LFS307-JP

Linux システム管理入門

Version 1.4

Version 1.4

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

The Linux Foundation により開発および提供されているトレーニング資料は、著作権および知的財産権により保護されています。

本書に記載されているオープンソースコードは他の著作権者が存在する可能性があり、該当のオープンソースライセンスに基づき使用されています。

トレーニング資料は受講者の個人利用のために提供されます。The Linux Foundation およびクリエーションライン株式会社の許可なしでの複製、修正、受講者以外の人への再配布、または他者へのトレーニング供与目的での使用を禁じます。

本書のいかなる部分も、事前の書面による許可なしでの複製、複写、送信、情報検索システムへの保存を禁じます。

発行元：

The Linux Foundation
https://www.linuxfoundation.org

No representations or warranties are made with respect to the contents or use of this material, and any express or implied warranties of merchantability or fitness for any particular purpose or specifically disclaimed.

Although third-party application software packages may be referenced herein, this is for demonstration purposes only and shall not constitute an endorsement of any of these software applications.

Linux is a registered trademark of Linus Torvalds. Other trademarks within this course material are the property of their respective owners.

If there are any questions about proper and fair use of the material herein, please contact https://trainingsupport.linuxfoundation.org.

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

目次

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

注目

** これらのセクションの一部または全体をオプション扱いとする場合があります。これらには補足資料、専門トピック、または高度な話題が含まれインストラクターが教室の状況や時間制約に応じてこれらの内容を紹介するかを判断します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

図目次

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

表目次

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 1 章

はじめに

1.1 Linux Foundation

Linux Foundation とは

- 以下の成長を推進することに専念した、非営利のコンソーシアムである:
  - **Linux**
  - その他多くの **オープンソース ソフトウェア** (**OSS**) 開発プロジェクト、と（それを支援する）コミュニティ
- 次項を提供することで持続可能な **OSS** エコシステムの創造をサポート:
  - 財政的・知的資源とサービス
  - トレーニング
  - イベント
- 元々は **Linux** 開発を保護・サポート・改善し **Linux** 創始者リーナス・トーバルズ氏のスポンサーとして設立
- 中立的な協調体制を保った形で、大手テック企業や開発者が活動を支援
- 詳細: https://www.linuxfoundation.org

Linux Foundation は、開発者がオープン テクノロジー プロジェクトをコーディング、管理、拡張するための中立的で信頼できるハブを提供しています。2000 年に設立された **Linux Foundation** は 1,000 人以上のメンバーに支えられ、オープンソース ソフトウェア、オープン スタンダード、オープン データ、オープン ハードウェアに関するコラボレーションのための世界有数の拠点となっています。**Linux Foundation** の手法は、ベストプラクティスを活用し、貢献者、ユーザー、ソリューションプロバイダーのニーズに応えることで、オープンなコラボレーションのための持続可能なモデルを構築することに焦点を当てています。

Linux Foundation は、歴史上世界最大かつ最も普及しているオープンソース ソフトウェア プロジェクトである **Linux** を主催しています。また **Linux** の生みの親であるリーナス・トーバルズとリードメンテナーのグレッグ・クロア-ハートマンの本拠地でもあります。**Linux** の成功は、オープンソース コミュニティの成長を促進し、オープンソースの商業的有効性を実証し、あらゆる業界や技術スタックのレベルにわたって無数の新しいプロジェクトを鼓舞しています。

その結果 **Linux Foundation** は今日 **Linux** をはるかに超えるものをホストしており、事実上すべての産業分野にまたがる、今日の企業に力を与える多くの重要なオープンソース プロジェクトの傘となっています。私たちが注目している技術には、ビッグ データとアナリティクス、ネットワーキング、組み込みシステムと IoT、ウェブ ツール、クラウド コンピューティング、エッジ コンピューティング、自動車、セキュリティ、ブロック チェーンなど、さまざまなものがあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Linux Foundation 主催のイベント

Linux Foundation が主催するイベントのごく一部
（北米・ヨーロッパ・アジアなど毎年複数の場所で行われるイベントを含む）

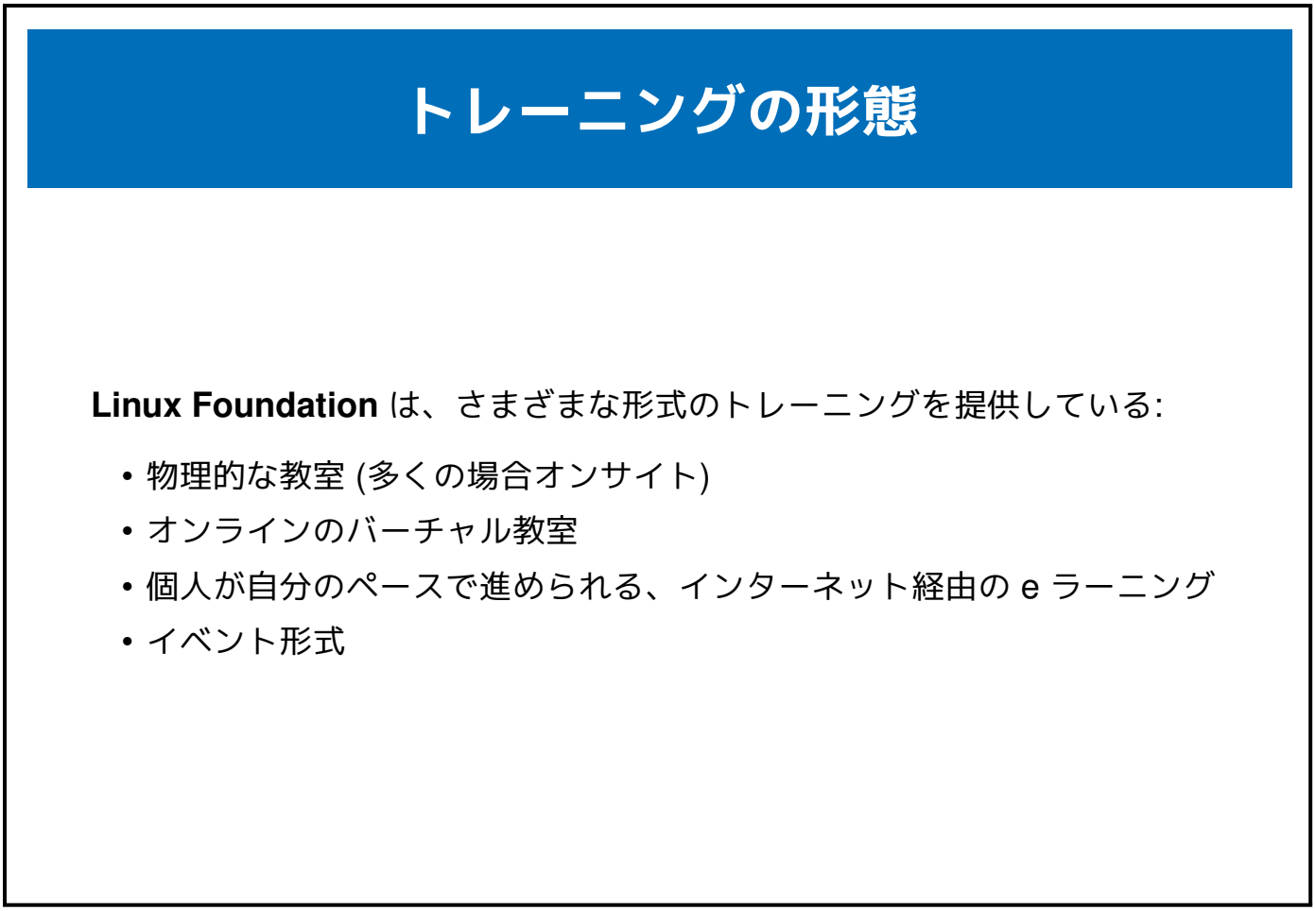
- Open Source Summit
- Embedded Linux Conference
- Open Networking & Edge Summit
- KubeCon + CloudNativeCon
- Automotive Linux Summit
- KVM Forum
- Linux Storage Filesystem and Memory Management Summit
- Linux Security Summit
- Linux Kernel Maintainer Summit
- The Linux Foundation Member Summit
- Open Compliance Summit
- その他、多数

世界中の 85,000 人以上のオープンソース技術者やリーダーが、毎年 **Linux Foundation** のイベントに集まり、アイデアを共有し、学び、コラボレーションしています。**Linux Foundation** のイベントは、オープンソースのメンテナー、開発者、アーキテクト、インフラストラクチャ マネージャー、オープンソース プログラム オフィスを率いるシステム アドミニストレーターや技術者、その他の重要なリーダーシップ機能にとって重要なミーティングの場となっています。

これらのイベントは、オープンソース コミュニティの中で迅速に知名度を上げ、次世代のテクノロジーを評価し創造する人々とつながりを持つことで、オープンソース開発作業を進めるのに最適な場所です。知識を共有し、得るためのフォーラムを提供し、組織がソフトウェアのトレンドを早期に把握し、将来の技術投資に役立てることを支援し、雇用者と人材を結びつけ、世界中の影響力のあるオープンソース専門家、メディア、アナリストに技術やサービスを紹介することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.2 Linux Foundation が提供するトレーニング



Linux Foundation のトレーニングは、コミュニティのための、コミュニティによるものであり、開発者コミュニティのリーダーから直接指導を受ける講師とコンテンツを特徴としています。

受講者は、OS や **Linux** ディストリビューションに柔軟に対応でき、技術的に高度なトレーニングを、開発コミュニティの実際のリーダーたちと共に受けることができます。**Linux Foundation** のコースでは、今日のキャリアで成功するために必要な幅広い基礎知識とネットワークが身につきます。オンラインまたは対面でのトレーニングにより、**Linux Foundation** のコースは、オープンソースの管理および開発のエッセンスについて、お客様またはお客様の開発者を常に先取りしておくことができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Linux Foundation が提供するトレーニング

現在受講可能な講座の種類:

- Linux プログラミング & 開発トレーニング
- エンタープライズ IT & Linux システム管理講座
- オープンソースコンプライアンス講座

詳細: https://training.linuxfoundation.org

Linux Foundation は、**edX** （https://edx.org）を通じて、無料の **MOOC**（**M**assively **O**pen **O**nline **C**ourses）を幅広く提供しています。これらのコースでは、オープンソースに関連する基本的なトピックから、かなり高度なトピックまでカバーしています。

edX のサイトで探すには、`"Linux Foundation"` で検索してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

著作権について

- この講座のコンテンツと関連する全ての教材（ハンドアウトを含む）は Copyright 2023 The Linux Foundation によって保護されています。

コピーや頒布はしないでください

このトレーニングのいかなる資料も、The Linux Foundation の書面による事前の許可なく、電子的、磁気的、機械的、写真複写的、記録的、またはこれに限定されないいかなる形式によっても、複製、配布、再出版、ダウンロード、表示、掲示、転送、または情報検索システムに格納することを禁じます。

本トレーニングは、ここで提供されるすべての資料を含め、**The Linux Foundation** からのいかなる保証もなく提供されるものです。**The Linux Foundation** は、ここに含まれる内容や詳細の使用または誤用から生じる損害や法的措置について、いかなる責任も負いません。

The Linux Foundation の資料が使用、コピー、またはその他の方法で不適切に配布されていると思われる場合は、https://trainingsupport.linuxfoundation.org にご連絡ください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

機密情報について

機密情報の非開示について

「秘密情報」には（たとえ 機密 または プロプラエタリ と表示されていたとしても）、次のいずれも含まないものとする

- オープンソースまたはオープン スタンダード プロジェクト（以下、総称して「オープン プロジェクト」）のコードベースに関連する既存または将来の貢献に関わる情報
- 一般的にオープン プロジェクトの形成や運営に関連する情報
- オープン プロジェクトに関わる一般的なビジネスに関する情報

このコースには機密情報は含まれておらず、また授業中にも機密情報を漏らしてはならない。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.3 Linux Foundation の資格

Linux Foundation 認定試験

- **Linux Foundation** は包括的な認定プログラムを提供している
- このプログラムの詳細については、https://training.linuxfoundation.org/certification で確認できる。この情報には各試験でカバーされる **ドメイン** と **コンピテンシー**についての詳細な説明が含まれる
- **LFCS**（**L**inux **F**oundation **C**ertified **S**ysadmin）試験以外にも、**Linux Foundation** は多くのオープンソース プロジェクトの認定試験を提供している。リストは常に拡大しているので、最新のリストは https://training.linuxfoundation.org/certification を参照
- コースの説明、技術的な必要条件、その他のロジスティクスなどの追加情報については、https://training.linuxfoundation.org を参照

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

トレーニングと試験の「ファイアウォール」

- **Linux Foundation (LF)** は、トレーニングを 2 つに分割している
  - 試験
  - 講座
- これらは、論理的な **ファイアウォール** によって隔離される
  - 第三者の企業や組織が **LF** 認証試験に向けた講座を作成し、提供できる
  - **LF** の講座でしか得られない秘訣（いわゆる秘密のレシピ）が生まれるのを防ぐ
  - **LF** の講座で **テスト対策だけを目標とした** 教え方を防ぐ
- （今日も含めて）講師は、一般的な公開情報だけに基いた講義を行う

カリキュラムの作成と管理を担当する **Linux Foundation** トレーニング部署は、認証試験の作成・管理・採点に直接関わることは一切ありません。

このように自ら**「ファイアウォール」**を立てることにより、**Linux Foundation** とは関係のない組織や企業でも受験者が認証試験に受かる手助けとなるトレーニング教材を第三者として作成できるのです。

また、試験に受かるために必要な秘策（秘密の勉強法など）がないことを保証できるのです。

さらに、**Linux Foundation** が提供するトレーニングが単なるテスト対策に留まらず、受講者が **Linux** システム管理者として成功するために必要な幅広い知識を提供できるような、堅牢で充実したものであることを保証できるのです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

試験準備用のリソース

- まず、こちらをダウンロード
  https://training.linuxfoundation.org/download-free-certification-prep-guide
- 次のセクションを含む
  - ドメインとコンピテンシー（トレーニングを通して身につくスキル）
  - 無料のトレーニングリソース
  - 有料のトレーニングリソース
  - 試験を受ける
- **受験者ハンドブック** はこちら

https://training.linuxfoundation.org/go/candidate_handbook

- 各試験の詳細はこちら

https://training.linuxfoundation.org/certification/lfcs

上記のドキュメントが全てではありませんが、多くの内容について触れています。そして、これらのドキュメントは全て **Linux Foundation** のウェブサイト https://training.linuxfoundation.org/certification でも確認できます。
ドキュメントには次のような内容を含みます。

- 試験範囲
- 試験を受ける必要条件（身分証明・認証・適任性・アクセシビリティ・機密性に関する必要条件）
- 受験費や返金ポリシーなど
- 利用可能な **Linux** ディストリビューション
- ハードウェア環境とソフトウェア環境が試験に適応しているかの確認
- 試験の始め方・終わり方
- 試験のインターフェイスとフォーマット
- 試験結果・採点と再採点の依頼・再受験ポリシー
- 認定証の発行・検証・失効・更新など
- 技術的サポートへのアクセス

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.4 Linux Foundation のデジタルバッジ

Linux Foundation は専門的な目標を達成するために必要な機能を提供することを約束しています。私達は技能や証明の提示が難しいことを理解しています。このため、私達は **Credly** 社とパートナー契約を結び、**Acclaim** プラットフォームを通してあなたの証明書の電子版を提供することにしました。これらのバッジは、メールの署名や電子レジュメ、**LinkedIn**、**Facebook**、**Twitter** のようなソーシャルメディアでも利用可能です。このデジタルイメージは、資格と獲得手順を記載した検証済みメタデータを含みます。

- バッジはあらゆる電子プラットフォーム経由で共有可能です。ソーシャルメディア、レジュメに埋め込み、電子メールの署名、ウェブ上など。
- すべてのバッジは、誰でもクリックするだけで検証が可能です。
- バッジは **Linux Foundation** から直接トレーニングコースを購入した、または認定トレーニングパートナーで受講し、いずれかの認定試験に合格した人全員に交付されます。
- 試験やトレーニングコース開発チームに貢献した方にもバッジを発行します。

動作方法

1. 私達のパートナーの **Credly** 社の **Acclaim** プラットフォームウェブサイトにバッジ要求したことを通知するメールを受信
2. メールに記載のリンクをクリック
3. **Acclaim** プラットフォームサイトにアカウントを作成しメールを確認
4. バッジを確認
5. 共有を開始

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.5 演習・解答・リソース

演習

- 各セッションの終わりに、ハンズオン演習を実施する
- 仮想マシンでも、ベアメタル（実マシン）でも実施可能となっている
  - 一部、仮想マシンでは出来ない演習がある
- 課題の一部は任意である
- 各課題の終わりには解答がある

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

講座の解答とその他リソースの入手方法

- **本講座用の演習の解答・参考資料などがある場合**
  https://training.linuxfoundation.org/cm/LFS307-JP でダウンロード可能
- 利用できるブラウザがない場合:

```
$ wget --user=LFtraining --password=<講師提供> \
  https://training.linuxfoundation.org/cm/LFS307-JP/LFS307-JP_V1.4_SOLUTIONS.tar.bz2
```

  RESOURCES ファイルも、次の手順で入手可能である:

```
$ wget --user=LFtraining --password=<講師提供> \
  https://training.linuxfoundation.org/cm/LFS307-JP/LFS307-JP_V1.4_RESOURCES.tar
```

- 誤字などの正誤表や、更新したソリューションもこのサイトで確認可能
- これらのファイルは、次の手順で展開が可能である:

```
$ tar xvf LFS307-JP_*SOLUTIONS*.tar.bz2
$ tar xvf LFS307-JP_*RESOURCES*.tar
```

SOLUTIONS ディレクトリと RESOURCES ディレクトリに各セッション用に s_01，s_10，s_13 のようなサブディレクトリがあります。ここには、必要なファイルや参考資料が入っています。これからのセクションでもこれらのファイルに触れることがあるでしょう。

講座によっては、RESOURCES ファイルがない可能性もあります。RESOURCES ファイルはアーカイブや動画のようなバイナリファイルのためのものです。演習のためのソースの tar ファイルなどのバイナリファイルは、必要に応じて説明付きで提供されます。デモ動画を含む講座もあります。その場合は実際の講座の時間以外で自習として観るものとして提供しています。ただし、講師がデモを紹介する場合や、あるいは他の **Linux** ディストリビューションでの実行例を見せる場合もあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.6 Linux とオープンソース プロジェクトの進化の影響

オープンソース ソフトウェアは止まらない

- **Linux Foundation** のコースは、**Kubernetes** や **Linux** カーネルなどのプロジェクトのアップストリームバージョンと同期するように常に更新されている
- (それでも) 時には非推奨となった機能が無くなるなど回避不能な破損が発生する事がある
- 私達は問題を最小限に抑えるよう努力はしているが、我々が制御できないアップストリーム側の変化に対応しなければならないことも多い
- そうしなければ、重要な機能が欠落した古くさい素材になってしまう
- そんなソースやターゲットの移り変わりへの対応もオープンソース ソフトウェアを扱う **楽しみ**のひとつである！

Linux Foundation のコースは、**Linux** カーネルや **Kubernetes** のようなプロジェクトの最新バージョンのリリースに併せて定期的に更新されています。例えば、私たちがどんなに最新の情報に追従するために努力してきたとしても、**Linux** はカーネルの機能を含めた技術レベルでも、ディストリビューションやインターフェイスのレベルでも、常に進化しています。

ですから、私たちはこのクラスをリリースする時にできるだけ最新の情報を提供するように努めましたが、私たちが説明していない変更や新機能があることを念頭に置いてください。それはやむを得ないことです。

その結果、どんなに努力しても次のような問題が発生する可能性があるのです。

- コースでサポートする **Linux** ディストリビューションで動作しなくなる。
- **API** が変更され、特定のカーネルやアップストリームのライブラリや製品を使用した実験実習ができなくなる。
- これまではサポートが続いていた、非推奨の機能がついに姿を消す。

クラスで使う全ての機能が常に動作することを期待する生徒がいます。私達はそれを実現するために努力しているのですが、同時に以下の事もわかっています。

- それは非現実的な期待である。
- 問題にぶつかってその解決方法を見つけ出すことは、授業の重要な構成要素であり、最先端のオープンソースプロジェクトに取り組む「冒険」の一部でもあるのです。動いている標的を狙うのはいつもとても難しいのです。

このような問題をすべて回避しようとすると、教材がすぐに古くなり、変化についていけなくなります。私たちは、そうした問題に気づいた講師や受講生が次のリリースに適切な改善策を盛り込めるようトレーニング マテリアルの設計者に知らせてくれることを期待しています。

これこそが真のオープンソースのあり方だと思います。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

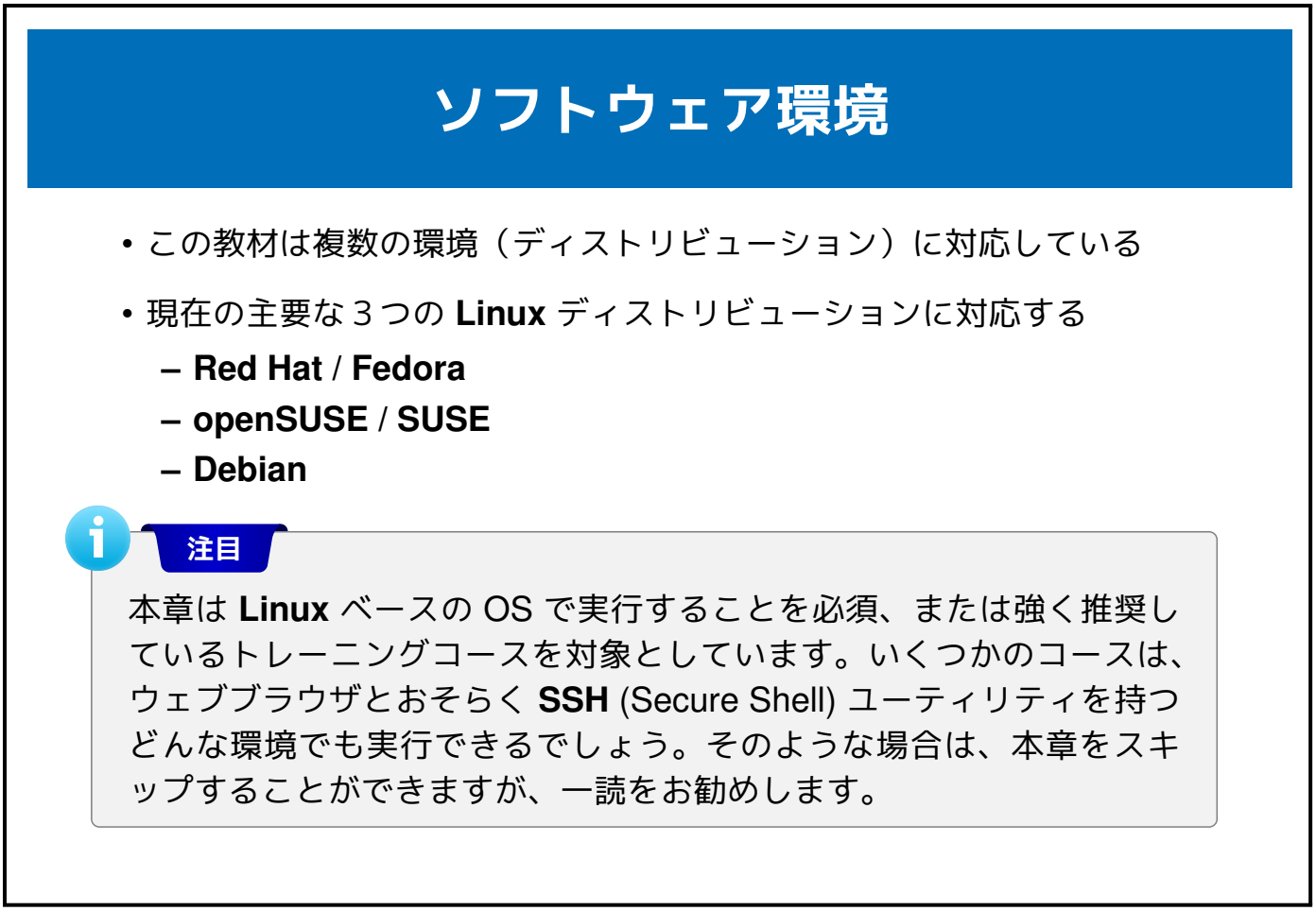
1.7 e ラーニング講座: LFS207-JP

関連 e ラーニング講座: LFS207-JP

- **Linux Foundation** はこの講座と非常によく似た内容の自分のペースで進められる e ラーニング版講座も提供しています。
  LFS207-JP: Linux システム管理入門
- この e ラーニング講座の大部分は、本講座と共通のコンテンツや演習を使用しています。
- １年間オンラインで教材にアクセスできるので、全てのコンテンツや演習を終えるために必要な時間をとることができます。
- 講師付きの場合は、全ての題材で必要な深さと幅を追求するための充分な時間をとることができません。また、いくつかの題材はオプショナル扱いとなっています。
- この講座への登録時に **LFS207-JP** へのサブスクリプションを受け取ったかもしれません。受けとっていない場合は https://training.linuxfoundation.org/ からアクセスすることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.8 ディストリビューションについての詳細



Linux Foundation が提供する教材は複数のディストリビューションに対応しているので１つのディストリビューションに縛られません。つまり技術的な説明、演習、手順の説明などはほとんどの最近のディストリビューションでそのまま実行可能です。また特定のベンダの商品を推奨することはありません。（ただし、具体例で紹介することはあります）

実践的にはこの教材の大部分は３つの主要な **Linux** ディストリビューションである **Red Hat** / **Fedora**、**openSUSE** / **SUSE**、**Debian** を対象として書かれています。受講者は通常この３つの系列のディストリビューション、あるいはそれらを元にした商品を利用しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ディストリビューションの選択

- 考慮すべき点
  - 会社で決められたディストリビューションがあるか
  - 新しいことを学びたいか
  - 保証が必要か
- ２つ以上のディストリビューションで試すことを推奨する

新しくディストリビューションを選ぶときはいくつかの質問を自分に投げかけるべきです。ある特定の **Linux** ディストリビューションに集中したくなる理由は多くあるでしょう。しかし私達は全てのディストリビューションを体験することをおすすめします。技術的な違いは主にパッケージ管理システム・ソフトウェアのバージョン・ファイルの位置などに限るということがすぐにわかるはずです。その違いさえ理解できれば１つの **Linux** ディストリビューションから他のディストリビューションへの切り替えが非常に簡単になります。あるツールや機能には（特定の、あるいは複雑なレポートの場合）ベンダーが提供するフロントエンドがあります。違うプラットフォームで実行するには教材に記載されている手順をところどころ修正する必要があるかもしれません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Red Hat / Fedora 系

- 教材は出版時での最新の **Red Hat Enterprise Linux (RHEL)** リリースに基づく
- **x86**・**x86-64**・**Itanium**・**PowerPC**・**IBM System Z** をサポート
- RPM ベースでインストールやアップデートには **dnf** (あるいは **yum**) を使用する
- リリースサイクルが長く、エンタープライズ級のサーバ環境向け
- **CentOS** と **Oracle Linux** の元となったディストリビューションである

注目

デモや演習では無料の **CentOS Stream** を使用しています。

Fedora は、**Red Hat Enterprise Linux**・**CentOS**・**Oracle Linux** の基盤となるコミュニティ主導ディストリビューションです。**Fedora** は **Red Hat** の商用版のバージョンよりはるかに多くのソフトウェアを含んでいます。その理由の 1 つは **Fedora** が 1 つの会社ではなく多様なコミュニティによって開発されているからです。

Fedora コミュニティは、半年に一度新しいバージョンをリリースしています。そのため教材の **Red Hat** / **Fedora** の部分ではリリースサイクルが長い **CentOS**/**CentOS Stream** の最新のバージョンを元に標準化することにしました。インストールしてしまえば **CentOS Stream** はエンタープライズ環境で最もポピュラーな **Linux** ディストリビューションである **Red Hat Enterprise Linux** (**RHEL**) とほぼ同じです。

CentOS と CentOS Stream

- **CentOS** は歴史的に見ても、基本的には RHEL のコピーで、アップデート後に多少の時間差があります。
- **CentOS Stream** は **RHEL** より **先に** アップデートが行われますが、それ以外は **RHEL** にかなり近い状態です。そのため、より新しい機能をいち早く吸収することができます。
- **Red Hat** は 2021 年末に **CentOS 8** のサポートを終了します。したがって、本コースは現在、**CentOS Stream** ディストリビューションでテストされており、**CentOS 8** や **RHEL 8** との差異は軽微であり気づくこともないはずです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

openSUSE 系

- 教材は、出版時での最新の **openSUSE** のリリースに基づく
- RPM ベースで、インストールやアップデートには **zypper** を使用する
- 管理者用に **YaST** も存在する
- **x86** と **x86-64**
- **SUSE Linux Enterprise Server (SLES)** の元となったディストリビューションである

注目

デモや演習では無料の **openSUSE** を使用しています。

openSUSE と **SUSE Linux Enterprise Server** の関係は、前述の **Fedora** と **Red Hat Enterprise Linux** の関係と似ています。しかし **SUSE Linux Enterprise Server** の無料版を見つけることが困難なため **openSUSE** 系の部分では **openSUSE** を元にすることにしました。この 2 つの商品は非常によく似ていて **openSUSE** に関する教材は、ほとんどの場合問題なく **SUSE Linux Enterprise Server** にも対応します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Debian 系

- サーバやデスクトップパソコンでの利用が多い
- **DPKG** ベースで、インストールやアップデートには **APT** とフロントエンドアプリを使用する
- **Ubuntu**・**Linux Mint** などの元となったディストリビューションである
- 教材は、最新の **Ubuntu** のリリースに基づく
- **x86**、**x86-64**、その他のアーキテクチャをサポートする
  - 長期保守版（Long Term Release = LTS)

注目

デモや演習では **Ubuntu** を使用しています。**Debian** も無償で利用可能ですが **Ubuntu** の方が新規 **Linux** ユーザーに多く使われているためです。

Debian ディストリビューションは、**Ubuntu** や **Linux Mint** を含むいくつかのディストリビューションの元となっています。**Debian** は純粋なオープンソースプロジェクトであり、安定性を重視しています。最も大きく完全なソフトウェアリポジトリをユーザーに提供します。一方 **Ubuntu** は使いやすさと長期的な安定性をうまくバランスすることを目標としています。**Ubuntu** はほとんどのパッケージを **Debian** の不安定版ブランチから取っているので **Ubuntu** でも非常に大きなソフトウェアリポジトリへアクセスできます。このような理由を元に **Debian** 系ディストリビューションに関しての演習では **Ubuntu** を使用することにしました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

最近のディストリビューション間の類似点

- ディストリビューションのトレンドの変化により、ディストリビューション間の違いは減少している
  - **systemd**: システムの起動や、サービス管理
  - **journald**: システムログの管理
  - **firewalld**: ファイアウォール管理デーモン
  - **ip**: ネットワークディスプレイ、設定ツール

注目

これらの機能は多くのディストリビューションに共通しているため、講座や演習はこれらの機能に基づいています。利用中のディストリビューションあるいはリリースがこれらのコマンドをサポートしていない場合は適宜読み替えてください。

systemd は **SysVinit** や **Upstart** パッケージに代わり、最も一般的なディストリビューションで使用されています。**service** や **chkconfig** コマンドを置き換えています。**journald** はログのデータの収集と格納を行う **systemd** のサービスです。さまざまなプロセスやアプリケーションから受け取ったログ情報を元に構造化された索引付きのジャーナルを作成・管理します。ディストリビューションによりテキストベースのシステムログが置き換えられている可能性があります。

firewalld は、ネットワークあるいはファイアウォールゾーンがネットワーク接続やインターフェイスの信頼レベルを定義できるようにする、動的に管理できるファイアウォールを提供します。IPv4 や IPv6 のファイアウォール設定やイーサネットブリッジをサポートしています。これは **iptables** 設定を置き換えるものです。

ip コマンドは、**net-tools** パッケージの一部であり、**ifconfig** コマンドを置き換えるものです。**ip** コマンドは、ルーティング・ネットワークデバイス・ルーティングの情報とトンネルの表示と操作をします。こちらの資料は古いコマンドを相当する **systemd** のコマンドに書き換える手助けとなるでしょう。

https://fedoraprojectorg/wiki/SysVinit_to_Systemd_Cheatsheet
https://wiki.debian.org/systemd/CheatSheet
https://en.opensuse.org/openSUSE:Cheat_sheet_13.1#Services

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

AWS 無料利用枠

- **Amazon Web Services** (**AWS**) はリモートユーザーがクラウド上でアクセスできる幅広い仮想マシン商品 (**インスタンス**) を提供
- **無料枠**のアカウントレベルが 1 年間利用可能
  Linux に慣れ **Linux Foundation** の講座・演習を行う上で必要な仮想ハードウェアとソフトウェアを利用可能
  ２つ以上の **Linux** ディストリビューションを学習する機会を得られる
  AWS の無料利用枠を試用する上で役に立つ説明
  https://training.linuxfoundation.org/cm/prep/aws.pdf
- **AWS** はコンソールへのアクセスができず GUI を提供していないので、これらの機能を必要とするタスクが実行できない可能性に注意
  特にカーネルレベルのトレーニングは困難である

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

1.9 演習

課題 1.1: sudo 用にシステムを設定

root シェル の実行は非常に危険です: 1 つのミスタイプや間違いで重大な（致命的な）損害を与えてしまう可能性があります。絶対に必要なとき以外は控えるべきです。

そこで、**sudo** メカニズムを用いて 1 つずつのコマンドをスーパーユーザーの権限で実行できるように設定するのが賢明です。**sudo** を使えば、ユーザーは自分のパスワードだけを知っていればよいので、root パスワードを知る必要はありません。

この演習は、本講座用に **sudo** を設定するものです。そのため、**Ubuntu** のようなディストリビューションを利用しているならこの演習は必要ないかもしれません。しかし、手順は知っておくとよいでしょう。

利用しているシステムが既にユーザーアカウントから **sudo** が実行できるように設定されているかどうかを確認するには、次の単純なコマンドを実行します:

```
$ sudo ls
```

ユーザーパスワードを求められコマンドが実行されるはずです。もし代わりにエラーメッセージが表示されれば、次の手順を行ってください。

su と入力し、自分のユーザーパスワードではなく **root** パスワードを与えることにより root シェルを起動します。

最近の全ての **Linux** ディストリビューションでは、`/etc/sudoers.d` サブディレクトリへ移動し、**sudo** アクセスを与えようとしているユーザーの名前でファイルを作成するべきです。しかし、この手順は実際は必要のない手順です。なぜなら、**sudo** がこのディレクトリ内の全てのファイルを必要に応じてスキャンするからです。ファイルには次の一文が含まれていれば充分です:

```
student ALL=(ALL)  ALL
```

前述の例はユーザーがstudent の場合です。

（今でも利用できる）古いやり方は `/etc/sudoers` ファイルにそのような一行を追加する方法です。これには、文法エラーを防いでくれる **visudo** プログラムを利用することを推奨します。

また、次の文を入力することによってファイルに適切なパーミッションを設定する必要もあるでしょう:

```
$ chmod 440 /etc/sudoers.d/student
```

（ディストリビューションによっては、パーミッションを 440 の代わりに 400 と設定する必要があるかもしれません）

これらの手順が終わったら `exit` と入力し、root シェルからログアウトしてからもう一度 `sudo ls` を試しましょう。

特定のユーザーに必要なパーミッションを付与することや、検索パスの限定など管理者が **sudo** を設定する方法は他にもあります。`/etc/sudoers` ファイルは初めて触る人でもファイル自身が良く書かれているので読むだけで内容がわかるものです。

しかし、システムに **sudo** が既に設定されている場合にも設定して頂きたい設定が 1 つあります。多くのディストリビューションでは、一般ユーザーと root ユーザーとでは実行ファイルを見つけるためのパスが違います。特に、**sudo** は root ユーザーではなく一般のユーザーのPATH を受け継ぐので、`/sbin` ディレクトリと `/usr/sbin` ディレクトリは検索対象に含まれません。

この場合だと、多くのツール・コマンドへのフルパスを常にこの講座でお伝えしなければいけないことになります。これほどの入力量の増加とディレクトリの位置を探し出す手間は、セキュリティの強化のためとはいえあまりに大きな手間です。従って、ホームディレクトリにある `.bashrc` ファイルに次の一行を追加することをおすすめします。

```
PATH=$PATH:/usr/sbin:/sbin
```

`source ~/.bashrc` を実行するか、一度ログアウトして改めてログインすれば設定が有効になります。（リブートの必要はありません）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 2 章

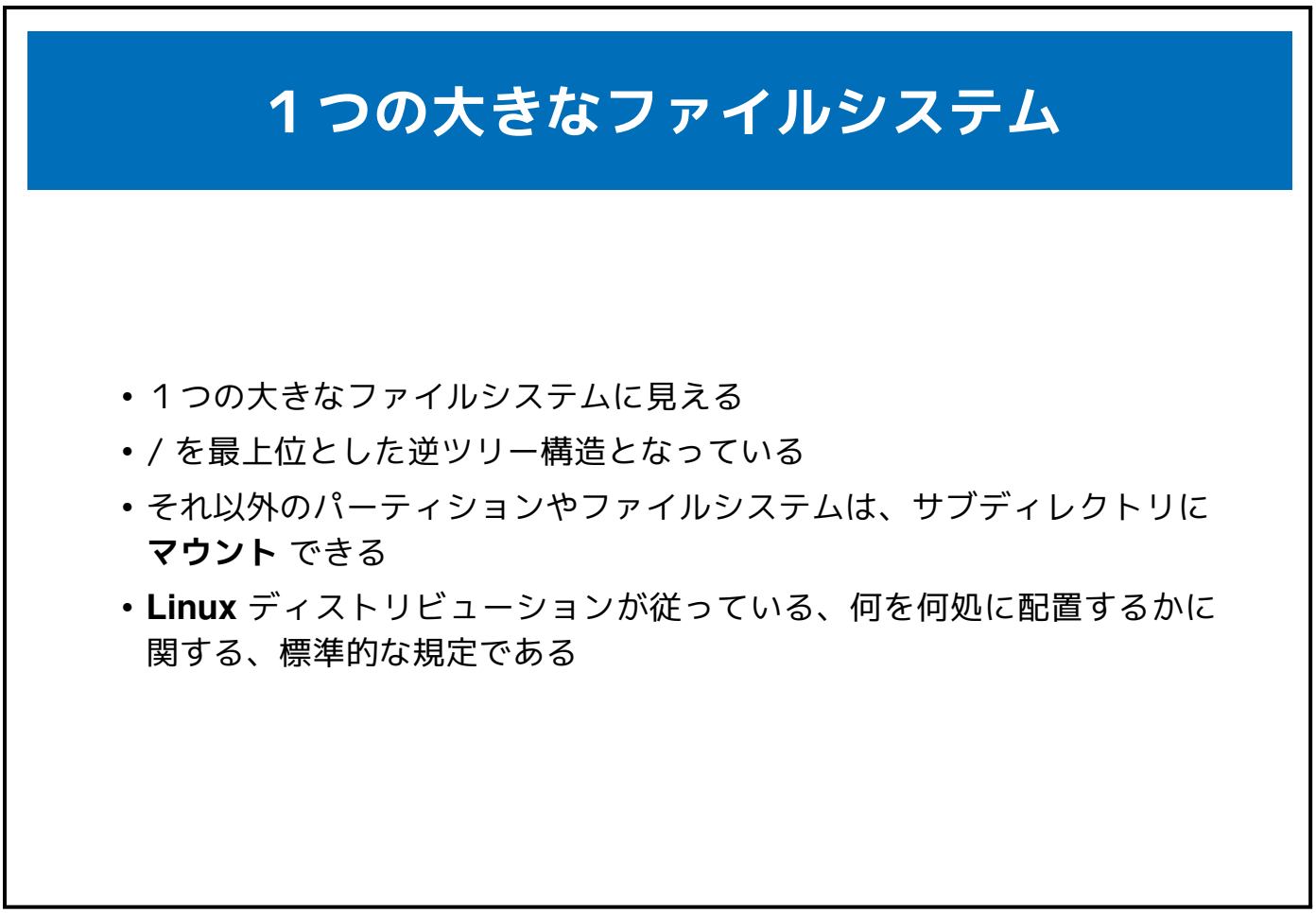
Linux ファイル システム階層構造

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- Linux が 1 つの大きなファイルシステム ツリーの構成を必要とする理由と、その方法に関する主な考慮事項を説明する。
- Filesystem Hierarchy Standard (**FHS**) が果たす役割について説明する。
- 起動時にルート（/）ディレクトリで利用可能でなければならないものと、システムが起動した後にだけ利用可能なものを説明する。
- 主なサブディレクトリの各ツリーを探索し、その目的を説明し、内容を検討する。

2.1 １つの大きなファイルシステム



Linux は全ての UNIX ベースのオペレーティング システムと同様に、１つの大きなファイルシステムツリー（階層）で構成されています。実際にはルートディレクトリ / がツリー（階層）の最上位にある逆ツリーの形で図示されることが一般的です。

この１つの大きな論理ファイルシステム内には、複数の異なるファイルシステムが複数のポイントで **マウント** され、サブディレクトリとして表示される場合があります。これらの個別のファイルシステムは、通常は異なるパーティションにあり、ネットワーク上のデバイスを含む任意の数のデバイスに存在します。

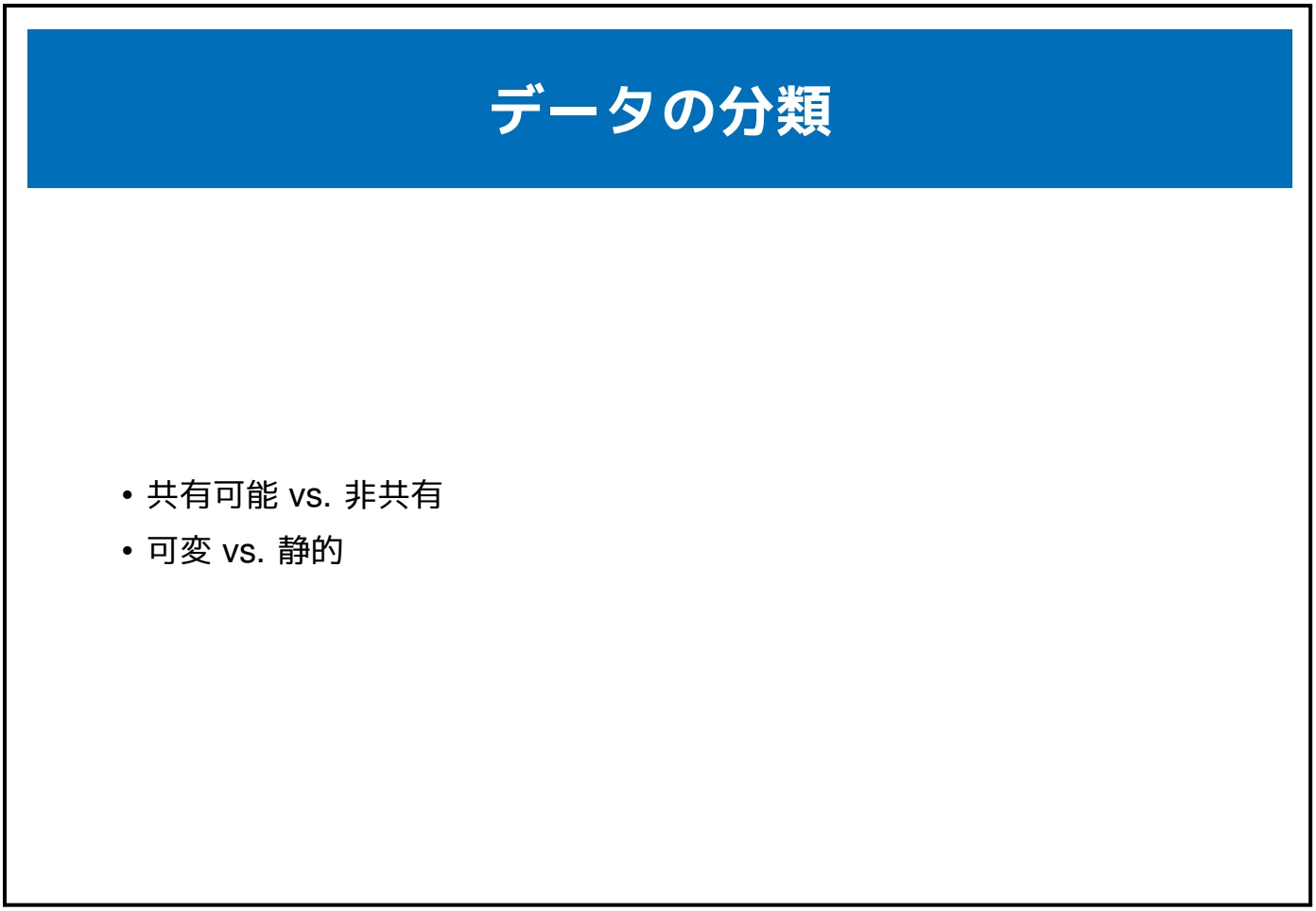
どのように結合されているかに関わらず、全てが１つの大きなファイルシステムのように見えます。アプリケーションは実際にどの物理デバイスファイルが存在するかについてまったく気にしません。

昔は、さまざまな **UNIX** 系のオペレーティング システムがこの１つの大きなツリー（階層）をさまざまな方法で編成しました。**Linux** ディストリビューションの間でも多くの違いがありました。このため、アプリケーションの作成と複数の種類のシステムでのシステム管理タスクの実行の両方が困難になり、イライラすることが多くありました。

結果として **Linux** エコシステムは、そのような苦労を最小限に抑えるための標準化手順の確立に努めてきました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.2 データの分類



1 つの大きなディレクトリツリーでファイルとデータをどのように整理するかにおいては、どの種類の情報を読み書きする必要があるかという分類が重要になります。大きくは 2 種類あります。

1. **共有可能 vs. 非共有**
   共有可能なデータとは異なるホスト間で共有できるデータです。共有できないデータとは特定のホストに固有なデータです。たとえばユーザーのホームディレクトリは共有可能ですが、デバイスロックファイルは共有できません。

2. **可変 vs. 静的**
   静的データにはバイナリ、ライブラリ、ドキュメント、そしてシステム管理者だけが変更できるデータが含まれます。可変データはシステム管理者以外でも変更できるデータです。

これらの論理的な違いは、いろいろなディレクトリ、パーティション、ファイルシステムに存在するさまざまな種類の情報で表わされます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.3 FHS Linux 標準ディレクトリツリー

標準ファイルシステム階層構成

- 標準的なディレクトリ構成や、その内容を定めたものである
  - ファイルの場所を予測しやすくする
- BSD、その他の歴史的な標準配置がベースとなっている
- **Linux Foundation** によって管理されている
  https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf

Filesystem Hierarchy Standard（FHS）は当初 **Free Standards Group** が管理していましたが、現在は The **Linux Foundation** が管理しています。FHS は主なディレクトリとその内容を定義しています。

FHS のドキュメントは https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf から取得できます。

FHS は標準レイアウトを定義することによりファイルの場所を予測し易くしています。ほとんどの **Linux** ディストリビューションは **FHS** を尊重しますが正確には従っていません。最後の公式バージョンはとても古くいくつかの新しい開発を考慮していないためです。

いくつかのディストリビューションは実験を好み、最終的には実験の一部を受け入れています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

FHS Linux ディレクトリツリー

Table 2.1: **主なディレクトリ　パート 1**

ディレクトリ	目的
/	ファイルシステムの階層全体の最上位ディレクトリ
/bin	**シングルユーザー モード** で使用できる必要がある必須の実行可能プログラム
/boot	カーネル、**initrd**、**initramfs** イメージなどのシステムのブートに必要なファイル、ブート構成ファイルおよびブートローダープログラム
/dev	ハードウェアおよびソフトウェアデバイスと対話するために使用される **デバイスノード**
/etc	システム全体の構成ファイル
/home	個人の設定、ファイルなどを含むユーザーの **ホームディレクトリ**
/lib	/bin と /sbin の実行可能バイナリに必要なライブラリ
/lib64	32 ビットプログラムと 64 ビットプログラムの両方を実行できるシステムの場合の /bin と /sbin の実行可能バイナリに必要な 64 ビット ライブラリ
/media	**CD**、**DVD**、**USB** スティックなどのリムーバブルメディアのマウントポイント
/mnt	一時的にマウントされたファイルシステム

ルートディレクトリの下に、ディストリビューション固有のディレクトリが追加されている場合があります。これらには、その他のデータに使用できる /misc や、**tftp** を使用した起動に使用される /tftpboot が含まれる場合があります。ディレクトリにファイルがある場合、それらはディスクレスワークステーションの起動に関連しています。他のディレクトリを持つことは **FHS** には違反しません。ただし標準で規定されているディレクトリ以外にコンポーネントを配置することには違反します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

FHS Linux ディレクトリツリー（つづき）

Table 2.2: **主なディレクトリ　パート 2**

ディレクトリ	目的
/opt	オプションのアプリケーションソフトウェアパッケージ
/proc	システムおよびその上で実行されているプロセスに関する情報を提供する仮想擬似ファイルシステム、システムパラメータの変更に使用可能
/run	実行時の可変データ、システムがブートしてからの情報が含まれている 旧 /var/run の代わり
/sys	システムおよびそこで実行されているプロセスに関する情報を提供する仮想擬似ファイルシステム、システムパラメータ変更に使用可能。**デバイスツリー**に似ており **統合デバイスモデル** の一部
/root	root ユーザーのホームディレクトリ
/sbin	必須のシステムバイナリ
/srv	システムが提供するサイト固有のデータ、めったに使用されない
/tmp	一時ファイル、多くのディストリビューションでは再起動すると失われるメモリー内の RAM ディスクである可能性もある
/usr	マルチユーザーアプリケーション、ユーティリティおよびデータ、読み取り専用
/var	システム操作中に変化する可変データ

これらは、メインの **root** (/) ディレクトリとは別の追加ディレクトリです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.4 ルート（/）ディレクトリ

ルート（/）ディレクトリ

- 以下のシステム要求に対応する
  - Boot（起動）
  - Restore（修復）
  - Recover（回復）
  - Repair（改修）
- アプリケーションまたはパッケージが、ルートディレクトリに新しいサブディレクトリを作成することは禁止されている

/ と /root

ここでは **スーパーユーザー** である **root** のホームディレクトリの `/root` ディレクトリについて説明しているわけではありません。

先ほど述べたように、ファイルシステム全体を 1 つの大きなツリー（階層）と見なすことができますが、複数のパーティションとファイルシステムが結合されている場合があります。

ルートディレクトリ自体が含まれるパーティションとファイルシステムはかなり特別です。多くの場合、後でマウントされる `/home`、`/var`、`/opt` などのディレクトリを持つ他のコンポーネントとともに特別な専用パーティションにあります。

ルート パーティションには、システムの起動後に他の全てのファイルシステムをマウントするために必要な、全ての重要なファイルが含まれている必要があります。したがって、ユーティリティ、構成ファイル、ブートローダー情報、およびその他の重要なスタートアップデータが含まれている必要があります。それは次のことを適切に行うためです。

- システムを起動します。
- テープやその他のリムーバブルメディア、**NAS** などの外部メディアのシステムバックアップからシステムを復元します。
- システムの回復か修復、またはその両方を行います。経験豊富なメンテナーは損傷したシステムを診断および再構築するツールを持っている必要があります。

FHS はアプリケーションまたはパッケージがルートディレクトリに新しいサブディレクトリを作成することを禁止しています。

/ は /root とは違います

`/root` はスーパーユーザーのホームディレクトリです。名前が紛らわしいので注意してください！

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.5 /bin

/bin

- システム管理者と非特権ユーザーの両方が必要とする、実行可能プログラムとスクリプトが含まれる。これらは他のファイルシステムがまだマウントされていない場合、たとえば **シングルユーザー モード** または、リカバリモードで起動する場合に必要
- スクリプトによって、間接的に使用される実行可能ファイルが含まれる場合もある
- サブディレクトリを含めることはできない

/bin と /usr/bin

最新のいくつかのディストリビューションは /bin と /usr/bin（および/sbinと/usr/sbin）を区別せず、シンボリックリンクを使って１つのディレクトリを２つのディレクトリのように見せています。そのディストリビューションでは、ブート後にマウントされる別のパーティションに /usr を配置するのは時代遅れであるとみなしています。

/bin に存在する必要があるプログラムには、次のものがあります。

cat、chgrp、chmod、chown、cp、date、dd、df、dmesg、echo、false、hostname、kill、ln、login、ls、mkdir、mknod、more、mount、mv、ps、pwd、rm、rmdir、sed、sh、stty、su、sync、true、umount、uname

[と **test** もここに置かれることがあります。オプションで以下のものも含まれることがあります。

csh、ed、tar、cpio、gunzip、zcat、netstat、ping

/bin の場所に値するほど重要ではないとみなされるコマンドバイナリは /usr/bin に置かれます。
ここには、非 root ユーザーだけが必要とするプログラムが置かれます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/bin の例

```
student@ubuntu:~$ ls /bin
bash                chacl          fsck.btrfs  lowntfs-3g      ntfscmp         run-parts                    tempfile
brltty              chgrp          fuser       ls              ntfsfallocate   sed                          touch
btrfs               chmod          fusermount  lsblk           ntfsfix         setfacl                      true
btrfsck             chown          getfacl     lsmod           ntfsinfo        setfont                      udevadm
btrfs-debug-tree    chvt           grep        mkdir           ntfsls          setupcon                     ulockmgr_server
btrfs-find-root     cp             gunzip      mkfs.btrfs      ntfsmove        sh                           umount
btrfs-image         cpio           gzexe       mknod           ntfsrecover     sh.distrib                   uname
btrfs-map-logical   dash           gzip        mktemp          ntfssecaudit    sleep                        uncompress
btrfs-select-super  date           hciconfig   more            ntfstruncate    ss                           unicode_start
btrfstune           dd             hostname    mount           ntfsusermap     static-sh                    vdir
btrfs-zero-log      df             ip          mountpoint      ntfswipe        stty                         wdctl
bunzip2             dir            journalctl  mt              open            su                           which
busybox             dmesg          kbd_mode    mt-gnu          openvt          sync                         whiptail
bzcat               dnsdomainname  keyctl      mv              pidof           systemctl                    ypdomainname
bzcmp               domainname     kill        nano            ping            systemd                      zcat
bzdiff              dumpkeys       kmod        nc              ping4           systemd-ask-password         zcmp
bzegrep             echo           less        nc.openbsd      ping6           systemd-escape               zdiff
bzexe               ed             lessecho    netcat          plymouth        systemd-hwdb                 zegrep
bzfgrep             efibootdump    lessfile    netstat         ps              systemd-inhibit              zfgrep
bzgrep              efibootmgr     lesskey     networkctl      pwd             systemd-machine-id-setup     zforce
bzip2               egrep          lesspipe    nisdomainname   readlink        systemd-notify               zgrep
bzip2recover        false          ln          ntfs-3g         red             systemd-sysusers             zless
bzless              fgconsole      loadkeys    ntfs-3g.probe   rm              systemd-tmpfiles             zmore
bzmore              fgrep          login       ntfscat         rmdir           systemd-tty-ask-password-agent znew
cat                 findmnt        loginctl    ntfscluster     rnano           tar
student@ubuntu:~$
```

図 2.1: **/bin ディレクトリ: Ubuntu 18.04**

RHEL、**CentOS**、**Fedora** や **Ubuntu** などの最近のディストリビューションでは、/bin と /usr/bin はシンボリックリンクで結ばれており、実際には同じディレクトリです。

上のスクリーンショットは **Ubuntu 18.04** で取得したものです。以降のバージョンにはリンクされていない独立したディレクトリは無くなっています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION | Training & Certification

2.6 /boot

/boot

- システムの起動に必要な全てのファイルが含まれる、
  以下の 2 つのファイルは特に重要である
  - `vmlinuz`: 圧縮された **Linux** カーネル
  - `initramfs`: 実際のルートファイルシステムが使用可能になる前に
    マウントされる **初期 RAM ファイルシステム**
- カーネルがユーザーモードプログラムの実行を開始する前に使用する
  データを格納する
- また、情報とデバッグに使用される 2 つのファイルも含まれる
  - `config` カーネルのコンパイルを構成するために使用する
  - `System.map`: デバッグ用のカーネル **シンボルテーブル**
- ディストリビューションによって、他のファイルが置かれることもある

`/boot` の正確な内容はディストリビューションとリリースされた時期によって異なります。以下に **Ubuntu 20.04** の例を示します。

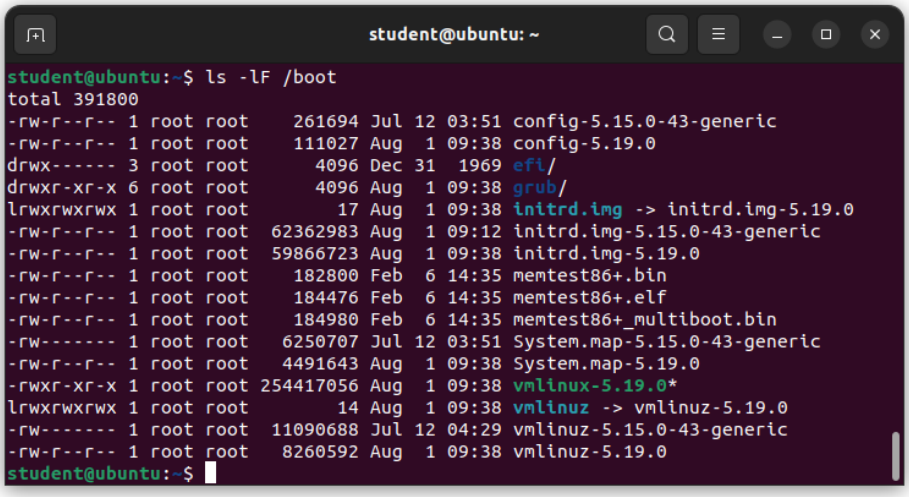


図 2.2: **Ubuntu 20.04 の /boot ディレクトリ**

これらのファイルは **Linux** ディストリビューションとカーネルバージョンに依存した長い名前を持っています。更に `initramfs` の代わりに `initrd` (**init**ial **r**am **d**isk) を持っている場合もあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.7 /dev

/dev

- キャラクター型／ブロック型の **デバイスノード** またはファイル
- 適切なシステム運用の要である
- 現代のシステムは **udev** でデバイスファイルの管理を自動化する
- 古いシステム（または組み込み機器）では必要に応じインストール時、または別の時間に **MAKEDEV** や **mknod** を使ってデバイスを生成する

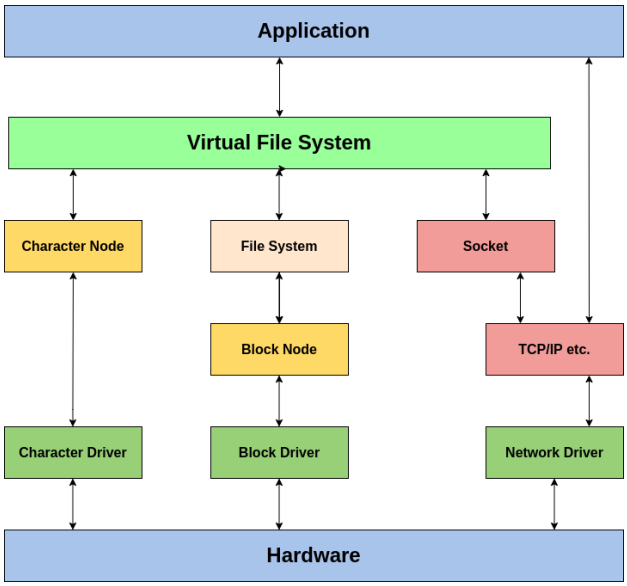


図 2.3: **デバイスノード**

このディレクトリにはシステムに組み込まれた、または、システムに接続されたデバイスを表す **スペシャル デバイス ファイル**（**デバイスノード** とも呼ばれます）が含まれています。これらの特殊ファイルはシステムが正しく機能するために不可欠です。このようなデバイスファイルには **キャラクター デバイス**（バイトストリーム）と **ブロック I/O デバイス** があります。**Linux** の **ネットワーク デバイス** にはデバイスノードがなく、代わりに `eth1` や `wlan0` などの名前で参照されます。

```
File  Edit  View  Search  Terminal  Help
c7:/tmp>ls -l /dev
total 0
....
crw-------   1 root root        5,   1 May 26 07:05 console
....
lrwxrwxrwx   1 root root             13 May 26 07:04 fd -> /proc/self/fd
....
brw-rw----   1 root disk        7,   0 May 26 07:05 loop0
crw-rw----   1 root disk       10, 237 May 26 07:05 loop-control
.....
crw-rw----   1 root lp          6,   0 May 26 07:05 lp0
crw-rw----   1 root lp          6,   1 May 26 07:05 lp1
....
brw-rw----   1 root disk        8,   0 May 26 07:05 sda
brw-rw----   1 root disk        8,   1 May 26 07:05 sda1
brw-rw----   1 root disk        8,   2 May 26 07:05 sda2
brw-rw----   1 root disk        8,   3 May 26 07:05 sda3
....
lrwxrwxrwx   1 root root             15 May 26 07:04 stderr -> /proc/self/fd/2
lrwxrwxrwx   1 root root             15 May 26 07:04 stdin -> /proc/self/fd/0
lrwxrwxrwx   1 root root             15 May 26 07:04 stdout -> /proc/self/fd/1
crw-rw-rw-   1 root tty         5,   0 May 26 07:05 tty
crw--w----   1 root tty         4,   0 May 26 07:05 tty0
....
c7:/tmp>
c7:/tmp>
```

図 2.4: **/dev ディレクトリ**

最新の **Linux** ディストリビューションは全て **udev** システムを使用し、デバイスが見つかったときだけ必要に応じて `/dev` にファイルを作成します。マウントされていないファイルシステムの `/dev` ディレクトリを見ると空であることがわかります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.8 /etc

このディレクトリには、マシンローカルの構成ファイルといくつかの起動スクリプトが含まれています。実行可能なバイナリプログラムはありません。このディレクトリには、以下のファイルやディレクトリが含まれているでしょう。

```
csh.login、exports、fstab、ftpusers、gateways、gettydefs、group、host.conf、hosts.allow、hosts.deny、
hosts.equiv、hosts.lpd、inetd.conf、inittab、issue、ld.so.conf、motd、mtab、mtools.conf、networks、
passwd、printcap、profile、protocols、resolv.conf、rpc、securetty、services、shells、syslog.conf
```

フロッピーディスクで使用される `mtools.conf` など、この中の一部のファイルは現在重要ではなくなっています。**FHS** が定義していてもソフトウェアが古くなったためもう見つからないものもあります。

