Red Hat Enterprise Linux 8

ファイルシステムの管理

Red Hat Enterprise Linux 8 でのファイルシステムの作成、変更、管理

Last Updated: 2019-05-10

Red Hat Enterprise Linux 8 ファイルシステムの管理

Red Hat Enterprise Linux 8 でのファイルシステムの作成、変更、管理

法律上の通知

概要

本書は、Red Hat Enterprise Linux 8 で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。

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 NFS 共有のマウント

システム管理者は、システムにリモート NFS 共有をマウントすると、共有データにアクセスできます。

1.1. NFS の概要

このセクションでは、NFS サービスの基本概念を説明します。

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

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に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。

1.2. サポートされている NFS バージョン

このセクションでは、Red Hat Enterprise Linux でサポートされている NFS のバージョンとその機能をリストで紹介します。

現在、Red Hat Enterprise Linux 8 は以下の NFS のメジャーバージョンをサポートしています。

- NFS バージョン 3 (NFSv3) は安全な非同期書き込みをサポートしており、以前の 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もサポートしているため、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。
- 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、アクセス制御リスト (ACL) をサポートし、ステートフルな操作を利用します。

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

デフォルトの NFS バージョン

Red Hat Enterprise Linux 8 のデフォルトの NFS バージョンは 4.2 です。NFS クライアントはデフォルトで NFSv4.2 を使用してマウントを試みます。サーバーが NFSv4.2 をサポートしていない場合は、NFSv4.1 にフォールバックします。マウントはその後 NFSv4.0 にフォールバックし、次に NFSv3 にフォールバックします。

マイナー NFS バージョンの機能

以下は Red Hat Enterprise Linux 8 における NFSv4.2 の機能です。

サーバー側コピー

NFS クライアントが **copy_file_range()** システムコールを使用してネットワークリソースを無駄にすることなく、データを効率的にコピーできるようにします。

スパースファイル

ファイルに 1 つ以上の**ホール**を持たせることができます。ホールとは、割り当てられていない、またはゼロのみで構成される未初期化データブロックです。NFSv4.2 の **lseek()** 操作は **seek_hole()** と **seek_data()** をサポートしています。これにより、アプリケーションはスパースファイルにホールの場所をマップできます。

スペースの予約

ストレージサーバーが空き領域を予約することを許可します。これにより、サーバーで領域が不足することがなくなります。NFSv4.2 は、スペースを予約するための **allocate()** 操作、スペースの予約を解除するための **deallocate()** 操作、およびファイル内のスペースの事前割り当てまたは割り当て解除を行う **fallocate()** 操作をサポートしています。

ラベル付き NFS

データアクセス権を強制し、NFS ファイルシステム上の個々のファイルに対して、クライアントとサーバー間の SELinux ラベルを有効にします。

レイアウトの機能強化

一部の Parallel NFS (pNFS) サーバーがより良いパフォーマンス統計を収集できるようにする **layoutstats()** 操作を提供します。

NFSv4.1 の機能は次のとおりです。

- ネットワークのパフォーマンスとセキュリティーを強化し、pNFS のクライアント側サポートも含みます。
- コールバックに個別の TCP 接続を必要としなくなりました。これにより、NAT やファイアウォールが干渉した場合など、クライアントと通信できない場合でも NFS サーバーは委任を許可できます。
- 応答が失われて、操作が 2 回送信された場合に特定の操作が不正確な結果を返すことがあるという以前の問題を防ぐために、1 回限りのセマンティクスを提供します (リブート操作を除く)。

1.3. NFS が必要とするサービス

このセクションでは、NFS サーバーの実行または NFS 共有のマウントに必要なシステムサービスをリストで紹介します。Red Hat Enterprise Linux はこれらのサービスを自動的に開始します。

Red Hat Enterprise Linux では、NFS ファイル共有を提供するのに、カーネルレベルのサポートとサービスのプロセスの組み合わせを使用します。NFS のすべてのバージョンは、クライアントとサーバー間の Remote Procedure Call (RPC) に依存します。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。

nfsd

共有 NFS ファイルシステムに対する要求を処理する NFS サーバー。

rpcbind

ローカルの RPC サービスからポート予約を受け取ると、これらのポートは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** サービスは、RPC サービスの要求に応答し、要求された RPC サービスへの接続のセットアップを行います。NFSv4 では、これは使用されません。

rpc.mountd

NFS サーバーは、このプロセスを使用して NFSv3 クライアントの **MOUNT** 要求を処理します。要求されている NFS 共有が現在 NFS サーバー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チェックします。マウントの要求が許可されると、**nfs-mountd** サービスは Success ステータスで応答し、この NFS 共有用の File-Handle を NFS クライアントに戻します。

rpc.nfsd

このプロセスでは、サーバーが公開している明示的な NFS のバージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的な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-server** サービスに対応します。

lockd

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

rpc.statd

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再起動すると、NFS クライアントに通知します。**rpc-statd** サービスは、**nfs-server** サービスにより自動的に起動されるため、ユーザー設定を必要としません。このプロセスは NFSv4 では使用されません。

rpc.rquotad

このプロセスは、リモートユーザーのユーザークォータ情報を提供します。**rpc-rquotad** サービスは **nfs-server** サービスにより自動的に起動するため、ユーザー設定を必要としません。

rpc.idmapd

このプロセスは、ネットワーク上の NFSv4 の名前 (**user@domain** 形式の文字列) とローカルの UID および GID とのマッピングを行う NFSv4 クライアントアップコールおよびサーバーアップコールを提供します。**idmapd** を NFSv4 で正常に動作させるには、**/etc/idmapd.conf** ファイルを設定する必要があります。最低でも NFSv4 マッピングドメインを定義する **Domain** パラメーターを指定する必要があります。NFSv4 マッピングドメインが DNS ドメイン名と同じであると、このパラメーターをスキップできます。クライアントとサーバーが ID マッピングの NFSv4 マッピングドメインに合意しないと、適切に動作しません。
rpc.idmapd を使用するのは NFSv4 サーバーだけで、 **nfs-idmapd** サービスにより起動されます。NFSv4 クライアントはキーリングベースの **nfsidmap** ユーティリティーを使用します。これは ID マッピングを実行するために、カーネルによりオンデマンドで呼び出されます。**nfsidmap** に問題がある場合、クライアントは **rpc.idmapd** の使用にフォールバックします。

NFSv4 を使用する RPC サービス

マウントとロックのプロトコルは NFSv4 プロトコルに組み込まれています。サーバーは、よく知られた TCP ポート 2049 もリッスンします。そのため、NFSv4 は **rpcbind**、**lockd**、および **rpc-statd** サービスと対話する必要はありません。**nfs-mountd** サービスは、エクスポートを設定するために NFS サーバー上で引き続き必要となりますが、ネットワーク上の操作には関係しません。

関連資料

- **rpcbind** を必要としない NFSv4 専用サーバーを設定するには、 「NFSv4 専用サーバーの設定」 を参照してください。

1.4. NFS ホスト名の形式

このセクションでは、NFS 共有をマウントまたはエクスポートするときにホストの指定に使用するさまざまな形式を説明します。

次の形式でホストを指定できます。

単独マシン

次のいずれかになります。

- 完全修飾ドメイン名 (これはサーバーにより解決されます)
- ホスト名 (これはサーバーにより解決されます)
- IP アドレス

ワイルドカードで指定された一連のマシン

文字列の一致を指定するには、* または ? 文字を使用します。
ワイルドカードは IP アドレスと一緒に使用しないでください。ただし、逆引き DNS ルックアップが失敗した場合は、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** が含まれますが、**one.two.example.com** は含まれません。

IP ネットワーク

以下のいずれかの形式が有効です。

- **a.b.c.d/z** (**a.b.c.d** がネットワークで **z** がネットマスクのビット数になります。たとえば **192.168.0.0/24**)
- **a.b.c.d/netmask** (**a.b.c.d** がネットワークで **netmask** がネットマスクになります。たとえば **192.168.100.8/255.255.255.0**)

Netgroup

@group-name 形式 (**group-name** は NIS netgroup 名です)

1.5. NFS のインストール

この手順では、NFS 共有のマウントまたはエクスポートに必要なすべてのパッケージをインストールします。

手順

- **nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

1.6. NFS エクスポートの検出

この手順では、特定の NFSv3 または NFSv4 サーバーがエクスポートしているファイルシステムを検出します。

手順

- NFSv3 をサポートしているサーバーの場合は、**showmount** ユーティリティーを使用してください。

```
$ showmount --exports my-server

Export list for my-server
/exports/foo
/exports/bar
```

- NFSv4 をサポートしているサーバーの場合は、root ディレクトリーをマウントして周囲を見て回ります。

```
# mount my-server:/ /mnt/
# ls /mnt/

exports
```

```
# ls /mnt/exports/

foo
bar
```

NFSv4 と NFSv3 の両方をサポートするサーバーでは、上記の方法はいずれも機能し、同じ結果を出します。

関連資料

- man ページの **showmount(8)**

1.7. MOUNT を使用した NFS 共有のマウント

この手順では、**mount** ユーティリティーを使用してサーバーからエクスポートされた NFS 共有をマウントします。

手順

- NFS 共有をマウントするには、次のコマンドを使用します。

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

このコマンドは以下のような変数を使用します。

options
: マウントオプションのカンマ区切りリスト

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

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

/local/directory
: **/remote/export** がマウントされたクライアントの場所

関連資料

- 「一般的な NFS マウントオプション」
- 「NFS ホスト名の形式」
- 「mount を使用したファイルシステムのマウント」
- man ページの **mount(8)**

1.8. 一般的な NFS マウントオプション

このセクションでは、NFS 共有をマウントするときに一般的に使用されるオプションを一覧で紹介します。これらのオプションは手動のマウントコマンド、**/etc/fstab** 設定、**autofs** とともに使用できます。

一般的な NFS マウントオプション

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 **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 **3**、**4**、**4.0**、**4.1**、または **4.2** になります。これは、複数の NFS サーバーを実行しているホストや、より低いバージョンでのマウントの再試行を無効化にするのに役立ちます。バージョンが指定されていない場合、NFS はカーネルと **mount** ユーティリティーでサポートされている最新バージョンを使用します。
オプション **vers** 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、非常に古いバージョンの NFS サーバーに接続する場合に必要となる場合があります。

noexec

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

nosuid

set-user-identifier および **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

NFS サーバーポートの数値を指定します。**num** が **0** (デフォルト値) の場合、**mount** は使用するポート番号について、リモートホストの **rpcbind** サービスに問い合わせます。リモートホストの NFS サービスがその **rpcbind** サービスに登録されていない場合は、代わりに TCP 2049 の標準 NFS ポート番号が使用されます。

rsize=num and **wsize=num**

これらのオプションは、単一の NFS 読み取りまたは書き込み操作で転送される最大バイト数を設定します。
rsize と **wsize** には固定のデフォルト値はありません。デフォルトでは、NFS はサーバーとクライアントの両方がサポートしている最大の値を使用します。Red Hat Enterprise Linux 8 では、クライアントとサーバーの最大値は 1,048,576 バイトです。詳細は、ナレッジベースの記事「What are the default and maximum values for rsize and wsize with NFS mounts?」を参照してください。

sec=mode

マウントされたエクスポート上のファイルにアクセスするために使用するセキュリティーフレーバーです。
デフォルト設定は **sec=sys** で、ローカルの UNIX UID および GID を使用します。 **AUTH_SYS** を使用して NFS 操作を認証します。

他のオプションは次のとおりです。

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

- **sec=krb5i** は、ユーザー認証に Kerberos V5 を使用し、データの改ざんを防ぐ安全なチェックサムを使用して、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。

関連資料

- man ページの **mount(8)**
- man ページの **nfs(5)**

1.9. 関連情報

- Linux NFS wiki - https://linux-nfs.org
- NFS 共有を永続的にマウントするには、「ファイルシステムの永続的なマウント」 を参照してください。
- NFS 共有をオンデマンドでマウントするには、「オンデマンドでのファイルシステムのマウント」 を参照してください。

第2章 NFS 共有のエクスポート

システム管理者は、NFS サーバーを使用してネットワーク上のシステムのディレクトリーを共有できます。

2.1. NFS の概要

このセクションでは、NFS サービスの基本概念を説明します。

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

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に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。

2.2. サポートされている NFS バージョン

このセクションでは、Red Hat Enterprise Linux でサポートされている NFS のバージョンとその機能をリストで紹介します。

現在、Red Hat Enterprise Linux 8 は以下の NFS のメジャーバージョンをサポートしています。

- NFS バージョン 3 (NFSv3) は安全な非同期書き込みをサポートしており、以前の 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もサポートしているため、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。
- 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、アクセス制御リスト (ACL) をサポートし、ステートフルな操作を利用します。

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

デフォルトの NFS バージョン

Red Hat Enterprise Linux 8 のデフォルトの NFS バージョンは 4.2 です。NFS クライアントはデフォルトで NFSv4.2 を使用してマウントを試みます。サーバーが NFSv4.2 をサポートしていない場合は、NFSv4.1 にフォールバックします。マウントはその後 NFSv4.0 にフォールバックし、次に NFSv3 にフォールバックします。

マイナー NFS バージョンの機能

以下は Red Hat Enterprise Linux 8 における NFSv4.2 の機能です。

サーバー側コピー

NFS クライアントが **copy_file_range()** システムコールを使用してネットワークリソースを無駄にすることなく、データを効率的にコピーできるようにします。

スパースファイル

ファイルに 1 つ以上の**ホール**を持たせることができます。ホールとは、割り当てられていない、またはゼロのみで構成される未初期化データブロックです。NFSv4.2 の **lseek()** 操作は **seek_hole()** と **seek_data()** をサポートしています。これにより、アプリケーションはスパースファイルにホールの場所をマップできます。

スペースの予約

ストレージサーバーが空き領域を予約することを許可します。これにより、サーバーで領域が不足することがなくなります。NFSv4.2 は、スペースを予約するための **allocate()** 操作、スペースの予約を解除するための **deallocate()** 操作、およびファイル内のスペースの事前割り当てまたは割り当て解除を行う **fallocate()** 操作をサポートしています。

ラベル付き NFS

データアクセス権を強制し、NFS ファイルシステム上の個々のファイルに対して、クライアントとサーバー間の SELinux ラベルを有効にします。

レイアウトの機能強化

一部の Parallel NFS (pNFS) サーバーがより良いパフォーマンス統計を収集できるようにする **layoutstats()** 操作を提供します。

