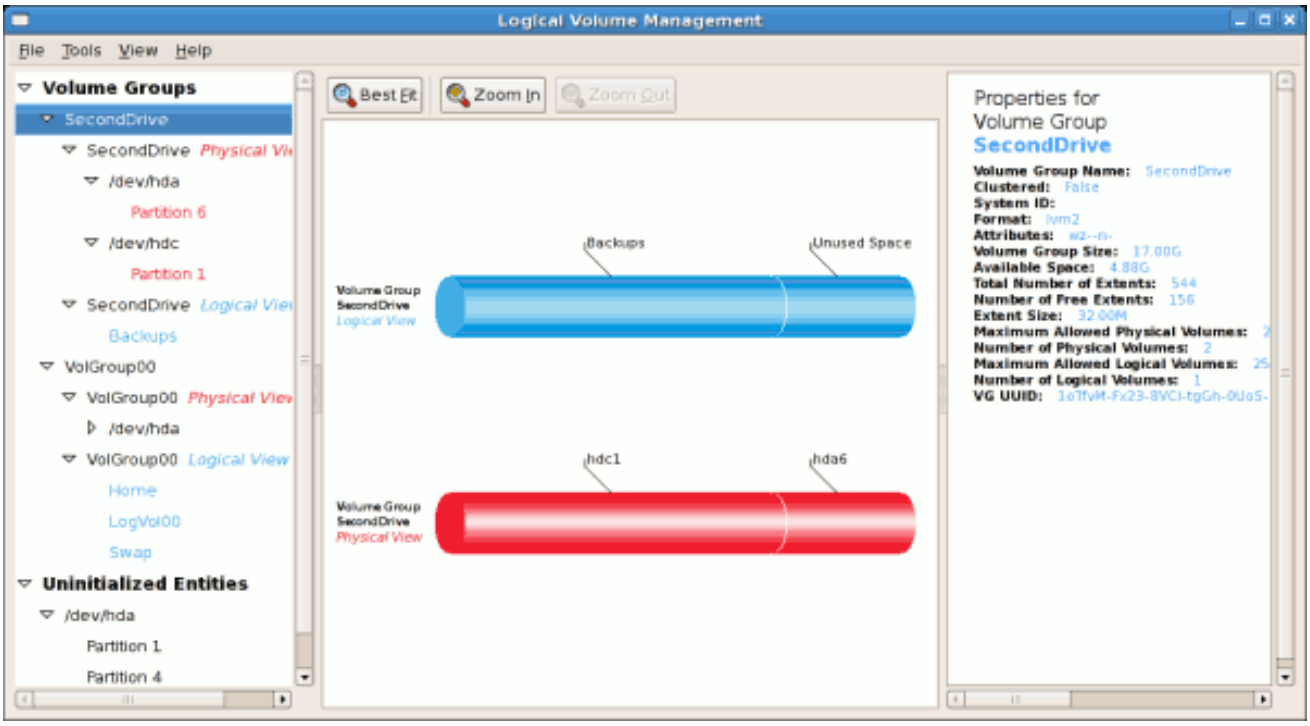
図14.15 ディスクエンティティーの選択



[D]

このボリュームを追加すると、ボリュームグループに「未使用スペース」として新規ボリュームが追加されます。以下の図は、拡張後のボリュームグループの論理および物理ビューを示しています。

図14.16 拡張ボリュームグループの論理および物理ビュー



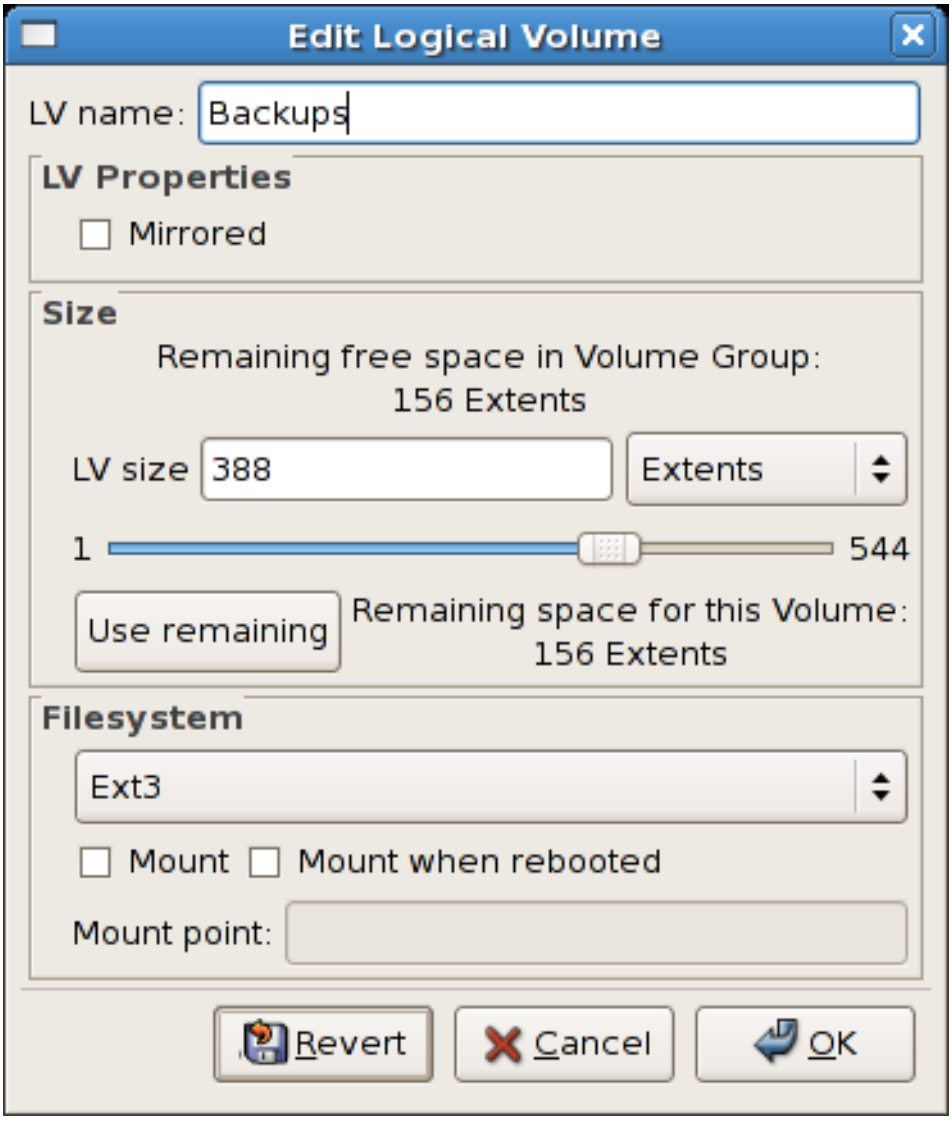
[D]

14.2.7. 論理ボリュームの編集

LVM ユーティリティーを使用すると、ボリュームグループで論理ボリュームを選択し、名前、サイズを変更し、ファイルシステムオプションを指定できます。この例では、「Backups」という名前の論理ボリュームが、ボリュームグループの残りのスペースに拡張されました。

プロパティーの編集 ボタンをクリックすると、論理ボリュームのプロパティーを編集できる「論理ボリュームの編集」のポップアップウィンドウが表示されます。このウィンドウでは、変更後にボリュームをマウントし、システムの再起動時にマウントすることもできます。マウントポイントを指定する必要があります。マウントポイントに指定したマウントポイントが存在しない場合は、作成するようにポップアップウィンドウが表示されます。「論理ボリュームの編集」ウィンドウを以下に示します。

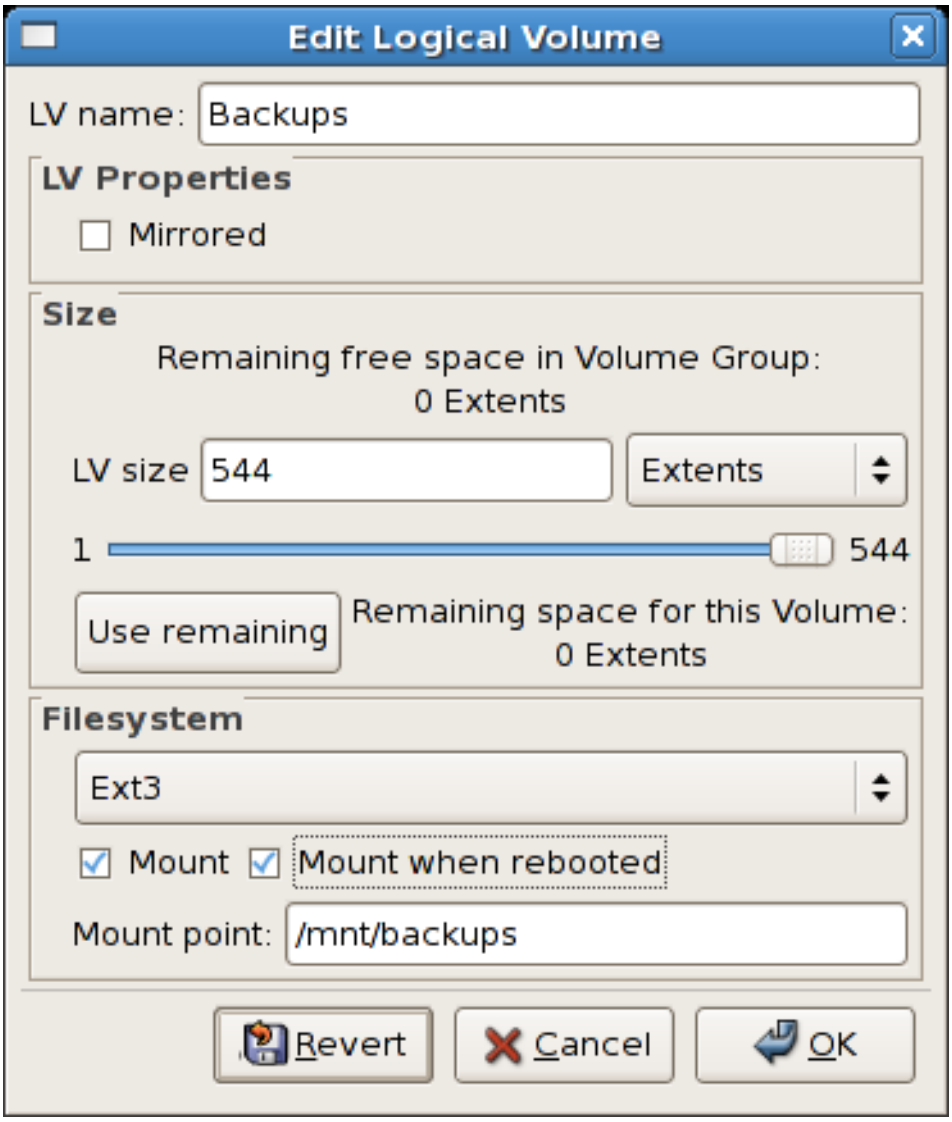
図14.17 論理ボリュームの編集



[D]

ボリュームをマウントする場合は、推奨されるマウントポイントを示す「Mount」チェックボックスを選択します。システムの再起動時にボリュームをマウントするには、「Mount when reboot」チェックボックスを選択します。この例では、新規ボリュームは **/mnt/backups** にマウントされます。これは、以下の図を示しています。

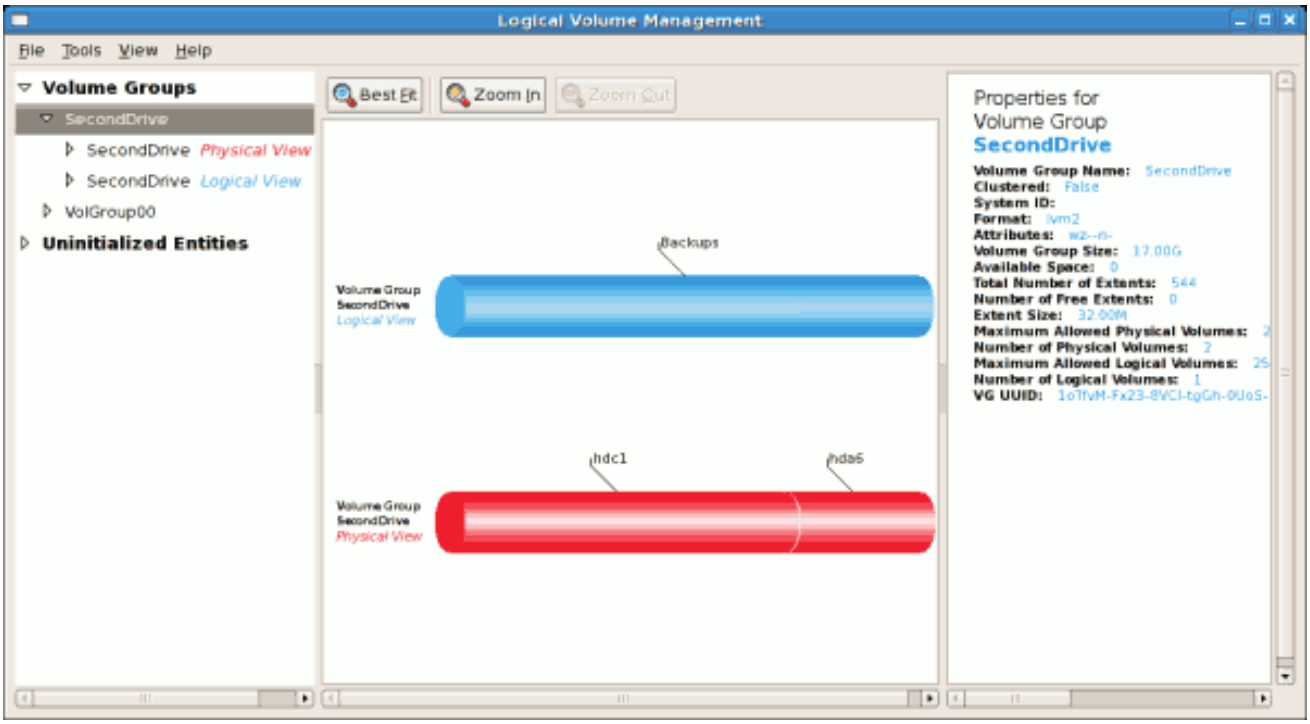
図14.18 論理ボリュームの編集 - マウントオプションの指定



[D]

以下の図は、論理ボリュームが未使用の領域に拡張された後にボリュームグループの論理および物理ビューを示しています。この例では、'Backups' という名前の論理ボリュームが2 つのハードディスクにまたがる。LVM を使用して、2 つ以上の物理デバイス全体でボリュームを取り除くことができます。

図14.19 論理ボリュームの編集



[D]

14.3. LVM リファレンス

これらのソースを使用して、LVM について確認してください。

インストールされているドキュメント

- **rpm -qd lvm2** - このコマンドは、man ページなど、**lvm パッケージで利用可能な** すべてのドキュメントを表示します。
- **LVM help** - このコマンドは、利用可能なLVM コマンドをすべて表示します。

便利な Web サイト

- http://sources.redhat.com/lvm2: LVM2 Web ページ（概要、メーリングリストへのリンクなど）
- http://tldp.org/HOWTO/LVM-HOWTO/: Linux ドキュメントプロジェクトのLVM HOWTO。

第15章 SWAP 領域

Linux のスワップ領域は、物理メモリー (RAM) が不足すると使用されます。システムに多くのメモリーリソースが必要で、RAM が不足すると、メモリーの非アクティブなページがスワップ領域に移動します。スワップ領域は、RAM が少ないマシンで役に立ちますが、RAM の代わりに使用しないようにしてください。スワップ領域はハードドライブにあり、そのアクセス速度は物理メモリーに比べると遅くなります。スワップ領域の構成は、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せが考えられます。

過去数年、推奨されるスワップ領域のサイズは、システムの RAM サイズに比例して増加していました。しかし、最近のシステムには通常、数百ギガバイトの RAM が含まれます。結果として、推奨されるスワップ領域は、システムのメモリーではなく、システムメモリーのワークロードの機能とみなされます。

表15.1「システムの推奨 swap 領域」 では、システムの RAM の容量別に推奨されるスワップパーティションのサイズと、システムをハイバネートするのに十分なメモリーが必要かどうかを提供します。推奨されるスワップパーティションのサイズは、インストール時に自動的に確定されます。ハイバネートを可能にするには、カスタムのパーティション分割段階でスワップ領域を編集する必要があります。

重要

表15.1「システムの推奨 swap 領域」 の推奨では、メモリーが少ないシステム (1 GB 以下) で特に重要になります。このようなシステムで十分なスワップ領域を割り当てられないと、不安定になる問題が生じたり、インストールしたシステムが起動できなくなる可能性があります。

表15.1 システムの推奨 swap 領域

システム内の RAM の容量	推奨されるスワップ領域	ハイバネートを許可する場合に推奨されるスワップ領域
⩽ 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されない

ご使用のシステムが 表15.1「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適宜判断してください。システムリソースに余裕がある場合は、スワップ領域を増やすとパフォーマンスが向上することがあります。

高速のドライブ、コントローラー、およびインターフェー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スワップ領域を分散すると、スワップ領域のパフォーマンスも向上します。

重要

スワップ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、変更時に **使用しない** *でください。システムプロセスまたはカーネルがスワップ領域を使用していると、スワップの修正に失敗します。* ***free*** *コマンドおよび* ***cat /proc/swaps*** *コマンドを使用して、スワップの使用量と、使用中の場所を確認します。*

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。『Red Hat Enterprise Linux 6 インストールガイド』 の レスキューモードでのコンピューターの起動 を参照してください。ファイルシステムをマウントするように指示されたら、* ***スキップ*** *を選択します。*

15.1. スワップ領域の追加

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。たとえば、システムの RAM 容量を 1 GB から 2 GB にアップグレードするときに、スワップ容量が 2 GB しかないとします。メモリーを大幅に消費する操作を実行している場合や、大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は、スワップ領域を 4 GB に増やすことが有益となる可能性があります。

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。この中では、既存論理ボリュームを拡張することが推奨されます。

15.1.1. LVM2 論理ボリュームでのスワップ領域の拡張

Red Hat Enterprise Linux 6 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は、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。その手順は、 「ボリュームグループへの未割り当てのボリュームの追加」 を参照してください。

swap 領域のボリュームグループにストレージを追加した後に、それを拡張することができるようになります。これを実行するには、次の手順に従います (***/dev/VolGroup00/LogVol01*** *ボリュームのサイズを 2 GB 拡張するとします)。*

手順15.1 LVM2 論理ボリュームでのスワップ領域の拡張

1. *関連付けられている論理ボリュームのスワップ機能を無効にします。*

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. *LVM2 論理ボリュームのサイズを 2 GB 増やします。*

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3. *新しいスワップ領域をフォーマットします。*

```
# mkswap /dev/VolGroup00/LogVol01
```

4. *拡張論理ボリュームを有効にします。*

```
# swapon -v /dev/VolGroup00/LogVol01
```

論理ボリュームが拡張されたかどうかをテストするには、**cat /proc/swaps** または **free** を使用してスワップ領域を調べます。

15.1.2. スワップの LVM2 論理ボリュームの作成

スワップボリュームグループを追加するには（スワップボリューム **/dev/VolGroup00/LogVol02**） を追加します。

1. サイズが 2 GB の LVM2 論理ボリュームを作成します。

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. 新しいスワップ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をテストするには、**cat /proc/swaps** または **free** を使用してスワップ領域を調べます。

15.1.3. スワップファイルの作成

swap ファイルを追加します。

手順15.2 スワップファイルの追加

1. 新しいスワップファイルのサイズをメガバイト単位で指定してから、そのサイズに 1024 をかけてブロック数を指定します。たとえばスワップファイルのサイズが 64 MB の場合は、ブロック数が 65536 になります。

2. **count** を必要なブロックサイズと等しくなるには、以下のコマンドを入力します。

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

3. 次のコマンドでスワップファイルをセットアップします。

```
# mkswap /swapfile
```

4. スワップが誰でも読み取り可能にならないようにパーミッションを変更することが推奨されます。

```
# chmod 0600 /swapfile
```

5. システムの起動時に自動的に swap ファイルは有効にしない場合は、次のコマンドを実行します。

```
# swapon /swapfile
```

6. システムの起動時に有効にするには、**/etc/fstab** を編集して以下のエントリーを追加します。

   /swapfile swap swap defaults 0 0

   次にシステムを起動すると、新しいスワップファイルが有効になります。

新しいスワップ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** を使用してスワップ領域を調べます。

15.2. スワップ領域の削除

インストールの後に swap 領域を減らすことが賢明な場合もあります。たとえば、システムの RAM 容量を 1 GB から 512 MB にダウングレードするとします。しかし、依然として 2 GB のスワップ容量が割り当てられています。ディスク領域が大きくなる (2 GB など) と無駄になる可能性があるため、スワップ領域を 1 GB に減らすことでメリットを得られることがあります。

ここでも選択肢が 3 つあります: swap 用に使用していた LVM2 論理ボリューム全体を削除、swap ファイルの削除、あるいは既存の LVM2 論理ボリューム上の swap 領域の低減。

15.2.1. LVM2 論理ボリュームでのスワップ領域の縮小

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ます)。

手順15.3 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スワップ機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. LVM2 論理ボリュームのサイズを変更して 512 MB 削減します。

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. 新しいスワップ領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 拡張論理ボリュームを有効にします。

```
# swapon -v /dev/VolGroup00/LogVol01
```

スワップの論理ボリュームのサイズを縮小するかどうかをテストするには、**cat /proc/swaps** または **free** を使用してスワップ領域を調べます。

15.2.2. スワップの 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ます)。*

手順15.4 swap ボリュームグループの削除

1. *関連付けられている論理ボリュームのスワップ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. ***/etc/fstab** ファイルから以下のエントリーを削除します。*

   /dev/VolGroup00/LogVol02 swap swap defaults 0 0

*論理ボリュームのサイズが正常に削除されたかどうかをテストするには、**cat /proc/swaps** または **free** を使用してスワップ領域を調べます。*

15.2.3. スワップファイルの削除

swap ファイルを削除します。

手順15.5 swap ファイルの削除

1. *シェルプロンプトで次のコマンドを実行してスワップファイルを無効にします (スワップファイルの場所が **/swapfile** であるとします)。*

```
# swapoff -v /swapfile
```