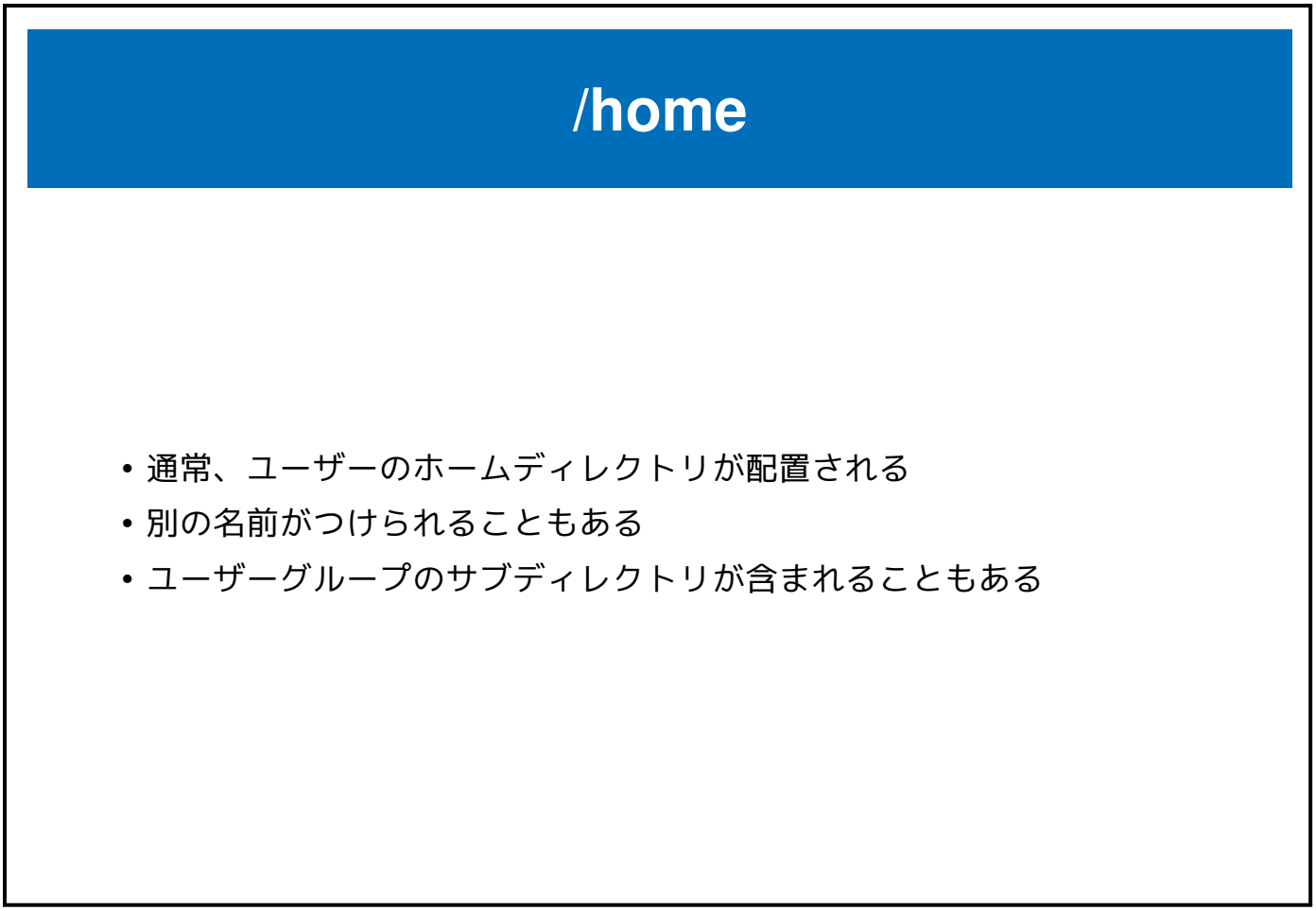
ディストリビューションは、`/etc` に構成ファイルとディレクトリを追加することがよくあります。たとえば **Red Hat** は `/etc/sysconfig` を含むいくつかのディレクトリを追加しています。そのディレクトリにはいくつかのシステム構成ファイルとディレクトリが存在します。

他の重要なサブディレクトリとしては

- `/etc/skel`: **雛形**ファイルが置かれています。このファイルは新しく作成したホームディレクトリ追加に使用されます。
- `/etc/systemd`: **systemd** を使用している時には、システムサービスを開始、停止、コントロールするための構成スクリプトの実体か、そのファイルへのシンボリックリンクが置かれています。
- `/etc/init.d`: **System V** の初期化を利用している場合の、起動、停止スクリプトが含まれています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.9 /home



Linux システムでは、通常、ユーザーディレクトリは `/home/coop`、`/home/student` などのように `/home` の下に置かれます。全ての個人設定、データ、および実行可能プログラムはこのディレクトリの階層に置かれます。`/home` には `/home/students`、`/home/staff`、`/home/aliens` などユーザーのさまざまなグループや関連付けされたサブディレクトリが置かれる場合もあります。

他の **UNIX** 系のオペレーティング システムでも `/home` ディレクトリツリーの概念は存在しますが微妙に異なる場合があります。たとえば、**Solaris** ではユーザーディレクトリは`/export/home` に作成され、最終的には **自動マウント** 機能によって `/home` にマウントされます。これは一般的にはホームディレクトリが企業ネットワーク上のどこか、おそらく **NFS** サーバー上にあり、使用時にホームディレクトリが自動的にマウントされるためです。

Linux にも同じ自動マウント機能がありますが、多くのユーザーはそれらを意識していません。また自己完結型システムではおそらく **NFS** マウントは使われないでしょう。

利用者は環境変数 `$HOME` をルートディレクトリの代わりに使用することも、略記 **〜(チルダ)** を使用することもできます。従って以下は同じ意味です。

```
$ ls -l $HOME/public_html
$ ls -l ~/public_html
```

例外が 1 つあります。**Linux** システムの **root** ユーザーのホームディレクトリは、常に `/root` の下にあります。古い **UNIX** システムの中には `/` を代わりに使用するものがあり混乱するかもしれません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/home の例

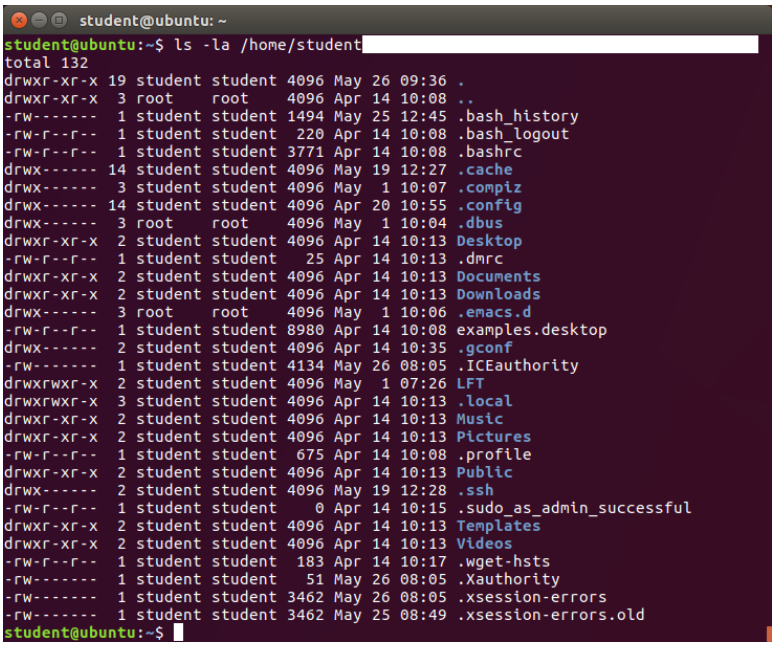


図 2.5: **/home ディレクトリ**

ここには、これまでに何回システムをアップグレードしたか、または **ホーム** ディレクトリを何回コピーしたかによって、さまざまな種類のファイルが置かれている可能性があります。

- （`~/Documents`、`~/Downloads`、`~/Pictures`、`~/Desktop` など）**XDG** で指定されたディレクトリにあるメディアとデータ
- **. で始まるファイル** や `~/.config` と `~/.local` のような新しい場所に置かれたプログラムのユーザーデータ
- `~/.xsession-errors` などのログファイル

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.10 /lib と /lib64

/lib と /lib64

- ここには /bin と /sbin にあるバイナリーを実行するのに必要なライブラリーだけが配置される
- カーネルモジュールは /lib/modules に配置される
- PAM モジュールは /lib/security に配置される
- いくつかのディストリビューションは 64bit ライブラリーを /lib64 に配置する

/lib と /usr/lib

最近のディストリビューションの一部には /lib と /usr/lib（および /lib64 と /usr/lib64 を）を区別せずにディレクトリを１つだけ用意してシンボリックリンクを使うことで、あたかも２つのディレクトリが存在するかのように見せているものがあります。それらのディストリビューションは、ブート後にマウントされる別のパーティションに /usr を置く昔からのコンセプトは時代遅れと考えています。

これらのディレクトリには、/bin と /sbin 内のバイナリを実行するために必要なライブラリだけが置かれています。これらのライブラリはシステムを起動しルート ファイルシステム内でコマンドを実行する上で特に重要なものです。

カーネルモジュール（デバイスやファイルシステムのドライバの多くが該当します）は /lib/modules/5.19.4 のような形で /lib/modules/<kernel-version-number> にあります。

PAM (**P**luggable **A**uthentication **M**odules) ファイルは /lib64/security や /lib/x86_64-linux-gnu/security などディストリビューション毎に異なる場所にあります。

32 ビットと 64 ビット両方のバイナリをサポートするシステムでは、システムに両方のライブラリを保持する必要があります。Red Hat ベースのシステムでは 32 ビットライブラリ（/lib）と 64 ビットライブラリ（/lib64）のディレクトリがあります。

```
$ ls -l /lib*
```

```
rwxrwxrwx 1 root root 8 Apr 23  2020 /sbin -> usr/sbin
lrwxrwxrwx 1 root root 7 Apr 23  2020 /lib -> usr/lib
lrwxrwxrwx 1 root root 9 Apr 23  2020 /lib64 -> usr/lib64
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.11 /media

/media

- リムーバブルメディアのマウント用
  - CD ROM
  - DVD
  - USB ドライブ
- 動的にマウントされることもある

/media と /run

最近のシステムでは /media ではなく /run/media/[username]/... を使う場合もある

このディレクトリは、ファイルシステムをリムーバブルメディアにマウントするために使用されていました。これらには **CD**、**DVD**、**USB**、さらには旧石器時代のフロッピーも含まれます。

Linux システムはそのようなメディアを挿入時に動的にマウントし、構成ファイルにある **udev** ルールで設定した名前で **udev** がディレクトリを作成して、そこにリムーバブル ファイルシステムをマウントします。アンマウントもしくは削除をすると、マウントポイントとして使用されていたディレクトリが消えます。

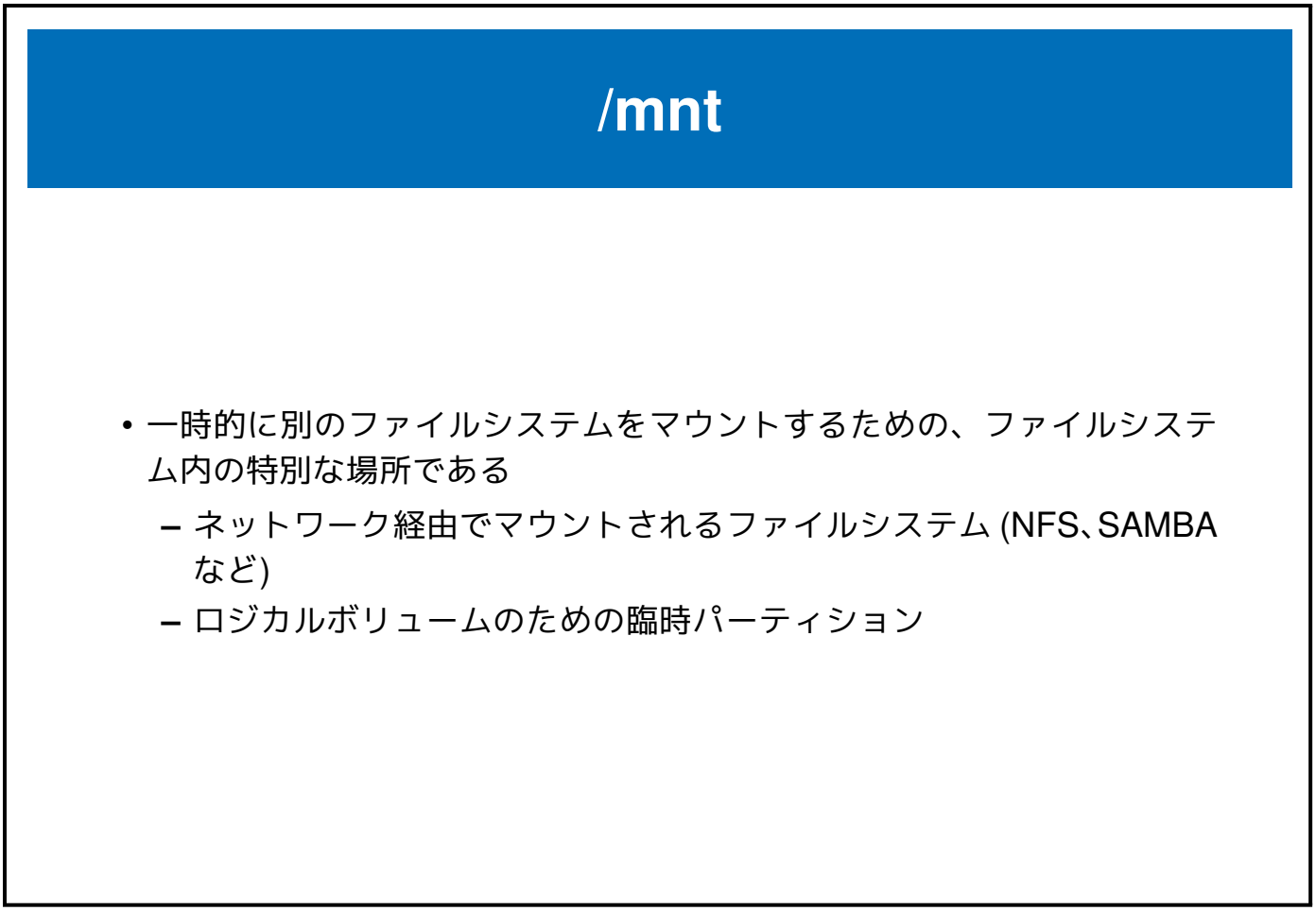
メディアに複数のパーティションとファイルシステムがある場合、複数のエントリが表示されます。多くの **Linux** ディストリビューションではメディアがマウントされるとファイルマネージャー (**Nautilus** など) が起動されます。

/media は /run に置き換わっています

留意: 現在の **Linux** ディストリビューションでは、リムーバブルメディアは /media ではなく /run/media/[username]/... の下に表示されます。/run については後で説明します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.12 /mnt



このディレクトリはシステム管理者が必要に応じて一時的にファイルシステムをマウントできるようにするために提供されています。一般的な用途は次のような **ネットワーク ファイルシステム** です。

- **NFS**
- **Samba**
- **CIFS**
- **AFS**

歴史的に `/mnt` は現在のシステムの `media`（または `/run/media`）にマウントされるファイルにも使用されていました。
ここにはプログラムをインストールしないでください。現在使用していない、別の一時ディレクトリが適切です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.13 /opt

/opt

- Add-on アプリケーション ソフトウエア パッケージのインストール用である
- パッケージのスタティックファイルは /opt/[somepackage] ディレクトリに置く必要がある

```
$ ls -l /opt
```

```
total 20K
drwxr-xr-x.  4 root root 4.0K Jul 26  2021 brother
drwxr-xr-x.  4 root root 4.0K Sep  1  2019 google
drwxr-xr-x   3 root root 4.0K Jul  7 08:03 rh
drwxr-xr-x  12 root root 4.0K Jul  1 09:55 VirtualBox
drwxr-xr-x. 30 root root 4.0K May 30 13:08 zoom
```

このディレクトリは他のソフトウェアと共有するディレクトリを使ってシステム全体に分散させるのではなく、ファイルの全てまたはほとんどを 1 つの専有場所に保持したいソフトウェアパッケージ用に設計されています。

たとえば **dolphy_app** が /opt に存在するパッケージの名前である場合、全てのファイルは /opt/dolphy_app の下のディレクトリに存在する必要があります。バイナリの場合は /opt/dolphy_app/bin、**man** ページの場合は /opt/dolphy_app/man などです。

このようにファイルをまとめることで、ソフトウェアのインストールとアンインストールの両方を比較的簡単に行うことができます。全てのファイルが、予測可能かつ構造化された 1 つの分離された場所にあるからです。また、システム管理者がパッケージ内の各ファイルの性質を簡単に判断できるようになります。

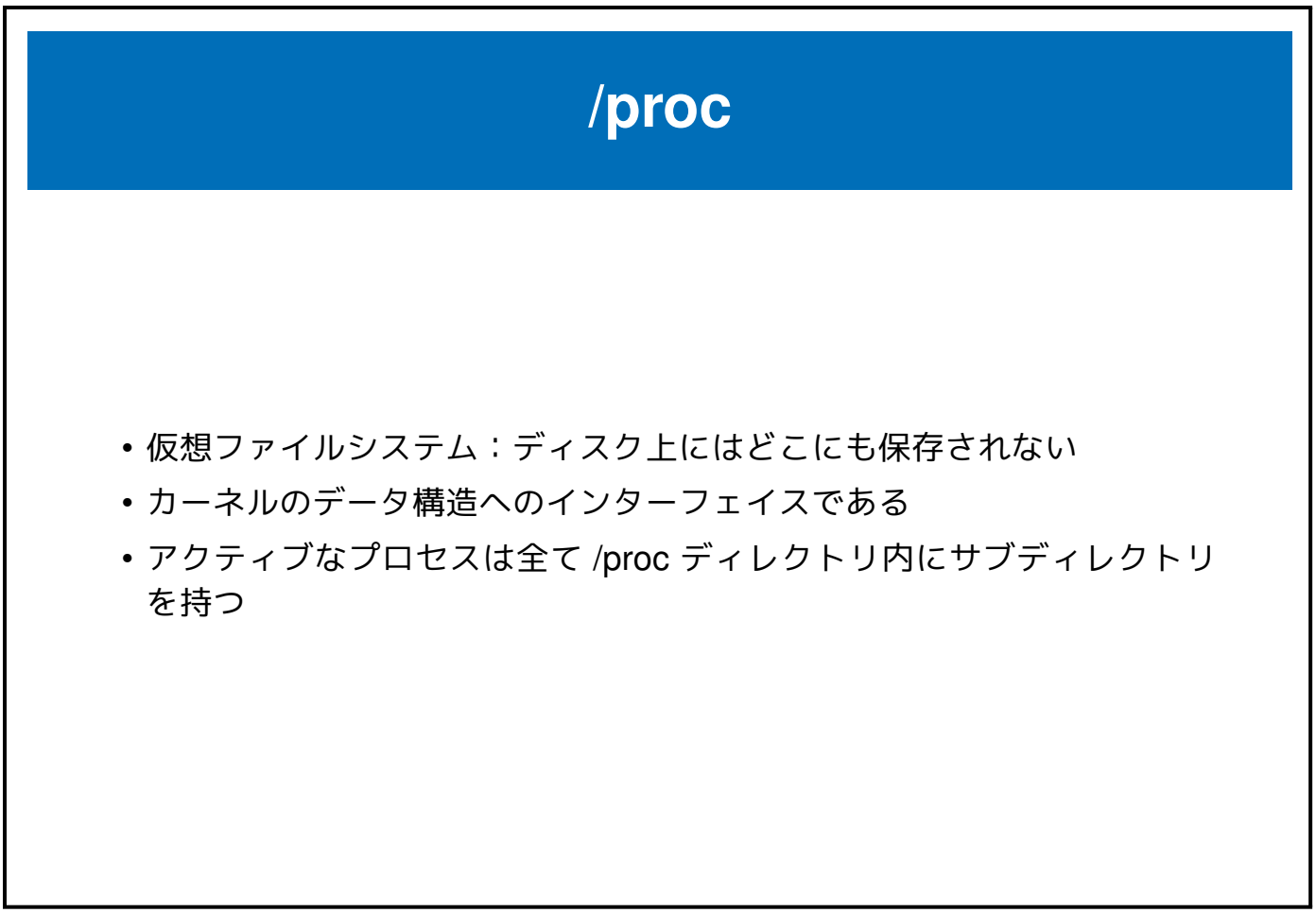
ただし **RPM** や **APT** などのパッケージシステムを使用している場合は、このようなことを意識しなくてもファイルの管理情報と配置が明確なので、簡単にインストールおよびアンインストールすることができます。

Linux では多くの場合、/opt ディレクトリはプロプライエタリなソフトウェアを使用するアプリケーションプロバイダ、またはディストリビューションの違いによる複雑さを回避したいアプリケーションプロバイダによって使用されます。たとえば **RHEL** システムでは、このディレクトリが用意され /opt/skype や /opt/google パッケージだけが置かれています。後者は chrome,earth のためのサブディレクトリを持っています。

また、ローカルシステム管理者が使用するためのディレクトリとして /opt/bin、/opt/doc、/opt/include、/opt/info、/opt/lib、/opt/man が用意されています。パッケージは、これら予約されたディレクトリにリンク、またはコピーされたファイルを提供する場合がありますが、一方で、これらの特別なディレクトリにプログラムが存在しなくても機能する必要があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.14 /proc



このディレクトリは、全ての情報がディスクではなくメモリーにだけ存在する **疑似ファイルシステム** のマウントポイントです。`/dev` と同様に、実行されていないシステムでは `/proc` ディレクトリは空です。

カーネルは `/proc` エントリを介していくつかの重要なデータ構造を提示します。さらに、システム上のアクティブな各プロセスには、プロセスの状態、使用しているリソース、およびその履歴に関する詳細情報を提供する独自のサブディレクトリがあります。

`/proc` 下に存在するエントリは通常 **仮想ファイル** と呼ばれ、興味深い性質を持っています。ほとんどがサイズが 0 バイトとして表示されますが、表示すると大量の情報を含んでいることがわかります。

さらに仮想ファイルのほとんどの日時設定は現在の日時を反映しており常に変化していることを示しています。実際これらのファイルの情報は表示するときにだけ取得され、常時もしくは定期的に更新されるわけではありません。

`/proc/interrupts`、`/proc/meminfo`、`/proc/mounts`、`/proc/partitions` を含む重要な疑似ファイルは、システムのハードウェアの最新情報を提供します。

`/proc/filesystems` や `/proc/sys/` などはシステム構成情報とインターフェイスを提供します。

組織化する目的で類似のトピックに関する情報を含むファイルは、仮想ディレクトリとサブディレクトリにグループ化されています。たとえば `/proc/scsi/` には全ての物理 **SCSI** デバイスの情報が含まれます。同様に **プロセス** ディレクトリにはシステムで実行中の各プロセスに関する情報が含まれています。

このコースでは `/proc` のエントリを見て行きます。カーネルの構成とシステムの監視に関する内容は、今後の章でさらに詳しく見ていきます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/proc の実例

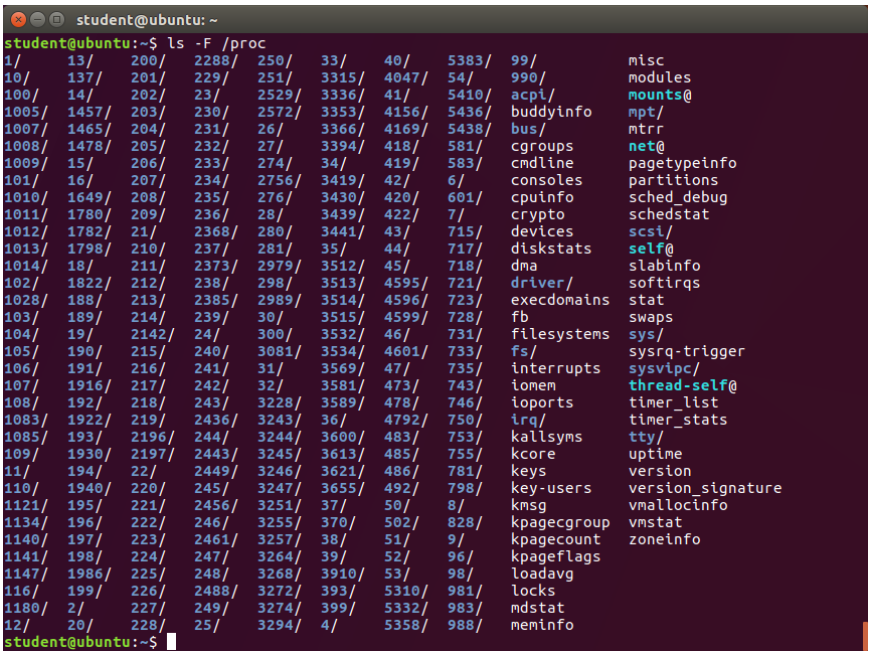


図 2.6: /**proc ディレクトリ**

以下のスクリーンショットでは、特定のプロセスに属する /proc ディレクトリの内容を確認できます。

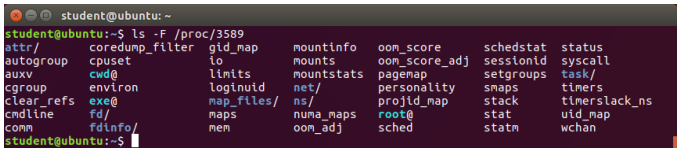


図 2.7: /**proc/[pid] ディレクトリ**

以下のスクリーンショットでは、重要なシステム情報を示す多くのファイルの 1 つである /proc/interrupts を確認できます。

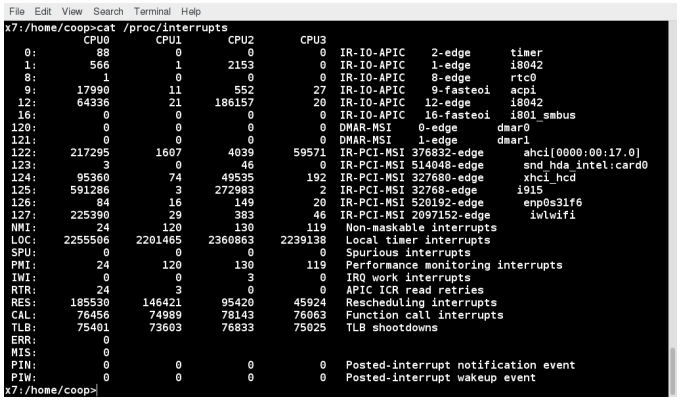


図 2.8: /**proc/interrupts の内容**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

2.15 /sys

/sys

- 別の仮想ファイルシステム：ディスク上にはどこにも保存されない
- デバイスとドライバー、カーネルモジュール、システム構成情報に関する情報が含まれる
- システムの挙動変更やパラメータ変更に利用される
- /proc よりうまくコントロールされている

このディレクトリは、全ての情報がディスクではなくメモリーにだけ存在する **sysfs 疑似ファイルシステム** のマウントポイントです。/dev や /proc と同様に、/sys ディレクトリは実行されていないシステムでは空です。デバイス、ドライバ、カーネルモジュール、システム構成構造などに関する情報が含まれています。

sysfs はシステムに関する情報を収集し、実行中にその動作を変更するために使用されます。その意味では /proc に似ていますが、それよりも歴史は新しくどのような種類のエントリを含めることができるかといった厳しい基準に準拠しています。たとえば /sys 内のほとんど全ての疑似ファイルには、1 行または値しか含まれていません。/proc にあるような長いエントリはありません。

/proc の場合と同様に、このコースでは /sys のエントリを見て行きます。カーネルの構成とシステム監視に関してはこの後の章で見て行きます。

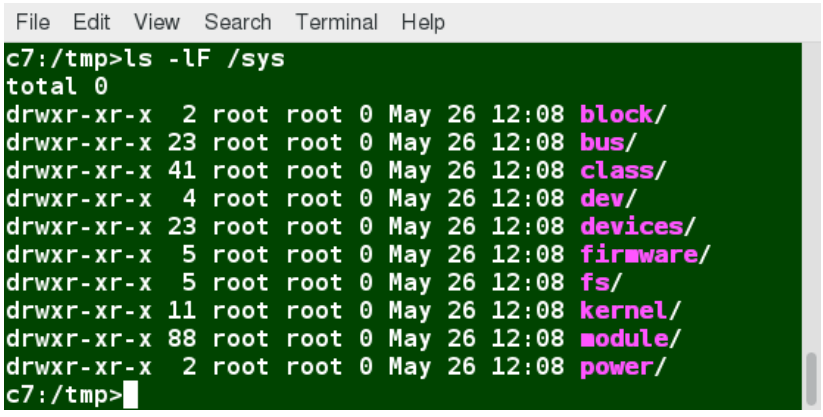


図 2.9: /sys ディレクトリ

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.16 /root

/root

- ルートユーザーのホームディレクトリである

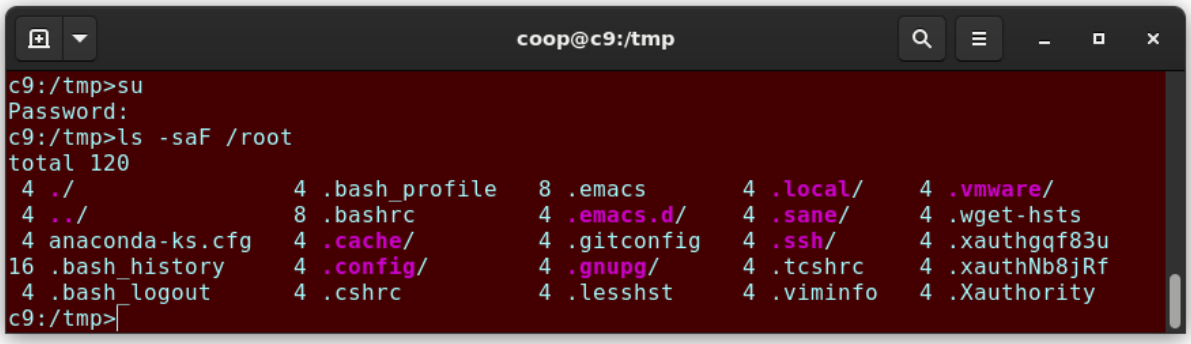


図 2.10: **/root ディレクトリ**

このディレクトリ（「スラッシュルート」と発音します）は root ユーザーのホームディレクトリです。

このディレクトリを所有する **root** アカウントは、スーパーユーザー権限が必要な作業の時にだけ使用してください。一般ユーザーとして実行する作業には、別のアカウントを使用します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.17 /sbin

/sbin

- システムの中核部分と保守ユーティリティ用である
- このディレクトリ内のプログラムは、特権ユーザーだけが利用できる
- このディレクトリ内の実行ファイルは (/bin にあるものに加え) 起動と /usr のマウントに使われる
- このディレクトリ内の実行ファイルは (/bin にあるものに加え) システムリカバリーに使われる

/sbin と /usr/sbin

前述したように、最近の一部のディストリビューションでは、/sbin と /usr/sbin を（/bin と /usr/bin と同様に）分離するという戦略を放棄し、１つのディレクトリに対してシンボリックリンクを配置することで２つのディレクトリビューをみせているのがあります。

このディレクトリには /bin ディレクトリ内のバイナリに加えて、起動、復元、回復、または修復、もしくはその全てに必須のバイナリが含まれています。ブート中にルート ファイルシステムが正常であると確認したら、必要に応じて /usr、/home、およびその他の場所に他のファイルシステムをマウントします。

次のプログラムはこのディレクトリにあります。（これらのサブシステムがインストールされている場合）

fdisk、fsck、getty、halt、ifconfig、init、mkfs、mkswap、reboot、route、swapon、swapoff、update

```
$ ls -l /sbin
```

```
rwxrwxrwx 1 root root 8 Apr 23  2020 /sbin -> usr/sbin
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/sbin の実例

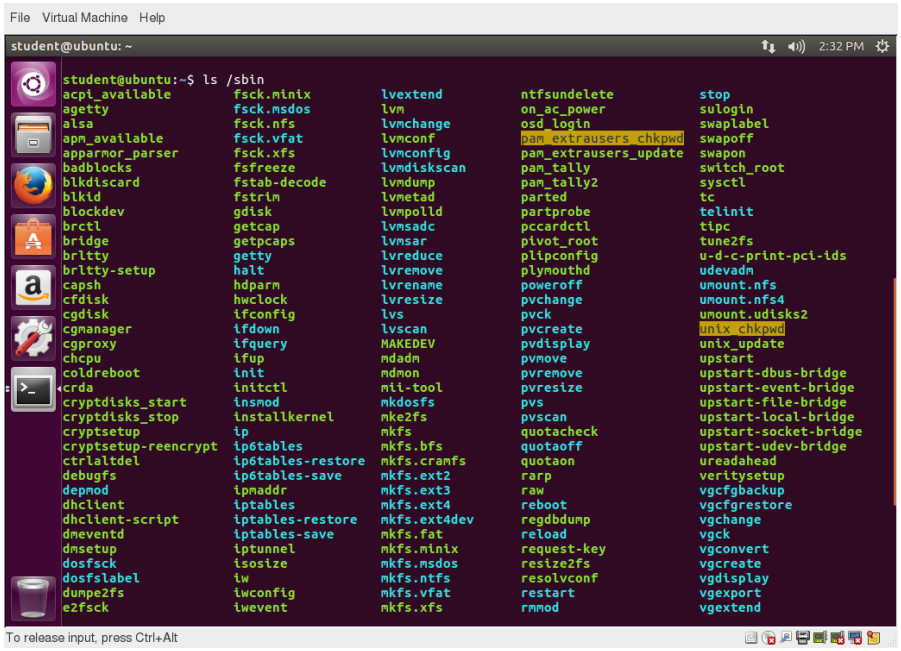
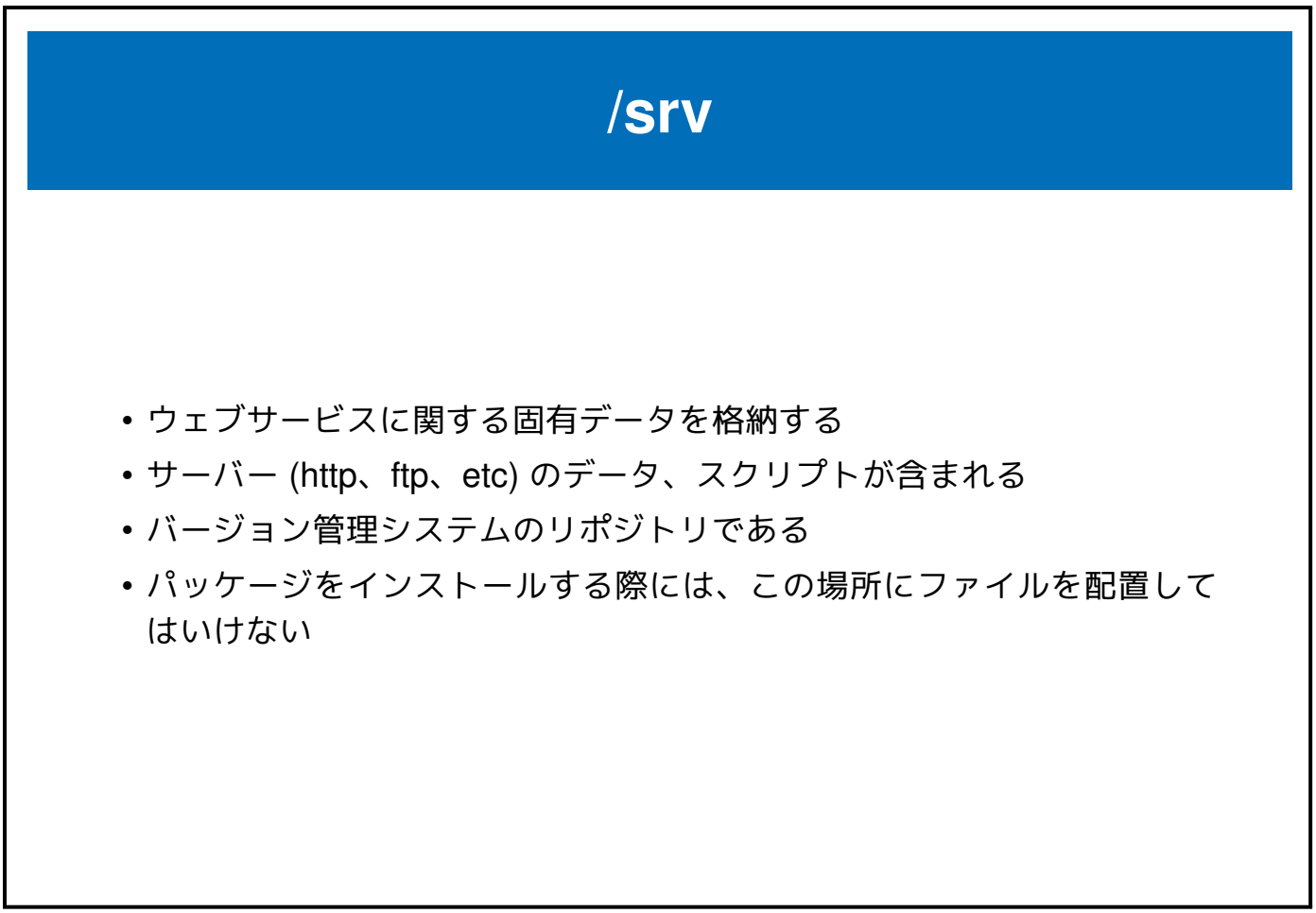


図 2.11: **/sbin ディレクトリ**

RHEL、**CentOS**、**Fedora**、**Debian** などの最近のバージョンでは `/sbin` と `/usr/sbin` は実際には同じものです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.18 /srv



FHS には、以下のように定義されています。

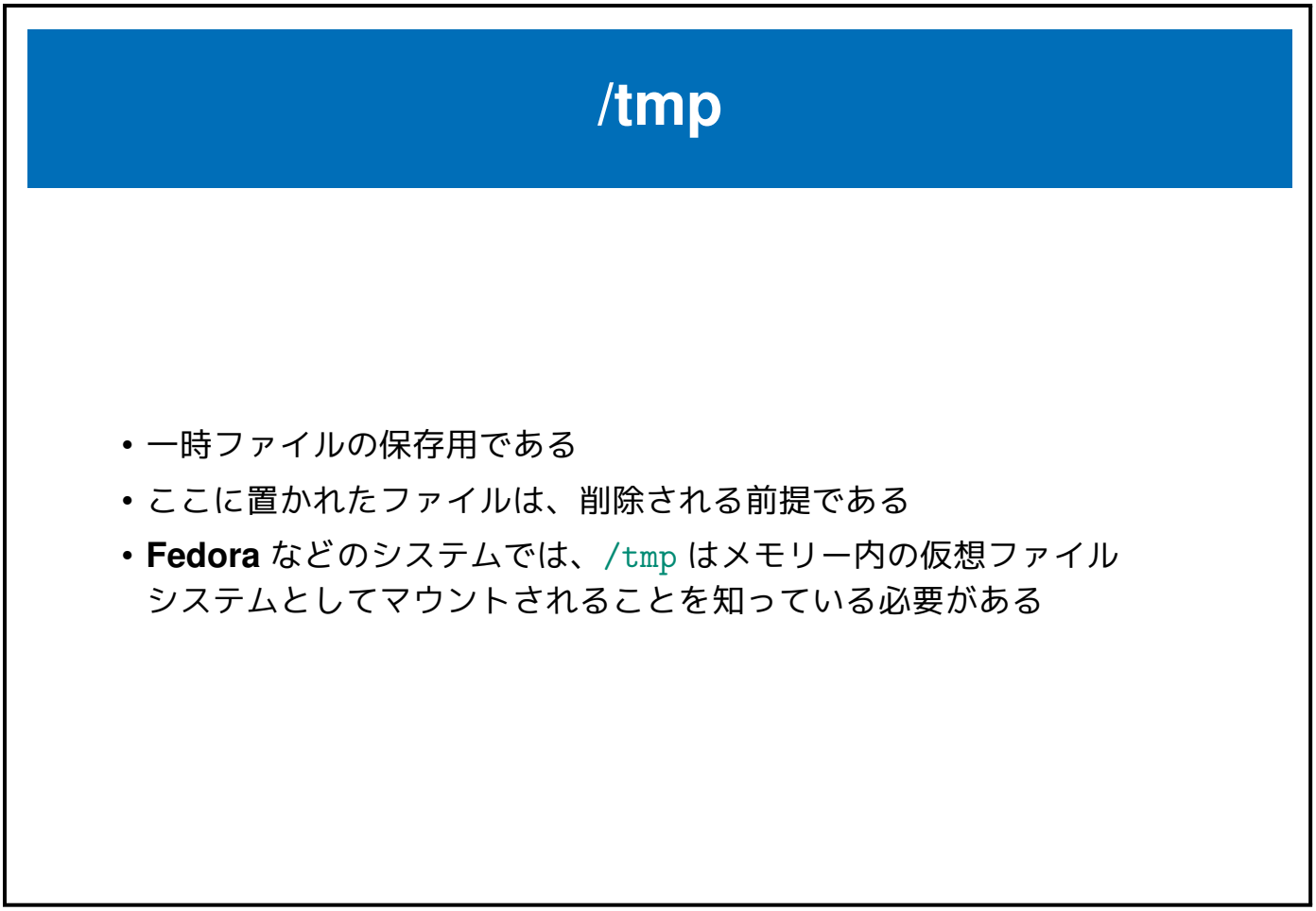
> /srv にはこのシステムによって提供されるサイト固有のデータが含まれています。
>
> これを指定する主な目的は、ユーザーが特定のサービスのデータファイルの場所を見つけることができるようにすることです。そして、読み取り専用データ、書き込み可能なデータ、およびスクリプト（cgi スクリプトなど）に対して単一のツリーを必要とするサービスを合理的に配置できるようにすることです。
>
> /srv のサブディレクトリに名前を付ける方法は、現時点ではこれがどのように行われるべきかの合意が得られていないため特定されていません。/srv の下でデータを構造化する 1 つの方法としてプロトコル（ftp、rsync、www、cvs など）があります。

システム管理者（およびディストリビューション）は `/srv` を使用するものと無視するものとに分かれます。
`/srv` と同じように `/var` に何を置くのが最善かについても、しばしば混乱が生じています。

Ubuntu や **Red Hat** 系などの **Linux** ディストリビューションでは、デフォルトで `/srv` は空になっています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.19 /tmp



このディレクトリは一時ファイルを保存するために使用され、任意のユーザーまたはアプリケーションがアクセスできます。ただし /tmp にあるファイルを長期間使用することはできません。

- 一部のディストリビューションでは、ファイル削除用スクリプトが変更されない限り 10 日以前のファイルを削除する定期実行ジョブが起動されます。
- 一部のディストリビューションでは、リブートするたびに /tmp の内容が削除されます。これは **Ubuntu** のポリシーです。
- 最近の一部のディストリビューションは仮想ファイルシステムを利用しており /tmp ディレクトリを **tmpfs** ファイルシステムを使用する **RAM ディスク**のマウントポイントとしてだけ使用します。これは **Fedora** システムのデフォルトポリシーです。システムが再起動すると全ての情報が失われます。/tmp は文字通り一時的なものです！

最後に注意ですが /tmp には大きなファイルを作成しないようにする必要があります。これらのファイルは実際にはディスクではなくメモリー内の領域を占有するので、メモリーの枯渇によってシステムに損害を加えたりクラッシュの原因になる可能性があります。ガイドラインではアプリケーションが /tmp に大きなファイルを置かないようにしていますが、このポリシーに違反し /tmp に大きな一時ファイルを作成するアプリケーションはたくさんあります。（おそらく環境変数を指定することによって）それらを別の場所に配置することが可能であっても、多くのユーザーはその方法を認識しておらず全てのユーザーが /tmp にアクセスしています。

Feodra の **systemd** を使用するシステムでは、次のコマンドを実行することによりこのポリシーをキャンセルできます。

```
$ sudo systemctl mask tmp.mount
```

設定はシステム再起動後に有効になります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.20 /usr

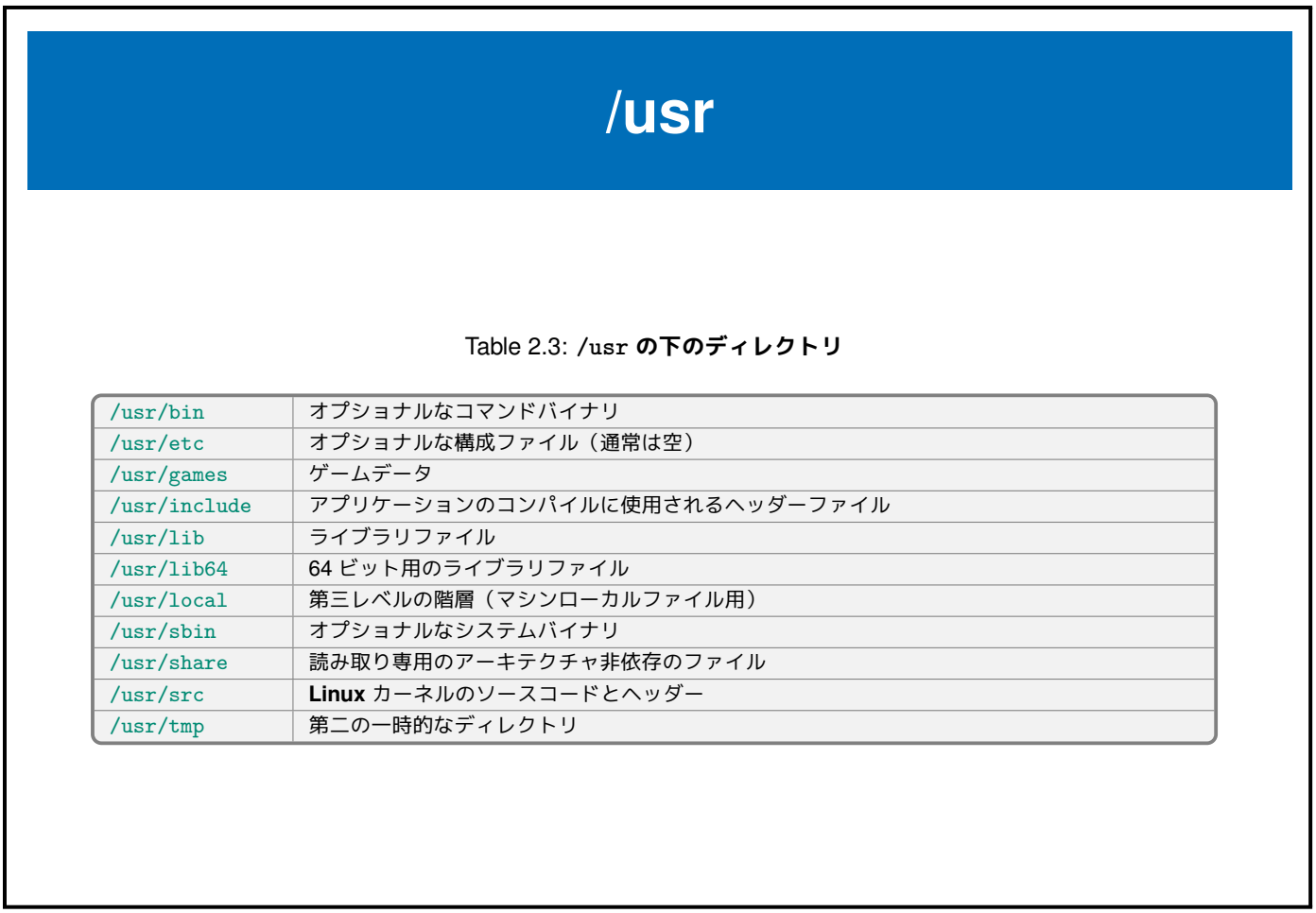


Table 2.3: /usr の下のディレクトリ

/usr/bin	オプショナルなコマンドバイナリ
/usr/etc	オプショナルな構成ファイル（通常は空）
/usr/games	ゲームデータ
/usr/include	アプリケーションのコンパイルに使用されるヘッダーファイル
/usr/lib	ライブラリファイル
/usr/lib64	64 ビット用のライブラリファイル
/usr/local	第三レベルの階層（マシンローカルファイル用）
/usr/sbin	オプショナルなシステムバイナリ
/usr/share	読み取り専用のアーキテクチャ非依存のファイル
/usr/src	**Linux** カーネルのソースコードとヘッダー
/usr/tmp	第二の一時的なディレクトリ

/usr ディレクトリは **二次的な階層** と考えることができます。システムの起動に必要ではないファイルのために使用します。実際 /usr はルートディレクトリと同じパーティションに存在する必要はありません。ネットワーク越しに同じシステムアーキテクチャを使用するホスト間で共有できます。

ソフトウェアパッケージは /usr の直下にサブディレクトリを作成しないようにしてください。互換性のために他の場所へのシンボリックリンクが存在する場合があります。

通常このディレクトリは読み取り専用データです。シングルユーザー モードには必要のないバイナリが含まれています。/usr/local ディレクトリが含まれローカルバイナリなどが保存されます。**man** ページは /usr/share/man に保存されます。

重要

最近の一部のディストリビューションは /usr 配下に /bin、/usr/bin、/sbin、/usr/sbin を直接配置せず、シンボリックリンクを使用してそれらを結合する戦略を放棄しています。彼らは /usr を別のパーティションに配置して起動後にマウントできるようにするという昔ながらの概念はもう使われていないと見ています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.21 /var

/var

- 可変データファイルを配置、読み込み専用ではマウントできない
  - ログファイル
  - スプールディレクトリとスプールファイル
  - 管理用データファイル
  - キャッシュコンテンツなど一時ファイル

Table 2.4: /var 以下のディレクトリ

ディレクトリ	目的
/var/ftp	ftp サーバーベースに使用
/var/lib	実行時にプログラムによって変更される恒久的なデータ
/var/lock	リソースへの同時アクセスを制御するために使用されるロックファイル
/var/log	ログファイル
/var/mail	ユーザーメールボックス
/var/run	ブート以降の実行中のシステムに関する情報
/var/spool	印刷キューなど **スプール** されたまたは処理を待機しているタスク
/var/tmp	システム再起動でも保持される一時ファイル。/tmp にリンクされることもある
/var/www	ウェブサイトの階層のルート

このディレクトリには、システム操作中に頻繁に変更 (=**var**iable) される変数（または **揮発性**）データファイルが含まれます。言うまでもなく **var** を読み取り専用ファイルシステムとしてマウントすることはできません。セキュリティ上の理由から、/var を別のファイルシステムとしてマウントすることは良い考えだと思います。さらにディレクトリがいっぱいになってもシステムはロックされるべきではありません。

/var/log はほとんどのログファイルが置かれている場所であり、/var/spool はメール、印刷、cron ジョブなどのプロセス用のローカルファイルがアクションを待っている間保存される場所です。

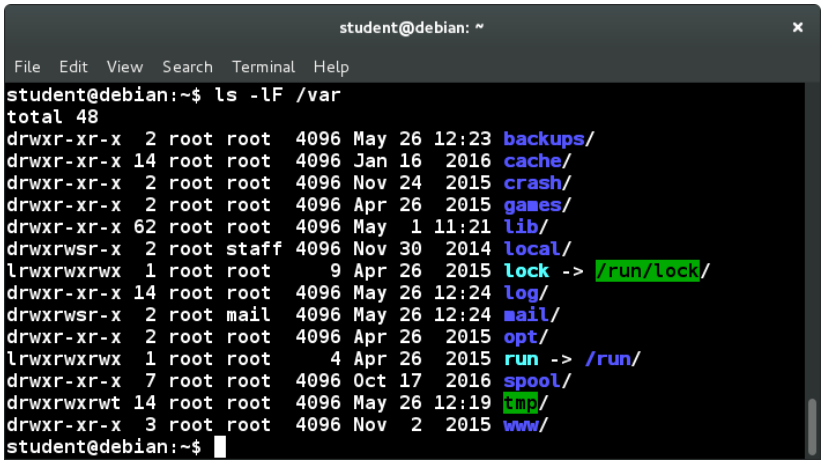


図 2.12: /var ディレクトリ

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

2.22 /run

/run

- 仮想ファイルシステム：実体はメモリー内にある
- 一時的なランタイムファイル用である
- しばしばリムーバブルメディアのマウントに使われる

`/run` は **一時的** なファイルが保存される場所です。これらはシステムの起動の早い段階で書き込まれ、再起動時に保持する必要がないランタイム情報が含まれます。

通常 `/run` は空のマウントポイントとして実装され、実行時に **tmpfs** RAM ディスク（`/dev/shm` など）がマウントされます。これはメモリーにだけ存在する疑似ファイルシステムです。

`/var/run` や `/var/lock` などは、`/run` の下のディレクトリへの単なるシンボリックリンクになります。ディストリビューションによっては他のディレクトリも `/run` の下の場所を指しているだけの場合があります。

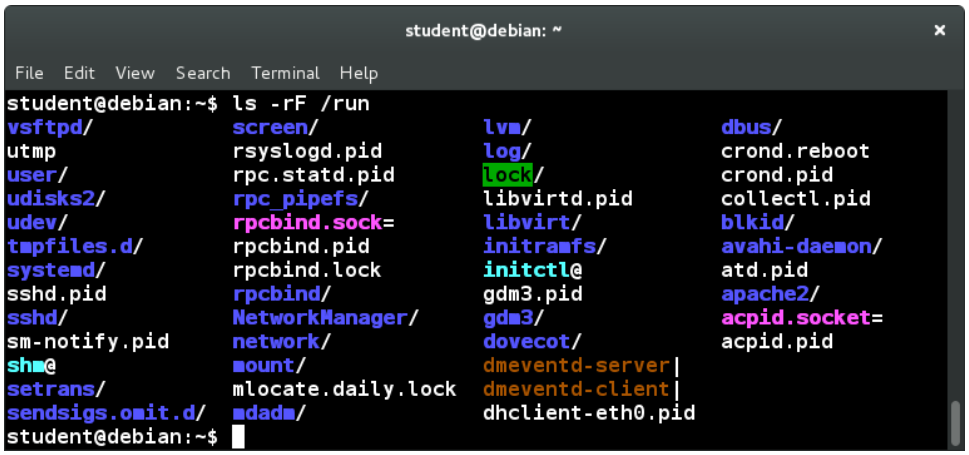


図 2.13: **/run ディレクトリ**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

2.23 演習

デモ教材ビデオ

using_linux_distros_demo.mp4
using_disk_usage_demo.mp4

課題 2.1: Linux の標準的なディレクトリのサイズ

du ユーティリティを利用して、システムの各最上位ディレクトリのサイズを計算します。

以下のコマンドを入力して、効率的に結果を取得し表示するヒントにしてください。

```
$ du --help
```

解 2.1 / 以下のディレクトリの一覧とサイズを表示します。

```
$ sudo du --max-depth=1 -hx /
```

```
4.3M    /home
16K     /lost+found
39M     /etc
4.0K    /srv
3.6M    /root
178M    /opt
138M    /boot
6.1G    /usr
1.1G    /var
16K     /mnt
4.0K    /media
869M    /tmp
8.4G    /
```

以下のオプションを使用できます。

- / から一階層下だけに注目し、そこある全てのものを再帰的に表示します。
- -h: 人間に分かりやすく表示します。(KB、MB、GB)
- -x 別のファイルシステムにあるディレクトリを参照しません。/ パーティションに無いディレクトリは無視されます。具体的には以下を無視します。

  ```
  /dev /proc /run /sys
  ```

  これらはメモリーだけに存在する疑似ファイルシステムです。システムが実行されていないときは、マウントポイントは空になります。上の例は **RHEL** システムで実行されたものなので、以下のマウントポイントも含まれていません。

  ```
  /bin /sbin /lib /lib64
  ```

  /usr 配下の対応するものにシンボリックリンクされています。

課題 2.2: /proc ファイルシステムを探索する

注目

本演習の内容はカーネルのバージョンに依存します。したがって、必ずしも出力は一致しません。

1. root ユーザーで /proc に **cd** しディレクトリを表示します。以下のように表示されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
$ cd /proc
$ ls -F
```

```
1/      128/    1510/   20/     2411/   30895/  53/    6925/  802/  951/         kmsg
10/     129/    1511/   2015/   2425/   31/     54/    7/     81/   952/         kpagecgroup
1002/   13/     1512/   2022/   2436/   31449/  55/    70/    813/  957/         kpagecount
1007/   130/    1513/   2023/   2444/   32/     56/    702/   814/  97/          kpageflags
10540/  131/    1514/   20300/  2451/   33/     58/    709/   816/  9742/        loadavg
10590/  13172/  152/    20354/  2457/   34/     585/   71/    817/  98/          locks
10798/  132/    15552/  20380/  2489/   35/     59/    718/   82/   99/          meminfo
10805/  133/    15663/  20388/  25/     36/     60/    719/   83/   9923/        misc
10806/  134/    15737/  20392/  2503/   37/     61/    72/    834/  acpi/        modules
10809/  135/    159/    20396/  2504/   374/    6193/  721/   835/  asound/      mounts@
10810/  136/    15981/  2086/   2531/   379/    62/    723/   84/   buddyinfo    mtrr
10813/  137/    16/     2090/   2546/   38/     63/    725/   841/  bus/         net@
10894/  138/    162/    211/    2549/   380/    634/   727/   842/  cgroups      pagetypeinfo
10925/  1384/   1632/   22/     2562/   40/     64/    73/    85/   cmdline      partitions
10932/  1385/   1636/   2205/   25794/  41/     65/    7300/  857/  config.gz    sched_debug
10934/  1387/   166/    2209/   26/     42/     662/   74/    86/   consoles     scsi/
10935/  139/    1670/   2212/   2610/   43/     663/   757/   864/  cpuinfo      self@
10941/  1390/   17/     2232/   26108/  44/     665/   758/   867/  crypto       slabinfo
10983/  1393/   17271/  2238/   2619/   4435/   666/   76/    87/   devices      softirqs
10998/  14/     17361/  2296/   2624/   45/     67/    761/   88/   diskstats    stat
11/     140/    1793/   2298/   2627/   46/     670/   762/   881/  dma          swaps
11047/  1410/   18/     23/     2644/   468/    671/   765/   886/  driver/      sys/
1105/   1415/   1831/   23042/  2645/   47/     673/   766/   887/  execdomains  sysrq-trigger
1121/   1429/   18880/  2344/   2679/   470/    674/   768/   888/  fb           sysvipc/
1123/   1437/   18903/  2348/   27/     484/    678/   769/   889/  filesystems  thread-self@
1135/   1445/   19/     2353/   2706/   49/     679/   77/    89/   fs/          timer_list
11420/  146/    19392/  2354/   2762/   492/    68/    771/   9/    interrupts   timer_stats
11499/  1463/   19488/  2365/   28/     493/    682/   78/    90/   iomem        tty/
11515/  147/    1954/   23683/  2858/   5/      683/   79/    92/   ioports      uptime
11530/  1476/   1963/   2370/   28730/  50/     686/   793/   921/  irq/         version
1163/   148/    19727/  2372/   28734/  51/     687/   794/   928/  kallsyms     vmallocinfo
1164/   1485/   19734/  2374/   29/     510/    69/    8/     930/  kcore        vmstat
12/     149/    19984/  24/     2973/   514/    690/   80/    931/  keys         zoneinfo
127/    15/     2/      2406/   3/      52/     691/   801/   944/  key-users
```

たくさんの数字のディレクトリが存在しています。これらはそれぞれプロセスに対応し、名称は **プロセス ID** です。重要なサブディレクトリとして、/proc/sys があります。その下に多数のシステムパラメータがあり、それらを参照したり、更新したりできます。このサブディレクトリについては以降で議論します。

2. 以下のファイルを参照します。

   - /proc/cpuinfo:
   - /proc/meminfo:
   - /proc/mounts:
   - /proc/swaps:
   - /proc/version:
   - /proc/partitions:
   - /proc/interrupts:

   名称から内容の想定がつきます。

   これらの値は常時更新されているわけではなく、参照したときに更新されることに注意してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

3. プロセスのディレクトリのどれかを参照してください。
（それ以外のディレクトリは **sudo** してない場合には見えない情報があります）

```
$ ls -F 4435
```

```
attr/       coredump_filter  gid_map     mountinfo   oom_score_adj  sessionid  syscall
autogroup   cpuset           io          mounts      pagemap        setgroups  task/
auxv        cwd@             limits      mountstats  personality    smaps      timerslack_ns
cgroup      environ          loginuid    net/        projid_map     stack      uid_map
clear_refs  exe@             map_files/  ns/         root@          stat       wchan
cmdline     fd/              maps        oom_adj     sched          statm
comm        fdinfo/          mem         oom_score   schedstat      status
```

以下の項目について確認してください。

- cmdline
- cwd
- environ
- mem
- status

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 3 章

ユーザー環境

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- **環境変数**の理解、使用、設定、変更、およびエクスポート（有効化）。
- 過去のコマンド履歴を調べ、以前のコマンドを取り出し、編集し、使用する。
- よく使うコマンドやオプションに **エイリアス（別名）**を設定する。

3.1 環境変数

環境変数

- 名前と値を関連付ける

  HOME
  HOST
  PATH

- 以下のように、いくつかの記述フォーマットが利用できる

  $ env
  $ export
  $ set

- すべての環境変数は、参照時は先頭に $ 記号をつける

  $ echo PATH=$PATH

- 既に定義されている場合は例外となる

  $ MYCOLOR=blue

  等号記号の周りにはスペースをつけてはいけない点に注意

env、**set**、**printenv** を使って環境変数を列挙すると多数表示されます。多くのアプリケーションやプログラムが環境変数を参照しているので、環境変数が定義されていない場合は失敗します。**Linux** では環境変数はシステム起動時に設定され、自分の目的で作成して操作できます。

```
$ env | head -2
```

```
XDG_VTNR=1
XDG_SESSION_ID=c259
```

```
$ set | head -2
```

```
ABRT_DEBUG_LOG=/dev/null
BASH=/usr/bin/bash
```

```
$ export | head -2
```

```
declare -x BITS="64"
declare -x CCACHE_COMPRESS="1"
```

```
$ echo $PATH
```

```
/usr/local/texlive/2020/bin/x86_64-linux/:/sbin:/usr/lib64/ccache:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin::/home/bjmoose/bin
```

```
$ echo $HOME
```

```
/home/bjmoose
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

いくつかの重要な環境変数

- HOME
  - ユーザーのホームディレクトリ
  - **cd** と打つとホームディレクトリに移動する
- PATH
  - 実行するプログラムを検索するための順序付きリスト
  - ディレクトリはコロンで区切る
- PS1
  - コマンドライン プロンプトで、簡単にカスタマイズできる
- SHELL
  - ユーザーのデフォルト シェル (**bash**、**csh** など)
- EDITOR
  - ユーザーのデフォルト エディター (**emacs**、**vi** など)

```
$ echo $HOME
  /home/rjsquirrel
$ echo $PATH
  /usr/lib64/ccache:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/rjsquirrel/bin
$ echo $PS1
  \h:$PWD>
$ echo $SHELL
  /bin/bash
$ echo $EDITOR
  /usr/bin/emacs
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

環境変数を設定する

- 特定の変数をリストする
  `$ echo $SHELL`
- 新しい変数を設定する
  `$ VARIABLE=value`
- 新しい変数を恒久的に追加する
  - VARIABLE=value を含むように ~/.bashrc を編集する
  - 新しいシェルを起動する、または ログアウト／ログイン する

```
$ echo $SHELL
```

```
/bin/bash
```

```
$ echo $EDITOR
```

```
/usr/bin/emacs
```

```
$ export EDITOR=/usr/bin/nano
$ echo $EDITOR
```

```
/usr/bin/nano
```

```
$ echo $MYVAR

$ vi ~/.bashrc          <-- "export MYVAR=myvalue" を追加する
$ echo $MYVAR

$ source ~/.bashrc
$ echo $MYVAR
```

```
myvalue
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

環境変数をエクスポートする

- デフォルトでは、環境変数は子プロセスに継承されない
- 変数を **エクスポート**すると、その挙動が変わる

```
$ VAR=value ; export VAR
$ export VAR=value
```

- エクスポートされた変数をリストアップできる

```
$ export
```

デフォルトでは、スクリプト内で作成された変数は、現在のシェルでだけ使用可能です：子プロセス（サブシェル）は、この変数の内容にアクセスすることはできません。子プロセス（サブシェル）はこの変数の内容にアクセスできません。変数を子プロセスから見えるようにするには、**export** コマンドを使用して変数を **エクスポート**する必要があります。変数のエクスポートは、1 つのステップで行うことができます。

```
$ export VAR=value
```

または、2 ステップで

```
$ VAR=value ; export VAR
```

子プロセスはエクスポートされた変数を変更することができますが、エクスポートされた変数は共有されずコピーされるだけなので、この場合の変更は親シェルに伝わらないことに留意してください。

```
$ export VERSION=$(uname-r)
```

```
$ export
```

```
declare -x BITS="64"
declare -x CCACHE_COMPRESS="1"
declare -x CCACHE_DIR="/mp/.ccache"
...
declare -x VERSION="4.0.3"
...
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

3.2 コマンド ヒストリー

コマンド ヒストリー

- 以前の操作の履歴を表示するコマンドである
  `$ history`
- シェルはヒストリー（履歴）を `~/.bash_history` に保存している
- ヒストリーファイルの場所は HISTFILE
- ヒストリーファイルの最大行数は HISTFILESIZE
- 現在のセッションで記録されるヒストリーの最大行数は HISTSIZE

ヒストリーファイルの場所を表示します。

```
$ echo $HISTFILE
   /home/jbond/.bash_history
```

ヒストリーファイルの最大サイズを行単位で表示します。

