Red Hat Enterprise Linux 8

ストレージデバイスの管理

Red Hat Enterprise Linux 8 における単一ノードストレージの導入と設定

Last Updated: 2019-05-07

Red Hat Enterprise Linux 8 ストレージデバイスの管理

Red Hat Enterprise Linux 8 における単一ノードストレージの導入と設定

法律上の通知

概要

本書は、Red Hat Enterprise Linux 8 でストレージデバイスを効果的に管理する方法を説明します。

目次

RED HAT ドキュメントへのフィードバック

ドキュメントの改善に関するご意見やご要望をお聞かせください。

- 特定の文章に簡単なコメントを記入する場合は、ドキュメントが Multi-page HTML 形式になっているのを確認してください。コメントを追加する部分を強調表示し、そのテキストの下に表示される **Add Feedback** ポップアップをクリックし、表示された手順に従ってください。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
    1. Bugzilla の Web サイトにアクセスします。
    2. Component で **Documentation** を選択します。
    3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも記入してください。
    4. **Submit Bug** をクリックします。

第1章 パーティションの使用

システム管理者は、以下の手順に従ってさまざまな種類のディスクパーティションを作成、削除、および変更できます。

ブロックデバイスでパーティションを使用することの長所と短所の概要は、ナレッジベースの記事「What are the advantages and disadvantages to using partitioning on LUNs, either directly or with LVM in between?」を参照してください。

1.1. パーティションテーブルの表示

システム管理者は、ブロックデバイスのパーティションテーブルを表示して、パーティションレイアウトと個々のパーティションに関する詳細を確認できます。

1.1.1. parted を使用したパーティションテーブルの表示

この手順では、**parted** ユーティリティーを使用してブロックデバイスのパーティションテーブルを表示する方法を説明します。

手順

1. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

   - **block-device** を、調べるデバイスへのパスに置き換えます。たとえば、**/dev/sda** に置き換えます。

2. パーティションテーブルを表示します。

```
(parted) print
```

3. 次のコマンドを使用して、次に調べるデバイスに切り替えることもできます。

```
(parted) select block-device
```

関連資料

- man ページの **parted(8)**

1.1.2. parted print の出力例

このセクションでは、**parted** シェルの **print** コマンドの出力例を示し、出力内のフィールドを説明します。

例1.1 **print** コマンドの出力

```
Model: ATA SAMSUNG MZNLN256 (scsi)
Disk /dev/sda: 256GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
```

```
Number  Start   End     Size    Type      File system  Flags
 1      1049kB  269MB   268MB   primary   xfs          boot
 2      269MB   34.6GB  34.4GB  primary
 3      34.6GB  45.4GB  10.7GB  primary
 4      45.4GB  256GB   211GB   extended
 5      45.4GB  256GB   211GB   logical
```

以下は、フィールドの説明です。

Model: ATA SAMSUNG MZNLN256 (scsi)

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。

Disk /dev/sda: 256GB

ディスクラベルの種類。

Number

パーティション番号。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 and **End**

デバイスにおけるパーティションの開始場所と終了場所。

Type

有効なタイプは、メタデータ、フリー、プライマリー、拡張、または論理です。

File system

ファイルシステムの種類。ファイルシステムの種類が不明な場合は、デバイスの **File system** フィールドに値が表示されません。**parted** ユーティリティーは、暗号化されたデバイスのファイルシステムを認識できません。

Flags

パーティションのフラグ設定一覧。利用可能なフラグは **boot**、**root**、**swap**、**hidden**、**raid**、**lvm** または **lba** です。

1.2. ディスクへのパーティションテーブルの作成

システム管理者は、さまざまな種類のパーティションテーブルを使用してブロックデバイスをフォーマットし、そのデバイスでパーティションを使用できるようにすることができます。

警告

パーティションテーブルを使用してブロックデバイスをフォーマットすると、そのデバイスに保存されているすべてのデータが削除されます。

1.2.1. ディスク上のパーティション変更前の留意事項

このセクションでは、パーティションを作成、削除、またはサイズ変更する前に考慮すべき重要な点を説明します。

注記

このセクションでは、IBM System z アーキテクチャーに固有の DASD パーティションテーブルを説明しません。DASD は、IBM Knowledge Center の「What you should know about DASD」を参照してください。

パーティションの最大数
デバイスのパーティション数は、パーティションテーブルの種類により制限されます。

- マスターブートレコード (MBR) パーティションテーブルでフォーマットされたデバイスでは、次のいずれかの数だけパーティションを設定できます。
  - 最大 4 つのプライマリーパーティション
  - 最大 3 つのプライマリーパーティションおよび 1 つの拡張パーティション、ならびにその拡張内に複数の論理パーティション
- GUID パーティションテーブル (GPT) を使用してフォーマットされたデバイスのパーティションの最大数は 128 個になります。GPT 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できますが、**parted** ユーティリティーで用いられる一般的な方法で得られるエリアは、128 個に制限されます。