2. ***/etc/fstab** ファイルから該当するエントリーを削除します。*

3. *実際のファイルを削除します。*

```
# rm /swapfile
```

15.3. SWAP 領域の移動

スワップ領域を別の場所に移動するには、スワップ領域を削除する手順に従い、スワップ領域を追加する手順に従ってください。

第16章 ディスククォータ

ディスク領域はディスククォータ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、ユーザーグループにも個別のユーザーにも設定できます。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。

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、以下の手順を行います。

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。

この各ステップは、以下のセクションで詳しく説明します。

16.1.1. クォータの有効化

root でテキストエディターを使用して、**/etc/fstab** ファイルを編集します。

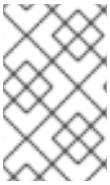
> **例16.1 /etc/fstab の編集**
>
> たとえば、テキストエディター **vim** を使用するには、以下を入力します。
>
> ```
> # vim /etc/fstab
> ```

クォータを必要とするファイルシステムに **usrquota および/または grpquota** オプションを追加します。

> **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。*



備考

*以下の例では、Red Hat Enterprise Linux のインストール時に別の **/home** パーティションが作成されたことを仮定します。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** および **grpquota オプションを追加し** たら、fstab **エントリー** が変更された各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使用されていない場合は、以下のコマンドを使用します。*

umount /mount-point

*たとえば、**umount /work です**。*

mount /file-system /mount-point

*たとえば、**/dev/vdb1 /work をマウントします**。*

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、クォータが有効なファイルシステムを検証し、現在のディスク使用状況のテーブルをファイルシステムごとに構築します。このテーブルは、ディスク使用状況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が更新されます。*

*ファイルシステムにクォータファイル（a**quota.user** および **aquota.group**）を作成するには、quota **check** コマンドの **-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、ユーザーおよびグループのクォータが **/home** ファイルシステムに対して有効になっている場合は、**/home** ディレクトリーにファイルを作成します。*

```
# quotacheck -cug /home
```

-c オプションは、クォータが有効になっている各ファイルシステムにクォータファイルを作成する必要があることを指定し、- u オプションはユーザークォータの確認を指定し、- g オプションはグループクォータをチェックするように指定します。

-u オプションまたは -g オプションがいずれも指定されていないと、ユーザーのクォータファイルのみが作成されます。-g のみを指定すると、グループクォータファイルのみが作成されます。

ファイルが作成されたら、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量の表を生成します。

```
# quotacheck -avug
```

使用されるオプションは以下のとおりです。

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）に対応するクォータファイルに、/ home などのローカルにマウントされた各ファイルシステムのデータが設定されます。

16.1.4. ユーザーごとのクォータ割り当て

最後のステップは、edquota コマンドを使用したディスククォータ割り当てです。

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 で次のコマンドを実行します。

```
# edquota username
```

クォータが必要な各ユーザーに対して、この手順を実行します。たとえば、クォータが /home パーティションの /etc/fstab (以下の例では /dev/VolGroup00/LogVol02) に対して有効であり、コマンド edquota testuser を実行すると、システムでデフォルトとして設定されたエディターで以下のような出力が表示されます。

```
Disk quotas for user testuser (uid 501):
Filesystem                blocks     soft     hard    inodes   soft   hard
/dev/VolGroup00/LogVol02  440436        0        0     37418      0      0
```

備考

EDITOR 環境変数により定義されたテキストエディターは、edquota により使用されます。エディターを変更するには、~/.bash_profile ファイルの EDITOR 環境変数を、使用するエディターのフルパスに設定します。

*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 列目には、ユーザーが現在使用しているブロック数が示されます。その次の 2 列は、ファイルシステム上のユーザーのソフトブロック制限およびハードブロック制限を設定するのに使用されます。**inodes** 列には、ユーザーが現在使用している inode の数が表示されます。最後の 2 列は、ファイルシステムのユーザーに対するソフトおよびハードの inode 制限を設定するのに使用されます。*

ハードブロック制限は、ユーザーまたはグループが使用できる最大ディスク容量 (絶対値) です。この制限に達すると、それ以上のディスク領域は使用できなくなります。

ソフトブロック制限は、使用可能な最大ディスク容量を定義します。ただし、ハード制限とは異なり、ソフト制限は一定時間超過する可能性があります。この時間は 猶予期間 として知られています。猶予期間の単位は、秒、分、時間、日、週、または月で表されます。

いずれかの値が 0 に設定されていると、その制限は設定されません。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***
>
> *以下に例を示します。*
>
> ```
> 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、グループごとに割り当てることもできます。**devel** グループのグループクォータを設定するには (グループはグループクォータを設定する前に存在している必要があります)、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
```

この制限を変更して、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. ソフト制限の猶予期間の設定

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 オプションまたは **-g** オプションがいずれも指定されていないと、ユーザーのクォータのみが無効になります。**-g** のみを指定すると、グループのクォータのみが無効になります。**-v** スイッチにより、コマンドの実行時に詳細なステータス情報が表示されます。

クォータを再度有効にするには、同じオプションを指定して **quotaon** コマンドを使用します。

たとえば、全ファイルシステムのユーザーおよび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 オプションまたは **-g** オプションがいずれも指定されていないと、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。

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** ユーティリティー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** により、以下のような出力が表示されます。

```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol02
Block grace time: 7days; Inode grace time: 7days
   Block limits   File limits
User  used soft hard grace used soft hard grace
----------------------------------------------------------------------
root      --      36       0       0              4     0     0
kristin   --     540       0       0            125     0     0
testuser  --  440400  500000  550000          37418     0     0
```

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

```
# repquota -a
```

レポートは読みやすいですが、いくつか説明しておくべき点があります。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。

grace 列は通常空白です。ソフト制限が超過した場合、その列には猶予期間に残り時間量に相当する時間指定が含まれます。猶予期間の期間が過ぎると、**none** がその場所に表示されます。

16.2.3. クォータの精度維持

ファイルシステムが正常にアンマウントされない場合（システムクラッシュなど）、**quotacheck** を実行する必要があります。ただし、システムがクラッシュしていないとしても、**quotacheck** は定期的に実行できます。**quotacheck** を定期的に実行する安全な方法には、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、定期的に再起動する (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、シェルスクリプトを **/etc/cron.daily/** または **/etc/cron.weekly/** ディレクトリーに置くか、**touch /forcequotacheck** コマンドが含まれる **crontab -e** コマンドを使用して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 スクリプトがこれを検索します。このディレクトリーが検出されると、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 を安全に実行する別の方法として、クオータファイルのデータ破損の可能性を回避するためにシングルユーザーモードでシステムを起動して、以下のコマンドを実行する方法があります。

```
# quotaoff -vaug /file_system
```

```
# quotacheck -vaug /file_system
```

```
# quotaon -vaug /file_system
```

実行中のシステムで 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** というコマンドを実行します。このコマンドは、**quotacheck** が指定された file_system を読み取り専用に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 の設定に関する詳細は、**man cron** を参照してください。

16.3. ディスククオータのリファレンス

ディスククォータの詳細は、次のコマンドの **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)、パリティーによるディスクのストライピング (RAID レベル 5) などの技術を使用して冗長性を得ながら待ち時間を抑え、帯域幅を増幅させることでハードディスクがクラッシュした場合の復元力を最大限に引き出します。

データは、一貫して同じサイズの大きさのチャンク (通常は 256K または 512K、ただし他の値も可) に分割され、アレイ内の各ドライブに分散されます。各チャンクは、使用している RAID レベルに応じて、RAID アレイ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ユーザーは、RAID 技術を使用することでメリットが得られます。RAID をデプロイする主な理由を以下に示します。

- 速度を高める
- 1 台の仮想ディスクを使用してストレージ容量を増加する
- ディスク障害によるデータ損失を最小限に抑える

17.1. RAID のタイプ

考えられる RAID アプローチには、ファームウェア RAID、ハードウェア RAID、およびソフトウェア RAID の 3 つがあります。

ファームウェア RAID

ファームウェア RAID (ATARAID とも呼ばれる) とは、ソフトウェア RAID の種類で、ファームウェアベースのメニューを使用して RAID セットを設定できます。このタイプの RAID で使用されるファームウェアは BIOS にもフックされるため、その RAID セットから起動できます。異なるベンダーは、異なるオンディスクメタデータ形式を使用して、RAID セットのメンバーをマークします。Intel Matrix RAID は、ファームウェア RAID システムの一例を示しています。

ハードウェア RAID

ハードウェアベースのアレイは、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への 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。高価ディスクコントローラーカードやホットスワップシャーシなど、最優先的な解決策を提供します。[4] 必須ではありません。ソフトウェア RAID は、SCSI ディスクだけでなく安価な IDE ディスクでも機能します。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 パフォーマンスと負荷によって異なります。

Linux ソフトウェア RAID スタックの主な機能の一部は次のとおりです。

- マルチスレッド設計
- 再構築なしで Linux マシン間でのアレイの移植性
- アイドルシステムリソースを使用したバックグラウンドのアレイ再構築
- 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
- アレイ内のディスク上にある不良セクターの自動修正
- 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
- 重要なイベントで指定された電子メールアドレスに送信された電子メールアラートによるアレイのプロアクティブな監視
- 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
- チェックポイントを再同期して、再同期中にコンピューターを再起動すると、起動時に再同期が中断したところから再開し、最初からやり直さないようにします。
- インストール後のアレイのパラメーター変更が可能です。たとえば、新しいデバイスを追加しても、4 つのディスクの RAID5 アレイを 5 つのディスク RAID5 アレイに増大させることができます。この拡張操作はライブで行うため、新しいアレイで再インストールする必要はありません。

17.2. RAID レベルとリニアサポート

RAID は、レベル 0、1、4、5、6、10、リニアなどのさまざまな設定に対応します。これらの RAID タイプは以下のように定義されます。

レベル 0

RAID レベル 0 は、多くの場合「ストライプ化」と呼ばれていますが、パフォーマンス指向のストライプ化データマッピング技術です。これは、アレイに書き込まれるデータがストライプに分割され、アレイのメンバーディスク全体に書き込まれることを意味します。これにより低い固有コストで高い I/O パフォーマンスを実現できますが、冗長性は提供されません。

多くの RAID レベル 0 実装は、アレイ内の最小デバイスのサイズまで、メンバーデバイス全体にデータをストライプ化します。つまり、複数のデバイスのサイズが少し異なる場合、それぞれのデバイスは最小ドライブと同じサイズであるかのように処理されます。そのため、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 内の最小メンバーディスクの容量と同じか、アレイ内のディスク数またはパーティション数で乗算したソフトウェアRAID 内の最小メンバーパーティションの容量と同じになります。

レベル1

RAID レベル1 または「ミラーリング」は、他のRAID 形式よりも長く使用されています。レベル1 は、アレイ内の各メンバーディスクに同一データを書き込むことで冗長性を提供し、各ディスクに対して「ミラーリング」コピーをそのまま残します。ミラーリングは、データの可用性の単純化と高レベルにより、いまでも人気があります。レベル1 は2 つ以上のディスクと連携して、非常に優れたデータ信頼性を提供し、読み取り集中型のアプリケーションに対してパフォーマンスが向上しますが、比較的コストが高くなります。[5]

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の容量と同じか、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 の冗長性は、すべてのRAID タイプの中で最も高いレベルであり、アレイは1 つのディスクのみで動作できます。

レベル4

レベル4 でパリティーを使用[6] データを保護するため、1 つのディスクドライブで連結します。専用パリティーディスクはRAID アレイへのすべての書き込みトランザクションに固有のボトルネックを表すため、レベル4 は、ライトバックキャッシュなどの付随技術なしで、またはシステム管理者がこれを使用してソフトウェアRAID デバイスを意図的に設計する特定の状況で使用されることはほとんどありません。ボトルネック(配列にデータが入力されると、書き込みトランザクションがほとんどまたはまったくない配列など) を念頭に置いてください。RAID レベル4 にはほとんど使用されないため、Anaconda ではこのオプションとしては使用できません。ただし、実際には必要な場合は、ユーザーが手動で作成できます。

ハードウェアRAID レベル4 のストレージ容量は、最小メンバーパーティションの容量にパーティションの数を掛けて、-1 を引いたものになります。RAID レベル4 アレイのパフォーマンスは常に非対称になります。つまり、読み込みは書き込みを上回ります。これは、パリティーを生成するときに書き込みが余分なCPU 帯域幅とメインメモリーの帯域幅を消費し、データだけでなくパリティーも書き込むため、実際のデータをディスクに書き込むときに余分なバス帯域幅も消費するためです。読み取りが必要なのは、アレイが劣化状態にない限り、パリティーではなくデータでを読み取るだけです。その結果、通常の動作条件下で同じ量のデータ転送を行う場合は、読み取りによってドライブへのトラフィック、またはコンピュータのバスを経由するトラフィックが少なくなります。

レベル5

これはRAID の最も一般的なタイプです。RAID レベル5 は、アレイのすべてのメンバーディスクドライブにパリティーを分散することにより、レベル4 に固有の書き込みボトルネックを排除します。パリティー計算プロセス自体のみがパフォーマンスのボトルネックです。最新のCPU とソフトウェアRAID では、最近のCPU がパリティーが非常に高速になるため、通常はボトルネックではありません。ただし、ソフトウェアRAID5 アレイに多数のメンバーデバイスがあり、組み合わせたすべてのデバイス間のデータ転送速度が十分であれば、このボトルネックは再生できます。

レベル4 と同様に、レベル5 のパフォーマンスは非対称であり、読み取りは書き込みを大幅に上回ります。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

パフォーマンスではなくデータの冗長性と保存が最重要事項であるが、レベル1 の領域の非効率性が許容できない場合は、これがRAID の一般的なレベルです。レベル6 では、複雑なパリティースキームを使用して、アレイ内の2 つのドライブから失われたドライブから復旧できます。複雑なパリ

ティースキームにより、ソフトウェア RAID デバイスで CPU 幅が大幅に高くなり、書き込みトランザクションの際に増大度が高まります。したがって、レベル 6 はレベル 4 や 5 よりもパフォーマンスにおいて、非常に非対称です。

RAID レベル 6 アレイの合計容量は、RAID レベル 5 および 4 と同様に計算されますが、デバイス数から追加パリティーストレージ領域用に 2 つのデバイス (1 ではなく) を引きます。

レベル 10

この RAID レベルでは、レベル 0 のパフォーマンスとレベル 1 の冗長性を組み合わせます。また、2 を超えるデバイスを持つレベル 1 アレイで領域の一部を軽減するのに役立ちます。レベル 10 では、データごとに 2 つのコピーのみを格納するように設定された 3 ドライブアレイを作成することができます。これにより、全体用のアレイサイズを最小デバイスのみと同じサイズ (3 つのデバイス、レベル 1 アレイなど) ではなく、最小サイズのデバイスが 1.5 倍になります。

レベル 10 アレイを作成する際の使用可能なオプションの数が多く、特定のユースケースに適切なオプションを選択することが簡単ではないため、インストール時に作成するのは実用的とは言えません。コマンドラインの **mdadm** ツールを使用して手動で作成できます。オプションの詳細と、それぞれのパフォーマンストレードオフに関する詳細は、**man md を参照してください。**

リニア RAID

リニア RAID は、より大きな仮想ドライブを作成するドライブの簡易グループ化です。リニア RAID では、あるメンバードライブからチャンクが順次割り当てられます。最初のドライブが完全に満杯になったときにのみ次のドライブに移動します。これにより、メンバードライブ間の I/O 操作が分割される可能性はないため、パフォーマンスの向上は見られません。リニア RAID には冗長性がなく、実際に、信頼性は低下します。いずれかのメンバードライブに障害が発生した場合は、アレイ全体を使用することができません。容量はすべての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 には、ハードウェア RAID コントローラーに固有の RAID サブシステムがありません。特殊な RAID チップセットを使用するため、ハードウェア RAID コントローラーには独自のドライバーが含まれます。これらのドライバーにより、システムは通常のディスクとして RAID セットを検出できるようになります。

mdraid

mdraid サブシステムは Linux 向けのソフトウェア RAID ソリューションとして設計されており、Linux 下のソフトウェア RAID に対する推奨ソリューションです。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 は外部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 6 は、外部のメタデータで **mdraid** を使用して ISW/IMSM (Intel 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されます。

dmraid

*Device-mapper RAID や **dmraid** はディスクをひとつの RAID セットにまとめるメカニズムを提供するデバイスマッパーのカーネルコードを参照します。この同じカーネルでは 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。したがって、**dmraid** は、さまざまなファームウェア RAID 実装で使用されます。**dmraid** は Intel ファームウェア RAID もサポートしますが、Red Hat Enterprise Linux 6 は **mdraid** を使用して Intel ファームウェア RAID セットにアクセスします。*

17.4. インストーラーにおける RAID サポート

*システム上のハードウェアおよびファームウェア RAID セットはすべて **Anaconda** インストーラーによって自動的に検出され、インストールが行えるようになります。また、**Anaconda** は **mdraid** を使用してソフトウェア RAID に対応しているため、既存の **mdraid** セットを認識することができます。*

***Anaconda** ではインストール時に RAID セットを作成するユーティリティーを提供しています。ただし、このユーティリティーでは新しいセットのメンバーにできるのはパーティションのみになります (ディスク全体とは対照的)。セットにディスク全体を使用するには、ディスク全体にまたがる 1 つのパーティションを作成し、そのパーティションを RAID セットのメンバーとして簡単に使用します。*

*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は、**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に **initrd** に指示します。*

インストール中に RAID を設定する説明は、Red Hat Enterprise Linux 6 『インストールガイド』 を参照してください。

17.5. 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ーを使用する必要があります。詳細は、ハードウェア RAID コントローラーのドライバーのドキュメント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用してソフトウェア RAID の管理を行います (**mdraid**)。**mdadm** モードおよびオプションの詳細は、**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、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ーを使用して 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 セットを作成した後で、**dmraid** ツールによる設定を行うことはできません。**dmraid** の使用に関する詳細は、**man dmraid を参照してください。***

17.6. RAID デバイスの詳細な作成

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。通常、**/boot** または root ファイルシステムのアレイは複雑な RAID デバイスに設定されます。このような場合は、**Anaconda** インストーラーで対応していないアレイオプションの使用が必要になる場合があります。これを回避するには、以下の手順を行います。

手順17.1 RAID デバイスの詳細な作成

1. 通常どおりにインストールディスクを挿入します。
2. 初回起動時に、**Install** または **Upgrade** ではなく、**Rescue Mode** を選択します。システムが **レスキューモード** で完全に起動すると、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、目的のハードドライブに RAID パーティションを作成します。次に **mdadm** を使用して、使用できるすべての設定およびオプションを使用して、このパーティションから RAID アレイを手動で作成します。詳細は、13章Partitions、**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、必要に応じてアレイにファイルシステムを作成することもできます。Red Hat Enterprise Linux 6 でサポートされるファイルシステムに関する基本的な技術情報は、「サポートされるファイルシステムの概要」 を参照してください。
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. システムのディスクの使い方を求められたら、**カスタムレイアウト** を選択して **次へ** をクリックします。デバイス一覧に、既存の MD RAID デバイスが表示されます。
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 (必要に応じて) 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** は、この既存の RAID デバイスにインストールを実行し、**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を保持します。

注記

インストーラーの制限された **レスキューモード** には **man** ページは含まれません。**man mdadm** および **man md** の両方には、カスタムの RAID アレイ作成に役立つ情報が含まれています。回避策全体にわたって必要になる場合があります。このため、**man** ページが含まれるマシンにアクセスできるようにしておくか、そのページを開いて、**レスキューモード** で起動してからカスタムアレイを作成しておくと便利です。

---

[4] ホットスワップシャーシを使用すると、システムの電源を切らずにハードドライブを削除できます。

[5] RAID レベル 1 は、アレイ内のすべてのディスクに同じ情報を書き込むため、コストが高まり、データの信頼性を提供しますが、レベル 5 のようなパリティーベースの RAID レベルよりもはやくなり領域の効率が低くなります。ただし、この領域の非効率性にはパフォーマンス上の利点があります。パリティーベースの RAID レベルは、パリティーを生成するためにかなり多くの CPU 電力を消費しますが、RAID レベル 1 は単に同じデータを、CPU オーバーヘッドが非常に少ない複数の RAID メンバーに同じデータを複数回書き込むだけです。そのため、RAID レベル 1 は、ソフトウェア RAID が使用されているマシンや、マシンの CPU リソースが一貫して RAID アクティビティー以外の操作でアレイ化されます。

[6] パリティー情報は、アレイ内の残りのメンバーディスクのコンテンツに基づいて計算されます。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データの再構築に使用できます。その後、再構築されたデータを使用して、交換前に失敗したディスクに I/O 要求に対応でき、交換後に失敗したディスクを接続します。

第18章 MOUNT コマンドの使い方

Linux、UNIX、および類似のオペレーティングシステムでは、さまざまなパーティションおよびリムーバブルデバイス (CD、DVD、USB フラッシュドライブなど) にあるファイルシステムをディレクトリーツリーの特定のポイント (マウントポイント) に接続して、再度切り離すことができます。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には、**mount** コマンドを実行します。いずれの引数も付けません。

```
mount
```

上記のコマンドで既知のマウントポイントの一覧が表示されます。行ごとにデバイス名、ファイルシステムのタイプ、マウントしているディレクトリー、およびマウントオプションなどの情報が以下のような形で表示されます。

```
ディレクトリー タイプ 上の デバイス （オプション）
```

マウントされたファイルシステムをツリーのような形式でリストできる **findmnt** ユーティリティーは、Red Hat Enterprise Linux 6.1 からも利用できます。現在割り当てられているファイルシステムをすべて表示するには、引数なしで **findmnt** コマンドを実行します。

```
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を持つ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
```

同様に、**findmnt** コマンドを使用して、特定のファイルシステム種別のデバイスのみを表示するには、次のコマンドを実行します。

```
findmnt -t type
```

一般的なファイルシステムタイプの一覧は、表18.1「一般的なファイルシステムのタイプ」 を参照してください。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** を使用するようにフォーマットされます。このファイルシステムを使用するマウントポイントのみを表示するには、シェルプロンプトで以下を入力します。