NFSv4.1 の機能は次のとおりです。

- ネットワークのパフォーマンスとセキュリティーを強化し、pNFS のクライアント側サポートも含みます。
- コールバックに個別の TCP 接続を必要としなくなりました。これにより、NAT やファイアウォールが干渉した場合など、クライアントと通信できない場合でも NFS サーバーは委任を許可できます。
- 応答が失われて、操作が 2 回送信された場合に特定の操作が不正確な結果を返すことがあるという以前の問題を防ぐために、1 回限りのセマンティクスを提供します (リブート操作を除く)。

2.3. NFSV3 と NFSV4 の TCP と UDP プロトコル

NFSv4 は、IP ネットワークで TCP (Transmission Control Protocol) の実行が必要です。

NFSv3 は、Red Hat Enterprise Linux の以前のバージョンで User Datagram Protocol (UDP) を使用することもできます。Red Hat Enterprise Linux 8 では、NFS over UDP はサポートされなくなりました。デフォルトでは、UDP は、NFS サーバーで無効になります。

2.4. NFS が必要とするサービス

このセクションでは、NFS サーバーの実行または NFS 共有のマウントに必要なシステムサービスをリストで紹介します。Red Hat Enterprise Linux はこれらのサービスを自動的に開始します。

Red Hat Enterprise Linux では、NFS ファイル共有を提供するのに、カーネルレベルのサポートとサービスのプロセスの組み合わせを使用します。NFS のすべてのバージョンは、クライアントとサーバー間の Remote Procedure Call (RPC) に依存します。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。

nfsd

共有 NFS ファイルシステムに対する要求を処理する NFS サーバー。

rpcbind

ローカルの RPC サービスからポート予約を受け取ると、これらのポートは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** サービスは、RPC サービスの要求に応答し、要求された RPC サービスへの接続のセットアップを行います。NFSv4 では、これは使用されません。

rpc.mountd

NFS サーバーは、このプロセスを使用して NFSv3 クライアントの **MOUNT** 要求を処理します。要求されている NFS 共有が現在 NFS サーバー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チェックします。マウントの要求が許可される

と、**nfs-mountd** サービスは Success ステータスで応答し、この NFS 共有用の File-Handle を NFS クライアントに戻します。

rpc.nfsd

このプロセスでは、サーバーが公開している明示的な NFS のバージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的な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-server** サービスに対応します。

lockd

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

rpc.statd

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再起動すると、NFS クライアントに通知します。**rpc-statd** サービスは、**nfs-server** サービスにより自動的に起動されるため、ユーザー設定を必要としません。このプロセスは NFSv4 では使用されません。

rpc.rquotad

このプロセスは、リモートユーザーのユーザークォータ情報を提供します。**rpc-rquotad** サービスは **nfs-server** サービスにより自動的に起動するため、ユーザー設定を必要としません。

rpc.idmapd

このプロセスは、ネットワーク上の NFSv4 の名前 (**user@domain** 形式の文字列) とローカルの UID および GID とのマッピングを行う NFSv4 クライアントアップコールおよびサーバーアップコールを提供します。**idmapd** を NFSv4 で正常に動作させるには、**/etc/idmapd.conf** ファイルを設定する必要があります。最低でも NFSv4 マッピングドメインを定義する **Domain** パラメーターを指定する必要があります。NFSv4 マッピングドメインが DNS ドメイン名と同じであると、このパラメーターをスキップできます。クライアントとサーバーが ID マッピングの NFSv4 マッピングドメインに合意しないと、適切に動作しません。
rpc.idmapd を使用するのは NFSv4 サーバーだけで、 **nfs-idmapd** サービスにより起動されます。NFSv4 クライアントはキーリングベースの **nfsidmap** ユーティリティーを使用します。これは ID マッピングを実行するために、カーネルによりオンデマンドで呼び出されます。**nfsidmap** に問題がある場合、クライアントは **rpc.idmapd** の使用にフォールバックします。

NFSv4 を使用する RPC サービス

マウントとロックのプロトコルは NFSv4 プロトコルに組み込まれています。サーバーは、よく知られた TCP ポート 2049 もリッスンします。そのため、NFSv4 は **rpcbind**、**lockd**、および **rpc-statd** サービスと対話する必要はありません。**nfs-mountd** サービスは、エクスポートを設定するために NFS サーバー上で引き続き必要となりますが、ネットワーク上の操作には関係しません。

関連資料

- **rpcbind** を必要としない NFSv4 専用サーバーを設定するには、 「NFSv4 専用サーバーの設定」 を参照してください。

2.5. NFS ホスト名の形式

このセクションでは、NFS 共有をマウントまたはエクスポートするときにホストの指定に使用するさまざまな形式を説明します。

次の形式でホストを指定できます。

単独マシン

次のいずれかになります。

- 完全修飾ドメイン名 (これはサーバーにより解決されます)
- ホスト名 (これはサーバーにより解決されます)
- IP アドレス

ワイルドカードで指定された一連のマシン

文字列の一致を指定するには、***** または **?** 文字を使用します。
ワイルドカードは IP アドレスと一緒に使用しないでください。ただし、逆引き DNS ルックアップが失敗した場合は、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** が含まれますが、**one.two.example.com** は含まれません。

IP ネットワーク

以下のいずれかの形式が有効です。

- **a.b.c.d/z** (**a.b.c.d** がネットワークで **z** がネットマスクのビット数になります。たとえば **192.168.0.0/24**)
- **a.b.c.d/netmask** (**a.b.c.d** がネットワークで **netmask** がネットマスクになります。たとえば **192.168.100.8/255.255.255.0**)

Netgroup

@group-name 形式 (**group-name** は NIS netgroup 名です)

2.6. NFS サーバーの設定

このセクションでは、NFS サーバーでエクスポートを構成する 2 つの方法の構文とオプションを説明します。

- 設定ファイル **/etc/exports** を手動で編集する方法
- コマンドラインで **exportfs** ユーティリティーを使用する方法

2.6.1. /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 (\) で囲むことができる。
- 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、ホスト識別子の直後に括弧を追加し、その中に指定する。ホストと最初の括弧の間には空白を入れない。

エクスポートエントリー
エクスポートするファイルシステムのエントリーは、以下のように指定します。

```
export host(options)
```

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 行で指定することもできます。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)
```

この構造では、次のようになります。

export
エクスポートするディレクトリー

host
エクスポートを共有するホストまたはネットワーク

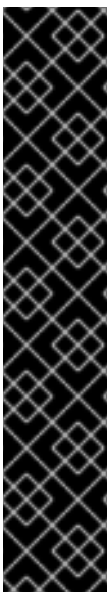
options
ホストに使用されるオプション

例2.1簡潔な /etc/exports ファイル

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。

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

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



重要

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

たとえば、以下の 2 つの行は意味が異なります。

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

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

デフォルトのオプション

エクスポートエントリーのデフォルトオプションは次のとおりです。

ro

エクスポートするファイルシステムは読み込み専用です。リモートホストは、このファイルシステムで共有されているデータを変更できません。このファイルシステムで変更 (読み込み、書き込み) を可能にするには、rw オプションを指定します。

sync

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

wdelay

NFS サーバーは、別の書き込み要求が差し迫っていると判断すると、ディスクへの書き込みを遅らせます。これにより、複数の書き込みコマンドが同じディスクにアクセスする回数を減らすことができるため、書き込みの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。これは、デフォルトの sync オプションが指定されている場合に限り利用可能になります。

root_squash

(ローカルからではなく) リモートから接続している root ユーザーが root 権限を持つことを阻止します。代わりに、そのユーザーには、NFS サーバーにより、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)、権限を持たずにリモートサーバーに書き込むのを阻止します。**no_root_squash** オプションと指定すると、この root squashing が無効になります。
(root を含む) すべてのリモートユーザーの権限を下げるには、**all_squash** オプションを使用します。特定ホストのリモートユーザーに対して、NFS サーバーが割り当てるユーザー ID とグループ ID を指定するには、**anonuid** と **anongid** オプションをそれぞれ以下のように使用します。

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

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

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 によりサポートされています。この機能を無効にするには、ファイルシステムをエクスポートする際に **no_acl** オプションを指定します。

デフォルトオプションと上書きオプション
エクスポートするファイルシステムのデフォルトはすべて、明示的に上書きする必要があります。たとえば、**rw** オプションを指定しないと、エクスポートするファイルシステムは読み込み専用として共有されます。以下は、**/etc/exports** の例になりますが、ここでは 2 つのデフォルトオプションを上書きしています。

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

この例では、**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です。

2.6.2. exportfs ユーティリティー

exportfs ユーティリティーを使用すると、root ユーザーは NFS サービスを再起動せずにディレクトリーを選択してエクスポートまたはアンエクスポートできます。適切なオプションが指定されると、**exportfs** ユーティリティーはエクスポートされたファイルシステムを **/var/lib/nfs/xtab** に書き込み

ます。ファイルシステムへのアクセス権を決定するときには、**nfs-mountd** サービスが **xtab** ファイルを参照するため、エクスポートされたファイルシステムのリストの変更は直ちに反映されます。

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

-r

/etc/exports に記載されるすべてのディレクトリーから、**/etc/lib/nfs/xtab** に新しいエクスポート一覧を作成することで、ディレクトリーがエクスポートされることになります。結果的にこのオプションが **/etc/exports** 内のいずれかの変更でエクスポート一覧をリフレッシュすることになります。

-a

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をエクスポートするかどうかを判断します。他のオプションが指定されていない場合、**exportfs** は **/etc/exports** で指定された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、多くの場合は、エクスポート用ファイルシステムの一覧に永続的に追加する前に、エクスポートされるファイルシステムをテストするために使用されます。

-i

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

-u

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

-v

詳細表示を意味します。**exportfs** コマンドを実行するときに表示されるエクスポート、または非エクスポートのファイルシステムの情報が、より詳細に表示されます。

exportfs ユーティリティーにオプションが渡されていない場合は、現在エクスポートされているファイルシステムのリストが表示されます。

関連資料

- ホスト名を指定するためのその他の方法は、「NFS ホスト名の形式」 を参照してください。
- エクスポートオプションの完全なリストは、man ページの **exports(5)** を参照してください。
- **exportfs** ユーティリティーの詳細は、man ページの **exportfs(8)** を参照してください。

2.7. NFS および RPCBIND

このセクションでは、NFSv3 で必要とされる **rpcbind** サービスの目的を説明します。

rpcbind サービスは、リモートプロシージャーコール (RPC) サービスを、それらのサービスがリッスンするポートにマッピングします。RPC のプロセスが開始すると、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およびそのプロセスが処理することが予想される RPC プログラ

ム番号が登録されます。クライアントシステムが、特定の RPC プログラム番号を使用して、サーバーの **rpcbind** との通信が行われると、**rpcbind** サービスによりクライアントが適切なポート番号にリダイレクトされ、要求されたサービスと通信できるようになります。

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

rpcbind のアクセス制御ルールは、すべての RPC ベース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。

関連資料

- アクセス制御ルールの正確な構文は、man ページの **rpc.mountd(8)** と **rpc.statd(8)** を参照してください。

2.8. NFS のインストール

この手順では、NFS 共有のマウントまたはエクスポートに必要なすべてのパッケージをインストールします。

手順

- **nfs-utils** パッケージをインストールします。

```
# yum install nfs-utils
```

2.9. NFS サーバーの起動

この手順では、NFS 共有をエクスポートするために必要な NFS サーバーの起動方法を説明します。

前提条件

- NFSv2 または NFSv3 接続をサポートしているサーバーの場合は、**rpcbind** サービスを実行している。**rpcbind** がアクティブであることを確認するには、次のコマンドを使用します。

```
$ systemctl status rpcbind
```

サービスが停止している場合は、起動して有効にします。

```
$ systemctl enable --now rpcbind
```

手順

- NFS サーバーを起動し、ブート時に自動的に起動するようにするには、次のコマンドを使用します。

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

関連資料

- **rpcbind** を必要としない NFSv4 専用サーバーを設定するには、「NFSv4 専用サーバーの設定」 を参照してください。

2.10. NFS と RPCBIND のトラブルシューティング

rpcbind サービス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** ユーティリティーを使用すると 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。

手順

1. **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、次のコマンドを使用します。

```
# rpcinfo -p
```

例2.2 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
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp  20048  mountd
    100005    1   tcp  20048  mountd
    100005    2   udp  20048  mountd
    100005    2   tcp  20048  mountd
    100005    3   udp  20048  mountd
    100005    3   tcp  20048  mountd
    100024    1   udp  37769  status
    100024    1   tcp  49349  status
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049  nfs_acl
    100021    1   udp  56691  nlockmgr
    100021    3   udp  56691  nlockmgr
    100021    4   udp  56691  nlockmgr
    100021    1   tcp  46193  nlockmgr
    100021    3   tcp  46193  nlockmgr
    100021    4   tcp  46193  nlockmgr
```

NFS サービスの 1 つが正しく起動しない場合、**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。

2. 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、サービスは **rpcbind** に正しく登録され、動作を開始します。

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

関連資料

- 詳しい情報と **rpcinfo** オプションの一覧は、man ページの **rpcinfo(8)** を参照してください。

- **rpcbind** を必要としない NFSv4 専用サーバーを設定するには、「NFSv4 専用サーバーの設定」 を参照してください。

2.11. ファイアウォールの内側で動作するように NFS サーバーの設定

NFS は **rpcbind** サービスを必要とします。これは RPC サービスのポートを動的に割り当て、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。この手順では、ファイアウォールの内側で機能するように NFS サーバーを設定する方法を説明します。

手順

1. クライアントがファイアウォールの内側で NFS 共有にアクセスできるようにするには、**/etc/nfs.conf** ファイルの **[mountd]** セクションで RPC サービスを実行するポートを設定します。

```
[mountd]

port=port-number
```

   これにより、**-p port-number** オプションが **rpc.mount** コマンドラインに追加されます (**rpc.mount -p port-number**)。

2. NFSv4.0 コールバックがファイアウォールを通過するように許可するには、**/proc/sys/fs/nfs/nfs_callback_tcpport** をセットして、サーバーがクライアント上のそのポートに接続できるようにします。
   この手順は、NFSv4.1 またはそれ以降には必要ありません。そして **mountd**、**statd**、および **lockd** の他のポート群は、純粋な NFSv4 環境では必要ありません。