```
$ echo $HISTFILESIZE
   2000
```

ヒストリーファイルの現在のサイズ（行数）を表示する。

```
$ echo $HISTSIZE
   1000
```

ヒストリーファイルの最初の 2 行を表示します。

```
$ history | head -2
   40  cd RELEASE
   41  ls -ltr
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

以前のコマンドを呼び出して、再利用する

- 過去の／最近のコマンドを呼び出す
  - 上／下 矢印キー
- 直前のコマンドを実行する
  - !!
- 過去に使用したコマンドを検索する
  - CTRL-R
  - 複数回押すと、以前のコマンドを順に遡る

矢印キーで **ヒストリー（履歴）**リストを上下に移動し、直前に実行したコマンドが最初にきます。!!（! はバンとも呼ばれることから " バンバン" と発音されることが多い）は、直前のコマンドを実行します。以下は CTRL-R を使用してヒストリーを検索する例です。

```
# 以下の操作を続けて実行する
$ CTRL-R  git
(reverse-i-search)`git': git rm -f exported-variables.inc
# エンターキーを押して表示されたコマンドを実行するか、更に CTRL-R を実行してコマンド履歴を検索する
$ CTRL-R
(reverse-i-search)`git': git ls-files
# エンターキーを押して実行する
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

ヒストリーを使った過去のコマンドの実行

- ヒストリーの置き換えを開始する
  `!`
- 行の最後の引数を参照する
  `!$`
- n 番目のコマンドラインを参照する
  `!n`
- string（文字列）で始まる最も新しいコマンドを参照する
  `!string`

ヒストリーの置換はすべて ! から始まります。したがって、コマンドの中で

```
$ ls -l /bin /etc /var
```

!$ は、上の行の最後の引数である /var を指します。

```
$ history
```

```
1  echo $SHELL
2  echo $HOME
3  echo $PS1
4  ls -a
5  ls -l /etc/passwd
6  sleep 1000
7  history
```

```
$ !1              # 上で表示された１番目のコマンドを実行する
```

```
echo $SHELL
/bin/bash
```

```
$ !sl             # "sl" で始まるコマンドを実行する
```

```
sleep 1000
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

3.3 コマンド エイリアス（別名）

エイリアス（別名）

- **エイリアス**を作成して、カスタム コマンドを作る

  ```
  $ alias name=command
  ```

  等号記号の周りにスペースが無いことに注意
- ~/.bashrc に上記の alias コマンドを追加すると永続化する
- 以下の例のように、標準プログラムの挙動の変更に最適である

  ```
  $ alias rm='rm -i'
  ```
- エイリアスの無効化

  ```
  $ unalias name
  ```
- 現在有効なエイリアスをリストアップ

  ```
  $ alias
  ```

エイリアスにはカスタム定義ができます。引数なしで **alias** と入力すると、定義されたエイリアスのリストが表示されます。**unalias** は、エイリアス設定を無効化します。

いくつかの例を見てみましょう。

```
$ alias l='ls -laF'
$ alias dir='ls -latF'
$ alias rm='rm -i'
$ alias mv='mv -i'
$ alias cp='cp -ipdv'
$ alias df='df -T'
$ alias gitstat='git status -uno'
$ alias diffside='diff --side-by-side --ignore-all-space'
$ alias scapt='gnome-screenshot -i -w'
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

3.4 演習

課題 3.1: あなたのファイルパスにディレクトリ ~/work を追加する

以下の行を含んだ小さなファイル ~/work/ls を作成します。

```
echo HELLO, this is the phony ls program.
```

そして、以下のようにファイルに実行権を付与します。

```
$ chmod +x ~/work/ls
```

1. ~/work をあなたのパスの最後に追加して、これが通常のパス検索の **後に** 検索されるようにします。**ls** と入力し、/bin/ls または ~/work/ls のどちらのプログラムが実行されるか見てみましょう。
2. ~/work をあなたのパスの先頭に追加して、これが通常のパス検索の **前の** 検索されるようにします。ふたたび、**ls** と入力し、/bin/ls または ~/work/ls のどちらのプログラムが実行されるか見てみましょう。

このようなパスの変更によって、どんなセキュリティ上の懸念が生じるでしょうか？

解 3.1

最初に、エディターを使ってニセの **ls** を作ってみましょう。または、直接打ち込んで実行してみましょう。

```
$ echo "echo HELLO, this is the phony ls program." > ~/work/ls
$ chmod +x ~/work/ls
```

次の 2 つのステップでは、別のターミナルウィンドウで作業するか、新しいシェルを起動するのがよいでしょう、そうすれば、変更は後で発行したコマンドまで永続しません。**bash** と入力すれば、新しいシェルを起動することができます。

1. ```
   $ bash
   $ PATH=$PATH:~/work
   ```

   ```
   bin  etc  games  include  lib  lib64  libexec  local  sbin  share  src  tmp
   ```

   ```
   $ exit
   ```

2. ```
   $ bash
   $ PATH=~/work:$PATH
   $ ls /usr
   ```

   ```
   HELLO, this is the phony ls program.
   ```

   ```
   $ exit
   ```

重要

2 番目の形式は極めて危険で、**トロイの木馬** プログラムを挿入するための巧妙な方法になることに注意してください。
誰かが悪意のあるプログラムを ~/work に置くことができれば、あなたを騙して誤って実行させることができます。
あなたのパス内のディレクトリに対して、他のユーザーが **書き込み権** を持っていないことを確認してください。

課題 3.2: コマンド ヒストリー

あなたは **Linux** ワークステーションで忙しく働いていて、ある特定の **bash** コマンドシェルで約 100 個コマンドを入力しています。

少し前に新しいコマンドを使ったが、正確な名称を忘れてしまいました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

あるいは、オプションや引数がたくさんあるかなり複雑なコマンドで、もう一度打ち直すときにミスをしたくありません。

コマンドがどんなものだったか、どうやって確認できますか？

ヒストリーでコマンドを見つけたら、プロンプトから全部再入力することなく、簡単にもう一度コマンドを実行するにはどうすれば良いのでしょうか？

解 3.2

コマンド ヒストリー

history コマンドは、あなたがこれまでタイプしたコマンドを表示する方法です。

```
$ history
```

```
    1  cd /
    2  ls
    3  cd
    4  pwd
    5  echo $SHELL
    6  ls /var/
    7  ls /usr/bin
    8  ls /usr/local/bin
    9  man fstab
   10  ls
          . . .
```

以前のコマンドを再実行するためには、いくつかの方法が選べます。例えば、最初にログインしたときに実行した **man** コマンドを再実行したいとします。その場合、次のように入力します。

```
$ !9
```

と入力すると、#9 として表示されているコマンドを再実行することができます。もし、入力した **man** コマンドがこれだけだった場合は、次のように入力することもできます。

```
$ !man
```

のように、入力したコマンド名を覚えておくとよいでしょう。最後に、いくつかの **man** コマンドを入力した場合、CTRL-R を使って履歴を遡って検索し、再実行したい特定の **man** コマンドを見つけ、Return を押すだけで実行することができます

課題 3.3: コマンド アイリアス

長いコマンドやファイル名を何度も入力するのは面倒ですし、タイプミスなどの些細なミスも多くなります。**エイリアス** を導入することで、このような入力の手間を軽減するためのショートカットを定義することができます。

あなたがプロジェクトチームのメンバーで、プロジェクトのための共通の共有ディレクトリで作業しているとします。このディレクトリは、/home/staff/RandD/projects/projectX/src に配置されています。

プロジェクト X の作業をしていると、このディレクトリにファイルを作成したり、修正したりする必要があることがよくあります。入力するまでにあまり時間をかけられません。

```
$ cd /home/staff/RandD/projects/projectX/src
```

これは煩雑ですね。

上記の **cd** コマンドを実行するために、"projx" というエイリアスを定義して使用します。

解 3.3

コマンド エイリアス:

エイリアス行は次のようになります。

```
$ alias projx='cd /home/staff/RandD/projects/projectX/src'
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

なお、一重引用符の代わりに二重引用符を使用することもできますし、定義したエイリアスに空白がない場合には引用符を全く使用しないことも可能です。

あとは、設定したエイリアスを実行することでディレクトリに移動できます。

```
$ projx
```

$HOME/.bashrc ファイルに追記（設定）すれば、エイリアスを永続化できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 4 章

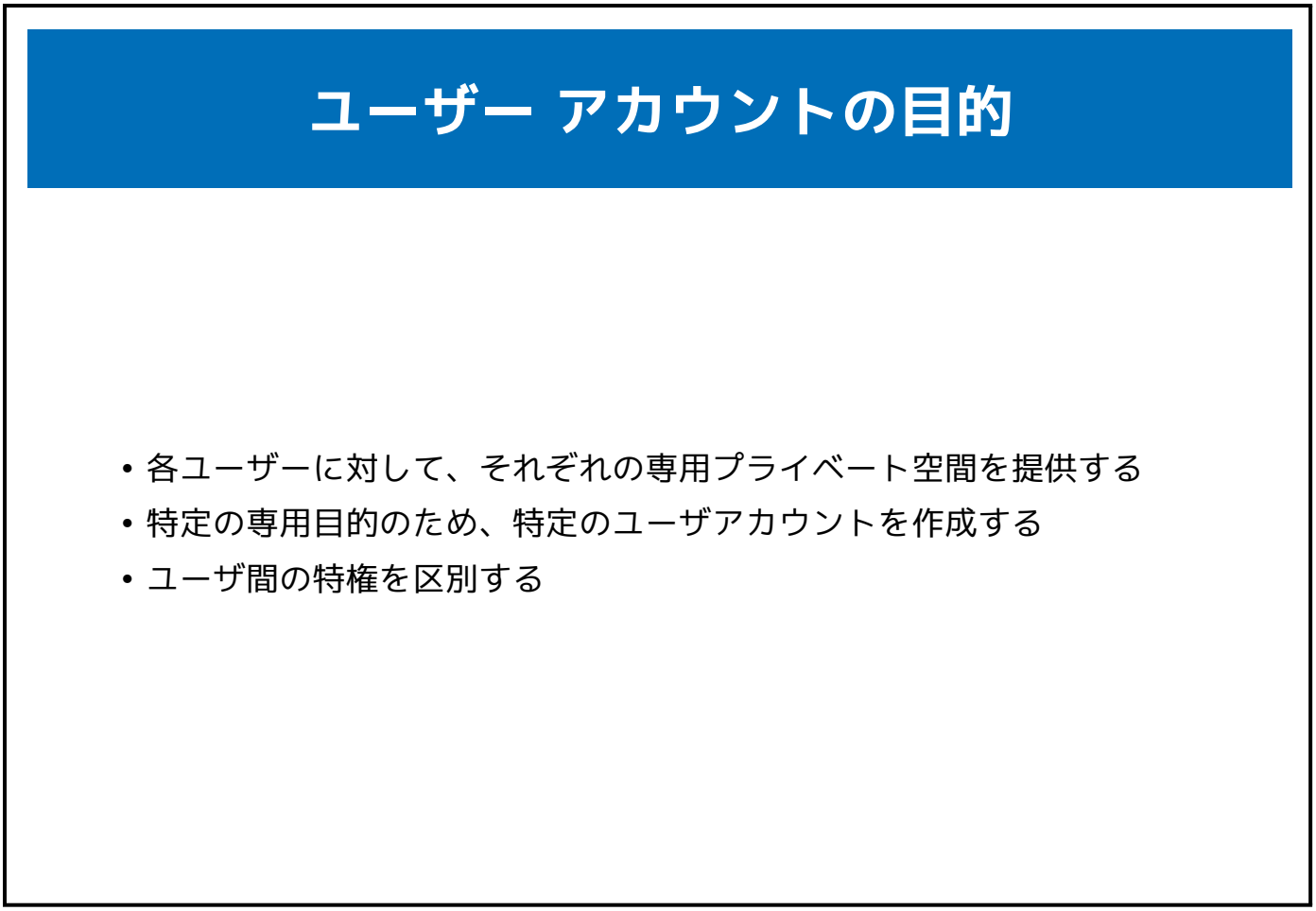
ユーザー アカウントの管理

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- 個々のユーザーアカウントの目的を説明し、その主な属性を列挙する。
- 新規ユーザーアカウントの作成、既存アカウントのプロパティの変更、アカウントの削除やロックができる。
- ユーザーパスワードの設定、暗号化、保存方法を理解し、セキュリティのためにパスワード変更を継続要求する方法を理解する。
- root アカウントの役割と使用するタイミングを理解する。
- Secure Shell（**SSH**）を使用し、ログインとコマンドを削除する。

4.1 ユーザー アカウント



Linux システムは、複数のユーザーやプロセスに対して並列的に独立した作業領域を許容する **マルチ ユーザー環境** を提供します。

特別なユーザーアカウントとして、システムで **どんなことでも** 実行できる **root** ユーザーがあります。二度と取り返しがつかない大失敗を回避するため、またセキュリティ上の理由から、root アカウントは真に必要がある場合に限って使用するべきです。

一般ユーザー アカウントは、システム上で作業する人向けです。プロセスが、root アカウントを使わずにプログラムを実行できるようにするための専用アカウント（**daemon** など）がいくつか存在します。

特定のユーザーの集団が、共通の目的のためにファイルや権限などを共有する **グループ管理** についても紹介します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ユーザー アカウントの属性

/etc/passwd より:

```
....
beav:x:1000:1000:Theodore Cleaver:/home/beav:/bin/bash
warden:x:1001:1001:Ward Cleaver:/home/warden:/bin/bash
dobie:x:1002:1002:Dobie Gillis:/home/dobie:/bin/bash
....
```

- ユーザー名
- ユーザーパスワード
- ユーザー識別番号 (**UID**)
- グループ識別番号 (**GID**)
- コメント、または **GECOS** 情報
- ホームディレクトリ
- ログインシェル

/etc/passwd ファイルの各行には、システムの各ユーザーの基本アカウント属性が記載されています。(パスワードとこのファイルについては後で説明します) 各行に含まれる 7 つの要素は

1. ユーザー名
   各ユーザーに割り当てられた一意の名前です。
2. ユーザーパスワード
   各ユーザーに割り当てられたパスワードです。
3. ユーザー識別番号 (**UID**)
   ユーザー アカウントに割り当てられた一意の番号です。システムはユーザーの特権の決定やアクティビティ追跡など、さまざまな目的で **UID** を使用します。
4. グループ識別番号 (**GID**)
   ユーザーの ***グループ*** がプライマリ グループ、プリンシパル グループ、またはデフォルト グループのいずれかを示します。
5. コメント、または **GECOS** 情報
   連絡先情報 (フルネーム、電子メールアドレス、オフィス、連絡先番号) の記録ために使用されるコメントフィールドです。(GECOS は非常に古い用語なので、その意味を意識する必要はありません)
6. ホームディレクトリ
   ほとんどのユーザーでは、そのユーザーのワーキングエリアとなる専用のディレクトリです。通常はユーザーがそのディレクトリの所有者になります。**root** 以外のユーザーについては、システムの /home の下に作られます。
7. ログインシェル
   通常、ここには **/bin/bash** や **/bin/csh** などのシェルプログラムを指定します。ただし、特別なケースではここに別のプログラムを指定することがあります。一般論で言えば、本フィールドには、任意の実行可能ファイルを指定することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

カレント ユーザーの確認

- 現在ログインしているユーザーを見つけるには
  - **who**
- 現在のユーザー ID を調べるには
  - **whoami**
- 現在のユーザーのユーザー ID、グループ ID、及びそれらに割り当てられた数字を見るには
  - **id**

例えば、以下のように

```
$ who
```

```
user     pts/0        2023-03-09 12:58 (:0)
user     pts/1        2023-03-09 23:23 (:0)
user     pts/2        2023-03-09 23:55 (:0)
user     pts/3        2023-03-09 23:37 (:0)
user     pts/4        2023-03-09 23:38 (:0)
user     pts/5        2023-03-31 09:47 (:0)
user     pts/6        2023-03-31 11:13 (:0)
badguy   pts/8        2023-04-01 16:54 (meow.local)
```

```
$ whoami
```

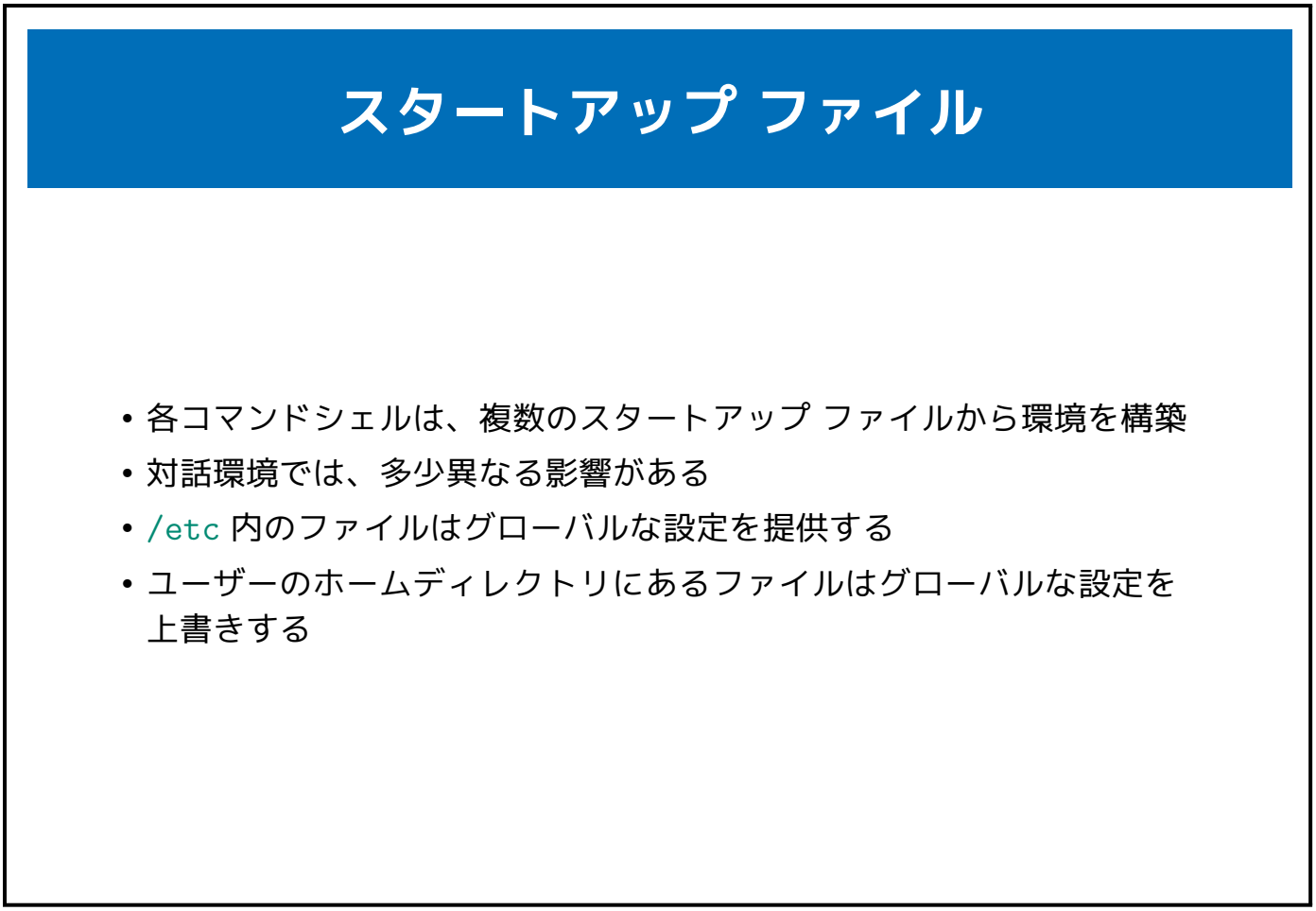
```
user
```

```
$ id
```

```
uid=1000(user) gid=1000(user) groups=1000(user),999(qubes)
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.2 シェル スタートアップ ファイル



コマンド ウィンドウまたは実行するスクリプトであっても、新しいシェルが起動するたびに適切な機能を実行するために必要な要素を含んだファイルがあります。これには以下のようなものがあります。

- 多くのプログラムやスクリプトで使用される **環境変数** （$PATH を含む）の定義
- コマンドやオプションの省略記法として使用される **エイリアス** の定義
- 後続のスクリプトで使用される **関数**の定義

通常 /etc には、システム全体に関わるグローバルな初期化ファイルがあり、ユーザー固有の個別設定ファイルの適用前に、通常すべてのユーザーから参照されます。

どのファイルをどのような順序で適用するかは、これから説明するように文脈によって異なります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

スタートアップ ファイルを利用するメリット

- ユーザーのプロンプトをカスタマイズする
- ユーザーの端末の種類を設定する
- コマンドの短縮形と別名（エイリアス）を設定する
- デフォルトのテキストエディターを設定する

スタートアップ（初期化）ファイルの処理がないと、コマンドやプログラムを実行するたびに、正しく機能するためのセットアップ作業が発生することがあります。

多くのプログラムは、実行開始時に特定の環境変数が設定されていることを評価し、それを利用して機能を制御しています。

例えば、テキストファイルを対話的に修正する必要があるプログラムは、`EDITOR` の設定から（おそらく vim、emacs、または nano でしょう）ユーザーの好みのエディターを確認します。

環境変数の項でその例を紹介します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

スタートアップ ファイルを評価する順番

- 全てのユーザー向けの初期値
  `/etc/profile`
- ログイン シェルの設定
  `~/.bash_profile`
- ログイン時の初期化
  `~/.bash_login`
- `/etc/profile` の上書き
  `~/.profile`
- 対話的な非ログインシェルの設定
  `~/.bashrc`
- 通常は `~/.bashrc` だけをカスタマイズすればよい

Linux にログインした時には, 常に `/etc/profile` が読み込まれて評価されます。

次に、以下のファイルがこの順序で検索されます。

`~/.bash_profile`

`~/.bash_login`

`~/.profile`

ファイルを見つけると、**Linux** ログイン シェルはそのスタートアップ ファイルを評価し、それ以外のファイルは無視します。

冗長に感じるかもしれませんが、色々な **Linux** ディストリビューションは異なるスタートアップ ファイル使う傾向があります。

ログインしたものとは別にシェルを作った時には、毎回 `~/.bashrc` だけが読み込まれ評価されます。ログインシェルでは読み込まれず、評価もされませんが、ほとんどのディストリビューションとユーザは、ユーザが所有する３つのスタートアップファイルのうちの１つから `~/.bashrc` を呼び出すので、実際には `~/.bashrc` に書いた設定はログインシェルからも参照されます。

なので、あなたのカスタマイズはほぼ全て `~/.bashrc` に追記すれば良いのです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.3 ユーザー アカウントの管理

useradd によるユーザー アカウント作成

- **useradd** は、デフォルト（ひな形）に従ってユーザーを作成する

```
$ sudo useradd bjmoose
```

  - bjmoose に対するアカウントを作成する
  - デフォルトの設定でユーザー／グループ ID、ホームディレクトリ、シェルを設定する
- **useradd** にオプションを付加して、デフォルトを上書きする

```
$ sudo useradd -s /bin/csh -m -k /etc/skel -c "Bullwinkle J Moose" bmoose
```

- オプションを使って、シェル、skel ディレクトリ、コメントを設定する

```
$ sudo useradd bjmoose
```

このコマンドは、以下のステップを実行します。

- bjmoose の UID のデフォルト値は (/etc/login.defs に指定されている) UID_MIN より一つ大きな値となります。
- bjmoose のグループ ID は GID=UID に設定され、同時に bjmoose のプライマリーグループに設定されます。
- ホームディレクトリ /home/bjmoose が作成され bjmoose を所有者に指定します。
- bjmoose のログインシェルを **/bin/bash** に指定します。
- /etc/skel のコンテンツを /home/bjmoose にコピーします。デフォルトでは /etc/skel には **bash** と **X Window** システムの初期化ファイルが含まれています。
- /etc/shadow ファイルの bjmoose 行のパスワード欄には「!!」が書き込まれます。これにより、このアカウントを利用可能な状態にするには、管理者がパスワードを付与することが求められます。

デフォルト設定は、以下のように **useradd** のオプションを使って簡単に上書きが可能となっています。

```
$ sudo useradd -s /bin/csh -m -k /etc/skel -c "Bullwinkle J Moose" bmoose
```

これは、ユーザー属性の一部にデフォルト以外の明示的な値を与える場合に使います。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ユーザー アカウントの変更と削除

- root ユーザーは **userdel** を使ってユーザー アカウントを削除できる

  ```
  $ sudo userdel rjsqirrel
  ```

  rjsqirrel のユーザー アカウントが削除されるが、ホーム ディレクトリは削除されない
- **usermod** を使ってユーザー アカウントを変更できる

  ```
  $ sudo usermod -L bjmoose
  ```

  bjmoose のユーザー アカウントをロックする、これでログインできなくなる

```
$ sudo userdel rjsqirrel
```

ユーザー rjsqirrel の全ての情報が /etc/passwd、/etc/shadow、/etc/group から消去されます。これでアカウントは削除されますが、後でアカウントが再設定される可能性に備えホーム ディレクトリ（通常は /home/rjsqirrel）は削除されません。**userdel** 実行時に -r オプションを指定すると、ホーム ディレクトリも消去されます。ただし削除されたユーザーが所有するシステム上の他の全てのファイルは残ります。

usermod は、グループ メンバーシップ、ホーム ディレクトリ、ログイン名、パスワード、デフォルトシェル、ユーザー ID など、ユーザー アカウント属性を変更するために使用できます。
使い方はとても簡単です。**usermod** は必要に応じて /etc ディレクトリ内のファイルを変更することに注意してください。

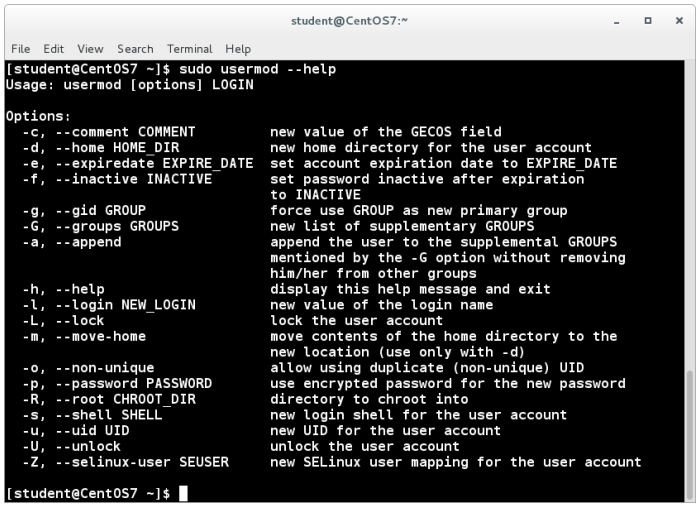


図 4.1: **usermod の利用**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.4 ロックされたアカウント

ロックされたアカウント

- **Linux** は、システム アカウントを **ロック** した状態で提供される
  - bin、daemon、sys などのアカウント
  - プログラムを実行する可能性はあるが、ログインには使われない
- 別の目的で、ロックされたアカウントが作られることもある
  - データベースなど主要なアプリケーションが利用するアカウント
- ロックされたアカウントには、有効なパスワードが存在しない
  - 通常 /etc/shadow の中で "!!" と表現される
- ロックされたアカウントには、全て無効なシェル（通常は /sbin/nologin）が設定されることを確認する

Linux にはロックされたアカウントがあります。このアカウントはプログラムの実行は可能ですが、システムへログインはできず、有効なパスワードも付与されません。たとえば、/etc/passwd には次のようなエントリがあります。

```
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

ロックされたユーザーがシステムにログインを試みると、**nologin** シェルは次を返します。

```
This account is currently not available.
```

または、/etc/nologin.txt に何かメッセージが設定されているときは、それを表示します。システムサービスまたはアプリケーションは、特別な目的でこのようなロックされたアカウントを作成します。/etc/passwd の中でログインシェルに **nologin** が指定されたユーザーを探すと、どのユーザーがいるか確認できます。次のように、特定のユーザーアカウントをロックすることもできます。

```
$ sudo usermod -L bjmoose
```

これらのアカウントはシステムに残りますが、ログインはできなくなります。-U オプションを使うと、ロックを解除できます。慣例として、ユーザーが組織を離れるとき、または長期の休職中は、ユーザーのアカウントをロックします。アカウント ロックの別の方法として **chage** を使用してアカウントの有効期限を変更する方法もあります。

```
$ sudo chage -E 2014-09-11 rjsqirrel
```

アカウント有効期限を過去の日付に設定すると、アカウントが無効となります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.5 パスワード

ユーザー ID と /etc/passwd

```
beav:x:1000:1000:Theodore Cleaver:/home/beav:/bin/bash
rsquirrel:x:1001:1001:Rocket J Squirrel:/home/rsquirrel:/bin/bash
```

- /etc/passwd ファイルには、１ユーザーあたり１レコード（１行）の情報が含まれる。各レコードのフィールドはコロン (:) で仕切られる
  - username: ユーザーのユニーク名
  - password: ハッシュ化データ (/etc/shadow 未使用時) またはプレースホルダー (/etc/shadow 使用時は "x" と表示される）
  - UID: ユーザー識別番号
  - GID: ユーザーのプライマリーグループ識別子
  - comment: コメント領域、通常はユーザーの実名
  - home: ユーザーのホームディレクトリのディレクトリパス名
  - shell: ログイン時に呼び出すシェルの絶対パス名
- UID は UID_MIN = 1000 から開始される

ほとんどの **Linux** ディストリビューションでは、**UID** が 1000 未満の値のを持つアカウントはシステムに属する特別なユーザーとみなされます。通常ユーザーのアカウントは 1000 から始まります。実際には、（1000 という実値ではなく）/etc/login.defs 内に UID_MIN として定義された値が使われます。

歴史的にみると **Red Hat** から派生したディストリビューションは、1000 ではなく UID_MIN=500 を使用していましたが **RHEL 7** 以降からは、一般的な値である 1000 になりました。

useradd の使用時に **UID** が指定されていない場合、システムは UID_MIN からインクリメンタルに ID を割り当てます。

さらに、各ユーザーはデフォルトで **UID** と同じ番号である **プライマリ グループ ID** を取得します。これらは **ユーザー プライベート グループ**（UPG）と呼ばれることもあります。

/etc/passwd、/etc/group、/etc/shadow を直接編集することは避けてください。代わりに **usermod** 、または上級ユーザーはファイルのロックやデータ破壊に配慮された **vipw** などの適切なユーティリティを使用してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.6 /etc/shadow

/etc/shadow を使用する理由

- ユーザー毎に、パスワードの期限管理が可能になる
- ハッシュ化されたパスワードのセキュリティを強化する

`/etc/passwd` のデフォルトのパーミッションは 644 (`-rw-r--r--`) なので、誰でもこのファイルを読むことができてしまいます。システム プログラムとユーザー アプリケーションがこのファイルの情報を読み取る必要があるため、残念ながらこのパーミッションにする必要があるのです。システム プログラムが root 権限で実行されないこともその理由ですが、しかしどんな状況でも、このファイルの変更を許されているのは root だけです。

懸念されるのは、ハッシュされたパスワードそのものです。それが `/etc/passwd` にあれば、誰でもハッシュ化されたパスワードのコピーを作成して **Crack** や **John the Ripper** などのユーティリティを使用して、ハッシュ化されたパスワードから元のテキストパスワードを推測できてしまうからです。これはセキュリティリスクになります！

`/etc/shadow` のパーミッションの設定は 400 (`-r--------`) です。つまり、このファイルにアクセスできるのは root だけです。これにより、誰かがハッシュされたパスワードを収集するのが難しくなっています。

説得力のある正当な理由がない限り `/etc/shadow` ファイルを使用すべきです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/etc/shadow のフォーマット

- /etc/shadow ファイルには１ユーザーにつき１つ（１行）のレコード
- 各レコードのフィールドはコロン (:) で仕切られている
  - username: ユニークなユーザー名
  - password: パスワードのハッシュ値 (sha512)
  - lastchange: 最後変更日（エポック日起点の日数）
  - mindays: パスワードを変更できるまでの日数
  - maxdays: パスワードを変更しなくてよい日数
  - warn: 有効期限切れをユーザーに警告開始する日数
  - grace: 期限切れからアカウントが無効になるまでの日数
  - expire: アカウントが無効になる日（エポック日起点の日数）
  - reserved: 予約フィールド

/etc/shadow には、次のようにユーザーごとに１つのレコード（１行）があります。

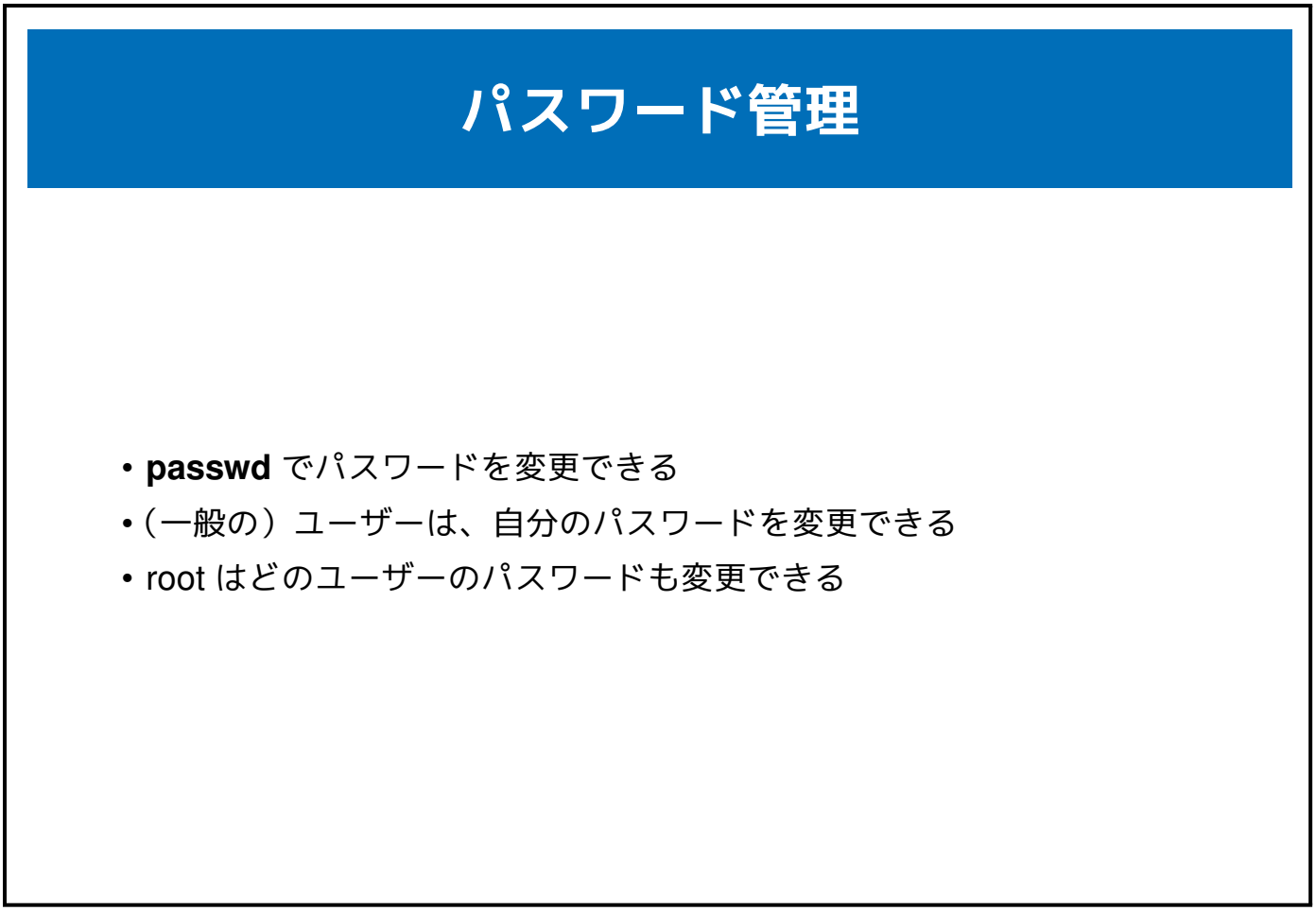
/etc/shadow

```
daemon:*:16141:0:99999:7:::
.....
beav:$6$iCZyCnBJH9rmq7P.$RYNm10Jg3wrhAtUnahBZ/mTMg.RzQE6iBXyqaXHvxxbK\
TYqj.d9wpoQFuRp7fPEE3hMK3W2gcIYhiXa9MIA9w1:16316:0:99999:7:::
```

各レコードのユーザー名は、/etc/passwd のユーザー名との完全一致が必要で、さらに同じ順序で表示される必要があります。
全ての日付は 1970 年 1 月 1 日（**エポック日**）からの日数として保存されます。
“6” の後に８文字のソルト値、その後に $ と 88 文字（sha512）のパスワード ハッシュが続きます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.7 パスワード管理



デフォルトでは、選択されたパスワードが pam_cracklib.so でチェックされ、より安全なパスワードを推薦されることがあります。
一般ユーザーが、自分のパスワードを変更する場合は

```
$ passwd
```

```
Changing password for bjmoose
(current) UNIX password: <bjmoose's password>
New UNIX password: <bjmoose's-new-password>
Retype new UNIX password: <bjmoose's-new-password>
passwd: all authentication tokens updated successfully
```

root が一般ユーザーのパスワードを変更する場合、root は現在のパスワードの入力を求められないことに気をつけてください。

```
$ sudo passwd rjsquirrel
```

```
New UNIX password:  <rjsquirrel's-new-password>
Retype new UNIX password: <rjsquirrel's-password>
passwd: all authentication tokens updated successfully
```

一般ユーザーは、短すぎるものや辞書にある単語など、不適切なパスワードを設定することが許されていません。ただし root には許可されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.8 パスワードの有効期限の設定・確認

パスワードの有効期限の設定・確認（chage）

- chage 使って管理する

```
$ chage [-m mindays] [-M maxdays] [-d lastday] \
      [-I inactive] [-E expiredate] [-W warndays] user
```

- 例

```
$ sudo chage -l beaver
$ sudo chage -m 14 -M 30 wally
$ sudo chage -E 2012-4-1 eddie
$ sudo chage -d 0 june
```

一般に、パスワードを定期的に変更することは、重要と考えられています。こうすることで、もしパスワードがクラックされても、それを利用して侵入できてしまう期間を短くできます。また、使われなくなったアカウントのロックにもなります。しかし欠点もあります。このポリシーを煩わしく思ったユーザーが、頻繁に変更されるパスワードを忘れないように目につくところに書き留めてしまい、結果的にパスワードが盗まれやすくなることです。**chage** ユーティリティを使ってパスワード更新を管理します。

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays] user
```

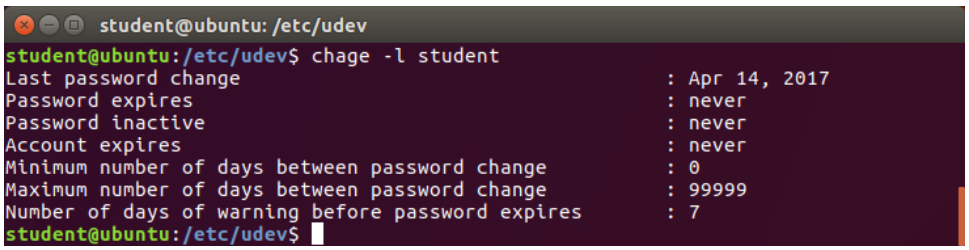


図 4.2: **chage の利用**

chage を利用できるのは root ユーザーだけです。ただし 1 つの例外があります。-l オプションを指定して実行すれば、誰でも自身のアカウントやパスワードの有効期限を確認できます。ユーザーに次回ログイン時にパスワード変更をを強制的するには

```
$ sudo chage -d 0 USERNAME
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.9 root アカウント

root アカウント

- root アカウントはシステムに対する全てのアクセス権がある
- ユーザーには決して root アクセス権を与えてはいけない
- root アカウントに変更するには

```
$ su
```

- root としてコマンドを実行するには

```
$ sudo command
```

- **sudo** の設定ファイルは `/etc/sudoers` と `/etc/sudoers.d`
- できる限り **su** を使わず、代わりに **sudo** を利用すべきである

root はあらゆるものにアクセスでき、あらゆることができる、非常に強力なアカウントです。

sudo は、通常のユーザーアカウントに一時的に root 権限を持たせることができます。**sudo** は、特定のアカウントだけにこの能力を持たせたり、特定のアカウントに特定のコマンドの昇格権限だけを持たせるように設定することができます。詳しくは **man sudo** を参照してください。

su（エスユーと発音し、スイッチまたは代理ユーザーを意味する）は、ユーザーがそのシェルを終了するまで昇格した特権を許可するサブシェル環境を作成します。そのサブシェルで実行される **すべて**のコマンドは、root ユーザーの昇格した権限で実行されます。

間違いは起こるものです！重要なファイルを削除してしまうような大きなミスを防ぐには昇格特権を必要としないコマンドを実行するために **su** サブシェルを使わないようにすべきです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

root アカウント

root の利用は、極めて限定的かつ例外的とすべきである

- 真に必要な場面に限定して root でログインする
  - 間違いを防ぐために
  - セキュリティ脆弱性を回避するために
- 権限を取得するには **su** を利用させる必要がある
  - 直接 root ではログインさせない
  - 誰が root になったかの履歴を監査できるよう設定する
- セキュリティ担保のために **sudo** を利用する

root アカウントの利用は管理目的だけに使用するもので、真に必須な場合だけに限定とし、決して通常アカウントとして使用してはいけません。root アカウントで何らかのミスをすると、整合性と安定性の両方にとって、そしてシステム セキュリティ面でも、致命的な犠牲を払うことになるリスクがあります。

セキュリティ上の理由から、デフォルトではネットワークを介した root ログインは禁止されています。**SSH** を使った **Secure SHell** ログインは許可されます。**SSH** は `/etc/ssh/sshd_config` と **PAM** (Pluggable Authentication Modules) で設定されます。**PAM** は pam_securetty.so モジュールと関連する `/etc/securetty` ファイルを使います。root ログインは `/etc/securetty` にリストされているデバイスからだけ許されます。

通常全ての root アクセスは **su** または **sudo** を使って行うことをお勧めします。（**sudo** を使うと全ての root アクセスを追跡調査できます）一部（**Ubuntu** など）のディストリビューションは、デフォルトでは root アカウントで直接ログインすることを禁止しています。

PAM を使用して **su** で root になることができるユーザーを制限することもできます。また root として実行された全てのコマンドのログを記録するように **auditd** を設定することも重要です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

4.10 SSH

SSH

- **SSH**: リモートシステムにログイン、またはコマンドを実行する
- **scp** が含まれる: リモートシステムとのファイルの送受信を行う
- セキュリティのために、強固な暗号アルゴリズムを利用する
  (そのため名前は: **S**ecure **SH**ell)

```
$ ssh remote_computer.com
$ ssh some_user@remote_computer.com
$ ssh some_user@remote_computer.com apt update
$ scp file.txt remote_computer.com:/tmp
$ scp usr1@rem1.com:/tmp/f.txt usr2@rem2.com:/tmp/nf.txt
```

- 複数のシステム上でコマンドを同時に実行

```
$ for machines in node1 node2 node3
  do
      (ssh $machines some_command &)
  done
```

または、もっと簡単なのは

```
$ pssh -viH node1 node2 node3 some_command
```

同一ユーザー名、または別ユーザー名でネットワーク経由でリモートシステムにログインする必要はよくあります。または、リモートマシンとの間でファイルを転送する必要があります。いずれの場合でも、傍受されない安全な方法でこれを行う必要があります。

SSH (**S**ecure **SH**ell) はこの目的を実現するためのものです。**SSH** は、強力なアルゴリズムに基づく暗号化を使用します。適切な **SSH** パッケージがシステムにインストールされていれば **SSH** を使うためにこれ以上セットアップする必要はありません。
リモートシステムにサインインするには以下のよう実行します。

```
$ ssh farflung.com
```

```
student@farflung.com's password: (ここにパスワードをタイプする)
```

farflung.com に `student` アカウントがあると仮定しています。別のユーザーとしてログインする場合は以下のように実行します。

```
$ ssh root@farflung.com
$ ssh -l root farflung.com
```

あるシステムから別のシステムにファイルをコピーするには（スペース節約のため、パスワード要求のやり取りは省略しました）

```
$ scp file.txt farflung.com:/tmp
$ scp file.tex student@farflung.com/home/student
$ scp -r some_dir farflung.com:/tmp/some_dir
```

複数のシステムで **pssh** (**Parallel SSH**) ユーティリティを使った、次のようなコマンドの一括実行が可能です。

```
$ pssh -viH machine1 machine2 machine3 do_something
```

全てのオプションを使いこなしてパスワードを扱うには **man** ページの **pssh** を読む必要があるかもしれません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SSH 設定ファイル

- ホームディレクトリには
  - `id_rsa`: ユーザーの **秘密鍵** である
  - `id_rsa.pub`: ユーザーの **公開鍵** である
  - `authorized_keys`: ログインを許可した公開鍵である
  - `known_hosts`: これまでにログインを認めたホストである
  - `config`: 色々なオプションを指定するファイルである
- **ssh-keygen** で、公開鍵と秘密鍵を生成する

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SSH をいろいろ設定するとさらに高度な使い方ができます。特に、パスワードなしのログインが可能になります。ユーザー固有の設定ファイルは、各ユーザーのホーム ディレクトリにある非表示の `.ssh` ディレクトリに作成されます。

ユーザーは最初に **ssh-keygen** を使用して、秘密鍵および公開鍵を生成する必要があります。

```
$ ssh-keygen
```

```
Generating public/private rsa key pair.
...
```

秘密鍵は **決して** 誰とも共有してはいけません！ 一方で公開鍵は、パスワード無しアクセスをしたいどんなマシンにも渡すことができます。

`authorized_keys` には、**そのマシンにあなたのアカウントでログインする** ことを許す公開鍵が含まれています。

```
$ cat authorized_keys
```

```
ssh-rsa AAAAB3Nza........student@debian
ssh-rsa AAAAB3Nza........student@fedora
```

`known_hosts` には、あたらしい SSH アクセスが発生するにつれて徐々に情報が蓄積されていきます。システムが SSH でログインしようとしているユーザーの情報が変更されたことを検知した場合には、警告が表示されアクセスを拒否するチャンスを提供しています。

```
$ cat known_hosts
```

```
x7 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXN..... M=
192.168.1.200 ecdsa-sha2-nistp256 AAAA..... bcs=
```

更に知りたい？

`man ssh_config` にアクセスして **SSH** のコンフィギュレーション パラメータに関する最新のベストプラクティスを確認してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SSH 設定ファイルの優先順位

設定ファイルを処理する順序は

1. **コマンド ライン** オプション
2. ~/.ssh/config
3. /etc/ssh/ssh_config

ファイルやコンテンツを参照する優先順序は **最初に見つかったものが優先**

/home/student/.ssh/config

```
Host apple
   HostName 192.168.0.196
   User student
   Port 4242
   IdentityFile /home/student/.ssh/custom

Host aws
   Hostname ec2-34-238-135-25.compute-1.amazonaws.com
   User ubuntu
   IdentityFile /home/student/.ssh/cloud1.pem
   ForwardX11 no
   PasswordAuthentication no

Host *
```

上記の ~/.ssh/config では、ホスト **apple** と **aws** に対して特定の設定情報が記載されています。これらのどちらも一致しない場合は、一般的なパラメータが適用されます。この構成に汎用パラメータはありません。コマンドの使用方法は次のとおりです。

```
$ ssh apple
```

または

```
$ ssh aws
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SSH を使ったクラウド接続

- クラウド上のホストにも SSH が機能し、一部が自動化される
- クラウド作成時にデフォルトユーザーに対する SSH キーが生成される
- 公開鍵はデフォルトユーザーの `authorized_keys` にコピーされる
- 公開鍵と秘密鍵をローカルシステムにコピーするオプションがある
- クラウド上のユーザーに対するパスワード認証は通常サポートされない

ssh-keygen: リモートシステムでは公開鍵を置き忘れることは珍しくありません。ssh-keygen を使えば公開鍵を再作成できます。

```
student@ubuntu:~/.ssh$ ~/.ssh$ ls -l
```

```
-rw------- 1 student student 411 Jul 11 07:01 id_ed25519
-rw-r--r-- 1 student student  96 Jul 11 07:01 id_ed25519.pub
^^I
```

公開鍵の名前を変更します。

```
student@ubuntu:~/.ssh$ mv id_ed25519.pub id_ed25519.pub.lost
student@ubuntu:~/.ssh$ ls -l
```

```
-rw------- 1 student student 411 Jul 11 07:01 id_ed25519
-rw-r--r-- 1 student student  96 Jul 11 07:01 id_ed25519.pub.lost
^^I
```

```
student@ubuntu:~/.ssh$ ssh-keygen -y -f ~/.ssh/id_ed25519 > ~/.ssh/id_ed25519.pub
student@ubuntu:~/.ssh$ ls -l
```

```
-rw------- 1 student student 411 Jul 11 07:01 id_ed25519
-rw-rw-r-- 1 student student  96 Jul 11 07:04 id_ed25519.pub
-rw-r--r-- 1 student student  96 Jul 11 07:01 id_ed25519.pub.lost
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

リモートからのグラフィカル ログイン

- フルグラフィカル デスクトップで、リモート マシンにログインする
- しばしば **VNC** (**V**irtual **N**etwork **C**omputing) が使われる
- よく使われる実装は **tigervnc** である
- 使い方は簡単である
  - リモート マシン側で (例えば some_machine):
    ```
    $ vncserver
    ```
  - ローカル マシン側で
    ```
    $ vncviewer -via server student@some_machine localhost:2
    ```

これをテストするには、まず **vnc** パッケージがインストールされていることを確認してください。

```
$ which vncserver vncviewer
```

```
/usr/bin/vncserver
/usr/bin/vncviewer
```

これらのプログラムが見つからない場合は、次のようにしてインストールする必要があります。

```
$ sudo [dnf|zypper|apt] install tigervnc*
```

適切なパッケージ管理システムを利用してください。(実際のパッケージ名は **Linux** ディストリビューションによってまちまちです。そのため、ここでは特定の名前をあげていません。結果的に必要がないものまでインストールすることになるかもしれませんが、パッケージのサイズは大きくはありません) 一般ユーザーとして VNC サーバーを起動するには以下のように実行します。

```
$ vncserver
```

以下の方法でテストができます。

```
$ vncviewer localhost:2
```

(現在実行しているものや、マシンの構成方法によっては、1、3、4 ... など、2 以外の数字を指定する必要があるかもしれません。)
リモートマシンから表示するには、少し変える必要があります。

```
$ vncviewer -via student@some_machine localhost:2
```

“color profile” のために認証する必要があるという奇妙なメッセージが表示され、パスワードが機能しない場合は次のように
サーバーマシンの **colord** デーモンを強制終了する必要があります。

```
$ sudo systemctl stop colord
```

これはバグで (機能ではない)、一部のディストリビューションおよび一部のシステムでだけ出現し、原因はよくわかっていません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4.11 演習

デモ教材ビデオ

using_user_accounts_demo.mp4

課題 4.1: アカウント

あなたは、どのアカウントでログインしていますか？ どうやってそれを確認しますか？

解 4.1 id ユーティリティを実行します。

```
$ id
```

```
uid=1000(user) gid=1000(user) groups=1000(user),999(qubes)
```

注目

これには色々な方法があります。いくつかの例を示します。
$ whoami
$ grep ^Uid /proc/$$/status

課題 4.2: ユーザー アカウントの処理

1. /etc/passwd と /etc/shadow について、それぞれのファイルの、特に一般のユーザーのアカウントの、各フィールドを比較します。同じものと違うものは何でしょうか。
2. **useradd** を使って user1 アカウントを作成します。
3. ssh で user1 としてログインします。以下のようにします。

   ```
   $ ssh user1@localhost
   ```

   user1 のパスワードが必要なので、これは失敗します; まだ接続できていません。
4. user1 のパスワードを user1pw に設定し改めて user1 としてログインします。
5. /etc/shadow ファイルの新しく作られたレコードを調べます。
6. /etc/default/useradd ファイルを見て、現在のデフォルトが何であるかを確認します。また /etc/login.defs ファイルも調べます。
7. デフォルトのシェルとして **Korn** シェル（**ksh**）を利用する新しいユーザーアカウントuser2 を作成します。（**/bin/ksh** が無ければインストールするか、代わりに **C** シェル（**/bin/csh**）を使ってください）パスワードをuser2pw に設定します。
8. /etc/shadow を見てください。user1 アカウントの現在の有効期限はいつですか？
9. **chage** を使って、user1 の有効期限を 2013 年 12 月 1 日に設定してください。/etc/shadow を見てください。新しい有効期限はいつですか？
10. **usermod** を使って、user1 アカウントをロックします。/etc/shadow を見て user1 のパスワードに関して、変更された点を見つけてください。パスワードを userp1 にリセットして、この演習を終えてください。

解 4.2

1. $ sudo grep student /etc/passwd /etc/shadow

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
/etc/passwd:student:x:1000:100:LF Student:/home/student:/bin/bash
/etc/shadow:student:$6$jtoFVPICHhba$iGFFUU8ctrtOGoistJ4/30DrNLi1FS66qnn0VbS6Mvm
  luKI08SgbzT5.IcOHo5j/SOdCagZmF2RgzTvzLb11H0:16028:0:99999:7:::
```

（student の代わりに、一般のユーザーの任意の名前が利用できます）ユーザー名欄だけがマッチします。

2. `$ sudo useradd user1`

3. `$ ssh user1@localhost`

```
user1@localhost's password:
```

最初に **sshd** サービスを起動しなければなりません。

```
$ sudo service sshd restart
```

または

```
$ sudo systemctl restart sshd.service
```

4. `$ sudo passwd user1`

```
Changing password for user user1.
New password:
```

5. `$ sudo grep user1 /etc/passwd /etc/shadow`

```
/etc/passwd:user1:x:1001:100::/home/user1:/bin/bash
/etc/shadow:user1:$6$OBE1mPMw$CIc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2
   TFpucuwRN1CCHZ19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

6. 例として **RHEL** や **openSUSE** システムで説明します:

```
$ cat /etc/default/useradd
```

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

```
$ cat /etc/login.defs
```

```
....
```

２つ目のファイルは長いので省略します。自分のシステム上で確認してください。

7. `$ sudo useradd -s /bin/ksh user2`
   `$ sudo passwd user2`

```
Changing password for user user2.
New password:
```

8. `$ sudo grep user1 /etc/shadow`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
user1:$6$OBE1mPMw$CIc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuwRN1CCHZ
   19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

有効期限が設定されていません。

9. `$ sudo chage -E 2013-12-1 user1`
   `$ sudo grep user1 /etc/shadow`

```
  user1:$6$OBE1mPMw$CIc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuwRN1CCHZ
19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7::16040:
```

10. `$ sudo usermod -L user1`

    `$ sudo passwd user1`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 5 章

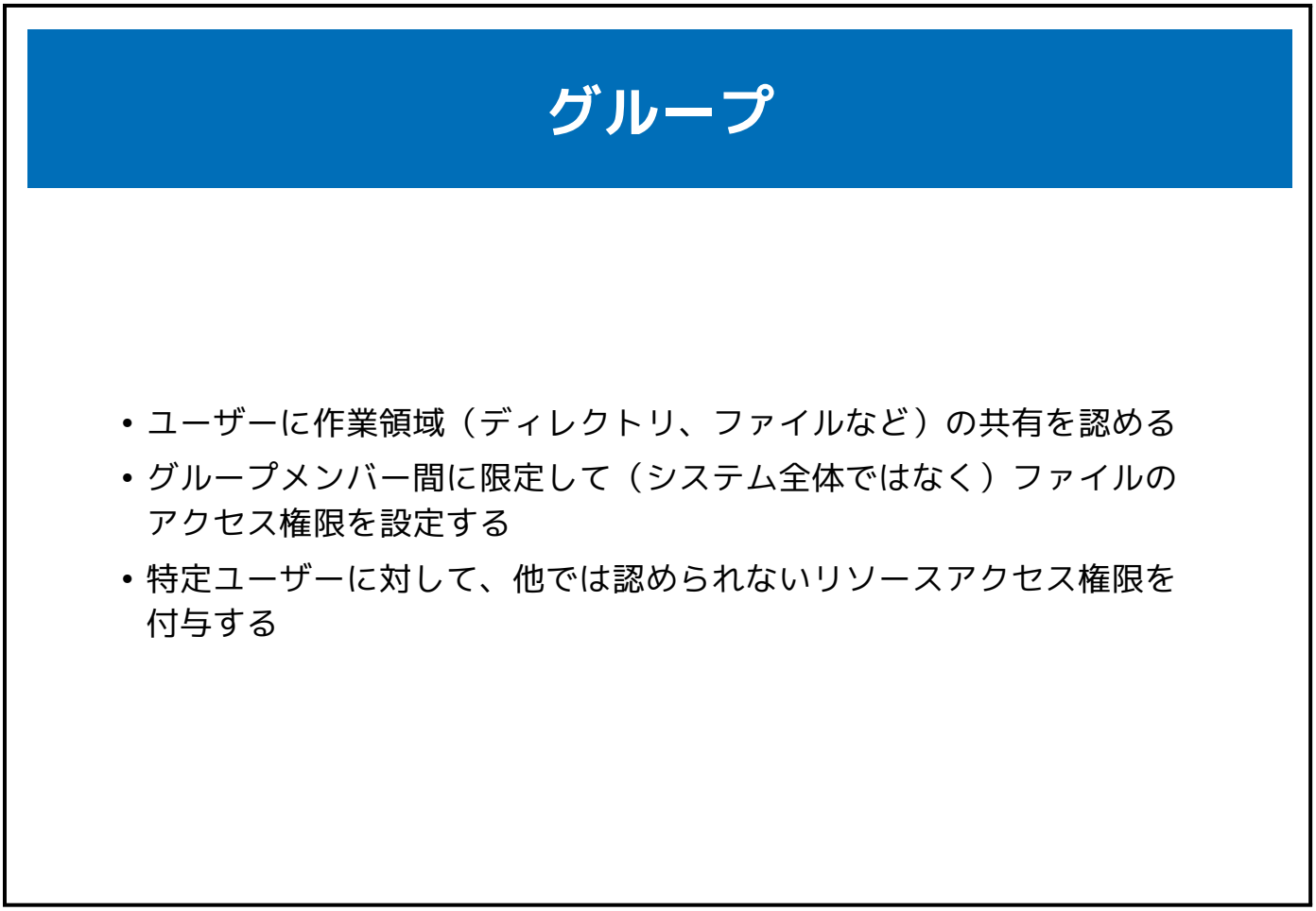
グループ管理

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- Linux ユーザを 1 つ以上のグループに所属させることが有用である理由を説明する。
- グループメンバーシップの概念について説明する。
- **groupadd**、**groupdel**、**groupmod**、**usermod** を使用して、グループとそのメンバーを作成、削除、操作する。
- ユーザープライベートグループを説明する。

5.1 グループ



Linux システムでは **グループ** と呼ばれるユーザーの集合を形成し、メンバーに複数の共通の目的を持たせることができます。さらに特定のファイルとディレクトリを共有しいくつかの特権を共有します。このようにシステム上の他のユーザーとは分けられた集団によって **世界（the world）**が形成されます。グループは、共同プロジェクトで非常に役立ちます。

ユーザーは、1 つ以上の **グループ**に属します。

グループは、`/etc/group` で定義されます。このファイルは、ユーザーに対して `/etc/passwd` が持っているのと同じ役割を果たします。ファイルの各行は次のようになります:

/etc/group

```
....
groupname:password:GID:user1,user2,...
....
```

- `groupname` は、グループの名前です。
- `password` は、パスワードのプレースホルダーです。グループ パスワードは、`/etc/gshadow` がある時だけ設定されます。
- `GID` は、グループ識別子です。0 から 99 はシステム グループに属します。100 から `GID_MIN` (`/etc/login.defs` で定義され通常は `UID_MIN` と同じ) は、特別なグループと見なされます。`GID_MIN` より大きな値は **UPG** (User Private Groups) となります。
- `user1,user2` は、グループのメンバーとなるユーザーをカンマ区切りしたリストです。ユーザーにとってこのグループがプリンシパル グループ（つまり、自分で作成したグループ）の場合は、ユーザーをここにリストする必要はありません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5.2 グループのメンバーシップ

グループのメンバーシップ

Linux ユーザーは:

- プライマリー グループを 1 つ持つ
  - プライマリー グループは、`/etc/passwd` に定義されている
  - `/etc/group` にもリストされる
- 0 から 15 個のセカンダリー グループに所属できる
- 以下のどちらかのコマンドで、所属するグループを確認できる

```
$ groups [user1 user2 ...]
$ id -Gn [user1 user2 ...]
```

- 引数を与えないと、各コマンドを実行しているユーザーについてレポートする（どちらのコマンドも共通である）

プライマリ グループの GID は、ユーザーがファイルまたはディレクトリを作成する時に常に使用されます。それ以外のセカンダリ グループのメンバーシップは、ユーザーにパーミッションを追加で付与します。

グループのメンバーシップは、**groups** または **id -Gn** コマンドのいずれかを実行することで識別できます。

デフォルト グループは、ディストリビューションとインストールに依存します。

CentOS では:

```
[student@CentOS ~] groups
```

```
student
```

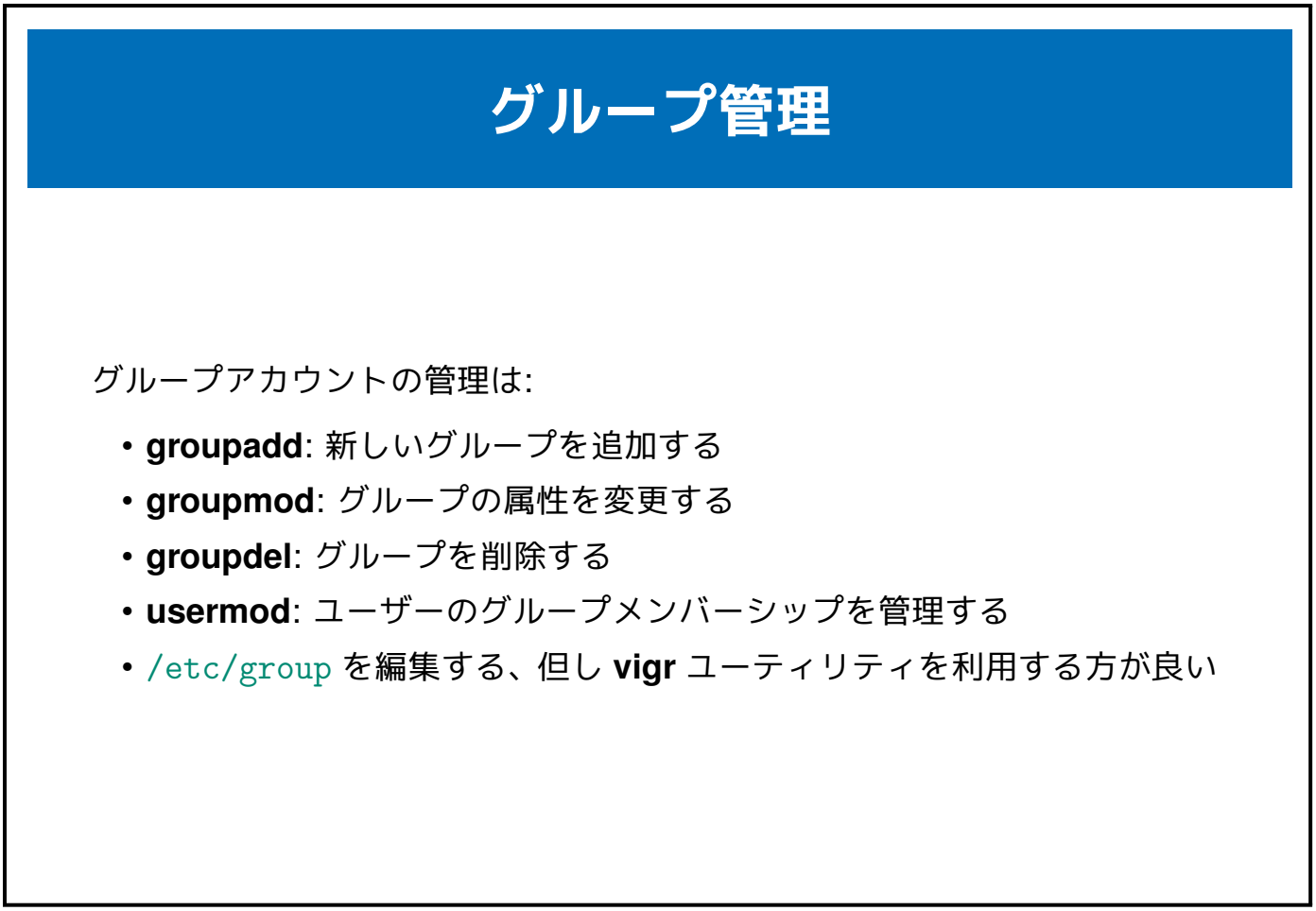
Ubuntu では:

```
student@ubuntu ~$ groups
```

```
student adm cdrom sudo dip plugdev lpadmin sambashare libvirt
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5.3 グループ管理



`/etc/group` を直接編集することも可能ですが、既に紹介した **vipw** ユーティリティに通常シンボリックリンクされている **vigr** ユーティリティを利用する方法が、より望ましいでしょう。

これらのグループ操作ユーティリティは、`/etc/group` と（存在する場合は）`/etc/gshadow` を変更します。
この操作は root だけが実行できます。

使用例

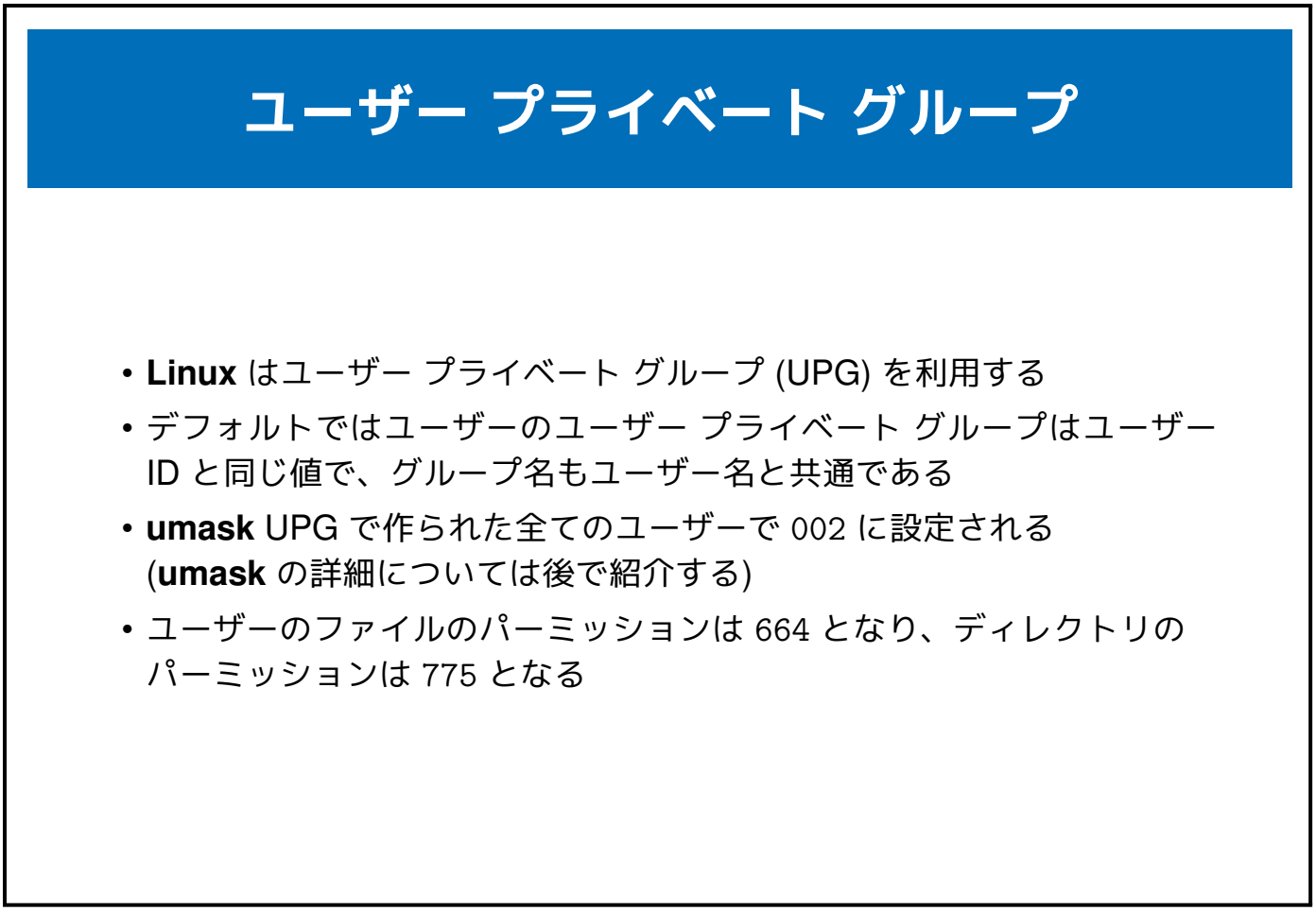
```
$ sudo groupadd -r -g 215 staff
$ sudo groupmod -g 101 blah
$ sudo groupdel newgroup
$ sudo usermod -G student,group1,group2 student
```

重要

注意：**usermod -G** コマンドはグループのリスト全体を対象として、一つのコマンドでグループの削除と追加を行います。
安全な（破壊的でない）使い方をしたいなら、既存のグループメンバーを保持する **-a** オプションを使うべきです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5.4 ユーザー プライベート グループ



Linux は **ユーザー プライベート グループ（UPG）**機能を利用します。

UPG の背景にある考え方は、各ユーザーに自分のグループを持たせることです。ただし、UPG は個人専用であるとは **保証されません**。`/etc/group` で定義したプライベートグループに、他のメンバーが追加できてしまうからです。

デフォルトでは **useradd** で作成されたアカウントを持つユーザーは、プライマリ GID = UID となっていて、グループ名もユーザー名と同じです。

`/etc/profile` で指定されているとおり、UPG で作成された全てのユーザーの **umask** は 002 に設定されます。したがって、この方式ではユーザーのファイルはパーミッション 664 (`rw-rw-r--`)、ディレクトリは 775 (`rwxrwxr-x`) で作成されます。（**umask** については次のセクションで説明します）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5.5 演習

課題 5.1: グループに関連する作業

1. ２つの新しいユーザーアカウント（下記に示すように rocky と bullwinkle）を作成します。ホームディレクトリが作成されていることを確認します。
2. ２つの新しいグループ、friends と bosses （GID を 490 として）を作成します。/etc/group を確認してください。各グループに割り当てられた GID を確認してください。
3. ２つの新しいグループに rocky を追加してください。
   グループ friends に winkle を追加してください。
   /etc/group の変化を確認してください。
4. rocky としてログインしてください。ディレクトリ somedir を作成し、グループ所有者を bosses に設定します。（次セッションで説明する **chgrp** を使います）
   （rocky のホームディレクトリにある全てに対して、実行権限を追加する必要があるでしょう）
5. bullwinkle としてログインし、**touch** コマンドを使って somefile というファイルを /home/rocky/somedir に作成してください。実行できましたか？できませんね。それは、グループ所有権とディレクトリに chmod a+x を実行したからです。
6. bbosses グループに bullwinkle を追加し、再実行してください。新メンバー追加に効力を持たせるために、一旦ログアウトしてログインする必要があります。以下を実行してください:

解 5.1

1. ```
   $ sudo useradd -m rocky
   $ sudo useradd -m bullwinkle
   $ sudo passwd rocky
   ```

   ```
   Enter new UNIX password:
   Retype new UNIX password:
   passwd: password updated successfully
   ```

   ```
   $ sudo passwd bullwinkle
   ```

   ```
   Enter new UNIX password:
   Retype new UNIX password:
   passwd: password updated successfully
   ```

   ```
   $ ls -l /home
   ```

   ```
   total 12
   drwxr-xr-x  2 bullwinkle bullwinkle 4096 Oct 30 09:39 bullwinkle
   drwxr-xr-x  2 rocky      rocky      4096 Oct 30 09:39 rocky
   drwxr-xr-x 20 student    student    4096 Oct 30 09:18 student
   ```

2. ```
   $ sudo groupadd friends
   $ sudo groupadd -g 490 bosses
   $ grep -e friends -e bosses /etc/group
   ```

   ```
   friends:x:1003:
   bosses:x:490:
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

3. $ sudo usermod -G friends,bosses rocky
   $ sudo usermod -G friends bullwinkle

   $ grep -e rocky -e bullwinkle /etc/group

   ```
   rocky:x:1001:
   bullwinkle:x:1002:
   friends:x:1003:rocky,bullwinkle
   bosses:x:490:rocky
   ```

   $ groups rocky bullwinkle

   ```
   rocky : rocky friends bosses
   bullwinkle : bullwinkle friends
   ```

4. $ ssh rocky@localhost
   $ cd ~
   $ mkdir somedir
   $ chgrp bosses somedir
   $ ls -l

   ```
   total 16
   -rw-r--r-- 1 rocky rocky  8980 Oct  4  2013 examples.desktop
   drwxrwxr-x 2 rocky bosses 4096 Oct 30 09:53 somedir
   ```

   $ chmod a+x .