パーティションの最大サイズ
デバイスのパーティションのサイズは、パーティションテーブルの種類により制限されます。

- マスターブートレコード (MBR) パーティションテーブルでフォーマットされたデバイスの最大サイズは 2TiB になります。
- GUID パーティションテーブル (GPT) でフォーマットされたデバイスの最大サイズは 8ZiB になります。

2TiB を超えるパーティションを作成する場合は、ディスクを GPT でフォーマットする必要があります。

サイズ調整
parted ユーティリティーを使用した場合は、パーティションサイズを指定する際のサフィックスを選択できます。

MiB、GiB、または TiB

サイズは 2 のべき乗で表されます。

- パーティションの開始点は、サイズが指定する正確なセクターに調整されます。
- 終了点は、指定されたサイズから 1 セクターを引いたサイズに調整されます。

MB、GB、または TB

サイズは 10 のべき乗で表されます。
開始点と終了点は、指定された単位の半分以内に置かれます。たとえば、MB サフィックスを使用する場合は ±500 KB です。

1.2.2. パーティションテーブルの種類の比較

このセクションでは、ブロックデバイスに作成できるさまざまな種類のパーティションテーブルのプロパティを比較します。

表1.1 パーティションテーブルの種類

パーティションテーブル	パーティションの最大数	パーティションの最大サイズ
マスターブートレコード (MBR)	4 つのプライマリー。または 3 つのプライマリーと、拡張パーティション内に 12 の論理	2TiB
GUID パーティションテーブル (GPT)	128	8ZiB

1.2.3. parted を使用したディスクでのパーティションテーブルの作成

この手順では、**parted** ユーティリティーを使用するパーティションテーブルでブロックデバイスをフォーマットする方法を説明します。

手順

1. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

   - **block-device** を、パーティションテーブルを作成するデバイスへのパスに置き換えます。たとえば、**/dev/sda** に置き換えます。

2. デバイスにパーティションテーブルがあるかどうかを確認します。

```
(parted) print
```

   デバイスにパーティションが含まれている場合は、次の手順でパーティションを削除します。

3. 新しいパーティションテーブルを作成します。

```
(parted) mklabel table-type
```

   - **table-type** を目的のパーティションテーブルの種類に置き換えます。
     - MBR の場合は **msdos**
     - GPT の場合は **gpt**

   例1.2 GPT テーブルの作成

   たとえば、ディスクに GPT テーブルを作成するには以下を使用します。

```
(parted) mklabel gpt
```

   このコマンドを入力するとすぐに変更が行われるため、実行する前によく確認してください。

4. パーティションテーブルを表示して、パーティションテーブルが存在することを確認します。

```
(parted) print
```

5. **parted** シェルを終了します。

```
(parted) quit
```

関連資料

- man ページの **parted(8)**

次のステップ

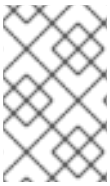
- デバイスにパーティションを作成します。詳細は「パーティションの作成」を参照してください。

1.3. パーティションの作成

システム管理者は、ディスクに新しいパーティションを作成できます。

1.3.1. ディスク上のパーティション変更前の留意事項

このセクションでは、パーティションを作成、削除、またはサイズ変更する前に考慮すべき重要な点を説明します。



注記

このセクションでは、IBM System z アーキテクチャーに固有の DASD パーティションテーブルを説明しません。DASD は、IBM Knowledge Center の「What you should know about DASD」を参照してください。

パーティションの最大数
デバイスのパーティション数は、パーティションテーブルの種類により制限されます。

- マスターブートレコード **(MBR)** パーティションテーブルでフォーマットされたデバイスでは、次のいずれかの数だけパーティションを設定できます。
  - 最大 4 つのプライマリーパーティション
  - 最大 3 つのプライマリーパーティションおよび 1 つの拡張パーティション、ならびにその拡張内に複数の論理パーティション
- **GUID** パーティションテーブル **(GPT)** を使用してフォーマットされたデバイスのパーティションの最大数は 128 個になります。GPT 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できますが、**parted** ユーティリティーで用いられる一般的な方法で得られるエリアは、128 個に制限されます。

パーティションの最大サイズ
デバイスのパーティションのサイズは、パーティションテーブルの種類により制限されます。

- マスターブートレコード **(MBR)** パーティションテーブルでフォーマットされたデバイスの最大サイズは 2TiB になります。
- **GUID** パーティションテーブル **(GPT)** でフォーマットされたデバイスの最大サイズは 8ZiB になります。