3. RPC サービスの **nlockmgr** が使用するポートを指定するには、**/etc/modprobe.d/lockd.conf** ファイルで **nlm_tcpport** と **nlm_udpport** オプションのポート番号を設定します。

4. NFS サーバーを再起動します。

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

   NFS が起動しない場合は、**/var/log/messages** を確認してください。一般的に、すでに使用されているポート番号を指定した場合、NFS は起動しません。

5. 変更が反映されたことを確認します。

```
# rpcinfo -p
```

関連資料

- **rpcbind** を必要としない NFSv4 専用サーバーを設定するには、「NFSv4 専用サーバーの設定」 を参照してください。

2.12. ファイアウォールからの RPC クォータのエクスポート

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

手順

1. **rpc-rquotad** サービスを有効化および起動します。

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

注記

rpc-rquotad サービスが有効になっている場合、nfs-server サービスが起動した後に自動的に起動されます。

2. ファイアウォールの内側で、クォータ RPC サービスにアクセスできるようにするには、TCP (UDP が可能な場合は UDP) ポート 875 を開く必要があります。デフォルトのポート番号は **/etc/services** ファイルで定義します。
デフォルトのポート番号は、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-p port-number** を追加すると上書きできます。

3. デフォルトで、リモートホストはクォータのみを読み取ることができます。クライアントにクォータの設定を許可したい場合は、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-S** オプションを追加します。

4. **/etc/sysconfig/rpc-rquotad** ファイルでの変更が反映されるように **rpc-rquotad** を再起動します。

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

2.13. RDMA で NFS の有効化 (NFSORDMA)

Red Hat Enterprise Linux 8 では、RDMA に対応するハードウェアが存在すると、RDMA (remote direct memory access) サービスが自動的に有効になります。

手順

1. **rdma** と **rdma-core** パッケージをインストールします。

```
# yum install rdma rdma-core
```

2. NFSoRDMA **server** モジュールの自動ロードを有効にするには、**/etc/rdma/rdma.conf** 設定ファイルの新しい行に **SVCRDMA_LOAD=yes** オプションを追加します。
/etc/nfs.conf ファイルの **[nfsd]** セクションにある **rdma=20049** オプションで、NFSoRDMA サービスがクライアントをリッスンするポート番号を指定します。RFC 5667 規格では、RDMA を介して NFSv4 サービスを提供する場合、サーバーはポート **20049** をリッスンする必要があると規定されています。

   /etc/rdma/rdma.conf ファイルには、デフォルトで **XPRTRDMA_LOAD=yes** オプションを設定する行が含まれています。これは **rdma** サービスに NFSoRDMA **client** モジュールをロードするように要求します。

3. **nfs-server** サービスを再起動します。

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

関連資料

- RFC 5667 規格 - https://tools.ietf.org/html/rfc5667

2.14. NFSV4 専用サーバーの設定

NFS サーバー管理者は、NFSv4 のみをサポートするように NFS サーバーを設定できます。これにより、システム上で開いているポートの数と実行中のサービスの数が最小限に抑えられます。

2.14.1. NFSv4 専用サーバーの利点と欠点

このセクションでは、NFSv4 のみをサポートするように NFS サーバーを設定する利点と欠点を説明します。

デフォルトで、NFS サーバーは Red Hat Enterprise Linux 8 の NFSv2、NFSv3、および NFSv4 接続をサポートします。ただし、NFS バージョン 4.0 以降のみをサポートするように NFS を設定することもできます。NFSv4 では **rpcbind** サービスがネットワークをリッスンする必要がないため、これにより、システム上で開いているポートと実行中のサービスの数が最小限に抑えられます。

NFS サーバーが NFSv4 専用として設定されていると、NFSv2 または NFSv3 を使用して共有をマウントしようとするクライアントは、次のようなエラーでマウントに失敗します。

```
要求された NFS バージョンまたはトランスポートプロトコルがサポートされていません。
```

任意で、**RPCBIND**、**MOUNT**、および **NSM** プロトコル呼び出しのリッスンを無効にすることもできます。これは NFSv4 専用の場合は不要です。

これらの追加オプションを無効にすると、次のような影響があります。

- NFSv2 または NFSv3 を使用してサーバーから共有をマウントしようとするクライアントが応答しなくなります。
- NFS サーバー自体が NFSv2 および NFSv3 ファイルシステムをマウントできなくなります。

2.14.2. NFS および rpcbind

このセクションでは、NFSv3 で必要とされる **rpcbind** サービスの目的を説明します。

rpcbind サービスは、リモートプロシージャーコール (RPC) サービスを、それらのサービスがリッスンするポートにマッピングします。RPC のプロセスが開始すると、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およびそのプロセスが処理することが予想される RPC プログラム番号が登録されます。クライアントシステムが、特定の RPC プログラム番号を使用して、サーバーの **rpcbind** との通信が行われると、**rpcbind** サービスによりクライアントが適切なポート番号にリダイレクトされ、要求されたサービスと通信できるようになります。

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

rpcbind のアクセス制御ルールは、すべての RPC ベース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。

関連資料

- アクセス制御ルールの正確な構文は、man ページの **rpc.mountd(8)** と **rpc.statd(8)** を参照してください。

2.14.3. NFSv4 のみをサポートするように NFS サーバーの設定

この手順では、NFS サーバーを NFS バージョン 4.0 以降のみをサポートするように設定する方法を説明します。

手順

1. **/etc/nfs.conf** 設定ファイルの **[nfsd]** に次の行を追加して、NFSv2 および NFSv3 を無効にします。

```
[nfsd]

vers2=no
vers3=no
```

2. 任意で、**RPCBIND**、**MOUNT**、および **NSM** プロトコル呼び出しのリッスンを無効にします。これは NFSv4 専用の場合は不要です。関連するサービスを無効にします。

```
# systemctl mask --now rpc-statd.service rpcbind.service rpcbind.socket
```

3. NFS サーバーを再起動します。

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

変更は、NFS サーバーを起動または再起動するとすぐに反映されます。

2.14.4. NFSv4 専用の設定の確認

この手順では、**netstat** ユーティリティーを使用して、NFS サーバーが NFSv4 専用モードで設定されていることを確認する方法を説明します。

手順

- TCP および UDP プロトコルでリッスンしているサービスを一覧表示するには、**netstat** ユーティリティーを使用します。

```
# netstat --listening --tcp --udp
```

例2.3 NFSv4 専用サーバーの出力

以下は、NFSv4 専用サーバーでの **netstat** の出力例です。**RPCBIND**、**MOUNT**、**NSM** のリッスンも無効になります。**nfs** が唯一リッスンする NFS サービスとなります。

```
# netstat --listening --tcp --udp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:nfs             0.0.0.0:*               LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 [::]:nfs                [::]:*                  LISTEN
udp        0      0 localhost.localdl:bootpc 0.0.0.0:*
```

例2.4 NFSv4 専用サーバーを設定する前の出力

対照的に、NFSv4 専用サーバーを設定する前の **netstat** 出力には、**sunrpc** サービスと **mountd** サービスが含まれています。

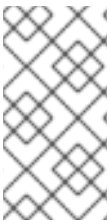
```
# netstat --listening --tcp --udp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:40189           0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:46813           0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:nfs             0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:sunrpc          0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:mountd          0.0.0.0:*       LISTEN
tcp6       0      0 [::]:ssh                [::]:*          LISTEN
tcp6       0      0 [::]:51227              [::]:*          LISTEN
tcp6       0      0 [::]:nfs                [::]:*          LISTEN
tcp6       0      0 [::]:sunrpc             [::]:*          LISTEN
tcp6       0      0 [::]:mountd             [::]:*          LISTEN
tcp6       0      0 [::]:45043              [::]:*          LISTEN
udp        0      0 localhost:1018          0.0.0.0:*
udp        0      0 localhost.locald:bootpc 0.0.0.0:*
udp        0      0 0.0.0.0:mountd          0.0.0.0:*
udp        0      0 0.0.0.0:46672           0.0.0.0:*
udp        0      0 0.0.0.0:sunrpc          0.0.0.0:*
udp        0      0 0.0.0.0:33494           0.0.0.0:*
udp6       0      0 [::]:33734              [::]:*
udp6       0      0 [::]:mountd             [::]:*
udp6       0      0 [::]:sunrpc             [::]:*
udp6       0      0 [::]:40243              [::]:*
```

2.15. 関連情報

- Linux NFS wiki - https://linux-nfs.org

第3章 RED HAT ENTERPRISE LINUX での SMB 共有のマウント

Server Message Block (SMB) プロトコルは、アプリケーションレイヤーネットワークプロトコルを実装します。これは、ファイル共有や共有プリンターなど、サーバー上のリソースにアクセスするために使用されます。



注記

SMB のコンテキストでは、SMB ダイアレクトである CIFS (Common Internet File System) プロトコルが言及されています。SMB と CIFS の両方のプロトコルがサポートされており、SMB と CIFS 共有のマウントに関連するカーネルモジュールとユーティリティーはどちらも **cifs** という名前を使用します。

このセクションでは、SMB サーバーから共有をマウントする方法を説明します。Samba を使用して Red Hat Enterprise Linux に SMB サーバーをセットアップする方法の詳細は、Configuring and deploying different types of servers ガイドの Samba の使用に関するセクションを参照してください。

3.1. 前提条件

cifs-utils パッケージがインストールされている (Microsoft Windows では、SMB がデフォルトで実装されているが、Red Hat Enterprise Linux では、カーネルの **cifs.ko** ファイルシステムモジュールが SMB 共有のマウントに対応しているため)。

```
# yum install cifs-utils
```

cifs-utils パッケージには以下を行うためのユーティリティーがあります。

- SMB と CIFS 共有をマウントする
- カーネルのキーリングで NT Lan Manager (NTLM) の認証情報を管理する
- SMB および CIFS 共有のセキュリティー記述子でアクセス制御リスト (ACL) を設定し、表示する

3.2. サポートされている SMB プロトコルバージョン

cifs.ko カーネルモジュールは、以下の SMB プロトコルバージョンをサポートします。

- SMB 1
- SMB 2.0
- SMB 2.1
- SMB 3.0

注記

プロトコルのバージョンにより、すべての SMB 機能が実装されているわけではありません。

3.3. UNIX 拡張機能のサポート

Samba は、SMB プロトコルの **CAP_UNIX** 機能ビットを使用して UNIX 拡張機能を提供します。これらの拡張機能は **cifs.ko** カーネルモジュールでもサポートされています。ただし、Samba とカーネルモジュールはどちらも、SMB 1 プロトコルでのみ UNIX 拡張機能をサポートしています。

UNIX 拡張機能を使用するには、以下の手順を実行します。

1. **/etc/samba/smb.conf** ファイルの **[global]** セクションにある **server min protocol** パラメーターを **NT1** に設定します。これは Samba サーバーのデフォルトです。
2. マウントコマンドに **-o vers=1.0** オプションを指定し、SMB 1 プロトコルを使用して共有をマウントします。たとえば、次のようになります。

   ```
   # mount -t cifs -o vers=1.0,username=user_name //server_name/share_name /mnt/
   ```

   デフォルトで、カーネルモジュールは SMB 2 またはサーバーでサポートされている最新のプロトコルバージョンを使用します。**-o vers=1.0** オプションを **mount** コマンドに渡すと、カーネルモジュールは UNIX 拡張機能の使用に必要な SMB 1 プロトコルを使用することを強制されます。

UNIX 拡張機能が有効になっているかどうかを確認するには、マウントされた共有のオプションを表示します。

```
# mount
...
//server/share on /mnt type cifs (...,unix,...)
```

マウントオプションの一覧に **unix** エントリーが表示されている場合は、UNIX 拡張機能が有効になっています。

3.4. SMB 共有の手動マウント

SMB 共有のみを一時的にマウントする必要がある場合は、**mount** ユーティリティーを使用して手動でマウントできます。

注記

手動でマウントされた共有は、システムを再起動しても自動的にはマウントされません。システムの起動時に Red Hat Enterprise Linux が自動的に共有をマウントするように設定するには、「システム起動時の SMB 共有の自動マウント」 を参照してください。

前提条件

- **cifs-utils** パッケージがインストールされている。

手順

SMB 共有を手動でマウントするには、**-t cifs** パラメーターを指定して**mount** ユーティリティーを使用します。

```
# mount -t cifs -o username=user_name //server_name/share_name /mnt/
Password for user_name@//server_name/share_name:  password
```

-o パラメーターでは、共有のマウントに使用されるオプションを指定できます。詳細は「よく使用されるマウントオプション」 と、man ページの **mount.cifs(8)** の **OPTIONS** セクションを参照してください。

例3.1暗号化された SMB 3.0 接続を使用した共有のマウント

暗号化された SMB 3.0 接続で **DOMAIN\Administrator** ユーザーとして **server\example**\ 共有を **/mnt/** ディレクトリーにマウントするには、次の手順を実行します。

```
# mount -t cifs -o username=DOMAIN\Administrator,seal,vers=3.0 //server/example /mnt/
Password for DOMAIN\Administrator@//server_name/share_name:  password
```

3.5. システム起動時の SMB 共有の自動マウント

マウントされた SMB 共有へのアクセスがサーバー上で恒久的に必要とされる場合は、起動時に共有を自動的にマウントします。

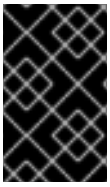
前提条件

- **cifs-utils** パッケージがインストールされている。

手順

システムの起動時に SMB 共有を自動的にマウントするには、共有のエントリーを **/etc/fstab** ファイルに追加します。たとえば、次のようになります。

```
//server_name/share_name  /mnt  cifs  credentials=/root/smb.cred  0 0
```



重要

システムが自動的に共有をマウントできるようにするには、ユーザー名、パスワード、およびドメイン名を認証情報ファイルに保存する必要があります。詳細は、「認証情報ファイルを使用した SMB 共有への認証」 を参照してください。

/etc/fstab の 4 行目のフィールドで、認証情報ファイルへのパスなど、マウントオプションを指定します。詳細は、「よく使用されるマウントオプション」 と、man ページの **mount.cifs(8)** の **OPTIONS** セクションを参照してください。

共有が正常にマウントされたことを確認するには、次のように入力します。

```
# mount /mnt/
```

3.6. 認証情報ファイルを使用した SMB 共有への認証

起動時に共有を自動的にマウントする場合など、特定の状況では、ユーザー名とパスワードを入力することなく共有がマウントされる必要があります。これを実装するには、認証情報ファイルを作成します。