5. $ ssh bullwinkle@localhost
   $ touch /home/rocky/somedir/somefile

   ```
   touch: cannot touch /home/rocky/somedir/somefile: Permission denied
   ```

   $ exit

6. $ sudo usermod -a -G bosses bullwinkle
   $ ssh bullwinkle@localhost
   $ touch /home/rocky/somedir/somefile
   $ ls -al /home/rocky/somedir

   (ファイルのオーナーを記録する)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 6 章

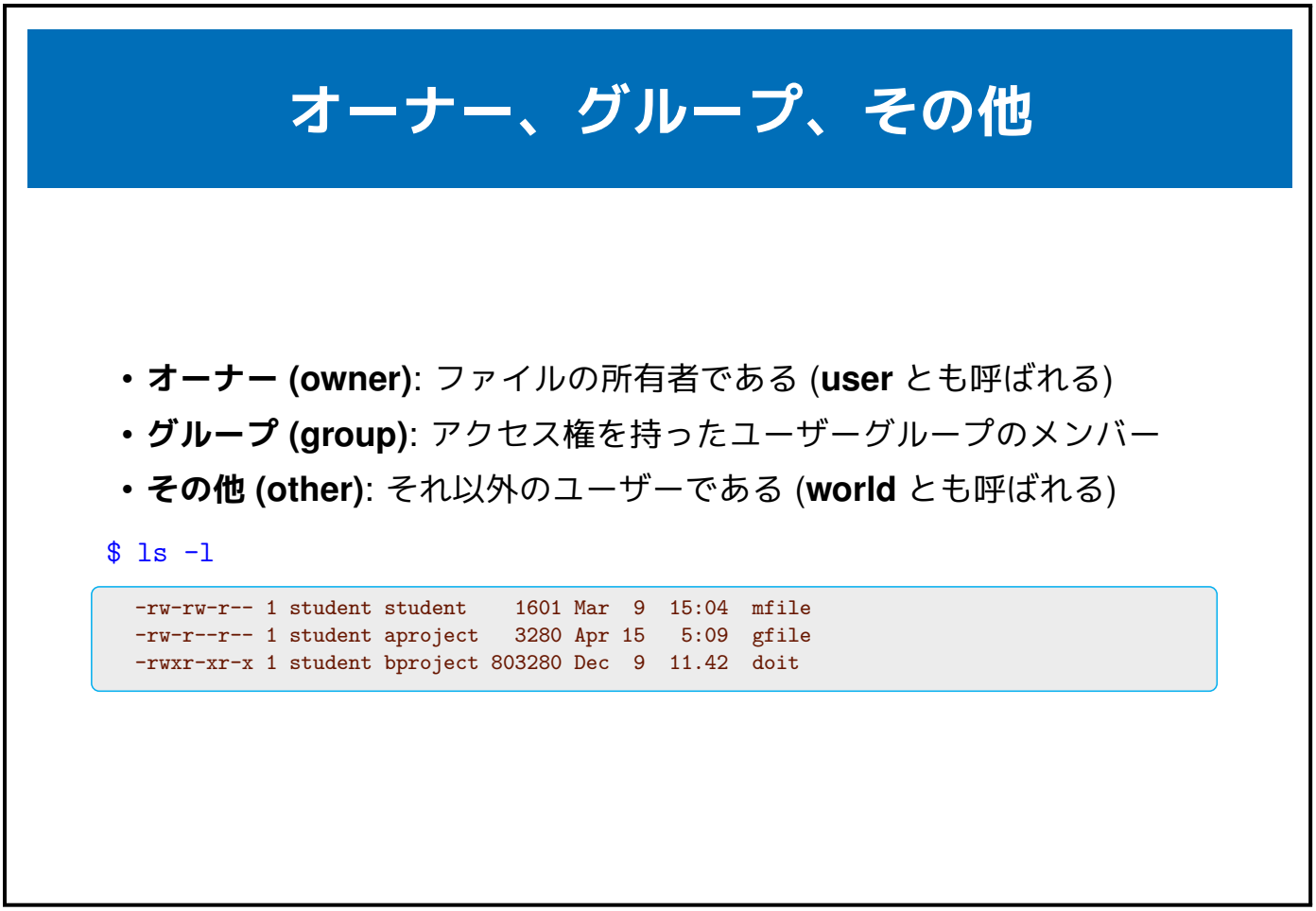
ファイルのパーミッションと所有権

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- オーナー、グループ、ワールドの概念を説明する。
- カテゴリーごとにファイルのアクセス権（読み取り、書き込み、実行）を設定する。
- ファイルへのアクセス要求を認証し、適切なパーミッションを遵守する。
- ファイルのパーミッションを変更するには **chmod** を、ユーザーの所有権を変更するには **chown** を、グループの所有権を変更するには **chgrp** を使用する。
- 新しく作成されたファイルに希望するパーミッションを設定する **umask** の役割を理解する。
- **ACL** を使ってユーザー、グループ、ワールドや読み取り、書き込み、実行を簡単に拡張する。

6.1 ファイルのパーミッションと所有権



以下を実行すると:

```
$ ls -l a_file
```

```
-rw-rw-r-- 1 coop aproject 1601 Mar 9 15:04  a_file
```

先頭の 1 文字（は、ファイルオブジェクトの種類を示しています）の後に続く 9 文字に、ファイルを利用するであろう人に付与される **アクセス権** が表現されています。まず、利用者が 3 つのカテゴリー（**オーナー**、**グループ**、**その他** (これは **world** と同じ)）に分類され、それぞれに対して 3 文字が付与されています。

上記の出力リストでは、ユーザーは `coop` でグループは `aproject` です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

6.2 ファイルのアクセス権

ファイルのアクセス権

- 以下の例のように -l を付けてファイルリストを取得すると

```
$ ls -l /usr/bin/vi
-rwxr-xr-x. 1 root root 910200 Jan 30  2014 /usr/bin/vi
```

それぞれの３文字の組み合わせ（２番文字目から１０文字目まで）には、許可する操作について該当箇所を設定し、許可しない場合には **-** とする

  - **r**: 読み込みを許可
  - **w**: 書き込みを許可
  - **x**: 実行を許可

- 以下の順に認証を試し、最初に成功したものを基準に権限が付与される
  - ファイル所有者か
  - グループのメンバーシップか
  - その他ユーザー（=world）のパーミッションで許可されるか

パーミッションが許可されない場合は、これらの文字の代わりに -（ダッシュ）が表示されます。

これに加えて **setuid/setgid** パーミッションなどカテゴリごとに他の特殊パーミッションが存在します。

これらの **ファイル アクセス パーミッション** は **Linux** セキュリティ システムの根幹をなすものです。全てのファイル アクセス要求に対して、要求したユーザーの権限とアカウント識別情報が、ファイルの所有者のものと比較されます。

この **認証** は、アクセスを要求したユーザーが利用者カテゴリー３種類のどれに属しているかに従って与えられます。
以下の順序で確認されます。

1. オーナー（＝ファイル所有者）からの要求であれば、ファイル所有者のパーミッションを使用されます。
2. 要求者がオーナー以外で、ファイルに付与されたグループに属している場合は、グループのパーミッションを適用します。
3. どちらにも該当しない場合は、その他ユーザー (world) のパーミッションが適用されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

6.3 chmod, chown と chgrp

chmod

- ファイルパーミッションの変更には **chmod** を使う
- 例えば「オーナ」と「その他」に実行権を付与し、「グループ」からは書き込み権を剥奪するには

```
$ ls -l a_file
-rw-rw-r-- 1 coop coop 1601 Mar  9 15:04 a_file

$ chmod uo+x,g-w a_file
$ ls -l a_file
-rwxr--r-x 1 coop coop 1601 Mar  9 15:04 a_file
```

u はユーザー（オーナー）の略である、o はその他（world）の略である、そして g はグループの略である

- スーパーユーザーでない限り、パーミッションを変更できるのは自分が所有するファイルだけである
- パーミッションは、しばしば 644 のように数字を使って指定される

パーミッションは、通常 8 進数ビットマップまたは記号で表現されます。
8 進数ビットマップでは 0755 のように、記号では `u+rwx,g+rwx,o+rx` のように記述します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

８進数ビットマップ表記

- **８進数表記**では、以下の各桁の設定の合計値を使う
  - 4（=3 ビット目）読み込み権を設定したい時
  - 2（=2 ビット目）書き込み権を設定したい時
  - 1（=1 ビット目）実行権を設定したい時

記号表記の構文は入力や覚えるのが煩雑なので、全てのパーミッションを１ステップで設定できる８進数ビットマップ表記を使用することが多いのです。アルゴリズムは単純で、８進数１桁で各エントリーの３つのパーミッション (r/w/x) を表現できます。

具体的には、7 なら読み取り／書き込み／実行、6 なら読み取り／書き込み、5 なら読み取り／実行、のパーミッションが付与されるという意味です。

chmod で 8 進数ビットマップ表記を利用する場合は３桁全てに値を指定します。

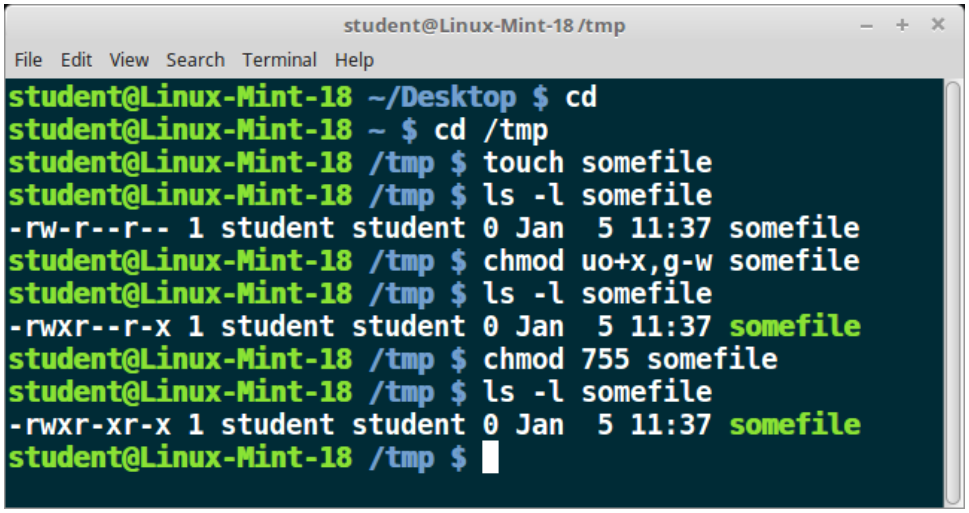


図 6.1: **chmod の利用**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

chown と chgrp

- **chown** でファイル所有者を変更する（スーパーユーザーだけが可能）

```
$ sudo chown wally somefile
```

- **chgrp** でグループ オーナーシップを変更する（グループメンバーだけが可能）

```
$ sudo chgrp cleavers somefile
```

- 両方を同時に変更する

```
$ sudo chown wally:cleavers somefile
```

オーナーとグループはコロン（または、ピリオド）で並べて表記する

- -R オプションを使うと再帰的に変更する

```
$ sudo chown -R wally:cleavers ./
$ sudo chown -R wally:wally subdir
```

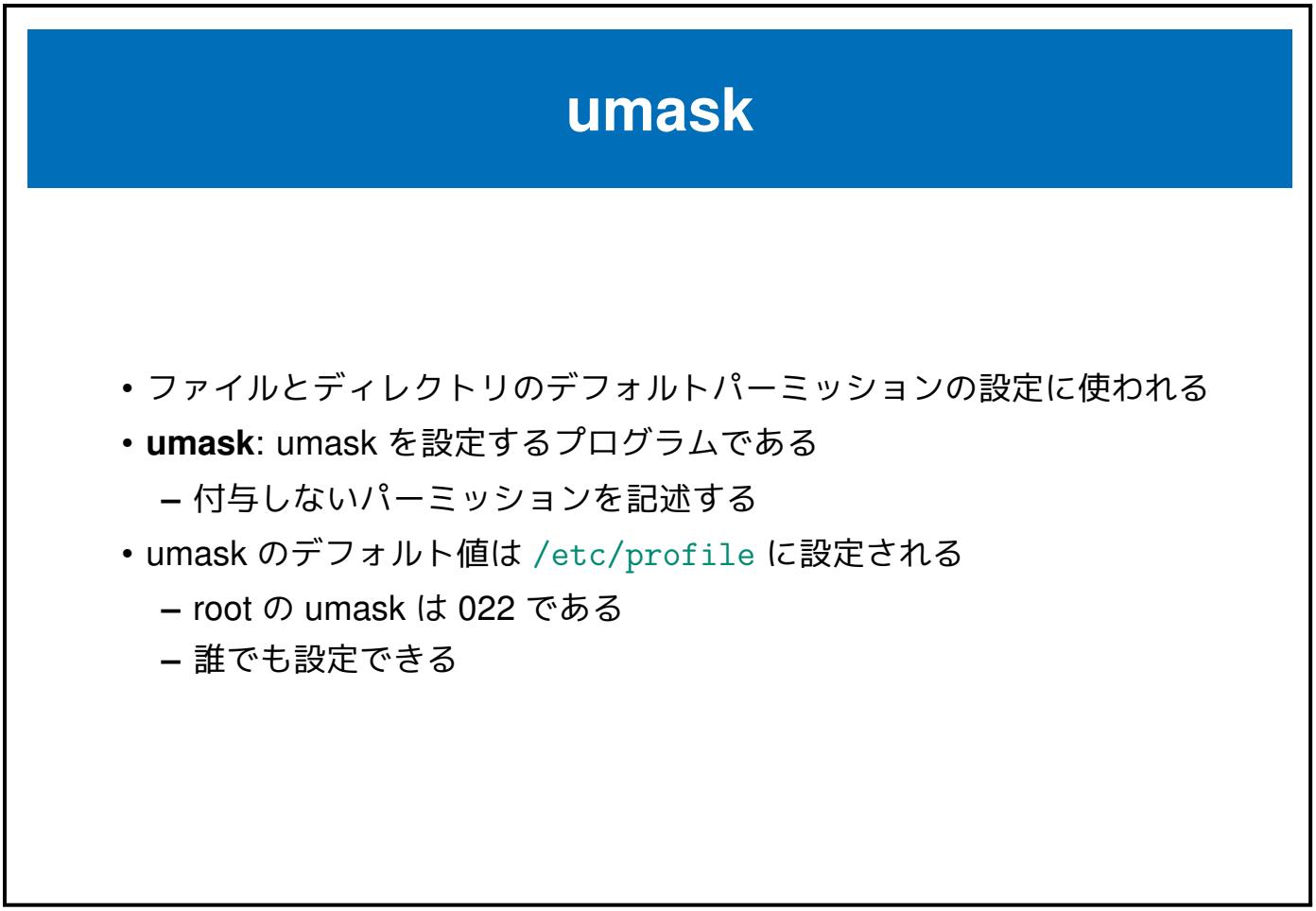
ファイル所有者の変更は **chown** で行い、グループの変更は **chgrp** で行います。

ファイル所有者の変更ができるのはスーパー ユーザーだけです。同様に、ファイルのグループ オーナーシップを変更できるのはグループのメンバーだけです。

-R オプションを使用すると、指定した場所以下の全てのファイルとディレクトリに対してオーナーシップが変更されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

6.4 umask



ファイル作成時に付与されるデフォルト パーミッションは、オーナー／グループ／その他ユーザー の全てに 読み取り／書き込み 許可 (0666) で、ディレクトリには全て 読み取り／書き込み／実行 (0777) です。しかし、以下を試すと

```
$ touch   afile
$ mkdir adir
$ ls -l | grep -e afile -e adir
```

```
drwxrwxr-x 2  coop coop 4096 Sep 16 11:18 adir
-rw-rw-r-- 1  coop coop    0 Sep 16 11:17 afile
```

実際にはファイルのパーミッションは 664 で、ディレクトリのパーミッションは 775 に変更されたことに気付くでしょう。これは、**umask** で定義された禁止パーミッション設定が反映された影響なのです。現在有効な **umask** 値を表示する方法は

```
$ umask
```

```
0002
```

これはシステム管理者がユーザーに設定する最も一般的な値です。**umask** 値とファイル作成時のパーミッション設定の論理積を計算することで最終的な値が決まります。

```
0666 & ~002 = 0664; つまり rw-rw-r--
```

次のように **umask** コマンドを使用して、いつでも **umask** の値を変更できます。

```
$ umask 0022
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

6.5 ファイルシステムの ACL

ファイルシステムの ACL

- **POSIX ACL** (**A**ccess **C**ontrol **L**ists) は「オーナー／グループ／その他」という単純なユーザーカテゴリー区分を拡張する
- パーミッション 777 を使わずに、ファイル／デフォルトを共有させる目的
- マウント時のオプション (`acl`) で利用可能になる
- インストール中のファイルシステム生成時に、デフォルトが設定される
- `getfacl`/`setfacl` を使って **ACL** の取得／設定を行う
- 例:

```
$ getfacl /home/stephane/file1
$ setfacl -m u:isabelle:rx /home/stephane/file1
$ setfacl -x u:isabelle    /home/stephane/file
```

オブジェクト、またはオブジェクト クラスにアクセスする、特定ユーザーまたはユーザーグループに、特別の権限を付与します。

Linux カーネルで **ACL** を有効にする時には、利用するファイルシステムにも **ACL** を実装する必要があります。最新の **Linux** ディストリビューションで使用される主要なファイルシステムには全て **ACL 拡張** が組み込まれています。マウント時にオプション `-acl` を指定すれば **ACL** を利用できます。**ACL** のデフォルト設定は、システムインストール時に作成されます。

新しくファイルを作ると、ファイルが生成されたディレクトリから（設定されていれば）デフォルトの **ACL** が継承されます。また **mv** と **cp -p** は **ACL を継承** することにも注意してください。

ACL の削除は

```
$ setfacl -x u:isabelle /home/stephane/file1
```

ディレクトリにデフォルトを設定するには

```
$ setfacl -m d:u:isabelle:rx somedir
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

6.6 演習

デモ教材ビデオ

using_umask_demo.mp4

課題 6.1: chmod を使う

chmod を使うとき、8 進数表示か記号表記かのいずれかを使うことができます。記号表記について学びましょう。

パーミッションは直接指定するか、追加するか、削除するか、いずれの方法でも指定できます。例を見てみましょう。

```
$ chmod u=r,g=w,o=x afile
$ chmod u+w,g-w,o+rw afile
$ chmod ug=rwx,o-rw afile
```

各実行の後:

```
$ ls -l afile
```

を実行してパーミッションがどうなったかを見てみます。そして、他のオプションを指定してみましょう。

課題 6.2: umask

空のファイルを作成します。

```
$ touch afile
$ ls -l afile
```

```
-rw-rw-r-- 1 coop coop 0 Jul 26 12:43 afile
```

所有者とグループが読み書きできることがわかります。その他のユーザーは、読み出しだけ許可されています。

オペレーティング システム レベルでは、ファイルやディレクトリを作成すると標準では所有者、グループが読み書きでき、**かつ**その他のユーザーは読む込だけが許可されています。(0666) **umask** コマンドで標準値を変更できます。

引数なしで **umask** を実行すると現在の値を表示します。

```
$ umask
```

```
0002
```

ユーザーにとって一番良いと管理者が指定した値が設定されています。これは、ファイル作成時のパーミッションと組み合わせて使用されます:

```
0666 & ~002 = 0664; すなわち rw-rw-r--
```

umask を変更してファイルを作成し、パーミッションがどうなるかを見てみましょう。

```
$ umask 0022
$ touch afile2
$ umask 0666
$ touch afile3
$ ls -l afile*
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 6.3: アクセス コントロール リストを使う

1. ユーザー名をファイル名称としてファイルを作成し **getfacl** を実行し、プロパティを見ます。
2. 標準のプロパティで、新しいアカウントを作成します。(または、以前の演習で作成したものを利用します)
3. ユーザーにログインし、最初に作成したファイルに一行追加します。これは失敗します。
4. **setfacl** を使って、新ユーザーの書き込みを許可します。
5. **setfacl** を使って、新ユーザーの書き込みを禁止します。
6. クリーンアップします。

解 6.3

端末を 2 つ開き、一つは自分のアカウント、他方は別のアカウントで作業します。

1. 端末 1 :

```
$ echo This is a file > /tmp/afile
$ getfacl /tmp/afile
```

```
getfacl: Removing leading '/' from absolute path names
# file: tmp/afile
# owner: coop
# group: coop
user::rw-
group::rw-
other::r--
```

2. 端末 2 :

```
$ sudo useradd fool
$ sudo passwd fool
...
```

3. 端末 2 :

```
$ sudo su - fool
$ echo another line > /tmp/afile
```

```
-bash: /tmp/afile: Permission denied
```

4. 端末 1 :

```
$ setfacl -m u:fool:rw /tmp/afile
$ getfacl /tmp/afile
```

```
getfacl: Removing leading '/' from absolute path names
# file: tmp/afile
# owner: coop
# group: coop
user::rw-
user:fool:rw-
group::rw-
mask::rwx
other::r--
```

端末 2 :

```
$ echo another line > /tmp/afile
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5. 端末１：

```
$ setfacl -m u:fool:w /tmp/afile
```

端末１：

```
$ echo another line > /tmp/afile
-bash: /tmp/afile: Permission denied
```

6. クリーンアップ

```
$ rm /tmp/afile
$ sudo userdel -r fool
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 7 章

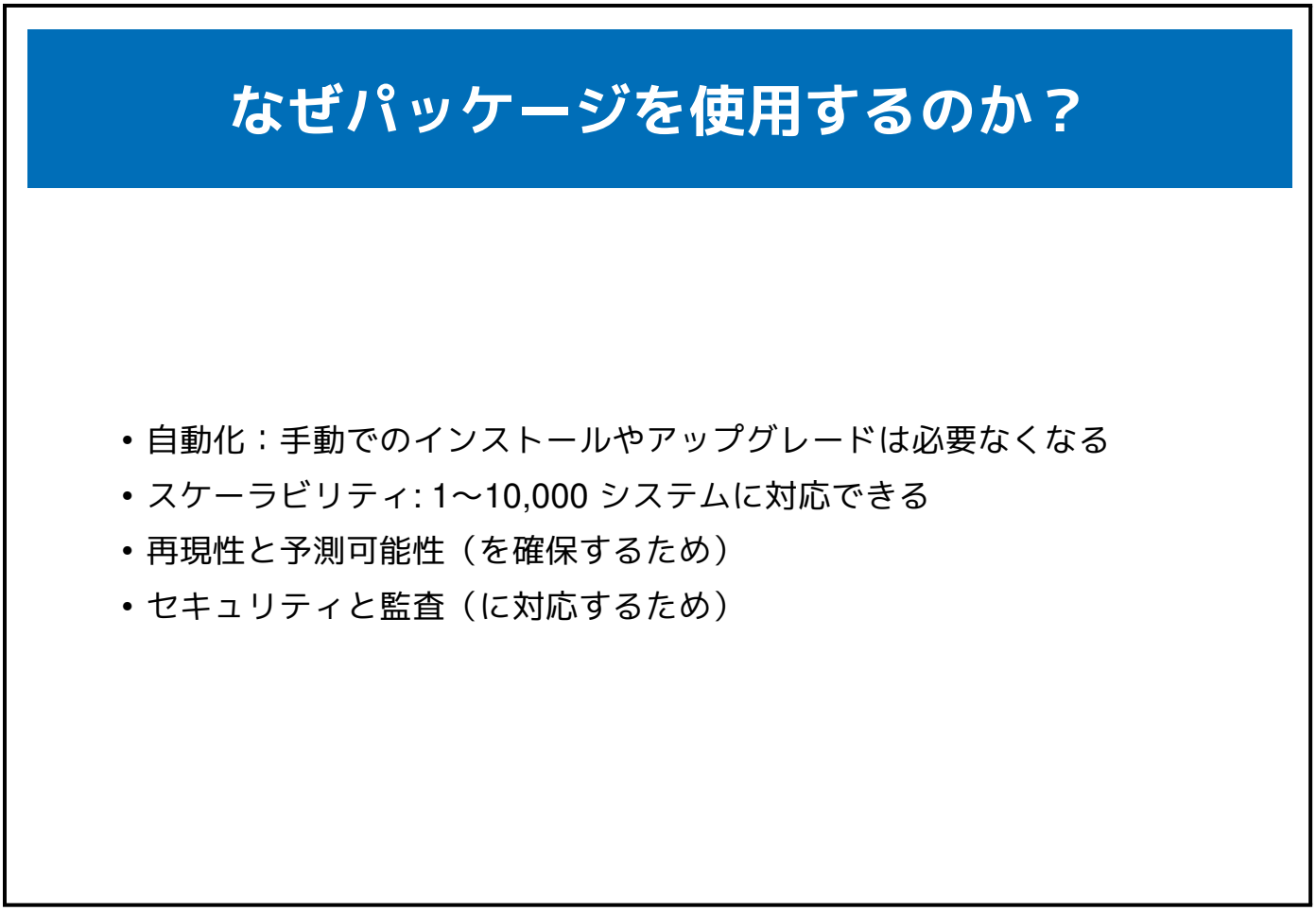
パッケージ管理システム

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- ソフトウェアパッケージ管理システムが必要な理由を説明する。
- バイナリ パッケージとソース パッケージの両方の機能を理解する。
- 利用可能な主なパッケージ マネジメント システムを列挙する。
- パッケージだけを扱うものと、パッケージ間の依存関係も解決する 2 種類のユーティリティがあることを理解する。
- 独自パッケージの作成で、ソフトウェアの中身やインストール方法を正確にコントロールできることを説明できる。
- ソース管理システムの役割を理解し、特に git を理解する。

7.1 なぜパッケージを使用するのか？



ソフトウェアパッケージ管理システムは、**Linux** がエンタープライズ IT 環境にもたらした最大の進歩の 1 つだと言われています。システム管理者は、自動化された汎用的で信頼性の高い方法でファイルとメタデータの記録を把握することにより、パッケージ管理システムを使用して、個々のシステムを手動で操作することなく数千のシステムにインストールプロセスを拡張できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.2 ソフトウェアパッケージの概念

ソフトウェアパッケージの概念

- 一つ以上の関連ソフトウェアを、１つのパッケージ（アーカイブ）に収集して圧縮、先に別パッケージがインストールされていることを前提としている場合もある
- ソフトウェアのインストールや削除を簡単にする
- 内部データベースを介して、ファイルの整合性を検証する
- パッケージの出所証明が可能となる
- アップグレードが容易となる
- 論理的な特徴によって、パッケージをグループ化できる
- パッケージ間の依存関係を管理できる

パッケージ管理システム は、システム管理者がソフトウェアパッケージのインストール、アップグレード、構成、および削除を既知で汎用的な方法で自動化できるツールを提供します。

特定のパッケージには、実行可能ファイル、データファイル、ドキュメント、インストールスクリプト、および構成ファイルが含まれる場合があります。またバージョン番号、チェックサム、ベンダー情報、依存関係、説明などの **メタデータ属性** も含まれます。

全ての情報は、インストール時に内部データベースにローカルに保存されます。これにより、バージョンと更新情報が簡単に照会できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.3 パッケージの種類

パッケージの種類

- **バイナリパッケージ** には実行可能ファイルやライブラリなどデプロイの準備ができているファイルが含まれる、バイナリーパッケージはアーキテクチャに依存する
- **ソースパッケージ** はバイナリパッケージの生成に使用される、ソースパッケージからバイナリパッケージを常に再構築できる必要がある、ソースパッケージは複数のアーキテクチャに対応する
- **アーキテクチャ非依存** パッケージにはドキュメントや構成情報と共に、インタプリターを使って動かすファイル／スクリプトが含まれる
- **メタパッケージ** は、デスクトップ環境やオフィススイーツのような比較的大規模なサブシステムのインストールに必要な、関連パッケージのグループが含まれる

通常、システム管理者はバイナリーパッケージを扱います。32 ビットプログラムを実行できる 64 ビットシステムの場合、ひとつのプログラムに対して 2 つのバイナリパッケージがインストールされていることもあります。片方の名前には **x86_64** または **amd64**、もう片方の名前には **i386** または **i686** が含まれます。

ソースパッケージは、バイナリパッケージの作成に使用されたソースコードの変更履歴追跡に役立ちます。通常デフォルトでは、システムにインストールされていませんが、ベンダーからいつでも入手できます。

必ずソースパッケージからバイナリパッケージが再構築できなければなりません。たとえば **RPM** ベースのシステムでは、以下を実行して **p7zip** バイナリパッケージを再構築できます。

```
# rpmbuild --rebuild -rb p7zip-16.02-16.el8.src.rpm
```

結果は `/root/rpmbuild` に置かれます。

```
# root/rpmbuild>find . -name "*rpm"
```

```
./RPMS/x86_64/p7zip-plugins-16.02-16.el8.x86_64.rpm
./RPMS/x86_64/p7zip-debugsource-16.02-16.el8.x86_64.rpm
./RPMS/x86_64/p7zip-plugins-debuginfo-16.02-16.el8.x86_64.rpm
./RPMS/x86_64/p7zip-16.02-16.el8.x86_64.rpm
./RPMS/noarch/p7zip-doc-16.02-16.el8.noarch.rpm
```

結果が置かれる場所は **Linux** ディストリビューションとバージョンに依存します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.4 利用可能なパッケージ管理システム

利用可能なパッケージ管理システム

図 7.1: **RPM と APT**

非常にポピュラーな 2 つのパッケージ管理システム

- **RPM** (**Red Hat Package Manager**)
  Red Hat Enterprise Linux、**Fedora**、**CentOS** など **Red Hat** 派生の全てのディストリビューション、および **SUSE** とその関連コミュニティ **openSUSE** ディストリビューションで使用されている
- **dpkg** (**Debian Package**)
  Debian、**Ubuntu**、**Linux Mint** など **Debian** から派生した全てのディストリビューションで使用されている

他にも **Gentoo** が使用する **portage/emerge**、**Arch** が使用する **pacman**、および **組み込み Linux** システムと **Android** が使用する特殊なパッケージ管理システムなどがあります。

もう 1 つの古いシステムとして、実際の管理や削除の戦略なしにパッケージを **tarballs** として提供する方法があります。このアプローチは、最も古い **Linux** ディストリビューションの 1 つである **Slackware** でまだ使われています。

しかしほとんどの場合 **RPM** または **dpkg** のいずれかを使いますので、このコースではこれらを学んでいきます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.5 パッケージングツールのレベルと種類

パッケージングツールのレベルと種類

- **低レベルのユーティリティ**:
  単一パッケージ、パッケージリストをインストール／削除するもの
  - 依存関係を完全には解決できない、ワーニングかエラーを出すだけ
  - 他のパッケージが先にインストールされている必要がある場合、インストールは失敗する
  - そのパッケージが別のパッケージに必要な場合、削除は失敗する
- 例: **rpm**、**dpkg**
- **高レベルのユーティリティ**: 依存関係の解決ができる
  - 他に必要なパッケージがある場合、それもインストールする
  - 既にインストール済みのパッケージとの競合を回避する
- 例: **yum**、**dnf**、**zypper**、**PackageKit**、**apt**

パッケージングシステムには、２つのレベルがあります。

1. **低レベルのユーティリティ**: 単一のパッケージまたはパッケージのリストをインストールするか削除するだけのものであり、各パッケージには個別に具体的な名前が付けられています。依存関係は完全には処理されず警告だけが行われるかエラーになります。

   - 他のパッケージが先にインストールされている必要がある場合、インストールは失敗します。
   - そのパッケージが別のパッケージに必要な場合、削除は失敗します。

   rpm と **dpkg** ユーティリティがパッケージングシステムに対してこの役割を果たします。

2. **高レベルのユーティリティ**: 依存関係を解決

   - 他に必要なパッケージまたはパッケージグループがある場合、依存関係解決のためにそれらを先にインストールします。
   - パッケージを削除する場合それが他にインストールされているパッケージと関連している場合には、管理者は関連する全てのソフトウェアを削除するか、それとも削除作業そのものを中止するかを選択することができます。

   dnf と **zypper** ユーティリティ（と以前の **yum**）が **rpm** システムの依存関係を解決し、**apt** と **apt-cache** や、その他のユーティリティが **dpkg** システムの依存関係を解決します。

このコースでは、コマンドラインインターフェイスを使ったパッケージングシステムについてだけ説明します。各 **Linux** ディストリビューションで使用されるグラフィカルなフロントエンドは便利ですが、ここでは特定のインターフェイスに縛られない柔軟性を持ちたいと考えています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.6 パッケージ ソース

パッケージ ソース

- ディストリビューションは、複数の **リポジトリ** を持つ
- 全てのパッケージは、競合関係や不整合を引き起こすこと無く動作する必要がある
- 外部リポジトリの利用も可能だが、内部リポジトリときれいに整合できるケースと、競合してしまうケースがある
  - いくつかは、問題なく整合する
  - コンフリクトが発生するものもある

全てのディストリビューションには、システムユーティリティがソフトウェアを取得し新しいバージョンに更新するための、１つ以上のパッケージ **リポジトリ** があります。ディストリビューションの仕事は、リポジトリ内のすべてのパッケージが相互に適切に動作するよう環境を整備することです。

また、ディストリビューションの標準サポートリストに、いつでも外部リポジトリを追加することができます。これらはディストリビューションと密接に連携しているので、追加しても重要な問題が発生することはほとんどありません。例としては、バージョン依存のリポジトリ **EPEL** (**E**xtra **P**ackages for **E**nterprise **L**inux) セットがあります。このソースは **Fedora** であり、メンテナー達も **Red Hat** と親密なため、**RHEL** に適合するように作られています。

ただし、一部の外部リポジトリにはあまり構築も保守もされていないものがあります。たとえば、あるパッケージがメインリポジトリで更新された場合、関連するパッケージが外部リポジトリでは更新されない場合があり、このようなことが **依存関係地獄** といった事態を引き起こすことになります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.7 ソフトウェアパッケージの作成

ソフトウェアパッケージの作成

- カスタムソフトウェアのパッケージインストールを簡単にする
- インストール中に、パッケージに追加タスクの実行を許す
  - 必要なシンボリックリンクを作成する
  - 必要に応じてディレクトリを作成する
  - パーミッションを設定する
  - スクリプト化できるもの全て
- 色々なフォーマットが利用できる
  - **RPM** は **Red Hat** と **SUSE** ベースのシステムが利用している
  - **DEB** は **Debian** と **Debian** ベースのシステムが利用している
  - **TGZ** は **Slackware** が利用している
  - **APK** は **Android** が利用している

独自のカスタムソフトウェアパッケージを構築すると、独自のソフトウェアを簡単に配布およびインストールできます。**Linux** のほとんど全てのバージョンにこれを行うためのメカニズムがあります。

独自のパッケージを作成すると、ソフトウェアの内容とインストール方法を正確に制御できます。パッケージは、新しいソフトウェアのインストールや、古いソフトウェアの削除に必要な全タスクを行うスクリプトを実行できるように作成する必要があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.8 リビジョン管理システム

リビジョン管理システムの利用目的

- 投稿者、ユーザー、テスターの増加に伴い、プロジェクト管理が複雑化
- **リビジョン コントロール システム** (**バージョン コントロール システム**)
  - 正確な変更履歴（ログ）を保存する
  - 以前のリリースへの巻き戻しに対応する
  - 競合する投稿の調停が可能である
- 色々なメソッドが使われている

ソフトウェア プロジェクトでは、プロジェクトサイズが大きくなったり貢献する開発者の数が増えたりするにつれ、管理がより複雑になります。

更新を整理し連携を促進するために、ソース管理にはさまざまなスキームが利用できます。このようなプログラムの標準機能には、変更の正確な履歴やログの保持、以前のリリースへのバックアップ、複数の開発者からの競合する可能性のある更新の調整などが含まれます。

ソース コントロール システム (または **リビジョン コントロール システム** とも一般に呼ばれる) は、共同開発を調整する役割を果たします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.9 利用可能なソース コントロール システム

利用可能なソース コントロール システム

Table 7.1: **利用可能なソース コントロール システム**

システム	URL
RCS	https://www.gnu.org/software/rcs/
CVS	https://www.nongnu.org/cvs
Subversion	https://subversion.apache.org
git	https://www.kernel.org/pub/software/scm/git/
GNU Arch	https://www.gnu.org/software/gnu-arch
Monotone	https://www.monotone.ca
Mercurial	https://www.mercurial-scm.org

利用可能なシステムにはプロプライエタリ、オープンそれぞれに多くの選択肢があります。上のリストは **GPL** ライセンスでリリースされたシステムの代表例です。（訳注：Subversion は Apache 2.0 でライセンスされています）

ここでは、Linux カーネル開発コミュニティから生まれ広く使用されているシステムである **git** だけに焦点を当てます。**git** は驚くほど短時間でオープンソースプロジェクトの主流となったもので、クローズドソース環境でもよく使用されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.10 Linux カーネルと git の誕生

Linux カーネルと git の誕生

- **Linux** カーネルの開発システムには特別な要件がある
  - 開発者は全世界に幅広く分散している
  - 開発に参加している開発者は数千人規模である
  - **GPL** ライセンスの下で完全に公開されている
- **git** はこれらのニーズを満たし **Linux** の効率的な成長を促した

git とその使用方法については、後の章で詳しく説明します。ここでは **git** がどのように動作し、どう使用するか演習で感じてみましょう。

Linux カーネルの開発システムには、実際のところ何千人もの開発者が関与し、世界中に広く配布されるという特別なニーズがあります。さらにそれは全て **GPL** ライセンスの下で公開されています。

Linux には長い間実質的なソースリビジョン管理システムはありませんでした。そこで主要なカーネル開発者は使用制限付きのライセンスが付与された商用プロジェクトの **BitKeeper** (https://www.bitkeeper.com) を **Linux** カーネル開発に使用しました。

しかし 2005 年春のライセンス制限に関する公開論争の結果、**Linux** カーネル開発に **BitKeeper** を無料で使用できなくなりました。

そこで求められたのが **git** の開発です。開発者は Linus Torvalds でした。**git** のソースコードは https://www.kernel.org/pub/software/scm/git/ のインデックスから入手でき、完全なドキュメントも https://www.kernel.org/pub/software/scm/git/docs/ から入手できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

git の仕組み

- 伝統的なバージョンコントロールシステムとは違うものである
- **ファイル** を基本単位とした管理ではない
- **オブジェクト データベース** には
  - **Blobs（ブロブ）**
  - **Trees（ツリー）**
  - **Commits（コミット）**
- **ディレクトリ キャッシュ** はディレクトリツリーの状態を取得する
- グラフィカルインターフェイスもある
- **github** などのサイトに数百万の **git** リポジトリがある

技術的にみて **git** は普通の意味でのソース管理システムでは **ありません**。またファイルを基本単位とした管理を行っていません。**オブジェクト** データベースとディレクトリキャッシュという 2 つの重要なデータ構造を持っています。

オブジェクトデータベースには 3 種類のオブジェクトが含まれます。

- **Blobs（ブロブ）：** ファイルの内容を含むバイナリデータの塊です。
- **Trees（ツリー）：** ファイル名と属性を含むブロブの集合。ディレクトリ構造を提供します。
- **Commits（コミット）：** ツリーのスナップショットを記述するチェンジセットです。

ディレクトリキャッシュは、ディレクトリツリーの状態をファイルに保存します。

ファイルをベースにしたシステムから管理システムを解放することにより、多くのファイルを含むチェンジセットを適切に処理できるようになりました。

git は早いペースで開発されており、グラフィカルインターフェイスも提供されています。たとえば https://www.kernel.org/git/ を参照してください。変更とソースのツリーを簡単に参照できます。

GitHub (https://www.github.com) は現在文字どおり何百万もの **git** リポジトリを公開／非公開にかかわらずホストしています。**git** を有効に使用する方法に関しては、記事、書籍、オンラインチュートリアルなどが簡単に見つかります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7.11 演習

デモ教材ビデオ

using_package_management_demo.mp4

課題 7.1: git によるバージョン コントロール

git の詳細については、後のセクションで説明します。ここでは **git** の使い方の基本的な感覚をつかむだけにしておきます。

git がインストール済であることを確認する

おそらく **git** はインストールされていると思います。`which git` コマンドでインストールされているかどうかがわかります。もしインストールされていなければ、ソースを入手、コンパイル、インストールする手もありますが、通常適切なコンパイル済のバイナリパッケージをインストールするのが簡単です。正確なパッケージ名称はいろいろありますが、ディストリビューションにより以下のどれかが使えるはずです。

```
$ sudo apt-get install git*  # Debian/Ubuntu
$ sudo zypper install git*   # openSUSE
$ sudo dnf install git*      # Fedora/RHEL 8/CentOS 8
$ sudo yum install git*      # RHEL 7/CentOS 7
```

使用しているディストリビューションによります。

どのように **git が使えるか** の感触を確かめて、その使いやすさを実感しましょう。ローカルのプロジェクトを作成します。

1. 作業用ディレクトリを作成し **git** を使うための初期化を実行します。

   ```
   $ mkdir git-test
   $ cd git-test
   $ git init
   ```

2. プロジェクトの初期化とはその下にバージョンコントロールに必要な情報を格納する `.git` ディレクトリを作成することです。元々のプロジェクトのファイルは変更しません。 `.git` ディレクトリの初期値は以下のような内容です。

   ```
   $ ls -l .git
   ```

   ```
   total 40
   drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 branches/
   -rw-rw-r-- 1 coop coop   92 Dec 30 13:59 config
   -rw-rw-r-- 1 coop coop   58 Dec 30 13:59 description
   -rw-rw-r-- 1 coop coop   23 Dec 30 13:59 HEAD
   drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 hooks/
   drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 info/
   drwxrwxr-x 4 coop coop 4096 Dec 30 13:59 objects/
   drwxrwxr-x 4 coop coop 4096 Dec 30 13:59 refs/
   ```

   あとで、このディレクトリとサブディレクトリの内容について説明します。開始時点ではほとんど空の状態です。

3. ファイルを作成し、プロジェクトに追加します。

   ```
   $ echo some junk > somejunkfile
   $ git add somejunkfile
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4. 以下のようにしてプロジェクトの状態を見ることができます。

```
$ git status
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

      new file: somejunkfile
```

これは、ファイルが **staged**（管理対象として追加された）の状態にあり、いまだ **commit** されていないことを意味します。

5. プロジェクトの責任者を **git** に登録します。

```
$ git config user.name "Another Genius"
$ git config user.email "b_genius@linux.com"
```

グローバルな構成ファイルで指定しない限り、全ての新プロジェクトで指定する必要があります。

6. それではファイルを修正して違いを見ましょう。

```
$ echo another line >> somejunkfile
$ git diff
```

```
diff --git a/somejunkfile b/somejunkfile
index 9638122..6023331 100644
--- a/somejunkfile
+++ b/somejunkfile
@@ -1 +1,2 @@
 some junk
+another line
```

7. 変更をレポジトリにコミットします。

```
$  git commit -m "My initial commit"
```

```
Created initial commit eafad66: My initial commit
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 somejunkfile
```

-m オプションで識別用のメッセージを指定しなかった場合、エディタが起動しメッセージを入力できるようになります。
コミットメッセージの入力は **絶対に** する必要があり、入力していない場合はコミットは拒否されます。エディタは EDITOR 環境変数で
指定したものが使われますが、GIT_EDITOR で設定を上書きすることができます。

8. 履歴を見ることができます。

```
$ git log
```

```
commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
Author: A Genius <a_genius@linux.com>
Date:   Wed Dec 30 11:07:19 2009 -0600

    My initial commit
```

ここで、いくつかの情報が表示されています。長い 16 進数の文字列は **コミット番号**（= commit から始まる文字列）です。それは 160 ビット、40 桁のユニークな識別子です。**git** はファイル名称ではなく、このちょっと想像が難しいものを利用します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

9. これで `git add` を使って、自由にファイルの修正、新規ファイルの追加ができます。しかし、新たに `git commit` するまではステージング（= commit 予定として登録されただけ）の状態です。
10. 入門的な内容を実施しただけですが、これで終了します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 8 章

dpkg

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- Debian Packaging System (**DPKG**) の考え方と使用方法について説明できる。
- バイナリーファイルとソース `.deb` ファイルの両方で使用される命名規則について説明できる。
- ソースパッケージがどのようなものか理解する。
- パッケージに対するクエリおよびベリファイ操作ができる。
- **Debian** パッケージのインストール、アップグレード、アンインストールができる。

8.1 DPKG (Debian パッケージ)

DPKG の要点

- **Debian** ベースの全ての **Linux** ディストリビューションが採用している（**Ubuntu** と **Linux Mint** を含む）
- **rpm** と同様に **dpkg** ユーティリティもパッケージ取得や依存性問題の解決は行わない
- パッケージファイルは拡張子に `.deb` を持つ
- **DPKG** データベースは `/var/lib/dpkg` に置かれている

DPKG (**Debian Package**) は、**Debian Linux** およびそれから派生した他のディストリビューションが採用する、ソフトウェアパッケージをインストール、削除、そして管理するためのパッケージングシステムです。**RPM** と同様に、インストールする（したい）パッケージをサーバーなどから直接取得する機能は提供されていないため、一度ローカルに保存されたパッケージ ファイルのインストールや削除を行います。

パッケージファイルの拡張子は、`.deb` で **DPKG** データベースは `/var/lib/dpkg` ディレクトリにあります。

rpm と同様に、**dpkg** プログラムは全体の一部分だけを見ているにすぎません。システムにインストールされているものと、コマンドラインによってインストールされたものは把握していますが、システム上のディレクトリやインターネット上に他のパッケージがあるのかどうかは把握していません。そのため、依存関係が満たされていない場合や、他のインストール済みパッケージに必要なパッケージを削除しようとした場合には失敗します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

8.2 パッケージのファイル名とソース

パッケージのファイル名とソース

- バイナリパッケージの標準的な命名形式

```
<name>_<version>-<revision_number>_<architecture>.deb
```

例：

```
logrotate_3.14.0-4_amd64.deb
logrotate_3.14.0-4ubuntu3_amd64.deb
```

- ソースパッケージは少なくとも３つのファイルで構成される
  - **アップストリーム tarball** (ファイル名の最後が .tar.gz など)
  - **ソースパッケージ概要ファイル** (ファイル名の最後が .dsc)
  - 二番目の tarball (ファイル名の最後が .diff.gz または .debian.tar.gz) にはアップストリームソースへのパッチとパッケージ用に作成された追加ファイルが含まれる

Debian パッケージのファイル名は、特定の情報を表すフィールドに基づいています。

歴史的な理由から、64 ビットの **x86** プラットフォームは **x86_64** ではなく **amd64** と呼ばれています。**Ubuntu** などのディストリビュータは、パッケージに独自の変更を加えた場合、Debian で提供されているパッケージとの違いを区別するために、パッケージのリビジョン番号にユニークな文字列（ubuntu3 など）を入れています。

Ubuntu システムでは、次に示すようにソースパッケージをダウンロードし、その後どのようなファイルがダウンロードまたは作成されたことを以下のように確認することができます。

図 8.1: **Ubuntu でソースパッケージを取得する**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

8.3 DPKG クエリ

DPKG クエリ

- インストールされている全てのパッケージをリストする

  ```
  $ dpkg -l
  ```

- wget パッケージにインストールされているファイルをリストする

  ```
  $ dpkg -L wget
  ```

- インストール済パッケージと、パッケージファイルに関する情報を表示する

  ```
  $ dpkg -s wget
  $ dpkg -I webfs_1.21+ds1-8_amd64.deb
  ```

- パッケージファイルに関する情報を表示する

  ```
  $ dpkg -c  webfs_1.21+ds1-8_amd64.deb
  ```

- /etc/init/networking.conf を所有しているパッケージを表示する

  ```
  $ dpkg -S /etc/init/networking.conf
  ```

- インストールされたパッケージの整合性を確認する

  ```
  $ dpkg -V package
  ```

特定のパッケージに関する情報を得たり、インストール後に変更されたファイルを探したりすることが重要な場合があります。例えば、特定のパッケージのどのバージョンがインストールされているかを確認するためです。

```
$ dpkg -s dpkg | grep -i version
```

```
Version: 1.19.7ubuntu3.2
```

Ubuntu 20.04 の例

追加の引数がない場合、ddpkg -V はシステム上のすべてのパッケージを検証します。

```
$ sudo dpkg -V
```

```
??5?????? c /etc/logrotate.conf
??5?????? c /etc/logrotate.d/apt
??5?????? c /etc/logrotate.d/bootlog
??5?????? c /etc/logrotate.d/rsyslog
??5?????? c /etc/logrotate.d/ufw
??5?????? c /etc/logrotate.d/apport
??5?????? c /etc/logrotate.d/speech-dispatcher
??5?????? c /etc/logrotate.d/unattended-upgrades
....
```

出力の解釈については **man** ページを参照してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

8.4 インストール／アップグレード／アンインストール

インストール／アップグレード／アンインストール

- **foobar** パッケージをインストール、またはアップグレードする

```
$ sudo dpkg -i foobar.deb
```

- 指定されたパッケージ（パッケージで提供されるファイル）を削除する（ただし構成ファイルは除く）

```
$ sudo dpkg -r package
```

- 構成ファイルを含めた、指定されたパッケージのファイルを削除する

```
$ sudo dpkg -P package
```

（インストールのため）`-i` オプションを使うときは

- パッケージが現在インストールされていない場合は、インストールします。
- パッケージが現在インストールされているパッケージよりも新しい場合は、アップグレードされます。

`-P` オプションは **パージ（purge）** (= パッケージに関する不要なデータを全て削除) を表すことに注意してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

8.5 演習

デモ教材ビデオ

using_dpkg_demo.mp4

課題 8.0: dpkg が必要です

Debian、Ubuntu、Linux Mint

本演習を実行するためには、**Debian**、**Ubuntu** や **Linux Mint** などの **Debian** ベースのシステムにアクセスできる必要があります。

課題 8.1: dpkg を使う

Debian パッケージを検索したり、検証したりする簡単な操作を学びます。

1. /etc/logrotate.conf が、どのパッケージに含まれているかを調べます。
2. パッケージに含まれている全てのファイルを参照しているパッケージに関する情報を表示します。
3. パッケージのインストールを検証します。
4. パッケージを削除します。

解 8.1

1. `$ dpkg -S logrotate.conf`

```
logrotate: /usr/share/man/man5/logrotate.conf.5.gz
logrotate: /etc/logrotate.conf
rsync: /usr/share/doc/rsync/examples/logrotate.conf.rsync
```