2TiB を超えるパーティションを作成する場合は、ディスクを GPT でフォーマットする必要があります。

サイズ調整
parted ユーティリティーを使用した場合は、パーティションサイズを指定する際のサフィックスを選択できます。

MiB、GiB、または TiB

サイズは 2 のべき乗で表されます。

- パーティションの開始点は、サイズが指定する正確なセクターに調整されます。
- 終了点は、指定されたサイズから 1 セクターを引いたサイズに調整されます。

MB、GB、または TB

サイズは 10 のべき乗で表されます。
開始点と終了点は、指定された単位の半分以内に置かれます。たとえば、MB サフィックスを使用する場合は ±500 KB です。

1.3.2. パーティションタイプ

このセクションでは、パーティションのタイプを指定するさまざまな属性を説明します。

パーティションタイプまたはフラグ
パーティションタイプ、またはフラグは、実行中のシステムではほとんど使用されません。ただし、パーティションタイプは、**systemd-gpt-auto-generator** など、デバイスを自動的に識別してマウントするためにパーティションタイプを使用するオンザフライジェネレーターにとって重要です。

- **parted** ユーティリティーは、パーティションタイプを **flags** にマッピングすることでパーティションタイプを制御します。parted ユーティリティーが処理できるのは、特定のパーティションタイプ (LVM、スワップ、または RAID など) のみです。
- **fdisk** ユーティリティーは、16 進コードを指定することにより、あらゆる種類のパーティションタイプをサポートします。

パーティションファイルシステムのタイプ
parted ユーティリティーは、パーティションを作成するときにオプションでファイルシステムタイプ引数を受け付けます。値は以下の目的で使用されます。

- MBR にパーティションフラグを設定します。
- GPT にパーティション UUID タイプを設定します。たとえば、ファイルシステムタイプの **swap**、**fat**、または **hfs** には、異なる GUID が設定されます。デフォルト値は Linux Data GUID です。

引数によりパーティション上のファイルシステムが変更されることはありません。サポートされているフラグまたは GUID を変更するだけです。

次のファイルシステムのタイプがサポートされています。

- **xfs**
- **ext2**
- **ext3**
- **ext4**

- **fat16**
- **fat32**
- **hfs**
- **hfs+**
- **linux-swap**
- **ntfs**
- **reiserfs**

1.3.3. parted を使用したパーティションの作成

この手順では、**parted** ユーティリティーを使用してブロックデバイスに新しいパーティションを作成する方法を説明します。

前提条件

- ディスクにパーティションテーブルがある。ディスクのフォーマット方法の詳細は、「ディスクへのパーティションテーブルの作成」 を参照してください。
- 2TiB を超えるパーティションを作成する場合は、ディスクを GUID パーティションテーブル (GPT) でフォーマットしておく。

手順

1. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

   - **block-device** を、パーティションを作成するデバイスへのパスに置き換えます。たとえば、**/dev/sda** に置き換えます。

2. 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。

```
(parted) 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。
   - パーティションテーブルから、以下を確認します。
     - 新しいパーティションの開始点と終了点
     - MBR で、どのパーティションタイプにすべきか

3. 新しいパーティションを作成します。

```
(parted) mkpart part-type name fs-type start end
```

   - **part-type** を、パーティションテーブルに基づき **primary**、**logical**、または **extended** に置き換えます。これは MBR パーティションテーブルにのみ適用されます。

- **name** を任意のパーティション名に置き換えます。これは GPT パーティションテーブルに必要です。
- **fs-type** を **xfs**、**ext2**、**ext3**、**ext4**、**fat16**、**fat32**、**hfs**、**hfs+**、**linux-swap**、**ntfs**、または **reiserfs** のいずれかに置き換えます。**fs-type** パラメーターは任意です。**parted** は、パーティション上にファイルシステムを作成しません。
- **start** と **end** をパーティションの開始点と終了点を決定するサイズに置き換えます (ディスクの開始からカウントします)。**512MiB**、**20GiB** または **1.5TiB** などのサイズサフィックスを使用できます。デフォルトサイズはメガバイトです。

例1.3 小さなプライマリーパーティションの作成

たとえば、MBR テーブルに 1024MiB から 2048MiB までのプライマリーパーティションを作成するには、次のコマンドを使用します。

```
(parted) mkpart primary 1024MiB 2048MiB
```

このコマンドを入力するとすぐに変更が行われるため、実行する前によく確認してください。

4. パーティションテーブルを表示して、作成されたパーティションがパーティションタイプ、ファイルシステムタイプ、サイズが、パーティションテーブルに正しく表示されていることを確認します。