```
~]$ mount -t ext4
/dev/sda2 on / type ext4 (rw)
/dev/sda1 on /boot type ext4 (rw)
```

findmnt コマンド使用してマウントポイントを一覧表示するには、以下を入力します。

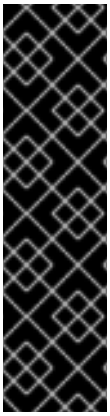
```
~]$ findmnt -t ext4
TARGET SOURCE    FSTYPE OPTIONS
/      /dev/sda2 ext4   rw,realtime,seclabel,barrier=1,data=ordered
/boot  /dev/sda1 ext4   rw,realtime,seclabel,barrier=1,data=ordered
```

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 へのフルパス (例: 「/dev/sda3」)、Universally Unique Identifier (UUID) (例: 「UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb」)、または ボリュームラベル (例: 「LABEL=home」) で識別できます。ファイルシステムがマウントされると、directory の元のコンテンツにアクセスできなくなります。



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 $?
```

ファイルシステムがディレクトリーにアタッチされていない場合は、上記のコマンドは **1** を返します。

必要な情報（デバイス名、ターゲットディレクトリー、またはファイルシステムタイプなし）なしで **mount** コマンドを実行すると、**/etc/fstab** 設定ファイルの内容を読み取り、指定したファイルシステムがリストされたかどうかを確認します。このファイルには、選択したファイルシステムがマウントされるデバイス名およびディレクトリーの一覧と、ファイルシステムのタイプおよびマウントオプションが含まれます。このため、このファイルで指定されたファイルシステムをマウントする場合は、以下のコマンドの以下のいずれかのバリアントを使用できます。

```
mount [option]... directory
mount [option]... device
```

root でコマンドを実行しない限り、ファイルシステムのマウントには権限が必要であることに注意してください (「マウントオプションの指定」 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** に関する情報を表示するには、以下を入力します。*

```
~]# 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

*ほとんどの場合は、**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動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、以下の形式で **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タイプの完全なリストは、 「man ページドキュメント」 に記載の関連マニュアルページを参照してください。*

表18.1 一般的なファイルシステムのタイプ

型	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。
iso9660	**ISO 9660** ファイルシステム。通常、これは光学メディア (通常は CD) で使用されます。
nfs	**NFS** ファイルシステム。通常、これはネットワーク経由でファイルにアクセスするために使用されます。
nfs4	**NFSv4** ファイルシステム。通常、これはネットワーク経由でファイルにアクセスするために使用されます。
udf	**UDF** ファイルシステム。通常、これは光学メディア (通常は DVD) で使用されます。
vfat	**FAT** ファイルシステム。通常、これは Windows オペレーティングシステムを実行しているマシンや、USB フラッシュドライブやフロッピーディスクなどの特定のデジタルメディアで使用されます。

使用例は、例18.2「USB フラッシドライブのマウント」 を参照してください。

> **例18.2 USB フラッシドライブのマウント**
>
> 多くの場合、古い USB フラッシュドライブは FAT ファイルシステムを使用します。このようなドライブが **/dev/sdc1** デバイスを使用し、**/media/flashdisk/** ディレクトリーが存在すると仮定して、**root** でシェルプロンプトに以下を入力し、このディレクトリーにマウントします。
>
> ```
> ~]# mount -t vfat /dev/sdc1 /media/flashdisk
> ```

18.2.2. マウントオプションの指定

追加のマウントオプションを指定するには、以下の形式でコマンドを実行します。

```
mount -o options device directory
```

複数のオプションを指定する場合は、コンマの後にスペースを挿入しないでください。挿入すると、**mount** はスペースに続く値を追加のパラメーターとして誤って解釈します。

表18.2「一般的なマウントオプション」 一般的なマウントオプションの一覧を提供します。利用可能なオプションの一覧は、「man ページドキュメント」 に記載の関連する man ページを参照してください。

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** コマンドを使用したファイルシステムの自動マウントを許可します。
defaults	**async,auto,dev,exec,nouser,rw,suid** のエイリアスを指定します。
exec	特定のファイルシステムでのバイナリーファイルの実行を許可します。
loop	イメージをループデバイスとしてマウントします。
noauto	**mount -a** コマンドを使用し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は許可しません。
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は許可しません。
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。

オプション	詳細
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存在し、**/media/cdrom/** ディレクトリーが存在する場合は、**root** で以下のコマンドを実行してイメージをこのディレクトリーにマウントします。

```
~]# 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しないといけない場合があります (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。さらに、**mount コマンドは、特定のマウント** を複製する手段を提供する **--bind** オプションを実装します。以下のような使用法になります。

```
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追加するには、以下を入力します。

```
mount --rbind old_directory new_directory
```

さらに Red Hat Enterprise Linux 6 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。マウントポイントが共有マウントとしてマークされている場合は、元のマウントポイント内のすべてのマウントが複製マウントポイントに反映されます (その逆も同様です)。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
```

または、選択したマウントポイントと、その下の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場合は、以下を入力します。

> ***mount --make-rshared*** *mount_point*

使用例は、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**
>
> その他のファイルシステムが一般的にマウントされている場所は、リムーバブルメディアの **/media** ディレクトリーと、ファイルシステムを一時的にマウントされた **/mnt** ディレクトリーという 2 つです。共有マウントを使用して、この 2 つのディレクトリーが同じコンテンツを共有できます。これを行うには、**root** で **/media** ディレクトリーを共有 「としてマークします」。
>
> ```
> ~]# mount --bind /media /media
> ~]# mount --make-shared /media
> ```
>
> 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** にも現れていることを確認できます。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することもできます。たとえば、**/dev/sdc1** デバイスを使用する空でないUSB フラッシュドライブが接続されていて、**/mnt/flashdisk/** ディレクトリーが存在する場合は、次のように入力します。
>
> ```
> ~]# 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。マウントポイントがスレーブマウントとしてマークされている場合は、元のマウントポイント内のすべてのマウントが複製マウントポイントに反映されますが、スレーブマウント内のマウントは元のポイントに反映されません。マウントポイントのタイプをスレーブマウントに変更するには、シェルプロンプトで次を入力します。

> ***mount --make-slave*** *mount_point*

選択したマウントポイントとその下にあるすべてのマウントポイントのマウントタイプを変更することも可能です。次のように入力します。

```
mount --make-rslave mount_point
```

使用例は、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

この例は、**/media** ディレクトリーのコンテンツが **/mnt** にも表示され、**/mnt** ディレクトリーのマウントが **/media** に反映されないようにする方法を示しています。**root** になり、まず **/media** ディレクトリーに 「shared」 のマークを付けます。

```
~]# mount --bind /media /media
~]# mount --make-shared /media
```

次に、**/mnt** に複製を作成しますが、「スレーブ」 としてマークします。

```
~]# 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
```

また、**/mnt** ディレクトリー内にマウントされているファイルシステムが **/media** に反映されていないことを確認します。たとえば、**/dev/sdc1** デバイスを使用する空でないUSB フラッシュドライブが接続されていて、**/mnt/flashdisk/** ディレクトリーが存在する場合は、次のように入力します。

```
~]# 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用して **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した。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** デバイスを使用する空でないUSB フラッシュドライブが接続されていて、**/mnt/flashdisk/** ディレクトリーが存在する場合は、次のように入力します。

```
~]# 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、次のようにシェルプロンプトに入力します。

```
mount --make-unbindable mount_point
```

または、選択したマウントポイントとその下にあるすべてのマウントポイントを変更することもできます。

```
mount --make-runbindable mount_point
```

使用例は、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 ディレクトリーを共有しないようにするには、**root** で次のコマンドを実行します。

```
~]# mount --bind /media /media
~]# mount --make-unbindable /media
```

これにより、これ以降はこのマウントを複製しようとすると、エラーが発生して失敗します。

```
~]# 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。

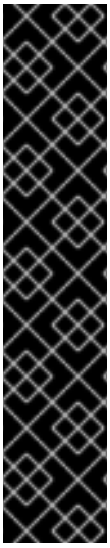
```
~]# ls /mnt/userdirs
~]# ls /home
jill  joe
```

18.3. 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は、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力して失敗します。次のように **fuser** コマンドを使用し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
```

たとえば、**/media/cdrom/** ディレクトリーにマウントされたファイルシステムにアクセスするプロセスを一覧表示するには、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

以前に **/media/cdrom/** ディレクトリーにマウントされた CD をアンマウントするには、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 ページドキュメント

- **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. 便利な 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 関数は、libvolume_key と **volume_key** の 2 つのツールを提供します。libvolume_key は、ストレージボリュームの暗号化キーを操作し、ボリュームとは別に保存するためのライブラリーです。**volume_key** は、暗号化したハードドライブへのアクセスを復元するために鍵およびパスフレーズを抽出するために使用される関連するコマンドラインツールです。

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 6 サーバーの標準インストールには含まれません。インストール方法については、http://fedoraproject.org/wiki/Disk_encryption_key_escrow_use_cases を参照してください。

19.1. 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 に対する操作におけるオペランドおよびモードは、以下のオプションのいずれかを指定して決定されます。

--save

このコマンドは、オペランド ボリューム [packet] を想定します。packet を指定すると、**volume_key** は、そこからキーおよびパスフレーズを抽出します。packet が指定しないと、**volume_key** は volume からキーおよびパスフレーズを抽出し、必要に応じてユーザーに要求します。これらの鍵とパスフレーズは、1 つまたは複数の出力パケットに保存されます。

--restore

このコマンドは、オペランド ボリュームパケット を想定します。次に volume を開き、packet のキーとパスフレーズを使用して volume を再度アクセス可能にし、必要に応じてユーザー (たとえば、ユーザーが新しいパスフレーズの入力を許可するなど) を求めるプロンプトを出します。

--setup-volume

このコマンドは、オペランドの ボリュームパケット名 を想定します。次に volume を開き、packet のキーとパスフレーズを使用して、復号されたデータを name として使用するための volume を設定します。

name は、dm-crypt ボリュームの名前です。この操作は、復号されたボリュームを **/dev/mapper/name** として利用可能にします。

この操作は、新しいパスフレーズを追加すると ボリュームを 永続的に変更しません（例：）。ユーザーは復号された volume にアクセスし、変更できます。

--reencrypt、--secrets、および --dump

*これらの 3 つのコマンドは、さまざまな出力方法で同様の機能を実行します。それぞれオペランド パケット が必要で、それぞれが パケット を開き、必要に応じて復号化を行います。次に 、**--reencrypt** は情報を 1 つ以上の新規出力パケットに保存します。**--secrets** は、パケット に含まれるキーおよびパスフレーズを出力します。**--dump** は パケット の内容を出力しますが、キーおよびパスフレーズはデフォルトで出力されません。これは、コマンドに **--with-secrets** を追加することで変更できます。また、--unencrypted **コマンドを使用すると**、パケットの暗号化されていない部分のみをダンプできます。パスフレーズや秘密鍵のアクセスは必要ありません。*

各オプションは、以下のオプションで追加できます。

-o, --output packet

*このコマンドは、デフォルトの鍵またはパスフレーズを パケット 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、ボリュームの形式により異なります。有効期限が切れないものではなく、**--restore** によるボリュームへのアクセスを復元できるようにしてください。*

--output-format format

このコマンドは、すべての出力パケットに指定された format 形式を使用します。現在、format は以下のいずれかになります。

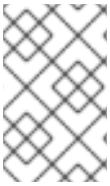
- ***asymmetric**: CMS を使用してパケット全体を暗号化し、証明書を必要とします。*
- ***asymmetric_wrap_secret_only**: シークレットまたはキーおよびパスフレーズのみをラップし、証明書を必要とします。*
- ***パスフレーズ**: GPG を使用してパケット全体を暗号化し、パスフレーズが必要です。*

--create-random-passphrase packet

このコマンドは、ランダムの英数字のパスフレーズを生成し、ボリュームに追加して （他のパスフレーズに影響を与えずに）、このランダムパスフレーズを パケット に保存します。

19.2. VOLUME_KEY を個別ユーザーとして使用する手順

*個々のユーザーとして、以下の手順に従って、**volume_key** を使用して暗号化キーを保存できます。*



注記

*このファイルのすべての例では、**/path/to/volume** は LUKS デバイスであり、内に含まれる平文デバイスではありません。**blkid -s type /path/to/volume** should report **type="crypto_LUKS"**.*

手順19.1 volume_key スタンドアロンの使用

1. *以下を実行します。*

```
volume_key --save /path/to/volume -o escrow-packet
```

次に、キーを保護するために、パケット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. *generatedescrow **-packet** ファイルを保存して、パスフレーズが記憶されないようにします。*

ボリュームのパスフレーズが記憶されたら、保存された escrow パケットを使用し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使用したデータへのアクセスの復元

1. **volume_key を実行できる環境でシステムを起動し**、エスクローパケットが利用できる（例：レスキューモード）
2. 以下を実行します。

```
volume_key --restore /path/to/volume escrow-packet
```

エスクローパケットの作成時に使用された escrow パケットパスフレーズ、およびボリュームの新しいパスフレーズ用に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用してボリュームをマウントします。

暗号化されたボリュームの LUKS ヘッダーでパスフレーズスロットを解放するには、**cryptsetup luksKillSlot** を使用して、以前のパスフレーズを削除します。

19.3. 大規模な組織での VOLUME_KEY の使用

大型の組織では、各システム管理者が認識するパスワード 1 つを使用して、各システムに個別のパスワードを追跡することは難しいため、セキュリティーリスクが伴います。これをカウンターするために、**volume_key** は非対称暗号を使用して、どのコンピューターで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っている人の数を最小限に抑えます。

本セクションでは、暗号化キーの保存、暗号鍵の保存、ボリュームへのアクセスの復元、緊急パスフレーズの設定に必要な手順について説明します。

19.3.1. 暗号化キーを保存するための準備

暗号化キーの保存を開始するために、準備が必要になります。

手順19.3 準備

1. X509 証明書/ プライベートのペアを作成します。
2. 秘密鍵を侵害しないように信頼できるユーザーを指定します。これらのユーザーは escrow パケットを復号化できます。
3. escrow パケットの復号に使用するシステムを選択します。このシステムでは、秘密鍵が含まれる NSS データベースを設定します。

   秘密鍵が NSS データベースに作成されていない場合は、以下の手順に従います。

   - 証明書および秘密鍵を **PKCS#12** ファイルに保存します。
   - 以下を実行します。

```
certutil -d /the/nss/directory -N
```

この時点で、NSS データベースパスワードを選択できます。各 NSS データベースは異なるパスワードを持つことができるため、指定したユーザーは各ユーザーが別の NSS データベースが使用される場合に、単一のパスワードを共有する必要はありません。

- 以下を実行します。

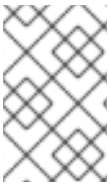
```
pk12util -d /the/nss/directory -i the-pkcs12-file
```

4. システムのインストールや既存システムに鍵を保存したり、証明書を配布します。

5. 保存された秘密鍵については、マシンおよびボリュームで検索できるストレージを準備します。たとえば、マシンごとに1つのサブディレクトリーを持つ簡単なディレクトリーや、他のシステム管理タスクに使用されるデータベースも指定できます。

19.3.2. 暗号化キーの保存

必要な準備が完了した後（「暗号化キーを保存するための準備」を参照）、以下の手順に従って暗号化キーを保存することができるようになりました。



注記

このファイルのすべての例では、**/path/to/volume** は LUKS デバイスであり、これに含まれるプレーンテキストデバイスではありません。**blkid -s type /path/to/volume** は **type="crypto_LUKS"** を報告します。

手順19.4 暗号化キーの保存

1. 以下を実行します。

```
volume_key --save /path/to/volume -c /path/to/cert escrow-packet
```

2. 準備済みストレージに generatedescrow **-packet** ファイルを保存して、システムとボリュームに関連付けます。

この手順は手動で実行することも、システムのインストールの一部としてスクリプト化したりできます。

19.3.3. ボリュームへのアクセスの復元

暗号化キーの保存後（「暗号化キーを保存するための準備」 および 「暗号化キーの保存」）は、必要に応じてドライバーへのアクセスを復元できます。

手順19.5 ボリュームへのアクセスの復元

1. パケットストレージからボリュームのエスクローパケットを取得し、これを復号化用に指定されたユーザーの1つに送信します。

2. 指定されたユーザーは以下を実行します。

```
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指定したら、指定したユーザーは、暗号化用のパスフレーズを選択して、escrow **-packet-out** です。このパスフレーズは毎回異なる可能性があり、指定したユーザーからターゲットシステムに移動された間、暗号鍵を保護することができます。

3. 指定したユーザーから、**escrow-packet-out** ファイルとパスフレーズを取得します。

4. ***volume_key*** *を実行し、レスキューモードなどでescrow* ***-packet-out*** *ファイルを利用できる環境でターゲットシステムを起動します。*

5. *以下を実行します。*

```
volume_key --restore /path/to/volume escrow-packet-out
```

指定されたユーザーが選択したパケットパスフレーズ、およびボリュームの新しいパスフレーズ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を使用してボリュームをマウントします。*

cryptsetup luksKillSlot *を使用して記憶されている古いパスフレーズを削除することも可能です。たとえば、暗号化されたボリュームのLUKS ヘッダーで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、コマンド* ***cryptsetup luksKillSlot device key-slot*** *コマンドで行います。詳細および例については、* ***cryptsetup --help*** *を参照してください。*

19.3.4. 緊急パスフレーズの設定

（ビジネスの移動など）場合によっては、システム管理者が影響を受けるシステムと直接動作させるのが難しい場合がありますが、ユーザーは引き続きそのデータにアクセスする必要があります。この場合、 ***volume_key*** *は、パスフレーズと暗号鍵と連携できます。*

システムのインストール時に、以下を実行します。

```
volume_key --save /path/to/volume -c /path/to/ert --create-random-passphrase passphrase-packet
```

これにより、ランダムの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、そのパスフレーズが ***passphrase-packet*** *に保存されます。* ***--create-random-passphrase*** *オプションおよび* ***-o*** *オプションを組み合わせて両方のパケットを同時に生成することもできます。*

パスワードを忘れると、指定したユーザーは以下を実行します。

```
volume_key --secrets -d /your/nss/directory passphrase-packet
```

これにより、ランダムのパスフレーズが表示されます。このパスフレーズをエンドユーザーに付与します。

19.4. VOLUME_KEY 参照 S

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*** *で man* ***volume_key を使用する man*** *ページ*
- *オンライン時刻 http://fedoraproject.org/wiki/Disk_encryption_key_escrow_use_cases*

第20章 アクセス制御リスト

ファイルとディレクトリーには、ファイルの所有者、そのファイルに関連したグループ、およびシステムを使用する他のすべてのユーザーの権限セットが設定されます。しかし、これらの権限には制限があります。たとえば、ユーザーごとに異なる権限を設定することはできません。そのため アクセス制御リスト (ACL) が実装されています。

Red Hat Enterprise Linux カーネルは、ext3 ファイルシステムと NFS でエクスポートしたファイルシステムに対して ACL サポートを提供します。ACL は、Samba 経由でアクセスする ext3 ファイルシステムでも認識されます。

ACL の実装には、カーネルでのサポートと **acl** パッケージが必要になります。このパッケージには、ACL 情報の追加、修正、削除および、取得のためのユーティリティーが同梱されています。

cp コマンドと **mv** コマンドは、ファイルとディレクトリーに関連するすべての ACL のコピーまたは移動を実行します。

20.1. ファイルシステムのマウント

ファイルやディレクトリー用に ACL を使用する前に、そのファイルまたはディレクトリーのパーティションを ACL サポートでマウントする必要があります。ローカルの ext3 ファイルシステムの場合は、以下のコマンドでマウントできます。

mount -t ext3 -o acl device-name partition

以下に例を示します。

mount -t ext3 -o acl /dev/VolGroup00/LogVol02 /work

もしくは、パーティションが **/etc/fstab** ファイルにリストされている場合は、パーティションのエントリーに **acl** オプションを含むことができます。

```
LABEL=/work     /work       ext3    acl        1 2
```

Samba 経由で ext3 ファイルシステムにアクセスし、そのアクセスに対して ACL が有効になっている場合は、ACL が認識されます。これは、**--with-acl-support** オプションでコンパイルされているためです。Samba 共有のアクセス時またはマウント時に特別なフラグは必要ありません。

20.1.1. NFS

デフォルトでは、NFS サーバーでエクスポートされているファイルシステムが ACL をサポートし、NFS クライアントが ACL を読み込める場合は、クライアントシステムで ACL が使用されます。クライアントに NFS 共有をマウントする際に ACL を無効にするには、コマンドラインで **noacl** オプションでマウントします。

20.2. アクセス ACL の設定

ACL には、アクセス ACL と デフォルト ACL と 2 つのタイプがあります。アクセス ACL は、特定のファイルまたはディレクトリーに対するアクセス制御リストです。デフォルト ACL は、ディレクトリーにのみ適用されます。ディレクトリー内のファイルにアクセス ACL が設定されていない場合は、そのディレクトリーにデフォルト ACL のルールが適用されます。デフォルト ACL は任意です。

ACL は以下のように設定できます。

1. 各ユーザー
2. 各グループ
3. 実効権マスクを使用して
4. ファイルのユーザーグループに属さないユーザーに対して

setfacl ユーティリティーは、ファイルとディレクトリー用の ACL を設定します。**-m** オプションを使用すると、ファイルまたはディレクトリーの ACL を追加または修正できます。

```
# setfacl -m rules files
```

ルール (rules) は、以下の形式で指定する必要があります。複数のルールをコンマで区切って同じコマンドに指定することもできます。

u:uid:perms

ユーザーにアクセス ACL を設定します。ユーザー名または UID を指定できます。システムで有効な任意のユーザーを指定できます。

g:gid:perms

グループにアクセス ACL を設定します。グループ名または GID を指定できます。システムで有効な任意のグループを指定できます。

m:perms

実効権マスクを設定します。このマスクは、所有グループの全権限と、ユーザーおよびグループの全エントリーを結合したものです。

o:perms

ファイルのグループに属さないユーザーにアクセス ACL を設定します。

権限 (perms) は、読み取り、書き込み、および実行を表す **r**、**w**、および **x** の文字の組み合わせで表示されます。

ファイルまたはディレクトリーにすでに ACL が設定されている状態で、**setfacl** コマンドを使用した場合は、設定するルールが既存の ACL に追加されるか、既存のルールが修正されます。

例20.1 読み取りと書き込みの権限付与

たとえば、ユーザー「andrius」に読み取りと書き込みの権限を付与するには以下を実行します。