前提条件

- **cifs-utils** パッケージがインストールされている。

手順

1. **/root/smb.cred** などのファイルを作成し、そのファイルのユーザー名、パスワード、およびドメイン名を指定します。

```
username=user_name
password=password
domain=domain_name
```

2. 所有者だけがファイルにアクセスできるようにパーミッションを設定します。

```
# chown user_name /root/smb.cred
# chmod 600 /root/smb.cred
```

mount ユーティリティーに **credentials=file_name** マウントオプションを渡すか、**/etc/fstab** ファイルでオプションを使用して、ユーザー名とパスワードの入力を求められずに共有をマウントできるようになります。

3.7. マルチユーザー SMB マウントの実行

共有をマウントするために指定した認証情報により、デフォルトでマウントポイントのアクセス権が決まります。たとえば、共有をマウントするときに **DOMAIN\example** ユーザーを使用した場合は、どのローカルユーザーが操作を実行しても、共有に対するすべての操作はこのユーザーとして実行されます。

ただし特定の状況では、システムの起動時に管理者が自動的に共有をマウントしたい場合でも、ユーザーは自分の認証情報を使用して共有のコンテンツに対して操作を実行する必要があります。そんなとき **multiuser** マウントオプションを使用すると、このシナリオを設定できます。

重要

multiuser マウントオプションを使用するには、認証情報ファイルの **krb5** や **ntlmssp** オプションなど、非対話式の方法で認証情報の提供をサポートするセキュリティータイプに **sec** マウントオプションを追加で設定する必要があります。詳細は、「ユーザーとして共有へのアクセス」 を参照してください。

root ユーザーは、**multiuser** オプションと共有の内容への最低限のアクセスを持つアカウントを使用して、共有をマウントします。通常のユーザーは、**cifscreds** ユーティリティーを使用して、現在のセッションのカーネルキーリングに自分のユーザー名とパスワードを渡すことができます。ユーザーがマウントされた共有のコンテンツにアクセスすると、カーネルは、共有のマウントに最初に使用されたものではなく、カーネルキーリングからの認証情報を使用します。

この機能の使用は以下の手順で構成されます。

- **multiuser** オプションを使用して共有をマウント
- 任意で、**multiuser** オプションを使用して共有が正常にマウントされたかを確認
- ユーザーとして共有にアクセス

3.7.1. 前提条件

- **cifs-utils** パッケージがインストールされている。

3.7.2. multiuser オプションを使用した共有のマウント

ユーザーが自分の認証情報を使用して共有にアクセスするには、パーミッションが制限されたアカウントを使用して **root** ユーザーとして共有をマウントします。

手順
システムの起動時に、**multiuser** オプションを使用して自動的に共有をマウントするには、次の手順を実行します。

1. **/etc/fstab** ファイルに共有のエントリーを作成します。たとえば、次のようになります。

```
//server_name/share_name  /mnt  cifs
multiuser,sec=ntlmssp,credentials=/root/smb.cred  0 0
```

2. 共有をマウントします。

```
# mount /mnt/
```

システムの起動時に共有を自動的にマウントしたくない場合は、**-o multiuser,sec=security_type** を **mount** コマンドに渡して手動で共有をマウントします。SMB 共有の手動マウントの詳細は、「SMB 共有の手動マウント」 を参照してください。

3.7.3. SMB 共有が multiuser オプションを使用してマウントされているかどうかの確認

共有が **multiuser** オプションを使用してマウントされているかどうかを確認するには、マウントオプションを表示します。

手順

```
# mount
...
//server_name/share_name on /mnt type cifs (sec=ntlmssp,multiuser,...)
```

マウントオプションのリストに **multiuser** エントリーが表示されている場合は、機能が有効になっています。

3.7.4. ユーザーとして共有へのアクセス

SMB 共有が **multiuser** オプションを使用してマウントされている場合、ユーザーはサーバーの認証情報をカーネルのキーリングに提供できます。

```
# cifscreds add -u SMB_user_name server_name
Password: password
```

ユーザーがマウントされた SMB 共有を含むディレクトリーで操作を実行すると、サーバーは、共有がマウントされたときに最初に使用されたものではなく、このユーザーのファイルシステムのパーミッションを適用します。

注記

複数のユーザーが、マウントされた共有上で自分の認証情報を使用して同時に操作を実行できます。

3.8. よく使用されるマウントオプション

SMB 共有をマウントすると、マウントオプションにより次のことが決まります。

- サーバーとの接続がどのように確立されるか。たとえば、サーバーに接続するときに使用される SMB プロトコルバージョンはどれか。
- 共有がローカルファイルシステムにどのようにマウントされるか。たとえば、複数のローカルユーザーがサーバー上のコンテンツにアクセスできるようにするために、システムがリモートファイルとディレクトリーのパーミッションを上書きする場合など。

/etc/fstab ファイルの 4 番目のフィールドまたはマウントコマンドの **-o** パラメーターで複数のオプションを設定するには、それらをカンマで区切ります。たとえば、「**multiuser** オプションを使用した共有のマウント」 を参照してください。

次のリストは、よく使用されるマウントオプションを示しています。

オプション	説明
credentials=**file_name**	認証情報ファイルへのパスを設定します。「認証情報ファイルを使用した SMB 共有への認証」 を参照してください。
dir_mode=**mode**	サーバーが CIFS UNIX 拡張機能をサポートしていない場合は、ディレクトリーモードを設定します。
file_mode=**mode**	サーバーが CIFS UNIX 拡張機能をサポートしていない場合は、ファイルモードを設定します。
password=**password**	SMB サーバーへの認証に使用されるパスワードを設定します。あるいは、**credentials** オプションを使用して認証情報ファイルを指定します。
seal	SMB 3.0 以降のプロトコルバージョンを使用した接続に対する暗号化サポートを有効にします。そのため、**seal** と **3.0** 以降に設定された **vers** マウントオプションを一緒に使用します。例3.1「暗号化された SMB 3.0 接続を使用した共有のマウント」 を参照してください。
sec=**security_mode**	**ntlmsspi** などのセキュリティーモードを設定して、NTLMv2 パスワードハッシュとパケット署名を有効にします。サポートされている値のリストは、man ページの **mount.cifs(8)** にあるオプションの説明を参照してください。 サーバーが **ntlmv2** セキュリティーモードをサポートしていない場合は、**sec=ntlmssp** (デフォルト) を使用します。 セキュリティー上の理由から、安全でない **ntlm** セキュリティーモードを使用しないでください。
username=**user_name**	SMB サーバーへの認証に使用されるユーザー名を設定します。あるいは、**credentials** オプションを使用して認証情報ファイルを指定します。
vers=**SMB_protocol_version**	サーバーとの通信に使用される SMB プロトコルバージョンを設定します。

完全なリストは、man ページの **mount.cifs(8)** の **OPTIONS** を参照してください。

第4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。

4.1. 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。この場合は、通常のデバイスプローブにより検出されません。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 (デバイスにより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。

4.2. 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。

4.3. /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。

4.3.1. 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。

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。

4.3.2. 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り獲得できます。

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4.1 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4.2 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。

4.4. 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

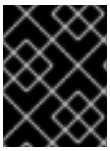
例4.3 マルチパス構成での WWID マッピング

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。

4.5. 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。

4.6. 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4.4 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4.5 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4.6 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
```

4.7. 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
```

第5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。

5.1. 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。

5.1.1. 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)**

5.1.2. 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5.1 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。

5.2. 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、ブロックデバイスでパーティションを使用できるように、さまざまな種類のパーティションテーブルを使用してそのブロックデバイスをフォーマットできます。

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。

5.2.1. 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。

5.2.2. 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5.1 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

5.2.3. 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5.2 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

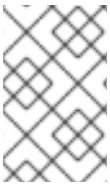
- デバイスにパーティションを作成します。詳細は「パーティションの作成」を参照してください。

5.3. 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。

5.3.1. 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。

5.3.2. 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**

5.3.3. 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5.3 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)**

5.3.4. 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
```

5.4. 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。

5.4.1. 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。

5.4.2. 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)**

5.5. 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。

5.5.1. 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。

5.5.2. 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5.4 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 (参考) を参照してください。

関連資料

- man ページの **parted(8)**

第6章 XFS の使用

これは、XFS ファイルシステムの作成およびメンテナンス方法の概要です。

6.1. XFS ファイルシステムの作成

システム管理者はブロックデバイス上に XFS ファイルシステムを作成して、ファイルやディレクトリーを格納できます。

6.1.1. mkfs.xfs を使用した XFS ファイルシステムの作成

この手順では、ブロックデバイス上に XFS ファイルシステムを作成する方法を説明します。

手順

1. ファイルシステムを作成するには、以下の手順を実行します。
   - デバイスが通常のパーティションである場合、LVM ボリューム、MD ボリューム、ディスク、または類似デバイスは次のコマンドを使用します。

     ```
     # mkfs.xfs block-device
     ```

     - **block-device** をブロックデバイスへのパスに置き換えます。たとえば、**/dev/sdb1**、**/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a** または **/dev/my-volgroup/my-lv** に置き換えます。
     - 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、そのファイルシステムを上書きする **-f** オプションを追加してください。
   - ハードウェア RAID デバイス上にファイルシステムを作成するには、システムがデバイスのストライプ配列を正しく検出しているかどうかを確認します。
     - ストライプ配列情報が正しい場合は、追加のオプションは必要ありません。ファイルシステムを作成します。

       ```
       # mkfs.xfs block-device
       ```

     - 情報が正しくない場合は、**-d** オプションの **su** パラメーターおよび **sw** パラメーターを使用して、ストライプ配列を手動で指定します。**su** パラメーターは RAID チャンクサイズを指定し、**sw** パラメーターは RAID デバイス内のデータディスクの数を指定します。
       以下に例を示します。

       ```
       # mkfs.xfs -d su=64k,sw=4 /dev/sda3
       ```

2. 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
   ```

関連資料

- man ページの **mkfs.xfs(8)**

6.1.2. 関連資料

- man ページの **mkfs.xfs(8)**

6.2. XFS ファイルシステムのバックアップ

システム管理者は、**xfsdump** を使用して XFS ファイルシステムをファイルまたはテープにバックアップできます。これは、簡単なバックアップシステムを提供します。

6.2.1. XFS バックアップの機能

このセクションでは、**xfsdump** ユーティリティーを使用して XFS ファイルシステムをバックアップする際の主な概念と機能を説明します。

xfsdump ユーティリティーを使用して次のことができます。

- 通常のファイルイメージへのバックアップ。
  通常のファイルに書き込むことができるバックアップは 1 つだけです。

- テープドライブへのバックアップ。
  xfsdump ユーティリティーを使用すると、同じテープに複数のバックアップを書き込むこともできます。バックアップは複数のテープをまたぐことができます。

  複数のファイルシステムを 1 つのテープデバイスにバックアップするには、XFS バックアップがすでに含まれているテープにバックアップを書き込みます。これにより、新しいバックアップが前のバックアップに追加されます。**xfsdump** は、デフォルトでは既存のバックアップを上書しません。

- 増分バックアップの作成。
  xfsdump ユーティリティーはダンプレベルを使用して、他のバックアップの相対的なベースバックアップを決定します。0 から 9 までの数字は、ダンプレベルの増加を表します。増分バックアップは、下位レベルの最後のダンプ以降に変更されたファイルのみをバックアップします。

  - フルバックアップを実行するには、ファイルシステムでレベル 0 のダンプを実行します。

  - レベル 1 のダンプは、フルバックアップ後の最初の増分バックアップです。次の増分バックアップはレベル 2 になります。これは、最後のレベル 1 のダンプ以降に変更されたファイルのみをバックアップします。レベル 9 まで同様です。

- ファイルを絞り込むサイズ、サブツリー、または i ノードフラグを使用してバックアップからファイルを除外。

関連資料

- man ページの **xfsdump(8)**

6.2.2. xfsdump を使用した XFS ファイルシステムのバックアップ

この手順では、XFS ファイルシステムの内容をファイルまたはテープにバックアップする方法を説明します。

前提条件

- バックアップできる XFS ファイルシステム。

- バックアップを保存できる別のファイルシステムまたはテープドライブ。

手順

- 次のコマンドを使用して XFS ファイルシステムをバックアップします。

```
# xfsdump -l level [-L label] \
        -f backup-destination path-to-xfs-filesystem
```

  - level を、バックアップのダンプレベルに置き換えます。フルバックアップを実行するには **0** を使用し、それに続く増分バックアップを実行するには **1** から **9** を使用します。
  - backup-destination を、バックアップを保存する場所のパスに置き換えます。宛先は通常のファイル、テープドライブ、またはリモートテープデバイスです。たとえば、ファイルの場合は **/backup-files/Data.xfsdump**、テープドライブの場合は **/dev/st0** に置き換えます。
  - path-to-xfs-filesystem をバックアップする XFS ファイルシステムのマウントポイントに置き換えます。たとえば、**/mnt/data/** と置き換えます。ファイルシステムをマウントする必要があります。
  - 複数のファイルシステムをバックアップして単一のテープデバイスに保存する場合は、復元時にそれらを簡単に識別できるように **-L label** オプションを使用して各バックアップにセッションラベルを追加します。label をバックアップの名前 (**backup_data** など) に置き換えます。

例6.1 複数の XFS ファイルシステムのバックアップ

- **/boot/** ディレクトリーおよび **/data/** ディレクトリーにマウントされている XFS ファイルシステムの内容をバックアップし、バックアップした内容をファイルとして **/backup-files/** ディレクトリーに保存するには、次の手順を実行します。

```
# xfsdump -l 0 -f /backup-files/boot.xfsdump /boot
# xfsdump -l 0 -f /backup-files/data.xfsdump /data
```

- 1 つのテープデバイスにある複数のファイルシステムのバックアップを作成するには、**-L label** オプションを使用して各バックアップにセッションラベルを追加します。

```
# xfsdump -l 0 -L "backup_boot" -f /dev/st0 /boot
# xfsdump -l 0 -L "backup_data" -f /dev/st0 /data
```

関連資料

- man ページの **xfsdump(8)**

6.2.3. 関連資料

- man ページの **xfsdump(8)**

6.3. バックアップからの XFS ファイルシステムの復元

システム管理者は、**xfsrestore** ユーティリティーを使用することで、**xfsdump** ユーティリティーで作成され、ファイルまたはテープに保存されている XFS バックアップを復元できます。