2. `$ dpkg -L logrotate`

```
...（表示内容省略）
```

3. `$ dpkg -V logrotate`

4. `$ sudo dpkg -r logrotate`

```
dpkg: dependency problems prevent removal of logrotate:
 ubuntu-standard depends on logrotate; however:
  Package logrotate is to be removed.
 libvirt-daemon-system depends on logrotate.

dpkg: error processing package logrotate (--remove):
 dependency problems - not removing
Errors were encountered while processing:
 logrotate
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 9 章

APT

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- **APT** とは何かを説明する。
- **APT** の主要なユーティリティプログラムを説明する。
- パッケージのクエリーを実行する。
- apt を使ったパッケージのインストール・削除・アップグレード。
- パッケージに関するシステム情報をクリーンアップしてコンパクトにする。

9.1 APT

APT とは何か？

- **A**dvanced **P**ackaging **T**ool の略である
- **apt** や **apt-cache** 等のユーティリティを含む (**dpkg** プログラムを下回りとして利用する)
- **Debian** パッケージを利用する (ファイル拡張子は `.deb` である)
- 複数の **Debian** ベースの **Linux** ディストリビューションのリポジトリを利用することも珍しくない

apt-get と apt

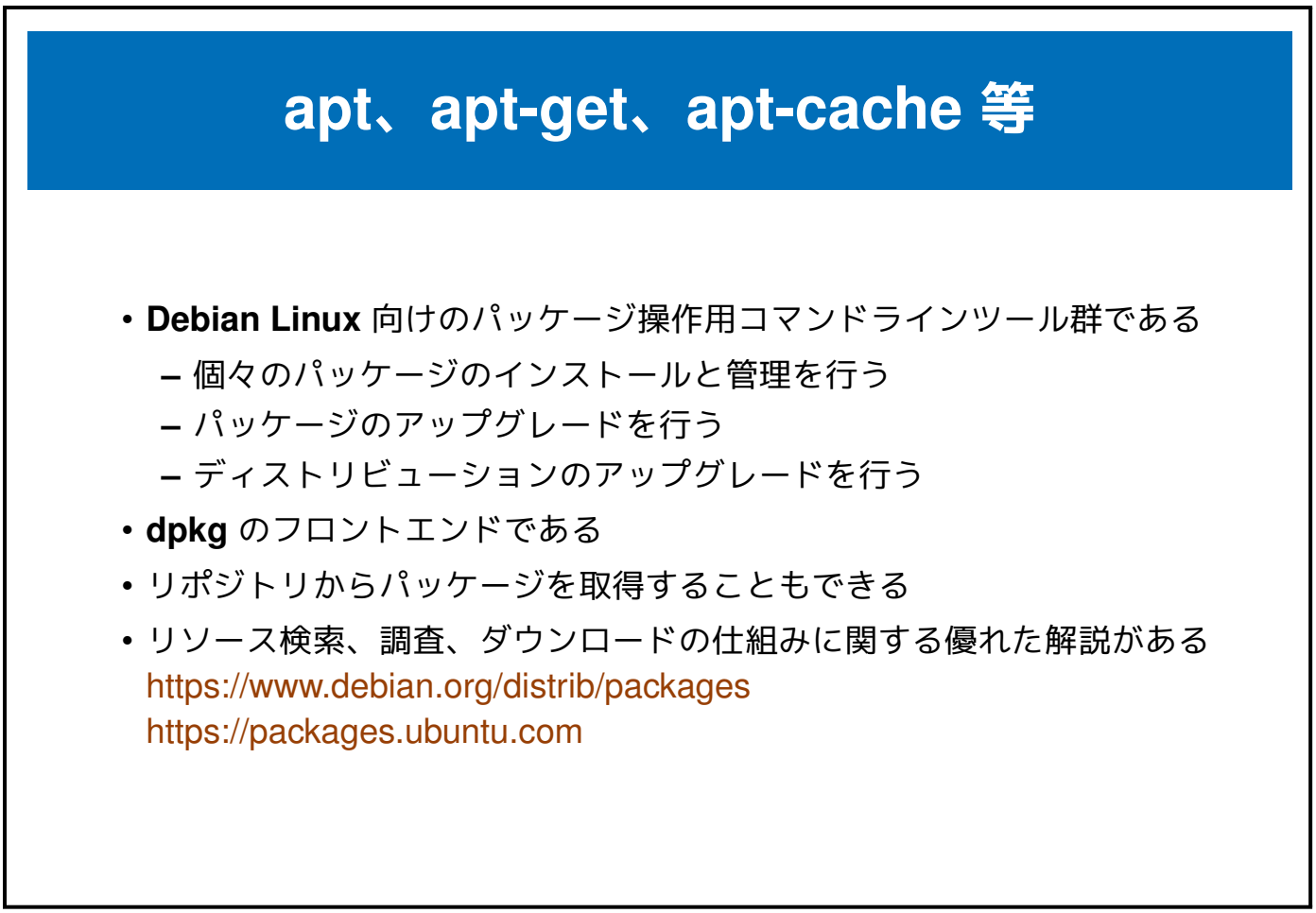
ほぼ全てのインタラクティブ操作に **apt-get** を使用する必要はなくなりました。**apt** という短い名前を使用できます。ただし **apt-get** の使用は習慣化されており、スクリプトでは使いやすいのです。参照するコマンドがコマンドラインとスクリプトの両方で利用可能という理由から、本コースでは **apt-get** を利用することがあります。

Debian ベースのシステムではプログラムの **APT** (**A**dvanced **P**ackaging **T**ool) は、**dpkg** プログラムを下回りに利用した高度なインテリジェントサービスを提供し、**Red Hat** ベースのシステムにおける **yum** と同様な役割を果たします。主なユーティリティは **apt** と **apt-cache** です。パッケージのインストール、更新、削除時に依存関係を自動的に解決できます。外部ソフトウェア **リポジトリ** にアクセスして同期を取り、必要に応じてソフトウェアを取得してインストールします。

繰り返しになりますが **synaptic** や **Ubuntu Software Center** のような（コンピュータ上の）グラフィカルインターフェイス、もしくは **aptitude** などの **APT** の古いフロントエンドはここでは扱いません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

9.2 APT ユーティリティ



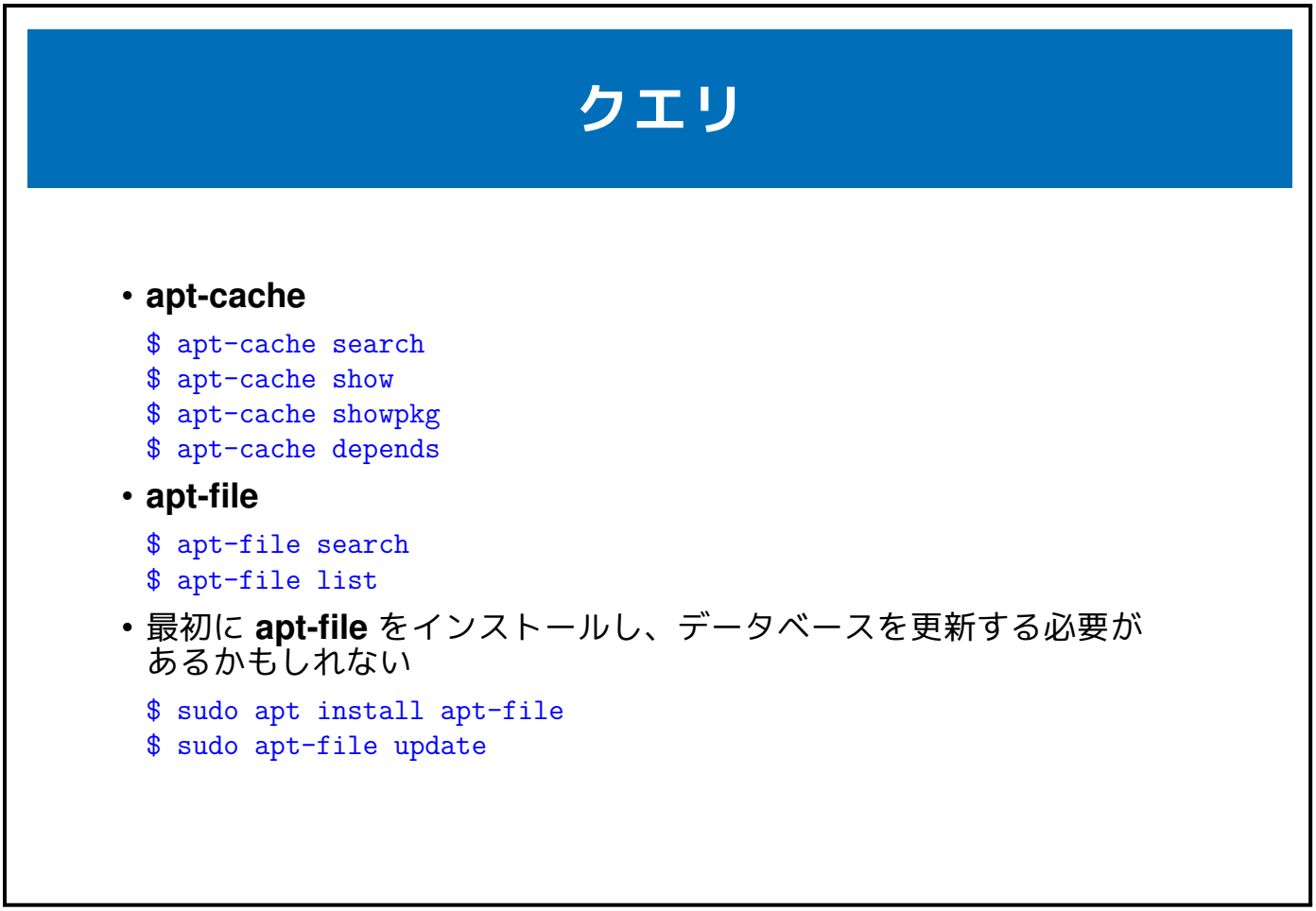
これらのユーティリティはパッケージ管理用の主要な **APT** コマンドラインツール郡です。個々のパッケージやシステム全体のインストール、管理、アップグレードに使用できます。ディストリビューションを完全に新しいリリースにアップグレードすることもできますが、これは難しい作業かもしれません。

apt が **rpm** ファイルを操作できるようにする（不完全な）拡張機能もあります。

dnf や **zypper** と同様に、**APT** は複数のリモートリポジトリを扱うことができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

9.3 クエリ




```
$ apt-cache search apache2
```

apache2 という名前のパッケージをリポジトリから検索します。

```
$ apt-cache show apache2
```

apache2 パッケージに関する基本的な情報を表示します。

```
$ apt-cache showpkg apache2
```

apache2 パッケージに関する詳細な情報を表示します。

```
$ apt-cache depends apache2
```

apache2 に依存関係のあるパッケージの完全なリストを表示します。

```
$ apt-file search apache2.conf
```

`apache2.conf` という名前のパッケージが含まれるファイルを検索します。

```
$ apt-file list apache2
```

apache2 パッケージに含まれる全てのファイルをリストします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

9.4 パッケージのインストール／削除／アップグレード

パッケージのインストール／削除／アップグレード

- パッケージのインストールと削除:

```
$ sudo apt install glibc
$ sudo apt remove glibc
$ sudo apt --purge remove glibc
```

- リポジトリ データベースの **アップデート** を行ってから、システムまたはパッケージを **アップグレード** する

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt dist-upgrade
```

個別パッケージ単位の更新は行わない

dnf や **zypper** とは異なり、個別パッケージ単位の更新／アップグレードは行いません。パッケージ全体がひとつのユニットになります。

```
$ sudo apt install [package]
```

新しいパッケージをインストールします。または、既にインストールされているパッケージを更新します。

```
$ sudo apt remove [package]
```

構成ファイルを削除せずに、パッケージをシステムから削除します。

```
$ sudo apt --purge remove [package]
```

システムから、パッケージとその構成ファイルを削除します。

```
$ sudo apt update
```

パッケージインデックスファイルをリポジトリソースと同期させます。利用可能なパッケージのインデックスを `/etc/apt/sources.list` で指定された（ひとつまたは複数の）場所から取得します。

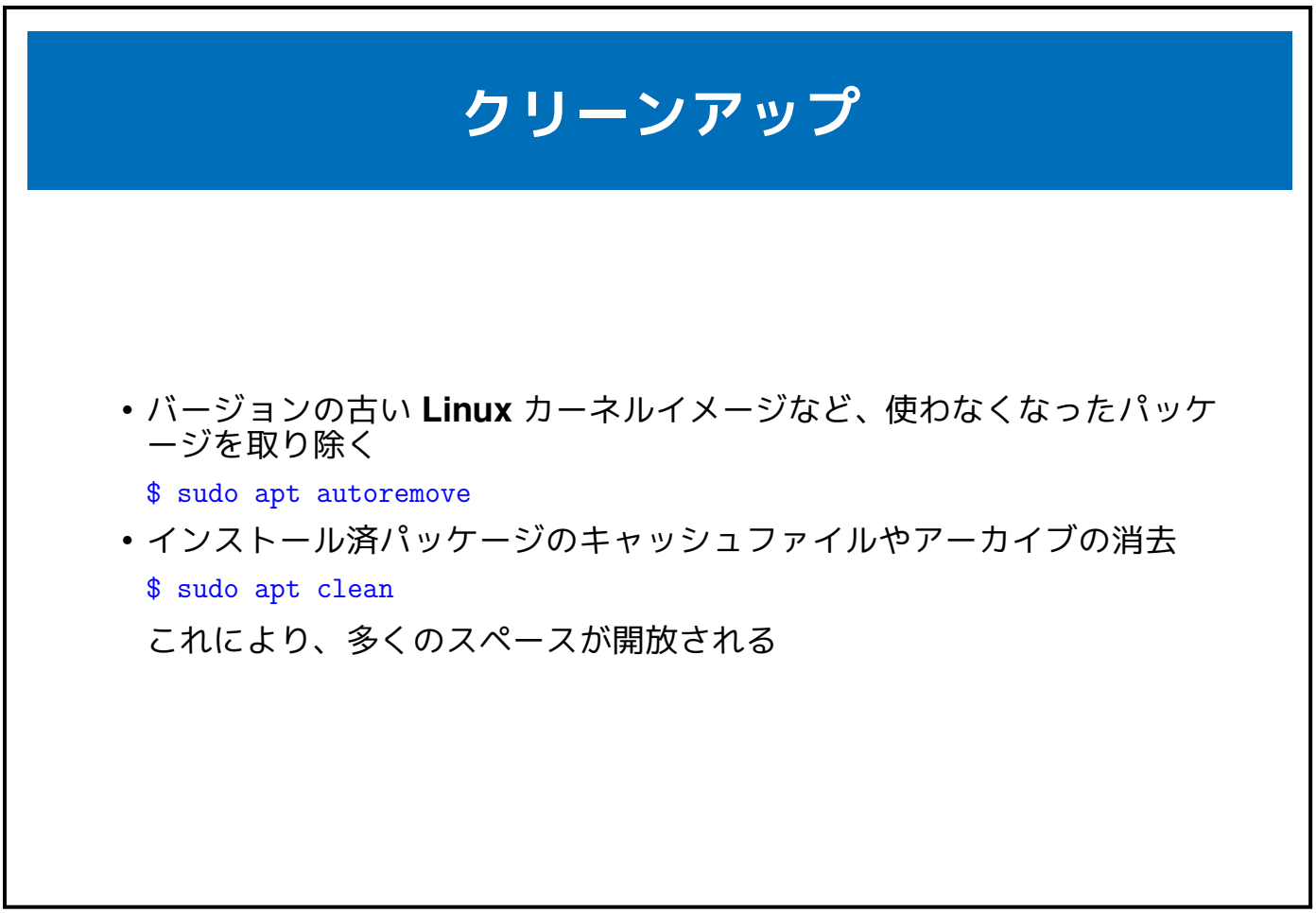
```
$ sudo apt upgrade
$ sudo apt dist-upgrade
```

`upgrade` は既にインストールされているパッケージに、利用可能なすべての更新を適用します。よく誤解されるようですが `dist-upgrade` はディストリビューションを新しいバージョンに更新するわけではありません。

dnf や **zypper** とは違って **upgrade** を実行する前には **update** を実行する必要があります。これらのユーティリティに慣れているには混乱する点かもしれません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

9.5 クリーンアップ



ディスク スペースが足りない時に **apt clean** と **apt autoremove** を使えば `/var` のディスクスペースを素早く非常に安全に開放することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

9.6 演習

デモ教材ビデオ

using_package_gui_ubuntu_demo.mp4

課題 9.0: apt が必要です

Debian、Ubuntu、Linux Mint 環境を利用

本演習を実行するためには **Debian**、**Ubuntu**、**Linux Mint** など **Debian** ベースのシステムへのアクセスが必要です。

課題 9.1: APT の基本コマンド

1. アップデートの有無を確認する。
2. インストール済のカーネル関係のパッケージをリストする、インストール済もしくはインストール可能なパッケージをリストする。
3. **apache2-dev** パッケージやその他の未インストール パッケージをインストールする場合には以下のようにします。

   ```
   $ apt-cache pkgnames
   ```

   本コマンドは全リストを表示します。ワイルドカードを使うことで、リストを短くすることも可能です。

解 9.1

1. パッケージインデックスファイルをリモートのレポジトリと同期させます。

   ```
   $ sudo apt update
   ```

   実際にアップグレードします。

   ```
   $ sudo apt upgrade
   $ sudo apt -u upgrade
   ```

   （以前に説明したように、dist-upgrade を用いることもできます）最初の形式はインストールだけ実行します。

2. ```
   $ apt-cache search "kernel"
   $ apt-cache search -n "kernel"
   $ apt-cache pkgnames "kernel"
   ```

   2 番目と 3 番目のコマンドは、名称に kernel を持つものを表示します。

   ```
   $ dpkg --get-selections "*kernel*"
   ```

   本コマンドは、インストールされたパッケージだけを表示します。**Debian** ベースのシステムでは、カーネル関係のパッケージを表示するために kernel ではなく linux を用います。なぜなら、パッケージの名称が kernel を含まないことが多々あるからです。

3. ```
   $ sudo apt install apache2-dev
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 9.2: apt コマンドを使ってパッケージの情報を調査する

apt-cache と **apt** を使用 (**dpkg** は使いません) して以下の情報を調べます。

1. 名称や説明中に **bash** を参照する全てのパッケージ。
2. インストール済またはインストール可能な **bash** パッケージ。
3. **bash** に関するパッケージ情報。
4. **bash** パッケージの依存情報。

`root` と一般のユーザーの両方で上記コマンドを実行し、その違いを確認してください。

解 9.2

1. `$ apt-cache search bash`
2. `$ apt-cache search -n bash`
3. `$ apt-cache show bash`
4. `$ apt-cache depends bash`
   `$ apt-cache rdepends bash`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 9.3: APT で、パッケージグループを管理する

metapackages を利用することで **APT** は **dnf** が提供するものと同様のパッケージグループの管理機能を提供します。グループとして同時にインストール、削除する必要がある関連する **virtual packages** と考えることができます。

利用可能な **metapackages** のリストを表示します。

```
$ apt-cache search metapackage
```

```
bacula - network backup service - metapackage
bacula-client - network backup service - client metapackage
bacula-server - network backup service - server metapackage
cloud-utils - metapackage for installation of upstream cloud-utils source
compiz - OpenGL window and compositing manager
emacs - GNU Emacs editor (metapackage)
....
```

以下のようにして、１つのパッケージをインストールすると同様にパッケージ群をインストールできます。

```
$ sudo apt install bacula-client
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bacula-common bacula-console bacula-fd bacula-traymonitor
Suggested packages:
  bacula-doc kde gnome-desktop-environment
The following NEW packages will be installed:
  bacula-client bacula-common bacula-console bacula-fd bacula-traymonitor
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 742 kB of archives.
After this operation, 1,965 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

未インストールのメタパッケージを選択して、削除します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 10 章

RPM

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- RPM システムがどう構成され、**rpm** プログラムがどの主要オペレーションで利用可能かを理解する。
- バイナリー ファイルとソース **rpm** ファイルの両方で使用される命名規則を説明する。
- パッケージの照会、検証、インストール、アンインストール、アップグレード、フレッシュアップが出来る。
- 新しいカーネルをアップグレードするのではなく、インストールすべき理由を把握する。
- **rpm2cpio** でパッケージされたファイルを **cpio** アーカイブにコピーし、ファイルをインストールせずに解凍する。

10.1 RPM (Red Hat Package Manager)

RPM を使用する利点

- ソフトウェアパッケージの管理
  - 特定のタスクやサブシステムに関連したファイルを集約したもの
  - プログラム、データ、ドキュメント、構成情報が含まれる
  - 依存関係に関する情報が含まれることもある
  - パッケージの検索、取得は行わない
- 管理者に対する利点
  - 容易にパッケージをインストール、アンインストール（消去）できる
  - パッケージが正常にインストールされたか容易に検証できる
  - パッケージのインストールに **FTP** や **HTTP** が利用できる
- 開発者に対する利点
  - オリジナルの’純正’ソースを利用できる
  - 異なるアーキテクチャ向けの **RPM** を生成できる

RPM は **Red Hat** が開発したパッケージ管理ユーティリティです。(名前は元々は **R**edhat **P**ackage **M**anager を表していました) 特定のタスクまたはサブシステムに関連する全てのファイルは、1 つのファイルにパッケージ化されます。このファイルには、ファイルをインストールおよびアンインストールする方法とその場所に関する情報も含まれています。開発者がプログラムの新しいバージョンを作成すると、通常新しい **RPM** パッケージがリリースされます。これらのファイルは、他の **Linux** ディストリビューションでは使用できない可能性があることに注意してください。

RPM によってシステム管理者はソフトウェアパッケージの管理が容易になります。特定のファイルがどのパッケージのものであるか、どのバージョンのパッケージがインストールされているか、それが正しくインストールされたかどうかを簡単に判断することができます。またパッケージを全て削除してディスク領域を解放することも簡単です。**RPM** はドキュメントファイルをパッケージの他の部分と区別し、システムにドキュメントをインストールするかどうか選択できるようにしています。

RPM によりソフトウェア開発者の仕事が楽になります。多くの場合、開発者は別のオペレーティング システム上で正しくコンパイルして実行できるようにコードを変更する必要があります。しかし **RPM** を使用すると、構築する人は **Linux** での構築に必要な変更を元のソースとは別に持つことができます。これにより、構築関係の変更が全て 1 か所に集約され新しいバージョンのコードの組み込みが容易になります。また、さまざまなアーキテクチャ向けの **Linux** バージョン構築も容易になります。

ネットワーク経由のインストレーション

特定の URL が明示的に示されない限り **rpm** はネットワークからパッケージを取得しません。絶対または相対パスを使ってローカルマシンからインストールします。

しかし **HTTP** や、非推奨となっている **FTP** プロトコルを使って、離れた場所から直接インストールすることも可能です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.2 パッケージ ファイル名

パッケージ ファイル名

- **RPM** の標準バイナリ パッケージの命名形式

```
<name>-<version>-<release>.<distro>.<architecture>.rpm
sed-4.5-1.el8.x86_64
```

- **RPM** の標準ソース パッケージの命名形式

```
<name>-<version>-<release>.<distro>.src.rpm
sed-4.5-1.el8.src.rpm
```

RPM 標準 (http://www.rpm.org/) に記載の通り **RPM** パッケージファイル名のフィールドには、特別な意味を持たせています。

RPM で動作する **yum**、**zypper** のときに詳しく説明しますが、特定のインストールでは、複数の異なるパッケージリポジトリを使用する可能性があるため、`distro` フィールドはパッケージの作成元のリポジトリを指定することに留意してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.3 RPM データベースと補助プログラム

/var/lib/rpm は、**Berkeley DB** ハッシュファイル形式で **RPM** データベースファイルを保持する、デフォルトのシステムディレクトリです。データベースファイルは **rpm** プログラムだけで更新されるので、手作業で変更するべきではありません。

代替データベースディレクトリは、**rpm** プログラムの --dbpath オプションで指定できます。たとえば、別のシステムからコピーされた **RPM** データベースを調べるためにこれを使うことができます。

--rebuilddb オプションを使用すると、インストールされたパッケージヘッダーからデータベースインデックスを再構築できます。これはどちらかというと修復目的のものであり、ゼロからの再構築に利用することを意図したものではありません。

RPM の補助プログラムとスクリプトは /usr/lib/rpm にあります。数が多いです。**RHEL 8** システムの場合：

```
$ ls /usr/lib/rpm | wc -l
```

```
74
```

ここで使っている **wc** は、出力の行数を表示するものです。

rpmrc ファイルを編集して **rpm** の検索場所の指定などが可能です。デフォルトでは **rpm** は以下の順番でファイルを探します。

1. /usr/lib/rpm/rpmrc
2. /etc/rpmrc
3. ~/.rpmrc

これらのファイルが全て読まれることに注意してください。**rpm** はパッケージを検出してもすぐには停止しません。

--rcfile オプションを使用して他の rpmrc ファイルを指定することもできます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.4 クエリ

クエリ

- -q オプションを使うと全ての **rpm** を照会できる
- -q オプションは他の多くの照会オプションと組み合わせることが可能
  - -f: ファイルがどのパッケージからインストールされたかを表示
  - -l: 特定パッケージのコンテンツを表示
  - -a: システムにインストールされた全てのパッケージをリスト
  - -i: パッケージに関する情報を表示
  - -p: パッケージデータベースではなくパッケージファイルからパッケージ情報を表示

Table 10.1: **rpm 照会コマンドの例**

どのバージョンのパッケージがインストールされているか？	`rpm -q bash`
ファイルがどのパッケージから来たものか？	`rpm -qf /bin/bash`
パッケージによってどのファイルがインストールされたか？	`rpm -ql bash`
パッケージに関する情報を表示	`rpm -qi bash`
パッケージデータベースではなくパッケージファイルからパッケージ情報を表示	`rpm -qip foo-1.0.0.1.noarch.rpm`
システムにインストールされている全てのパッケージをリスト	`rpm -qa`

その他の便利なオプションとして `--requires` と `--whatprovides` があります。

`--requires` オプションはパッケージの前提条件のリストを返します。`--whatprovides` オプションは、インストールされたどのパッケージが特定の必須パッケージを提供しているかどうかを表示します。

```
$ rpm -q  --requiresbash
$ rpm -qp --requires foo-1.0.0-1.noarch.rpm
$ rpm -q  --whatprovides libc.so.6
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.5 パッケージの検証

パッケージの検証

- パッケージの検証には rpm -V コマンドを利用する
- 全てが正常な場合は、何も出力されない
  - S: ファイルサイズが異なる
  - M: ファイルのパーミッションや種類が異なる
  - 5: MD5 チェックサムが異なる
  - D: デバイスのメジャー/マイナー番号の不一致
  - L: シンボリックリンクパスの不一致
  - U: ユーザーの所有権が異なる
  - G: グループ所有権が異なる
  - T: 修正時刻が異なる
- システム上の全てのパッケージを確認するには

```
$ sudo rpm -Va
```

rpm の -V オプションを使用して、特定パッケージのファイルがシステムの RPM データベースと一致しているかを確認できます。rpm -Va を使ってシステム上の全てのパッケージを確認することができます。

問題がある場合にだけ出力が表示されます。出力では各文字は、ファイルの属性をデータベースに記録された属性と比較した結果を示しています。単体の「.」（ピリオド）はテストを通過したことを、単体の「?」（疑問符）はテストが実行できなかったことを意味しています。（たとえば、ファイルのパーミッションが読み取り禁止だった場合など）それ以外の（覚えやすいように太字にされた）文字は対応する --verify テストの失敗を意味しています。

全てが正常な場合、何も出力されません。

```
$ rpm -V bash
```

ファイルのサイズ、チェックサム、および修正時刻が変更されたことを出力する例です。

```
$ rpm -V logrotate
```

```
S.5....T.c/etc/logrotate.conf
```

ファイルが欠落していることを出力する例です。

```
$sudo mv /sbin/logrotate /sbin/logrotate_KEEP
$rpm -V logrotate
```

```
S.5....T. c /etc/logrotate.conf2
missing    /usr/sbin/logrotate
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.6 パッケージのインストールと削除

パッケージのインストール

- **RPM** はパッケージのインストールに関わる色々なタスクを実行する
  - 依存関係をチェックする
  - 競合をチェックする
  - インストール前に要求されるコマンドを実行する
  - 構成ファイルの知的な処理を行う
  - パッケージからファイルを解凍し、正しい属性でインストールする
  - インストール後に必要なコマンドを実行する
  - システムの **RPM** データベースを更新する
- `rpm -i` でパッケージをインストールする

```
$ sudo rpm -ivh bash-4.4.19-12.el8_0.x86_64
```

`-i` はインストール、`-v` は詳細表示、`-h` は進捗に伴ってハッシュマークを表示させるオプション指定である

一部のパッケージは、他のパッケージがインストールされていないと適切に動作しないため、依存性のチェックが必要です。

インストール済のパッケージを上書きインストールしようとした場合や、新しいバージョンの上に古いバージョンをインストールする場合には、競合が発生します。

パッケージを作成する開発者は、インストールの前後に特定のタスクを実行するように指定できます。

構成ファイルをインストールするとき、ファイルが存在し以前のバージョンのパッケージがインストールされてから変更された **RPM** はサフィックス `.rpmsave` を使って古いバージョンを保存します。これにより、古い構成ファイルに加えた変更を新しいバージョンのファイルに統合できます。この機能を使うには RPM パッケージが適切に作成されている必要があります。

RPM は適切な場所にファイルをインストールすることに加えて、パーミッション、所有権、変更（ビルド）時刻などの属性も設定します。

RPM はパッケージをインストールするたびにシステムデータベース内の情報を更新します。この情報は、競合をチェックに使用されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

パッケージのアンインストール

- 通常、アンインストールが成功した場合には何も出力されない

```
$ sudo rpm -e system-config-lvm
```

```
package system-config-lvm is not installed
```

- 依存関係に起因するエラーの例

```
$ sudo rpm -e xz
```

```
error: Failed dependencies:
    xz is needed by (installed) pcp-5.1.1-4.el8_3.x86_64
    xz is needed by (installed) sos-3.9.1-6.el8.noarch
    xz is needed by (installed) gettext-devel-0.19.8.1-17.el8.x86_64
    xz is needed by (installed) libreport-2.9.5-15.el8.x86_64
    xz is needed by (installed) dracut-049-95.git20200804.el8_3.4.x86_64
    xz is needed by (installed)
    ↪  libvirt-daemon-driver-qemu-6.0.0-28.module+el8.3.0+7827+5e65edd7.x86_64
    xz is needed by (installed) rpm-build-4.14.3-4.el8.x86_64
    xz is needed by (installed) libguestfs-1:1.40.2-25.module+el8.3.0+7421+642fe24f.x86_64
    xz is needed by (installed) sysstat-11.7.3-5.el8.x86_64
    /usr/bin/xz is needed by (installed) file-roller-3.28.1-3.el8.x86_64
```

-e オプションにより rpm はパッケージをアンインストール（消去）します。通常、アンインストールしようとしているパッケージがシステム上の他のパッケージで必要な場合、`rpm -e` はエラーメッセージを出力して失敗の形で終了します。

-e オプションとともに `--test` オプションを使用すると、実際にアンインストールを行うことなくアンインストールが成功するか失敗するかを確認できます。操作が成功すると rpm は何も出力しません。-vv オプションを追加すれば、更に詳細な情報を取得できます。

削除する時のパッケージ引数はパッケージ名です。**rpm** ファイル名ではないことに注意してください。

重要な注意事項: 決して **rpm** パッケージそのものを削除（消去／アンインストール）しないでください。もし削除してしまった場合、オペレーティング システムの再インストールか、レスキュー環境の起動以外に問題を解決する手段はありません。

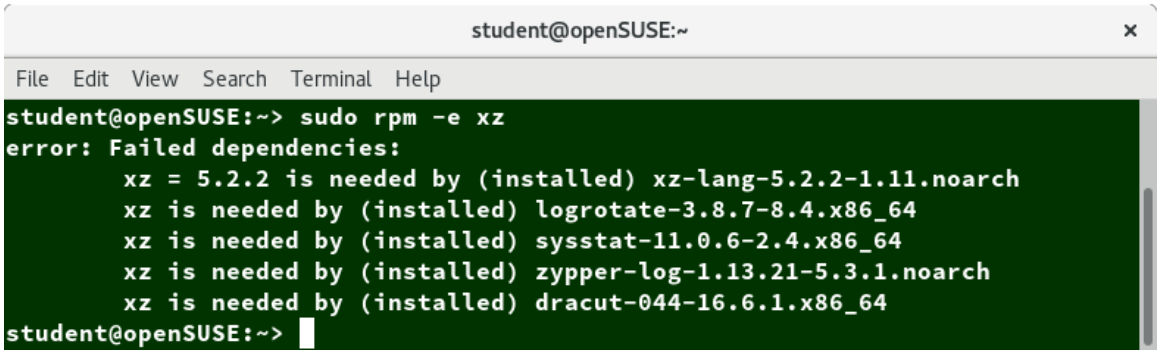


図 10.1: **RPM パッケージのアンインストール**

openSUSE で取得されたスクリーンショットは、スライドの **RHEL** で取得されたものと依存関係が多少異なる場合があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.7 アップデート、アップグレード、RPM パッケージの更新

パッケージのアップデート

- 元のパッケージが置き換えられる（インストールされている場合）

```
$ rpm -Uvh bash-4.4.19-10.el8.x86_64.rpm
```

- 現在インストールされていなければパッケージをインストール
- 同一ファイルを含む 2 つのパッケージのインストールは出来ない
- `--oldpackage` オプションを使ってダウングレードが可能

アップグレードする場合、新しいバージョンがインストールされると、既にインストールされているパッケージは削除されます。唯一の例外は元のインストールの構成ファイルで、拡張子が `.rpmsave` に変更されて保存されます。

パッケージがまだインストールされておらず、`-U` オプションが指定された場合には、単純にインストールされ、エラーは出ません。

`-i` オプションはアップグレード用として設計されていません。古い **RPM** パッケージの上に新しい **RPM** パッケージをインストールしようとすると、既存システムファイルの上書きというエラーメッセージを表示して失敗します。

ただし、パッケージの各バージョンに同じファイルが含まれていない場合、同じパッケージの異なるバージョンがインストールできる場合があります。カーネルパッケージとライブラリパッケージは、代替アーキテクチャから複数回インストールされる唯一のパッケージです。

rpm -U でダウングレードする（つまり現在のバージョンを以前のバージョンに置き換える）場合は、コマンドラインに `--oldpackage` オプションを追加する必要があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

パッケージの更新

- カレントディレクトリの全てのパッケージを **更新** するには

```
$ sudo rpm -Fvh *.rpm
```

  - 古いバージョンのパッケージがインストールされている場合、ディレクトリ内の新しいバージョンにアップグレード
  - システム上のバージョンがディレクトリ内のバージョンと同じ場合、何もしない
  - パッケージのバージョンがインストールされていない場合、ディレクトリ内のパッケージは無視される
- 更新（-F、新しいものに代えるという意味の Freshen の頭文字である）は多数のパッチ（すなわちアップグレードされたパッケージ）を一度に適用するのに便利である

更新とアップグレードは似ていますが、ひとつ違いがあります。アップグレードの際にパッケージが存在しない場合はインストールされます。更新の際にパッケージが存在しない場合には単に無視されます。アップグレードの際にも更新の際にも、パッケージが既にロードされている場合は新しいパッケージがインストールされます。

-F オプションは、新しいパッケージのインストールをしないで、既にインストールされているパッケージだけを更新するのに便利です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.8 Linux カーネルのアップグレード

Linux カーネルのアップグレード

- 新しいカーネルは常にインストールする（でもアップグレード［-U オプションによる入れ替え］ではない）
  Red Hat ベースのシステムでは

```
$ sudo rpm -ivh kernel-{version}.{arch}.rpm
```

- 問題があったら古いカーネルに戻せる
- 新カーネルのテスト終了後には古いカーネルを削除できるが、動作確認済の古いカーネルを１つ以上残して置くことが推奨される

旧バージョンのパッケージの削除

もし本当に古いバージョンのカーネルやその他のパッケージを削除したいのであれば、**Red Hat** ベースのシステムではそれが可能です

```
$ sudo dnf remove --oldinstallonly
```

このコマンドは注意して使用してください。

システムに新しいカーネルをインストールした場合、それを有効にするためには再起動が必要です。（再起動が必要な数少ないアップデートの１つです）カーネルのアップグレード (`-U`) を実行しないでください。アップグレードを行うと現在実行中の古いカーネルが削除されます。

アップグレードしてもシステムは停止しませんが、再起動後に問題が発生した場合古いカーネルがシステムから削除されているため、再起動ができなくなります。ただしインストール (`-i`) を使った場合両方のカーネルは共存しどちらを起動するかを選択できます。つまり、必要に応じて古いカーネルにも戻せます。

これを行うと、新しいバージョンのカーネルが含まれるように **GRUB** 構成ファイルが自動更新されます。他の目的でシステムを再構成していない限り、ブート時のデフォルト選択になります。

新しいカーネルバージョンをテストしたら、必要なら古いバージョンを削除できますが、これは必須ではありません。スペースが不足していない限り、１つ以上古いカーネルも使用可能にしておくことをお勧めします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.9 rpm2archive と rpm2cpio

rpm2archive と rpm2cpio の利用

- RPM パッケージファイルをアーカイブに変換するには

```
$ rpm2archive bash-XXXX.rpm
```

bash-XXXX.rpm.tgz が生成される

- １ステップで展開するには

```
$ cat bash-XXXX.rpm | rpm2archive - | tar -xvz
```

- RPM パッケージファイルを cpio にアーカイブに変換するには

```
$ rpm2cpio bash-XXXX.rpm > bash.cpio
```

- 一つ以上のファイルの展開するには

```
$ rpm2cpio bash-XXXX.rpm | cpio -ivd bin/bash
$ rpm2cpio logrotate-XXXX.rpm | cpio --extract --make-directories
```

- パッケージ内のファイルをリストするには

```
$ rpm -qilp bash-XXXX.rpm
```

rpm2archive を使って RPM パッケージファイルを **tar** アーカイブに変換できます。コマンドに - が与えられた場合には入力と出力はそれぞれ stdin と stdout になります。

rpm2archive は、以前のパッケージファイルを **cpio** アーカイブに変換する、またはパッケージファイルからファイルを解凍する **rpm2cpio** よりも直接的で効率的です。これを行うと現在のディレクトリの相対パスに展開されます。もし /home/student ディレクトリにいるユーザーが bin/bash ファイルを展開した場合、上の例のようにファイルが /home/student/bin/bash に展開されます。

パッケージ内のファイルを一覧表示したいだけなら、一番簡単な方法は **rpm** コマンドを使うことです。

```
$ rpm -qilp package.rpm
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

10.10 演習

デモ教材ビデオ

using_rpm_demo.mp4

課題 10.0: RPM が必要です

RHEL、CentOS、Fedora、SUSE、openSUSE

本演習を実行するためには、**RHEL**、**CentOS**、**Fedora**、**SUSE** や **openSUSE** などの **RPM** ベースのシステムにアクセスできる必要があります。

課題 10.1: RPM を使う

rpm パッケージを扱い、調べるためのいくつかの簡単な操作を行います。

本演習は **Red Hat** と **SUSE** ベースのシステム両方で実施できます。

1. /etc/logrotate.conf というファイルを有するパッケージを検索する。
2. パッケージが含む全てのファイルとパッケージ自身の情報を表示する。
3. パッケージのインストール状況を確認する。
4. パッケージを削除する。

解 10.1

1. `$ rpm -qf /etc/logrotate.conf`

   ```
   logrotate-3.14.0-3.el8.x86_64
   ```

2. `$ rpm -qil logrotate`

   ```
   ...
   ```

   この 2 つのステップを一度にやる方法もあります。

   ```
   $ rpm -qil $(rpm -qf /etc/logrotate.conf)
   ```

3. `$ rpm -V logrotate`

   ```
   ..?......    /etc/cron.daily/logrotate
   S.5....T.  c /etc/logrotate.conf
   ```

4. `$ sudo  rpm -e logrotate`

On Red Hat RHEL 8

```
$ sudo rpm -e logrotate
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
error: Failed dependencies:
   logrotate is needed by (installed) vsftpd-3.0.3-28.el8.x86_64
   logrotate >= 3.5.2 is needed by (installed) rsyslog-8.37.0-13.el8.x86_64
```

On openSUSE openSUSE-Leap 15.1

```
$ sudo  rpm -e logrotate
```

```
error: Failed dependencies:
   logrotate is needed by (installed) xdm-1.1.11-lp151.13.2.x86_64
   logrotate is needed by (installed) wpa_supplicant-2.6-lp151.4.4.x86_64
   logrotate is needed by (installed) chrony-3.2-lp151.8.6.x86_64
   logrotate is needed by (installed) net-snmp-5.7.3-lp151.7.5.x86_64
   logrotate is needed by (installed) syslog-service-2.0-lp151.3.3.noarch
   logrotate is needed by (installed) vsftpd-3.0.3-lp151.6.3.x86_64
   logrotate is needed by (installed) libvirt-daemon-5.1.0-lp151.7.6.1.x86_64
   logrotate is needed by (installed) iscsiuio-0.7.8.2-lp151.13.6.1.x86_64
   logrotate is needed by (installed) mcelog-1.60-lp151.2.3.1.x86_64
```

正確なパッケージの依存関係は、ディストリビューションやインストール済のソフトウェアに関係していることに注意してください。

課題 10.2: RPM データベースの再構築

/var/lib/rpm に格納されている **RPM** データベースが破壊されてしまう場合があります。本演習では、新しいデータベースを構築しその整合性を確認します。

本演習は **Red Hat** と **SUSE** ベースのシステム両方で実施できます。

1. 再構築の過程で上書きするので、/var/lib/rpm の内容をバックアップします。バックアップをしていない状況で何か問題が起きたら、深刻な状況に陥ります。
2. データベースを再構築します。
3. 新しいディレクトリの内容とバックアップした内容を比較します。バイナリデータなので実際のファイルの中味を比較しないでください。代わりにファイルの数と名前を比較します。
4. システムにインストールされている全ての **rpm** のリストを表示します。再構築の手順を実行する前にリストした内容と比較します。正しくクエリコマンドが動いていれば、新しいデータベースファイルができたはずです。
5. 両方のディレクトリ内容を比較します。同じファイルを含んでいますか？
6. バックアップを削除しても構いません。（おそらく 100MB 程度でしょう）しかし、破棄する前にシステムがきちっと動作するかをしばらく監視したほうが良いと思います。

解 10.2

1.
```
$ cd /var/lib
$ sudo cp -a rpm rpm_BACKUP
```

2.
```
$ sudo rpm --rebuilddb
```

3.
```
$ ls -l rpm rpm_BACKUP
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4. `$ rpm -qa | tee /tmp/rpm-qa.output`

5. `$ ls -l rpm rpm_BACKUP`

6. システムに問題が無いことを確認してから、実行してください！

   `$ sudo rm -rf rpm_BACKUP`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 11 章

dnf と yum

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- パッケージインストーラーとその特徴について説明する。
- **dnf** が高レベルのパッケージ管理システムとしてどのように機能するかを説明する。
- **dnf** がリポジトリを使用するように設定する。
- **dnf** が使用するクエリについて説明する。
- **dnf** を使用してパッケージの検証、インストール、削除、アップグレードを行う。
- 追加コマンドと新しいリポジトリのインストール方法について説明します。

11.1 dnf

dnf とは何か？

- RPM のフロントエンドである
- リモートリポジトリからパッケージを取得する
  - 複数の別々のリポジトリを利用可能である
  - リポジトリ構成ファイルは /etc/yum.repos.d/ である
- 依存関係の解決に対応している
- **RHEL**、**CentOS**、**Fedora** などが採用している

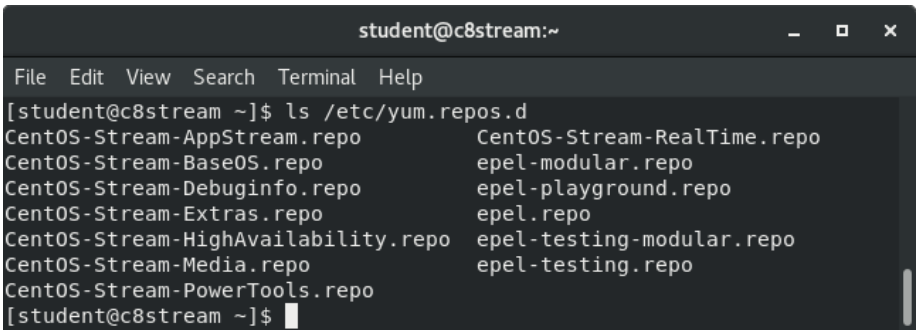


図 11.1: **rpm リポジトリ**

dnf にはパッケージマネージメントに役立つ機能が多く含まれます。**RPM** のフロントエンドであるだけでなく、一つないし複数のリモートリポジトリからパッケージを取得する機能も有しています。特に優れた機能の一つは、依存関係解決ができることです。

リポジトリの構成ファイルは /etc/yum.repos.d に置かれており、.repo という拡張子を持っています。

repo ファイルの基本は以下のようなものです。

Repo の例

```
[repo-name]
    name=Description of the repository
    baseurl=http://somesystem.com/path/to/repo
    enabled=1
    gpgcheck=1
```

dnf は、処理性能を高めるためにキャッシュ情報を持ちます。以下のコマンドで一部、ないし全部のキャッシュを削除できます。

```
$ dnf clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]
```

enabled の値を変更するか、--disablerepo somerepo と --enablerepo somerepo を使うと、特定の repo を使うか使わないかを切り替えることができます。

dnf の操作には、ディストリビューションが提供するグラフィカルインターフェイスを使わず、コマンドラインだけを利用します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

11.2 yum

yum

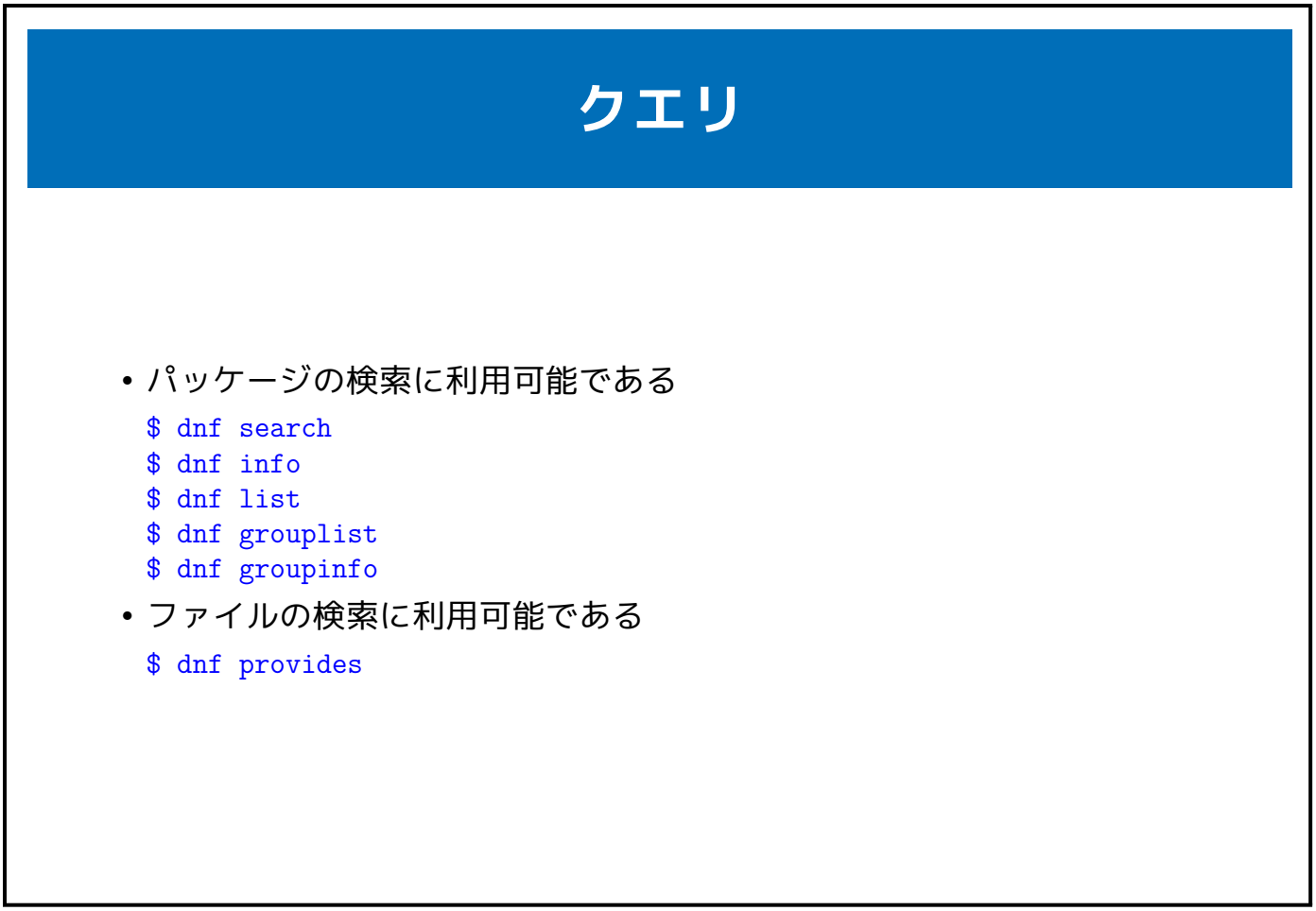
- **RHEL/CentOS 7** が **8** に移行するタイミングで **yum** を **dnf** に変更した
- **Fedora** は、それ以前から採用していた
- **dnf** には（**yum** との）後方互換性がある
  - ほとんど全ての **yum** コマンドは継続して動作する
- 以下 URL に、参照可能な詳細ドキュメントがある
  https://docs.fedoraproject.org/en-US/quick-docs/dnf/
  https://dnf.readthedocs.io/en/latest/

dnf のバージョンによっては、使おうとした **yum** コマンドが廃止予定と警告され、正しいコマンドが示されるかもしれません。

あなたが既に **yum** を使い込んでいるなら、日々の業務で利用する大部分の **yum** のコマンドのサブセットは **dnf** でも動作するので、**dnf** を少しづつ覚えていくことができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

11.3 クエリ




```
$ dnf search keyword
```

パッケージをキーワードで検索することが可能です。

```
$ dnf info package-name
```

パッケージ情報を表示します。

```
$ dnf list [installed | updates | available]
```

インストール済み、利用可能、更新済みのパッケージをリストします。

```
$ dnf grouplist
```

インストール済み、利用可能、更新済みのパッケージグループをリストします。

```
$ dnf groupinfo packagegroup
```

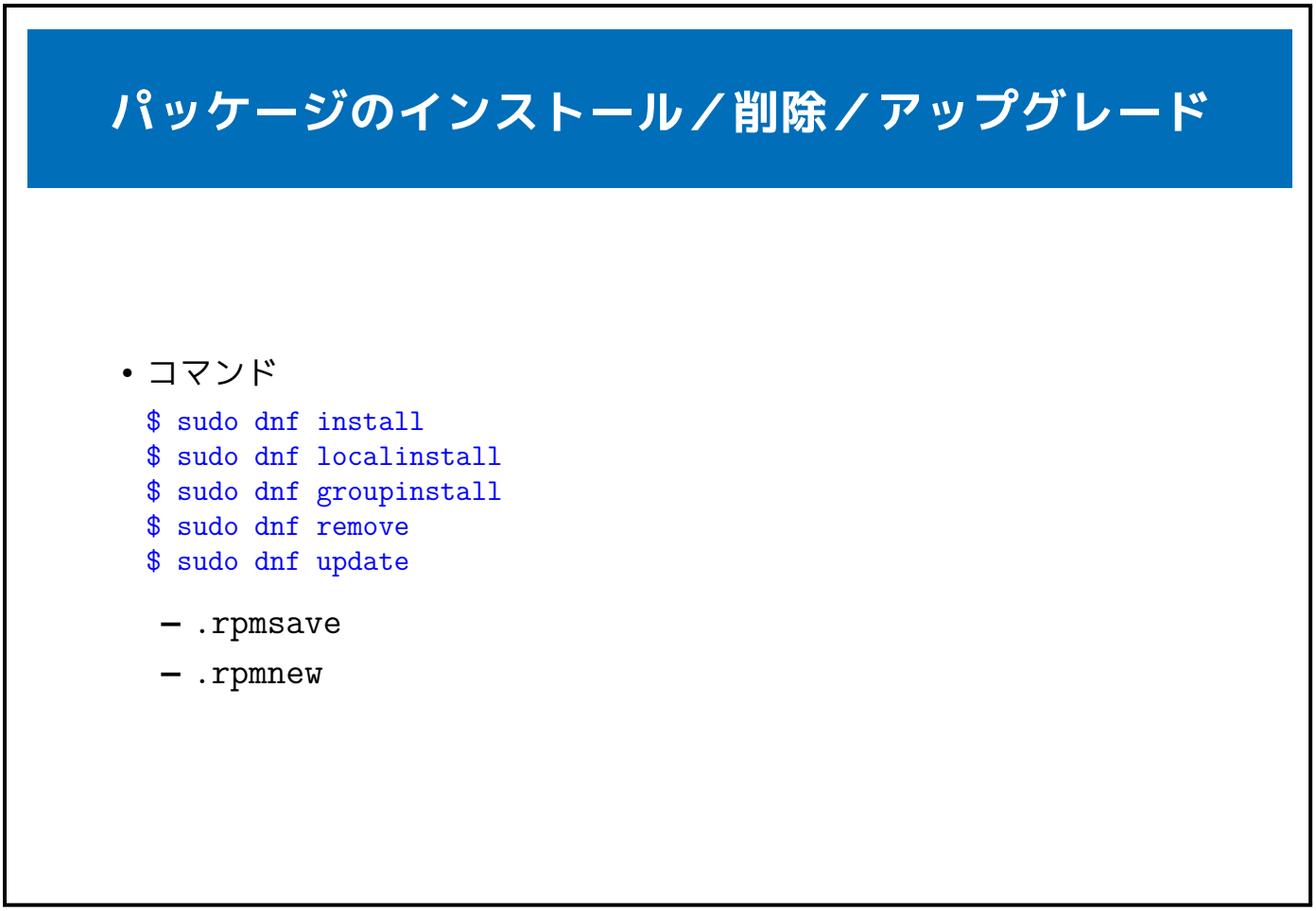
パッケージグループ情報を表示します。

```
$ dnf provides /path/to/file
```

ファイル名からパッケージのオーナーを表示します。(紛らわしいのは、ファイル名に少なくとも一つの / が必要なことです。)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

11.4 パッケージのインストール／削除／アップグレード




```
$ sudo dnf install package
```

リポジトリから、パッケージをインストールします。同時に、インストール依存関係も解決します。

```
$ sudo dnf localinstall package-file
```

ローカルな rpm ファイルからパッケージをインストールします。

```
$ sudo dnf groupinstall 'group-name'
```

リポジトリから、特定ソフトウェアグループをインストールします。同時に、グループ内の各パッケージのインストール依存関係を解決します。

```
$ sudo dnf remove package
```

システムからパッケージを削除します。

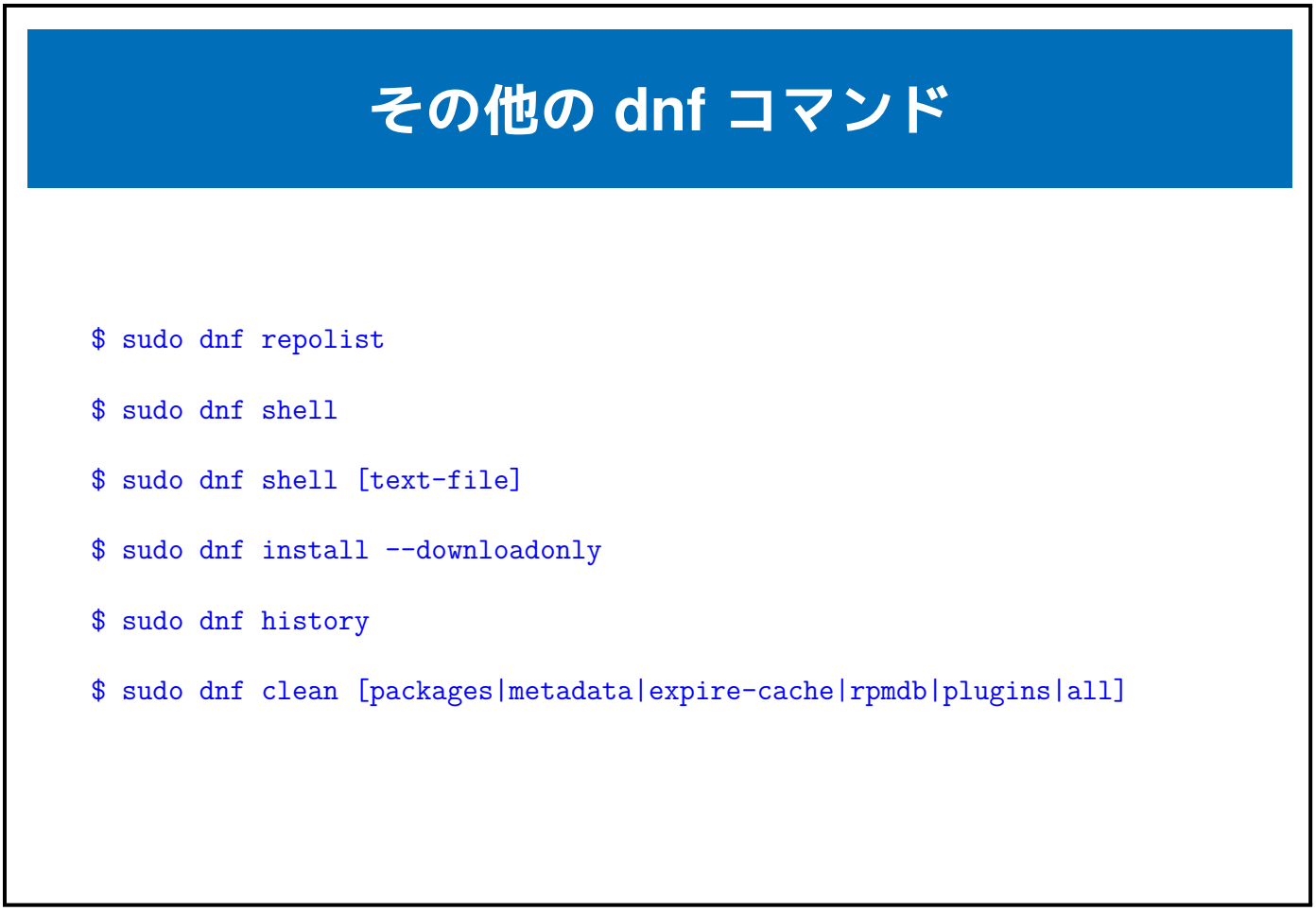
```
$ sudo dnf update [package]
```

リポジトリからパッケージを更新します。パッケージ名が指定されていない場合は、全てのパッケージを更新します。

インストール（または更新）中にパッケージに更新された構成ファイルを見つけると、古い構成ファイルの名前を `.rpmsave` 拡張子に変更します。古い構成ファイルを新しいソフトウェアで引き続き使用する場合は、新しい構成ファイルの名前に `.rpmnew` 拡張子を付けます。これらのファイル名拡張子を検索して、何らかの調整を行う必要があるかを調査できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

11.5 その他の dnf コマンド




```
$ sudo dnf list "dnf-plugin*"
```

追加的な **dnf** プラグインをリストします。

```
$ sudo dnf repolist
```

利用可能なリポジトリを表示します。

```
$ sudo dnf shell
$ sudo dnf shell file.txt
```

複数の dnf コマンドを利用できる対話型シェルを提供します。２番目では `file.txt` からコマンドを実行します。

```
$ sudo dnf install --downloadonly package
```

パッケージのダウンロードだけを行います。ファイルは `/var/cache/dnf` に保存されます。

```
$ sudo dnf history
```

dnf コマンドの履歴の閲覧、修正オプションを付加して取り消し（undo）や 再実行（redo）ができます。

```
$ sudo dnf clean [packages|metadata|expire-cache|rpmdb|plugins|all]
```

ローカルに保存されたファイルと `/var/cache/dnf` に保存されたメタデータを削除します。陳腐化したデータを削除してスペースを空けることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

11.6 演習

課題 11.0: dnf か yum が必要

RHEL、CentOS、Fedora、SUSE、openSUSE

以下の演習を実行するには **RHEL/CentOS** や **fedora** など、**dnf** ベースのシステムにアクセスできる必要があります。**dnf** には後方互換性があるので、**RHEL/CentOS 7** では **dnf** を **yum** に置き換えて以下の演習を行ってください。

課題 11.1: dnf 基本コマンド

1. システムに最新のアップデートがあるか確認します。
2. 特定のパッケージを更新します。
3. インストール済のカーネル関係のパッケージをリストします。そして、全てのインストール済または利用可能な kernel パッケージのリストを表示します。
4. **httpd-devel** パッケージ、または他のまだインストールしていないパッケージのインストールをします。以下のように入力すると

   ```
   $ sudo dnf list
   ```

   インストール済みパッケージの完全なリストが表示されます; 引数にワイルドカードを指定することで、リストを絞ることができます。

解 11.1

1. ```
   $ sudo dnf update
   $ sudo dnf check-update
   $ sudo dnf list updates
   ```

   1 番目のコマンドだけが、インストールを試します。

2. ```
   $ sudo dnf update bash
   ```

3. ```
   $ sudo dnf list installed "kernel*"
   $ sudo dnf list "kernel*"
   ```

4. ```
   $ sudo dnf install httpd-devel
   ```

課題 11.2: パッケージの情報を探す

（直接 **rpm** を使わず）**dnf** を使い、以下の情報を取得します。

1. 名称や説明に **bash** に関する内容を含む全てのパッケージ
2. インストール済で利用可能な **bash** パッケージ
3. **bash** のパッケージ情報
4. **bash** パッケージの依存情報

これらのコマンドを、`root` と一般のユーザーの両方で試してください。違いがわかりますか？

解 11.2

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

注目

ディストリビューションのバージョンによっては、**sudo** を使わず以下のコマンドを実行すると、情報を得るだけにもかかわらずパーミッションエラーが出ます。

```
$ sudo dnf search bash
$ sudo dnf list bash
$ sudo dnf info bash
$ sudo dnf deplist bash
```

課題 11.3: パッケージ グループを管理する

注目

RHEL/CentOS 上で **sudo** を使わず以下のコマンドを実行すると、情報を得るだけにもかかわらずパーミッションエラーが出ます。

dnf は、パッケージグループ管理機能を提供しています。

1. システムで利用可能な全てのパッケージグループをリストするには、以下のようにします。

   ```
   $ dnf grouplist
   ```

2. `Virtualization Host` グループを指定して、その情報を得るには、以下のようにします。

   ```
   $ dnf groupinfo "Virtualization Host"
   ```

3. 次のようにしてインストールします。

   ```
   $ sudo dnf groupinstall "Virtualization Host"
   ```

4. システムにインストール済で不要になったパッケージグループがあります。`dnf groupremove` を使い、以下のように削除します。

   ```
   $ sudo dnf groupremove "Virtualization Host"
   ```

   削除の確認プロンプトがでます。これにより、安全にこのコマンドの動作を確かめることができます。

   `groupremove` が、インストール済のものを全ては削除するわけでは **ない** ことに注意してください; これがバグなのか、機能なのかは議論があるところです。

課題 11.4: 新しいレポジトリの追加

作者によれば（http://www.webmin.com/index.htm）：

“ **Webmin** は、UNIX の管理者向けウェブベースのインターフェイスを提供します。最新のブラウザを使って、ユーザー アカウント、Apache、DNS、ファイル共有、その他の設定が行えます。Webmin を使えば、/etc/passwd などのシステム構成ファイルを手動編集する必要がなくなります。コンソールやリモートで、システムを管理できるようになるのです。”

インストールとアップグレード用のレポジトリを作成します。ダウンロードと最新の **rpm** を入手できるページを作成します。自動的な更新機能は提供しません。

1. `/etc/yum.repos.d` 中に `webmin.repo` というレポジトリを作成します。内容は以下のとおりです。

webmin.repo の内容

```
[Webmin]
name=Webmin Distribution Neutral
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
baseurl=http://download.webmin.com/download/yum
mirrorlist=http://download.webmin.com/download/yum/mirrorlist
enabled=1
gpgcheck=0
```

2. webmin パッケージをインストールします。

```
$ sudo dnf install webmin
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 12 章

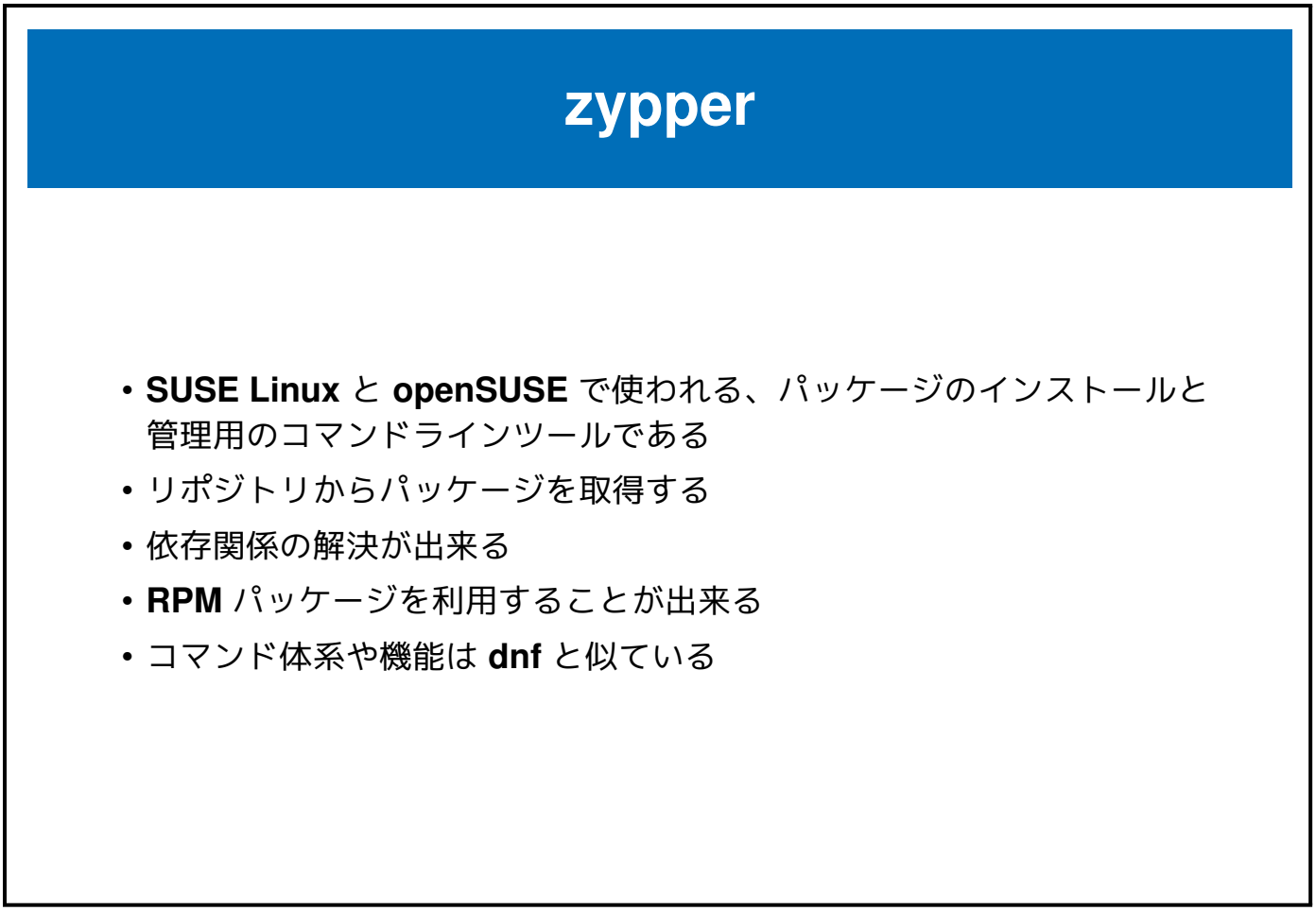
zypper

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- **zypper** とは何かを説明する。
- **zypper** が使用できるクエリについて説明する。
- **zypper** を使ってパッケージのインストール、削除、アップグレードを行う。
- 追加でより高度な **zypper** コマンドを学ぶ。

12.1 zypper

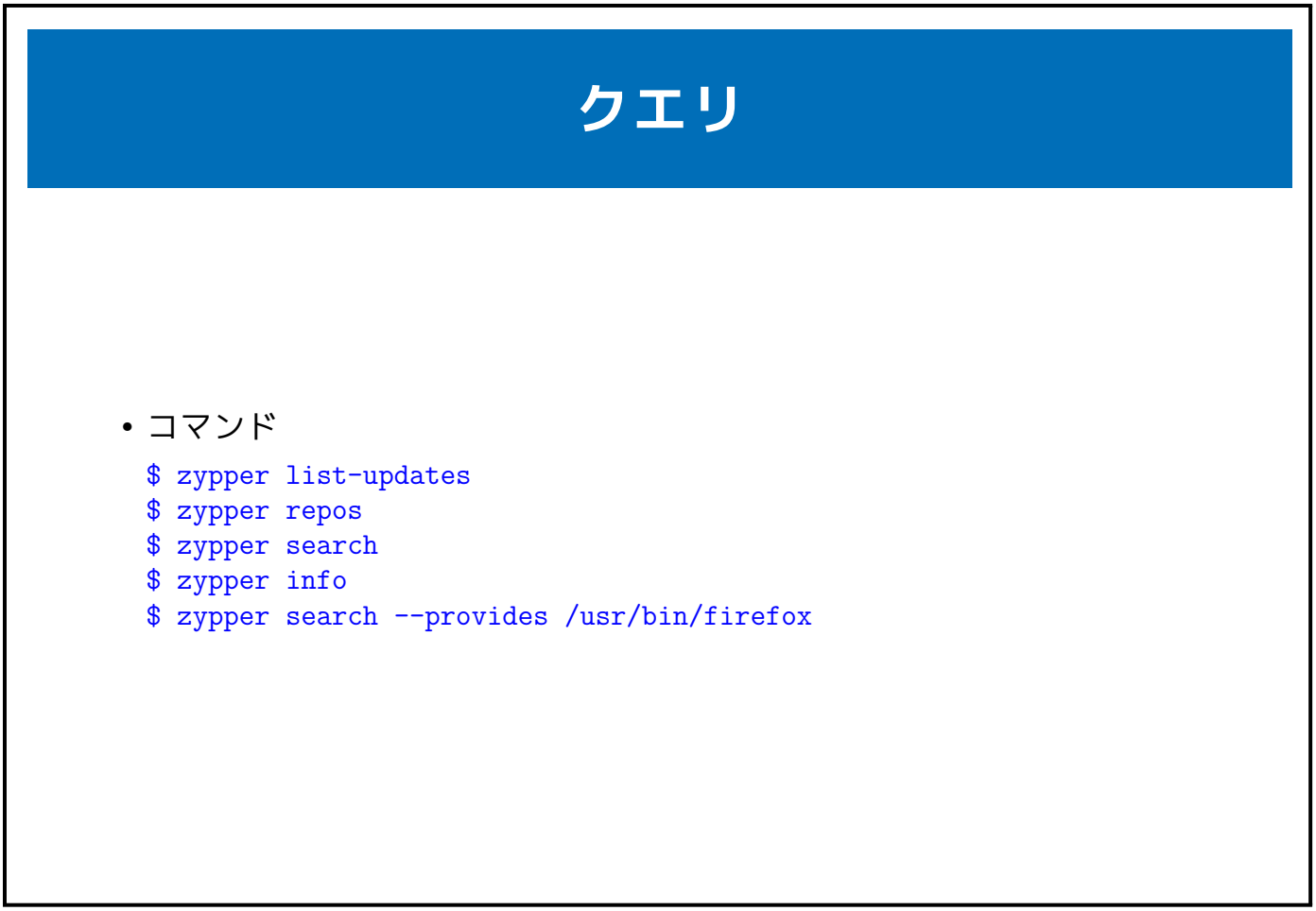


zypper は **SUSE** ベースのシステム向けに、**rpm** プログラムの下回りを利用するためのハイレベルのインテリジェントサービスを提供するもので、**Red Hat** ベースのシステムにおける **dnf**/**yum** と同様な役割を果たします。インストール時には、依存関係を自動解決する機能を持っています。外部のソフトウェア **リポジトリ** にアクセスし、必要に応じてリポジトリの同期、パッケージの取得、インストールを行います。

zypper は **SUSE Linux** と **openSUSE** でパッケージをインストールおよび管理するためのコマンドラインツールです。
機能と基本的なコマンド構文は **dnf** と非常によく似ており **rpm** パッケージも利用できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

12.2 クエリ