```
# setfacl -m u:andrius:rw /project/somefile
```

ユーザー、グループ、またはその他のユーザーからすべての権限を削除するには、**-x** オプションにいずれの権限も指定せずにコマンドを実行します。

```
# setfacl -x rules files
```

例20.2 すべての権限の削除

たとえば、UID 500 のユーザーからすべての権限を削除するには以下を実行します。

```
# setfacl -x u:500 /project/somefile
```

20.3. デフォルト ACL の設定

デフォルト ACL を設定するには、**d:** をルールの前に追加してから、ファイル名ではなくディレクトリー名を指定します。

例20.3 デフォルト ACL の設定

たとえば、**/share/** ディレクトリーにデフォルト ACL を設定し、ユーザーグループに属さないユーザーの読み取りと実行を設定するには、以下のコマンドを実行します (これにより、個別ファイルのアクセス ACL が上書きされます)。

```
# setfacl -m d:o:rx /share
```

20.4. ACL の取り込み

ファイルまたはディレクトリーに設定されている既存の ACL を確認するには、**getfacl** コマンドを使用します。以下の例では、**getfacl** でファイルの既存の ACL を確認します。

例20.4 ACL の取り込み

```
# getfacl home/john/picture.png
```

上記のコマンドは、次のような出力を返します。

```
# file: home/john/picture.png
# owner: john
# group: john
user::rw-
group::r--
other::r--
```

ディレクトリーにデフォルト ACL が指定されている場合は、以下のようにデフォルト ACL も表示されます。たとえば、**getfacl home/sales/** を実行すると以下のような出力になります。

```
# file: home/sales/
# owner: john
# group: john
user::rw-
user:barryg:r--
group::r--
mask::r--
other::r--
default:user::rwx
default:user:john:rwx
```

```
default:group::r-x
default:mask::rwx
default:other::r-x
```

20.5. ACL が設定されているファイルシステムのアーカイブ作成

デフォルトでは、**dump** コマンドによるバックアップ操作時に ACL が保存されます。**tar** コマンドで、ファイルまたはファイルシステムのアーカイブを作成する場合は、**--acls** オプションを付けて ACL を保存します。同様に、**cp** コマンドで、ACL が設定されているファイルをコピーする場合は、**--preserve=mode** オプションを付けて ACL もコピーされるようにします。さらに、**cp** の **-a** オプション (**-dR --preserve=all** と同等) も、バックアップ時にタイムスタンプ、SELinux コンテキストなどの情報と一緒に ACL を保存します。**dump**、**tar**、または **cp** の詳細は、それぞれの **man** ページを参照してください。

star ユーティリティーは、ファイルのアーカイブ生成に使用される点で **tar** ユーティリティーと似ています。しかし、一部のオプションは異なります。最も一般的に使用されるオプションの一覧は 表 20.1「**star** のコマンドラインオプション」 を参照してください。すべての利用可能なオプションは、**man star** を参照してください。このユーティリティーを使用するには **star** パッケージが必要になります。

表20.1 star のコマンドラインオプション

オプション	詳細
-c	アーカイブファイルを作成します。
-n	ファイルを抽出しません。**-x** と併用すると、ファイルが行う抽出を表示します。
-r	アーカイブ内のファイルを入れ替えます。パスとファイル名が同じファイルが置き換えられ、アーカイブファイルの末尾に書き込まれます。
-t	アーカイブファイルのコンテンツを表示します。
-u	アーカイブファイルを更新します。このファイルは、以下で指定されたアーカイブの最後に書き込まれます。 ● アーカイブにまだ存在しません。 ● 更新されるファイルは、アーカイブにすでにある同じ名前のファイルよりも新しい順です。 このオプションは、アーカイブがファイルであるか、または（0 に戻ってではなく）逆になることができるブロックされていないテープの場合にのみ機能します。
-x	アーカイブからファイルを抽出します。**-U** と併用すると、アーカイブ内のファイルがファイルシステムにあるファイルよりも古い場合、そのファイルは抽出されません。
-help	最も重要なオプションを表示します。

オプション	詳細
-xhelp	最も重要ではないオプションを表示します。
-/	アーカイブからファイルを抽出する際に、ファイル名から先頭のスラッシュを削除します。デフォルトでは、ファイルの抽出時に先頭のスラッシュが削除されます。
-acl	作成時または抽出時に、ファイルとディレクトリーに関連付けられているすべてのACL をアーカイブするか、または復元します。

20.6. 旧システムとの互換性

指定したファイルシステムのいずかのファイルにACL が設定されている場合、そのファイルシステムには **ext_attr** 属性があります。この属性は、以下のコマンドを使用すると確認できます。

```
# tune2fs -l filesystem-device
```

ext_attr 属性を持つファイルシステムは古いカーネルでマウントできますが、それらのカーネルは設定されているACL を強制しません。

バージョン1.22 以降の **e2fsprogs** パッケージ(Red Hat Enterprise Linux 2.1 および4 のバージョンも含む) に含まれている **e2fsck** ユーティリティーのバージョンは、**ext_attr** 属性を使用してファイルシステムを確認できます。古いバージョンではこの確認が拒否されます。

20.7. ACL 参照情報

詳細情報は以下のman ページを参照してください。

- **man acl**: ACL の説明
- **man getfacl**: ファイルアクセス制御リストの取得方法
- **man setfacl**: ファイルアクセス制御リストの設定方法
- **man star**: **star** ユーティリティーとそのオプションの詳細説明

第21章 ソリッドステートディスクデプロイメントのガイドライン

使用されるブロック数は、ディスク容量に近づけるため、パフォーマンスが低下します。パフォーマンスへの影響度は、ベンダーによって大きく異なります。ただし、すべてのデバイスでパフォーマンスが低下する可能性があります。

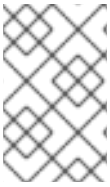
パフォーマンスの低下に対処するために、ホストシステム（Linux カーネルなど）は破棄要求を使用して、特定のブロックが使用されていないことをストレージに通知します。SSD はこの情報を使用して、対象レベルに空きブロックを使用して、内部で領域を解放できます。破棄は、ストレージプロトコル（ATA または SCSI 経由）でストレージサポートを公開する場合にのみ発行されます。破棄要求は、ストレージプロトコル固有の negotiated discard コマンドを使用してストレージに発行されます（ATA の場合は**TRIM コマンド**、および **UNMAP セットでの WRITE SAME**、SCSI 用の **UNMAP** コマンド）。

破棄 サポートを有効にすると、ファイルシステムに利用可能な空き領域があるものの、ファイルシステムは、基礎となるストレージデバイスのほとんどの論理ブロックに既に書き込まれている場合に便利です。**TRIM** の詳細は、以下のリンクの 『Data Set Management T13 Specifications』 を参照してください。

http://t13.org/Documents/UploadedDocuments/docs2008/e07154r6-Data_Set_Management_Proposal_for_ATA-ACS2.doc

UNMAP の詳細は、以下のリンクの 『SCSI Block Commands 3 T10 Specification』 のセクション 4.7.3.4 を参照してください。

http://www.t10.org/cgi-bin/ac.pl?t=f&f=sbc3r26.pdf



注記

市場のすべてのソリッドステートデバイスが、**破棄** サポートを持つわけではありません。ソリッドステートデバイスが破棄されたかどうかを判断するには、**/sys/block/sda/queue/discard_granularity を確認してください。**

21.1. デプロイメントに関する考慮事項

SSD の内部レイアウトと操作により、内部 消去ブロック境界 のパーティション作成が最適です。Red Hat Enterprise Linux 6 のパーティショニングユーティリティーが、SSD トポロジー情報をエクスポートする場合は、デフォルトの sane を選択します。

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**、最初のパーティションを 1MB の境界に作成することを推奨します。

Red Hat Enterprise Linux 6.5 MD MD が破棄要求を渡すようになりました。6.5 より前のバージョンでは、サポート対象外でした。一方、LVM が、論理ボリュームマネージャー(LVM)と、LVM が破棄をサポートしているデバイスマッパー(DM)ターゲットになります。破棄をサポートしない唯一の DM ターゲットは、dm-snapshot、dm-crypt、および dm-raid45 です。Red Hat Enterprise Linux 6.1 に、dm-mirror のサポートが破棄されました。

Red Hat は、SSD 上の LVM RAID に RAID1 または RAID10 を使用することを推奨しています。このレベルは破棄をサポートしているためです。他の RAID レベルの初期化段階では、一部の RAID 管理ユーティリティー（**mdadm**など）が、ストレージデバイス **上のすべての** ブロックへの書き込みを行い、そのチェックサムが適切に動作するようになります。これにより、SSD のパフォーマンスがすぐに低下します。

注記

*RAID1、RAID10、およびパリティー RAID で **--nosync** オプションを使用すると、最初の書き込みが進んだ分のストライプが計算されるため、一貫性が保たれます。ただし、スクラビング操作を実行する場合は、書き込まれていない部分 **が一致しない/inconsistent** としてカウントされます。*

*Red Hat Enterprise Linux 6.4 では、ext4 および XFS は、**破棄に対応する唯一の完全対応ファイルシステムです**。以前のバージョンの Red Hat Enterprise Linux 6 は、ext4 は完全にサポートされている **discard です**。デバイスで **discard** コマンドを有効にするには、**マウントオプション discard** を使用します。たとえば、**discard** を有効にして **/dev/sda2** を **/mnt** にマウントするには、次のコマンドを実行します。*

```
# mount -t ext4 -o discard /dev/sda2 /mnt
```

*デフォルトでは、ext4 は **discard** コマンドを発行しません。これはほとんどの場合、**破棄** コマンドを適切に実装できないデバイスの問題を回避するためのものです。Linux **swap** コードは **discard** コマンドで **-**enabled デバイスを破棄し、この動作を制御するオプションはありません。*

21.2. チューニングの留意事項

このセクションでは、SSD のパフォーマンスに影響する可能性がある設定を設定する際に考慮すべき要因について説明します。

I/O スケジューラー

I/O スケジューラーは、ほとんどの SSD で適切に動作するはずです。ただし、Red Hat は、他のストレージタイプと同様に、特定のワークロードに最適な設定を決定するためにベンチマーク設定を行うことを推奨します。

Red Hat は、SSD を使用する場合には、特定のワークロードのベンチマーク設定を行う場合のみ、I/O スケジューラーを変更することを推奨します。各種タイプの I/O スケジューラーについての詳細は、I『I/O の調整ガイド』（Red Hat が提供）を参照してください。以下のカーネルドキュメントには、I/O スケジューラー間の切り替え方法が記載されています。

/usr/share/doc/kernel-version/Documentation/block/switching-sched.txt

仮想メモリー

*I/O スケジューラーと同様に、仮想メモリー (VM) サブシステムには特別なチューニングは必要ありません。SSD の I/O の高速性質を考慮すると、書き込みアウトアクティビティーの増加がディスク上の他の操作のレイテンシーに悪影響を及ぼすことはないため、**vm_dirty_background_ratio** および **vm_dirty_ratio** 設定をオフにする必要があります。ただし、これにより **全体的な I/O** が生成されます。したがって、ワークロード固有のテストを使用せずに通常は推奨されません。*

スワップ

SSD は swap デバイスとしても使用できます。これは、適切なページアウト/ページ内のパフォーマンスを生成する可能性があります。

第22章 書き込みバリア

*書き込みバリア は、揮発性書き込みキャッシュのあるストレージデバイスが電源を失った場合でも、ファイルシステムのメタデータが正しく書き込まれ、永続ストレージに順序付けられます。また、書き込みバリアが有効にされたファイルシステムは、**fsync（）を介して送信されるデータ** が電源損失時に永続的になるようにします。*

*書き込みバリアを有効にすると、一部のアプリケーションにパフォーマンスが大幅に低下します。具体的には、**fsync（）** を大きく使用したり、小規模なファイルを作成したり、削除したりするアプリケーションは、非常に遅くなる可能性があります。*

22.1. WRITE BARRIERS の重要性

ファイルシステムは、メタデータを安全に更新して、一貫性を確保できるように細心の注意を払ってください。ジャーナル化されたファイルシステムバンドルメタデータがトランザクションに更新され、以下の方法で永続ストレージに送信します。

1. *まず、ファイルシステムはトランザクションのボディーをストレージデバイスに送信します。*
2. *次に、ファイルシステムはコミットブロックを送信します。*
3. *トランザクションと対応する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、トランザクションがpower failure 後も存続することを想定します。*

ただし、電源の失敗時のファイルシステムの整合性は、追加のキャッシュを持つストレージデバイスの複雑になります。ローカルのS-ATA やSAS ドライブなどのストレージターゲットデバイスは、サイズが32MB から64MB（最新のドライブで）書き込みキャッシュを持つ可能性があります。ハードウェアRAID コントローラーには、多くの場合、内部書き込みキャッシュが含まれます。さらに、NetApp、IBM、Hitachi、EMC（その他）には大きなキャッシュがあります。

キャッシュに格納されると、データの書き込みキャッシュI/O が「complete」として報告します。キャッシュが機能しなくなった場合は、データも失われます。キャッシュデステージから永続ストレージへのキャッシュのデメンテーションであるため、元のメタデータの順番が変わる可能性があります。これが発生すると、完全なトランザクションに関連するトランザクションがなくてもコミットブロックがディスクに存在する可能性があります。これにより、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非初期化されたトランザクションブロックをファイルシステムに再生できます。これにより、データの不整合や破損が発生します。

Write Barriers の仕組み

*書き込みバリアは、I/O の前後にストレージ書き込みキャッシュのフラッシュによりLinux カーネルで実装されます（**順番は重要** ）。トランザクションが書き込まれると、ストレージキャッシュがフラッシュされ、コミットブロックが書き込まれ、キャッシュは再度フラッシュされます。これにより、以下が確保されます。*

- *ディスクにはすべてのデータが含まれます。*
- *順序の並べ替えは発生していません。*

*バリアが有効になると、**fsync（）** 呼び出しでストレージキャッシュのフラッシュも発行されます。これにより、**fsync()** が返された直後に電源損失が発生した場合でも、ファイルデータがディスク上の永続化されるようになります。*

22.2. WRITE BARRIERS の有効化/無効化

電源の損失時にデータ破損のリスクを軽減するには、ストレージデバイスの中には、バッテリーでバックアップされた書き込みキャッシュを使用します。通常、ハイエンドアレイや一部のハードウェアコントローラーはバッテリーベースの書き込みキャッシュを使用します。ただし、キャッシュのボランティはカーネルには表示されません。したがって、Red Hat Enterprise Linux 6 は、サポートされているすべてのジャーナルファイルシステムで、デフォルトで書き込みバリアを有効にします。

不揮発性なデバイスと、バッテリーでバックアップされた書き込みキャッシュと、write-caching が無効になっているデバイスの場合、マウントに **-o nobarrier** オプションを使用して、マウント時の書き込みバリアを安全に無効にできます。ただし、一部のデバイスは書き込みバリアをサポートしていません。このようなデバイスは、エラーメッセージを **/var/log/messages** に記録します（ 表22.1「ファイルシステムごとの書き込みバリヤーエラーメッセージ」を参照）。

表22.1 ファイルシステムごとの書き込みバリヤーエラーメッセージ

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**

注記

Red Hat Enterprise Linux 6 では、書き込みバリアのネガティブなパフォーマンスへの影響（約3%）となるため、Red Hat Enterprise Linux 6 では **nobarrier** を使用することは推奨していません。通常、書き込みバリアの利点は、無効にしたパフォーマンス上のメリットを得られます。また、仮想マシンに設定されるストレージでは、**barrier** オプションは使用しないでください。

22.3. アバージーの作成に関する留意事項

一部のシステム設定では、データを保護する書き込みバリアは必要ありません。ほとんどの場合、書き込みバリアを有効にすると、パフォーマンスが大幅に向上するため、その他の方法がバリアを書くことが推奨されます。

書き込みキャッシュの無効化

また、データの整合性の問題を回避する1つの方法は、書き込みキャッシュがpower failures 時にデータを損失しないようにすることです。これを設定する最善の方法は、単に書き込みキャッシュを無効にすることです。1つ以上のSATA ドライブ（ローカルのSATA コントローラーIntel AHCI 部分をオフに）を備えた簡単なサーバーまたはデスクトップでは、以下のように、h **dparm** コマンドを使用して、ターゲットSATA ドライブ上の書き込みキャッシュを無効にすることができます。

```
# hdparm -W0 /device/
```

バッテリーバックグラウンド書き込みキャッシュ

また、システムがバッテリーベース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する場合は、書き込みバリアも不要になります。システムがこのようなコントローラーで同等で、

そのコンポーネントドライブで書き込みキャッシュが無効になった場合、コントローラーは自身をライトスルーキャッシュとしてアドバタイズします。これにより、書き込みキャッシュデータが電源の損失後も存続することをカーネルに通知します。

*ほとんどのコントローラーはベンダー固有のツールを使用して、ターゲットドライブをクエリーし、操作します。たとえば、LSI Megaraid SAS コントローラーはバッテリーベースの書き込みキャッシュを使用します。このタイプのコントローラーは、ターゲットドライブを管理するために **MegaCli64** ツールを必要とします。LSI Megaraid SAS 用のすべてのバックエンドドライブの状態を表示するには、以下のコマンドを使用します。*

```
# MegaCli64 -LDGetProp  -DskCache  -LAll -aALL
```

LSI Megaraid SAS 用のすべての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には、以下のコマンドを使用します。

```
# MegaCli64 -LDSetProp -DisDskCache -LaII -aALL
```

注記

ハードウェアRAID カードは、システムが稼働中にバッテリーを再課金します。システムの電源が長期間オフになっている場合、バッテリーはその課金を失い、電源異常中に脆弱なデータを残します。

high-End Arrays

ハイエンドアレイは、電源障害時にデータを保護するさまざまな方法を提供します。そのため、外部RAID ストレージで内部ドライブの状態を確認する必要はありません。

NFS

NFS クライアントでは、データの整合性がNFS サーバー側によって処理されるため、書き込みバリアを有効にする必要はありません。そのため、（書き込みバリアやその他の手段を介して）電源損失全体でデータの永続性を保証するようにNFS サーバーを設定する必要があります。

第23章 ストレージI/O アライメントとサイズ

SCSI およびATA 規格に対する最近の機能拡張により、ストレージデバイスが優先される（必要な場合は必須）の I/O 調整 と I/O サイズ を示すことができます。この情報は、物理セクターサイズを 512 バイトから 4k バイトに増やす新しいディスクドライブで特に便利です。この情報は、RAID デバイスにも利点があります。この場合、チャンクサイズとストライプサイズがパフォーマンスに影響を及ぼす可能性があります。

Linux I/O スタックが強化され、ベンダー提供の I/O 調整と I/O サイズ情報を処理し、ストレージ管理ツール（**parted**、**lvm**、**mkfs など）を処理し、** データの配置とアクセスを最適化できるようになりました。レガシーデバイスが I/O アライメントとサイズデータをエクスポートしていない場合は、Red Hat Enterprise Linux 6 のストレージ管理ツールは伝統的に 4k (または 2 の累乗) の境界に I/O を調整します。これにより、必要な I/O 調整およびサイズを指定しなくても、4k-sector デバイスが正しく動作できるようになります。

デバイスから取得したオペレーティングシステムに関する情報を確認するには、「ユーザー空間アクセス」 を参照してください。このデータは、データの配置を決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。

Red Hat Enterprise Linux 7 の IO スケジューラーが変更されました。デフォルトの IO スケジューラーは Deadline （SATA ドライブを除く）になりました。CFQ は、SATA ドライブのデフォルト IO スケジューラーです。ストレージの速度を短縮するために、Deadline outperforms CFQ 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向上されます。

デフォルトが一部のディスク（SAS ローテーションディスクなど）に適切な場合は、IO スケジューラーを CFQ に変更します。このインスタンスはワークロードによって異なります。

23.1. ストレージアクセスのパラメーター

オペレーティングシステムは以下の情報を使用して、I/O 調整とサイズを決定します。

physical_block_size

デバイスが動作できる最小の内部ユニット

logical_block_size

デバイス上の場所指定に外部で使用される

alignment_offset

基礎となる物理的なアライメントのオフセットとなる Linux ブロックデバイス (パーティション/MD/LVM デバイス) の先頭部分のバイト数

minimum_io_size

ランダムな I/O に対して推奨のデバイス最小ユニット

optimal_io_size

I/O ストリーミングにデバイスの優先単位

たとえば、特定の 4K セクターデバイスは、内部的に 4K **physical_block_size** を使用し、より粒度の 512 バイトの **logical_block_size** を Linux に公開します。この不一致により、不適切な I/O の可能性が発生します。これに対処するために、Red Hat Enterprise Linux 6**の I/O スタックは、ブロックデバイスの最初が基礎となる物理**アライメントからオフセットである場合に alignment_offset がアジュアルされた境界ですべてのデータ領域の起動を試みます。

*また、ストレージベンダーは、デバイスのランダム I/O (**minimum_io_size**) やストリーミング I/O (**optimal_io_size**) に適した最小単位に関する I/O ヒント を提供することができます。たとえば、**minimum_io_size** および **optimal_io_size** は、それぞれ RAID デバイスのチャンクサイズとストライプのサイズに対応する場合があります。*

23.2. ユーザー空間アクセス

*常に適切に調整された I/O を使用し、サイズ付けされた I/O を使用するようにしてください。これは、ダイレクト I/O アクセスに特に重要です。ダイレクト I/O は **logical_block_size** の境界で調整する必要があり、また **logical_block_size の倍数に調整する必要があります**。*

*ネイティブ 4K デバイス（つまり **logical_block_size** が 4K）の場合、アプリケーションはデバイスの **logical_block_size の倍数に対してダイレクト I/O を実行することが重要になります**。つまり、アプリケーションは 4k-aligned I/O ではなく、512 バイトのアラインメント I/O を実行するネイティブの 4k デバイスで失敗します。*

*これを回避するには、適切な I/O 調整とサイズを使用するように、アプリケーションはデバイスの I/O パラメーターを参照してください。上記のように、I/O パラメーターは **sysfs** インターフェースおよびブロックデバイス **ioctl** インターフェースの両方で公開されます。*

*詳細は **man libblkid** を参照してください。この **man** ページは、**libblkid-devel** パッケージで提供されます。*

sysfs インターフェース

- */sys/block/**disk**/alignment_offset*
- */sys/block/**disk**/**partition**/alignment_offset*
- */sys/block/**disk**/queue/physical_block_size*
- */sys/block/**disk**/queue/logical_block_size*
- */sys/block/**disk**/queue/minimum_io_size*
- */sys/block/**disk**/queue/optimal_io_size*

*カーネルは、I/O パラメーター情報を提供しない「レガシー」デバイスの **sysfs** 属性をエクスポートします。以下に例を示します。*

例23.1 sysfs インターフェース