```
(parted) print
```

5. **parted** シェルを終了します。

```
(parted) quit
```

6. 次のコマンドを使用して、システムが新しいデバイスノードを登録するまで待機します。

```
# udevadm settle
```

7. カーネルが新しいパーティションを認識していることを確認します。

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

関連資料

- man ページの **parted(8)**

1.3.4. fdisk を使用したパーティションタイプの設定

この手順では、**fdisk** ユーティリティーを使用してパーティションタイプまたはフラグを設定する方法を説明します。

前提条件

- ディスクにパーティションがある。

手順

1. インタラクティブな **fdisk** シェルを起動します。

```
# fdisk block-device
```

   - **block-device** を、パーティションタイプを設定するデバイスへのパスに置き換えます。たとえば、**/dev/sda** に置き換えます。

2. 現在のパーティションテーブルを表示して、パーティションのマイナー番号を確認します。

```
Command (m for help): print
```

   現在のパーティションタイプは **Type** 列で、それに対応するタイプ ID は **Id** 列で確認できます。

3. パーティションタイプコマンドを入力し、マイナー番号を使用してパーティションを選択します。

```
Command (m for help): type
Partition number (1,2,3 default 3): 2
```

4. 必要に応じて、利用可能な 16 進コードを一覧表示します。

```
Hex code (type L to list all codes): L
```

5. パーティションタイプを設定します。

```
Hex code (type L to list all codes): 8e
```

6. 変更を書き込み、**fdisk** シェルを終了します。

```
Command (m for help): write
The partition table has been altered.
Syncing disks.
```

7. 変更を確認します。

```
# fdisk --list block-device
```

1.4. パーティションの削除

システム管理者は、ディスク領域を解放するために使用されなくなったディスクパーティションを削除できます。

警告

パーティションを削除すると、そのパーティションに保存されているすべてのデータが削除されます。

1.4.1. ディスク上のパーティション変更前の留意事項

このセクションでは、パーティションを作成、削除、またはサイズ変更する前に考慮すべき重要な点を説明します。

注記

このセクションでは、IBM System z アーキテクチャーに固有の DASD パーティションテーブルを説明しません。DASD は、IBM Knowledge Center の「What you should know about DASD」を参照してください。

パーティションの最大数
デバイスのパーティション数は、パーティションテーブルの種類により制限されます。

- マスターブートレコード **(MBR)** パーティションテーブルでフォーマットされたデバイスでは、次のいずれかの数だけパーティションを設定できます。
  - 最大 4 つのプライマリーパーティション
  - 最大 3 つのプライマリーパーティションおよび 1 つの拡張パーティション、ならびにその拡張内に複数の論理パーティション
- **GUID** パーティションテーブル **(GPT)** を使用してフォーマットされたデバイスのパーティションの最大数は 128 個になります。GPT 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できますが、**parted** ユーティリティーで用いられる一般的な方法で得られるエリアは、128 個に制限されます。

パーティションの最大サイズ
デバイスのパーティションのサイズは、パーティションテーブルの種類により制限されます。

- マスターブートレコード **(MBR)** パーティションテーブルでフォーマットされたデバイスの最大サイズは 2TiB になります。
- **GUID** パーティションテーブル **(GPT)** でフォーマットされたデバイスの最大サイズは 8ZiB になります。

2TiB を超えるパーティションを作成する場合は、ディスクを GPT でフォーマットする必要があります。

サイズ調整
parted ユーティリティーを使用した場合は、パーティションサイズを指定する際のサフィックスを選択できます。

MiB、GiB、または TiB

サイズは 2 のべき乗で表されます。

- パーティションの開始点は、サイズが指定する正確なセクターに調整されます。
- 終了点は、指定されたサイズから 1 セクターを引いたサイズに調整されます。

MB、GB、または TB

サイズは 10 のべき乗で表されます。
開始点と終了点は、指定された単位の半分以内に置かれます。たとえば、MB サフィックスを使用する場合は ±500 KB です。

1.4.2. parted を使用したパーティションの削除

この手順では、**parted** ユーティリティーを使用してディスクパーティションを削除する方法を説明します。

手順

1. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

   - **block-device** を、パーティションを削除するデバイスへのパスに置き換えます。たとえば、**/dev/sda** に置き換えます。

2. 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。

```
(parted) print
```

3. パーティションを削除します。

```
(parted) rm minor-number
```

   - **minor-number** を削除するパーティションのマイナー番号に置き換えます (たとえば、**3**)。

   このコマンドを入力するとすぐに変更が行われるため、実行する前によく確認してください。

4. パーティションテーブルからパーティションが削除されたことを確認します。