```
$ zypper list-updates
```

利用可能な更新をリストします。

```
$ zypper repos
```

利用可能なリポジトリをリストします。

```
$ zypper search <string>
```

文字列に該当するリポジトリを検索します。

```
$ zypper info firefox
```

パッケージの情報を表示します。

```
$ zypper search --provides /usr/bin/firefox
```

ファイルがどのパッケージによってもたらされたかを表示するために、リポジトリを検索します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

12.3 パッケージのインストール／削除／アップグレード

パッケージのインストール／削除／アップグレード

- zypper インストール

```
$ sudo zypper install firefox
```

- zypper 更新

```
$ sudo zypper --non-interactive update
$ sudo zypper update firefox
```

- zypper 削除

```
$ sudo zypper remove firefox
```

- --non-interactive (または -n) を付与すると、コマンド実行時に確認を求められない

```
$ sudo zypper install firefox
```

システムのパッケージをインストール、ないし更新します。

```
$ sudo zypper --non-interactive install firefox
```

インストールや更新時に確認を求められないので、スクリプトで使うのに便利です。

```
$ sudo zypper update
```

リポジトリから全てのパッケージを更新します。

```
$ sudo zypper --non-interactive update
```

リポジトリから全てのパッケージを更新します。実行時に確認を要求されないので、スクリプト内で使う時に便利です。

```
$ sudo zypper remove firefox
```

システムからパッケージを削除します。
dnf 同様パッケージ削除コマンドを実行すると、一緒に使われている他のパッケージも同時に削除される点に注意してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

12.4 その他の zypper コマンド

その他の zypper コマンド

- **zypper** のコマンドラインシェルを起動する

  ```
  $ sudo zypper shell
  ```

- 新しいリポジトリを追加する

  ```
  $ sudo zypper addrepo
  ```

- リポジトリを削除する

  ```
  $ sudo zypper removerepo
  ```

- /var/cache/zypp をクリーンアップして、空き領域を拡大する

  ```
  $ sudo zypper clean [--all]
  ```

いくつかの **zypper** コマンドは、続けて実行する必要があります。
コマンド発行の都度データベース全体を再読み取りしないように、次のように **シェルモード** で **zypper** を実行できます。

```
$ sudo zypper shell
> install bash
...
> exit
```

zypper は **readline** ライブラリをサポートしているため、**zypper** シェルでも **bash** シェルと同じコマンドライン編集機能を使用できます。

新しいリポジトリを追加するには

```
$ sudo zypper addrepo URI alias
```

これは指定された URI にあり、追加後は指定された 別名（エイリアス）を使用します。

リストからリポジトリを削除するには

```
$ sudo zypper removerepo alias
```

alias を使った削除 repo の指定

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

12.5 演習

デモ教材ビデオ

using_yast_demo.mp4

課題 12.0: zypper が必要です

On openSUSE

本演習を行うには **SUSE**、**opeSUSE** など **zypper** ベースのシステムへのアクセスが必要です。

課題 12.1: zypper 基本コマンド

1. 最新のアップデートを確認する。
2. 特定のパッケージを更新する。
3. 有効かどうかに関係なく、システムが認識している全てのリポジトリをリストする。
4. インストール済のカーネル関係のパッケージをリストする。そして、全てのインストール済または利用可能なパッケージをリストする。
5. **apache2-devel** パッケージ、または、まだインストールしていないパッケージをインストールする。（**httpd** は **SUSE** システムにおける **apache2** です）以下のように入力します。

   ```
   $ sudo zypper search
   ```

   これで完全なリストが表示されます； ワイルドカードを指定してリストを絞り込むことができます。

解 12.1

1. `$ zypper list-updates`
2. `$ sudo zypper update bash`
3. `$ zypper repos`
4. ```
   $ zypper search -i kernel
   $ zypper search kernel
   ```
5. `$ sudo zypper install apache2-devel`

課題 12.2: zypper を使って、パッケージの情報を得る

（直接 **rpm** を使うのではなく）**zypper** を使って、以下の情報を見つけてください。

1. 名称や説明に **bash** に関する内容を含む全てのパッケージ
2. インストール済で利用可能な **bash** パッケージ
3. **bash** のパッケージ情報
4. **bash** パッケージの依存情報

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
   ```

これらのコマンドを root と一般のユーザーの両方で試してください。違いがわかりますか？

解 12.2

1. `$ zypper search -d bash`

   -d オプションを付けないと、実際の名称に bash を含むパッケージだけが表示されます。どこに **bash** の記載があるかを見るためには、zypper info をパッケージに対して実行する必要があります。

2. `$ zypper search bash`

3. `$ zypper info bash`

4. `$ zypper info --requires bash`

   このコマンドは **bash** が要求するファイルをリストします。**bash** をインストールするときに依存関係を確認する簡単な方法です。

   ```
   $ sudo zypper remove --dry-run bash
   ```

   bash はシステムに統合されているので、演習の例題としては良い選択とは言えません。実際、削除することはできません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 13 章

Git の基礎

すべての **リビジョン管理システム**の概念を説明し、特にオープンソースで最も人気のあるシステムである **git** を使用するための最初のステップを学びます。

ここでは、**git** を使ったソフトウエア開発の方法ではなく、システムソフトウェアやユーザーソフトウェアをインストールしたりアップデートしたりする目的で、（ソフトウェアパッケージなどの）ファイルセットを参照する複数の方法に焦点を当てます。

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- なぜリビジョン管理システムが重要なのかを説明する。
- **clone**、**add**、**commit**、**tag**、**checkout** など **git** で使用する主要なコマンドを熟知する。
- **git help** で特定のコマンドに関するシステム上のヘルプを得ることができる。
- **git** を利用したシステム設定ファイルの変更履歴を追跡する一連の方法を、実例を使って理解する。
- **tag** や **branch** を使った方法を含め、**アップストリーム** バージョンのソフトウエアのダウンロード方法を学習する。

13.1 リビジョン 管理

リビジョン 管理システムとは

git は **リビジョン 管理システム** ...

リビジョン管理 システムは、ある時点のファイルのバージョンを保存して、変更を追跡するために使われる

- 何時 – 変更時のタイムスタンプ
- 何を – 変更したファイルのバージョン
- 誰が – 変更を実施したユーザー
- なぜ – 変更内容の把握に有益な追加情報

RCS のような非常に古いシステムでも、これらの情報を追跡していた

初期のリビジョン管理システムでは、変更内容はファイル単位で **RSC/** というサブディレクトリにファイル名に **,v** を付与したテキストファイルとして追跡されていました。このシステムでは複数ファイルにまたがる変更をグループ化する機能が欠けていて、ファイルのロックとストレージの効率に欠点もありましたが、上に列挙した基本的な考え方がサポートされていたので十分に有用なツールとなっていました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

リビジョン 管理システムとは

git は **リビジョン 管理システム** . . .

個々のファイル単位で変更を逐次追跡するシステムからの進化の次ステップは、複数変更の同時追跡と **真実のソース（＝マスターソース）**をネットワーク上のサーバーで集中管理させることだった。**CVS** はファイルのチェンジセット（＝複数ファイルの変更内容の集約）の記録にこの基本的なメカニズムを利用していたが、一緒に作業する開発者には同じネットワークへのアクセスを許していた。ファイルの排他的アクセスと、同一ファイルへ適用される変更管理の問題から､残念ながらこのシステムの運用は困難を極めた。

分散リビジョン管理システムは、多くのコントリビュータ（**Linux** カーネルの場合、**数千人規模**）の同時利用に対応している。

- 多くのプロジェクトでは、別々に働く開発者が共通のリビジョン管理システムによって結びつけられています。中央集権的なリポジトリを利用する **CVS** のようなシステムであっても、これに対応することができます。
- **git** は、技術的な意味では中央集権的なリポジトリが存在しないという点が特徴で、基本的なフレームワークは **ピア・ツー・ピア型**となっています。ある特定の場所が果たす重要性や中心的な役割は、仕組みに起因するものではなく、政治的・社会学的に決められるものです。
- **分散開発**とは、プロジェクトのさまざまな部分が別々のホスト リポジトリに分割されて別々に作業することではありません。**git** による分散開発では、各リポジトリは同じ権威を持ち、そこにはプロジェクトの一部分でなくコードベース全体が含まれています。
- **git** には政治的な要素はなく、望ましい組織モデルもありません。開発コミュニティの階層は、非常に中央集権的なトップダウンの場合もあれば、非常にフラットな場合もあります。情報や変更のやり取りは、いくつもの開発ツリーがトップレベルのものに合体しているピラミッド型になることもあれば、非常に平等主義的になることもあります。
- **git** はツールであり、厳格なメソッドではないので、プロジェクトのニーズや哲学に適合した方法で使うことができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

13.2 基本コマンド

いくつかの必須コマンド

```
$ git --version
$ git help [subcommand]
$ git clone
$ git branch
$ git status
$ git add
$ git commit
$ git merge
$ git pull
$ git push
```

インストールされている **git** のバージョンの確認は、以下のようにします。

```
$ git --version
```

```
git version 2.30.2
```

組み込まれたヘルプ

git help サブコマンドを使うと、各 git 呼び出しに関する詳細な man ページを参照できます。

```
$ git help
$ git help コマンド
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

基本的な git コマンド

```
start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects
```

`git` と入力すると、上の写真のような **git** コマンドの基本的なリストが表示されます。

適用されるグローバル オプションは、上記のリストで先頭に `--` が付いているものだけです。サブコマンドの多くには独自のオプションがあり、それらは上記の `[ARGS]` に含まれています。

更に深く知るには

公式レファレンスページを見ると、サブコマンドや環境変数などを含む完全なリストを見ることができます。特定の git バージョン毎に分けて表示されます。
https://git-scm.com/docs/git#_git_commands

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

13.3 いくつかの git コマンドを試す

最小限のグローバル設定

他の最近のリビジョン管理システムとは異なり、git は単純に現在のユーザー ID を使うのではなく **著者名（プログラマーの名前）**とメールアドレスのセットアップすることを要求する

ここでそれらを設定し、更により現代的なデフォルトブランチ名の *main* の設定も追加する

```
$ git config --global user.name "Gladys West"
$ git config --global user.email "gwest@example.com"
$ git config --global init.defaultBranch main
```

これらの設定は `~/.gitconfig` に書き込まれる

`~/.gitconfig` ファイルには、さらに多くの内容を含めることができます。多くの人が、頻繁に入力するコマンドのショートカットや、カスタマイズした変更履歴の表示方法、さまざまな **diff** コマンドを保存しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

リポジトリの作成と初期化

新しいディレクトリを作成し、git リポジトリとして初期化する

```
$ mkdir git-test
$ cd git-test
$ git init
```

```
Initialized empty Git repository in /tmp/LFgit/.git/
```

```
$ git status
```

```
On branch main

No commits yet

nothing to commit (create/copy files and use "git add'' to track)
```

ここまでで、何が起こったのか検証してみましょう。

```
$ ls -la /tmp/LFgit
```

```
total 12
drwxr-xr-x  3 user user 4096 May  9 18:12 .
drwxrwxrwt 16 root root 4096 May  9 18:16 ..
drwxr-xr-x  7 user user 4096 May  9 18:15 .git
```

```
$ ls -la /tmp/LFgit/.git
```

```
ls -l /tmp/LFgit/.git/
total 32
drwxr-xr-x 2 user user 4096 May  9 18:12 branches
-rw-r--r-- 1 user user   92 May  9 18:12 config
-rw-r--r-- 1 user user   73 May  9 18:12 description
-rw-r--r-- 1 user user   21 May  9 18:12 HEAD
drwxr-xr-x 2 user user 4096 May  9 18:12 hooks
drwxr-xr-x 2 user user 4096 May  9 18:12 info
drwxr-xr-x 4 user user 4096 May  9 18:12 objects
drwxr-xr-x 4 user user 4096 May  9 18:12 refs
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

リポジトリにファイルを追加する

/etc/motd を、我々のリポジトリ – にコピーする
これは小さく、（想定通りなら）テキストだけが 含まれて いる

```
# （まだ /tmp/LFgit にいる）
$ cp /etc/motd .
$ git status -sb -uall
```

```
## No commits yet on main
?? motd
```

```
$ git add motd
$ git status -sb -uall
```

```
## No commits yet on main
A  motd
```

現時点では、ファイルはステージングエリアに **追加**されている

少し異なる呼び出し方で **git status** を使ったことに注目してください。

-s は、短縮形で表示する
-b は、短縮形だがブランチ情報まで表示する
-uall は、追跡されていないディレクトリを含め、追跡されていないファイルを表示する

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ファイルを変更する

ステージングエリアに追加されたファイルに変更を加えていく。しかし、まだコミットしない。これは問題か？

```
$ echo meow >> motd
$ git status -sb -uall
```

```
## No commits yet on main
AM motd
```

ステータスが更新された。ファイルはまだ **追加**されているが、同時に **変更**されている。変更内容はどんな内容か？

```
$ git diff
```

```
diff --git a/motd b/motd
index 0c87dd3..c3ce8b3 100644
--- a/motd
+++ b/motd
@@ -5,3 +5,4 @@ individual files in /usr/share/doc/*/copyright.

 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
 permitted by applicable law.
+meow
```

ファイルを追加し、変更し、まだ何もコミットしていない状態でも、その変更内容（差分）を見ることができます。

現代的なリビジョン管理システムが全て、このような挙動をとるのではない点に注意してください。 例えば *mercurial (hg)* はステージングエリアに対してこのような扱いをしません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION | Training & Certification

変更をコミットする

ファイルをコミットしてステータスを確認する。どのバージョンのファイルがコミットされるか？

```
$ git commit -m 'message of the day'
$ git status -sb -uall
```

```
## main
 M motd
```

これでステージングエリアに置かれたファイルのバージョンがコミットされる。もし、更に”meow” を追加した現在のファイルのバージョンをコミットしたいのであれば、改めて **git add** と **git commit** を行う必要がある。

あるいは、**git commit --all --amend** を使うことで、別のコミットを作成せずに一度に行うこともできる。

--all（新規ファイルではなく）変更されたファイルや削除されたファイルを自動的にステージングする
--amend 最後のコミットを新しいコミットに置き換える

```
$ git commit --all --amend
```

コミット メッセージ

```
add critical 'meow' to motd

        * Closes #4563: Where did meow go?

        * After serious testing, we've discovered a missing word that is
          causing production issues.

# あなたが行った変更に対するコミットメッセージを入力してください。
# '#' で始まる行は無視され、メッセージが空のコミットは受け付けられません。
```

正しくフォーマットされたコミットメッセージを書くことは、あなたの仕事にプロフェッショナルな印象を与えるのに大いに役立ちます。コミットメッセージを書くときには、以下のルールに従いましょう。

- 一番上の要約は短く書く - だいたい５０文字程度にする
- 時制は現在形を利用する – 適用された時点でコミット内容がわかるように
- 更に詳細な内容は後続行に書く - この時には一行空行を入れてインデントする

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ステータスを確認する

現状はどうなっているか？

```
$ git status
```

```
On branch main
nothing to commit, working tree clean
```

```
$ git log
```

```
commit 8933f422d17d107d35b80f8a1c55b09dbee289fe (HEAD -> main)
Author: Gladys West <gwest@example.com>
Date:   Fri May 9 20:01:53 2023 -0400

    add critical 'meow' to motd

            * Closes #4563: Where did meow go?

            * After serious testing, we've discovered a missing word that is
              causing production issues.

commit 0872249ec1c9daeb60cb491fe6be7647cf606149
Author: Gladys West <gwest@example.com>
Date:   Fri May 9 19:31:03 2023 -0400

    message of the day
```

これで作業ディレクトリはクリーンな状態になり、コミット内容が全てログに表示されるようになりました。

git log コマンドには、相対時間、変更されたファイルの差分、通常は表示されない追加フィールドなど、多くの設定が用意されています。

以下は、コミットログを一行に収まる **コミット ハッシュの短縮形**で表示させた例です。

```
git  log --pretty=oneline -n 20 --graph --abbrev-commit
```

```
* 8933f42 (HEAD -> main) add critical 'meow' to motd
* 0872249 message of the day
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

13.4 Git を使ってソフトウエアを入手する

ソフトウエアのダウンロード

ファイルの変更や、変更内容の追跡には興味がなく、単純に git を使ってソフトウエアパッケージや設定を入手したいだけのケースも多い、何故なら

- それがソフトウエア著作者が推奨するソフトウエアの入手方法だから
- ディストリビューションのバージョンより、新しいか、より完全だから
- それが、ソースと変更内容を確認するための最も簡単な手段だから

包括的なエコシステムや **ソフトウエア フォージ（＝開発支援サイト）**によるソフトウエア開発は git を中心に進められており、web ブラウザーだけでそれらを利用できる。　人気のあるものは以下のようなものである。

- **GitHub**
- **GitLab**
- **Codeberg**
- 開発者個人や組織が ”自主的にホストしている” 開発支援サイト

通常は、特定のリリースバージョンを指定したダウンロードは web ブラウザーだけがサポートしているのですが、ディスカションへの参加や、問題の報告や、ソースコードの閲覧は、モバイル ブラウザーからでも可能となっている場合が多いのです。

以下のようにすれば、多くのプロジェクトをランダムに見ることができます。上記のサイトでは、それぞれホスト名の最後に `"/explore/"` を追加することでこの機能をサポートしています。

- https://github.com/explore/
- https://gitlab.com/explore/
- https://codeberg.org/explore/

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Linux カーネルの入手

clone サブコマンドで **カーネル**のソースコードの自分用のローカル コピーを入手する

```
$ git clone --depth 1 -b master \
  https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git linux/
```

```
Cloning into '/home/user/linux'...
remote: Enumerating objects: 85093, done.
remote: Counting objects: 100% (85093/85093), done.
remote: Compressing objects: 100% (82795/82795), done.
remote: Total 85093 (delta 6421), reused 17237 (delta 1404), pack-reused 0
Receiving objects: 100% (85093/85093), 237.53 MiB | 1.34 MiB/s, done.
Resolving deltas: 100% (6421/6421), done.
Updating files: 100% (80335/80335), done.
```

大規模なプロジェクトの場合、ダウンロード 時間や使用するディスク容量を減らすために、履歴を全て削除した最新の（または別の特定に）コミットだけを見たい場合には、いくつかの追加オプションを指定できます。

これらのオプションを指定しなかった時には、**git clone** は全ての履歴を含めて **デフォルト ブランチ**を取得します。

git clone に以下のオプションを指定しました。

-b を使うと、指定されたブランチをクローンします。

-depth を指定すると、指定された深さのコミットだけを含む浅いクローンを作成します。

clone コマンドの結果を確認してみましょう。

```
$ git log
```

```
commit 9561de3a55bed6bdd44a12820ba81ec416e705a7 (grafted, HEAD -> master, tag: v6.4-rc5, origin/master, origin/HEAD)
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Sun Jun 4 14:04:27 2023 -0400

    Linux 6.4-rc5
```

コミットログには１つのエントリーしかない点に注目してください。

Linux カーネルについて、更に知りたい？

The Linux Foundation には、組み込みシステムやカーネル セキュリティなどを含めたカーネル開発に関するトピックスを扱うクラスが用意されています。https://training.linuxfoundation.org/full-catalog/?_sft_topic_area=linux-kernel-development

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ツリー式（tree-ish）

たった今、最新 **Linux** カーネルのダウンロード時に **ブランチ** を指定した。
この意味は何か？

- ほとんどの git コマンドは、**コミット レファレンス**（＝特定の瞬間の特定のファイルの集まり）に対して動作する。
- **ツリー式（tree-ish）**は **コミット レファレンス** を指定する構文で、以下を含む
  - 完全な **コミット ハッシュ** – `9561de3a55bed6bdd44a12820ba81ec416e705a7`
  - そのハッシュの短縮版 – `9561de3a5`
  - **ブランチ** – `main` または `devel`
  - **tag** – `v6.4-rc5`
  - 特定の **リモート** バージョン – `refs/remotes/origin/main`

プロジェクトのディレクトリにある `README` や `INSTALL` ファイルには、そのプロジェクトの特定のバージョンを取得するための最適な方法が指定されていることがよくあります。**git** や **コード フォージ（＝開発支援サイト）**の機能の命名方法はプロジェクトによって大きく異なるため、この情報を確認することは重要です。

例えば、`main`（または、古いプロジェクトでは `master`）が不安定な作業ブランチを指すこともあれば、テスト済でリリースに適したバージョンを指すこともあります。

更に深く知りたい？

git-revisions のマニュアル ページには、特定のコミットを指定するためのツリー式（tree-ish）の使用例が紹介されています。
https://mirrors.edge.kernel.org/pub/software/scm/git/docs/gitrevisions.html#_specifying_revisions

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ブランチという概念のアナロジー（＝たとえ話）

git の考え方を理解するために、物理的なものになぞらえて考えてみるとわかりやすいかもしれない。もし各コミットをバインダーの１ページだとみなすなら、**ブランチ**は最新の最後のページを簡単にめくれるようにする整理用のタブが付いたバインダーと考えることができる。

バインダーの最後にページを追加して **コミット**をした時にも、末尾のタブはやはり最新の最後のページ／**コミット**を素早く指し示している。

このブランチから新しい **ブランチ**を作るのは、全てのページをコピーして新しいバインダーにいれるのと同じことである。新しいバインダーに新しいページを追加したとしても、古いバインダーの元の最終ページと最後のタブが変更されることはない。

最後に、作成したそれぞれのバインダー／**ブランチ** が全て同じ棚に置かれていると考えてみる。その棚が git **リポジトリ** で、`.git` サブディレクトリのあるディレクトリにローカルに存在している。

現在のブランチをコピー元として、新しいブランチを作成するには次のようにします。

```
$ git checkout -b <name>
```

アップストリーム ブランチをコピー元として、新しいブランチを作成するには次のようにします。

```
$ git checkout -b <name> remotes/origin/main
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

フォークか、クローンか

たとえ話を続けると、私達の棚をいくつものバインダーの形をとった **ブランチ**を含む git リポジトリーだと見なした場合、それらのバインダー／**ブランチ** の１つ以上をコピーして **クローン** を作成し、まったく新しい **物理的な** 棚／**リポジトリ** に置く。

フォーク と **クローン** は git の世界ではでは事実上同じことを意味している。**Github** が **フォーク** という用語の利用を広めたが **クローン** は、もともと git プログラム内の機能の名前である。

重要

同じファイル システム上にある `git clone` がバックアップになると期待してはいけません。ローカルで **git clone** を実行すると、**git** は新しいサブディレクトリに `.git` のコピーを作成しますが、スペースを節約するため可能な限り **ハードリンク** を使用しているからです。

フォークという単語には過剰に多様な意味で使われています。歴史的には、開発経路から分岐されたソフトウエアのコピーの意味でした。現在では、ブランチやブランチのスタンドアロン（物理的な）コピーを指すこともあります。しかし、開発経路から分岐したスタンドアロン コピーを意味すること **も** あるのです。これは混乱の元になります。

ハードリンクを使わない形で同じファイルシステムにローカルを作る（＝クローンする）には、以下のようにします。

```
$ cd /tmp/LFgit
$ git clone . --no-hardlinks /tmp/meow/
```

```
Cloning into '/tmp/meow'...
done.
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

特定の場所にタグを付ける

例え話の最後に **タグ** を考えてみる。**タグ** は、新しくコミットしても **自動的に付け替えられない** バインダーのもうひとつの クイック レファレンス タブ だと考えることができる

タグ は特定のコミットに別名を付ける目的で利用され、しばしば、リリース バージョンの明示に利用される

あるコミットに付けられたタグを別のコミットに付け替えるには、タグを一度削除してから再作成する必要がある

git tag のデフォルトの挙動は利用しない

多くのプロジェクトでは **注釈付きタグ** や **署名付きタグ** を使用しているので、独自のコミットメッセージが要求されます。**git tag** のデフォルトの挙動ではコミット メッセージは聞かれませんが、そのようないわゆる **軽量** タグはツールによっては無視されます。

git tag で **sign** オプションを使用するためには、GnuPG の設定をする必要がありますが、ここではそれについては触れません。しかし、最後のステップを除けば、それは **注釈付き** タグを作成する手順と同じです。

git tag

-a, –annotated < タグ名 > は、符号なし注釈付きタグ オブジェクトを作成します

例題を実行しているディレクトリに移動します。

```
$ cd /tmp/LFgit
$ git tag -a v1.0 -m 'release intial version'
$ git log --pretty=oneline v1.0
```

```
8933f422d17d107d35b80f8a1c55b09dbee289fe (HEAD -> main, tag: v1.0) add critical 'meow' to motd
0872249ec1c9daeb60cb491fe6be7647cf606149 message of the day
```

バージョンを表す v

ブランチ と **タグ**に別々の命名ルールを適用するのは良い習慣です。しばしば、**タグ** の追跡リリースバージョンは **v** で始まります。そのため、バージョン 3.17 に向けて作業を行っているブランチは単純に *3.17* とし、完成したリリースは *v3.17* というタグが付けられるかもしれません。

この命名ルールに従うことで、**ブランチ** や **タグ**を最も単純なツリー式（tree-ish）で参照できるようになります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

13.5 演習

課題 13.1: アップストリーム ソフトウエア リポジトリをクローンし、ローカルに変更を加える

システム管理では比較的一般的な作業である、ディストリビューションに含まれていない、または、含まれているがバージョンが古いソフトウエア パッケージへの対処について説明します。

使用するパッケージは http://joeyh.name/code/moreutils/ にある *moreutils* で、おそらくあなたが利用しているディストリビューションではパッケージ化され、最新版になっているはずですが、この例ではそうなっていません。

1. かなりの数の **ブランチ** と **タグ** を見る必要があります
2. それは、多くの優れた Rust や Go ユーティリティほどには、多くの依存関係を必要としません
3. これを、インストールして利用したいと思うなら実際役にたちます

Debian バージョンをターゲットにすることで、ソースサーバーに少し優しくなり、もう少しリポジトリ情報を利用できるようになります。演習には次の url を使ってください。

https://salsa.debian.org/nsc/moreutils

1. まず、リポジトリを **クローン** します。
2. `debian/latest` ブランチをベースに、`local/latest` という名前のローカル ブランチを作成します。
3. リポジトリ内に `README.local` というファイルを作成し、何か好きなことお書いてください。
4. 新しいファイルを新しいブランチにコミットします。`user.name` と `user.email` の設定が必要な場合に、は前の **最小限のグローバル設定** のセクションを参照してください。
5. （オプション）現在のバージョンに基づいて注釈付きタグを作成します。
6. （オプション）パッケージをビルドし、ツールを使って実験します。

これらすべてを行うさまざまな段階で、`.git` ディレクトリの中身を調べ、どのファイルが変更されているか、その内容は何かなどを確認してみてください。できる限り多くのことを学んでください。

解 13.1

1. `$ git clone https://salsa.debian.org/nsc/moreutils`

```
$ git clone https://salsa.debian.org/nsc/moreutils
Cloning into 'moreutils'...
warning: redirecting to https://salsa.debian.org/nsc/moreutils.git/
remote: Enumerating objects: 2107, done.
remote: Counting objects: 100% (229/229), done.
remote: Compressing objects: 100% (105/105), done.
remote: Total 2107 (delta 129), reused 208 (delta 121), pack-reused 1878
Receiving objects: 100% (2107/2107), 510.79 KiB | 1.45 MiB/s, done.
Resolving deltas: 100% (1349/1349), done.
```

2. `$ git checkout debian/latest`

```
Already on 'debian/latest'
Your branch is up to date with 'origin/debian/latest'.
```

（または）

```
$ git status
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
On branch debian/latest
Your branch is up to date with 'origin/debian/latest'.

nothing to commit, working tree clean
```

実際には **何も起こりません。**なぜなら **デフォルトのブランチ** がすでに debian/latest になっているからです。
ですから、これらのコマンドは以下のどちらでも動作します。

```
$ git checkout -b local/latest
```

```
Switched to a new branch 'local/latest'
```

（または）

```
$ git checkout -b local/latest debian/latest
```

```
Switched to a new branch 'local/latest'
```

しかし、最後のものは、現在チェックアウトしている **ブランチ** に関係なく機能します。

3. `$ touch README.local`

4. ```
   $ git add README.local
   $ git commit README.local
   ```

   ```
   [local/latest cc7d808] add local Tracking file
    1 file changed, 0 insertions(+), 0 deletions(-)
    create mode 100644 README.local
   ```

5. `$ git tag -a vlocal/1.0 -m 'first local version'`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
   ```

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 14 章

プロセス

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- プロセスおよびそれに関連するリソースについて説明する。
- プロセス、プログラム、スレッドの違いを理解する。
- プロセスの属性、権限、状態を理解し、制限を制御する方法を把握する。
- プロセスの生成、バックグラウンドへの移動、フォアグラウンドへの移動を理解する。
- **ジョブ**（バックグラウンドプロセス）の管理、**at** を使って時間経過後にプロセスを開始する方法、**スリープ**を使った実行の一時停止、そして **cron** を使って定期的な実行をスケジューリングする。
- プロセスが取り得る状態を説明する。
- ユーザーモードとカーネルモードでの実行の違いを説明する。
- デーモンプロセスについて説明する。
- 新しいプロセスがフォーク（作成）される仕組みを理解する。
- nice と renice を使って優先順位の設定、変更を行う。

14.1 プログラムとプロセス

プログラムとは何か

- **プログラム** とは、命令の集まり（と実行に必要なデータ）である
- CPU が直接実行できるマシンレベルの命令を含むことがある
- 別のプログラムによる翻訳が必要な命令を含むこともある
- プログラムの中では、命令を記述するために色々な言語（**C, C++, Perl,** その他）が使われる

プログラム 対 スクリプト

しばしば、**コンパイル** によってバイナリーの実行形式に変換されるプログラムは、実行に **bash**、**Python**、**Perl** などのインタープリターが必要となる **スクリプト** とは区別されます。

プログラムとは、一連の命令と命令を実行する時に使用する内部データのことを言います。またプログラムが外部データを使用する場合もあります。内部データには、例えばユーザープロンプトの表示に使用されるプログラム内のテキスト文字列があります。外部データには、例えばデータベースのデータがあります。

ls、**cat**、**rm** などの多くのユーザコマンドは、オペレーティング システムのカーネルやシェルとは別のものです。（つまり、ディスク上にある独立した実行可能プログラムです）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

プロセスとは何か？

- **プロセス** とは実行中のプログラムのインスタンスである
- **Linux** は、実行中ないしこれから実行するプログラムそれぞれに新しいプロセスを作る
- ひとつのプログラムが、複数のプロセスを同時に実行することがある
- オペレーティング システムの重要な役割は、ユーザーに代わってプロセスの実行を管理することである

マルチスレッディング

プログラムは、同時実行される複数の **スレッド** によって実行されることがあり、それぞれのスレッドが独自のプロセスと見なされます。（訳注：一つのプロセスは、実行実体としてのスレッドを複数持つことができ、スレッド単位でスケジューリングされます）

プロセスとは、実行中プログラムのインスタンスです。実行中 や スリープ中 などさまざまな状態（ステート）を持ちます。

全てのプロセスには **pid**（プロセス ID）、**ppid**（親プロセス ID）、**pgid**（プロセスグループ ID）があります。さらに全てのプロセスはプログラムコード、データ、変数、ファイル記述子、および環境を持っています。

プロセスは完全にプリエンプティブに **スケジューリング** されます。プロセスをプリエンプションする権限を持つのはカーネルだけです。プロセス同士ではできません。

過去のいろいろな経緯から、**pid** の最大値は 16 ビット数、つまり 32768 に制限されています。`/proc/sys/kernel/pid_max` を変更することでこの値を変更することができます。大規模なサーバーでは、デフォルトの最大値では足りない場合があるためです。プロセスが生成され続けて行くと、最終的にその数は `pid_max` になります。そうすると再び `PID = 300` から割り当てられます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

プロセス属性

- プロセスは、全ていくつかの属性を持つ
  - 実行されたプログラム
  - コンテキスト（ステート）
  - 権限
  - 関連づけられたリソース

全てのプロセスが何らかのプログラムを実行しています。プロセスは CPU レジスタの状態、プログラム内で実行されている箇所、プロセスのメモリーの内容、およびその他の情報を補足することにより、いつでも自身のスナップショットを取ることができます。このスナップショットがプロセスの **コンテキスト** です。

プロセスは CPU 時間を他のユーザーと共有する（または、ユーザー要求やデータ到着を待つなどの条件が満たされるまでスリープしなければならない）ときにプロセスのスケジュール切り替えができるので、プロセスをスワップアウトする時に全てのコンテキストを保存し実行時に復帰させる **コンテキスト スイッチング** の機能はカーネルの重要な役割になります。

全てのプロセスには、実行のために呼び出したユーザーに基づいたパーミッションがあります。また、プログラムファイルの所有者に基づいたパーミッションもあります。「s」実行ビットでマークされたプログラムには、「実行」ユーザー ID とは異なる「実」ユーザー ID があります。これらのプログラムは **setuid** プログラムと呼ばれます。これらはプログラムを所有するユーザーのユーザー ID で実行されます。これに対して非 setuid プログラムは、プログラムを開始したユーザーのパーミッションの範囲内で実行されます。ただし root が所有する setuid プログラムの実行はセキュリティ上の問題になる可能性があります。

setuid プログラムの例として **passwd** コマンドがあります。これは全てのユーザーが実行できます。ユーザーがこのプログラムを実行するとプロセスは root 権限で実行され、ユーザーのパスワードが保持されている書き込み制限があるファイル `/etc/passwd` と `/etc/shadow` を更新できるようになります。

全てのプロセスは、割り当てられたメモリー、ファイルハンドラなどのリソースを持っています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

プロセスとリソースの分離

- カーネルは（デフォルトでは）、個々のプロセス同士を隔離する
  - セキュリティ確保のため
  - 安定性の向上のため
- カーネルはプロセスがハードウェア資源にアクセスすることを認めない
  - ハードウェアへのアクセスはカーネルが管理
  - **システムコール** を利用してアクセスさせる

プロセスが開始されると、そのプロセスは他のプロセスから保護するために独自のユーザー空間を持ちます。これによりセキュリティが向上し、安定性が向上します。

プロセスはハードウェアに直接アクセスできません。ハードウェアはカーネルにより管理されるため、プロセスは **システムコール** を使用して間接的にハードウェアにアクセスします。システムコールはアプリケーションとカーネルの間の基本的なインターフェイスです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.2 プロセスに対するリミット設定

ulimit でプロセスを制御する

- ulimit は **bash** に組み込まれたコマンドである
- プロセスに関連するリソース制限の表示、または再設定を行う
- 個々のユーザーやプロセスが、以下のようなシステムリソースを使い果たすことがないように機能を **制限** するために利用される
  - メモリー
  - CPU 時間
  - システム上のアクティブなプロセスやファイルの最大数
- システム管理者は、全てのユーザーに影響するようなシステム全体の制限をかけることができる

システム管理者は、以下のどちらかの方向に向けていくつかの値を変更する必要があるでしょう。

- 機能を **制限** して、個々のユーザーやプロセスがメモリー、CPU 時間、システム上のプロセスの最大数などのシステムリソースを使い果たすことがないようにする
- 機能を **拡張** して、プロセスがリソースの限界に達しないようにする。たとえば多くのクライアントを処理するサーバーでは、デフォルトのファイルオープン数の制限 1024 のままではサーバーとして機能しない場合があります。

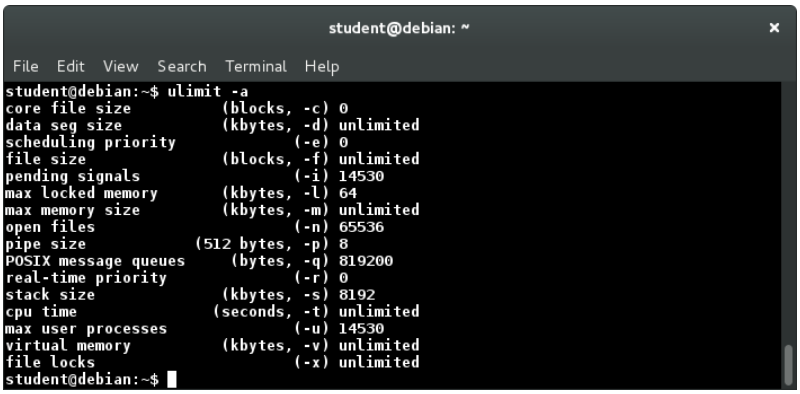


図 14.1: **ulimit**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

リミットの設定

- 以下を実行することにより特定の制限値を設定できる

```
$ ulimit [options] [limit]
```

```
$ ulimit -n 1600
```

同時に開くことができるファイルの最大数が 1600 になる

- /etc/security/limits.conf の設定値を修正してリブートすると設定が恒久化する

変更は、現在のシェルにだけ影響することに注意してください。ログインしている全てのユーザーに有効な変更を加えるには /etc/security/limits.conf （とても良いドキュメントが付属しています）を変更してから再起動する必要があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ソフトリミット と ハードリミット

リミットには、次の 2 種類がある

- **ハード**: ユーザーが引き上げることができるリソース制限の最大値、root ユーザーだけが設定可能
- **ソフト**: 現在の制限値。一般ユーザーが変更できるがハードリミットを超えることはできない

```
$ ulimit -H -n
```

```
4096
```

```
$ ulimit -S -n
```

```
1024
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.3 プロセスの生成

プロセスの生成

- 最初のユーザープロセスは **init** と呼ばれ `pid 1` を持つ
- 後続プロセスは **init** や他の実行中プロセスから **派生 (fork)** して作られる
- **親プロセス** から派生された **子プロセス** は、親プロセスの完全なクローンとしての特徴を持つ
- 親プロセスが死んだ場合には子プロセスは **init** の養子になる
- カーネルもプロセスを生成する。**ps** コマンドで見るとそれらは [..] で囲まれた名前を持つ

注目

systemd ベースのシステムでは `pid=2` の **kthreadd** という孤児となったプロセスを養子にするための特別なプロセスがある。養子となったプロセスの親プロセス ID は `ppid=2` となる

通常の **Linux** システムは常に新しいプロセスを生成します。新しくプロセスを生成することをしばしば **派生 (fork)** と呼びます。新しい子プロセスがスタートしても元の親プロセスも動き続けます。

しばしば fork だけで終わらずに後に **exec** が続きます。この時親プロセスは終了し、子プロセスが親プロセスのプロセス ID を継承します。しばしば **fork and exec** とペアで呼ばれるのでこれが一つの命令だと考えている人も多いのです。

古い **UNIX** システムでは **spawn** と呼ばれるプログラムが多用されました。これは多くの点で fork and exec と似た動作をするものですが細部で違いがあります。**spawn** は **POSIX** との互換性がなく標準的な **Linux** には使われていません。

どうやって新しいプロセスが起動するかを見るのに、多くのクライアントが接続されたウェブサーバーを考えてみましょう。クライアントから新しい接続がある毎に新しいプロセスが立ち上がります。または、同じプロセスの中で新しい **thread** をスタートさせるだけの場合もあります。**Linux** では、技術的にみれば、完全に新しいプロセスを作るのと単にスレッドを作るのには大きな違いはありません。どちらの方法でも起動時間はほぼ同じで消費するリソースも大差ありません。

別の、**init** プロセスが **sshd** init スクリプトを実行して **sshd** デーモンを起動するケースでは、**init** プロセスの役割は **sshd daemon** を起動するまでです。以降はデーモンがリモートユーザーからの **ssh** 接続要求を待ち受けます。

接続要求が届くと **sshd** は接続リクエストに応答するために自分自身のコピーを作成します。リモートから接続する各ユーザーは自分専用の **sshd** デーモンを使ってリモートログインを処理できるようになります。ログインプログラムがリモートユーザーを確認すると **sshd** がスタートします。認証が成功するとログインプロセスが (**bash** と呼ばれる) シェルを fork してユーザーのコマンドを解釈できるようにします。などなど

カーネルの内部プロセスは色々なメンテナンス業務を担っています。例えば、バッファーされたデータをディスクに書き出すとか、各 CPU の負荷がバランスするようにワークロードを調整するとか、デバイスドライバーに要求されているデバイス操作を実行させるといった仕事です。これらのプロセスはシステム稼働中ずっと何かやることが発生するまではスリープした状態で残り続けます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

コマンドシェル 内で プロセス を生成

ユーザーが **bash** などのコマンドシェルインタープリター上でコマンドを実行すると

- 新しいプロセスが生成 (ユーザーのログインシェルから fork される)
- wait システムコールは、親シェル プロセスをスリープ状態にする
- **exec** システムコールによってコマンドが子プロセスにロードされ **bash** を置き換える
- コマンド完了時には子プロセスは exit システムコールを発行して死ぬ
- 子プロセスが死ぬと親プロセスが再開、新たなシェルプロンプトを出す
- 親プロセスはユーザーの次のコマンド入力を待つ、このサイクルが繰り返される

もしコマンドが（コマンドラインの最後にアンパーサント (&) を付加することによって）**バックグラウンド** 処理するように起動された場合には、親プロセスは wait リクエストをスキップしてすぐにシェルプロンプトを出します。この時バックグラウンドに回ったプロセスも並列に実行されます。そうではなく **フォアグランド** 実行の場合シェルは子プロセスの処理が完了するかシグナルによって停止するまで待機します。

いくつかのシェルコマンド（**echo** や **kill** など）はシェルのビルトインコマンドなので、実行時に外部のプログラムをロードしません。これらのコマンドを実行する時には **fork** も **exec** も発行されません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.4 プロセス コントロール

バックグラウンド プロセス と フォアグランド プロセス

- プロセスはデフォルトでは **フォアグランド** で実行
- コマンドの後ろに (&) を付けるとバックグラウンドで実行

```
$ sudo updatedb &
```

- CTRL-Z でフォアグランドプロセスをサスペンド
  - **bg** でバックグラウンド実行に移行
  - **fg** でフォアグランド実行に移行
- CTRL-C でフォアグランド プロセスを停止

フォアグラウンドジョブはシェルから実行され、ジョブが終了するまでシェルを拘束します。。通常これは問題ではありませんが、ジョブ完了まで長時間がかかる場合、バックグラウンドに置くことでシェルを解放し、次の 対話的な作業が開始できます。バックグラウンドジョブは低優先度で実行され、対話的な作業をスムーズに進めるられます。また、ログオフして端末を終了してもバックグラウンドジョブには影響を与えません。

```
 $ updatedb &
[1] 22209
$ sleep 100
^Z
    [2]+  Stopped                 sleep 100

$ bg
    [2]+ sleep 100 &

$ fg
    sleep 100
    ^Z
    [2]+  Stopped                 sleep 100

$ bg
    [2]+ sleep 100 &

$ fg
    sleep 100
    ^C
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ジョブの管理

- **jobs** は、現在の端末のバックグラウンド プロセスを表示する
- ジョブ ID、ステータス、コマンド名を表示する

```
$ jobs
```

```
  1 -  Running     updatedb &
  2 +  Stopped     sleep 10
```

- ジョブ ID を使って **bg** と **fg** を実行できる

```
$ jobs -l
```

を実行すると、そのジョブの PID を返します。

バックグラウンドジョブは、ログオフするとそのウィンドウから開始された **ジョブ** が表示されなくなるという点で、ターミナル ウィンドウとまだ多少つながっています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

14.5 プロセスの実行開始を遅らせる

at を使って実行開始を遅らせる

- 任意の非対話型コマンドを指定時刻に実行する:
- コマンドの実行開始を遅延させる:

```
$ at now + 2 days
```

```
at> mail < /var/log/messages admin@example.com
at> <EOT>
job 1 at 2013-01-16 13:24
```

- CTRL-D を使って EOT 文字を挿入する
- **atq** で実行中ないし待機中のジョブを確認する

未来の時間を指定して **at** を入力することで、**at** のインタラクティブな部分が開始されます。プロンプトで、実行するコマンドを入力し、エンターキーを押し、CTRL-D で終了します。その後 **atq** を使用してジョブ情報を確認します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

cron

- **cron** is used to schedule commands at specific intervals
- **cron** はコマンドを指定間隔で実行するようにスケジュールする
- 指定方法にはいくつかのやり方がある
  - **crontab** を使うとユーザーがジョブを指定可能
  - /etc/cron.d/ は規定のジョブファイルで拡張可能
  - /etc/cron.{hourly,daily,weekly,monthly} に任意のシステムスクリプトを含めることが可能
- **cron** は何十年も前から **UNIX** 系 OS で使われてきましたが、最近の **Linux** ディストリビューションでは、次に説明する **anacron** に移行しています。

cron は非常に便利で柔軟なツールです。バックアップなど定期実行するジョブの設定に使用します。タスクは、1 時間毎、毎日、週 1 回、月 1 回、あるいは 10 秒毎に実行するようキューイングできます。ジョブが完了時や失敗時には、メールが送信されます。

```
$ ls -l /etc/cron.d
```

```
total 16
-rw-r--r-- 1 root root 128 Mar 29  2017 0hourly
-rw-r--r-- 1 root root  78 Dec 29 09:24 atop
-rw-r--r-- 1 root root 108 Jun 13  2017 raid-check
-rw------- 1 root root 235 Mar 29  2017 sysstat
```

```
$ ls -l /etc/cron.daily
```

```
total 68
-rwxr-xr-x 1 root root   302 Mar  7 08:55 docker-latest-logrotate
-rwxr-xr-x 1 root root 18812 Mar 20 00:01 google-chrome
-rwxr-xr-x 1 root root     1 Jan 17  2017 google-earth
lrwxrwxrwx 1 root root    45 Jun 16  2016 google-talkplugin ->
↪  /opt/google/talkplugin/cron/google-talkplugin
-rwx------ 1 root root   219 Jan 24  2017 logrotate
-rwxr-xr-x 1 root root   618 Mar 17  2014 man-db.cron
-rwx------ 1 root root   208 Feb  4  2016 mlocate
-rwxr-x--- 1 root root  2126 Aug 17  2016 prelink
-rwx------ 1 root root   256 Sep  1  2017 rhsmd
-rwxrwxr-x 1 root root 18775 Mar  5 13:55 slack
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

anacron

- 古い **Linux** システムでは、`crontab` は `/etc/cron.*` サブディレクトリにジョブを実行するタイミングに関する情報を含んでいた
- しかし、械は常に動いているものという暗黙の了解に基づいていた
- もしマシンの電源がオフなら、スケジュールされたジョブは実行されない
- **anacron** は最近のシステムで **cron** を置き換えた
- **anacron** はシステム稼働時間内で、予定されていたジョブを制御された状態で時間をずらして実行する
- キーとなる設定ファイルは `/etc/anacrontab` である

anacron 設定ファイルの例:

/etc/anacrontab

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days delay in minutes job-identifier   command
1       5       cron.daily        nice run-parts /etc/cron.daily
7       25      cron.weekly       nice run-parts /etc/cron.weekly
@monthly 45     cron.monthly      nice run-parts /etc/cron.monthly
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

スリープ

- **sleep** を使ってコマンドの実行を簡単に遅らせることが可能

```
$ (sleep 600; launchbackup.sh) &
```

sleep はシェルスクリプトでも使用できて、例えば、会議の時間が近づいていることを 10 分ごとにリマインダとして表示できます。引数のデフォルトは 秒 ですが、秒 を *s* 、分 を *m*、時 を *h* 、日 を *d* と明示的に指定することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

14.6 プロセスの状態

プロセスの状態

- プロセス次の何れかの状態になる
  - **実行 (Running)**: CPU 上で実行中、または OS が許可すれば実行できる状態
  - **待機 (Waiting)**: ディスク、ないし何らかの外部リソースからの情報を待っている状態、**スリープ** とも呼ばれる
  - **中断 (Stopped)**: デバッガー、または Ctrl-Z によって実行を止められた状態
  - **ゾンビ (Zombie)**: 終了済みだが、正しく親プロセスから正しく切り離されていない状態

プロセスは、現在 CPU または CPU コアで **実行中** か、実行キューにいて新しいタイムスライスの割り当てを待っています。スケジューラが CPU を占有するべきと判断した場合、または別の CPU がアイドル状態になりスケジューラがその CPU にプロセスを移行した場合、実行を再開します。

待機 (または **スリープ**) プロセスは、リクエスト（通常は I/O）が実行されるのを待っている状態です。リクエストが完了するまで待ち続けます。割り込みイベントによってリクエストが完了すると OS は待機していたプロセスに再び CPU 資源の最割り当てを許可しプロセスを続行させます。

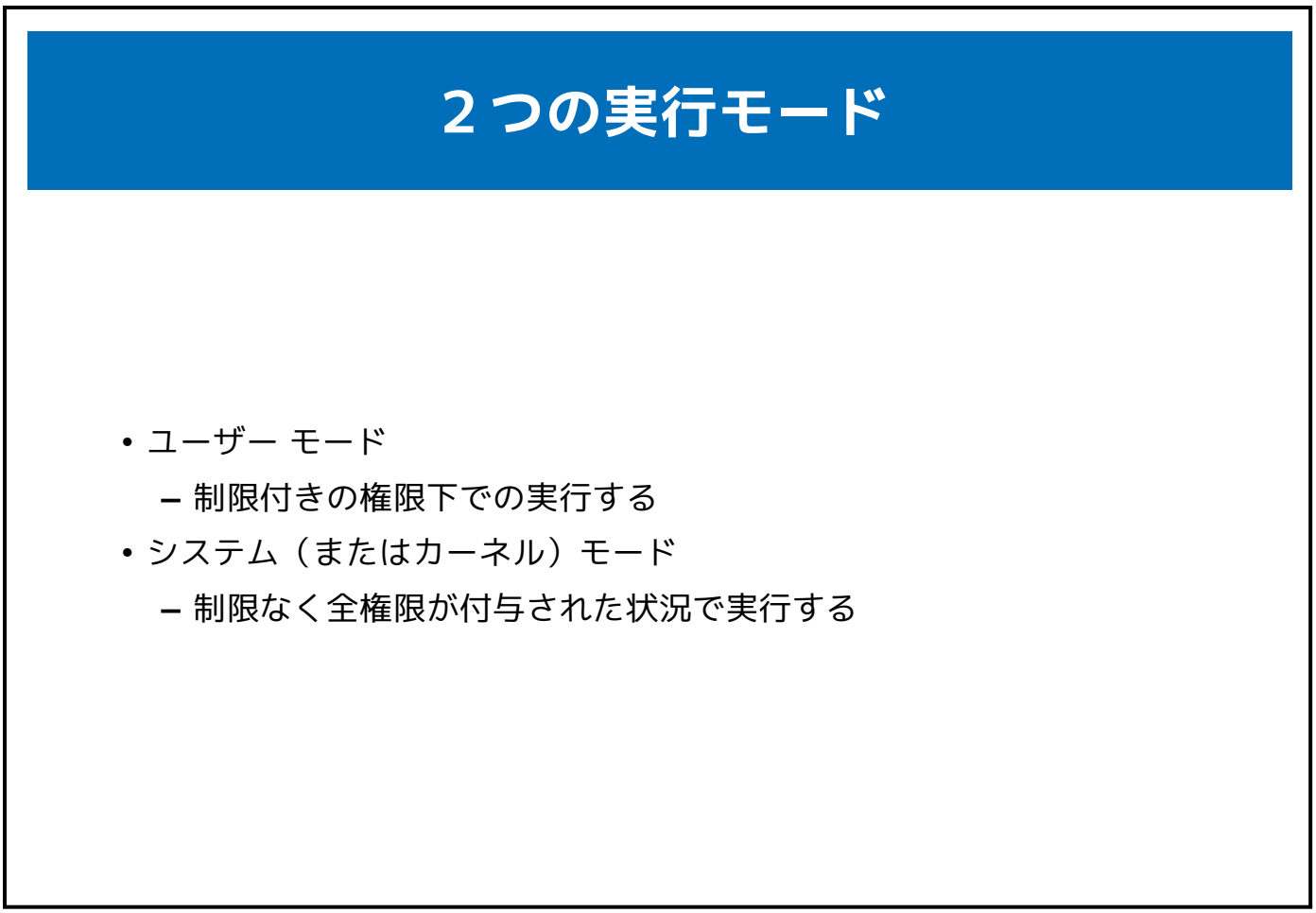
プロセスは **中断** されています。この状態はプログラマが実行中のプログラムのメモリー、CPU レジスタ、フラグ、またはその他の属性を調べたい場合によく発生します。この調査が完了するとプロセスを再開できます。

プロセスが終了すると **ゾンビ** 状態になります。ゾンビ状態となったプロセスも OS のプロセステーブルには残っていることに注意してください。プロセステーブルにはシステム内の全てのアクティブプロセスのエントリーが登録されています。ゾンビ状態となるとプロセステーブルへのエントリー以外の全てのリソースは開放されます。

スケジューラは全てのプロセスを管理します。プロセスの状態はプロセスリストによって知ることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.7 実行モード



いつでもプロセス（またはマルチスレッドプロセスの特定のスレッド）は **ユーザー モード** または **システム モード** で実行できます。カーネル開発者は、通常、システム モードを **カーネル モード** と呼びます。

実行できる命令はモードによって異なり、ソフトウェアではなくハードウェアレベルで実行されます。

モードはシステムの状態ではありません。これはプロセッサの状態です。マルチコアまたはマルチ CPU システムでは各ユニットが個別に独自の状態になることができます。

Intel はユーザー モードを **Ring 3** 、システム モードを **Ring 0** と呼びます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ユーザー モード

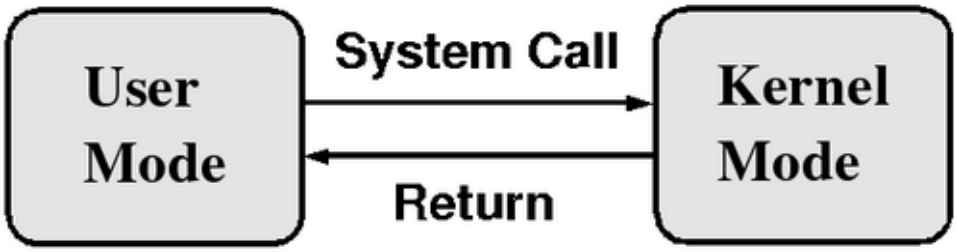


図 14.2: **ユーザーとカーネル モードとシステム コール**

- **ユーザーモード** は **カーネルモード** より権限が限定的である
- 通常のプログラムはこのモードで実行される

システムコール（次のセクションで説明します）を実行する場合を除きプロセスは **ユーザーモード** で実行されます。ユーザー モードはカーネル モードよりも権限が低くなります。

プロセスが開始されると、そのプロセスは他のプロセスから保護するために独自のユーザー空間を持ちます。これによりセキュリティが向上し安定性が向上します。これを **プロセス リソース分離** と呼ぶことがあります。

ユーザー モードで実行する各プロセスには独自のメモリー空間があり、その一部は他のプロセスと共有できます。共有メモリー セグメントを除きユーザー プロセスは他のプロセスのメモリー空間の読み書きはできません。

root ユーザーによって実行される、または setuid プログラムとして実行されるプロセスもユーザーモードで実行されます。システムコールにジャンプする場合を除きハードウェアへのアクセスは限られています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

システム（カーネル）モード

- プログラムによるシステムコール発行で起動される
- システムコールの完了まで継続する
- ユーザーモードより高い権限が付与される
- システム内でカーネルコードを実行するプロセスである
  - ディスクアクセス（読み込み、書き込み）
  - 追加メモリーの要求 (brk, sbrk)
  - カレントディレクトリの変更 (cd)

システム（カーネル）モードでは CPU は、周辺機器、メモリー、ディスクなど、システム上の全てのハードウェアに完全にアクセスできます。アプリケーションがこれらのリソースにアクセスする必要がある場合、**システムコール** を発行する必要があります。これにより、ユーザーモードからカーネルモードへの **コンテキスト スイッチング** が発生します。ファイルの読み取りや書き込み、新しいプロセスの生成などを行う場合はこの手順に従う必要があります。

アプリケーションコードはカーネルモードで実行されることはなく、カーネルコードであるシステムコールだけが実行されます。システムコールが終了すると戻り値が生成されプロセスは逆コンテキストスイッチでユーザーモードに戻ります。

ハードウェア割り込みを処理したり、システムのスケジューリング ルーチンやその他の管理タスクを実行したりするときなど、プロセスとは関係なくシステムがカーネルモードにある場合もあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.8 デーモン

デーモン

- **デーモン プロセス** は、ユーザーにシステムの特定サービスを提供することが唯一の目的である
- 必要なときにだけ動作するため、非常に効率的である
- 多くのデーモンはブート時に開始される
- デーモンの名前の多くは（常にではありませんが）最後に **d** が付く
  例えば **httpd** や **systemd-udevd** など
- 外部イベント (**systemd-udevd**) や経過時間 (**crond**) の影響を受ける場合がある
- 通常デーモンには制御用端末や標準入出力デバイスはない
- より優れたセキュリティ制御を提供できることがある

デーモンプロセスは、ユーザーにシステムの特定サービスを提供することを唯一の目的とするバックグラウンドプロセスです。例えば **xinetd**、**httpd**、**lpd**、**vsftpd** があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.9 nice 値

nice を使用した優先順位の設定

- 全てのプロセスには **nice** 値が設定され、それによってプロセス実行時の優先度が決定される
- **nice** 値の調整によって、nice がプロセスの優先度を上げたり下げたりできる
- nice 値が大きいほど優先度は低くなる
- nice 値が小さいほど優先度は高くなる
- 非特権ユーザーができるのは nice 値を上げることだけ
- スーパーユーザーだけが nice 値を下げることができる

nice 値の範囲は -20（最少の nice 値、最高の優先順位）から +19（最高の nice 値、最低の優先順位）までです。スーパーユーザーだけが nice 値を下げられますが、一般ユーザーも nice 値をあげることができる点に注意してください。プロセスは生成時にシェルから **nice** 値 **0** を引き継ぎます。

nice の実行方法は次のとおりです。

nice はプロセスを特定の **nice** 値で実行させるために使います。

```
$ nice -n 10 myprog
```

myprog は nice 値 10 で実行されます。

```
$ nice -n 19 myprog
```

myprog は nice 値 19 の最低優先度で実行されます。

```
$ sudo nice -n -20 myprog
```

myprog は nice 値 -20 の最高優先度で実行されます。

nice コマンドで nice 値を指定しない場合はデフォルトで nice 値は 10 加算されます。引数を指定しない場合は、現在の nice 値が報告されます。

nice 値を上げてもそのプロセスが実行されないわけではありません。競合するものが他に何もない場合は全ての CPU 時間を取得することもあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

nice 値の変更

- **renice** を使って実行中のプロセスの nice 値の変更ができる

```
$ renice --help
Usage:
 renice [-n] <priority> [-p|--pid] <pid>...
 renice [-n] <priority>  -g|--pgrp <pgid>...
 renice [-n] <priority>  -u|--user <user>...
```

- pid 20003 の nice 値を 5 にあげる

```
$ renice +5 -p 20003
```

- スーパーユーザーだけがプロセスの nice 値を下げることができる

renice を使うことで既に実行中のプロセスの nice 値の上げ下げができます。これによって稼働中に nice 値の変更ができるようになっています。

デフォルトではスーパーユーザーだけが nice 値を下げる（すなわち優先度を上げる）ことができます。しかし、通常ユーザーもあらかじめ `/etc/security/limits.conf` で設定された範囲内であれば自分のプロセスの nice 値を下げることができます。

非特権ユーザーが nice 値を上げた場合、それを元の値に下げることが出来るのはスーパーユーザーだけです。

実行中のプロセスの nice 値変更は **renice** コマンドの利用が簡単です。

```
$ renice +3 13848
```

このコマンドで **pid = 13848** の nice 値は 3 に再設定されます。1 つ以上のプロセスの値を同時に変更することや、その他のオプションもあります。詳細については **man renice** を参照してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

14.10 演習

デモ教材ビデオ

using_renice_demo.mp4

課題 14.1: ulimit でプロセスを制御する

以下を実行してください:

```
$ help ulimit
```

次に行く前に、/etc/security/limits.conf を読んでください。

1. **bash** コマンドを実行して（または、新しい端末を開いて）、新しいシェルを開始します。以降の変更は新たに開いたシェルだけに有効となります。オープンできるファイル数の制限の現在値とハードとソフトリミットを見ます。
2. 制限値をハードリミット値に設定し、動くことを確認します。
3. ハードリミットを 2048 に設定し、動くことを確認します。
4. リミットを元の値に戻します。うまく動きましたか？

解 14.1

1. ```
   $ bash
   $ ulimit -n
   ```