```
alignment_offset:    0
physical_block_size: 512
logical_block_size:  512
minimum_io_size:     512
optimal_io_size:     0
```

ブロックデバイス ioctls

- ***BLKALIGNOFF*: *alignment_offset***
- ***BLKPBSZGET*: *physical_block_size***

- ***BLKSSZGET*: *logical_block_size***
- ***BLKIOMIN*: *minimum_io_size***
- ***BLKIOOPT*: *optimal_io_size***

23.3. STANDARDS（標準）

本セクションでは、ATA デバイスおよびSCSI デバイスが使用する I/O 標準を説明します。

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 デバイスは、**physical_block_size**、**logical_block_size**、および **alignment_offset** の I/O パラメーターだけを報告します。追加の I/O ヒントは、ATA コマンドセットの範囲外にあります。*

SCSI

*Red Hat Enterprise Linux 6 における I/O パラメーターのサポートには、少なくとも SCSI Primary Commands (SPC-3) **プロトコルのバージョン 3** が必要です。カーネルは、SPC-3 に準拠しを要求するデバイスに対して、拡張された inquiry （**BLOCK LIMITS VPD** ページへのアクセスを取得する）および **READ CAPACITY(16)** コマンドのみを送信します。*

***READ CAPACITY(16)** コマンドは、ブロックサイズと調整オフセットを提供します。*

- ***LOGICAL BLOCK LENGTH IN BYTES** **/sys/block/disk/queue/physical_block_size**を派生するのに使用されます。*
- ***LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT** **/sys/block/disk/queue/logical_block_size**を派生するために使用されます。*
- ***LOWEST ALIGNED LOGICAL BLOCK ADDRESS** 派生するのに使用されます。*
  - ***/sys/block/disk/alignment_offset***
  - ***/sys/block/disk/partition/alignment_offset***

***BLOCK LIMITS VPD** ページ(**0xb0**)は、I/O ヒントを提供します。また、**OPTIMAL TRANSFER LENGTH GRANULARITY** および **OPTIMAL TRANSFER LENGTH** を使用して以下の派生します。*

- ***/sys/block/disk/queue/minimum_io_size***
- ***/sys/block/disk/queue/optimal_io_size***

***sg3_utils** パッケージは、**sg_inq** ユーティリティーを提供します。これは、**BLOCK LIMITS VPD** ページにアクセスできます。これを行うには、以下を実行します。*

```
# sg_inq -p 0xb0 disk
```

23.4. スタッキング I/O パラメーター

Linux I/O スタックのすべてのレイヤーがエンジンで、さまざまな I/O パラメーターをスタックに伝播します。レイヤーが属性を消費するか、多くのデバイスを集約すると、レイヤー層デバイスまたはツールが変換されるストレージの正確なビューを持つように、レイヤーは適切な I/O パラメーターを公開す

る必要があります。実用的な例は以下のとおりです。

- *I/O スタックにおける 1 つのレイヤーのみがゼロでない **alignment_offset** に対して調整する必要があります。レイヤーが調整されると、そのレイヤーがゼロの **alignment_offset** を持つデバイスをエクスポートします。*
- *LVM で作成したストライプ化デバイスマッパー(DM)デバイスは、ストライプ数（ディスクの数）**とユーザーが指定するチャンクサイズとの対比で、minimum _io _size と optimal** _io_size をエクスポートする必要があります。*

Red Hat Enterprise Linux 6 では、デバイスマッパーと Software Raid(MD)デバイスドライバーを使用して、デバイスと、異なる I/O パラメーターとデバイスを統合することができます。カーネルのブロック層は、個々のデバイスの I/O パラメーターを合理的に組み合わせます。カーネルは異種デバイスを組み合わせることはできませんが、実行に関連するリスクに注意してください。

*たとえば、512 バイトのデバイスと 4K デバイスが 1 つの論理 DM デバイスに統合され、これにより **logical_block_size** が 4K になります。このようなハイブリッドデバイスでファイルシステムを重ねるのは、4K がアトミックに書き込まれていると仮定しますが、実際には 512 バイトデバイスに発行すると 8 つの論理ブロックアドレスにまたがることになります。より高いレベルの DM デバイスに 4K **logical_block_size** を使用すると、システムクラッシュがある場合に、512 バイトデバイスへの部分的な書き込みが可能になります。*

複数デバイスの I/O パラメーターを組み合わせると競合が生じると、ブロック層は、デバイスが部分的な書き込みの影響を受けたり、もしくはアグリーメントされていることに気付くという警告が表示される場合があります。

23.5. 論理ボリュームマネージャー

*LVM は、カーネルの DM デバイスを管理するのに使用するユーザー空間ツールを提供します。LVM は、（指定の DM デバイスが使用するデータ領域の開始）を移動して、LVM が管理するデバイスに関連付けられたゼロ以外の **alignment_offset** に対応します。これは、論理ボリュームが正しく調整されることを意味します (**alignment_offset=0**)。*

*デフォルトでは、LVM は **alignment_offset に合わせて調整しますが**、この動作を無効にするには、**/etc/lvm/lvm.conf** の **0** に **data_alignment_offset_detection** を設定します。これを無効にすることは推奨していません。*

*LVM は、デバイスの I/O ヒントも検出します。デバイスのデータ領域の開始は、sysfs に公開される **minimum_io_size** または **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 (例: **0**)、LVM は **minimum_io_size** を使用します。*

*デフォルトでは、LVM は自動的にこれらの I/O ヒントを決定しますが、この動作を無効にするには、/**etc/lvm/lvm.conf** の **data_alignment_detection** を **0** に設定します。これを無効にすることは推奨していません。*

23.6. パーティションおよびファイルシステムツール

このセクションでは、さまざまなパーティションおよびファイルシステム管理ツールがデバイスの I/O パラメーターと対話する方法を説明します。

util-linux-ng の libblkid および fdisk

util-linux-ng** パッケージで提供される **libblkid** ライブラリーには、デバイスの I/O パラメーターにアクセスするためのプログラムによる API が含まれます。**libblkid** は、特に Direct I/O を使用するアプリケーションを使用して、I/O リクエストを適切にサイズできるようにします。**util-linux-ng** の **fdisk

ユーティリティーは、**libblkid** を使用して、すべてのパーティションの配置を最適なデバイスの I/O パラメーターを決定します。**fdisk** ユーティリティーは、すべてのパーティションを 1MB 境界に合わせます。

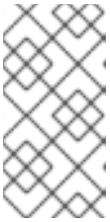
parted および libparted

parted の **libparted** ライブラリーは、**libblkid** の I/O パラメーター API も使用します。Red Hat Enterprise Linux 6 インストーラー(**Anaconda**)は **libparted** を使用します。これは、インストーラーまたは **parted** によって作成されたすべてのパーティションが適切に調整されることを意味します。I/O パラメーターを提供していないデバイスで作成されたすべてのパーティションに対して、デフォルトの調整は 1MB になります。

ヒューリスティック **parted** が使用する機能は以下のとおりです。

- 常に、最初のプライマリーパーティションの開始のオフセットとして、報告された **alignment_offset** を使用します。
- **optimal_io_size** が定義されている場合（例： **0**ではない）、すべてのパーティションを **optimal_io_size** 境界に合わせます。
- **optimal_io_size** が未定義である場合（例：0）、recon **se _offset** は **0** で、**min_io_size** は 2 の累乗の場合は、1MB のデフォルトの調整を使用します。

  これは、I/O ヒントを提供するように見えない「レガシー」デバイスの catch-all です。そのため、デフォルトではすべてのパーティションが 1MB の境界に調整されます。



注記

Red Hat Enterprise Linux 6 は、I/O ヒントを提供していないデバイスと、reconse **_offset=0** と **optimal_io_size=0** を持つデバイスを区別することができません。このようなデバイスは、SAS 4K デバイスに 1 つある可能性があります。したがって、ディスクの起動時に最も悪い 1 MB の領域が失われます。

ファイルシステムツール

異なる **mkfs.filesystem** ユーティリティーが強化され、デバイスの I/O パラメーターが使用されるようになりました。このユーティリティーを使用すると、ファイルシステムをフォーマットして、基になるストレージデバイスの **logical_block_size** よりも小さいブロックサイズを使用することができません。

mkfs.gfs2 を除き、他のすべての **mkfs.ファイルシステム** ユーティリティーは、I/O ヒントを使用して、ディスク上のデータ構造と、下層のストレージデバイスの **minimum_io_size** と **optimal_io_size** に関連するデータ領域のレイアウトも使用します。これにより、ファイルシステムをさまざまな RAID（ストライプ）レイアウトに対して最適にフォーマットできます。

第24章 リモートディスクレスシステムの設定

ネットワークブートサービス（**system-config-netboot**で提供）は、Red Hat Enterprise Linux 6 で利用できなくなりました。本リリースでは、**system-config-netboot** を使用せずにディスクレスシステムをデプロイできるようになりました。

PXE 経由でブートする基本的なリモートディスクレスシステムを設定するには、以下のパッケージが必要です。

- ***tftp-server***
- ***xinetd***
- ***dhcp***
- ***syslinux***
- ***dracut-network***

リモートディスクレスシステムを起動するには、**tftp** サービス (**tftp-server** が提供) および DHCP サービス (**dhcp** が提供) の両方が必要です。**tftp** サービスは、PXE ローダーを介してネットワーク経由でカーネルイメージと **initrd** を取得するために使用されます。

次のセクションでは、ネットワーク環境にリモートディスクレスシステムをデプロイするのに必要な手順の概要を説明します。

24.1. ディスクレスクライアントの TFTP サービスの設定

tftp サービスはデフォルトで無効になっています。このネットワーク経由で PXE ブートを許可するには、**/etc/xinetd.d/tftp** の **Disabled** オプションを **no** に設定します。**tftp** を設定するには、以下の手順を実行します。

手順24.1 tftp を設定するには、以下を行います。

1. **tftp** root ディレクトリー (**chroot**) は **/var/lib/tftpboot** にあります。**/usr/share/syslinux/pxelinux.0** を、以下のように **/var/lib/tftpboot/** にコピーします。

   cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/

2. **tftp** の root ディレクトリーに **pxelinux.cfg** ディレクトリーを作成します。

   mkdir -p /var/lib/tftpboot/pxelinux.cfg/

tftp トラフィックを許可するようにファイアウォールルールを正しく設定する必要もあります。tftp は TCP ラッパー **をサポートするため、/etc/hosts.allow** を使用して **tftp** へのホストアクセスを設定できます。TCP ラッパーおよび **/etc/hosts.allow** 設定ファイルの設定に関する詳細は、『Red Hat Enterprise Linux 6 『セキュリティーガイド』を参照してください』。また、**man hosts_access** は、**/etc/hosts.allow** に関する情報も提供します。

ディスクレスクライアントの **tftp** を設定した後に、DHCP、NFS、エクスポートしたファイルシステムを適宜設定します。その手順は、「ディスクレスクライアントの DHCP の設定」 および 「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」 を参照してください。

24.2. ディスクレスクライアントの DHCP の設定

tftp サーバーを設定したら、同じホストマシンに DHCP サービスを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 6 『デプロイメントガイド』を参照してください』。また、DHCP サーバー上で PXE ブートを有効にする必要があります。これには、以下の設定を **/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 を、**tftp** サービスおよびDHCP サービスが置かれているホストマシンの IP アドレスに置き換えます。**tftp** およびDHCP が設定されているため、残りはすべてNFS およびエクスポートしたファイルシステムを設定します。手順については、「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」 を参照してください。

24.3. 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のルートディレクトリー(ネットワーク上のディスクレスクライアントが使用) は、NFS 経由で共有されます。**/etc/exports** に root ディレクトリーを追加して、root 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。その手順は、「**/etc/exports** 設定ファイル」 を参照してください。

ディスクレスクライアントに完全に対応できるようにするために、root ディレクトリーには Red Hat Enterprise Linux の完全なインストールが含まれている必要があります。これは、以下のように **rsync** を使用して実行中のシステムと同期できます。

```
# rsync -a -e ssh --exclude='/proc/*' --exclude='/sys/*' hostname.com:/ /exported/root/directory
```

hostname.com を、**rsync** を介して同期する稼働中のシステムのホスト名に置き換えます。**/exported/root/directory** を、エクスポートしたファイルシステムへのパスに置き換えます。

または 、**--installroot** オプションを指定して **yum** を使用して、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。以下に例を示します。

```
yum groupinstall Base --installroot=/exported/root/directory
```

エクスポートするファイルシステムは、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追加で設定する必要があります。空き領域が足りない場合は、次の手順を実行します。

手順24.2 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、以下の設定を含む（最低でも）設定します。

```
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. ディスクレスクライアントが使用するカーネル (**vmlinuz-kernel-version**) を選択し、**tftp** の boot ディレクトリーにコピーします。

```
# cp /boot/vmlinuz-kernel-version /var/lib/tftpboot/
```

3. ネットワークサポートで **initrd** （つまり **initramfs-kernel-version.img**）を作成します。

```
# dracut initramfs-kernel-version.img kernel-version
```

作成された **initramfs-kernel-version.img** を **tftp** のブートディレクトリーにコピーします。

4. **/var/lib/tftpboot** 内の **initrd** およびカーネルを使用するように、デフォルトのブート設定を編集します。この設定は、エクスポートしたファイルシステム (**/exported/root/directory**) を読み書きとしてマウントするよう、ディスクレスクライアントの root に指示を出します。これを行うには、**/var/lib/tftpboot/pxelinux.cfg/default** を以下のように設定します。

```
default rhel6

label rhel6
  kernel vmlinuz-kernel-version
  append initrd=initramfs-kernel-version.img root=nfs:server-ip:/exported/root/directory rw
```

server-ip を、**tftp** サービスおよび DHCP サービスが置かれているホストマシンの IP アドレスに置き換えます。

これで、NFS 共有がディスクレスクライアントにエクスポートできるようになりました。これらのクライアントは、PXE 経由でネットワーク経由で起動できます。

第25章 デバイスマッパーマルチパスおよび仮想ストレージ

Red Hat Enterprise Linux 6 は DM Multipath および 仮想ストレージ もサポートします。どちらの機能も、Red Hat の 『DM Multipath および仮想化』 『管理ガイド』 で説明されています。

25.1. 仮想ストレージ

Red Hat Enterprise Linux 6 は、仮想ストレージの以下のファイルシステム/オンラインストレージ方法をサポートします。

- *ファイバーチャンネル*
- *iSCSI*
- *NFS*
- *GFS2*

*Red Hat Enterprise Linux 6 の仮想化は、**libvirt** を使用して仮想インスタンスを管理します。**libvirt** ユーティリティーは、ストレージプール の概念を使用して、仮想化ゲストのストレージを管理します。ストレージプールは、小規模なボリュームに分割したり、ゲストに直接割り当てることができるストレージです。ストレージプールのボリュームは、仮想化ゲストに割り当てることができます。利用可能なストレージプールには、以下のカテゴリーが2 つあります。*

ローカルストレージのプール

ローカルストレージは、ホストに直接接続ストレージデバイス、ファイル、またはディレクトリーに対応します。ローカルストレージには、ローカルディレクトリー、直接割り当てられたディスク、および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は、標準プロトコルを使用してネットワーク上で共有されるストレージデバイスに対応します。ファイバーチャネル、iSCSI、NFS、GFS2、およびSCSI RDMA プロトコルを使用して共有ストレージデバイスが含まれ、ゲスト仮想化ゲストをホスト間で移行するための要件となります。

重要

環境内の仮想ストレージインスタンスのデプロイメントおよび設定に関する包括的な情報は、Red Hat が提供する 『仮想化 『ガイドの』 「仮想化ストレージ」セクションを参照してください』。

25.2. DM MULTIPATH

Device Mapper Multipath(DM-Multipath)は、 サーバーノードとストレージアレイ間の複数のI/O パスを1 つのデバイスに設定することができる機能です。これらのI/O パスは、個別のケーブル、スイッチ、コントローラーを含むことができる物理的なSAN 接続です。複数のI/O パスを集め、集めたパスで構成される新しいデバイスを1 つ作成するのがマルチパス機能です。

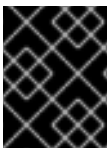
DM Multipath は、主に以下の理由に使用されます。

冗長性

DM-Multipath は、アクティブ/パッシブ構成でフェイルオーバーを提供できます。アクティブ/パッシブ構成では、パスは、I/O には常に半分しか使用されません。I/O パスの要素 (ケーブル、スイッチ、またはコントローラー) に障害が発生すると、DM-Multipath は代替パスに切り替えます。

パフォーマンスの向上

DM-Multipath を active/active モードに設定すると、I/O はラウンドロビン式でパス全体に分散されます。一部の設定では、DM Multipath は I/O パスの負荷を検出し、負荷を動的に再分散できます。



重要

お使いの環境での DM Multipath のデプロイメントおよび設定に関する包括的な情報は、Red Hat 『が提供する DM Multipath の使用』 ガイドを参照してください。

パート III. オンラインストレージ

オンラインストレージのセクションでは、iSCSI および FCoE ストレージ接続を設定し、管理する方法を説明します。

オンラインストレージの再設定は注意して行う必要があります。プロセス中のシステム障害や中断により、予期しない結果が生じる可能性があります。Red Hat は、変更操作時に可能な最大エクステントへのシステムの負荷を軽減することを推奨します。これにより、I/O エラー、メモリー不足のエラー、または設定変更の midst で発生した同様のエラーが削減されます。以下のセクションでは、これに関するより具体的なガイドラインを説明します。

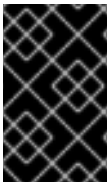
また、Red Hat は、オンラインストレージの再設定前にすべてのデータのバックアップを作成することを推奨します。

第26章 ファイバーチャンネル

このセクションでは、これらのドライバーのファイバーチャネル API、ネイティブ Red Hat Enterprise Linux 6 ファイバーチャネルドライバー、およびファイバーチャネル機能について説明します。

26.1. ファイバーチャネル API

*ユーザー空間 API の提供に使用されるファイルが含まれる **/sys/class/** ディレクトリーのリストは次のとおりです。各項目では、ホスト番号が **H**、バス番号が **B** で指定され、ターゲットは **T**、論理ユニット番号 (LUN) は **L** で、リモートのポート番号は **R** になります。*



重要な影響

マルチパスソフトウェアを使用している場合は、このセクションに記載される値のいずれかを変更する前に、ハードウェアベンダーにお問い合わせになることが推奨されます。

トランスポート - /sys/class/fc_transport/targetH:B:T/

- ***port_id** - 24 ビットポート ID/ アドレス*
- ***node_name** - 64 ビットノード名*
- ***port_name** - 64 ビットポート名*

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** - scsi デバイスがシステムから削除されるタイミングを制御します。**dev_loss_tmo** がトリガーされると、scsi デバイスが削除されます。*

  ***multipath.conf** では、dev **_loss_tmo** を **infinity** に設定できます。これにより、その値を 2,147,483,647 秒または 68 年間 に設定し、dev **_loss_tmo** の最大値を設定できます。*

  *Red Hat Enterprise Linux 6 では、**fast_io_fail_tmo** はデフォルトで設定されないため、**dev_loss_tmo** の値は 600 秒に制限されます。*

- ***fast_io_fail_tmo** - リンクに「bad」のマークを付けるまでの待機秒数を指定します。リンクに「bad」のマークが付けられると、対応するパス上の既存の実行中の I/O または新しい I/O が失敗します。*

  *I/O がブロックされたキューに存在する場合は、**dev_loss_tmo** の期限が切れ、キューのブロックが解除されるまでエラーを起こしません。*

  ***fast_io_fail_tmo** を **off** 以外の値に設定すると、**dev_loss_tmo** は取得されません。**fast_io_fail_tmo** を **off** に設定すると、システムからデバイスが削除されるまで I/O は失敗します。**fast_io_fail_tmo** を数字に設定すると、**fast_io_fail_tmo** がトリガーされると、I/O は即座に失敗します。*

ホスト - /sys/class/fc_host/hostH/

- ***port_id***
- ***issue_lip*** - リモートポートを再検出するようにドライバーに指示します。

26.2. ネイティブファイバーチャネルドライバーおよび機能

Red Hat Enterprise Linux 6 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***
- ***mptfc***
- ***bfa***

表26.1「ファイバーチャネル API 機能」 では、各ネイティブ Red Hat Enterprise Linux 6 ドライバーの異なる fibre-channel API 機能を説明します。x は機能のサポートを表します。

表26.1 ファイバーチャネル API 機能

	lpfc	qla2xxx	zfcp	mptfc	bfa
Transport **port_id**	X	X	X	X	X
Transport **node_name**	X	X	X	X	X
Transport **port_name**	X	X	X	X	X
リモートポート **dev_loss_tmo**	X	X	X	X	X
Remote Port **fast_io_fail_tmo**	X	X[a]	X[b]		X
Host **port_id**	X	X	X	X	X
Host **issue_lip**	X	X			X

[a] Red Hat Enterprise Linux 5.4 でサポートされています。

[b] Red Hat Enterprise Linux 6.0 でサポート

第27章 iSCSI ターゲットおよびイニシエーターの設定

注記

多数の iSCSI LUN を使用して hal デーモンを使用する場合は、ブートが失敗しないように **--child-timeout** オプションを使用する必要があります。**--child-timeout** オプションは、すべてのディスクプローブの実行を待つ秒数を設定します。たとえば、hal デーモンが 10 分と 30 秒待機するように強制するには、このオプションは **--child-timeout=630** を読み取ります。デフォルトの時間は 250 秒です。これは、hal デーモンの起動に時間がかかることを意味しますが、すべてのディスクデバイスが認識され、起動の失敗を防ぐために十分な時間を確保できます。

この回避策の理由は、hal デーモンが作成する際に 2003 では目立つために、これは dozen iSCSI ディスク以上のものを持つためです。このため、hal デーモンは Red Hat Enterprise Linux 7 で削除され、udisks に置き換えられました。