6.3.1. バックアップから XFS を復元する機能

このセクションでは、**xfsrestore** ユーティリティーを使用してバックアップから XFS ファイルシステムを復元する際の主な概念と機能を説明します。

xfsrestore ユーティリティーは、**xfsdump** により作成されたバックアッ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 2 つのモードがあります。

- **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。これがデフォルトのモードです。
- **cumulative** モードでは、増分バックアップ (つまりレベル 1 からレベル 9) からファイルシステムを復元できます。

各バックアップは、**session ID** または **session label** で一意に識別されます。複数のバックアップを含むテープからバックアップを復元するには、対応するセッション ID またはラベルが必要です。

バックアップから特定のファイルを抽出、追加、または削除するには、**xfsrestore** インタラクティブモードを起動します。インタラクティブモードでは、バックアップファイルを操作する一連のコマンドが提供されます。

関連資料

- man ページの **xfsrestore(8)**

6.3.2. xfsrestore を使用してバックアップから XFS ファイルシステムを復元

この手順では、XFS ファイルシステムの内容をファイルまたはテープのバックアップから復元する方法を説明します。

前提条件

- XFS ファイルシステムのファイルまたはテープのバックアップ。「XFS ファイルシステムのバックアップ」 を参照してください。
- バックアップを復元できるストレージデバイス。

手順

- バックアップを復元するコマンドは、フルバックアップから復元するのか、増分バックアップから復元するのか、または単一のテープデバイスから複数のバックアップを復元するのかによって異なります。

```
# xfsrestore [-r] [-S session-id] [-L session-label] [-i]
           -f backup-location restoration-path
```

  - **backup-location** をバックアップの場所に置き換えます。これは通常のファイル、テープドライブ、またはリモートテープデバイスになります。たとえば、ファイルの場合は **/backup-files/Data.xfsdump**、テープドライブの場合は **/dev/st0** に置き換えます。
  - **restoration-path** をファイルシステムを復元するディレクトリーへのパスに置き換えます。たとえば、**/mnt/data/** と置き換えます。
  - ファイルシステムを増分 (レベル 1 からレベル 9) バックアップから復元するには、**-r** オプションを追加します。

- 複数のバックアップを含むテープデバイスからバックアップを復元するには、**-S** または **-L** オプションを使用してバックアップを指定します。
  -S オプションではセッション ID でバックアップを選択でき、**-L** オプションではセッションラベルで選択できます。セッション ID とセッションラベルを取得するには、**xfsrestore -I** コマンドを使用します。

  session-id をバックアップのセッション ID に置き換えます。たとえば、**b74a3586-e52e-4a4a-8775-c3334fa8ea2c** と置き換えます。session-label をバックアップのセッションラベルに置き換えます。たとえば、**my_backup_session_label** と置き換えます。

- **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。
  インタラクティブダイアログは、**xfsrestore** が指定されたデバイスの読み込みを終了した後に始まります。インタラクティブな **xfsrestore** シェルの使用可能なコマンドには、**cd**、**ls**、**add**、**delete**、**extract** があります。コマンドの全リストを見るには、**help** コマンドを使用します。

例6.2 複数の XFS ファイルシステムの復元

- XFS バックアップファイルを復元し、その内容を **/mnt/** の下のディレクトリーに保存するには、次の手順を実行します。

```
# xfsrestore -f /backup-files/boot.xfsdump /mnt/boot/
# xfsrestore -f /backup-files/data.xfsdump /mnt/data/
```

- 複数のバックアップを含むテープデバイスから復元するには、各バックアップをセッションラベルまたはセッション ID で指定します。

```
# xfsrestore -L "backup_boot" -f /dev/st0 /mnt/boot/
# xfsrestore -S "45e9af35-efd2-4244-87bc-4762e476cbab" \
             -f /dev/st0 /mnt/data/
```

関連資料

- man ページの **xfsrestore(8)**

6.3.3. テープから XFS バックアップを復元するときの情報メッセージ

複数のファイルシステムのバックアップを使用してテープからバックアップを復元するとき、**xfsrestore** ユーティリティーがメッセージを出すことがあります。メッセージは、**xfsrestore** がテープ上の各バックアップを順番に調べたときに、要求されたバックアップと一致するものが見つかったかどうかを通知します。次に例を示します。

```
xfsrestore: preparing drive
xfsrestore: examining media file 0
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the
media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)
xfsrestore: examining media file 1
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the
media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)
[...]
```

一致するバックアップが見つかるまで、情報メッセージが表示され続けます。

6.3.4. 関連資料

- man ページの **xfsrestore(8)**

6.4. XFS ファイルシステムの修復

システム管理者は、破損した XFS ファイルシステムを修復できます。

6.4.1. XFS のエラー処理メカニズム

このセクションでは、XFS がファイルシステム内のさまざまな種類のエラーを処理する方法を説明します。

不完全なアンマウント
ジャーナリングは、ファイルシステムで発生したメタデータの変更のトランザクション記録を保持します。

システムクラッシュ、停電、またはその他の不完全なアンマウントが発生した場合、XFS はジャーナル (ログとも呼ばれる) を使用してファイルシステムを回復します。カーネルは XFS ファイルシステムをマウントするときにジャーナルの回復を実行します。

破損
この文脈では、**破損**は、次のような原因によるファイルシステムのエラーを意味します。

- ハードウェア障害
- ストレージファームウェア、デバイスドライバー、ソフトウェアスタック、またはファイルシステム自体のバグ
- ファイルシステムの一部がファイルシステム外の何かにより上書きされる問題

XFS は、ファイルシステムまたはファイルシステムのメタデータの破損を検出すると、ファイルシステムをシャットダウンして、システムログにインシデントを報告します。**/var** ディレクトリーをホストしているファイルシステムで破損が発生した場合、これらのログは再起動後に利用できなくなります。

例6.3 XFS の破損を報告するシステムログエントリー

```
# dmesg --notime | tail -15

XFS (loop0): Mounting V5 Filesystem
XFS (loop0): Metadata CRC error detected at xfs_agi_read_verify+0xcb/0xf0 [xfs], xfs_agi block 0x2
XFS (loop0): Unmount and run xfs_repair
XFS (loop0): First 128 bytes of corrupted metadata buffer:
00000000027b3b56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000005f9abc7a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000005b0aef35: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000da9d2ded: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000001e265b07: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000006a40df69: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000000b272907: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000e484aac5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
XFS (loop0): metadata I/O error in "xfs_trans_read_buf_map" at daddr 0x2 len 1 error 74
XFS (loop0): xfs_imap_lookup: xfs_ialloc_read_agi() returned error -117, agno 0
XFS (loop0): Failed to read root inode 0x80, error 11
```

ユーザースペースユーティリティーは通常、破損した XFS ファイルシステムにアクセスしようとすると **Input/output error** メッセージを報告します。破損したログを使用して XFS ファイルシステムをマウントすると、次のエラーメッセージが表示されます。

```
mount: /mount-point: mount(2) system call failed: Structure needs cleaning.
```

破損を修復するには、手動で **xfs_repair** ユーティリティーを使用する必要があります。他のファイルシステム修復ユーティリティーとは異なり、**xfs_repair** は 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動作しません。不完全なアンマウントが発生した場合、XFS はマウント時にログを単純に再生し、一貫したファイルシステムを確保します。**xfs_repair** は、最初にマウントし直さずに、ダーティーログを持つ XFS ファイルシステムを修復することはできません。

関連資料

- man ページの **xfs_repair(8)** に、XFS 破損チェックの詳細なリストがあります。

6.4.2. xfs_repair を使用した XFS ファイルシステムの修復

この手順では、**xfs_repair** ユーティリティーを使用して破損した XFS ファイルシステムを修復します。

手順

1. ファイルシステムを再マウントしてログを消去します。

```
# mount file-system
# umount file-system
```

2. アンマウントされたファイルシステムを修復するには、**xfs_repair** ユーティリティーを使用します。
   - マウントが成功した場合、追加のオプションは必要ありません。

```
# xfs_repair block-device
```

   - マウントが **Structure needs cleaning** エラーで失敗した場合、ログは破損しているため再生できません。ログを消去するには、**-L** オプション (**force log zeroing**) を使用します。

警告

このコマンドを実行すると、クラッシュ時に進行中だったすべてのメタデータの更新が失われます。これにより、ファイルシステムに重大な損傷やデータ損失が生じる可能性があります。これは最後の手段としてのみ使うべきです。

```
# xfs_repair -L block-device
```

3. ファイルシステムをマウントします。

```
# mount file-system
```

関連資料

- man ページの **xfs_repair(8)**

6.5. XFS ファイルシステムのサイズの拡大

システム管理者は、XFS ファイルシステムのサイズを大きくして、より大きな記憶容量を利用できます。

重要

現在 XFS ファイルシステムのサイズを縮小することはできません。

6.5.1. xfs_growfs を使用した XFS ファイルシステムのサイズの拡大

この手順では、**xfs_growfs** ユーティリティーを使用して XFS ファイルシステムを拡張する方法を説明します。

前提条件

- 基礎となるブロックデバイスが後でサイズ変更されるファイルシステムを保持するのに十分なサイズがある。該当するブロックデバイスのサイズを変更する場合は、ブロックデバイスに適した方法を使用してサイズ変更を行ってください。
- XFS ファイルシステムをマウントしている。

手順

- XFS ファイルシステムのマウント時に、**xfs_growfs** ユーティリティーを使用してサイズを大きくします。

  ```
  # xfs_growfs file-system -D new-size
  ```

  - **file-system** を XFS ファイルシステムのマウントポイントに置き換えます。
  - **-D** オプションを指定して、**new-size** を、ファイルシステムブロックの数で指定されているファイルシステムの新しいサイズに置き換えます。
    特定の XFS ファイルシステムのブロックサイズ (KB 単位) を調べるには、**xfs_info** ユーティリティーを使用します。

    ```
    # xfs_info block-device

    ...
    data     =               bsize=4096
    ...
    ```

  - **-D** オプションを指定しない場合、**xfs_growfs** はファイルシステムを基礎となるデバイスがサポートする最大サイズまで拡張します。

関連資料

- man ページの **xfs_growfs(8)**

第7章 ファイルシステムのマウント

システム管理者は、システムにファイルシステムをマウントすると、それらのデータにアクセスできます。

7.1. LINUX のマウントメカニズム

このセクションでは、Linux でのファイルシステムのマウントに関する基本概念を説明します。

Linux、UNIX、および類似のオペレーティングシステムでは、異なるパーティションおよびリムーバブルデバイス (CD、DVD、または 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) に接続してから切り離すことができます。ファイルシステムがディレクトリーにマウントされている間は、そのディレクトリーの元の内容にアクセスすることはできません。

Linux では、ファイルシステムがすでに接続されているディレクトリーにファイルシステムをマウントできます。

マウント時には、次の方法でデバイスを識別できます。

- 全世界で一意の ID (UUID) - たとえば、**UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb**
- ボリュームラベル - たとえば、**LABEL=home**
- 非永続的なブロックデバイスへのフルパス - たとえば、**/dev/sda3**

デバイス名、目的のディレクトリー、ファイルシステムタイプなど、必要な情報をすべて指定せずに **mount** コマンドを使用してファイルシステムをマウントすると、**mount** ユーティリティーは **/etc/fstab** ファイルの内容を読み取り、指定のファイルシステムが記載されているかどうかを確認します。**/etc/fstab** ファイルには、選択したファイルシステムがマウントされるデバイス名およびディレクトリーのリスト、ファイルシステムタイプ、およびマウントオプションが含まれます。そのため、**/etc/fstab** で指定されたファイルシステムをマウントする場合は、以下のコマンド構文で十分です。

- マウントポイントによるマウント:

```
# mount directory
```

- ブロックデバイスによるマウント:

```
# mount device
```

関連資料

- man ページの **mount(8)**
- UUID などの永続的な命名属性を一覧表示する方法は、「永続的な命名属性の一覧表示」 を参照してください。

7.2. 現在マウントされているファイルシステムの一覧表示

この手順では、コマンドラインに、現在マウントされているファイルシステムの一覧を表示する方法を説明します。

手順

- マウントされているファイルシステムの一覧を表示するには、**findmnt** ユーティリティーを使用します。

```
$ findmnt
```

- 一覧表示されているファイルシステムを特定のファイルシステムタイプのみに制限するには、**--types** オプションを追加します。

```
$ findmnt --types fs-type
```

以下に例を示します。

例7.1 XFS ファイルシステムのみの一覧表示

```
$ findmnt --types xfs

TARGET  SOURCE                                          FSTYPE OPTIONS
/       /dev/mapper/luks-5564ed00-6aac-4406-bfb4-c59bf5de48b5 xfs    rw,relatime
├─/boot /dev/sda1                                       xfs    rw,relatime
└─/home /dev/mapper/luks-9d185660-7537-414d-b727-d92ea036051e xfs    rw,relatime
```

関連資料

- man ページの **findmnt(8)**

7.3. MOUNT を使用したファイルシステムのマウント

この手順では、**mount** ユーティリティーを使用してファイルシステムをマウントする方法を説明します。

前提条件

- 選択したマウントポイントにファイルシステムがマウントされていない。

```
$ findmnt mount-point
```

手順

1. 特定のファイルシステムを添付するには、**mount** ユーティリティーを使用します。

```
# mount device mount-point
```

例7.2 XFS ファイルシステムのマウント

たとえば、UUID により識別されるローカル XFS ファイルシステムをマウントするには、次のように入力します。

```
# mount UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /mnt/data
```

2. **mount** がファイルシステムタイプを自動的に認識できない場合は、**--types** オプションで指定します。

```
# mount --types type device mount-point
```

例7.3 NFS ファイルシステムのマウント

たとえば、リモート NFS ファイルシステムをマウントするには、次のように入力します。

```
# mount --types nfs4 host:/remote-export /mnt/nfs
```

関連資料

- man ページの **mount(8)**

7.4. マウントポイントの移動

この手順では、マウントされたファイルシステムのマウントポイントを別のディレクトリーに変更する方法を説明します。

手順

1. 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、以下を使用します。

```
# mount --move old-directory new-directory
```

例7.4 ホームファイルシステムの移動

たとえば、**/mnt/userdirs/** ディレクトリーにマウントされたファイルシステムを **/home/** マウントポイントに移動するには、以下のように入力します。

```
# mount --move /mnt/userdirs /home
```

2. ファイルシステムが想定どおりに移動したことを確認します。

```
$ findmnt
$ ls old-directory
$ ls new-directory
```