   ```
   1024
   ```

   ```
   $ ulimit -S -n
   ```

   ```
   1024
   ```

   ```
   $ ulimit -H -n
   ```

   ```
   4096
   ```

2. ```
   $ ulimit -n hard
   $ ulimit -n
   ```

   ```
   4096
   ```

3. ```
   $ ulimit -n 2048
   $ ulimit -n
   ```

   ```
   2048
   ```

4. ```
   $ ulimit -n 4096
   ```

   ```
   bash: ulimit: open files: cannot modify limit: Operation not permitted
   ```

   ```
   $ ulimit -n
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
2048
```

もはやリミット値を元に戻すことは出来ません！

もし別のリミット、たとえばスタックサイズ（-s）を選んだとしたら、ハードリミットは 無制限 なので元に戻すことができます。

課題 14.2: System V の IPC の動作を確認する

System V IPC は **UNIX** の初期からある比較的古い **I**nter **P**rocess **C**ommunication です。３つの機構を持っています。

1. **Shared Memory Segments**
2. **Semaphores**
3. **Message Queues**

POSIX IPC を使いこれら３つの機構を実現する新しいプログラムもありますが、**System V IPC** を利用するアプリケーションも多数残っています。

以下のようにして **System V IPC** の状況の概況をみます。

```
$ ipcs
```

```
------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x01114703 0          root       600        1000       6
0x00000000 98305      coop       600        4194304    2          dest
0x00000000 196610     coop       600        4194304    2          dest
0x00000000 23068675   coop       700        1138176    2          dest
0x00000000 23101444   coop       600        393216     2          dest
0x00000000 23134213   coop       600        524288     2          dest
0x00000000 24051718   coop       600        393216     2          dest
0x00000000 23756807   coop       600        524288     2          dest
0x00000000 24018952   coop       600        67108864   2          dest
0x00000000 23363593   coop       700        95408      2          dest
0x00000000 1441811    coop       600        2097152    2          dest

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
0x00000000 98304      apache     600        1
0x00000000 131073     apache     600        1
0x00000000 163842     apache     600        1
0x00000000 196611     apache     600        1
0x00000000 229380     apache     600        1
```

現在実行中の共有メモリーセグメントのほとんど全てがキー値 0 （IPC_PRIVATE）を持っています。これは、それらが親子関係にあるプロセスで共有されていることを意味します。さらに、ひとつのセグメントを除きアタッチ終了後に破棄されることを意味する (status 欄に dest が表示されている) マークがついています。

生成した共有セグメントに、最後にアタッチされたプロセスに関する情報を集めることができます。

```
$ ipcs -p
```

```
------ Message Queues PIDs --------
msqid      owner      lspid      lrpid

------ Shared Memory Creator/Last-op PIDs --------
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
shmid      owner      cpid       lpid
0          root       1023       1023
98305      coop       2265       18780
196610     coop       2138       18775
23068675   coop       989        1663
23101444   coop       989        1663
23134213   coop       989        1663
24051718   coop       20573      1663
23756807   coop       10735      1663
24018952   coop       17875      1663
23363593   coop       989        1663
1441811    coop       2048       20573
```

この表示結果から cpid と lpid に相当するプロセスを検索するため、つぎを実行します。

```
$ ps aux |grep -e 20573 -e 2048
```

```
coop      2048  5.3  3.7 1922996 305660 ?      Rl   Oct27  77:07 /usr/bin/gnome-shell
coop     20573  1.9  1.7 807944 141688 ?       Sl   09:56   0:01 /usr/lib64/thunderbird/thunderbird
coop     20710  0.0  0.0 112652  2312 pts/0    S+   09:57   0:00 grep --color=auto -e 20573 -e 2048
```

thunderbird は **gnome-shell** が作成した共有セグメントを利用していることがわかります。

これらのステップを実行することで、さまざまなリソースを誰が利用しているかがわかります。たとえば **leaks**（どのプロセスも利用していない共有資源）があるかを見るためには

```
$ ipcs
```

```
....
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
....
0x00000000 622601     coop       600        2097152    2          dest
0x0000001a 13303818   coop       666        8196       0
....
```

アタッチされていなく、また破棄ともマークされていない共有メモリーが表示されています。つまり、これらはこのあと何かのプロセスがアタッチしない限り、メモリーの無駄として残り続けるのです。

課題 14.3: アップタイムとロードアベレージの取得

システムの稼働時間を把握し、ロードアベレージも表示します。

解 14.3

一番簡単な方法は **uptime** ユーティリティを使う方法です

```
$ uptime
```

```
10:26:40 up  3:19,  5 users,  load average: 1.46, 1.40, 1.19
```

２番目の方法は **top** の出力の一行目を見ることです。

```
$ top | head
```

```
top - 10:28:11 up  3:20,  5 users,  load average: 1.93, 1.52, 1.25
Tasks: 313 total,   1 running, 312 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.3 sy,  0.0 ni, 98.2 id,  0.5 wa,  0.0 hi,  0.0 si,  0.0
KiB Mem : 16284472 total,  6556792 free,  1029760 used,  8697920 buff/cache
KiB Swap:  8290300 total,  8290300 free,        0 used. 10364220 avail Mem
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2615 coop      20   0  504536 186312  65488 S   6.7  1.1   6:28.30 skype-b+
18248 coop      20   0  655804  50816  30884 S   6.7  0.3   0:20.11 emacs
    1 root      20   0  204912   6508   3956 S   0.0  0.0   0:00.92 systemd
```

3つ目の方法は **w** の利用です。

```
 10:30:51 up  3:23,  5 users,  load average: 0.55, 1.11, 1.14
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
coop     :0       :0               07:08   ?xdm?  16:51   0.19s gdm-session-
coop     pts/0    :0               07:09    2:22m  0.12s  0.12s bash
coop     pts/1    :0               07:09    1:37m  0.42s  0.42s bash
coop     pts/2    :0               07:09    0.00s 51.09s  0.00s w
coop     pts/3    :0               07:09   27:08   0.25s  0.25s bash
```

課題 14.4: バックグラウンドとフォアグランドジョブ

ターミナルウィンドウからグラフィカルなプログラムを起動し、ウィンドウ内で文字を入力することができなくなるようにします。**gedit** を利用するのが簡単ですが、他のプログラムをつかっても問題ありません。

1. **gedit** を使って新しいファイルを開きます。

   ```
   $ gedit somefile
   ```

   ターミナルウィンドウには文字が入力できなくなりました。

2. マウスポインターをターミナル上においた状態で CTRL-Z と打ちます。

   ```
   ^Z
   [3]+  Stopped                 gedit somefile
   ```

   gedit への文字入力ができなくなりました。

3. **jobs -l** を使って、このターミナウからどのプロセスが起動されたか確認します。

   ```
   $ jobs -l
   [1]  17705 Running                 evince *pdf &
   [2]- 18248 Running                 emacs /tmp/hello.tex &
   [3]+ 19827 Stopped                 gedit somefile
   ```

4. これで最後に起動したジョブ（**gedit 何かのファイル**）がバックグラウンドに移動します。

   ```
   $ bg
   [3]+ gedit somefile &
   ```

   gedit 画面への文字入力が可能になりました。

5. プロセスを再びフォアグランドにもってくる。

   ```
   $ fg
   gedit somefile
   ```

   ターミナルへの文字入力が出来なくなりました。

6. クリーンアップするには、再度プロセスをサスペンドし、**kill** で終了させます。

   ```
   ^Z
   [3]+  Stopped                 gedit somefile
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
$ jobs -l
```

```
[1]  17705 Running                 evince *pdf &
[2]- 18248 Running                 emacs /tmp/hello.tex &
[3]+ 19827 Stopped                 gedit somefile
```

```
$ kill -9 19827
$ jobs -l
```

```
[1]  17705 Running                 evince *pdf &
[2]- 18248 Running                 emacs /tmp/hello.tex &
[3]+ 19827 Killed                  gedit somefile
```

```
$ jobs -l
```

```
[1]- 17705 Running                 evince *pdf &
[2]- 18248 Running                 emacs /tmp/hello.tex &
```

課題 14.5: at を使ってバッチ プロセスの実行を遅らせる

Schedule a very simple task to run at a future time from now. This can be as simple as runningv **ls** or **date** and saving the output. (You can use a time as short as one minute in the future.) 非常にシンプルなタスクを未来の時間に実行するようスケジュールする。**ls** や **date** のようなコマンドを実行して出力を保存するといったシンプルなもので良いでしょう。(1 分後のようなごく近未来の時間を指定することもできます)

これをするには

1. From a short **bash** script.
2. Interactively

解 14.5

1. 以下の内容を含んだ testat.sh ファイルを作成します。

   ```
   #!/bin/bash
   date > /tmp/datestamp
   ```

   そして、実行権限をつけて **at** を付けてキューに登録します。

   ```
   $ chmod +x testat.sh
   $ at now + 1 minute -f testat.sh
   ```

   atq を実行するとジョブがキューに登録されているか確認できます。

   ```
   $ atq
   ```

   ```
   17 Wed Apr 22 08:55:00 2015 a student
   ```

   ジョブが確かに実行されたことを確認します。

   ```
   $ cat /tmp/datestamp
   ```

   ```
   Wed Apr 22 08:55:00 CDT 2015
   ```

   コマンドから >/tmp/datestamp を取り除くとどうなるのでしょうか？(ヒント: メールと入力するよう促されていない場合は、**mail** と入力してください!)

2. インタラクティブに実行する場合も、基本的に同じ手順です。以下のようにジョブをキューに入れるだけです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
$ at now + 1 minute
at> date > /tmp/datestamp
CTRL-D
$ atq
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 14.6: cron を使って周期的に実行するタスクをスケジュールする

毎日午前 10 時にシンプルなタスクを実行するように **cron** を設定する。

解 14.6

以下の内容が書かれた mycrontab というファイルを作成する。

```
0 10 * * * /tmp/myjob.sh
```

そして、以下の内容が書かれた /tmp/myjob.sh というファイルを作成する。

SH /tmp/myjob.sh

```
#!/bin/bash
echo Hello I am running $0 at $(date)
```

実行権限を付与します。

```
$ chmod +x /tmp/myjob.sh
```

Put it in the **crontab** system with:

```
$ crontab mycrontab
```

以下のコマンドでローディングされていることを確認します。

```
$ crontab -l
```

```
0 10 * * * /tmp/myjob.sh
```

```
$ sudo ls -l /var/spool/cron/student
```

```
-rw------- 1 student student 25 Apr 22 09:59 /var/spool/cron/student
```

```
$ sudo cat /var/spool/cron/student
0 10 * * * /tmp/myjob.sh
```

あなたが以下のようなメッセージが印刷されたメールを毎日受信したくなければ

```
Hello I am running /tmp/myjob.sh at Wed Apr 22 10:03:48 CDT 2015
```

以下のようにすれば止めることができます。

```
$ crontab -r
```

もしマシンが指定された日の午前 10 時に不稼働だった場合は、**anacron** はジョブを適切な時間にタスクを実行します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 15 章

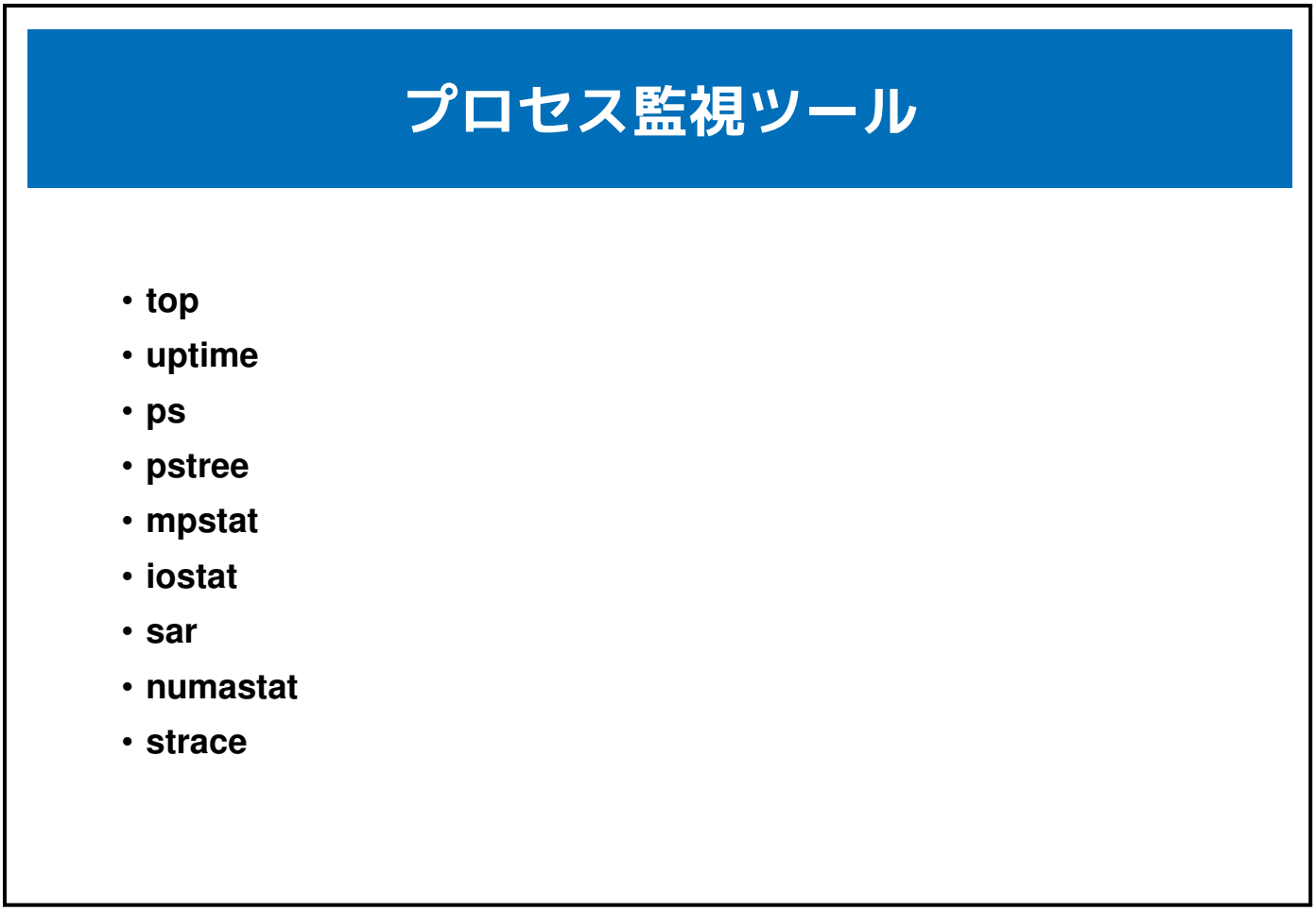
プロセス監視

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- コマンドラインのプロセス監視ツールを特定し、使用する。
- システムトラブルのトラブルシューティングを開始する。
- **ps** を使用して、プロセスに関連する特性や統計情報を表示する方法について説明する。
- **ps** 出力フィールドの違いを識別し、ps 出力をカスタマイズする。
- プロセスの祖先とマルチスレッドアプリケーションの可視化に **pstree** を使用する。
- システム負荷をインタラクティブに表示するに **top** を使用する。

15.1 プロセス監視



Linux 管理者は、プロセス監視に **ps**、**pstree**、**top** などの多くのユーティリティを使用します。これらは全て **UNIX** 系のオペレーティング システムで長く使われてきたものです。

プロセス監視の主要ツールのリストを確認してみましょう。

Table 15.1: **プロセスと負荷の監視ユーティリティ**

ツール	目的
top	動的に更新、プロセスの活動状況の表示
uptime	システムの稼働時間と平均負荷の表示
ps	プロセスに関する詳細情報の表示
pstree	プロセスとその親子関係のツリー表示
mpstat	マルチプロセッサの使用量の表示
iostat	CPU 使用率と I/O 統計情報の表示
sar	システムのアクティビティに関する情報の収集・表示
numastat	NUMA（Non-Uniform Memory Architecture）に関する情報の表示
strace	プロセスが行う全てのシステムコールに関する情報を表示

`/proc` もシステム上のプロセスや他のアイテムの監視に役立ちます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

15.2 トラブル シューティング

トラブル シューティングのレベル

トラブルシューティングの３つのレベル:

- **初心者:** 短期間で習得できるレベル
- **経験者:** 数年の実務経験が必要なレベル
- **達人 (Wizard):** このレベルのスキルは生まれ持った才能と考える人もいるが、それはナンセンスである。どのようなスキルでも、学習すれば習得できる

最高水準で管理されている **Linux** システムでも、問題は発生します。

トラブル シューティング によって、問題がソフトウェア、またはハードウェアに起因するのか、システムローカルなのか、ローカルネットワークまたはインターネット内から発生するのかを特定することができます。

トラブルシューティングを適切に行うには、判断力と経験が必要です。この作業にはある程度は芸術的な要素も含まれるのですが、それでも、適切に系統だった手順に従うことで、将来再利用が可能な問題の原因切り分け手法の確立につながるのです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

基本的なトラブル シューティング テクニック

- 問題を特定する
- 問題を再現させる
- いつでも、簡単にできることから試す
- 原因となる可能性がある要因を、一つずつ排除していく
- 一度に変更するのは１つだけとする、もしそれで解決しなければその変更は元に戻す
- 詳細な情報を取得するために、システムログを確認する (`/var/log/messages` 、`/var/log/secure` など)

（トラブル シュートの進め方の）ルール策定の方針と解析手段は、十分に確立された手順に従うことが求められます、直感に基づいて結論に飛びつくことはお勧めできません。チェックリストと統一された手順を使用する動機は、ウィザード（達人）への依存を回避し、良く知られた手順に忠実に従うだけで最終的にシステム管理者が問題解決できるようにするためです。そうしなければ、もしウィザードが組織を離れてしまった時に、難しい問題の解決に必要なスキルを持った人が誰もいなくなります。

一方、直感をベースに解析を進める時は、生産性を基準にどこまで直感的な方法を続けるかを判定できるデータが、十分な速さで取得できることが担保されるべきです。

直感を無視すると問題の解決に時間がかかる場合もあるので、トラブルシューター（＝解析者）がこれまで直感に頼る方法でどれだけ問題を解決できたかの実績に基づいて、この方法にリソースを投入するか決めるべきでしょう。別の言い方をすると「的を得た直感」は魔法などではなく、経験値の積み重ねなのです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

15.3 ps

ps を使用したプロセス状態の表示

- /proc から情報を収集する
- 何をどう表示するかは、設定次第でフレキシブルに調整可能である
- 代表的なオプションの使用例:

```
$ ps aux
$ ps -elf
$ ps -eL
$ ps -C "bash"
```

ps は、プロセスに関連付けられた特性と統計情報を表示する有用なツールです。これらの情報は全て、プロセスに関連付けられた /proc ディレクトリから収集されます。

このコマンドユーティリティは、全ての **UNIX** 系のオペレーティング システムに異なる形で存在します。その多様性は **Linux** バージョンの **ps** のオプションに現れており、次の３つのカテゴリに分類されます。

1. **UNIX** では前に「-」を付ける必要があり、グループ化ができます。
2. **BSD** では前に「-」を付けることはできません。グループ化はできます。
3. **GNU** の長いオプションは、それぞれの前に「--」が必要です。

オプション付け方の違いは混乱を生みます。ほとんどのシステム管理者は、通常は１つか２つの標準的な組み合わせを使用します。

```
c8:/tmp>ps auxf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           2  0.0  0.0      0     0 ?        S    06:43   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   06:43   0:00  \_ [rcu_gp]
....
coop        3068  0.0  0.2 846052 38012 ?        Ssl  06:53   0:11  \_ /usr/libexec/gnome-terminal-server
coop        3103  0.0  0.0 237040  5648 pts/0    Ss   06:53   0:00  |   \_ bash
coop       51808  0.0  0.0 248280  6272 pts/0    S+   13:41   0:00  |   |   \_ git log
coop       51809  0.0  0.0 219456  2496 pts/0    S+   13:41   0:00  |   |       \_ /usr/bin/less
coop        3158  0.0  0.0 237512  5980 pts/1    Ss+  06:53   0:00  |   \_ bash
c8:/tmp>
```

図 15.1: **BSD オプションでの ps の使用例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ps 出力のカスタマイズ

- -o オプションの後にコンマで区切られたフィールド識別子のリストを指定すると、カスタマイズされた ps フィールドのリストを出力できる
  - pid: プロセス ID 番号
  - uid: プロセスオーナーのユーザー ID 番号
  - cmd: 全ての引数を指定したコマンド
  - cputime: CPU 時間の累積
  - pmem: プロセスのマシン上の物理メモリー利用率、パーセント表示
- その他の多くの出力オプションは **ps** の **man** ページを参照する

```
c8:/tmp>ps -o pid,user,uid,priority,cputime,pmem,size,command
    PID USER       UID PRI     TIME %MEM  SIZE COMMAND
  47619 coop      1000  20 00:00:00  0.0  2656 bash
  49543 coop      1000  20 00:00:00  0.0  1100 ps -o pid,user,uid,priority,cputime,pmem,size,command
c8:/tmp>ps -o pid,user,uid,priority,cputime,pmem,size,command
```

図 15.2: **ps 出力のカスタマイズ**

これ以外にオプション -elf も良く利用されます。

```
c8:/tmp>ps -elf
F S UID          PID    PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root           1       0  0  80   0 - 60929 -      06:45 ?        00:00:02 /usr/lib/systemd/systemd --swit
ched-root --system --deserialize 18
1 S root           2       0  0  80   0 -     0 -      06:45 ?        00:00:00 [kthreadd]
1 I root           3       2  0  60 -20 -     0 -      06:45 ?        00:00:00 [rcu_gp]
....
0 S coop        6302    2365  1  80   0 - 182213 -     06:50 tty2     00:02:33 gnome-system-monitor
0 S coop       47591   37378  0  80   0 - 288155 -     07:55 pts/2    00:00:22 evince LFS301.pdf
0 S coop       47602    2089  0  80   0 - 53719 -      07:55 ?        00:00:00 /usr/libexec/evinced
....
0 S coop       54206    2715  0  80   0 - 58937 -      09:23 pts/4    00:00:00 bash
0 S root       54295    1140  0  80   0 - 54263 -      09:23 ?        00:00:00 sleep 60
4 R coop       54296   54206  0  80   0 - 67123 -      09:23 pts/4    00:00:00 ps -elf
c8:/tmp>
```

図 15.3: **UNIX オプションでの ps の使用例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

15.4 pstree

pstree の使用方法

- ユーザーに、プロセスの親子関係を表示する
- -p でプロセス ID を表示する
- -H [pid] で [pid] とその祖先（分岐元）を表示する

```
$ pstree -aAps 31478
```

```
systemd,1 --switched-root --system --deserialize 21
  `-vmplayer,31478
      |-{vmplayer},31570
      |-{vmplayer},31593
      |-{vmplayer},32572
      `-{vmplayer},32698
```

pstree はプロセスの親子関係と、マルチスレッドアプリケーションについて視覚的に示します。

```
$ pstree -aAp 2408
```

```
bash,2408
|-emacs,24998 pmonitor.tex
|   |-{emacs},25002
|   `-{emacs},25003
|-evince,18036 LFS201-SLIDES.pdf
|   |-{evince},18040
|   |-{evince},18046
|   `-{evince},18047
```

オプションの説明については **pstree** の **man** ページを参照してください。上記は pid=2408 の情報表示を選択しています。
子プロセスの１つ (**evince**、pid=18036) は、子プロセスが３つあることに注目してください。これを別の方法で確認するには:

```
$ ls -l /proc/18036/task
```

```
total 0
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18036
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18040
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18046
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18047
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

15.5 top

top

- CPU 使用率の高いプロセスを表示する目的で利用される
- プロセスは CPU 利用率の高い順に並べられる
- セキュアモード (top s) でない場合はプロセスにシグナルを送信可能
  - k キーを押す
  - プロンプトが出たら PID を入力する
  - プロンプトにシグナル番号を入力する
- 古いユーティリティなので多数のオプションがあり、また多くの対話操作が可能である
- h を押すとキー割り付けを表示する
- 並べ替えオプションなどは使いやすい

"h" でヘルプ表示が出ます。さまざまな条件で、表示を並べ替えることが可能です。ユーザーが (Ctrl-C) で中断するまで 5 秒間隔で更新します。(時間は設定可能です)

```
top - 09:42:28 up  2:56,  1 user,  load average: 1.20, 1.16, 1.09
Tasks: 339 total,   1 running, 338 sleeping,   0 stopped,   0 zombie
%Cpu0  :  1.0 us,  0.3 sy,  0.0 ni, 98.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :  1.3 us,  0.3 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  :  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  :  2.0 us,  0.7 sy,  0.0 ni, 97.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  :  1.7 us,  0.3 sy,  0.0 ni, 98.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  : 17.7 us,  1.3 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  15874.0 total,   3964.8 free,   3177.2 used,   8732.0 buff/cache
MiB Swap:   8096.0 total,   8086.7 free,      9.3 used.  11455.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3605 coop      20   0 3793748 484588 123312 S  18.6   3.0   1:07.44 thunderbird
   2131 coop      20   0 1086964  83868  50788 S   3.3   0.5   8:21.00 Xorg
   2107 coop       9 -11 2729100  20408  13368 S   2.7   0.1   2:10.98 pulseaudio
   6302 coop      20   0  728788  49084  36336 S   1.7   0.3   2:52.34 gnome-system-mo
  44118 coop      20   0 3472692 219620 116372 S   1.3   1.4   0:55.83 spotify
   2365 coop      20   0 4663756 432076  73556 S   1.0   2.7   7:15.12 gnome-shell
   3691 coop      20   0   36.8g 244560  89984 S   1.0   1.5   1:54.33 slack
  50372 coop      20   0 1225676  67700  58684 S   0.7   0.4   0:25.04 slack
    405 root       0 -20       0      0      0 I   0.3   0.0   0:09.96 kworker/u17:2-i915_flip
   2499 root      20   0  414964  31624   9704 S   0.3   0.2   0:08.90 sssd_kcm
  44149 coop      20   0 2183944 226052  78704 S   0.3   1.4   0:23.67 spotify
  56244 coop      20   0  275384   5384   4544 S   0.3   0.0   0:00.21 top
      1 root      20   0  243716  12144   8300 S   0.0   0.1   0:02.18 systemd
```

図 15.4: **top の利用**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/proc の詳細

- /proc ファイルシステムは、カーネルデータ構造へのインターフェイス
- /proc には、各アクティブなプロセスに対応したプロセス ID 番号（PID）名のサブディレクトリが存在する
- /proc/self は、現在実行中のプロセスである
- /proc ディレクトリ内のいくつかのパラメータは変更できる
- 詳細は proc の man ページを参照する

下記のサンプル画面に示した /proc のサブディレクトリには、数字の名前のサブディレクトリが含まれています。サブディレクトリ名の数字は、システム上の全てのアクティブなプロセスの PID を表しています。それぞれのサブディレクトリには、そのプロセスに関する情報が格納されたファイルがあります。

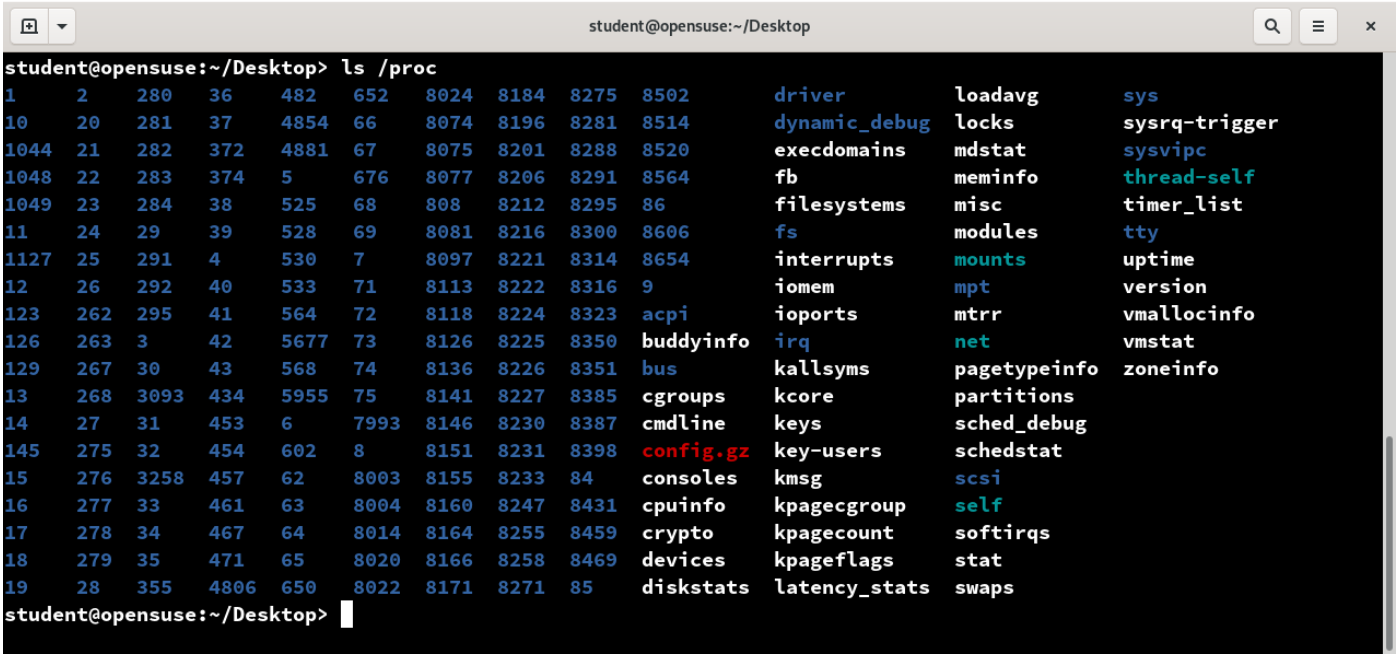


図 15.5: /proc の内容

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

15.6 演習

デモ教材ビデオ

top_demo.mp4
ps_demo.mp4

課題 15.1: プロセス

1. オプション -ef を付けて **ps** を実行してください。その次にオプション aux で実行します。出力に違いに注意してください。
2. **ps** を実行すると、プロセス ID、プライオリティ、nice 値とプロセスのコマンドラインが表示されます。
3. コマンドラインで **bash** と入力し、新しい **bash** のセッションを開始します。nice 値を 10 に指定した **nice** コマンドを使い、もうひとつ別の **bash** セッションを開始します。
4. ステップ 2 と同様に **ps** を実行し、プライオリティと nice 値の違いに注目してください。2 つの **bash** セッションのプロセス ID にも注意してください。
5. **renice** を使ってどちらかの **bash** セッションの nice 値を変更します。もう一度プライオリティと nice 値の変化を監視してください。
6. **top** を実行して出力の変化を見てください。終了するには q を押下してください。

解 15.1

1. ```
   $ ps -ef
   $ ps aux
   ```

2. ```
   $ ps -o pid,pri,ni,cmd
   ```

   ```
     PID PRI  NI CMD
    2389  19   0 bash
   22079  19   0 ps -o pid,pri,ni,cmd
   ```

   （注意：パラメータの間には空白は不要です）

3. ```
   $ bash
   $ nice -n 10 bash
   $ ps -o pid,pri,ni,cmd
   ```

   ```
    2389  19   0 bash
   22115  19   0 bash
   22171   9  10 bash
   22227   9  10 ps -o pid,pri,ni,cmd
   ```

4. ```
   $ renice 15 -p 22171
   $ ps -o pid,pri,ni,cmd
   ```

   ```
     PID PRI  NI CMD
    2389  19   0 bash
   22115  19   0 bash
   22171   4  15 bash
   22246   4  15 ps -o pid,pri,ni,cmd
   ```

5. ```
   $ top
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
   ```

課題 15.2: プロセスの状態を監視する

1. /dev/urandom から読み出し /dev/null へ書き込む **dd** をバックグラウンドで実行します。
2. プロセスの状態をチェックします。どうなっているべきでしょうか？
3. **fg** コマンドで、プロセスをフォアグラウンドに持ってきます。次に Ctrl-Z を入力します。何が起こるでしょうか。プロセスの状態を見てみます。どうですか。
4. **jobs** プログラムを実行します。何が表示されますか。
5. ジョブをフォアグラウンドの持ってきます。次に他のウィンドウから **kill** を使って終了させます。

解 15.2

1.
```
$ dd if=/dev/urandom of=/dev/null &
```

2.
```
$ ps -C dd -o pid,cmd,stat
```

```
  25899 dd if=/dev/urandom of=/dev/ R
```

S か R になっているはずです。

3.
```
$ fg
$ ^Z
$ ps -C dd -o pid,cmd,stat
```

```
    PID CMD                         STAT
  25899 dd if=/dev/urandom of=/dev/ T
```

状態は T になっているはずです。

4. **jobs** コマンドを入力します。出力は、どうなっていますか？

```
$ jobs
```

```
  [1]+  Stopped                 dd if=/dev/urandom of=/dev/null
```

5. ジョブをフォアグラウンドに戻します。他の端末から **kill** コマンドで、終了させます。

```
$ fg
$ kill  25899
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 16 章

メモリーの監視と利用、swap の設定

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- メモリーチューニングに関わる主な（相互に関連する）検討項目と操作を列挙する。
- `proc/sys/vm` のエントリーを使用して `/proc/meminfo` を解読する。
- **vmstat** でメモリー、ページング、I/O、プロセッサ動作、およびプロセスのメモリー消費の情報を表示する。
- **swap** がどう利用されるか、スワップエリアの作り方、監視、規制する方法を理解する。
- **OOM-killer** がいつ発動するか、メモリー開放のために終了させるプロセスをどうやって選択するか理解する。

16.1 メモリー監視とチューニング

メモリー監視

Linux における、メモリー監視とチューニングを担う重要なツールである

Table 16.1: **メモリー監視ユーティリティ**

ユーティリティ	目的	パッケージ
free	メモリー使用量の概要の表示	procps
vmstat	動的に更新される仮想メモリー統計とブロック I/O の詳細情報	procps
pmap	プロセスのメモリーマップの表示	procps

```
$ free -m
```

```
              total        used        free      shared  buff/cache   available
Mem:          15901        2080        9317         992        4502       12457
Swap:          8095           0        8095
```

時代の進展に伴ってシステムのメモリーリソース要求は肥大化しましたが、同時に RAM の価格も下がったため、結果的にはパフォーマンスは向上しました。

それでも、メモリー不足がシステム全体のパフォーマンスとスループットのボトルネックとなることはよくあります。
CPU と I/O サブシステムが、メモリーからのデータの取得や、メモリーへのデータの書き込みで待たされることがあるからです。

メモリーに関連したシステムの挙動監視、デバッグ、調整ツールは多数あります。

メモリーサブシステムのチューニングは複雑な作業です。まずメモリー使用量と I/O スループットが本質的に関連していることを理解する必要があります。大部分のメモリーは、ディスク上のファイルのコンテンツをキャッシュするために使用されます。

したがって、メモリーパラメータの変更が、I/O パフォーマンスに大きな影響を与える可能性があります。
逆に I/O パラメータを変更した結果、仮想メモリーサブシステムの効果が大きく影響を受ける可能性もあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/proc/meminfo

```
$ cat /proc/meminfo
```

```
MemTotal:       16254132 kB     Writeback:             0 kB     VmallocChunk:          0 kB
MemFree:          162652 kB     AnonPages:       2780004 kB     Percpu:             8352 kB
MemAvailable:   12234416 kB     Mapped:           679632 kB     HardwareCorrupted:     0 kB
Buffers:          847972 kB     Shmem:            523880 kB     AnonHugePages:    958464 kB
Cached:         10959104 kB     KReclaimable:    1124064 kB     ShmemHugePages:        0 kB
SwapCached:        13276 kB     Slab:            1233300 kB     ShmemPmdMapped:        0 kB
Active:          3252848 kB     SReclaimable:    1124064 kB     FileHugePages:         0 kB
Inactive:       11343484 kB     SUnreclaim:       109236 kB     FilePmdMapped:         0 kB
Active(anon):     393416 kB     KernelStack:       16512 kB     HugePages_Total:       0
Inactive(anon):  2919760 kB     PageTables:        79092 kB     HugePages_Free:        0
Active(file):    2859432 kB     NFS_Unstable:          0 kB     HugePages_Rsvd:        0
Inactive(file):  8423724 kB     Bounce:                0 kB     HugePages_Surp:        0
Unevictable:       75032 kB     WritebackTmp:          0 kB     Hugepagesize:       2048 kB

Mlocked:              32 kB     CommitLimit:    16417364 kB     Hugetlb:               0 kB
SwapTotal:       8290300 kB     Committed_AS:   10351404 kB     DirectMap4k:      497740 kB
SwapFree:        7911980 kB     VmallocTotal:   34359738367 kB  DirectMap2M:    16152576 kB
Dirty:                60 kB     VmallocUsed:       28256 kB     DirectMap1G:     1048576 kB
```

疑似ファイル /proc/meminfo には、どのようにメモリーが使用されているかについての数多くの情報が含まれます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION | Training & Certification

16.2 /proc/sys/vm

/proc/sys/vm

- 仮想メモリーシステムの動作を設定するための、多くの「調整つまみ」が含まれる
- ほとんど全ての項目は（root によって）、書き込みが可能である
- このディレクトリに具体的にどんな項目が見えるかは、カーネルバージョンなどに依存する
- 設定値は、直接書き込むことも **sysctl** 利用して変更することも可能

`/proc/sys/vm` のパラメータを調整するときのベストプラクティスは、一度には１項目だけを調整して効果を確認することです。重要な（相互に関連する）タスクは次のとおりです。

- フラッシングパラメータ制御; つまり、ダーティにできるページの数とそれをディスクに書き出す頻度の設定です。
- スワップ挙動の制御; ファイルコンテンツをどれだけオンメモリーに残すか、逆に領域不足時にスワップアウトさせるかの設定です。
- 許容メモリーオーバーコミット量の制御。多くのプログラムは、コピーオンライト（COW）により要求メモリーの全ては使用しません。

メモリーのチューニングは繊細です。特定のシステム状況／負荷下では有効な設定でも、他では全く効果が無いこともあります。

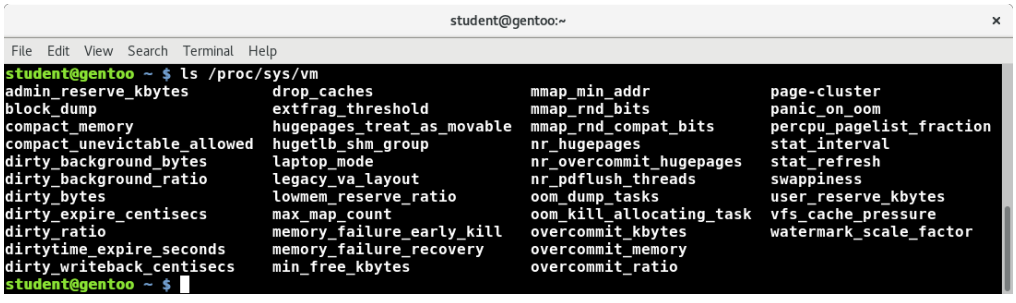


図 16.1: **/proc/sys/vm**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

16.3 vmstat

vmstat

- メモリー、ページング、I/O、プロセッサアクティビティ、プロセスに関する情報を表示する
- 多数のオプションがある

```
$ vmstat [options] [delay] [count]
```

- [delay] は秒単位で指定、count 回結果がレポートされる
- 最初の行は、再起動以降の平均値である
- ２行目以降は、指定間隔でのアクティビティを表示する

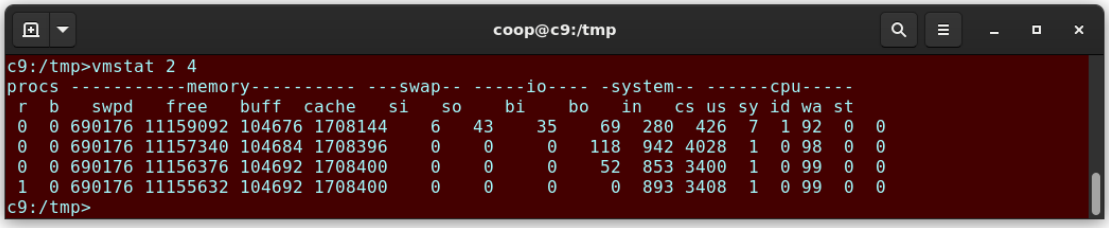


図 16.2: **vmstat**

オプション -S m が指定された場合、メモリー統計の単位は KB ではなく MB になります。

-a オプションを使用すると **vmstat** はアクティブおよび非アクティブなメモリーに関する情報を表示します。アクティブページは最近使用されたページです。ページはクリーン（ディスクの内容が最新）またはダーティ（最終的にディスクに書き出す必要がある）のどちらかです。対照的に、非アクティブページは最近使用されていないのでクリーンである可能性が高く、メモリーに負荷がかかるとすぐに解放されます。

メモリーは、アクティブリストと非アクティブリストの間を行ったり来たりします。これは、新しく参照されたり、使用と使用の間隔が長くなったりするためです。

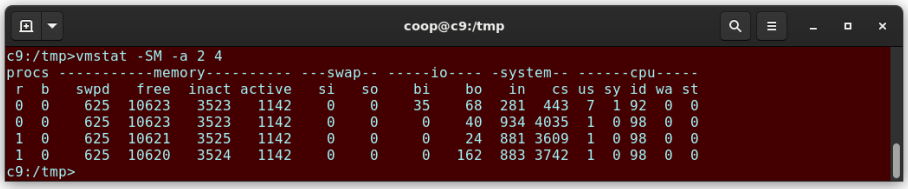


図 16.3: **vmstat の使用例**

一つのパーティションの簡単な統計を取りたいなら -p が利用できます。

図 16.4: **1 つのディスクに対して vmstat を使用する例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

16.4 スワップ

スワップの使用

- ディスク上に置かれた、仮想的なメモリー領域である
- **fdisk** でスワップ パーティションを作成する
- **mkswap**: スワップ パーティション、またはファイルをフォーマット
- **swapon**: スワップ パーティション、またはファイルを有効化する
- **swapoff**: スワップ パーティション、またはファイルを無効化する
- (`/etc/fstab` を使って) 起動時にマウント可能である
- １つ以上のスワップ パーティションを設定できる

```
$ cat /proc/swaps
```

```
Filename                Type            Size    Used    Priority
/dev/sda6               partition       8290300 0       -1
/tmp/swpfile            file            102396  0       -2
```

Linux は **仮想メモリー**システムを採用しており、オペレーティング システムは、実際よりも多くのメモリーがあるかのように機能します。この種の **メモリーのオーバーコミッション（＝投機的な割り当て）**は２つの方法で実現されています。

- 多くのプログラムは、使用すると宣言したメモリーを実際に全て使用するわけではありません。それは、子プロセスが **COW** (**C**opy **O**n **W**rite) 手法を使用して、親のメモリー領域のコピーを継承するためです。子プロセスは、変更があったメモリー領域だけ（ページ毎に）固有のコピーを作成します。
- メモリーの負荷が高くなってくると、アクティブでないメモリー領域はディスクに **スワップアウト** され、再び必要になったときにだけ、読み戻されます。

このようなスワッピングは、通常１つ以上の専用パーティションまたはファイルに対して行われます。**Linux** では複数の **スワップ領域** が認められているため、スワップの要求を動的に調整できます。各スワップ領域には優先順位があり、優先順位の高い領域がいっぱいになるまで優先順位の低い領域は使用されません。

多くの場合、推奨されるスワップサイズはシステムの **RAM** サイズです。`cat /proc/swaps` で現在システムが何をスワップに使用しているかを確認することが可能で、`free` でシステムが現在使用しているスワップの量を確認できます。

いつでも大部分のメモリーは、必要以上のディスクへアクセスや、最善とは言えない順序やタイミングでのディスクアクセスを防ぐための、ファイルコンテンツのキャッシュ用途に使用されています。そのようなメモリーページのデータは、一時的なキャッシュデータが書かれているだけなので（＝バックアップ記憶なので)、スワップに書き出す意味は無く、決してスワップアウトされません。代わりに **ダーティページ** (メモリー上のデータが元々のディスク上の値から更新されている) は、ディスクに書き戻されます。

Linux では、アプリケーションメモリーとは対照的に、カーネル自体が使用するメモリーは **決してスワップアウトされない** ことにも触れておくべきでしょう。この挙動は他のオペレーティング システムとの違いです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

16.5 Out of Memory Killer (OOM)

OOM Killer

- システムの物理メモリーは、枯渇する可能性がある。この時
  1. メモリーが開放されるまで、新たなメモリー要求を拒否する
  2. **swap** 領域を使って、物理メモリーサイズを拡大する
  3. メモリー消費を削減するために、インテリジェントにプロセスを停止させ、システムを延命させる
- **Linux** は、通常２番目、ないし３番目の手段を採用する
- **OOM** killer はどのプロセスを停止するかを選択するメカニズムである

メモリーの負荷に対処する最も簡単な方法は、空きメモリーがある限りメモリー割り当てを成功させ、全てのメモリーを使い切ったら失敗させることです。

２番目の方法は、ディスク上のスワップ領域を使用して、常駐メモリーの一部を書き出すことです。この場合（少なくとも理論的には）、使用可能なメモリーサイズの合計は実際の RAM のサイズにスワップ領域のサイズを加えたものになります。この戦略では、負荷がかかった時にどのメモリーのページをスワップアウトするかの決定方法が課題となります。このアプローチでは、もしスワップ領域自体が一杯になったら、以降のメモリー要求は必ず失敗させる必要があります。

しかし Linux はもう少し賢く動きます。Linux はシステムにメモリーのオーバーコミットを許すのです。そのため RAM とスワップの合計サイズを超えるメモリー要求であっても許可するのです。これは無謀に思えるかもしれませんが（全てではないにしても）多くのプロセスは要求したメモリーの全ては使用しないのです。

例としては 1MB のバッファを確保するが、実際には数ページ分のメモリーだけを使用するプログラムがあります。また別の例として、子プロセスは fork されるたびに親のメモリー空間全体のコピーを受け取ります。Linux は COW（copy on write）を採用しているので、プロセスが実際にメモリーを変更するまで実際のコピーは発生しないのです。ただし、カーネルはコピーが発生する可能性は想定しておかなければないません。

このように、カーネルはメモリーのオーバーコミットを許しますが、これはユーザープロセス用のページに対してだけです。カーネルが使用するページは、スワップさせることはできないので、要求時に常に割り当てられます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

OOM Killer のアルゴリズム

- (OOM killer が採用する) 推論アルゴリズムは、通常は動作には影響を与えないように考えられている。これより穏便な停止、または削減の方法はあるだろうか？
- 各プロセスの `/proc/[pid]/oom_score` ファイルに格納された **badness** に基づいて、どのプロセスを停止させるか選択される
- 同ディレクトリにあるプロセスの `oom_adj_score` を利用して、各タスクの調整が可能である

`/proc/sys/vm/overcommit_memory` の値を設定により、オーバーコミット動作を変更したり無効にすることもできます。

- 0:（デフォルト設定）オーバーコミットを許可しますが、明らかなオーバーコミットは拒否し root ユーザーには一般のユーザーよりもいくらか多くのメモリーを割り当てます。
- 1: 全てのメモリーリクエストのオーバーコミットを認めます。
- 2: オーバーコミットをオフにします。メモリーコミットの合計が、スワップ領域のサイズと構成変更可能なパーセンテージ（デフォルトでは 50）の RAM のサイズとの合計に達すると、メモリー要求は失敗します。このパーセンテージは `/proc/sys/vm/overcommit_ratio` で変更できます。

Andries Brouwer がこれについての面白い見解を示しました。(https://lwn.net/Articles/104185/):

> 「航空機会社は、より少ない燃料で飛行機を飛ばす方が格安であることを発見しました。飛行機はより軽くなり、燃料の使用はより少なくなり、お金の節約が実現しました。しかし、まれなケースですが燃料の量が不足し飛行機が墜落することがありました。この問題は、特別な OOF（out-of-fuel：燃料切れ）メカニズムを開発した会社のエンジニアによって解決されました。緊急の場合、乗客が一人選択され飛行機から放り出されました。（必要に応じてこの手順が繰り返されました）多数の理論が開発され、放り出される犠牲者の適切な選択方法に関して多くの発表がありました。犠牲者はランダムに選ばれるべきか？それとも最も体重が重い人を選ぶべきか？それとも年長者を選ぶべきか？乗客は追い出されないようにお金を支払うべきだがそうすると犠牲者は機内で最も貧しい人になるのか？さらに例えば、最も重い人が選ばれた場合それがパイロットだったら特別な例外があるべきか？ファーストクラスの乗客は免除されるべきか？現在も OOF メカニズムは存在しており、時々稼働しています。そして燃料不足がなくても乗客を放り出します。エンジニアは、今もこの誤動作がどのように引き起こされるかを研究しています。」

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

16.6 演習

課題 16.1: スワップ領域の管理

現在のスワップ領域を確認します。

```
$ cat /proc/swaps
```

```
Filename                                Type            Size    Used    Priority
/dev/sda11                              partition       4193776 0       -1
```

新パーティションかファイルをスワップ領域として追加します。ファイルの場合:

```
$ dd if=/dev/zero of=swpfile bs=1M count=1024
```

```
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 1.30576 s, 822 MB/s
```

```
$ mkswap swpfile
```

```
Setting up swapspace version 1, size = 1048572 KiB
no label, UUID=85bb62e5-84b0-4fdd-848b-4f8a289f0c4c
```

(実パーティションの場合 **mkswap** のオプションとしてパーティション名を与えますが、その場合、パーティションの内容はすべて消去されますので、注意しましょう !)

新スワップ領域をアクティブにします。

```
$ sudo swapon swpfile
```

```
swapon: /tmp/swpfile: insecure permissions 0664, 0600 suggested.
swapon: /tmp/swpfile: insecure file owner 500, 0 (root) suggested.
```

RedHat、Centos、Fedora では

RHEL はセキュアでないと警告を出します。次のように対応します:

```
$ sudo chown root:root swpfile
$ sudo chmod 600 swpfile
```

`swpfile` が使用されていることを確認します。

```
$ cat /proc/swaps
```

```
Filename                                Type            Size    Used    Priority
/dev/sda11                              partition       4193776 0       -1
/tmp/swpfile                            file            1048572 0       -2
```

`Priority` に注目してください; 高プライオリティのスワップ領域が満杯になるまで、低プライオリティのスワップ パーティションもしくはファイルは使用されません。

領域を確保するために、スワップファイルを無効にして削除します。

```
$ sudo swapoff swpfile
$ sudo rm swpfile
```

課題 16.2: OOM キラーの発動

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

- **Linux** カーネルが極度のメモリー不足に陥ると、恐ろしい **OOM** (**O**ut **O**f **M**emory) **K**iller が起動します。これは、システムを穏便に回復させるために、どのプロセスを殺すのが「最善」かを選択するものです。
- システムを強制的にメモリー不足状態にして、何が起こるかを観察します。ターミナルを開いて以下のように入力します。

```
$ sudo tail -f /var/log/messages
```

こうすると、kernel から出されたメッセージを見ることができます。

- もっと良い方法は以下のようにすることです。

```
$ dmesg -w
```

こうすれば kernel 以外のメッセージを表示させないようにできます。

- 以下のコマンドですべてのスワップをオフにすると、より簡単にこの演習を実行できるようになります。

```
$ sudo /sbin/swapoff -a
```

後で設定を元に戻すのを忘れずに

```
$ sudo /sbin/swapon -a
```

- さて、これからシステムにメモリーの負荷をかけようと思います。方法はご自由ですが、メモリーを消費するためのプログラムも用意しています。

lab_wastemem.c

```c
/* simple program to defragment memory, J. Cooperstein 2/04
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/sysinfo.h>
#include <signal.h>

#define MB (1024*1024)
#define BS 16                          /* will allocate BS*MB at each step */
#define CHUNK (MB*BS)
#define QUIT_TIME 20
void quit_on_timeout(int sig)
{
        printf("\n\nTime expired, quitting\n");
        exit(EXIT_SUCCESS);
}

int main(int argc, char **argv)
{
        struct sysinfo si;
        int j, m;
        char *c;

        /* get total memory on the system */
        sysinfo(&si);
        m = si.totalram / MB;
        printf("Total System Memory in MB = %d MB\n", m);
        m = (9 * m) / 10;         /* drop 10 percent */
        printf("Using somewhat less: %d MB\n", m);

```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```c
        if (argc == 2) {
                m = atoi(argv[1]);
                printf("Choosing instead mem = %d MB\n", m);
        }

        signal(SIGALRM, quit_on_timeout);
        printf("Will quite in QUIT_TIME seconds if no normal termination\n");
        alarm(QUIT_TIME);

        for (j = 0; j <= m; j += BS) {
                /* yes we know this is a memory leak, no free,
                 * that's the idea!
                 */
                c = malloc(CHUNK);
                /* just fill the block with j over and over */
                memset(c, j, CHUNK);
                printf("%8d", j);
                fflush(stdout);
        }
        printf("\n\n    Sleeping for 5 seconds\n");
        sleep(5);
        printf("\n\n    Quitting and releasing memory\n");
        exit(EXIT_SUCCESS);
}
```

引数に何 MB メモリーを消費するか指定できます。繰り返し実行して徐々にメモリー消費量を増やしていくと、最終的にはシステムがメモリーを使い果たしてしまいます。

注目

以下のようにプログラムをコンパイルして、実行することができます。

```
$ gcc -o lab_wastemem lab_wastemem.c
$ ./lab_wastemem 4096
```

実行するとメモリーを 4GB を消費します。実行中に **gnome-system-monitor** や、他のメモリー監視プログラムを実行してみるとよいでしょう。(しばらく表示が固まるかもしれませんが !)

- **OOM** (Out of Memory) killer が発動して、システムを停止させないためにプロセスを殺そうとする（強制停止させる）様子を見ることができるでしょう。どのプロセスが最初に停止されられるでしょうか？

解 16.2

lab_wastemem.c
lab_waste.sh

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 17 章

I/O の監視とチューニング

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- I/O アクティビティ監視の重要性と、それがシステムパフォーマンスのボトルネックになる可能性を理解する。
- **iostat** を使用して、システムの I/O デバイスのアクティビティを監視する。
- **iotop** を使って現在の I/O 使用状況の変化を連続的に表示する。

17.1 I/O 監視

I/O の監視とディスクのボトルネック

- 入力／出力 (I/O) の挙動監視は、ピーク性能確保のための基本となる
- システムは I/O バウンド（i/o-bound）、CPU バウンド（CPU bound）、メモリーバウンド（memory bound）に分類される;
  測定してみないと、どのタイプに該当するかは判定できない
- いくつかの重要なツール:
  - **iostat**
  - **iotop**

ディスクのパフォーマンス問題は、メモリー不足や不適切なネットワークハードウェア、チューニング不足などと強く結びついている可能性があります。問題の切り分けは困難です。

CPU が I/O の完了待ちでアイドル状態にあるとか、ネットワークがバッファクリアを待っているような場合、システムは I/O バウンド（i/o-bound）であると呼びます。

極端に I/O が遅すぎるのをメモリー不足と見誤るケースがあります。もし読み書きに使用されているメモリーバッファが溢れると、それはメモリー不足に見えるでしょう。実際には、バッファーへの読み込みや書き出しに時間がかかり過ぎなのが根本原因です。同様に、ネットワーク転送でも I/O に時間がかかり過ぎだと、ネットワークのスループットが犠牲になります。

リアルタイム監視とトレースは、ディスクのボトルネックを特定して軽減するために欠かせないツールです。ただし、発生頻度の低い問題や再現性の低い問題は、解決が困難です。

I/O チューニングは、関連する調整項目が多く複雑です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

17.2 iostat

iostat

- 全ての I/O アクティビティを監視する基本的なツールである
- オプションを使って様々なレポートを出力できる

```
$ iostat [OPTIONS]  [devices] [interval] [count]
```

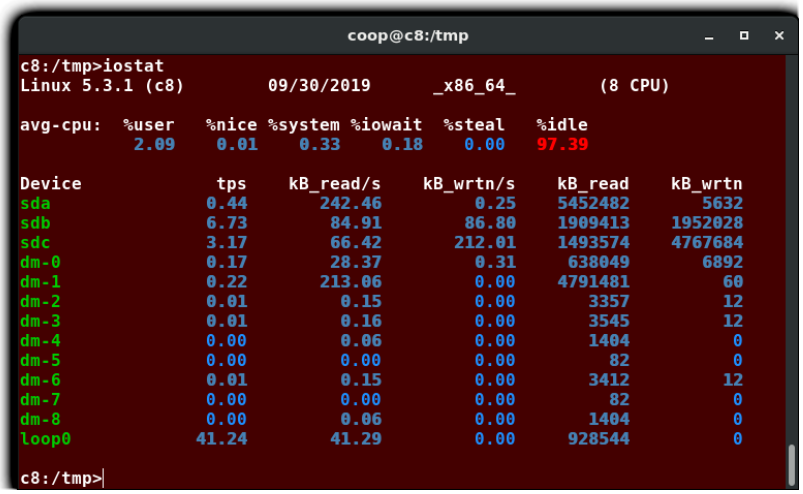


図 17.1: **iostat**

iostat は、システム上の I/O デバイスのアクティビティを監視するための基本的なツールです。詳細なコンテンツをオプションで制御して、多くの情報を含むレポートを作成できます。

CPU 使用率の簡単な説明の後、I/O の統計情報を示します。tps（1 秒あたりの I/O トランザクション。いくつかの論理的なリクエストを 1 つのリクエストにまとめることができます）、ブロックの単位時間ごとの読み書き数、このブロックはほとんどの場合 512 バイトのセクターです。そして、読み取りと書き込みの合計ブロック数です。

情報は、ディスク パーティション（と **LVM** が使われている場合は、デバイスマッパーの論理パーティション）に分解されます。

更に詳しいレポートを示します:

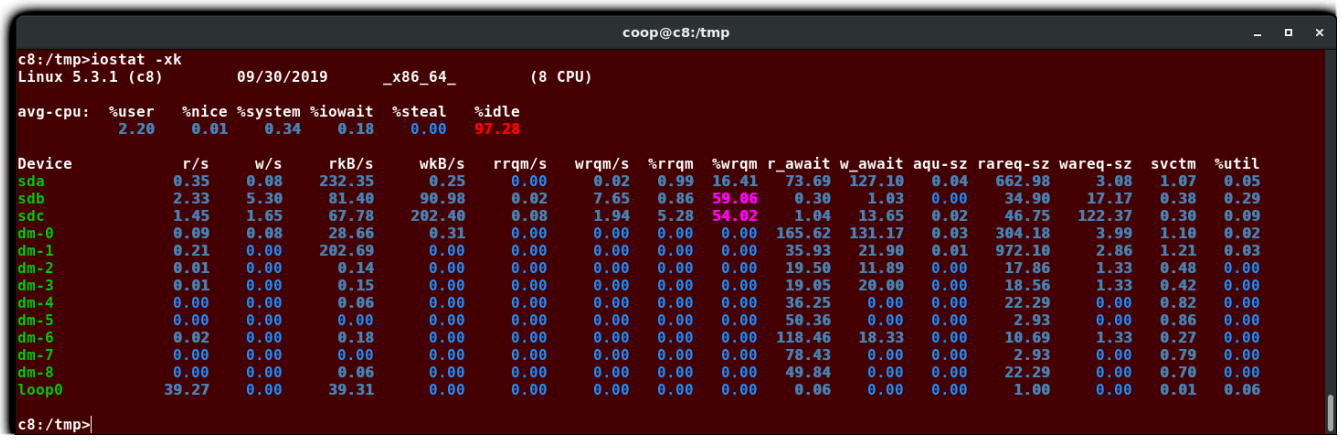


図 17.2: **iostat の例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

17.3 iotop

iotop

- **iotop** は、現在の I/O アクティビティを表示する
- 表示は **top** のように定期的に更新される
- -o オプションを使用すると I/O 処理を実施しているプロセス／スレッドだけが表示されるので、混乱を避けられる

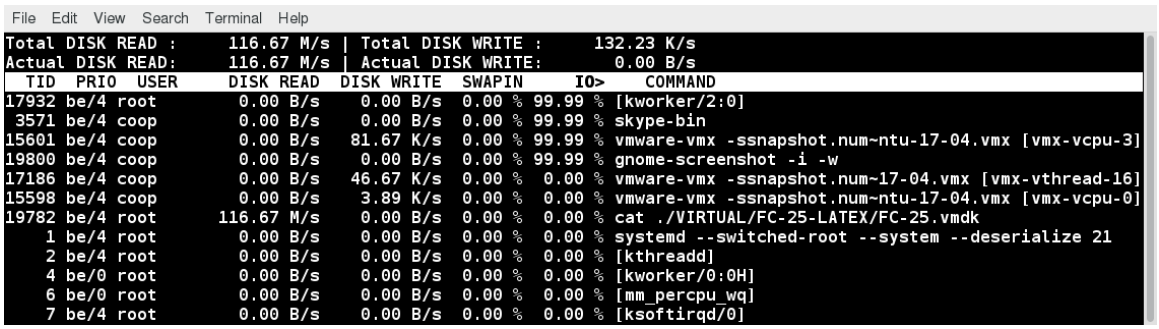


図 17.3: **iotop の例**

もう 1 つの非常に便利なユーティリティは **iotop** です。これは root 権限で実行する必要があります。現在の I/O 使用量を表示し、表示は定期的に更新されます。

PRIO フィールド（プライオリティ）の **be** は **ベストエフォート**、**rt** は **リアルタイム** を表しています。

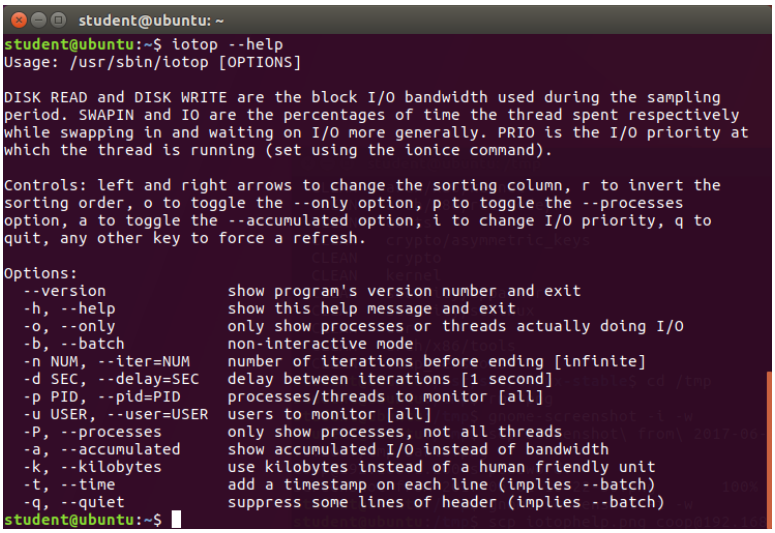


図 17.4: **iotop オプション**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

17.4 演習

課題 17.1: bonnie++

bonnie++ は、ドライブとファイルシステムのテストと性能測定に広く利用されているベンチマークプログラムです。初期の実装である **bonnie** の後継です。

結果を端末に表示したりファイルに出力できる他、**csv**（**c**omma **s**eparated **v**alue の略）フォーマットにも格納できます。一緒に使うプログラムとして **bon_csv2html** と **bon_csv2txt** があり、html やプレーンテキスト形式への変換に利用されます。

使用を開始する前に **bonnie++** の **man** を読んで、性能テストに関するオプションや、それがどのくらいシステムを消費しストレスをかけるかを理解してください。

```
$ bonnie++ --help
```

```
bonnie++: invalid option -- '-'
usage:
bonnie++ [-d scratch-dir] [-c concurrency] [-s size(MiB)[:chunk-size(b)]]
      [-n number-to-stat[:max-size[:min-size][:num-directories[:chunk-size]]]]
      [-m machine-name] [-r ram-size-in-MiB]
      [-x number-of-tests] [-u uid-to-use:gid-to-use] [-g gid-to-use]
      [-q] [-f] [-b] [-p processes | -y] [-z seed | -Z random-file]
      [-D]