詳細は、以下の Red Hat ナレッジベースソリューション「hal daemon fails to start on system with a large many disks in RHEL 5 and RHEL 6」を参照してください。

27.1. iSCSI ターゲットの作成

iSCSI ターゲットは、ネットワーク内の専用の物理デバイスか、ネットワークストレージサーバー上にある iSCSI ソフトウェア構成の論理デバイスになります。ターゲットは、SCSI バス通信内のエンドポイントです。イニシエーターによってアクセスされるターゲット上のストレージは、LUN で定義されます。

手順27.1 iSCSI ターゲットの作成

1. **scsi-target-utils** をインストールします。

```
~]# yum install scsi-target-utils
```

2. ファイアウォールでポート 3260 を開きます。

```
~]# iptables -I INPUT -p tcp -m tcp --dport 3260 -j ACCEPT
~]# service iptables save
```

3. ターゲットサービスを開始して有効にします。

```
~]# service tgtd start
~]# chkconfig tgtd on
```

4. LUN にストレージを割り当てます。この例では、ブロックストレージ用に新しいパーティションが作成されます。

```
~]# fdisk /dev/vdb
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.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x43eb8efd.
```

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): *Enter*
Using default response p
Partition number (1-4, default 1): *Enter*
First sector (2048-2097151, default 2048): *Enter*
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): +250M
Partition 1 of type Linux and of size 250 MiB is set

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

5. ***/etc/tgt/targets.conf*** *ファイルを編集し、ターゲットを作成します。*

```
~]# cat /etc/tgt/targets.conf
...
default-driver iscsi
<target iqn.2015-06.com.example.test:target1>
 backing-store /dev/vdb1
 initiator-address 10.10.1.1
</target>
```

*上記の例では、1 つのバッキングストアと許可されたイニシエーター1 つを含む簡単なターゲットが作成されます。**YYYY-MM.reverse.domain.name : OptionalIdentifierの形式で、iqn の名前を指定して** 名前を指定する必要があります。バッキングストアは、ストレージが置かれているデバイスです。initiator-address は、ストレージにアクセスするためのイニシエーターの IP アドレスです。*

6. *ターゲットサービスを再起動します。*

```
~]# service tgtd restart
Stopping SCSI target daemon:     [ OK ]
Starting SCSI target daemon:     [ OK ]
```

7. *設定を確認します。*

```
~]# tgt-admin --show
Target 1: iqn.2015.06.com.example.test: server
 System information:
  Driver: iscsi
  State: ready
 I_T nexus information:
 LUN information:
  Lun: 0
   Type: controller
   SCSI ID: IET    00010000
   SCSI SN: beaf10
   Size: 0 MB, Block size: 1
```

```
    Online: Yes
    Removable media: No
    Prevent removal: No
    Readonly: No
    Backing store type: null
    Backing store path: None
    Backing store flags:
  LUN: 1
    Type: disk
    SCSI ID: IET     00010001
    SCSI SN: beaf11
    Size: 2147 MB, Block size: 512
    Online: Yes
    Removable media: No
    Prevent removal: No
    Readonly: No
    Backing store type: rdwr
    Backing store path: /dev/vdb1
    Backing store flags:
  Account information:
  ACL information:
    10.10.1.1
```

27.2. iSCSI イニシエーターの作成

iSCSI イニシエーターは、ターゲットまたはサーバーのストレージにアクセスしたいクライアントです。このプロセスには、ターゲットの IP アドレスが認識される必要があります。

手順27.2 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. *ターゲットを検出します。ターゲットの IP アドレスを使用します。以下に使用されている IP アドレスは、例としてのみ提供されます。*

   ```
   ~]# iscsiadm -m discovery -t sendtargets -p 192.168.1.1
   Starting iscsid:     [ OK ]
   192.168.1.1:3260,1 iqn.2015-06.com.example.test:target1
   ```

   上記は、ターゲットの IP アドレスと IQN アドレスを示しています。これは、今後のステップに必要な IQN アドレスです。

3. *ターゲットに接続します。*

   ```
   ~]# iscsiadm -m node -T iqn.2015-06.com.example:target1 --login
   Logging in to [iface: default, target: iqn.2015-06.com.example:target1, portal:
   192.168.1.1,3260] (multiple)
   Login in to [iface: default, target: iqn.2015-06.com.example:target1, portal: 192.168.1.1,3260]
   successful.
   ```

4. *iSCSI ディスク名を検索します。*

   -

```
~]# grep "Attached SCSI" /var/log/messages
Jun 19 01:30:26 test kernel: sd 7:0:0:1 [sdb] Attached SCSI disk
```

5. *そのディスクにファイルシステムを作成します。*

```
~]# mkfs.ext4 /dev/sdb
```

6. *ファイルシステムをマウントします。*

```
~]# mkdir /mnt/iscsiTest
~]# mount /dev/sdb /mnt/iscsiTest
```

7. ***/etc/fstab** ファイルを編集して、再起動後も持続します。*

```
~]# blkid /dev/sdb
/dev/sdb: UUID="766a3bf4-beeb-4157-8a9a-9007be1b9e78" TYPE="ext4"
~]# vim /etc/fstab
UUID=766a3bf4-beeb-4157-8a9a-9007be1b9e78 /mnt/iscsiTest ext4 _netdev 0 0
```

第28章 永続的な命名

オペレーティングシステムは、アクセスに使用するパスを参照してストレージデバイスへの I/O を発行します。SCSI デバイスの場合、パスは以下で構成されます。

- ホストバスアダプター(HBA)の PCI 識別子
- HBA 上のチャンネル番号
- リモート SCSI ターゲットアドレス
- 論理ユニット番号(LUN)

このパスベースのアドレスは永続的ではありません。（このマニュアルの説明のように、オンライン再設定、またはシステムのシャットダウン、再設定、および再起動時など）に、システムが再設定されるたびに変更する可能性があります。物理再設定が行われていない場合や、システムの起動時に検出プロセスのタイミングをタイミングで差異したり、バスの再スキャンを行うときなど、パス識別子が変更する可能性があります。

オペレーティングシステムは、ストレージデバイスへのこれらのアクセスパスを表す永続的な名前を複数提供します。もう 1 つは **/dev/sd** 名です。もう 1 つは **major:minor** 番号です。3 つ目は、**/dev/disk/by-path/** ディレクトリーにあるシンボリックリンクです。このシンボリックリンクは、パス識別子から現在の **/dev/sd** 名にマッピングします。たとえば、ファイバーチャネルデバイスの場合、PCI 情報 **およびホスト:BusTarget:LUN** info は以下のようになります。

```
pci-0000:02:0e.0-scsi-0:0:0:0 -> ../../sda
```

iSCSI デバイスの場合は、ターゲット名とポータル情報から **sd** 名までパス / 名前でマッピングされます。

通常、これらのパスベースの名前を使用するアプリケーション**には適していません。**これは、これらのパスを参照するストレージデバイスが変更され、これにより誤ったデータがデバイスに書き込まれる可能性があるためです。パスベースの名前はマルチパスデバイスには適していません。これは、パスベースの名前が別のストレージデバイスで間違いとなる可能性があるため、データのアクセスと意図しない変更が生じる可能性があるためです。

また、パスベースの名前はシステム固有の名前です。これにより、クラスターなど、複数のシステムでデバイスにアクセスすると、意図しないデータが変更する可能性があります。

このような理由から、デバイスを識別する永続的なシステムに依存しない方法が複数開発されました。以下のセクションでは、これらについて詳しく説明します。

28.1. WWID

World Wide Identifier (WWID)は、確実に識別するデバイスで使用することができます。これは、SCSI 標準がすべての SCSI デバイスに必要な、永続的なシステムに依存しない ID です。各ストレージデバイスの WWID 識別子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。

この識別子を取得するには、SCSI Inquiry を実行して Device Identification Vital Product Data （ページ **0x83**）または Unit Serial Number（ページ **0x80**）を取得します。これらの WWID から現在の **/dev/sd** 名へのマッピングは、/ **dev/disk/by-id/** ディレクトリーにあるシンボリックリンクで確認できます。

例28.1 WWID

> たとえば、ページ **0x83** の識別子を持つデバイスには、以下が含まれます。
>
> > scsi-3600508b400105e210000900000490000 -> ../../sda
>
> または、ページ **0x80** の識別子を持つデバイスには、以下が含まれます。
>
> > 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/** 名を使用できます。

システムからデバイスへのパスが複数あると、device **-mapper-multipath はこれを** 検出するのに WWID を使用します。**device-mapper-multipath** は、**/dev/mapper/3600508b400105df70000e00000ac0000** など、**/dev/mapper/wwid** に単一の「pseudo-device」を表示します。

コマンド **multipath -l** は、非永続的な識別子へのマッピングです。**ホスト:Channel:ターゲット:LUN**、**/dev/sd** 名、および **major:minor** 番号が表示されます。

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=0][active]
 \_ 5:0:1:1 sdc 8:32  [active][undef]
 \_ 6:0:1:1 sdg 8:96  [active][undef]
\_ round-robin 0 [prio=0][enabled]
 \_ 5:0:0:1 sdb 8:16  [active][undef]
 \_ 6:0:0:1 sdf 8:80  [active][undef]
```

device **-mapper-multipath** は、各 WWID ベースのデバイス名の対応する **/dev/sd** 名へ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、パスが変更しても持続し、他のシステムからデバイスにアクセスする際に一貫性を保持します。

user_friendly_names 機能(**device-mapper-multipath**)機能を使用すると、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** ファイルに保持されています。これらの **mpathn** 名は、そのファイルが維持されている限り永続的です。

重要

user_friendly_names を使用する場合は、クラスター内で一貫した名前を取得するために追加の手順が必要です。『『Using DM Multipath Configuration and Administration』の「Consistent Multipath』 Device Names in a Cluster」セクションを参照してください。

システムにより提供される永続的な名前のほかに、**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。これに関する詳細情報は、を参照してください http://kbase.redhat.com/faq/docs/DOC-7319。

28.2. UUID およびその他の永続識別子

ストレージデバイスにファイルシステムが含まれている場合は、そのファイルシステムは以下のいずれかまたは両方を提供できます。

- *UUID(Universally Unique Identifier)*
- *ファイルシステムのラベル*

*これらの識別子は永続的であり、特定のアプリケーションによってデバイスに書き込まれるメタデータに基づいています。**/dev/disk/by-label/** のオペレーティングシステムが管理するシンボリックリンクを使用してデバイスにアクセスすることもできます（例：**Boot -> ../../sda1**）および **/dev/disk/by-uuid/**（例：**f8bf09e3-4c16-4d91-bd5e-6f62da165c08 -> ../../sda1**）ディレクトリー。*

***md** および LVM 書き込みメタデータは、ストレージデバイス上でメタデータを読み込み、デバイスのスキャン時にデータを読み取ります。それぞれのケースでは、メタデータには UUID が含まれ、デバイスへのアクセスに使用するパス（またはシステム）に関係なくデバイスを特定することができます。その結果、メタデータが変更されない限り、これらの機能で提示するデバイス名は永続化されます。*

第29章 ストレージデバイスの削除

ストレージデバイスへのアクセスを削除する場合は、最初にデバイスからのデータのバックアップを作成することが推奨されます。その後、I/O をフラッシュして、デバイスへのオペレーティングシステム参照をすべて削除します（以下の説明にあるように）。デバイスがマルチパスを使用する場合には、マルチパス「pseudo device」（「WWID」）とデバイスへのパスを表す各識別子に対してこれを実施します。マルチパスデバイスへのパスを削除するだけで、その他のパスが残る場合は、31章ストレージデバイスまたはパスの追加 の説明に従って手順が簡単です。

I/O フラッシュが負荷に追加されるため、システムがメモリー不足の状態になるとストレージデバイスの削除は推奨されません。メモリープレッシャ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。以下の場合はデバイスの削除は推奨されません。

- 空きメモリーの合計は、100 あたり 10 個以上のサンプルで合計メモリーの 5% 未満です（コマンド **free** を使用すると合計メモリーの合計が表示されます）。
- swapping はアクティブです（**vmstat** 出力の **si** および **so** 列以外）。

デバイスへのアクセスを削除する一般的な手順は次のとおりです。

手順29.1 デバイスの完全削除

1. 必要に応じて、デバイスおよびバックアップデバイスデータのすべてのユーザーを閉じます。
2. **umount** を使用して、デバイスをマウントしたファイルシステムのマウントを解除します。
3. **md** ボリュームおよびLVM ボリュームを使用して、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は、**pvmove** コマンドを使用してデバイスからデータを移動する必要がある場合は、**vgreduce** コマンドを使用して物理ボリュームを削除（オプションで **pvremove**）、ディスクからLVM メタデータを削除する必要がある場合があります。
4. デバイスがマルチパス機能を使用する場合は、**multipath -l** を実行して、デバイスへのパスをすべて書き留めておきます。その後、**multipath -f デバイスを使用してマルチパスデバイスを削除します。**
5. **blockdev --flushbufs デバイス** を実行し、未処理の I/O をデバイスのすべてのパスにフラッシュします。これは特に、**umount** または **vgreduce** 操作がない raw デバイスに特に重要になります。これにより、I/O フラッシュが発生します。
6. システムのアプリケーション、スクリプト、ユーティリティーの **/dev/sd**、**/dev /disk/by-path**、**major:minor** 番号などのデバイスのパスベースの名前への参照を削除します。これは、今後追加される異なるデバイスが現在のデバイスには間違いにならないようにするために重要です。
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、コマンド **echo 1 > /sys/block/device-name/device/delete** コマンドを使用します。たとえば、device **-name** がde になります。

   この操作に関するもう **1 つのバリエーションは、1 > /sys/class/scsi_device/h:c:t:l/device/delete です**。h は HBA 番号で、c **は** HBA のチャンネル、c **は** SCSI ターゲット ID、**l** が LUN です。

注記

こ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** は非推奨となりました。

lsscsi、 ***scsi_id*、*multipath -l*、および *ls -l /dev/disk/by-** などのさまざまなコマンドから、デバイス名、HBA 番号、HBA チャンネル、SCSI ターゲット ID および LUN を確認できます。***

手順29.1「デバイスの完全削除」 を実行した後は、稼働中のシステムからデバイスを物理的に削除できます。他のデバイスへの I/O を停止する必要はありません。

デバイスの物理削除などのその他の手順の後に、（34章ストレージ Interconnect のスキャン で説明されているように）SCSI バスの再スキャンを行い、オペレーティングシステムの状態が変更を反映させるように更新することは推奨されません。これにより、I/O タイムアウトが原因で遅延が発生し、デバイスが予想外に削除される可能性があります。相互接続の再スキャンを実行する必要がある場合は、34章ストレージ Interconnect のスキャン の説明に従って I/O が一時停止している間に実行する必要があります。

第30章 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へのパスを削除する場合（デバイスの他のパスに影響を与えずに）、一般的な手順は以下のようになります。

手順30.1 ストレージデバイスへのパスの削除

1. システムのアプリケーション、スクリプト、ユーティリティーの **/dev/sd または /dev/ disk/by-path** または **major:minor** 番号などのデバイスのパスベースの名前への参照を削除します。これは、今後追加される異なるデバイスが現在のデバイスには間違いにならないようにするために重要です。

2. **echo offline > /sys/block/sda/device/state** を使用して、パスをオフラインにします。

   これにより、このパスのデバイスに送信された後続の I/O が即座に失敗します。device - **mapper-multipath** は、デバイスへの残りのパスを引き続き使用します。

3. SCSI サブシステムからパスを削除します。そのためには、コマンド **echo 1 > /sys/block/ device-name /device/delete** コマンドを使用します（ **例**: 手順29.1「デバイスの完全削除」）。

手順30.1「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できます。この処理が行われている間は、I/O を停止する必要はありません。**device-mapper-multipath** は、設定されたパスグループ化やフェイルオーバーポリシーに従って、I/O を残りのパスに再ルーティングします。

ケーブルの物理削除など、その他の手順に SCSI バスの再スキャンを行い、オペレーティングシステムの状態が変更を反映させることは推奨していません。これにより、I/O タイムアウトが原因で遅延が発生し、デバイスが予想外に削除される可能性があります。相互接続の再スキャンを実行する必要がある場合は、34章ストレージ Interconnect のスキャン の説明に従って I/O が一時停止している間に実行する必要があります。

第31章 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場合、パスベースのデバイス名（**/dev/sd** 名、**major:minor** 番号、および **/dev/disk/by-path** 名）は、システムが新しいデバイスに割り当てると、新しいデバイスに割り当てると、削除以降以降のデバイスにより使用されていた可能性があることに注意してください。そのため、パスベースのデバイス名への古い参照がすべて削除されていることを確認します。そうしないと、古いデバイスの新しいデバイスが間違っている可能性があります。

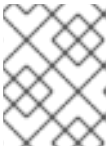
手順31.1 ストレージデバイスまたはパスの追加

1. ストレージデバイスまたはパスを追加する最初の手順は、新しいストレージデバイスまたは既存のデバイスへ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ます。これを実行する際には、ホストに表示される新規ストレージの LUN 値を書き留めておいてください。ストレージサーバーがファイバーチャネルである場合、ストレージサーバーの World Wide Node Name (WWNN) にも認識し、ストレージサーバーにあるすべてのポートに WWNN が 1 つあるかどうかを確認します。そうでない場合は、新しい LUN へのアクセスに使用される各ポートの World Wide Port Name (WWPN) をメモします。

2. 次に、オペレーティングシステムが新しい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ます。使用が推奨されるコマンドは以下のとおりです。

```
$ echo "c t l" > /sys/class/scsi_host/hosth/scan
```

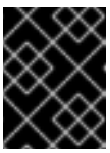
上記のコマンドでは、h **が** HBA 番号、c **は** HBA のチャンネル、t **は** SCSI ターゲット ID で、**l** が LUN になります。



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** は非推奨となりました。

a. 一部のファイバーチャネルハードウェアでは、LIP(Loop Initialization Protocol)操作を実行するまで、RAID アレイで新たに作成された LUN がオペレーティングシステムに表示されません。これを行う方法については、34章ストレージ Interconnect のスキャン を参照してください。



重要

LIP が必要な場合、この操作を実行する前に 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、オペレーティングシステムによって設定されていない場合には、sg3_utils パッケージの一部で、sg **_luns** コマンドを使用して、アレイでエクスポートされる LUN の一覧を確認します。これにより、**SCSI REPORT LUNS** コマンドを RAID アレイに発行し、存在する LUN の一覧を返します。

すべてのポートに単一の WWNN が実装されているファイバーチャネルストレージサーバーの場合は、**sysfs** で WWNN を検索することで、正しい **h**、**c**、および **t** の値 (すなわち、HBA 番号、HBA チャンネル、SCSI ターゲット ID) を決定することができます。

例31.1 Determin correct h、c、および t の値

*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るはずです。

```
/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
```

*これは、このターゲットに対する 4 つのファイバーチャネルルートがあることを示します（それぞれが 2 つのファイバーチャネル HBA。それぞれが 2 つのストレージポートになります）。LUN の値が **56** の場合、以下のコマンドは最初のパスを設定します。*

```
$ echo "0 2 56" > /sys/class/scsi_host/host5/scan
```

これは、新規デバイスへのパスごとに行う必要があります。

*すべてのポートに単一の WWNN を実装しないファイバーチャネルストレージサーバーの場合、**sysfs** で各 WWPN を検索して、正しい HBA 番号、HBA チャネル、および SCSI ターゲット ID を確認できます。*

*HBA 番号、HBA チャネル、および SCSI ターゲット ID を決定する別の方法は、新規デバイスと同じパスに設定されている別のデバイスを参照することです。これは、lsscsi、**scsi_id**、**multipath -l**、**ls -l /dev/disk/by-*** など、さまざまなコマンドで実行できます。この情報および新規デバイスの LUN 番号が上記のように使用して、新しいデバイスへのパスをプローブおよび設定できます。*

3. *デバイスへの SCSI パスを追加したら、**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ことを確認します。この時点で、デバイスは **md**、LVM、**mkfs**、または **mount** に追加できます。*

上記の手順の後には、実行中のシステムにデバイスを安全に追加できます。この場合、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）に関連するその他の手順により、オペレーティングシステムが現在のデバイスの接続を反映するように状態を更新できるようになります。ストレージ I/O が進行中の場合には推奨されません。

第32章 イーサネットインターフェース上でのファイバーチャネルの設定

Fibre-channel over Ethernet(FCoE)インターフェースを設定し、デプロイするには、以下の2 つのパッケージが必要です。

- **fcoe-utils**
- **lldpad**

これらのパッケージがインストールされたら、以下の手順に従って、仮想LAN(VLAN)でFCoE を有効にします。

手順32.1 FCoE を使用するためにイーサネットインターフェースの設定

1. 既存のネットワークスクリプト（例：**/etc/fcoe/cfg-eth0**）を、FCoE に対応するイーサネットデバイスの名前にコピーして、新しいVLAN を設定します。これにより、設定するデフォルトファイルが提供されます。FCoE デバイスが**ethX** の場合は、次のコマンドを実行します。

   ```
   # cp /etc/fcoe/cfg-eth0  /etc/fcoe/cfg-ethX
   ```

   必要に応じて**cfg-ethX** の内容を変更します。注記： ハードウェアの **DCBX クライアントを実装するネットワークインターフェースには、DCB_REQUIRED** を **no** に設定する必要があります。

2. システムの起動時にデバイスを自動的に読み込むように設定するには、対応する**/etc/sysconfig/network-scripts/ifcfg-ethX** ファイルに **ONBOOT=yes** を設定します。たとえば、FCoE デバイスがeth2 の場合は、それに応じて **/etc/sysconfig/network-scripts/ifcfg-eth2** を編集します。

3. 以下のコマンドを使用して、データセンターのブリッジデーモン(**dcbd**)を起動します。

   ```
   # /etc/init.d/lldpad start
   ```

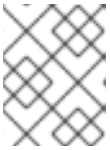
4. ハードウェアのDCBX クライアントを実装するネットワークインターフェースの場合は、この手順を省略して次の手順を実施します。

   ソフトウェアDCBX クライアントを必要とするインターフェースの場合は、次のコマンドを使用して、イーサネットインターフェースでデータセンターのブリッジングを有効にします。

   ```
   # dcbtool sc ethX dcb on
   ```

   次に、イーサネットインターフェースでFCoE を有効にするには、次のコマンドを実行します。

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

   注記

   これらのコマンドは、イーサネットインターフェースの **dcbd** 設定が変更されていない場合にのみ機能します。

5. 以下を使用してFCoE デバイスを読み込みます。