関連資料

- man ページの **mount(8)**

7.5. UMOUNT を使用したファイルシステムのアンマウント

この手順では、**umount** ユーティリティーを使用してファイルシステムをアンマウントする方法を説明します。

手順

1. 次のいずれかのコマンドを使用してファイルシステムをアンマウントします。
   - マウントポイントにより行う場合:

```
# umount mount-point
```

- デバイスにより行う場合:

```
# umount device
```

コマンドが次のようなエラーで失敗した場合は、プロセスがリソースを使用しているためにファイルシステムが使用中であることを意味します。

```
umount: /run/media/user/FlashDrive: target is busy.
```

2. ファイルシステムが使用中の場合は、**fuser** ユーティリティーを使用して、ファイルシステムにアクセスしているプロセスを特定します。たとえば、次のようになります。

```
$ fuser --mount /run/media/user/FlashDrive

/run/media/user/FlashDrive: 18351
```

その後、ファイルシステムを使用してプロセスを終了し、再度アンマウントを試みます。

7.6. 一般的なマウントオプション

このセクションでは、**mount** ユーティリティーでよく使われるオプションを示します。

次の構文でこれらのオプションを使用できます。

```
# mount --options option1,option2,option3 device mount-point
```

表7.1 一般的なマウントオプション

オプション	説明
async	ファイルシステム上での非同期の入出力を可能にします。
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可能にします。
defaults	**async,auto,dev,exec,nouser,rw,suid** オプション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	デフォルトの動作として、**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無効します。
noexec	特定のファイルシステムでのバイナリーファイルの実行を拒否します。

オプション	説明
nouser	普通のユーザー (つまり root 以外のユーザー) によるファイルシステムのマウントおよびアンマウントを拒否します。
remount	ファイルシステムがすでにマウントされている場合は再度マウントを行います。
ro	読み取り専用でファイルシステムをマウントします。
rw	ファイルシステムを読み取りと書き込み両方でマウントします。
user	普通のユーザー (つまり root 以外のユーザー) によるファイルシステムのマウントおよびアンマウントを許可します。

7.7. 複数のマウントポイントでのマウント共有

システム管理者は、マウントポイントを複製して、ファイルシステムに複数のディレクトリーからアクセスできます。

7.7.1. 共有マウントのタイプ

使用できる共有マウントには複数のタイプがあります。種類によって、共有マウントポイントの 1 つに別のファイルシステムをマウントしたときに起こることが異なります。共有マウントは **shared subtrees** 機能を使用して実装されます。

タイプは以下のとおりです。

プライベートマウント

このタイプは伝播イベントを受信または転送しません。
複製マウントポイントまたは元のマウントポイントのどちらかに別のファイルシステムをマウントしても、それは他方には反映されません。

共有マウント

このタイプは与えられたマウントポイントの正確なレプリカを作成します。
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。

これは、root ファイルシステムのデフォルトのマウントタイプです。

スレーブマウント

このタイプは、指定したマウントポイントの限定的な複製を作成します。
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。

バインド不可能なマウント

このタイプは、指定のマウントポイントの複製をまったく行いません。

7.7.2. プライベートマウントポイントの複製の作成

この手順では、マウントポイントをプライベートマウントとして複製します。後で複製または元のマウントポイントにマウントしたファイルシステムは、他のマウントポイントには反映されません。

手順

1. 元のマウントポイントから仮想ファイルシステム (VFS) ノードを作成します。

```
# mount --bind original-dir original-dir
```

2. 元のマウントポイントをプライベートとしてマークします。

```
# mount --make-private original-dir
```

あるいは、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には、**--make-private** ではなく **--make-rprivate** オプションを使用します。

3. 複製を作成します。

```
# mount --bind original-dir duplicate-dir
```

例7.5 プライベートマウントポイントとして /mnt に /media を複製

1. **/media** ディレクトリーから VFS ノードを作成します。

```
# mount --bind /media /media
```

2. **/media** ディレクトリーをプライベートとしてマークします。

```
# mount --make-private /media
```

3. そのコピーを **/mnt** に作成します。

```
# mount --bind /media /mnt
```

4. これで、**/media** と **/mnt** はコンテンツを共有しているが、**/media** 内のマウントはいずれも **/mnt** に現れていないことを確認できます。たとえば、CD-ROM ドライブに空でないメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は、以下を使用します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
#
```

5. 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をプラグインしており、**/mnt/flashdisk/** ディレクトリーが存在する場合は、以下を使用します。

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US  publican.cfg
```

関連資料

- man ページの **mount(8)**

7.7.3. 共有マウントポイントの複製の作成

この手順では、マウントポイントを共有マウントとして複製します。後で元のディレクトリーまたは複製にマウントしたファイルシステムは、他のマウントポイントに常に反映されます。

手順

1. 元のマウントポイントから仮想ファイルシステム (VFS) ノードを作成します。

```
# mount --bind original-dir original-dir
```

2. 元のマウントポイントを共有としてマークします。

```
# mount --make-shared original-dir
```

あるいは、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には、**--make-shared** ではなく **--make-rshared** オプションを使用します。

3. 複製を作成します。

```
# mount --bind original-dir duplicate-dir
```

例7.6 共有マウントポイントとして /mnt に /media を複製

/media ディレクトリーと **/mnt** ディレクトリーが同じコンテンツを共有するようにするには、次の手順を行います。

1. **/media** ディレクトリーから VFS ノードを作成します。

```
# mount --bind /media /media
```

2. **/media** ディレクトリーを共有としてマークします。

```
# mount --make-shared /media
```

3. そのコピーを **/mnt** に作成します。

```
# mount --bind /media /mnt
```

4. 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 ドライブに空でないメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は、以下を使用します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

5. 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をプラグインしており、**/mnt/flashdisk/** ディレクトリーが存在する場合は、以下を使用します。

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
en-US  publican.cfg
# ls /mnt/flashdisk
en-US  publican.cfg
```

関連資料

- man ページの **mount(8)**

7.7.4. スレーブマウントポイントの複製の作成

この手順では、マウントポイントをスレーブマウントとして複製します。後で元のマウントポイントにマウントしたファイルシステムは複製に反映されますが、他のマウントポイントには反映されません。

手順

1. 元のマウントポイントから仮想ファイルシステム (VFS) ノードを作成します。

```
# mount --bind original-dir original-dir
```

2. 元のマウントポイントを共有としてマークします。

```
# mount --make-shared original-dir
```

あるいは、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には、**--make-shared** ではなく **--make-rshared** オプションを使用します。

3. 複製を作成し、それをスレーブとしてマークします。

```
# mount --bind original-dir duplicate-dir
# mount --make-slave duplicate-dir
```

例7.7 スレーブマウントポイントとして /mnt に /media を複製

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。

1. **/media** ディレクトリーから VFS ノードを作成します。

```
# mount --bind /media /media
```

2. **/media** ディレクトリーを共有としてマークします。

```
# mount --make-shared /media
```

3. そのコピーを **/mnt** に作成し、スレーブとしてマークします。

```
# mount --bind /media /mnt
# mount --make-slave /mnt
```

4. **/media** 内のマウントが **/mnt** にも表示されていることを確認します。たとえば、CD-ROM ドライブに空でないメディアがあり、**/media/cdrom/** ディレクトリーが存在する場合は、以下を使用します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

5. 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をプラグインしており、**/mnt/flashdisk/** ディレクトリーが存在する場合は、以下を使用します。

```
# mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
# ls /mnt/flashdisk
en-US  publican.cfg
```

関連資料

- man ページの **mount(8)**

7.7.5. マウントポイントが複製されないようにする

この手順では、別のマウントポイントに複製されないようにマウントポイントをバインド不可能としてマークします。

手順

- マウントポイントのタイプをバインド不可能なマウントに変更するには、以下を使用します。

```
# mount --bind mount-point mount-point
# mount --make-unbindable mount-point
```

あるいは、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には、**--make-unbindable** ではなく **--make-runbindable** オプションを使用します。

これ以降にこのマウントの複製を作成しようとすると、以下のエラーが出て失敗します。

```
# mount --bind mount-point duplicate-dir

mount: wrong fs type, bad option, bad superblock on mount-point,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail  or so
```

例7.8 /media が複製されないようにする

- **/media** ディレクトリーが共有されないようにするには、以下を使用します。

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

関連資料

- man ページの **mount(8)**

7.7.6. 関連情報

- Linux Weekly News の **共有サブツリー** の記事 - https://lwn.net/Articles/159077/

7.8. ファイルシステムの永続的なマウント

システム管理者は、ファイルシステムを永続的にマウントして、非リムーバブルストレージを構成できます。

7.8.1. /etc/fstab ファイル

このセクションでは、ファイルシステムの永続的なマウントポイントを制御する **/etc/fstab** 設定ファイルを説明します。ファイルシステムを永続的にマウントするには、**/etc/fstab** を使用することをお勧めします。

/etc/fstab ファイルの各行は、ファイルシステムのマウントポイントを定義します。空白で区切られた 6 つのフィールドが含まれています。

1. **/dev** ディレクトリーの永続的な属性またはパスで識別されるブロックデバイス。
2. デバイスがマウントされるディレクトリー。
3. デバイス上のファイルシステム。
4. ファイルシステムのマウントオプション。オプション **defaults** は、パーティションが起動時にデフォルトのオプションでマウントされることを意味します。このセクションでは、**x-systemd.option** 形式の **systemd** マウントユニットオプションも取り上げます。
5. **dump** ユーティリティーのオプションをバックアップします。
6. **fsck** ユーティリティーの順序を確認します。

例7.9 /etc/fstab の /boot ファイルシステム

ブロックデバイス	マウントポイント	ファイルシステム	オプション	バックアップ	チェック
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b	**/boot**	**xfs**	**defaults**	**0**	**0**

systemd サービスは **/etc/fstab** のエントリーからマウントユニットを自動的に生成します。

関連資料

- man ページの **fstab(5)**
- man ページの **systemd.mount(5)** の fstab セクション

7.8.2. /etc/fstab へのファイルシステムの追加

この手順では、**/etc/fstab** 設定ファイルでファイルシステムの固定マウントポイントを設定する方法を説明します。

手順

1. ファイルシステムの UUID 属性を調べます。

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

以下に例を示します。

例7.10 パーティションの UUID の表示

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

NAME FSTYPE LABEL UUID                                 MOUNTPOINT
sda1 xfs    Boot  ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot
```

2. このマウントポイントのディレクトリーがない場合は、作成します。

```
# mkdir --parents mount-point
```

3. root として、**/etc/fstab** ファイルを編集し、UUID で識別されるファイルシステムの行を追加します。
以下に例を示します。

例7.11 /etc/fstab の /boot マウントポイント

```
UUID=ea74bbec-536d-490c-b8d9-5b40bbd7545b /boot xfs defaults 0 0
```

4. システムが新しい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
```

5. ファイルシステムをマウントして、設定が機能することを確認します。

```
# mount mount-point
```

関連資料

- ファイルシステムを識別するために使用できるその他の永続的な属性 - 「/dev/disk/ にある udev メカニズムにより管理されるデバイス名」

7.9. オンデマンドでのファイルシステムのマウント

システム管理者は、NFS などのファイルシステムをオンデマンドで自動的にマウントするように設定できます。

7.9.1. autofs サービス

このセクションでは、ファイルシステムをオンデマンドでマウントするために使用される **autofs** サービスの利点と基本概念を説明します。

/etc/fstab 設定を使用した恒久的なマウントの欠点の 1 つは、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が低くても、システムが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リソースを割り当てる必要があることです。これは、システムが一度に多数のシステムへの NFS マウントを維持している場合などに、システム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 **autofs** サービスの使用です。これは以下のコンポーネントで構成されています。

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

autofs サービスはファイルシステムの自動マウントおよび自動アンマウントが可能なため (オンデマンド)、システムのリソースを節約できます。このサービスは、NFS、AFS、SMBFS、CIFS、およびローカルのファイルシステムなどのファイルシステムをマウントする場合にも使用できます。

関連資料

- man ページの **autofs(8)**

7.9.2. autofs 設定ファイル

このセクションでは、**autofs** サービスで使用される設定ファイルの使用方法と構文を説明します。

マスターマップファイル
autofs サービスは、デフォルトの主要設定ファイルとして **/etc/auto.master** (マスターマップ) を使用します。これは、**/etc/autofs.conf** 設定ファイルの **autofs** 設定を Name Service Switch (NSS) メカニズムとともに使用することで、サポートされている他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。

すべてのオンデマンドマウントポイントはマスターマップで設定する必要があります。マウントポイント、ホスト名、エクスポートされたディレクトリー、オプションはすべて、ホストごとに手動で設定するのではなく、一連のファイル (またはサポートされているその他のネットワークソース) で指定できます。

マスターマップファイルには、**autofs** により制御されるマウントポイントと、それに対応する設定ファイルまたは自動マウントマップと呼ばれるネットワークソースが一覧表示されます。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。たとえば、**/mnt/data/** です。

map-file
: 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ファイルです。

options
: 指定した場合に、エントリーにオプションが指定されていなければ、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。

例7.12 /etc/auto.master ファイル

以下は **/etc/auto.master** ファイルのサンプル行です。

```
/mnt/data  /etc/auto.data
```

マップファイル
マップファイルは、個々のオンデマンドマウントポイントのプロパティを設定します。

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。タイムアウトが指定されているときに、タイムアウト期間中ディレクトリーにアクセスしなかった場合、ディレクトリーは自動的にアンマウントされます。

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オプションフィールドはエントリーの末尾ではなく、マウントポイントと場所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** のマウントポイントを参照しています。これは 1 つのインダイレクトマウントの単一ディレクトリー名であっても、複数のダイレクトマウント用のマウントポイントの完全パスであっても構いません。ダイレクトマップとインダイレクトマップの各エントリーキー (**mount-point**) の後に空白で区切られたオフセットディレクトリー (/ で始まるサブディレクトリー名) が記載されます。これがマルチマウントエントリーと呼ばれるものです。

options
: オプションを指定すると、これはそのマップエントリー用のマウントオプションになります。エントリー自体にはオプション指定を行いません。このフィールドは任意です。

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 / で始まる)、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。

例7.13 マップファイル

以下は、マップファイルのサンプルです (例: **/etc/auto.misc**)。