```
(parted) print
```

5. **parted** シェルを終了します。

```
(parted) quit
```

6. パーティションが削除されたことをカーネルが認識していることを確認します。

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

7. パーティションが存在する場合は、**/etc/fstab** ファイルからパーティションを削除します。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。

8. システムが新しい **/etc/fstab** 設定を登録するようにマウントユニットを再生成します。

```
# systemctl daemon-reload
```

9. スワップパーティション、または LVM 部分を削除した場合は、**/etc/default/grub** ファイルのカーネルコマンドラインからパーティションへの参照をすべて削除して、GRUB 設定を再生成します。

   - BIOS ベースのシステムの場合:

```
# grub2-mkconfig --output=/etc/grub2.cfg
```

- UEFI ベースのシステムの場合:

```
# grub2-mkconfig --output=/etc/grub2-efi.cfg
```

10. アーリーブートシステムに変更を登録するには、**initramfs** ファイルシステムを再構築します。

```
# dracut --force --verbose
```

関連資料

- man ページの **parted(8)**

1.5. パーティションのサイズ変更

システム管理者は、パーティションを拡張して未使用のディスク容量を利用したり、パーティションを縮小して作成した容量をさまざまな目的に使用できます。

1.5.1. ディスク上のパーティション変更前の留意事項

このセクションでは、パーティションを作成、削除、またはサイズ変更する前に考慮すべき重要な点を説明します。

注記

このセクションでは、IBM System z アーキテクチャーに固有の DASD パーティションテーブルを説明しません。DASD は、IBM Knowledge Center の「What you should know about DASD」を参照してください。

パーティションの最大数
デバイスのパーティション数は、パーティションテーブルの種類により制限されます。

- マスターブートレコード **(MBR)** パーティションテーブルでフォーマットされたデバイスでは、次のいずれかの数だけパーティションを設定できます。
  - 最大 4 つのプライマリーパーティション
  - 最大 3 つのプライマリーパーティションおよび 1 つの拡張パーティション、ならびにその拡張内に複数の論理パーティション
- **GUID** パーティションテーブル **(GPT)** を使用してフォーマットされたデバイスのパーティションの最大数は 128 個になります。GPT 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できますが、**parted** ユーティリティーで用いられる一般的な方法で得られるエリアは、128 個に制限されます。

パーティションの最大サイズ
デバイスのパーティションのサイズは、パーティションテーブルの種類により制限されます。

- マスターブートレコード **(MBR)** パーティションテーブルでフォーマットされたデバイスの最大サイズは 2TiB になります。
- **GUID** パーティションテーブル **(GPT)** でフォーマットされたデバイスの最大サイズは 8ZiB になります。

2TiB を超えるパーティションを作成する場合は、ディスクを GPT でフォーマットする必要があります。

サイズ調整
parted ユーティリティーを使用した場合は、パーティションサイズを指定する際のサフィックスを選択できます。

MiB、GiB、または TiB

サイズは 2 のべき乗で表されます。

- パーティションの開始点は、サイズが指定する正確なセクターに調整されます。
- 終了点は、指定されたサイズから 1 セクターを引いたサイズに調整されます。

MB、GB、または TB

サイズは 10 のべき乗で表されます。
開始点と終了点は、指定された単位の半分以内に置かれます。たとえば、MB サフィックスを使用する場合は ±500 KB です。

1.5.2. parted を使用したパーティションのサイズ変更

この手順では、**parted** ユーティリティーを使用してディスクパーティションのサイズを変更する方法を説明します。

前提条件

- パーティションを縮小する場合は、そこに格納されているデータのバックアップを作成する。

警告

パーティションを縮小すると、パーティションのデータが失われる可能性があります。

- パーティションを 2TiB を超えるサイズに変更する場合は、ディスクを GUID パーティションテーブル (GPT) でフォーマットする必要があります。ディスクのフォーマット方法の詳細は、「ディスクへのパーティションテーブルの作成」 を参照してください。

手順

1. パーティションを縮小する場合は、サイズを変更したパーティションより大きくならないように最初にファイルシステムを縮小してください。XFS は縮小をサポートしていないことに注意してください。
2. インタラクティブな **parted** シェルを起動します。

```
# parted block-device
```

- **block-device** を、パーティションをサイズ変更するデバイスへのパスに置き換えます。たとえば、**/dev/sda** に置き換えます。

3. 現在のパーティションテーブルを表示します。

```
(parted) print
```

パーティションテーブルから、以下を確認します。

- パーティションのマイナー番号
- 既存のパーティションの位置とサイズ変更後の新しい終了点

4. パーティションのサイズを変更します。