```
# ifconfig ethX up
```

6. 以下を使用してFCoE を起動します。

```
# service fcoe start
```

ファブリック上の他のすべての設定が正しいと仮定して、FCoE デバイスがすぐに表示されるはずです。設定したFCoE デバイスを表示するには、次のコマンドを実行します。

```
# fcoeadm -i
```

Red Hat は、FCoE を使用す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後に、FCoE と **lldpad** を起動時に実行させることを推奨します。これを行うには、以下のように **chkconfig** を使用します。

```
# chkconfig lldpad on
```

```
# chkconfig fcoe on
```

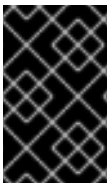
警告

DCB を実装する CNA で、ソフトウェアベースのDCB またはLLDP を実行しないでください。

一部の Combined Network Adapters(CNA)は、ファームウェアに Data Center Bridging(DCB)プロトコルを実装します。DCB プロトコルは、特定のネットワークリンクに DCB の送信元が1 つであることを前提としています。つまり、DCB の高レベルソフトウェア実装またはLink Layer Discovery Protocol(LLDP)は、DCB を実装する CNA で無効にする必要があります。

32.1. FIBRE-CHANNEL OVER ETHERNET(FCOE)ターゲットのセットアップ

32章イーサネットインターフェース上でのファイバーチャネルの設定 で説明しているように、FCoE でLUN をマウントする他に、FCoE 経由の他のマシンへのLUN のエクスポートもサポートされます。



重要

次に進む前に、32章イーサネットインターフェース上でのファイバーチャネルの設定 を参照して基本的なFCoE 設定が完了し、**fcoeadm -i** が設定済みのFCoE インターフェースを表示していることを確認します。

手順32.2 FCoE ターゲットの設定

1. FCoE ターゲットを設定するには、**fcoe-target-utils** パッケージとその依存関係をインストールする必要があります。

```
# yum install fcoe-target-utils
```

2. FCoE のターゲットは LIO カーネルターゲットをベースとしており、ユーザー空間デーモンは必要ありません。ただし、fcoe-target サービスが必要なカーネルモジュールを読み込み、再起動後も設定を維持するには、引き続き設定する必要があります。

```
# service fcoe-target start
```

```
# chkconfig fcoe-target on
```

3. FCoE ターゲットの設定は、期待通りに a **.conf** を編集するのではなく、**targetcli** ユーティリティーを使用して実行します。その後、設定が保存されるので、システムの再起動時に復元できます。

```
# targetcli
```

targetcli は、階層設定シェルです。シェル内のノード間での移動は **cd** を使用して、**ls** は現在の設定ノードでコンテンツを表示します。その他のオプションを表示するには、**コマンドのヘルプ** も利用できます。

4. バックストアとしてエクスポートするために、ファイル、ブロックデバイス、またはパススルー SCSI デバイスを定義します。

例32.1 デバイスを定義する例 1

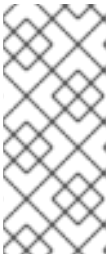
```
/> backstores/block create example1 /dev/sda4
```

これにより、**/dev/sda4** ブロックデバイスにマップするバックストア **example1** が作成されます。

例32.2 デバイスを定義する 2 の例

```
/> backstores/fileio create example2 /srv/example2.img 100M
```

これにより、指定されたファイルへマップする **example2** という名前のバックストアが作成されます。ファイルが存在しない場合は作成されます。ファイルサイズは K、M、または G の略語を使用でき、バッキングファイルが存在しない場合にのみ必要です。



注記

グローバルの **auto_cd_after_create** オプションがオンになっている場合（デフォルト）、create コマンドを実行すると、現在の設定ノードを新規作成されるオブジェクトに変更します。これは、**グローバル auto_cd_after_create=false を設定して** 無効にできます。**cd / を使用すると、root ノードに戻ることができます。**

5. FCoE インターフェースで FCoE ターゲットインスタンスを作成します。

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

FCoE インターフェースがシステムに存在する場合は、作成後のタブコンプリットブレッシングにより、 利用可能なインターフェースの一覧が表示されます。そうでない場合は、**fcoeadm -i** に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ます。

6. バックストアをターゲットインスタンスにマップします。

例32.3 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の例

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

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

7. FCoE イニシエーターから LUN へのアクセスを許可します。

```
/> acls/ create 00:99:88:77:66:55:44:33
```

LUN がイニシエーターからアクセス可能になるはずです。

8. **exit** または **ctrl+D** を入力して **targetcli** を終了します。

targetcli を終了すると、デフォルトで設定を保存します。ただし、**saveconfig** コマンドで明示的に保存することもできます。

詳細は、**targetcli** の man ページを参照してください。

第33章 システムの起動時にFCOE インターフェースを自動マウントする設定

注記

本セクションの手順は、Red Hat Enterprise Linux 6.1 より、**/usr/share/doc/fcoe-utils-version/README** で参照できます。マイナーリリース間における変更については、該当するドキュメント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**、およびその他の同様の方法でマウントできます。ただし、特定のサービスでは、システムの起動時にFCoE ディスクをマウントする必要がある場合があります。このような場合は、FCoE ディスクを必要とするサービスが開始される**前** に、**fcoe** サービスの実行**するやいなや** FCoE ディスクをマウントする必要があります。

システムの起動時にFCoE ディスクを自動的にマウントするように設定するには、**fcoe** サービスの起動スクリプトに適切なFCoE マウントコードを追加します。**fcoe** 起動スクリプトは **/etc/init.d/fcoe** です。

FCoE マウントコードは、シンプルなフォーマットのFCoE ディスク、LVM、またはマルチパスのデバイスノードを使用しているかどうかに関わらず、システム設定によって異なります。

例33.1 FCoE マウントコード

以下は、**/etc/fstab** 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FCoE マウントコードの例です。

```
mount_fcoe_disks_from_fstab()
 {
    local timeout=20
    local done=1
    local fcoe_disks=($(egrep 'by-path\/fc-.*_netdev' /etc/fstab | cut -d ' ' -f1))

    test -z $fcoe_disks && return 0

    echo -n "Waiting for fcoe disks . "
    while [ $timeout -gt 0 ]; do
  for disk in ${fcoe_disks[*]}; do
   if ! test -b $disk; then
    done=0
    break
   fi
  done

  test $done -eq 1 && break;
  sleep 1
  echo -n ". "
  done=1
  let timeout--
    done

    if test $timeout -eq 0; then
  echo "timeout!"
    else
  echo "done!"
```

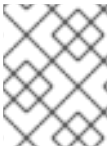
```
    fi

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

mount_fcoe_disks_from_fstab *関数は、* **fcoe サービススクリプトが fcoe mon** *デーモンを起動* **した後に** *呼び出す必要があります。これにより、* **/etc/fstab** *の以下のパスで指定された FCoE ディスクがマウントされます。*

```
/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- *および* **_netdev** *サブ文字列を含むエントリーは、* **mount_fcoe_disks_from_fstab** *関数が FCoE ディスクマウントエントリーを識別できるようにします。* **/etc/fstab** *エントリーの詳細は、* **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 マウントコードは独自のタイムアウト期間を実装する必要があります。*

第34章 ストレージINTERCONNECT のスキャン

特定のコマンドにより、1 つまたは複数の相互接続をリセット、スキャンし、スキャンできます。これにより、1 回の操作に複数のデバイスを追加または削除できます。このタイプのスキャンは、I/O 操作がタイムアウトし、デバイスを削除する際に遅延が発生する可能性があるため、中断する可能性があります。Red Hat は、**必要な場合にのみ相互** 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際には、以下の制限を確認してください。

- 実施された相互接続上のすべての I/O は、手順を実行する前に一時停止してフラッシュする必要があります。また、I/O を再開する前にチェックされたスキャンの結果が必要になります。
- デバイスを削除するのと同様に、システムがメモリー不足の状態になると相互スキャンは推奨されません。メモリープレッシャ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。空きメモリーが合計メモリーの 5% 未満の場合、Interconnect スキャンは 100 あたり 10 を超える合計メモリーの 5% 未満の場合は推奨されません。また、スワップがアクティブな場合には相互スキャンは推奨されません（**vmstat** 出力のゼロ以外の **si** および **so** 列）。**free** コマンドは、合計メモリーも表示することができます。

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することができます。

echo "1" > /sys/class/fc_host/hostN/issue_lip

（N はホスト番号に置き換えます。）

この操作はLIP(Loop Initialization Protocol) を実行し、相互接続をスキャンし、SCSI レイヤーが更新され、バス上のデバイスを反映させます。LIP は、基本的にはバスリセットで、デバイスの追加と削除を生じさせます。この手順は、ファイバーチャネルの相互接続に新しい SCSI ターゲットを設定するために必要です。

issue_lip は非同期操作であることに注意してください。このコマンドは、スキャン全体が完了する前に完了できます。**issue_lip** を終了するタイミングを決定するには、**/var/log/messages** を監視する必要があります。

lpfc ドライバー、**qla2xxx** ドライバー、および **bnx2fc** ドライバーは **issue_lip** をサポートします。Red Hat Enterprise Linux の各ドライバーがサポートする API 機能に関する詳細は、表26.1「ファイバーチャネル API 機能」 を参照してください。

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

Red Hat Enterprise Linux 5.4 に **/usr/bin/rescan-scsi-bus.sh** スクリプトが導入されました。デフォルトでは、このスクリプトはシステム上のすべての SCSI バスをスキャンし、SCSI レイヤーを更新して、バス上の新しいデバイスが反映されます。このスクリプトは、デバイスの削除と LIP の発行を可能にする追加のオプションを提供します。既知の問題など、このスクリプトの詳細は、38章論理ユニットの rescan-scsi-bus.sh の追加/削除 を参照してください。

echo "- - -" > /sys/class/scsi_host/hosth/scan

これは、31章ストレージデバイスまたはパスの追加 で説明されているコマンドと同じで、ストレージデバイスまたはパスを追加します。ただし、この場合は、チャンネル番号、SCSI ターゲット ID、および LUN の値はワイルドカードに置き換えられます。識別子とワイルドカードの組み合わせが許可されるため、必要に応じてコマンドを詳細または広範囲にすることができます。この手順では LUN を追加しますが、削除しません。

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

modprobe --remove driver-name コマンドを実行して、**modprobe driver-name** コマンドにより、ドライバーが制御するすべての相互接続の状態を完全に再初期化します。極端な場合もありますが、上記のコマンドは特定の状況に適しています。別のモジュールパラメーター値を使用して、

ドライバーを再起動する場合など、コマンドを使用できます。

第35章 iSCSI オフロードおよび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 NIC ポートにセッションをバインドするために iSCSI インターフェースを設定する方法を説明します。また、オフロードに対応するネットワークデバイスで使用するインターフェースを設定する方法を説明します。つまり、Chelsio、Broadcom、および ServerEngines からのデバイスです。

network サブシステムを設定すると、iSCSI インターフェースがバインディングに使用するパス/NIC を判断できます。たとえば、ポータルと NIC が異なるサブネットに設定されている場合には、バインディング用に iSCSI インターフェースを手動で設定する必要はありません。

iSCSI インターフェースをバインディング用に設定を試みる前に、最初に以下のコマンドを実行します。

```
$ ping -I ethX target_IP
```

ping が失敗すると、セッションを NIC にバインドできなくなります。その場合は、最初にネットワーク設定を確認してください。

35.1. 利用可能な IFACE 設定の表示

Red Hat Enterprise Linux 5.5 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 （**scsi_tcp** モジュールおよび **ib_iser** モジュールと同様）は、セッションごとに 1 つの接続とともに、セッションごとに iSCSI ホストインスタンス（**scsi_host** など）を割り当てます。これにより、**/sys/class_scsi_host** および **/proc/scsi** は、ログに記録した各接続/セッションの **scsi_host** を報告します。
- オフロード iSCSI （Chelsio **cxgb3i**、**B**roadx **bnx2i** および ServerEngines **be2iscsi** モジュールと同様）、このスタックは各 PCI デバイスに **scsi_host** を割り当てます。そのため、ホストバスアダプター上の各ポートは別の PCI デバイスとして表示され、HBA ポートごとに異なる **scsi_host** が表示されます。

両方のイニシエーターの実装を管理するために、**iscsiadm** は **iface** 構造を使用します。この構造では、セッションのバインドに使用される各 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**) の **/var/lib/iscsi/ifaces** に **iface** 設定を入力する必要があります。

利用可能な **iface** 設定を表示するには、**iscsiadm -m iface** を実行します。これにより、**iface** 情報が次の形式で表示されます。

```
iface_name transport_name,hardware_address,ip_address,net_ifacename,initiator_name
```

各値/setting の説明は以下の表を参照してください。

表35.1 iface Settings

設定	詳細
iface_name	**iface** 設定名。
transport_name	ドライバーの名前

設定	詳細
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** またはエイリアスバインディングに使用される名前。iSCSI オフロードの場合には、この値は再起動後も維持されないため、net **_iface_name** は **<empty>** になります。
initiator_name	この設定は、**/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オーバーライドするために使用されます。

例35.1 iscsiadm -m iface コマンドの出力例

以下は、**iscsiadm -m iface** コマンドの出力例です。

```
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、各 **iface** 設定には一意の名前（65 文字未満）が必要です。オフロードをサポート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35.2 iscsiadm -m iface output with a Chelsio network card

たとえば、Chelsio ネットワークカードを使用するシステムで **iscsiadm -m iface** の出力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>
```

また、より使いやすい方法で、特定の **iface** 設定も表示することができます。これを行うには、**-I iface_name** オプションを使用します。これにより、以下の形式で設定が表示されます。

```
iface.setting = value
```

例35.3 Chelsio コンバージドモードでの iface 設定の使用

前述の例を使用すると、同じ Chelsio コンバージドモードの **iface** 設定（つまり **iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07**）は以下のようになります。

```
# 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
```

35.2. ソフトウェア ISCSI のペースの設定

前述のように、セッションのバインドに使用される各ネットワークオブジェクトに **iface** 設定が必要です。

前

ソフトウェア iSCSI のペース **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
```

これにより、指定した **iface_name** を持つ新しい empty **iface** 設定が作成されます。既存の **iface** 設定が同じ **iface_name** である場合は、新しい空の設定で上書きされます。

iface 設定の特定の設定を設定するには、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35.4 Set MAC address of iface0

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** は、iface **名として** 使用しないでください。どちらの文字列も、後方互換性のために **iscsiadm** で使用される特殊な値です。**default** または **iser という名前の** 状態で手動で作成されると、後方互換性は無効になります。

35.3. ISCSI オフロードの該当設定

デフォルトでは、**iscsiadm** は各 Chelsio、Broadcom、および ServerEngines ポートごとに **iface** 設定を作成します。ペース **設定** を利用できるようにするには、ソフトウェア iSCSI でこのコマンドと同じコマンド（つまり **iscsiadm -m iface** ）を使用します。

iSCSI オフロード ***の*** *ネットワークカードの例を使用する前に、最初にデバイスが使用する IP アドレス (****target_IP****) を設定します。****be2iscsi*** *ドライバー（例： ServerEngines iSCSI HBA）を使用する ServerEngines デバイスの場合、IP アドレスは ServerEngines BIOS 設定画面に設定されます。*

Chelsio デバイスおよび Broadcom デバイスの場合、IP アドレスを設定する手順は、他の ***iface*** *設定と同じです。****iface*** *の IP アドレスを設定するには、以下を使用します。*

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

例35.5 Chelsio カードの iface IP アドレスの設定

たとえば、Chelsio カードの ***iface*** *IP アドレスを* ***cxgb3i.00:07:43:05:97:07****に設定するには 、 以下を使用します。****20.15.0.66***

```
# iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07 -o update -n iface.ipaddress -v 20.15.0.66
```

35.4. ポータルへのペースのバインディング/選択解除

iscsiadm *を使用して相互接続をスキャンするたびに、最初に* ***/var/lib/iscsi/ifaces*** *の各* ***iface*** *設定の* ***iface.transport*** *設定を確認します。次に、****iscsiadm*** *ユーティリティーは、検出されたポータルを* ***iface .transport が tcp の任意の iface*** *にバインドします。*

この動作は、互換性のために実装されました。これを上書きするには、以下のように ***-I iface_name*** *を使用して* ***iface*** *にバインドす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
```

*デフォルトでは、****iscsiadm*** *ユーティリティーは、オフロードを使用する* ***iface*** *設定に自動的にポータルをバインドしません。これは、お使いの環境の設定が* ***iface .transport*** *が* ***tcp*** *に設定されないためです。そのため、Chelsio、Broadcom、および ServerEngines ポートの* ***iface*** *設定は、検出されたポータルに手動でバインドする必要があります。*

また、ポータルが既存の ***iface*** *にバインドされないようにすることもできます。これを行うには、以下のように* ***default*** *を* ***iface_name*** *として使用します。*

```
# iscsiadm -m discovery -t st -p IP:port -I default -P 1
```

ターゲット間のバインディングと ***iface*** *を削除するには、以下を使用します。*

```
# iscsiadm -m node -targetname proper_target_name -I iface0 --op=delete[7]
```

特定の ***iface*** *のすべてのバインディングを削除するには、以下を使用します。*

```
# iscsiadm -m node -I iface_name --op=delete
```

特定のポータル（例： Equalogic ターゲット）のバインディングを削除するには、以下を使用します。

```
# iscsiadm -m node -p IP:port -I iface_name --op=delete
```

注記

/var/lib/iscsi/iface 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、*iscsiadm* を使用すると、ネットワークサブシステムが特定のポートで使用するデバイスを決定できます。

---

[7] Refer to 36章複数の LUN またはポータルを使用した iSCSI ターゲットのスキャン

proper_target_name の情報

第36章 複数の LUN またはポータルを使用した ISCSI ターゲットのスキャン

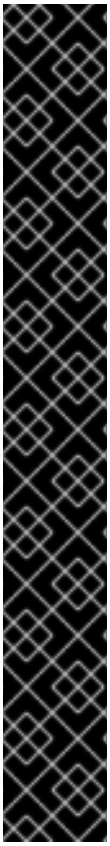
一部のデバイスモデル（例：EMC および Netapp）がある場合、1 つのターゲットに複数の論理ユニットまたはポータルが含まれる場合があります。この場合は、最初に **sendtargets** コマンドをホストに発行し、ターゲット上の新しいポータルを見つけます。次に、以下を使用して既存のセッションを再スキャンします。

```
# iscsiadm -m session --rescan
```

また、以下のようにセッションの **SID** 値を指定して、特定のセッションを再スキャンすることもできます。

```
# iscsiadm -m session -r SID --rescan[8]
```

デバイスが複数のターゲットに対応している場合は、ホストに **sendtargets** コマンドを実行し、**各ターゲット** の新しいポータルを見つける必要があります。次に、既存のセッションを再スキャンして、既存のセッションで新しい論理ユニットを検出します（つまり **--rescan** オプションを使用）。



重要

--targetname および -- portal の値の取得に使用される **sendtargets** コマンドは、**/var/lib/iscsi/nodes** データベースの内容を上書きします。次に、**/etc/iscsi/iscsid.conf** の設定を使用して、このデータベースが再設定されます。ただし、セッションが現在ログインしており、使用中の場合にはこれは発生しません。

新規のターゲット/ポータルを安全に追加したり、古いターゲットを削除したりするには、- **o new** または **-o delete** オプションをそれぞれ使用します。たとえば、**/var/lib/iscsi/nodes** を上書きせずに新規ターゲット/ポータルを追加するには、以下のコマンドを使用します。

```
iscsiadm -m discovery -t st -p target_IP -o new
```

検出中にターゲットが表示されなかった **/var/lib/iscsi/nodes** エントリーを削除するには、以下を使用します。

```
iscsiadm -m discovery -t st -p target_IP -o delete
```

また、以下のように両方のタスクを同時に実行することもできます。

```
iscsiadm -m discovery -t st -p target_IP -o delete -o new
```

sendtargets コマンドは以下の出力を生成します。

```
ip:port,target_portal_group_tag proper_target_name
```

例36.1 sendtargets コマンドの出力

たとえば、単一のターゲット、論理ユニット、およびポータルを持ち、**target_name** として **equallogic-iscsi1** を使用するデバイスの場合、出力は次のようになります。

> > *10.16.41.155:3260,0 iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1*

*proper **_target_name** と **ip:port**、**target_portal_group_tag** は、「iSCSI イニシエーターの作成」の同じ名前を持つ値と同じであることに注意してください。*

*この時点で、iSCSI デバイスを手動でスキャンするために必要な適切な **--targetname** および **--portal** の値が得られます。これには以下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*
> *[9]*

> ***例36.2 完全な iscsiadm コマンド***
>
> *前述の例（suitable **_target_name** が **equallogic-iscsi1**）を使用すると、完全なコマンドは以下のようになります。*
>
> > *# iscsiadm --mode node --targetname \ iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1 \ --portal 10.16.41.155:3260,0 --login[9]*

---

[8] For information on how to retrieve a session's SID value, refer to 「iSCSI イニシエーターの作成」

.

[9] This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (\) — should be treated as one command, sans backslashes.

第37章 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 を完全に変更するには、論理ユニット自体のサイズ変更と、対応するマルチパスデバイスのサイズ変更を反映しています（システムでマルチパスが有効になっている場合）。

オンラインの論理ユニットのサイズを変更するには、ストレージデバイスの配列管理インターフェースを使用して、論理ユニットサイズを変更して開始します。この手順はアレイごとに異なります。そのため、詳細については、ストレージアレイベンダーのドキュメントを参照してください。

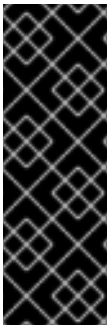
注記

オンラインファイルシステムのサイズを変更するには、パーティションが設定されたデバイスにファイルシステムを配置しないでください。

37.1. ファイバーチャネル論理ユニットのサイズ変更

オンラインの論理ユニットサイズを変更したら、論理ユニットを再スキャンして、システムが更新されたサイズを検出するようにします。ファイバーチャネルの論理ユニットにこれを行うには、次のコマンドを使用します。