```
payroll  -fstype=nfs4  personnel:/dev/disk/by-uuid/52b94495-e106-4f29-b868-fe6f6c2789b1
sales    -fstype=xfs   :/dev/disk/by-uuid/5564ed00-6aac-4406-bfb4-c59bf5de48b5
```

マップファイルの最初の列は、**autofs** マウントポイント (**personnel** というサーバーからの **sales** と **payroll**) を示しています。2 列目は **autofs** マウントのオプションを示しています。3 列目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。

与えられた設定を使用すると、プロセスが **/home/payroll/2006/July.sxc** などのアンマウントされたディレクトリー **autofs** へのアクセスを要求する場合、**autofs** サービスは自動的にディレクトリーをマウントします。

amd マップ形式
autofs サービスは、**amd** 形式のマップ設定も認識します。これは Red Hat Enterprise Linux から削除された、**am-utils** サービス用に書き込まれた既存の自動マウント機能の設定を再利用する場合に便利です。

しかし、Red Hat は前述のセクションで説明したシンプルな **autofs** 形式の使用を推奨しています。

関連資料

- man ページの **autofs(5)**、**autofs.conf(5)**、**auto.master(5)**
- **amd** マップ形式の詳細は、**/usr/share/doc/autofs/README.amd-maps** ファイルを参照してください。**autofs** パッケージにあります。

7.9.3. autofs マウントポイントの設定

この手順では、**autofs** サービスを使用してオンデマンドマウントポイントを設定する方法を説明します。

前提条件

- **autofs** パッケージをインストールします。

```
# yum install autofs
```

- **autofs** サービスを起動して有効にします。

```
# systemctl enable --now autofs
```

手順

1. **/etc/auto.identifier** にあるオンデマンドマウントポイント用のマップファイルを作成します。**identifier** をマウントポイントを識別する名前に置き換えます。
2. マップファイルで、「autofs 設定ファイル」 の説明に従って、マウントポイント、オプション、および場所の各フィールドを入力します。
3. マスターマップファイルで、「autofs 設定ファイル」 の説明に従って、マップファイルを登録します。
4. オンデマンドディレクトリーのコンテンツへのアクセスを試みます。

```
$ ls automounted-directory
```

7.9.4. autofs サイトの設定ファイルの上書き/拡張

クライアントシステム上の特定のマウントポイントについて、サイトのデフォルトを上書きすると便利なことがあります。

例7.14 初期条件

たとえば、次の条件を検討します。

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

```
automount:    files nis
```

- **auto.master** ファイルには以下が含まれている。

```
+auto.master
```

- NIS の **auto.master** マップファイルには以下が含まれている。

```
/home auto.home
```

- NIS の **auto.home** マップには以下が含まれている。

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

- **/etc/auto.home** ファイルマップが存在しない。

手順

例7.15 別のサーバーからのホームディレクトリーのマウント

上記の条件で、クライアントシステムが NIS マップの **auto.home** を上書きして別のサーバーからホームディレクトリーをマウントする必要があるとします。

- この場合、クライアントは次の **/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** マップにエントリーが含まれています。

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

自動マウント機能は最初に出現したマウントポイントのみを処理するため、**/home** ディレクトリーには NIS **auto.home** マップではなく **/etc/auto.home** の内容が含まれます。

例7.16 選択されたエントリーのみを使用した auto.home の拡張

別の方法として、サイト全体の **auto.home** マップを少しのエントリーを使用して拡張するには、次の手順を行います。

1. **/etc/auto.home** ファイルマップを作成し、そこに新しいエントリーを追加します。最後に、NIS の **auto.home** マップを含めます。すると **/etc/auto.home** ファイルマップは次のようになります。

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

2. これらの NIS の **auto.home** マップ条件で、**/home** ディレクトリーの出力内容を一覧表示すると次のようになります。

```
$ ls /home

beth joe mydir
```

autofs は、読み込み中のファイルマップと同じ名前のファイルマップの内容を組み込まないため、上記の例は期待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。

7.9.5. LDAP を使用した自動マウント機能マップの格納

この手順では、**autofs** マップファイルではなく LDAP 設定で自動マウント機能マップを格納するように **autofs** を設定します。

前提条件

- 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 **autofs** パッケージの依存関係として自動的にインストールされます。

手順

1. LDAP アクセスを設定するには、**/etc/openldap/ldap.conf** ファイルを変更します。**BASE**、**URI**、**schema** の各オプションがサイトに適切に設定されていることを確認します。

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

例7.17 autofs 設定のセッティング

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

3. 他のすべてのスキーマエントリーが設定内でコメントされていることを確認してください。**automountKey** 属性は、**rfc2307bis** スキーマの **cn** 属性を置き換えます。以下は、LDAP データ交換形式 (LDIF) 設定の例です。

例7.18 LDF の設定

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

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

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

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

automountKey: /home
automountInformation: auto.home

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

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

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

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

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

関連資料

- **rfc2307bis** ドラフト - https://tools.ietf.org/html/draft-howard-rfc2307bis

7.10. ROOT ファイルシステムに対する読み取り専用パーミッションの設定

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

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

システムが正しく機能するためには、一部のファイルやディレクトリーで書き込みパーミッションが必要とされます。root ファイルシステムが読み取り専用モードでマウントされると、これらのファイルは **tmpfs** 一時ファイルシステムを使用して RAM にマウントされます。

そのようなファイルとディレクトリーのデフォルトセットは、**/etc/rwtab** ファイルから読み込まれ、以下のような内容になっています。

```
dirs /var/cache/man
dirs /var/gdm
<content truncated>

empty /tmp
empty /var/cache/foomatic
<content truncated>

files /etc/adjtime
files /etc/ntp.conf
<content truncated>
```

/etc/rwtab ファイルのエントリーは以下の形式に従います。

```
copy-method    path
```

この構文で、以下のことを行います。

- **copy-method** を、ファイルまたはディレクトリーを tmpfs にコピーする方法を指定するキーワードの 1 つに置き換えます。
- **path** をファイルまたはディレクトリーへのパスに置き換えます。

/etc/rwtab ファイルは、ファイルまたはディレクトリーを **tmpfs** にコピーする方法として以下を認識します。

empty

空のパスが **tmpfs** にコピーされます。以下に例を示します。

```
empty /tmp
```

dirs

ディレクトリーツリーが空の状態で **tmpfs** にコピーされます。以下に例を示します。

```
dirs /var/run
```

files

ファイルやディレクトリーツリーはそのまま **tmpfs** にコピーされます。以下に例を示します。

```
files /etc/resolv.conf
```

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

7.10.2. ブート時に読み取り専用パーミッションでマウントするように root ファイルシステムの設定

この手順を行うと、root ファイルシステムは以降のすべてのブートで読み取り専用としてマウントされます。

手順

1. **/etc/sysconfig/readonly-root** ファイルで、**READONLY** オプションを **yes** に設定します。

```
# Set to 'yes' to mount the file systems as read-only.
READONLY=yes
```

2. **/etc/fstab** ファイルの root エントリー (/) に **ro** オプションを追加します。

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

3. **ro** オプションを **/etc/default/grub** ファイルの **GRUB_CMDLINE_LINUX** ディレクティブに追加し、ディレクティブに **rw** が含まれていないことを確認します。

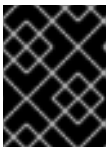
```
GRUB_CMDLINE_LINUX="rhgb quiet... ro"
```

4. GRUB2 設定ファイルを再作成します。

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

5. **tmpfs** ファイルシステムに書き込みパーミッションでマウントするファイルとディレクトリーを追加する必要がある場合は、**/etc/rwtab.d/** ディレクトリーにテキストファイルを作成し、そこに設定を置きます。
たとえば、**/etc/example/file** ファイルを書き込みパーミッションでマウントするには、この行を **/etc/rwtab.d/example** ファイルに追加します。

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



重要

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

6. システムを再起動して変更を適用します。

トラブルシューティング

- 誤って読み取り専用パーミッションで root ファイルシステムをマウントした場合は、次のコマンドを使用して、読み書きパーミッションで再度マウントできます。

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

第8章 STRATIS を使用した階層化ローカルストレージの管理

Stratis ハイレベルシステムに統合されている複雑なストレージ設定を簡単に設定および管理できます。

8.1. STRATIS ファイルシステムの設定

システム管理者は、Stratis ボリューム管理ファイルシステムをシステム上で有効にしてセットアップし、階層化ストレージを簡単に管理できます。

8.1.1. Stratis の目的と機能

Stratis は、Linux 用のローカルストレージ管理ソリューションです。これは、シンプルさと使いやすさに力を入れており、高度なストレージ機能にアクセスできます。

Stratis を使用すると、以下の活動をより簡単に行うことができます。

- ストレージの初期設定
- 後で変更を加えます。
- 高度なストレージ機能の使用

Stratis は、高度なストレージ機能をサポートする、ユーザーとカーネルのハイブリッドローカルストレージ管理システムです。Stratis は、ストレージ**プール**の概念を中心としています。このプールは 1 つ以上のローカルディスクまたはパーティションから作成され、ボリュームはプールから作成されます。

プールにより、次のような多くの便利な機能を使うことができます。

- ファイルシステムのスナップショット
- シンプロビジョニング
- 階層化

8.1.2. Stratis ボリュームの構成要素

外部的には、Stratis は、コマンドラインインターフェースおよび API に次のボリュームコンポーネントを表示します。

blockdev

ディスクやディスクパーティションなどのブロックデバイス。

pool

1 つ以上のブロックデバイスで構成されています。
プールの合計サイズは固定で、ブロックデバイスのサイズと同じです。

プールには、**dm-cache** ターゲットを使用した不揮発性データキャッシュなど、ほとんどの Stratis レイヤーが含まれています。

Stratis は各プールの **/stratis/my-pool/** ディレクトリーを作成します。このディレクトリーには、プール内の Stratis ファイルシステムを表すデバイスへのリンクが含まれています。

filesystem

各プールには、ファイルを格納する 1 つ以上のファイルシステムを含めることができます。

ファイルシステムはシンプロビジョニングされており、合計サイズは固定されていません。ファイルシステムの実際のサイズは、そこに格納されているデータとともに大きくなります。データのサイズがファイルシステムの仮想サイズに近づくと、Stratis はシンボリュームとファイルシステムを自動的に拡張します。

ファイルシステムは XFS でフォーマットされています。

重要

Stratis は、Stratis を使用して作成したファイルシステムに関する情報を追跡し、XFS はそれを認識しません。また、XFS を使用して変更を行っても、自動的に Stratisに更新を作成しません。ユーザーは、Stratis が管理する XFS ファイルシステムを再フォーマットまたは再構成してはいけません。

Stratis は、パスが **/stratis/my-pool/my-fs** のファイルシステムへのリンクを作成します。

注記

Stratis は、**dmsetup** リストと **/proc/partitions** ファイルに表示される多くの Device Mapper デバイスを使用します。同様に、**lsblk** コマンドの出力は、Stratis の内部の仕組みとレイヤーを反映しています。

8.1.3. Stratis で使用可能なブロックデバイス

このセクションでは、Stratis に使用できるストレージデバイスを一覧で紹介します。

対応デバイス
Stratis プールは、次の種類のブロックデバイスで動作するかどうかをテスト済みです。

- LUKS
- LVM 論理ボリューム
- MDRAID
- DM Multipath
- iSCSI
- HDD および SSD
- NVMe デバイス

警告

現行バージョンでは、Stratis は、ハードドライブまたはその他のハードウェアの不具合に対応しません。複数のハードウェアデバイスに Stratis プールを作成するには、データにアクセスするには、複数のデバイスが動作状態になっているため、データを損失するリスクが高まります。

対応していないデバイス
Stratis にはシンプロビジョニングレイヤーが含まれているため、Red Hat はすでにシンプロビジョニングされているブロックデバイスに Stratis プールを配置することを推奨していません。

関連資料

- iSCSI や、ネットワークを必要とするその他のブロックデバイスの **_netdev** マウントオプションに関する情報は、man ページの **systemd.mount(5)** を参照してください。

8.1.4. Stratis のインストール

この手順では、Stratis の使用に必要なすべてのパッケージをインストールします。

手順

1. Stratis サービスとコマンドラインユーティリティーを提供するパッケージをインストールします。

```
# yum install stratisd stratis-cli
```

2. **stratisd** サービスが有効になっていることを確認してください。

```
# systemctl enable --now stratisd
```

8.1.5. Stratis プールの作成

この手順では、1 つ以上のブロックデバイスから Stratis プールを作成します。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis プールを作成するブロックデバイスは使用されておらず、マウントされていない。
- Stratis プールを作成するブロックデバイスが、それぞれ 1 GiB 以上である。
- IBM Z アーキテクチャーでは、**/dev/dasd*** ブロックデバイスをパーティションに分割している。Stratis プールでパーティションを使用します。

手順

1. 選択したブロックデバイスにファイルシステム、パーティションテーブル、または RAID 署名が含まれている場合は、それらを消去します。

```
# wipefs --all block-device
```

block-device をブロックデバイスへのパスに置き換えます。たとえば、**/dev/sdb** に置き換えます。

2. ブロックデバイスに Stratis プールを作成するには、以下を使用します。

```
# stratis pool create my-pool block-device
```

- **my-pool** を、プールの任意の名前に置き換えます。
- **block-device** を、空または消去したブロックデバイスへのパスに置き換えます。たとえば、**/dev/sdb** に置き換えます。

複数のブロックデバイスからプールを作成するには、すべてのブロックデバイスをコマンドラインに追加します。

```
# stratis pool create my-pool device-1 device-2 device-n
```

3. 確認するために、システム上のすべてのプールを一覧表示します。

```
# stratis pool list
```

関連資料

- man ページの **stratis(8)**

次のステップ

- プールに Stratis ファイルシステムを作成します。「Stratis ファイルシステムの作成」 を参照してください。

8.1.6. Stratis ファイルシステムの作成

この手順では、既存の Stratis プールに Stratis ファイルシステムを作成します。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis プールを作成している。「Stratis プールの作成」を参照してください。

手順

1. Stratis ファイルシステムをプール上に作成するには、以下を使用します。

```
# stratis fs create my-pool my-fs
```

- **my-pool** を、既存の Stratis プールの名前に置き換えます。
- **my-fs** を、ファイルシステムの任意の名前に置き換えます。

2. 確認するために、プール内のファイルシステムを一覧表示します。