```
(parted) resizepart minor-number new-end
```

- **minor-number** をサイズ変更するパーティションのマイナー番号に置き換えます (たとえば、**3**)。
- **new-end** をサイズ変更するパーティションの新しい終了点を決定するサイズに置き換えます (ディスクの開始からカウントします)。**512MiB**、**20GiB** または **1.5TiB** などのサイズサフィックスを使用できます。デフォルトサイズはメガバイトです。

例**1.4** パーティションの拡張

たとえば、ディスクの先頭にあるパーティションを 2GiB のサイズに拡張するには、次のコマンドを使用します。

```
(parted) resizepart 1 2GiB
```

このコマンドを入力するとすぐに変更が行われるため、実行する前によく確認してください。

5. パーティションテーブルを表示して、サイズ変更したパーティションのサイズが、パーティションテーブルで正しく表示されていることを確認します。

```
(parted) print
```

6. **parted** シェルを終了します。

```
(parted) quit
```

7. カーネルが新しいパーティションを認識していることを確認します。

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

8. パーティションを拡張した場合は、そこにあるファイルシステムも拡張します。詳細は (関連資料) を参照してください。

関連資料

- man ページの **parted(8)**

第2章 永続的な命名属性の概要

システム管理者は、永続的な命名属性を使用してストレージボリュームを参照し、再起動を行っても信頼できるストレージ設定を構築する必要があります。

2.1. 非永続的な命名属性のデメリット

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

従来、**/dev/sd(**メジャー番号**)(**マイナー番号**)** の形式の非永続的な名前は、ストレージデバイスを参照するために Linux 上で使用されます。メジャー番号とマイナー番号の範囲、および関連する **sd** 名は、検出されると各デバイスに割り当てられます。つまり、デバイスの検出順序が変わると、メジャー番号とマイナー番号の範囲、および関連する **sd** 名の関連付けが変わる可能性があります。

このような順序の変更は、以下の状況で発生する可能性があります。

- システム起動プロセスの並列化により、システム起動ごとに異なる順序でストレージデバイスが検出された場合。
- ディスクが起動しなかったり、SCSI コントローラーに応答しなかった場合。この場合、通常のデバイスプローブによって検出されません。ディスクはシステムにアクセスできなくなり、後続のデバイスは関連する次の **sd** 名が含まれる、メジャーおよびマイナー番号の範囲があります。たとえば、通常 **sdb** と呼ばれるディスクが検出されないと、**sdc** と呼ばれるディスクが **sdb** として代わりに表示されます。
- SCSI コントローラー (ホストバスアダプターまたは HBA) が初期化に失敗し、その HBA に接続されているすべてのディスクが検出されなかった場合。後続のプローブされた HBA に接続しているディスクは、別のメジャー番号およびマイナー番号の範囲、および関連する別の **sd** 名が割り当てられます。
- システムに異なるタイプの HBA が存在する場合、ドライバー初期化の順序が変更する可能性があります。これにより、これらの HBA に接続されているディスクが異なる順序で検出される可能性があります。また、HBA がシステムの他の PCI スロットに移動した場合でも発生する可能性があります。
- ストレージアレイや干渉するスイッチの電源が切れた場合など、ストレージデバイスがプローブされたときに、ファイバーチャネル、iSCSI、または FCoE アダプターを持つシステムに接続されたディスクがアクセスできなくなる可能性があります。システムが起動するまでの時間よりもストレージアレイがオンラインになるまでの時間の方が長い場合に、電源の障害後にシステムが再起動すると、この問題が発生する可能性があります。一部のファイバーチャネルドライバーは WWPN マッピングへの永続 SCSI ターゲット ID を指定するメカニズムをサポートしますが、メジャーおよびマイナー番号の範囲や関連する **sd** 名は予約されず、一貫性のある SCSI ターゲット ID 番号のみが提供されます。

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

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

2.2. ファイルシステムとデバイスの識別子

このセクションでは、ファイルシステムとブロックデバイスを識別する永続的な属性の違いを説明します。

ファイルシステムの識別子
ファイルシステムの識別子は、ブロックデバイス上に作成された特定のファイルシステムに関連付けられています。識別子はファイルシステムの一部としても格納されます。ファイルシステムを別のデバイスにコピーしても、ファイルシステム識別子は同じです。一方、**mkfs** ユーティリティーでフォーマットするなどしてデバイスを書き換えると、デバイスはその属性を失います。

ファイルシステムの識別子に含まれるものは、次のとおりです。

- 一意の ID (UUID)
- ラベル

デバイスの識別子
デバイス識別子は、ブロックデバイス (ディスクやパーティションなど) に関連付けられています。**mkfs** ユーティリティーでフォーマットするなどしてデバイスを書き換えた場合、デバイスはファイルシステムに格納されていないため、属性を保持します。

デバイスの識別子に含まれるものは、次のとおりです。

- World Wide Identifier (WWID)
- パーティション UUID
- シリアル番号

推奨情報

- 論理ボリュームなどの一部のファイルシステムは、複数のデバイスにまたがっています。Red Hat は、デバイスの識別子ではなくファイルシステムの識別子を使用してこれらのファイルシステムにアクセスすることをお勧めします。

2.3. /DEV/DISK/ にある UDEV メカニズムにより管理されるデバイス名

このセクションでは、**udev** サービスが **/dev/disk/** ディレクトリーで提供する、さまざまな種類の永続的な命名属性を説明します。

udev メカニズムは、ストレージデバイスだけでなく、Linux のすべてのタイプのデバイスに使用されます。ストレージデバイスの場合、Red Hat Enterprise Linux には **/dev/disk/** ディレクトリーにシンボリックリンクを作成する **udev** ルールが含まれています。これにより、次の方法でストレージデバイスを参照できます。

- ストレージデバイスのコンテンツ
- 一意の ID
- シリアル番号

udev 命名属性は永続的なものですが、システムを再起動しても自動的には変更されないため、設定可能なものもあります。

2.3.1. ファイルシステムの識別子

/dev/disk/by-uuid/ の UUID 属性

このディレクトリーのエントリーは、デバイスに格納されているコンテンツ (つまりデータ) 内の 一意の **ID** (UUID) によりストレージデバイスを参照するシンボリック名を提供します。たとえば、次のようになります。