```
$ echo 1 > /sys/block/sdX/device/rescan
```



重要

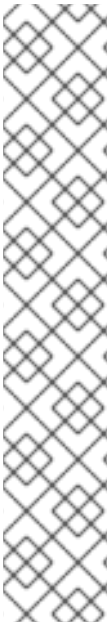
マルチパスを使用するシステムで fibre チャンネルの論理ユニットを再スキャンするには、マルチパスされた論理ユニットのパスを表す、sd デバイス（sd **1**、sd**2** など）ごとに前述のコマンドを実行します。マルチパス論理ユニットのパスであるデバイスを確認するには、multipath **multipath -ll** を使用します。次に、サイズ変更している論理ユニットと合致するエントリーを検索します。各エントリーの WWID を参照することが推奨されます。これにより、サイズ変更している論理ユニットと一致するものを簡単に見つけることができます。

37.2. ISCSI 論理ユニットのサイズ変更

オンラインの論理ユニットサイズを変更したら、論理ユニットを再スキャンして、システムが更新されたサイズを検出するようにします。iSCSI デバイスに対してこれを行うには、以下のコマンドを使用します。

```
# iscsiadm -m node --targetname target_name -R
```

target_name を、デバイスが置かれているターゲット名に置き換えます。



備考

以下のコマンドを使用して、iSCSI 論理ユニットを再スキャンすることもできます。

```
# iscsiadm -m node -R -I interface
```

***interface** を、サイズ変更された論理ユニット（例： **iface0**）の対応するインターフェース名に置き換えます。*このコマンドは、2 つの操作を実行します。

- *これは、コマンド **echo "- - -" > /sys/class/scsi_host/host/scan** と同じように新規デバイスをスキャンします（36章複数の LUN またはポータルを使用した iSCSI ターゲットのスキャンを参照）。*
- *これは、new/modified 論理ユニットに再スキャンします。これは、コマンド **echo 1 > /sys/block/sdX/device/rescan** と同じように実行できます。このコマンドは、fibre-channel の論理ユニットの再スキャンに使用されるものと同じであることに注意してください。*

37.3. マルチパスデバイスのサイズの更新

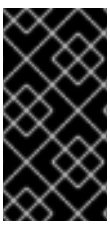
*システムでマルチパスが有効になっている場合は、論理ユニットサイズの論理ユニットサイズの変更を論理ユニットの対応するマルチパスデバイスに反映させる必要があります（論理ユニットのサイズ変更**後**）。Red Hat Enterprise Linux 5.3（以降）の場合は、**multipathd** を使用してこれを実行できます。そのためには、最初に **service multipathd status** のステータスを使用して、**multipathd** が実行していることを確認します。**multipathd** が動作していることを確認したら、以下のコマンドを実行します。*

```
# multipathd -k"resize map multipath_device"
```

***multipath_device** 変数は、**/dev/mapper** 内のデバイスの対応するマルチパスエントリーです。マルチパスがシステムでどのように設定されているかに応じて、**multipath_device** は 2 つの形式のいずれかになります。*

- ***mpathX。X** は、デバイスの対応するエントリーです（例： **mpath0**）*
- *WWID（例： **3600508b400105e210000900000490000**）*

*マルチパスエントリーのサイズを変更している論理ユニットを確認するには、**multipath -ll** を実行します。これにより、システム内の既存のマルチパスエントリーの一覧と、対応するデバイスのメジャー番号およびマイナー番号が表示されます。*



重要な影響

***multipath_deviceに置かれたコマンドがある場合は、multipathd -k"resize マップ multipath_device** を使用しないでください。つまり、(**/etc/multipath.conf** で) **no_path_retry** パラメーターを **"queue"** に設定し、デバイスへのアクティブなパスがない場合は、このコマンドを使用しないでください。*

*システムが Red Hat Enterprise Linux 5.0-5.2 を使用している場合は、**multipathd** デーモンに、サイズを変更した論理ユニットに行った変更を認識（調整）するよう指示する必要があります。*

手順37.1 Corresponding Multipath デバイスのサイズ変更（Red Hat Enterprise Linux 5.0 - 5.2 で必須）

1. *以下を使用して、マルチパスを設定したデバイスのデバイスマッパーテーブルをダンプします。*

dmsetup table multipath_device

2. ダンプされたデバイスマッパーテーブルを **table_name として保存し** ます。このテーブルはリロードされ、後で編集されます。

3. デバイスマッパーの表を確認します。各行の最初の 2 つの数字は、それぞれディスクの最初と終了セクターに対応します。

4. デバイスマッパーのターゲットを一時停止します。

dmsetup suspend multipath_device

5. 先に保存したデバイスマッパーテーブル（例： **table_name**）を開きます。ディスク内で 512 バイトのセクター数を反映するために 2 番目の数（ディスク終了セクターなど）を変更します。たとえば、新規ディスクサイズが 2GB の場合は、2 番目の数を 4194304 に変更します。

6. 変更したデバイスマッパー表を再読み込みします。

dmsetup reload multipath_device table_name

7. デバイスマッパーのターゲットを再開します。

dmsetup resume multipath_device

マルチパスの詳細は、『Red Hat Enterprise Linux 6 DM Multipath』ガイドを参照してください。

37.4. オンライン論理ユニットの読み取りおよび書き込み状態の変更

特定のストレージデバイスは、デバイスの状態を Read/Write(R/W) から Read-Only(RO) に変更し、RO から R/W に変更する機能を提供します。これは通常、ストレージデバイス上の管理インターフェースを使用して行われます。オペレーティングシステムは、変更時にデバイスの状態ビューを自動的に更新しません。本章の手順に従って、オペレーティングシステムが変更を認識できるようにします。

以下のコマンドを実行し、XYZ を必要なデバイス設計ator に置き換え、オペレーティングシステムの（R/W 状態）の現在のビューを特定します。

```
# blockdev --getro /dev/sdXYZ
```

以下のコマンドは、Red Hat Enterprise Linux 6 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** コマンドの出力の 2 行目の **ro** または **rw** フィールドを参照してください。以下に例を示します。

```
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 の状態を変更するには、以下の手順に従います。

手順37.2 R/W 状態の変更

1. デバイスを RO から R/W に移動するには、手順 2 を参照してください。

   デバイスを R/W から RO に移動するには、それ以上の書き込みが行われないようにします。これは、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ます。

   以下のコマンドを使用して、未処理の書き込み I/O がすべて完了していることを確認します。

   ```
   # blockdev --flushbufs /dev/device
   ```

   device を希望のデザインレーターに置き換えます。デバイスマッパーマルチパスの場合、これは **dev/mapper** のデバイスのエントリーです。たとえば、**/dev/mapper/mpath3 のようになります**。

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、論理ユニットの状態を R/W から RO または RO から R/W に変更します。その手順は、アレイごとに異なります。詳細は、該当するストレージアレイベンダーのドキュメントを参照してください。

3. デバイスの再スキャンを実行して、デバイスの R/W 状態のオペレーティングシステムのビューを更新します。デバイスマッパーのマルチパスを使用する場合には、そのデバイスマップを再読み込みするようにマルチパスを発行する前に、デバイスへの各パスに対してこの再スキャンを実行します。

   このプロセスについては、「論理ユニットの再スキャン」 で詳しく説明します。

37.4.1. 論理ユニットの再スキャン

「オンライン論理ユニットの読み取りおよび書き込み状態の変更」 で説明しているように、オンラインの論理ユニットの Read/Write 状態の変更後に、論理ユニットを再スキャンして、以下のコマンドでシステムが更新された状態を検出するようにします。

```
# echo 1 > /sys/block/sdX/device/rescan
```

マルチパス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のパスを表す各 sd デバイスに対して、上記のコマンドを実行します。たとえば、sd1、sd2、およびその他の sd デバイスすべてでコマンドを実行します。マルチパスユニットのパスであるデバイスを確認するには、**multipath -ll** を使用して、変更する論理ユニットに一致するエントリーを検索します。

例37.1 multipath -ll コマンドの使用

たとえば、上記の **multipath -ll** は、WWID 36001438005deb4710000500000640000 の LUN のパスを示しています。この場合は、以下を入力します。

```
# 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
```

37.4.2. マルチパスデバイスの R/W 状態の更新

マルチパスが有効になっている場合は、論理ユニットの再スキャン後に、その状態の変更を論理ユニットの対応するマルチパスドライブに反映される必要があります。次のコマンドを使用して、マルチパスデバイスマップを再読み込みします。

```
# multipath -r
```

multipath -ll コマンドを使用して変更を確認できます。

37.4.3. 資料

詳細は、Red Hat ナレッジベース を参照してください。これにアクセスするには、ログイン https://www.redhat.com/wapps/sso/login.html?redirect=https://access.redhat.com/knowledge/ に移動してログインします。その後に記事にアクセスします https://access.redhat.com/kb/docs/DOC-32850。

第38章 論理ユニットの RESCAN-SCSI-BUS.SH の追加/削除

sg3_utils パッケージは **rescan-scsi-bus.sh** スクリプトを提供し、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できます（デバイスがシステムに追加された後）。**rescan-scsi-bus.sh** スクリプトでは、サポートされているデバイスで **issue_lip** も実行できます。このスクリプトの使用方法に関する詳細は、**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。

既知の問題(RESCAN-SCSI-BUS.SH)

rescan-scsi-bus.sh スクリプトを使用する場合は、以下の既知の問題に注意してください。

- **rescan-scsi-bus.sh** が適切に機能するには、**LUN0** を最初にマッピングした論理ユニットである必要があります。**rescan-scsi-bus.sh** は、**LUN0** の場合、最初にマップされた論理ユニットのみを検出できます。**--nooptscan** オプションを使用する場合でも、最初にマップされた論理ユニットを検出する場合を除き、**rescan-scsi-bus.sh** は、他の論理ユニットをスキャンできません。

- 競合状態では、論理ユニットが初めてマッピングされている場合は、rescan **-scsi-bus.sh** を 2 回実行する必要があります。最初のスキャン中に、**rescan-scsi-bus.sh** は **LUN0** のみを追加します。その他の論理ユニットはすべて 2 つ目のスキャンに追加されます。

- **rescan-scsi-bus.sh** スクリプトのバグにより **--remove** オプションが使用される場合に、論理ユニットサイズの変更を認識するための機能が誤って実行されます。

- **rescan-scsi-bus.sh** スクリプトは、ISCSI 論理ユニットの削除を認識しません。

第39章 リンクループ動作の変更

本セクションでは、fibre チャンネルまたは iSCSI プロトコルのいずれかを使用するデバイスのリンク切れ動作を変更する方法を説明します。

39.1. ファイバーチャンネル

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は、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、以下のコマンドを実行します。

```
$ cat /sys/block/device/device/state
```

このコマンドは、デバイスがブロックされている場合は **blocked** を返します。デバイスが正常に動作している場合、このコマンドは通常 **running** を返します。

手順39.1 リモートポートの状態の決定

1. リモートポートの状態を確認するには、以下のコマンドを実行します。

```
$ cat
/sys/class/fc_remote_port/rport-H:B:R/port_state
```

2. このコマンドは、リモートポート（そのポートからアクセスしたデバイスとともに）がブロックされると **Blocked** を返します。リモートポートが正常に動作している場合、コマンドは **Online** を返します。

3. **dev_loss_tmo** 秒以内に問題が解決されない場合、rport およびデバイスはブロックされず、そのデバイスで実行されているすべての I/O（そのデバイスに送信された新しい I/O とともに）は失敗します。

手順39.2 dev_loss_tmoの変更

- **dev_loss_tmo** の値を変更するには、必要な値の **echo** をファイルに送信します。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。

リンクまたはターゲットポートの損失が **dev_loss_tmo** を超えると、**scsi_device** デバイスおよび **sdN** デバイスが削除されます。ターゲットポートの SCSI ID バインディングが保存されます。ターゲットが返されると、SCSI アドレスおよび **sdN**割り当てが変更される可能性があります。LUN 設定がターゲットポートの背後で変更された場合は、SCSI アドレスが変更されます。**sdN** 名は、LUN 検出プロセスのタイミングのバリエーションや、ストレージ内で LUN 設定の変更により変化する可能性があります。これらの割り当ては、28章永続的な命名 に記載されているように永続されません。永続的なデバイスの命名方法は、28章永続的な命名 セクションを参照してください。

39.2. DM-MULTIPATHを使用した ISCSI 設定

dm-multipath を実装する場合は、iSCSI タイマーを設定して、即座にマルチパスレイヤーにコマンドを定義することが推奨されます。これを設定するには、**/etc/multipath.conf** の **device {** に以下の行をネストします。

```
features "1 queue_if_no_path"
```

これにより、**dm-multipath** 層ですべてのパスが失敗すると、I/O エラーが再試行され、キューに入れられます。

SAN で問題の監視を改善するには、iSCSI タイマーをさらに調整する必要がある場合があります。設定可能な iSCSI タイマーは、NOP -Out Interval/Timeouts および **replacement_timeout** で説明されています。これについては、以下のセクションで説明します。

39.2.1. NOP-Out Interval/Timeout

SAN の問題の監視を容易にするために、iSCSI レイヤーは NOP-Out リクエストを各ターゲットに送信します。NOP-Out 要求時間がタイムアウトすると、実行中のコマンドが失敗し、可能な場合は SCSI レイヤーにコマンドを再びキューに入れることで iSCSI レイヤーが応答します。

dm-multipath を使用するとき、SCSI レイヤーは実行中のコマンドに失敗し、マルチパス層に対応します。次に、マルチパスレイヤーは別のパスでこれらのコマンドを再再試行します。**dm-multipath** を使用 **しない** 場合、これらのコマンドは、すべて失敗する前に 5 回リトライされます。

NOP-Out 要求の間隔は、デフォルトで 10 秒です。これを調整するには、**/etc/iscsi/iscsid.conf** を開き、以下の行を編集します。

```
node.conn[0].timeo.noop_out_interval = [interval value]
```

これが設定されると、iSCSI レイヤーは [interval value] 秒ごとに NOP-Out リクエストを各ターゲットに送信します。

デフォルトでは、NOP-Out 要求は 10 秒でタイムアウトします。[10]. これを調整するには、**/etc/iscsi/iscsid.conf** を開き、以下の行を編集します。

```
node.conn[0].timeo.noop_out_timeout = [timeout value]
```

これにより 、[timeout 値] 秒後に NOP-Out 要求をタイムアウトするように iSCSI レイヤーが設定されます。

SCSI エラーハンドラー

SCSI Error Handler が実行されている場合は、NOP-Out 要求がそのパスでタイムアウトすると、パスでコマンドの実行がすぐに失敗します。代わりに、これらのコマンドは **replacement_timeout** 秒 **後** に失敗します。**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し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。

```
# iscsiadm -m session -P 3
```

39.2.2. replacement_timeout

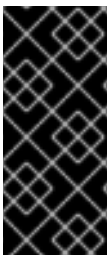
replacement_timeout は、タイムアウトしたパス/セッションが自身を再確立するまで iSCSI レイヤーが待機してから、コマンドが失敗する時間を制御します。デフォルトの **replacement_timeout** 値は 120 秒です。

replacement_timeout を調整するには、**/etc/iscsi/iscsid.conf** を開き、以下の行を編集します。

```
node.session.timeo.replacement_timeout = [replacement_timeout]
```

/etc/multipath.conf の **1 つの queue_if_no_path** オプションは、iSCSI タイマーを直ちにマルチパスレイヤーに遅延させるように設定します（「**dm-multipath** を使用した iSCSI 設定」を参照）。この設定により、I/O エラーがアプリケーションに伝搬されないようにします。これにより **replacement_timeout** を 15-20 秒に設定することができます。

下限の **replacement_timeout** を設定することで、I/O は新しいパスへすぐに送信され、(NOP-Out タイムアウトの場合) iSCSI レイヤーが失敗したパス/セッションの再確立を試行します。すべてのパスがタイムアウトになると、マルチパスおよび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 I/O をキューに入れます。



重要な影響

考慮事項がフェイルオーバーの速度またはセキュリティーであるかどうかに関わらず、**replacement_timeout** に推奨される値は他の要素によって異なります。これらの要因には、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が含まれます。そのため、ミッションクリティカルなシステムに適用する前に、新しい設定を **replacements_timeout** の新しい設定を徹底的にテストすることが推奨されます。

39.3. iSCSI ルート

iSCSI ディスクから直接 root パーティションにアクセスする場合は、iSCSI レイヤーがパス/セッションの再確立を試行する機会が数回あるように、iSCSI タイマーを設定する必要があります。さらに、コマンドは、SCSI レイヤーに素早く再度キューに入れないでください。これは、**dm-multipath を実装する際に実行すべき内容の** 反対になります。

まずは、NOP-Outs を無効にする必要があります。これは、NOP-Out の間隔とタイムアウトをゼロに設定すると実行できます。これを設定するには、**/etc/iscsi/iscsid.conf** を開き、以下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
```

この行では、**replacement_timeout** を高い数に設定する必要があります。これにより、システムに対して、パス/セッションが自身を再確立するまでの待機時間を待機するように指示されます。**replacement_timeout** を調整するには、**/etc/iscsi/iscsid.conf** を開き、以下の行を編集します。

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

/etc/iscsi/iscsid.conf の設定後に、影響を受けるストレージの再検索を実行する必要があります。これにより、システムが **/etc/iscsi/iscsid.conf** の新しい値を読み込み、使用できます。iSCSI デバイスの検出方法に関する詳細は、36章複数の LUN またはポータルを使用した iSCSI ターゲットのスキャン を参照してください。

特定のセッションのタイムアウトの設定

*特定のセッションのタイムアウトを設定し、（**/etc/iscsi/iscsid.conf**を使用する代わりに）非永続的なものにすることもできます。それを行うには、以下のコマンドを実行します（それに応じて変数を置き換えます）。*

```
# iscsiadm -m node -T target_name -p target_IP:port -o update -n
node.session.timeo.replacement_timeout -v $timeout_value
```

重要な影響

*ここで説明している設定は、root パーティションアクセスに関連する iSCSI セッションに推奨されます。他のタイプのストレージ（**dm-multipath**を使用するシステムでは）iSCSI セッションは、「**dm-multipath**を使用した iSCSI 設定」 を参照してください。*

---

[10] 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 リクエストのタイムアウトは 15 秒でした。

第40章 SCSI コマンドタイマーおよびデバイスステータスの制御

Linux SCSI レイヤーは、各コマンドにタイマーを設定します。このタイマーが期限切れになると、SCSI レイヤーは ホストバスアダプター (HBA) を累積し、未処理のすべてのコマンドがタイムアウトまたは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アクティベートします。

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（同様に 1 回実行します）。

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. ホストをリセットします。

この操作をすべて失敗すると、デバイスは **offline** 状態に設定されます。これが発生すると、問題が修正され、ユーザーがデバイスを **running** に設定するまで、そのデバイスへのすべての I/O が失敗します。

ただし、デバイスが fibre チャンネルのプロトコルを使用しており、**rport** がブロックされている場合は、プロセスは異なります。このような場合、ドライバーは、**rport** が再びオンラインになるまで数秒待機してからエラーハンドラーを有効にします。これにより、一時的なトランスポートの問題が原因でデバイスをオフラインにすることはできません。

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、次のコマンドを実行します。

```
$ echo running > /sys/block/device-name/device/state
```

コマンドタイマー

コマンドタイマーを制御するには、**/sys/block/device-name/device/timeout** に書き込むことができます。これを行うには、以下を実行します。

echo value /sys/block/device-name/device/timeout

value は、実装するタイムアウト値（秒単位）です。

第41章 オンラインストレージ設定のトラブルシューティング

本セクションでは、オンラインストレージの再設定中に、一般的な問題の解決策を説明します。

論理ユニットの削除ステータスは、ホストに反映されません。

設定されたファイルャーで論理ユニットが削除されると、その変更はホストに反映されません。このような場合は、論理ユニットが 古い 状態になっていたため、**dm-multipath** が使用されたときに **lvm** コマンドが無期限にハングします。

これを回避するには、以下の手順を行います。

手順41.1 Around Stale 論理ユニット

1. **/etc/lvm/cache/.cache** のどの **mpath** リンクエントリーが古い論理ユニットに固有のものであるかを判断します。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41.1 特定の mpath リンクエントリーの特定

たとえば、**stale-logical-unit** が 3600d0230003414f30000203a7bc41a00 の場合、以下のような結果が表示されます。

```
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** リンクにマッピングされることを意味します。

2. 次に、**/etc/lvm/cache/.cache** を開きます。**stale-logical-unit** が含まれるすべての行と、**stale-logical-unit** がマップする **mpath** リンクをすべて削除します。

例41.2 関連する行の削除

直前の手順で同じ例を使用して、削除する必要のある行は以下のとおりです。

```
/dev/dm-4
/dev/dm-5
/dev/mapper/3600d0230003414f30000203a7bc41a00
/dev/mapper/3600d0230003414f30000203a7bc41a00p1
/dev/mpath/3600d0230003414f30000203a7bc41a00
/dev/mpath/3600d0230003414f30000203a7bc41a00p1
```

付録A 改訂履歴

改訂 2-82 Mon Jun 4 2018 Marek Suchánek
非同期のアップデート

改訂 2-81 Wed Mar 21 2018 Marek Suchánek
非同期のアップデート

改訂 2-70 Mon Mar 13 2017 Milan Navratil
6.9 GA 公開用ドキュメントの準備。

改訂 2-64 Thu May 10 2016 Milan Navratil
6.8 GA 公開用ドキュメントの準備

改訂 2-63 Thu Mar 31 2016 Milan Navratil
複数のマイナーな改善による非同期更新。

改訂 2-52 Wed Mar 25 2015 Jacquelynn East
ext のバックアップおよびリストアの章を追加

改訂 2-51 Thu Oct 9 2014 Jacquelynn East
6.6 GA リリース向けバージョン。

改訂 2-38 Mon Nov 18 2013 Jacquelynn East
6.5 GA のビルド

改訂 2-35 Thu Sep 05 2013 Jacquelynn East
fsck のドキュメントセクションを追加。

改訂 2-11 Mon Feb 18 2013 Jacquelynn East
6.4 GA リリース向けバージョン。

改訂 2-1 Fri Oct 19 2012 Jacquelynn East
6.4 ベータ版のブランチ化。
大きな構造の並べ替えに基づいて新規編集が作成されました。

改訂 1-45 Mon Jun 18 2012 Jacquelynn East
6.3 リリースのバージョン。