```
# stratis fs list my-pool
```

関連資料

- man ページの **stratis(8)**

次のステップ

- Stratis ファイルシステムをマウントします。「Stratis ファイルシステムのマウント」 を参照してください。

8.1.7. Stratis ファイルシステムのマウント

この手順では、コンテンツにアクセスするために既存の Stratis ファイルシステムをマウントします。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis ファイルシステムを作成している。「Stratis ファイルシステムの作成」 を参照してください。

手順

- ファイルシステムをマウントするには、**/stratis/** ディレクトリーで Stratis が維持するエントリーをします。

```
# mount /stratis/my-pool/my-fs mount-point
```

これでファイルシステムは **mount-point** ディレクトリーにマウントされ、使用できるようになりました。

関連資料

- man ページの **mount(8)**

8.1.8. Stratis ファイルシステムを永続的に維持

この手順では、Stratis ファイルシステムを永続的にマウントして、システムが起動した後に自動的に利用できるようにします。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis ファイルシステムを作成している。「Stratis ファイルシステムの作成」 を参照してください。

手順

1. ファイルシステムの UUID 属性を調べます。

```
$ lsblk --output=UUID /stratis/my-pool/my-fs
```

以下に例を示します。

例8.1 Stratis ファイルシステムの UUID の表示

```
$ lsblk --output=UUID /stratis/my-pool/fs1
```

```
UUID
a1f0b64a-4ebb-4d4e-9543-b1d79f600283
```

2. このマウントポイントのディレクトリーがない場合は、作成します。

```
# mkdir --parents mount-point
```

3. root として、**/etc/fstab** ファイルを編集し、UUID で識別されるファイルシステムに行を追加します。**xfs** をファイルシステムのタイプとして使用し、**x-systemd.requires=stratisd.service** オプションを追加します。
以下に例を示します。

例8.2 /etc/fstab の /fs1 マウントポイント

```
UUID=a1f0b64a-4ebb-4d4e-9543-b1d79f600283 /fs1 xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

4. システムが新しい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
```

5. ファイルシステムをマウントして、設定が機能することを確認します。

```
# mount mount-point
```

関連資料

- 「ファイルシステムの永続的なマウント」

8.1.9. 関連情報

- **Stratis Storage** の Web サイト - https://stratis-storage.github.io/

8.2. 追加のブロックデバイスを使用した STRATIS ボリュームの拡張

Stratis ファイルシステムのストレージ容量を増やすために、追加のブロックデバイスを Stratis プールに追加できます。

8.2.1. Stratis ボリュームの構成要素

外部的には、Stratis は、コマンドラインインターフェースおよび API に次のボリュームコンポーネントを表示します。

blockdev

ディスクやディスクパーティションなどのブロックデバイス。

pool

1 つ以上のブロックデバイスで構成されています。
プールの合計サイズは固定で、ブロックデバイスのサイズと同じです。

プールには、**dm-cache** ターゲットを使用した不揮発性データキャッシュなど、ほとんどの Stratis レイヤーが含まれています。

Stratis は各プールの **/stratis/my-pool/** ディレクトリーを作成します。このディレクトリーには、プール内の Stratis ファイルシステムを表すデバイスへのリンクが含まれています。

filesystem

各プールには、ファイルを格納する 1 つ以上のファイルシステムを含めることができます。ファイルシステムはシンプロビジョニングされており、合計サイズは固定されていません。ファイルシステムの実際のサイズは、そこに格納されているデータとともに大きくなります。データのサイズがファイルシステムの仮想サイズに近づくと、Stratis はシンボリュームとファイルシステムを自動的に拡張します。

ファイルシステムは XFS でフォーマットされています。

重要

Stratis は、Stratis を使用して作成したファイルシステムに関する情報を追跡し、XFS はそれを認識しません。また、XFS を使用して変更を行っても、自動的に Stratisに更新を作成しません。ユーザーは、Stratis が管理する XFS ファイルシステムを再フォーマットまたは再構成してはいけません。

Stratis は、パスが **/stratis/my-pool/my-fs** のファイルシステムへのリンクを作成します。

注記

Stratis は、**dmsetup** リストと **/proc/partitions** ファイルに表示される多くの Device Mapper デバイスを使用します。同様に、**lsblk** コマンドの出力は、Stratis の内部の仕組みとレイヤーを反映しています。

8.2.2. Stratis プールへのブロックデバイスの追加

この手順では、Stratis ファイルシステムで使用できるように、1 つ以上のブロックデバイスを Stratis プールに追加します。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis プールに追加するブロックデバイスは使用されておらず、マウントされていない。
- Stratis プールに追加するブロックデバイスは使用されておらず、それぞれ 1 GiB 以上である。

手順

- 1 つ以上のブロックデバイスをプールに追加するには、以下を使用します。

```
# stratis pool add-data my-pool device-1 device-2 device-n
```

関連資料

- man ページの **stratis(8)**

8.2.3. 関連情報

- **Stratis Storage** の Web サイト - https://stratis-storage.github.io/

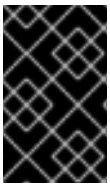
8.3. STRATIS ファイルシステムの監視

Stratis ユーザーは、システム上の Stratis ボリュームに関する情報を表示して、その状態と空き容量を監視できます。

8.3.1. さまざまなユーティリティーが報告する Stratis のサイズ

このセクションでは、**df** などの標準的なユーティリティーと、**stratis** ユーティリティーにより報告される Stratis サイズの違いを説明します。

df などの標準的な Linux ユーティリティーは、Stratis 上の XFS ファイルシステムレイヤーのサイズを報告します。これは 1 TiB です。Stratis の実際のストレージ使用量は、シンプロビジョニングにより少なくなっており、また XFS レイヤーが満杯に近くなると Stratis が自動的にファイルシステムを拡張するので、これは特に有用な情報ではありません。



重要

Stratis ファイルシステムに書き込まれているデータ量を定期的に監視します。これは **Total Physical Used** の値として報告されます。これが **Total Physical Size** の値を超えていないことを確認してください。

関連資料

- man ページの **stratis(8)**

8.3.2. Stratis ボリュームの情報表示

この手順では、Stratis ボリュームに関する合計サイズ、使用済みサイズ、空きサイズ、ファイルシステム、プールに属するブロックデバイスなどの統計情報を一覧表示します。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。

手順

- システム上で Stratis に使用されているすべての**ブロックデバイス**に関する情報を表示するには、次の手順を実行します。

```
# stratis blockdev

Pool Name  Device Node    Physical Size   State  Tier
my-pool    /dev/sdb           9.10 TiB  In-use  Data
```

- システム上のすべての Stratis **プール**に関する情報を表示するには、次の手順を実行します。

```
# stratis pool
```

```
Name    Total Physical Size  Total Physical Used
my-pool            9.10 TiB             598 MiB
```

- システム上のすべての Stratis **ファイルシステム**に関する情報を表示するには、次の手順を実行します。

```
# stratis filesystem

Pool Name  Name  Used     Created            Device
my-pool    my-fs 546 MiB  Nov 08 2018 08:03  /stratis/my-pool/my-fs
```

関連資料

- man ページの **stratis(8)**

8.3.3. 関連情報

- **Stratis Storage** の Web サイト - https://stratis-storage.github.io/

8.4. STRATIS ファイルシステムでのスナップショットの使用

Stratis ファイルシステムのスナップショットを使用して、ファイルシステムの状態を任意の時点でキャプチャーし、後でそれを復元できます。

8.4.1. Stratis スナップショットの特徴

このセクションでは、Stratis ファイルシステムのスナップショットのプロパティと制限事項を説明します。

Stratis では、スナップショットは、別の Stratis ファイルシステムのコピーとして作成した通常の Stratis ファイルシステムです。スナップショットには最初、元のファイルシステムと同じファイルの内容が含まれていますが、スナップショットが変更すると変更する可能性があります。スナップショットにどんな変更を加えても、元のファイルシステムには反映されません。

Stratis の現在のスナップショット実装は、次のような特徴があります。

- ファイルシステムのスナップショットは別のファイルシステムです。
- スナップショットとそのスナップショット元は、有効期間中はリンクされません。スナップショットされたファイルシステムは、作成元のファイルシステムよりも長く存続します。
- スナップショットを作成するためにファイルシステムをマウントする必要はありません。
- 各スナップショットは、XFS ログに必要な実際のバッキングストレージの約半分のギガバイトを使用します。

8.4.2. Stratis スナップショットの作成

この手順では、既存の Stratis ファイルシステムのスナップショットとして Stratis ファイルシステムを作成します。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。

- **stratisd** サービスが実行されている。
- Stratis ファイルシステムを作成している。「Stratis ファイルシステムの作成」 を参照してください。

手順

- Stratis スナップショットを作成するには、以下を使用します。

```
# stratis fs snapshot my-pool my-fs my-fs-snapshot
```

関連資料

- man ページの **stratis(8)**

8.4.3. Stratis スナップショットのコンテンツへのアクセス

この手順では、Stratis ファイルシステムのスナップショットをマウントして、読み書き操作にアクセスできるようにします。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis スナップショットを作成している。「Stratis スナップショットの作成」 を参照してください。

手順

- スナップショットにアクセスするには、**/stratis/my-pool/** ディレクトリーから通常のファイルシステムとしてマウントします。

```
# mount /stratis/my-pool/my-fs-snapshot mount-point
```

関連資料

- 「Stratis ファイルシステムのマウント」
- man ページの **mount(8)**

8.4.4. Stratis ファイルシステムを以前のスナップショットに戻す

この手順では、Stratis ファイルシステムの内容を Stratis スナップショットでキャプチャーされた状態に戻します。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis スナップショットを作成している。「Stratis スナップショットの作成」 を参照してください。

手順

1. 必要に応じて、後でそれにアクセスできるように、現在のファイルシステムの状態のバックアップを取得します。

```
# stratis filesystem snapshot my-pool my-fs my-fs-backup
```

2. 元のファイルシステムをアンマウントして削除します。

```
# umount /stratis/my-pool/my-fs
# stratis filesystem destroy my-pool my-fs
```

3. 元のファイルシステムの名前でスナップショットのコピーを作成します。

```
# stratis filesystem snapshot my-pool my-fs-snapshot my-fs
```

4. 元のファイルシステムと同じ名前でアクセスできるようになった、スナップショットをマウントします。

```
# mount /stratis/my-pool/my-fs mount-point
```

my-fs という名前のファイルシステムの内容は、スナップショット **my-fs-snapshot**と同じになりました。

関連資料

- man ページの **stratis(8)**

8.4.5. Stratis スナップショットの削除

この手順では、Stratis スナップショットをプールから削除します。スナップショットのデータは失われます。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis スナップショットを作成している。「Stratis スナップショットの作成」 を参照してください。

手順

1. スナップショットをアンマウントします。

```
# umount /stratis/my-pool/my-fs-snapshot
```

2. スナップショットを破棄します。

```
# stratis filesystem destroy my-pool my-fs-snapshot
```

関連資料

- man ページの **stratis(8)**

8.4.6. 関連情報

- **Stratis Storage** の Web サイト - https://stratis-storage.github.io/

8.5. STRATIS ファイルシステムの削除

既存の Stratis ファイルシステム、または Stratis プールを削除して、そこに含まれるデータを破棄できます。

8.5.1. Stratis ボリュームの構成要素

外部的には、Stratis は、コマンドラインインターフェースおよび API に次のボリュームコンポーネントを表示します。

blockdev

ディスクやディスクパーティションなどのブロックデバイス。

pool

1 つ以上のブロックデバイスで構成されています。
プールの合計サイズは固定で、ブロックデバイスのサイズと同じです。

プールには、**dm-cache** ターゲットを使用した不揮発性データキャッシュなど、ほとんどの Stratis レイヤーが含まれています。

Stratis は各プールの **/stratis/my-pool/** ディレクトリーを作成します。このディレクトリーには、プール内の Stratis ファイルシステムを表すデバイスへのリンクが含まれています。

filesystem

各プールには、ファイルを格納する 1 つ以上のファイルシステムを含めることができます。
ファイルシステムはシンプロビジョニングされており、合計サイズは固定されていません。ファイルシステムの実際のサイズは、そこに格納されているデータとともに大きくなります。データのサイズがファイルシステムの仮想サイズに近づくと、Stratis はシンボリュームとファイルシステムを自動的に拡張します。

ファイルシステムは XFS でフォーマットされています。

重要

Stratis は、Stratis を使用して作成したファイルシステムに関する情報を追跡し、XFS はそれを認識しません。また、XFS を使用して変更を行っても、自動的に Stratisに更新を作成しません。ユーザーは、Stratis が管理する XFS ファイルシステムを再フォーマットまたは再構成してはいけません。

Stratis は、パスが **/stratis/my-pool/my-fs** のファイルシステムへのリンクを作成します。

注記

Stratis は、**dmsetup** リストと **/proc/partitions** ファイルに表示される多くの Device Mapper デバイスを使用します。同様に、**lsblk** コマンドの出力は、Stratis の内部の仕組みとレイヤーを反映しています。

8.5.2. Stratis ファイルシステムの削除

この手順では、既存の Stratis ファイルシステムを削除します。そこに保存されているデータは失われます。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis ファイルシステムを作成している。「Stratis ファイルシステムの作成」 を参照してください。

手順

1. ファイルシステムをアンマウントします。

```
# umount /stratis/my-pool/my-fs
```

2. ファイルシステムを破棄します。

```
# stratis filesystem destroy my-pool my-fs
```

3. ファイルシステムがもう存在しないことを確認します。

```
# stratis filesystem list my-pool
```

関連資料

- man ページの **stratis(8)**

8.5.3. Stratis プールの削除

この手順では、既存の Stratis プールを削除します。そこに保存されているデータは失われます。

前提条件

- Stratis がインストールされている。「Stratis のインストール」 を参照してください。
- **stratisd** サービスが実行されている。
- Stratis プールを作成している。「Stratis プールの作成」を参照してください。

手順

1. プール上のファイルシステムを一覧表示します。

```
# stratis filesystem list my-pool
```

2. プール上のすべてのファイルシステムをアンマウントします。

```
# umount /stratis/my-pool/my-fs-1 \
        /stratis/my-pool/my-fs-2 \
        /stratis/my-pool/my-fs-n
```

3. ファイルシステムを破棄します。

```
# stratis filesystem destroy my-pool my-fs-1 my-fs-2
```

4. プールを破棄します。

```
# stratis pool destroy my-pool
```

5. プールがもう存在しないことを確認します。

```
# stratis pool list
```

関連資料

- man ページの **stratis(8)**

8.5.4. 関連情報

- **Stratis Storage** の Web サイト - https://stratis-storage.github.io/