```
/dev/disk/by-uuid/3e6be9de-8139-11d1-9106-a43f08d823a6
```

次の構文を使用することで、UUID を使用して **/etc/fstab** ファイルのデバイスを参照できます。

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

ファイルシステムを作成するときに UUID 属性を設定できます。また、後で変更することもできます。

/dev/disk/by-label/ のラベル属性

このディレクトリーのエントリーは、デバイスに格納されているコンテンツ (つまりデータ) 内のラベルによってストレージデバイスを参照するシンボリック名を提供します。

以下に例を示します。

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

次の構文を使用することで、ラベルを使用して **/etc/fstab** ファイルのデバイスを参照できます。

```
LABEL=Boot
```

ファイルシステムを作成するときにラベル属性を設定できます。また、後で変更することもできます。

2.3.2. デバイスの識別子

/dev/disk/by-id/ の WWID 属性

World Wide Identifier (WWID) は永続的で、SCSI 標準がすべての SCSI デバイスに必要とするシステムに依存しない識別子 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。この識別子はデバイスのプロパティですが、デバイスのコンテンツ (つまりデータ) には格納されません。

この識別子は、SCSI Inquiry を発行して Device Identification Vital Product Data (**0x83** ページ) または Unit Serial Number (**0x80** ページ) を取得することによって獲得できます。

Red Hat Enterprise Linux では、WWID ベースのデバイス名からそのシステム上の現在の **/dev/sd** 名への正しいマッピングを自動的に維持します。デバイスへのパスが変更したり、別のシステムからそのデバイスへのアクセスがあった場合にも、アプリケーションはディスク上のデータ参照に **/dev/disk/by-id/** を使用できます。

例2.1 WWID マッピング

WWID シンボリックリンク	非永続的なデバイス	備考
/dev/disk/by-id/scsi-3600508b400105e210000900000490000	**/dev/sda**	ページ **0x83** の識別子を持つデバイス
/dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6	**/dev/sdb**	ページ **0x80** の識別子を持つデバイス

WWID シンボリックリンク	非永続的なデバイス	備考
/dev/disk/by-id/ata-SAMSUNG_MZNLN256HMHQ-000L7_S2WDNX0J336519-part3	**/dev/sdc3**	ディスクパーティション

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

/dev/disk/by-partuuid のパーティション UUID 属性

パーティション UUID (PARTUUID) 属性は、GPT パーティションテーブルにより定義されているパーティションを識別します。

例2.2 パーティション **UUID** のマッピング

PARTUUID シンボリックリンク	非永続的なデバイス
/dev/disk/by-partuuid/4cd1448a-01	**/dev/sda1**
/dev/disk/by-partuuid/4cd1448a-02	**/dev/sda2**
/dev/disk/by-partuuid/4cd1448a-03	**/dev/sda3**

/dev/disk/by-path/ のパス属性

この属性は、デバイスへのアクセスに使用されるハードウェアパスがストレージデバイスを参照するシンボル名を提供します。

警告

パス属性は信頼性が低く、Red Hat では使用をお勧めしていません。

2.4. DM MULTIPATH を使用した WORLD WIDE IDENTIFIER

このセクションでは、Device Mapper Multipath 構成における World Wide Identifier (WWID) と非永続的なデバイス名のマッピングを説明します。

システムからデバイスへのパスが複数ある場合、DM Multipath はこれを検出するために WWID を使用します。その後、DM Multipath は **/dev/mapper/wwid** ディレクトリーに単一の「疑似デバイス」を表示します (**/dev/mapper/3600508b400105df70000e00000ac0000** など)。

コマンド **multipath -l** は、非永続的な識別子へのマッピングを示します。

- **Host:Channel:Target:LUN**

- **/dev/sd** name
- **major:minor** number

例2.3 マルチパス構成での **WWID** マッピング

multipath -l コマンドの出力例

```
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]
```

DM Multipath は、各 WWID ベースのデバイス名から、システムで対応する **/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。

DM Multipath の **user_friendly_names** 機能を使用すると、WWID は **/dev/mapper/mpathN** 形式の名前にマップされます。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。

2.5. UDEV デバイス命名規則の制約

udev 命名規則の制約の一部は次のとおりです。

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

2.6. 永続的な命名属性の一覧表示

この手順では、非永続的なストレージデバイスの永続命名属性を確認する方法を説明します。

手順

- UUID 属性とラベル属性を一覧表示するには、**lsblk** ユーティリティーを使用します。

```
$ lsblk --fs storage-device
```

以下に例を示します。

例2.4 ファイルシステムの **UUID** とラベルの表示

```
$ lsblk --fs /dev/sda1

NAME FSTYPE LABEL UUID                                 MOUNTPOINT
sda1 xfs    Boot  afa5d5e3-9050-48c3-acc1-bb30095f3dc4 /boot
```

- PARTUUID 属性を一覧表示するには、**--output +PARTUUID** オプションを指定して **lsblk** ユーティリティーを使用します。

```
$ lsblk --output +PARTUUID
```

以下に例を示します。

例2.5 パーティションの **PARTUUID** 属性の表示

```
$ lsblk --output +PARTUUID /dev/sda1

NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT PARTUUID
sda1   8:1    0  512M  0 part /boot      4cd1448a-01
```

- WWID 属性を一覧表示するには、**/dev/disk/by-id/** ディレクトリーのシンボリックリンクのターゲットを調べます。たとえば、次のようになります。

例2.6 システムにある全ストレージデバイスの **WWID** の表示

```
$ file /dev/disk/by-id/*

/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001:
symbolic link to ../../sda
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part1:
symbolic link to ../../sda1
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part2:
symbolic link to ../../sda2
/dev/disk/by-id/dm-name-rhel_rhel8-root:                                    symbolic link to
../../dm-0
/dev/disk/by-id/dm-name-rhel_rhel8-swap:                                     symbolic link
to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUDivKOz5ofkgFhP0RMFsNyySVihqEl2cWWbR7MjXJolD6g:
symbolic link to ../../dm-1
/dev/disk/by-id/dm-uuid-LVM-
QIWtEHtXGobe5bewlIUDivKOz5ofkgFhXqH2M45hD2H9nAf2qfWSrlRLhzfMyOKd:
```

```
symbolic link to ../../dm-0
/dev/disk/by-id/lvm-pv-uuid-atlr2Y-vuMo-ueoH-CpMG-4JuH-AhEF-wu4QQm:
symbolic link to ../../sda2
```

2.7. 永続的な命名属性の変更

この手順では、ファイルシステムの UUID またはラベルの永続的な命名属性を変更する方法を説明します。

注記

udev 属性の変更はバックグラウンドで行われ、時間がかかる場合があります。**udevadm settle** コマンドは変更が完全に登録されるまで待機します。これにより、次のコマンドが新しい属性を正しく利用できるようになります。

以下のコマンドでは、次の手順を行います。

- **new-uuid** を設定する UUID に置き換えます。たとえば、**1cdfbc07-1c90-4984-b5ec-f61943f5ea50** に置き換えます。**uuidgen** コマンドを使用して UUID を生成できます。
- **new-label** をラベルに置き換えます。たとえば、**backup_data** に置き換えます。

前提条件

- XFS ファイルシステムの属性を変更する場合は、アンマウントしておく。

手順

- **XFS** ファイルシステムの UUID またはラベル属性を変更するには、**xfs_admin** ユーティリティーを使用します。

```
# xfs_admin -U new-uuid -L new-label storage-device
# udevadm settle
```

- **ext4**、**ext3**、**ext2** ファイルシステムの UUID またはラベル属性を変更するには、**tune2fs** ユーティリティーを使用します。

```
# tune2fs -U new-uuid -L new-label storage-device
# udevadm settle
```

- スワップボリュームの UUID またはラベル属性を変更するには、**swaplabel** ユーティリティーを使用します。

```
# swaplabel --uuid new-uuid --label new-label swap-device
# udevadm settle
```