Version: 1.98
```

簡単なテストは次のコマンドで実行できます。

```
$  time sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

それぞれ

- `-n 0` ファイルサイズとして 0 を指定すると、ファイルを作成しません。
- `-u 0` ユーザー id を指定します。この場合は root 権限で実行します。
- `-r 100` RAM サイズの指定。100MB の RAM しかないと見せかけます。
- `-f` キャラクタ入出力テストを省略します。
- `-b` 書き込みのたびに fsync を実行することを意味します。これにより、キャッシュへの書き込みを行わず、強制的にディスクへ書き出します。
- `-d /mnt` 一時ファイルを格納するディレクトリを指定します;　十分な領域（ここでは 300MB）を確保してください。

メモリー サイズを指定しなかった場合、プログラムは自動的にシステムのメモリーサイズを検出し、その 2〜3 倍の大きさのテスト用ファイルを作成します。それを行った場合には時間がかかりすぎるため、今回はそのようなことはしません。

On Red Hat

RHEL/CentOS 9 では、現在利用可能なパッケージはありませんが、以前のバージョンは動作し、**EPEL 8** のリポジトリから入手できます。https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/Packages/b/bonnie++-1.98-1.el8.x86_64.rpm 以下のようにインストールします。

```
$ sudo dnf localinstall bonnie++-*rpm
```

RHEL/CentOS 8 システムの場合

```
c8:/tmp>sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
Using uid:0, gid:0.
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Version  1.98       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Name:Size etc        /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
c8             300M            271m  20  288m  15           +++++ +++  3916  22
Latency                          30us      20us                12us   18075us

1.98,1.98,c8,1,1616174245,300M,,8192,5,,,277062,20,294543,15,,,+++++,+++,3916,22,,,,,,,,,,,,,,,,,,,\
30us,20us,,12us,18075us,,,,,,
```

On Ubuntu

同じ物理マシン上の **VMware** ハイパーバイザー下で仮想マシンとして動作する **Ubuntu 20.04** システムの場合:

```
$ sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

```
Using uid:0, gid:0.
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Version  1.98       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Name:Size etc        /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
ubuntu         300M            287m  88 +++++ +++           +++++ +++ +++++ +++
Latency                        3805us    4650us              2136us    1852us

1.98,1.98,ubuntu,1,1616156268,300M,,8192,5,,,293955,88,+++++,+++,,,+++++,+++,+++++,+++,,,,,,,,,,,,,,,\
,,,,3805us,4650us,,2136us,1852us,,,,,,
```

明らかに性能低下が見られます。

bonnie++.outファイルに先の結果を書き込み、html 形式に変換します。

```
$ bon_csv2html < bonnie++.out > bonnie++.html
```

もしくは、平文で良ければ:

```
$ bon_csv2txt < bonnie++.out > bonnie++.txt
```

ドキュメントを読んで、もっと時間がかかり大きく大変なテストに挑戦してください。ここでやらなかったテストを実行してみてください。システムが問題なく動作したらその結果を残しておき、将来システムの調子が悪くなったときに比較してください。

課題 17.2: fs_mark

fs_mark ベンチマークは、ファイルシステムに低レベルのストレスをかけます。複数のディレクトリとドライブ間で相互にまたがる重い非同期 I/O を実行したりします。Rick Wheeler が開発した少し古く、テストに時間がかかるプログラムです。

http://sourceforge.net/projects/fsmark/ からダウンロードできます。tar ボールを入手したら、解凍してコンパイルします。

```
$ tar zxvf fs_mark-3.3.tgz
$ cd fs_mark
$ make
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ここでは簡単に説明をしていますので、詳細は README ファイルを読んでください。

もしコンパイルが失敗したら

```
$ make
```

```
....
/usr/bin/ld: cannot find -lc
```

glibc の**スタティック・バージョン**をインストールしていないためです。**Red Hat** ベース システムの場合、次のようにしてインストールします。

```
$ sudo dnf install glibc-static
```

SUSE 関連のシステムの場合

```
$ sudo zypper install glibc-devel-static
```

Debian ベース システムの場合、静的ライブラリは共有ライブラリと一緒にインストールされるため、追加でインストールする必要はありません。

Red Hat, Centos, Fedora では

- **RHEL 8** では glibc-static は codeready-builder repo に存在するので利用可能にする必要があるかもしれません。
- しかし、**CentOS 8** ではこれは **PowerTools** repo に含まれています。あなたのシステムですでに /etc/yum.repos.d/CentOS-PowerTools.repo が存在するのであれば、そのファイルにenabled=1 が存在するか確認してください。そうでない場合、このファイルのコピーを用意してあります、このコースの SOLUTIONS セクションにあります。

テスト用に大きさ 10KB のファイルを 2500 個作成します。それぞれ書き込み後、ディスクへフラッシュするため **fsync** を実行します。これらを /tmp ディレクトリ下で行います。

```
$ fs_mark -d /tmp -n 2500 -s 10240
```

実行中に、別端末で **iostat** の拡張情報を収集します。

```
$ iostat -x -d /dev/sda 2 10
```

fs_mark がレポートした秒ごとのファイル数と、**iostat** がレポートしたバンド幅の使用率に注意してください。これが 100%に近いと I/O が限界です。(I/O バウンド)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 18 章

コンテナの概要

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- コンテナを定義する基本的なパラメータとメソッドを理解する。
- アプリケーションを仮想化することの意味を知る。
- コンテナと仮想マシンの違いを理解する。
- 小さなユーティリティ プログラムをシステムに追加する詳細な例を使って **Docker** の使い方を学ぶ。
- **再現可能なビルド (Reproducible Builds)** の入門として、バージョン仕様の問題点を分析する。

18.1 コンテナ

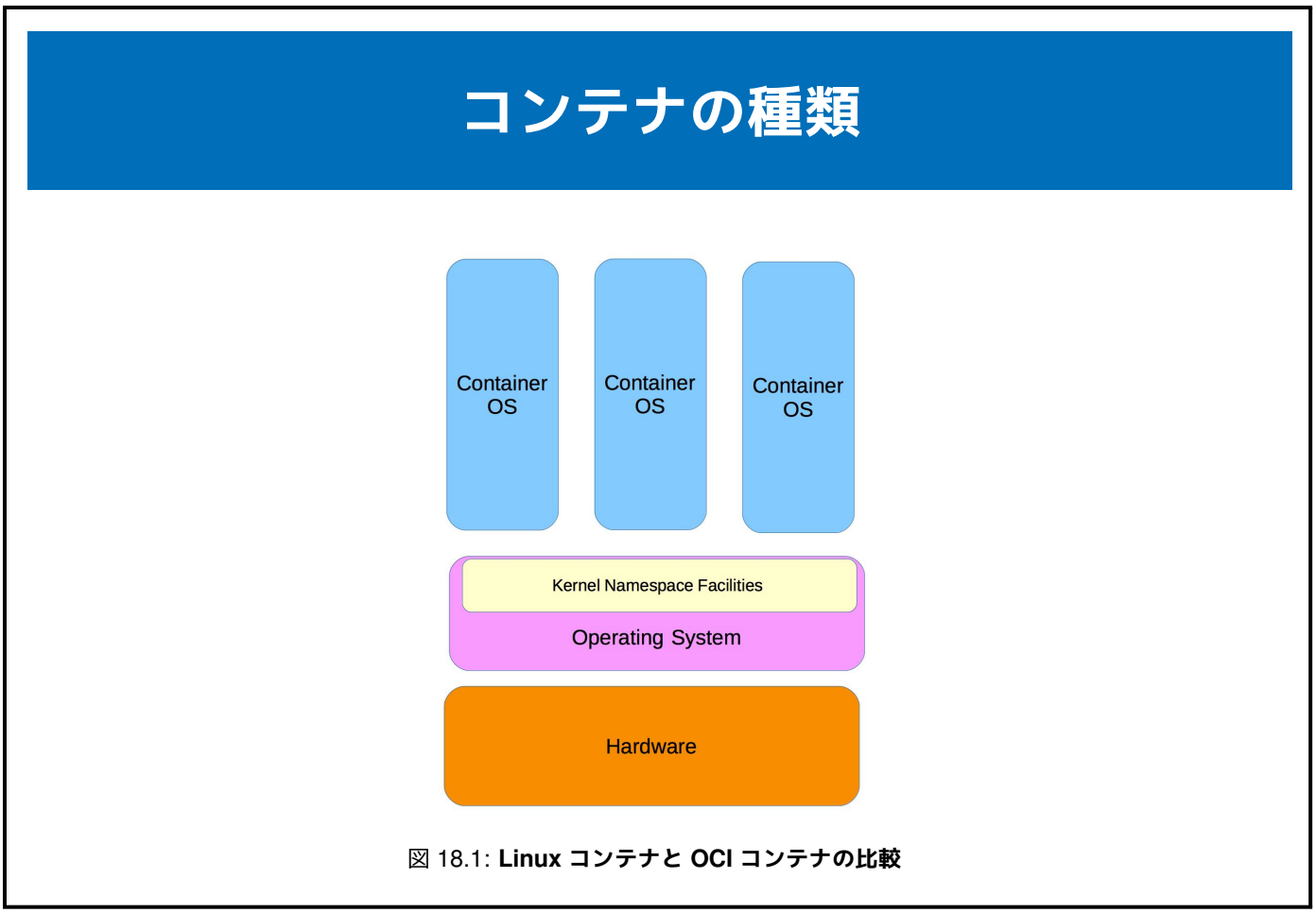


図 18.1: **Linux コンテナと OCI コンテナの比較**

名前空間 や **cgroupes** を含む **Linux** カーネルの機能を使うことで、ホスト カーネルを共有する OS やアプリケーションの軽量なインスタンスを提供することができます。当初 **Linux コンテナ**または **LXC** は、init、システムツール一式、およびアプリケーションを含む仮想 OS 全体をコンテナとして仮想化できるようにするために作成されました。

次に登場したのが **アプリケーション コンテナ** で、**Docker** から始まり、OCI コンテナという一般的な名前で呼ばれるようになっています。これらのコンテナは、通常は OS 全体を含みません。（含めることもできます）この仕組みを使うと、ひとつのアプリケーションまたは機能だけを含んだ小さなイメージを作ることが可能になります。

現代では、このようなアプリケーション コンテナを実行するための OCI 互換の技術が沢山あり、*podman*、*runc*、*firecracker*、*apptainer*、*rkt* などが利用できます。

この章では **Docker** と **docker compose** 機能に焦点を当てますが、ほとんどのテクニックは他の OCI ランタイムにもそのまま適用できます。一般的に **Docker** を指定した **コンテナ レジストリ**のイメージは、少し修正するだけで他のランタイムでも使用することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

18.2 コンテナ vs 仮想マシン

コンテナ vs 仮想マシン

- プロセスとサービスを実行する
  - VM はフル機能のオペレーティング システムを実行するので、多くのサービスやアプリケーションを動かすことができる
  - コンテナは、通常は 1 つのこと（アプリ）に専念する
- VM はコンテナより多くのリソースを利用する
- 複数のコンテナが 1 つの OS カーネルを共有する。一方 VM ではそれぞれが専用のカーネルを持つ
- コンテナの方が移植性が高い
- VM 内でコンテナを動かすこともできる
- コンテナでセキュリティを確保するのは容易ではない
- コンテナの起動は、通常より高速である
- それでも、仮想マシン（= VM）が最適解となるケースは多い

仮想マシンとコンテナは、両方とも重要なニーズを満たすものです。登場してからしばらくの間は両方とも順調で、ほとんど全てのものに適用されると思われていました。

両方とも長い歴史があります

- メインフレームコンピュータは何十年もの間、かなり特殊な方法でソフトウェア分離と仮想マシンを採用していました。
- オペレーティング システムに長年実装されている **chroot** と **BSD Jail** の基本的な分離の考え方は、コンテナと同じです。

多くの異なるサービスとアプリケーションを緊密に統合させる必要がある場合、サービス単位で仮想マシンを動かす、が最適なソリューションとなります。

アプリケーションが、多くのサービスやライブラリなどを持ったフル機能のオペレーティング システム環境を前提に記述されている場合も、仮想マシンが最適でしょう。

コンテナと仮想マシンでは、ワークロードのスケーリングの考え方が異なります。**Kubernetes** や **Mesos** などの **オーケストレーション** システムは、必要に応じてコンテナ数の変更、負荷分散、イメージの複製や削除を行うことができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

18.3 Docker

Docker

Docker はアプリケーションレベルの仮想化である。対象アプリケーション実行に必要なサービスを構築するため、多くの個別イメージを使用する。これらのイメージが、コンテナ内にパッケージ化されている

- イメージは、コンテナ内のコンポーネントである
- イメージに含まれる可能性のあるのは
  - アプリケーションプログラム
  - ランタイムライブラリー
  - システムツール
  - その他、アプリ実行に必要なもの全て
- イメージは **Docker Hub** または **レジストリー サーバー** に置かれる

Docker の広範なドキュメントが https://docs.docker.com にある

Docker の最も魅力的な機能の１つは、全ての依存コードとサービスをアプリケーションと共にパッケージ化し、最小限のオーバーヘッドで単一ユニットとしてデプロイできることです。このデプロイは、要求に応じて何度でも簡単に繰り返すことができます。これにより、アプリケーションをサポートするために何層ものサービスをサーバーに構築する必要が少なくなります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Docker イメージを見つける

Docker でコンテナ化されたアプリケーションは、数ステップで起動できる

- お気に入りのツールで **Docker** サービスパッケージをインストールする
- **Docker** サービスを開始する
- **Docker Hub** 、またはプライベートリポジトリでイメージを検索する
- イメージを取得する
- イメージを実行する
- 最後に、アプリケーションをテストする

上記の詳細な手順は、**Docker** アプリケーションのテストの最小限の例です。

もちろん、イメージを作成し、システム変数または構成パラメータを設定し、結果を新しいイメージとして保存する機能を含めるなど、使用できるオプションが多数あります。場合によっては、書き込み不可ではなく書き込み可能なイメージが必要となる場合もあります。

ほとんどの **Docker** コマンドには、個別の **man** ページが用意されています。具体的には `docker(1)`、`docker-search(1)`、`docker-pull(1)`、`docker-create(1)`、`docker-run(1)` などがあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Docker イメージの拡張

設定値の変更や機能追加のために、既存のコンテナ イメージを変更したいことはよくある。これを行う方法はいくつかある。

- 既存のイメージに対して `Dockerfile` の `FROM` コマンドを使って、新しいイメージを作成する
- **docker-compose** を利用する
- イメージのソース `Dockerfile` を見つけて、それをあなたのプロジェクトのベースとなるように修正する

ほとんどの場合、たとえそれがオペレーティング システムのベースイメージであったとしても、`FROM` コマンドで既存イメージを継承します。

セキュリティや内部的なコンプライアンスの理由から、ローカル ソースや複数のオペレーティング システムのベースイメージを含むこともあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

18.4 Cowsay の例

Cow は言う

ここでは、*cowsay* コマンドを使った非常に基本的な例を示す
これは、文字列をそのまま表示する簡単なものである

cowsay-minimal.Dockerfile

```
# moo
FROM alpine:edge
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay
ENTRYPOINT ["cowsay"]
```

```
$ docker run --rm cowsay-minimal 'and.. Action!'
```

```
 ---------------
< and.. Action! >
 ---------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

この例では **Alpine Linux** を使用していますが、これはサイズが小さく、コンテナのベースとして多く利用されているものです。

Alpine Linux は、これまで取り上げてきたメインラインのディストリビューションとは、いくつかの点で異なっています。

- デフォルトで読み取り専用メディアから実行するように設計されています
- **標準 C ライブラリ** として **glibc** の代わりに **musl** を利用しています
- 全く異なるパッケージ マネージャーを使用しています

更に知りたい？

ここでは、このディストリビューションの詳細について説明しませんが、以下から情報を得ることができます。

1. **Alpine Linux** – https://alpinelinux.org/about/
2. **musl** C ライブラリ – https://musl.libc.org/
3. **apk** パッケージ マネージャ – https://wiki.alpinelinux.org/wiki/Alpine_Package_Keeper

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Cow の使い方

中身を見てみる

```
# moo
FROM alpine:edge
```

コンテナの **ベース** には、OS のローリング リリースを使用している

```
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay
```

システム パッケージ コマンドを使用して cosway をインストールする。デフォルトではないパッケージ リポジトリを利用し、次回のためのパッケージ情報は何も保存しない。

```
ENTRYPOINT ["cowsay"]
```

cowsay コマンドとして、**docker run** をデフォルトに設定する

上の例の RUN ステップでは非常に複雑なコマンドを使用しました。同じことを複数のコマンドで実行して、わかりやすくできなかったのでしょうか？

Docker イメージのサイズと構成が重いことに批判も多いのですが、多くの人は次の意見に賛成するのではないでしょうか。

- サイズは小さい方が望ましい
- **レイヤー** は少ない方が望ましい

Dockerfile の各 RUN 行は、コマンドが **アトミック**に実行されることを表しています。これらのアクションはそれぞれ **レイヤー** と呼ばれるスナップショットに保存され、最終的なイメージを作成するためには、これらすべてのレイヤーをダウンロードする必要があるのです。

レイヤーを少なくするテクニックとして、次のようなものが利用できます。

1. 多くのアクションを一行で記述します。&& 演算子を使うこともあります
2. この詳細な手順を自動的に処理するシステムを構築します
3. **多段階ビルド**を利用します

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Cow のビルド

Dockerfile を作成したら、それをビルドする必要がある

```
$ docker build  -f cowsay-minimal.Dockerfile -t  cowsay-minimal  .
```

```
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM alpine:edge
 ---> 855f10c27d71
Step 2/3 : RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing
↪  cowsay
 ---> Using cache
 ---> 8d2935a86a26
Step 3/3 : ENTRYPOINT ["cowsay"]
 ---> Using cache
 ---> 90a4fc4e505d
Successfully built 90a4fc4e505d
Successfully tagged cowsay-minimal:latest
```

docker build のデフォルトの動作を変更するために、いくつかの引数を指定しました。

-f, --file 使用する Dockerfile のパス／ファイル名

-t, --tag ビルド成功時に作成されるタグの名前

.

最後の必須の引数は、作業ディレクトリの指定です。ここではカレント ディレクトリを指定しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Cow の実行

docker **イメージ** が手に入ったら、それを使ってシステムと情報をやり取りすることが可能になる

```
docker run --rm cowsay-minimal "$(cat /etc/debian_version)"
```

```
 ------
< 11.7 >
 ------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

docker run コマンドの内訳は次の通りです。

--rm **イメージ** から作成された **コンテナ** を削除し、毎回新しくスタートできるようにします。

`cowsay-minimal`

画像の名前、または **タグ** で、実行する特定の画像を指定します。

`"$(cat /etc/debian_version)"`

docker ENTRYPOINT プログラムに渡される残りの引数を指定です。このケースでは、ホスト上で実行されたサブシェルの結果としては、引数は 1 つでした。

重要

動的な引数は、docker に渡される前に処理されます。

注目

別のコマンドを実行するコマンドを実行する場合、最初のプログラムへの引数を **--** で指定するのは、たいていの場合良い結果をもたらします。
具体的には、このコマンドを次のように書くこともできるのです。

```
docker run --rm cowsay-minimal -- "$(cat /etc/debian_version)"
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

18.5 再現可能なビルド (= Reproducible Builds)

異なる Cow

コンテナを使う理由の一つは、同じ環境を複数のマシン上に構築すること。– これにより「私の環境では動作する」問題から開放される。しかしながら、単純に docker または他のコンテナを使うだけでは、完全にこれを担保することができない。我々の例題に戻ってみよう。

cowsay-minimal.Dockerfile

```
# moo
FROM alpine:edge
#    ^^^^^^^^^^^
```

（残りの部分は今のところ無視するが）edge は **Debian** `Testing` や **CentOS** `Stream` と同じような **ローリング リリース** の名前であると以前述べた。この種のディストリビューションは頻繁に更新され、場合によっては毎晩更新される。明日、同じ **Docker イメージ** をビルドしたら、どれくらい変わるのだろうか？

上記の例の `edge` を選択せず、別のバージョンを指定したとしても、セキュリティ アップデートやそれ以外の変更が公開される可能性は残ります。ソフトウエア開発のベースとして **docker** を使っている場合、その **docker イメージ** の予期しない変更が、プロジェクトのコードや設定に加えた変更のように見えているかもしれないのです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

より具体的に

FROM 行には、バージョンでなく sha256 ハッシュによる **ダイジェスト**を指定することもできる

cowsay-digest.Dockerfile

```
#moo
FROM alpine:edge@sha256:2d01a16bab53a8405876cec4c27235d47455a7b72b75334c614f2fb0968b3f90
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay
ENTRYPOINT ["cowsay"]
```

Dockerfile のコピーを作成し、現在のイメージのダイジェストを指定し、再ビルドを行った

```
$ docker build  -f cowsay-digest.Dockerfile -t  cowsay-digest  .
```

```
### よく似た出力が続く ###
```

通常、新しいイメージのダイジェスト ハッシュを見つけるには、docker コマンドライン ツールよりも、web ブラウザーの方が早いです。

ここでは docker **タグ** と **ラベル** の両方を指定しているが、**OCI** コンテナ プログラムによってはそのまま、または sha256 ハッシュを追加するだけで動作するものもあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

イメージの比較

```
$ docker images
```

```
REPOSITORY                TAG                 IMAGE ID       CREATED          SIZE
cowsay-digest             latest              90a4fc4e505d   13 minutes ago   43.2MB
cowsay-minimal            latest              90a4fc4e505d   13 minutes ago   43.2MB
```

追加したハッシュ情報から、docker イメージとしては同じものが作成されていると確認できた。これなら、もし `alpine:edge` の新しいバージョンが公開されても、誤ってアップグレードしてしまう心配はない。

重要

特定のバージョンを宣言（*pin*）しておけば、誤ってアップグレードすることは無くなりますが、重要なセキュリティパッチを受け取れるように、何らかのアップグレード手順を用意することは重要です。

いくつかの情報セキュリティ企業が、公表されている公開コンテナの約半数にはセキュリティ上の脆弱性があり、中には深刻なものもあるという報告書を発表しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ところで、これはどうだろう？

この例で変更できるのは、ソフトウエアのベース イメージだけでは無いことに気づいているだろうか？

cowsay-digest.Dockerfile

```
RUN apk add --no-cache -X http://dl-cdn.alpinelinux.org/alpine/edge/testing cowsay
```

ここでは、ベース システムの一部ではないプログラムをインストールしているが、これで変化しないのか？

変化する。もし **全てが完全に（bit-by-bit）**同じバージョンのイメージを作りたいのであれば、一時ファイルやファイルシステム内のタイムスタンプまで考慮しなければならない。

これは難問だが、実現できればメリットも大きい。そして、この課題に対する重要な研究がすでに始められている。

更に知りたい？

Reproducible Builds

詳細は https://reproducible-builds.org/ を参照

タイムスタンプが変更された同じソースからリビルドした後でも、同じパッケージのバージョンを提供する方法の一つは「スナップショット サーバー」の利用です。

Debian でのやりかたについては、こちらのブログ記事を参照してください。
https://www.qubes-os.org/news/2021/10/08/reproducible-builds-for-debian-a-big-step-forward/

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

18.6 演習

課題 18.1: 小さなユーティリティを docker で使えるようにする

cowsay の例を参考に、**httpie** ユーティリティ (https://github.com/httpie/httpie) 用に同じような小さな Docker コンテナを構築してみましょう。

解 18.1

Alpine Linux のソリューション

1. **パッケージ名を確認する**
   “httpie” だろうと推測したのなら、それで問題は無いでしょう。または https://pkgs.alpinelinux.org/package/edge/community/x86_64/httpie のパッケージ検索を利用するか、アップストリームの README 情報を確認することもできます。
2. **Dockerfile を作成する**
   ここで利用するバージョンは **ミニマル** でも **ダイジェスト** でも問題はないのですが、この例では各ビルドの最新バージョンを利用したいでしょうから、おそらく実際には **ミニマル** バージョンを使うことになるでしょう。加えて、**テスト** のために追加の repo 引数は必要ではありませんが、引数があっても何も支障はありません。

httpie.Dockerfile

```
FROM alpine:edge
RUN apk add --no-cache httpie
CMD ["httpie"]
```

3. **イメージのビルド**

```
$ docker build  -f httpie.Dockerfile -t  httpie:latest .
```

```
Sending build context to Docker daemon  30.72kB
Step 1/3 : FROM alpine:edge
 ---> 855f10c27d71
Step 2/3 : RUN apk add --no-cache httpie
 ---> Running in 2f6b1436e5e3
fetch https://dl-cdn.alpinelinux.org/alpine/edge/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/edge/community/x86_64/APKINDEX.tar.gz
(1/57) Installing libbz2 (1.0.8-r6)
(2/57) Installing libexpat (2.5.0-r2)
(3/57) Installing libffi (3.4.4-r3)
(4/57) Installing gdbm (1.23-r1)
(5/57) Installing xz-libs (5.4.3-r1)
(6/57) Installing libgcc (13.1.1_git20230617-r0)
(7/57) Installing libstdc++ (13.1.1_git20230617-r0)
(8/57) Installing mpdecimal (2.5.1-r2)
(9/57) Installing ncurses-terminfo-base (6.4_p20230625-r0)
(10/57) Installing libncursesw (6.4_p20230625-r0)
(11/57) Installing libpanelw (6.4_p20230625-r0)
(12/57) Installing readline (8.2.1-r2)
(13/57) Installing sqlite-libs (3.42.0-r2)
(14/57) Installing python3 (3.11.4-r0)
(15/57) Installing python3-pycache-pyc0 (3.11.4-r0)
(16/57) Installing pyc (0.1-r0)
(17/57) Installing py3-multidict (6.0.4-r2)
(18/57) Installing py3-multidict-pyc (6.0.4-r2)
(19/57) Installing py3-parsing (3.1.0-r0)
(20/57) Installing py3-parsing-pyc (3.1.0-r0)
(21/57) Installing py3-packaging (23.1-r1)
(22/57) Installing py3-packaging-pyc (23.1-r1)
(23/57) Installing py3-setuptools (68.0.0-r0)
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
(24/57) Installing py3-setuptools-pyc (68.0.0-r0)
(25/57) Installing py3-pip (23.1.2-r0)
(26/57) Installing py3-pip-pyc (23.1.2-r0)
(27/57) Installing py3-pygments (2.15.1-r0)
(28/57) Installing py3-pygments-pyc (2.15.1-r0)
(29/57) Installing py3-pysocks (1.7.1-r5)
(30/57) Installing py3-pysocks-pyc (1.7.1-r5)
(31/57) Installing py3-certifi (2023.5.7-r0)
(32/57) Installing py3-certifi-pyc (2023.5.7-r0)
(33/57) Installing py3-charset-normalizer (3.1.0-r1)
(34/57) Installing py3-charset-normalizer-pyc (3.1.0-r1)
(35/57) Installing py3-idna (3.4-r4)
(36/57) Installing py3-idna-pyc (3.4-r4)
(37/57) Installing py3-urllib3 (1.26.15-r2)
(38/57) Installing py3-urllib3-pyc (1.26.15-r2)
(39/57) Installing py3-requests (2.31.0-r0)
(40/57) Installing py3-requests-pyc (2.31.0-r0)
(41/57) Installing py3-requests-toolbelt (1.0.0-r0)
(42/57) Installing py3-requests-toolbelt-pyc (1.0.0-r0)
(43/57) Installing py3-attrs (23.1.0-r1)
(44/57) Installing py3-attrs-pyc (23.1.0-r1)
(45/57) Installing py3-mdurl (0.1.2-r2)
(46/57) Installing py3-mdurl-pyc (0.1.2-r2)
(47/57) Installing py3-markdown-it-py (3.0.0-r0)
(48/57) Installing py3-markdown-it-py-pyc (3.0.0-r0)
(49/57) Installing py3-rich (13.4.2-r0)
(50/57) Installing py3-rich-pyc (13.4.2-r0)
(51/57) Installing py3-wheel (0.40.0-r1)
(52/57) Installing py3-wheel-pyc (0.40.0-r1)
(53/57) Installing httpie-pyc (3.2.2-r0)
(54/57) Installing py3-defusedxml-pyc (0.7.1-r4)
(55/57) Installing python3-pyc (3.11.4-r0)
(56/57) Installing py3-defusedxml (0.7.1-r4)
(57/57) Installing httpie (3.2.2-r0)
Executing busybox-1.36.0-r5.trigger
OK: 95 MiB in 72 packages
Removing intermediate container 2f6b1436e5e3
 ---> 227f4a0c96ab
Step 3/3 : ENTRYPOINT ["httpie"]
 ---> Running in f19a66bab6cb
Removing intermediate container f19a66bab6cb
 ---> c0b613745c84
Successfully built c0b613745c84
Successfully tagged httpie:latest
```

4. **イメージを実行する**

```
$ docker run --rm httpie
```

```
usage: httpie [-h] [--debug] [--traceback] [--version] {cli,plugins} ...
httpie: error: Please specify one of these: 'cli', 'plugins'

This command is only for managing HTTPie plugins.
To send a request, please use the http/https commands:

  $ http POST pie.dev/post hello=world

  $ https POST pie.dev/post hello=world
```

そして、提案されたコマンドを試してみます。

```
$ docker run --rm httpie https
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
usage:
    http [METHOD] URL [REQUEST_ITEM ...]

error:
    the following arguments are required: URL

for more information:
    run 'http --help' or visit https://httpie.io/docs/cli
```

5. （オプション）**エイリアスに登録する**

```
$ alias https='docker run --rm httpie https -- '
```

```
$ https httpie.io/hello
{"ahoy":["Hello, World!  Thank you for trying out HTTPie ","We hope this will become a
↪  friendship."],"
↪  links":{"homepage":"https://httpie.io","twitter":"https://twitter.com/httpie",
    "discord":"https://httpie.io/discord","github":"https://github.com/httpie"}}
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 19 章

Linux ファイルシステムと VFS

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- ファイルシステムの基本的な構成について説明する。
- VFS (Virtual File System) の役割を理解する。
- **Linux** で利用できるファイルシステムは何か、実際のシステムで利用できるファイルシステムは何かを知る。
- ジャーナリングファイルシステムは、なぜ大きな進化を遂げたのかを理解する。
- **Linux** での特殊なファイルシステムの使用について説明する。

19.1 ファイルシステムの基本

ファイルシステムの基本

- **Linux** プログラムは、ファイルに対してリード／ライトを行う（ディスクのセクターに対してではない）
- **ファイル** とは物理的な I/O レイヤーをカモフラージュし抽象化したもの
- **ファイルシステム** は物理パーティション上に利用可能なフォーマットを作る
- **UNIX** 系のファイルシステムは、ツリー型の階層構造を持つ
  - ディレクトリにはファイルやディレクトリを持つ
  - 全てのパスやノードはルート (/) ディレクトリの配下に置かれる
- 複数のファイルシステムを単一のツリー構造に統合できる（通常は統合されている）
- **Linux** は、仮想ファイルシステムレイヤー (**VFS**) を介し、ファイルシステムと連携する

アプリケーションプログラムは、**ファイル** が保存されている実際のハードウェア上の物理的な場所にアクセスするのではなく、ファイルを読み書きします。

ファイルとその名前は、物理 I/O レイヤーをカモフラージュして抽象的に見せたものです。（**ファイルシステムレイヤー** を無視して）**コマンド** を使って直接ディスクに書き込むのは非常に危険です。ディスクへの書き込みは、ユーザーアプリケーションではなく、低レベルのオペレーティング システムソフトウェアによってだけ行われます。例外は、エンタープライズデータベースなど非常にハイエンドなソフトウェアで、これらはファイルシステムに起因するレイテンシを回避するために直接 **コマンド** アクセスをします。

ローカルファイルシステムは、通常ディスク上の物理パーティション上に置かれます。または、**論理ボリュームマネージャー（LVM）** によって制御される論理パーティション上に置かれることもあります。ファイルシステムはネットワークの先に配置することもでき、この場合にはファイルシステムの実体はネットワークを介してローカルシステムからは完全に隠蔽されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

19.2 ファイルシステムのコンセプト

inode

- **名前** はファイルの 1 つのプロパティに過ぎず、本質は inode である
- inode はファイルシステム内に保存され、同時にメモリー上のデータ構造である
- inode には、ファイルに関する以下の情報が記載され保存されている
  - パーミッション
  - ユーザー、グループのオーナー
  - サイズ
  - タイムスタンプ (ナノ秒)
    - 最終アクセス時間
    - 最終変更時間
    - inode 情報の最終変更時間
- ファイル名は inode には **保存されていない**、**ディレクトリ** に保存されている

inode は、場所を含むファイル属性を記述し保存するディスク上のデータ構造体です。

Linux の全てのファイルは、個別の inode に関連付けられています。ファイルに関するデータは、全てそのファイルの inode に含まれます。オペレーティング システムは、最新のファイル名、場所、アトリビュート（パーミッション、オーナーシップなど）、アクセス時間 などを inode から取得します。このためファイルに関わる全ての I/O アクティビティには、通常そのファイルの inode が含まれます。

タイムスタンプが 3 つあることに注意してください

1. **アクセス時間**: ファイルが何らかの目的で最後にアクセスされた時間
2. **修正時間**: ファイルの **内容**が最後に修正された時間
3. **変更時間**: パーミッション、所有権、ファイル名、ハードリンクなどの変更により、ファイルの inode が最後に変更された時間

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ハードリンクとソフトリンク

- **ハード** リンクは inode そのものを指す
  - 2 つ以上のファイルが同じ inode を指している状態 (ハードリンク)。
  - ハードリンクを指定できるのは、同一ファイルシステム内のファイルに限定される
- **ソフト** (**シンボリック**) リンクはファイル名を関連させる inode と結びつける
  （訳注、ソフトリンクはリンク先のファイルの実体（=inode）のショートカットで実体は別のファイル）
  - ソフトリンクは、別ファイルシステム上のファイルとのリンクが可能である
  - ターゲットは存在していなくても、さらにマウントさえされていなくてもリンクの設定は可能であり、**宙ぶらりん（未解決）**で良い
- ハードリンクされたファイルに注意
  - 一カ所で行ったコンテンツ変更（訳注：ファイル名変更、削除など）は別の場所には反映されない

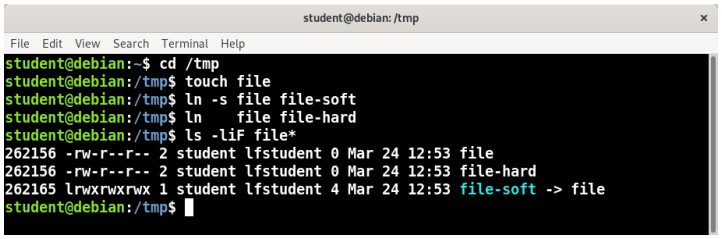


図 19.1: **ハードリンクとソフトリンク**

ディレクトリファイルは、ファイル名と inode を関連付けるために使用される特殊な種類のファイルです。ファイル名を inode に関連付ける（または **リンク**する）には 2 つの方法があります。

- **ハードリンク** は一つの（＝同じ）inode を指します。オプション無しの **ln** コマンドでハードリンクを作ることができます。
- **ソフトリンク** (または **シンボリック**) リンクは inode に関連付いたファイル名を指す。**ln** に `-s` オプションを付加して作成します。

ディレクトリファイルの内容と inode の関連付けはリンクと呼ばれます。リンクを追加するには **ln** を使用します。

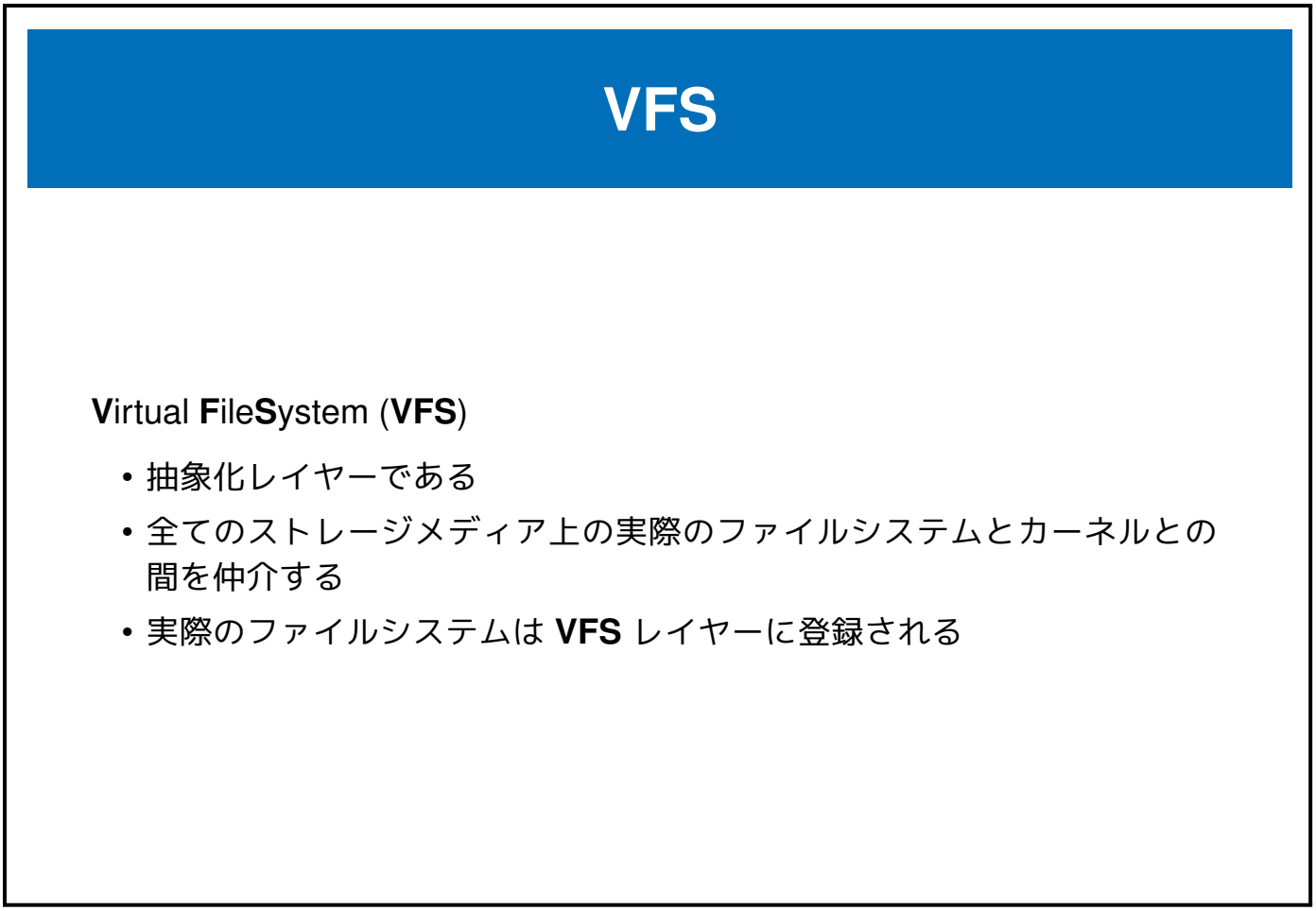
2 つ以上のディレクトリエントリが、同じ inode（ハードリンク）を指す可能性があります。(多く使われます）このようなファイルは、別の名前からアクセス可能でそれぞれ独自のディレクトリ構造内に独自の場所を持てます。ただし、どの名前から呼ばれても inode は 1 つだけです。

プロセスがパス名を参照すると、カーネルはディレクトリを検索して対応する inode 番号を見つけます。名前が inode 番号に変換された後、inode はメモリーにロードされ、後続のリクエストで使用されます。

通常、ファイルを変更しても、同じ inode を参照するハードリンクが壊れることはありません。しかし、ファイルをコピーして変更してから置き換えたり、ファイルを削除して置き換えたりして、その過程でリンクがなくなった新しいファイルを作成する（行儀の悪い／バグを持った）アプリケーションもあります。ですから、この動作が意図されたものでない場合は、目を離さないようにしてください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

19.3 仮想ファイルシステム (VFS)



Linux は、最新の全てのオペレーティング システムと同様に **仮想ファイルシステム**（VFS）を実装しています。アプリケーションがファイルにアクセスする必要がある場合 **VFS** 抽象化レイヤーとやり取りし、全ての I/O システムコール（読み取り、書き込みなど）を特定の実ファイルシステム用の固有のコードに変換します。

そのため、アプリケーションは特定の実ファイルシステムやそれが存在する物理媒体、およびハードウェアを考慮する必要がありません。さらに、ネットワーク ファイルシステム（**NFS** など）も透過的に（＝ローカルファイルと全く同様に）処理できます。これにより **Linux** では、他のどのオペレーティング システムよりも多くの種類のファイルシステムが動作します。あらゆるファイルシステムが平等にサポートされたことは、**Linux** 成功の大きな要因となりました。

ほとんどのファイルシステムには完全な読み書きのアクセス権がありますが、一部のファイルシステムには読み取りのアクセス権と実験的な書き込みアクセス権しかありません。一部の種類のファイルシステム、特に非 **UNIX** ベースのものは、**VFS** として見せるためにより多くの操作が必要になる場合があります。

vfat などには owner/group/world フィールドに対して個別の 読み取り／書き込み／実行 のパーミッションがありません。**VFS** ではこの３種類のユーザーに対するパーミッションをそれぞれ想定する必要があります。この想定がマウント操作の影響を受ける可能性があります。また、**ntfs-3g** (https://sourceforge.net/projects/ntfs-3g/) のような 読み取り／書き込み をサポートするカーネル外のファイルシステム実装もあります。これは信頼性はありますが、カーネル内ファイルシステムよりもパフォーマンスは低くなります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

19.4 利用可能なファイルシステム

利用可能なファイルシステム

- **Linux** では、他のオペレーティング システムより多くの種類のファイル システムを利用することが可能である
- ファイルシステム選択の自由度（公平性）は Linux 成功の大きな要因
- （kernel の）ネイティブ ファイルシステムの多くは、完全な読み／書き アクセスができるが、他のオペレーティング システム由来や特殊なファ イルシステムの場合は読み取り、または実験的な書き込みアクセスしか できない場合がある
- （S3 互換など）**key-value 型** のファイルシステムには、全てのファイル 操作が許可されないものもあるが、これらではユーザー空間から FUSE 経由でアクセスできる場合が多い
- **ext4、xfs、btrfs、squashfs、nfs、vfat** が幅広く利用されている
- `/proc/filesystems` には、現在サポートされている カーネル ファイル システムのリストがある

現在実行中の **Linux** カーネルに登録され、認識されているファイル システム タイプの一覧は、次のようにして見ることができます。

```
$ cat /proc/filesystems
```

```
nodev   sysfs           nodev   bpf             nodev   autofs
nodev   tmpfs           nodev   pipefs          nodev   mqueue
nodev   bdev            nodev   ramfs           nodev   pstore
nodev   proc            nodev   hugetlbfs               fuseblk
nodev   cgroup          nodev   rpc_pipefs      nodev   fuse
nodev   cgroup2         nodev   devpts          nodev   fusectl
nodev   cpuset                  squashfs                ext3
nodev   devtmpfs                vfat                    ext2
nodev   configfs                msdos                   ext4
nodev   debugfs                 exfat                   ntfs3
nodev   tracefs         nodev   nfs             nodev   overlay
nodev   securityfs      nodev   nfs4
nodev   sockfs          nodev   nfsd
```

(nodev を持つものはストレージ上に存在しない **スペシャル ファイルシステム** です。) 追加のファイルシステムは、システムがそれらを使用するパーティションにアクセスしようとするときだけ、そのコードがモジュールとしてロードされるかもしれません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ファイルシステムの種類

Linux では多くの種類のファイルシステムを利用可能、大部分のファイルシステムは完全な読み書きをサポートする、一例を示すと

- **ext4**: **Linux** のネイティブ ファイルシステムである (**ext2**、**ext3** も同様)
- **XFS**: 元々は **SGI** が開発した高性能ファイルシステム
- **JFS**: 元々は **IBM** が開発した高性能ファイルシステム
- **Windows** ネイティブ: **FAT12**、**FAT16**、**FAT32**、**VFAT**、**NTFS**
- メモリーに常駐する仮装ファイルシステムには **proc**、**sysfs**、**devfs**、**debugfs** などがある
- **NFS**、**coda**、**afs** などネットワーク ファイルシステムもある
- その他

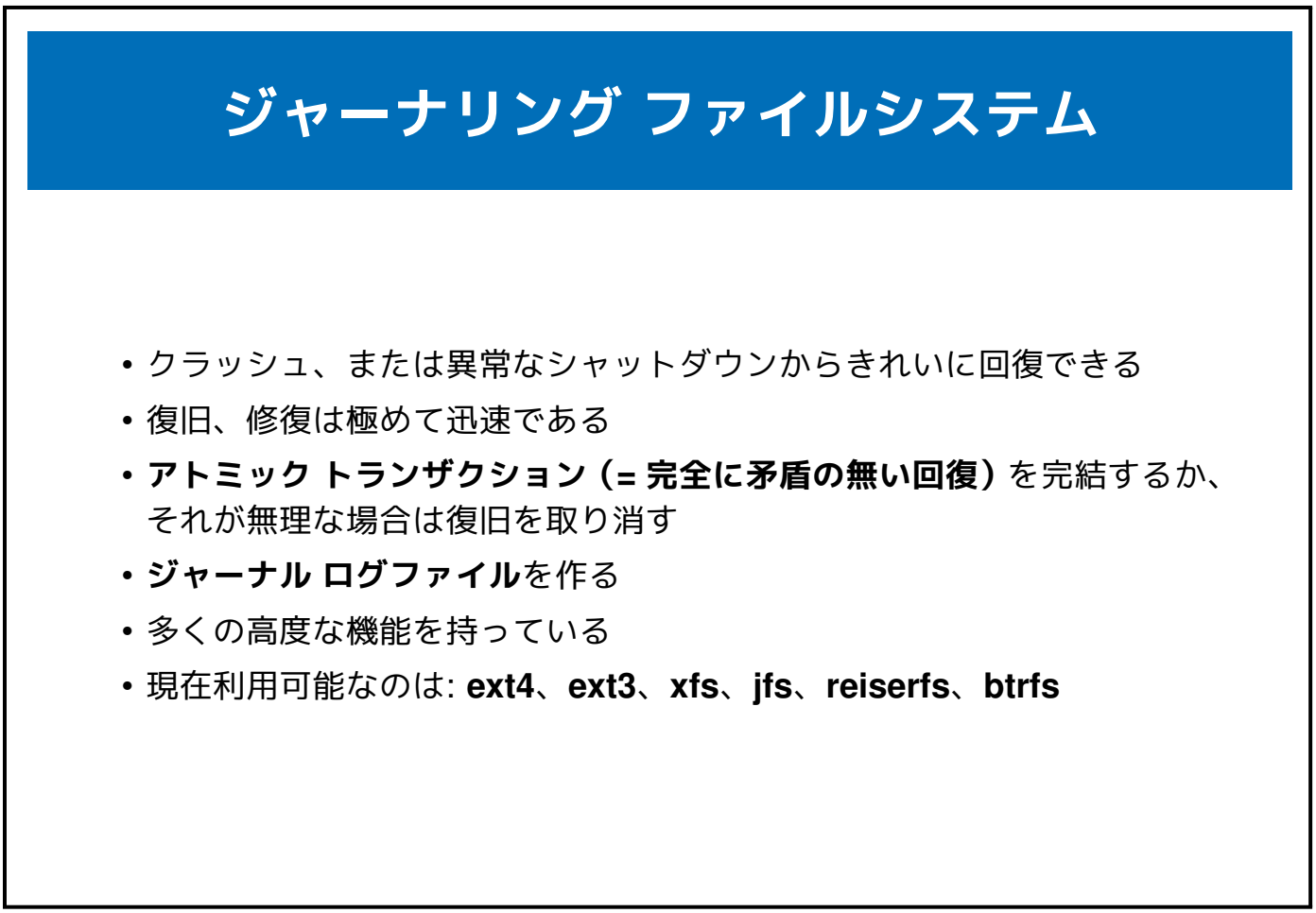
Linux は多くの種類のファイルシステムをサポートしており、常に追加され続けています。これらには、以下のようなものがあります

- **ext4** など、**Linux** 向けとして直接開発されたネイティブ ファイル システム
- **xfs** や **ntfs** など、他の OS から持ち込まれたファイル システム
- **debugfs** のように、実際のファイルシステムではないが、特定の目的のためにインフラを利用するスペシャル ファイル システム

Linux は高度にモジュール化された構造になっており、この柔軟性が活かされている。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

19.5 ジャーナリング ファイルシステム



ジャーナリング ファイルシステムを使うと、システムクラッシュや異常なシャットダウンから、ほとんどまたはまったくファイルを破損することなく非常に迅速に回復させることができます。これを利用するには、さらに多くの操作を行うコストが伴いますが、追加の機能強化にはコストを払う価値があります。

ジャーナリング ファイルシステムでは、操作は **トランザクション** にグループ化されます。トランザクションは **アトミック** に実行され、エラーなしで終了する必要があります。それが出来ない時にはファイルシステムは変更されません。トランザクションのログファイルは保持されます。エラーが発生した場合、通常は最後のトランザクションだけを調べます。

Linux では、以下のジャーナリング ファイルシステムを自由に利用できます。

- **ext3** は、以前の非ジャーナリング **ext2** ファイルシステムの拡張です。
- **ext4** は、**ext3** を大幅に強化したものです。機能にはエクステントや 48 ビットのブロック番号を含み、対応できる最大サイズは 16TB です。ほとんどの **Linux** ディストリビューションがかなり長い間、**ext4** をデフォルトのファイルシステムとして使用しています。
- **reiserfs** は **Linux** で使用された最初のジャーナリング実装でしたが、リーダーシップを失い開発は中止されました。
- **JFS** は元々は **IBM** IBM の製品であり、**IBM** の **AIX** オペレーティング システムから移植されました。
- **XFS** は元々は **SGI** の製品であり、**SGI** の **IRIX** ペレーティングシステムから移植されました。**RHEL 7** はデフォルトのファイルシステムとして **XFS** を採用しました。
- **btrfs** は最新のジャーナリング ファイルシステムであり、今も急速に開発が進んでいます。**SUSE** と **openSUSE** システムのデフォルトのファイルシステムです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

19.6 スペシャル ファイルシステム

スペシャル ファイルシステム

- スペシャル ファイルシステムは、システムの内部へのアクセスや特定の機能を実装するために使われる
- マウントポイントを持つもの (`/proc` の **proc** や `/sys` の **sys** など)
- マウントポイントを持たないもの (**sockfs** や **pipefs** など)
- スペシャル ファイルシステムは、本当のファイルシステムではない。それらは、ファイルシステムの構造的な抽象構造を、利用し易いデータと機能として見せるためのカーネルの仕組み（や、そのサブシステム）である

Linux は、特定のタスクのために **スペシャル ファイルシステム** を幅広く採用しています。これらは、さまざまなカーネルデータ構造へのアクセス、カーネルの動作調整、特定の機能の実装に非常に役に立ちます。一例としては:

Table 19.1: **スペシャル ファイルシステム**

ファイルシステム	マウントポイント	目的
rootfs	なし	カーネルのローディング中に、空のルートディレクトリを提供する
hugetlbfs	任意	拡張ページアクセスを提供する（**x86** では 2 または 4 MB）
bdev	なし	ブロックデバイスで利用される
proc	`/proc`	各種カーネル機構とそのサブシステムにアクセスするための仮想ファイル
sockfs	なし	**BSD ソケット** で利用される
tmpfs	任意	スワップ や リサイズ を伴う RAM ディスク
shm	なし	System V IPC 共有メモリーによって使われる
pipefs	なし	パイプによって使われる
binfmt_misc	任意	さまざまな実行プログラムに利用される
devpts	`/dev/pts`	**Unix98** の仮想端末によって利用される
usbfs	`/proc/bus/usb`	**USB** サブシステムの動的接続されるデバイスによって利用される
sysfs	`/sys`	**デバイス ツリー** によって使われる
debugfs	`/sys/kernel/debug`	簡易的なデバッグ用のファイルアクセスに利用される

これらのスペシャル ファイルシステムの中には、マウントポイントがないものがあることに注意してください。つまりユーザーアプリケーションは、それらのファイルシステムとはやり取りしません。しかしカーネルは **VFS** レイヤーとコードを利用してそれらを使用します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

19.7 演習

デモ教材ビデオ

using_available_filesystems_demo.mp4

課題 19.1: tmpfs スペシャル ファイルシステム

tmpfs は **Linux** で使われている色々なスペシャル ファイルシステムの１つです。これらのスペシャル ファイルシステムの中には、実際のファイルシステムとして使われるのではなくファイルシステムに見せる抽象化の利点だけを利用しているものがあります。しかし、**tmpfs** は実際にアプリケーションが I/O を行うことができるファイルシステムです。

tmpfs はメモリー内に存在し基本的に **ramdisk** として機能します; しかし従来の RAM ディスクが持っていなかった素晴らしい機能を有しています。

1. 大きさは動的に調整されます（使用するメモリー量を調整します）; ０から開始してマウント時に指定された最大値まで必要に応じて広がります。
2. RAM が枯渇したら **tmpfs** はスワップ領域を利用します。（しかし、許される最大値を越える量をファイルシステムに書き込むことはできません）
3. **tmpfs** は **ext3** や **vfat** と違いファイルシステムを構築する必要がありません; 独自の方法でメモリー内の領域を考慮しながらファイルと I/O を処理します。（それはブロックデバイスではありません）そしてスピードを最適化しています。
   つまり `mkfs` コマンドでファイルシステムをフォーマットする必要がありません; マウントして使うだけです。

新しく **tmpfs** を適当なディレクトリにマウントします。

```
$ sudo mkdir /mnt/tmpfs
$ sudo mount -t tmpfs none /mnt/tmpfs
```

ファイルシステムに割り当てられたスペースと、どのくらい使用されているかを見ます。

```
$ df -h /mnt/tmpfs
```

デフォルトで RAM の半分のサイズが割り当てられています; しかし全く利用されていません、`/mnt/tmpfs` にファイルを格納すると利用されます。

マウント時のオプションで割り当てサイズを変更できます。

```
$ sudo mount -t tmpfs -o size=1G none /mnt/tmpfs
```

最大容量まで格納して何が起こるか見てください。終了したら忘れずにアンマウントしてください。

```
$ sudo umount /mnt/tmpfs
```

最近の **Linux** ディストリビューションは、**tmpfs** を `/dev/shm` にマウントします。

```
$ df -h /dev/shm
```

```
Filesystem     Type   Size  Used Avail Use% Mounted on
tmpfs          tmpfs  3.9G   24M  3.9G   1% /dev/shm
```

たくさんのアプリケーションが **POSIX** の共有メモリーを使うときに、**tmpfs** をプロセス間の通信手段として利用します。誰でも `/dev/shm` にファイルを作成、読み出し、書き込みができます。これはメモリーに一時ファイルを作成するのにちょうど良いのです。

いくつかのファイルを `/dev/shm` 中に作成し、**df** でファイルシステムの使用状況を調べてください。

さらに、たくさんのディストリビューションが複数の **tmpfs** をマウントしています; **RHEL** システムを例にあげます。

```
$ df -h | grep ' tmpfs'
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
tmpfs          tmpfs     7.8G   38M  7.8G   1% /dev/shm
tmpfs          tmpfs     7.8G   18M  7.8G   1% /run
tmpfs          tmpfs     7.8G     0  7.8G   0% /sys/fs/cgroup
tmpfs          tmpfs     1.6G  1.2M  1.6G   1% /run/user/42
tmpfs          tmpfs     1.6G   56K  1.6G   1% /run/user/1000
```

このシステムには 16GB の RAM が搭載されています。そのため **tmpfs** ファイルシステムが全て（同時に）、デフォルトで割り当てられた約 8GB を使うことはできません！

注目

いくつかのディストリビューション（**Fedora** など) は、（標準で）**tmpfs** を `/tmp` にマウントします; そのときは `/tmp` に巨大なファイルを格納することを避けて、メモリーが枯渇しないようにします。または `/tmp` について、以前に説明した方法でこれを停止させます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 20 章

ディスクのパーティション分割

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- 一般的なハードディスクとデータバスを比較して説明する。
- ディスクジオメトリやその他のパーティショニングの概念を説明する。
- ディスクデバイスの名前の付け方、関連するデバイスノードの特定方法を理解する。
- fdisk、blkid、lsblk などのユーティリティを使用する。
- 各種のパーティションの考え方の違いを理解したうえで選択する。
- パーティションテーブルのバックアップと復元。

20.1 一般的なディスクの種類

一般的なディスクの種類

- **SATA** (Serial AT Attachment)
  - データ転送が高速で、**IDE** と比べてケーブルが小さい
  - **SCSI** デバイスと見なされる
- **SCSI** (Small Computer Systems Interface)
  - 全般に高速である
  - Fast、Wide、Ultra、Ultrawide など多くのバージョンがある
- **SAS** (Serial Attached SCSI)
  - 新しいポイントツーポイント（point-to-point）プロトコルである
  - **SATA** ディスクよりも優れたパフォーマンスが出せる
- **USB**
  - フラッシュドライブとフロッピーが含まれ、SCSI デバイスと見なされる
- **SSD**
  - 可動部品がなく、汎用コントローラに接続できる
- **IDE** と **EIDE** (Integrated Drive Electronics, Enhanced IDE)
  - 使われなくなった

さまざまな種類のハードディスクがあり、それぞれが接続されている **データバス**の種類、速度、容量、複数のドライブが同時に動作する頻度や、その他の要因によって特徴付けられます。

SATA ディスクは、古い **IDE** ドライブを置き換えるために設計されました。小さなケーブルサイズ（７ピン）、ネイティブホットスワップ、そして、より高速で効率的なデータ転送を提供します。

SCSI ディスクの範囲は narrow（8 ビットバス）から wide（16 ビットバス）で、転送速度は毎秒 5MB（narrow、標準 **SCSI**）から毎秒 160MB（**Ultra-Wide SCSI-3** ）です。

大半の PC は、シングルエンド（＝信号線が１本）または差動（＝信号線がペア）のドライブを使っています。残念ながらこれらに互換性はありません。幸いなことに、２種類のデバイスが同じコントローラ上に共存することはできます。

シングルエンドのデバイスコントローラは、最大７台のデバイスを接続することが可能で、最大ケーブル長は約６メートルです。差動タイプのコントローラは、最大１５台のデバイスをサポートし、ケーブル長の最長は１２メートルです。

最新の **SSD** ドライブ (Solid State Drive) は価格が下がってきています。また、可動部品がなく回転式のメディアを備えたドライブよりも、消費電力が少なく転送速度が高速です。内蔵 **SSD** は、従来のドライブと同じ外形寸法で同じ筐体に取り付けられます。

SSD の価格はまだ少し高いですが、価格は下がってきています。同じマシンに **SSD** と回転式ドライブの両方を搭載するのが一般的です。**SSD** は、頻繁にアクセスされパフォーマンスがクリティカルなデータ転送に使われます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.2 ディスク ジオメトリー

ディスク ジオメトリー

- **回転式** のディスクは、１つ以上のプラッタで構成され、それぞれが１つ以上のヘッドで読み取られる
- ディスクが回転すると、ヘッドはプラッタ上の円形トラックを読み取る
- 円形トラックはセクターと呼ばれるデータブロックに分割されている
- 通常セクターサイズは、512 バイトである
- シリンダは、全プラッタ上の同一トラックで構成されるグループである
- **SSD** には、このようなジオメトリーの概念はない

```
$ sudo fdisk -l /dev/sdc |grep -i sector
```

```
Disk /dev/sdc: 1.8 TiB, 2000398934016 bytes, 3907029168 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
```

ディスクコントローラが、その殆どを隠蔽するようになっているので、物理的なディスク上のどこにデータが配置されているかは以前ほど重要ではなくなっています。さらに **SSD** には何も可動パーツは無く、ジオメトリーの概念自体がなくなりました。
fdisk でジオメトリーを表示することができます。

```
File  Edit  View  Search  Terminal  Help
c7:/tmp>sudo fdisk -l /dev/sda

Disk /dev/sda: 2000.4 GB, 2000398934016 bytes, 3907029168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: dos
Disk identifier: 0x000852df

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1            2048  1048578047   524288000   8e  Linux LVM
/dev/sda2      1048578048  2097154047   524288000   8e  Linux LVM
/dev/sda3      2097154048  3907028991   904937472    5  Extended
/dev/sda5      2097156096  3145732095   524288000   8e  Linux LVM
/dev/sda6      3890448384  3907028991     8290304   82  Linux swap / Solaris
c7:/tmp>
```

図 20.1: **fdisk の利用**

-l オプションを使うと対話モードに入らずに、単にパーティション テーブルをリストします。

現在、セクターサイズが 512 バイト以上のディスクの製造が始まっていて、4KB 品の入手が可能になっています。セクターサイズが大きい方が転送速度が稼げるのですが、オペレーティング システムの大きな（セクター）サイズのサポートはまだ枯れていません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

20.3 パーティション

パーティション構成

- ディスクは **パーティション** に分割される
- パーティションは、ディスク上の物理的に連続する領域である
- ２つのパーティション レイアウトが使われている
  - **MBR** (Master Boot Record)
  - **GPT** (GUID Partition Table)
- **MBR** の原点は、**MSDOS** の初期にまでさかのぼる
  最大４つまでプライマリパーティションを持つことができ、そのうち１つは **拡張パーティション**に指定可能である、**拡張パーティション** に最大 15 個に分割可能な **論理パーティション** を持たせることができる
- **UEFI** (Unified Extensible Firmware Interface) に基づく **GPT** は、全ての最新システムに搭載され、デフォルトで最大 128 個の **プライマリー パーティション** を持つことができる

- **MBR** スキームを使用する場合

  たとえば `/dev/sda` などの **SCSI** の場合、`/dev/sda1` が最初のプライマリ パーティションで、`/dev/sda2` が２番目のプライマリ パーティションとなります。拡張パーティション `/dev/sda3` を作成した場合、それを論理パーティションに分割できます。４つ目以上のパーティションは、全て論理パーティションです。（拡張パーティション内に含まれることを意味します）拡張パーティションは１つしか設定できませんが、多数の論理パーティションに分割できます。

  注意: **Linux** ではパーティションはシリンダ境界で開始または終了している必要はありませんが、他のオペレーティング システムではそうでないかもしれません。このため広く導入されている **Linux** のパーティション分割ユーティリティも、シリンダ境界で終了させようとします。もちろんパーティションは重複してはいけません。

- **GPT** スキームを使用する場合:

  拡張パーティションを使わずに（デフォルトで）128 個のプライマリ パーティションを持つことができます。

  パーティションのサイズは、最大 2^{33} TB です。（MBR ではわずか 2TB です）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

パーティションを使う理由

- ユーザやアプリのデータを OS のファイルと **分離** する
- OS とマシン（ハードウェア）間で情報を **共有** する
- システムの領域毎に異なるサイズ制限や権限を与えて、**セキュリティ** を拡張する
- 一時変数や揮発性データを通常データと分割し **サイズ** を有効利用する
- 利用頻度の高いデータを高速ストレージへ配置し **性能** を向上させる
- データ退避用の **スワップ** 領域、ハイバネーションストレージにもなる

何を分割し、パーティションをどのように分離するか、よく考えて決定しなければなりません。個別のパーティションを持つ理由は、セキュリティ、クォータ設定、サイズ制限の細分化があります。データ保護のために、パーティションを分けることもできます。

一般的なパーティション レイアウトには、`/boot` パーティション、ルートファイルシステム `/` のパーティション、スワップパーティション、および `/home` ディレクトリツリーのためのパーティションが含まれます。

インストール時に、パーティション作成した後でサイズを変更するのは難しいことを念頭に置いて計画しましょう。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.4 パーティション テーブル

MBR パーティション テーブル

- パーティション テーブルは、ディスクの最初のセクターにある 512 バイトの **MBR** (**M**aster **B**oot **R**ecord) の後半の 64 バイトの領域に置かれる
- パーティション テーブルのエントリは 16 バイト長、4 つのプライマリ パーティションのうちの 1 つを記述する
  - アクティブビット
  - パーティションの開始セクターのアドレス。シリンダ／ヘッド／セクター (**CHS**) 形式
  - パーティションの種類のコード: **Linux**、**Linux LVM**、**ntfs**、**swap**
  - **CHS** 形式のパーティションの最後のセクターのアドレス
  - 開始セクターのアドレス、0 からの通し番号
  - パーティション内のセクターの数

Linux は、リニアブロックアドレス (**LBA 方式**) を使用、最後の 2 つのフィールドだけでアドレスを指定する

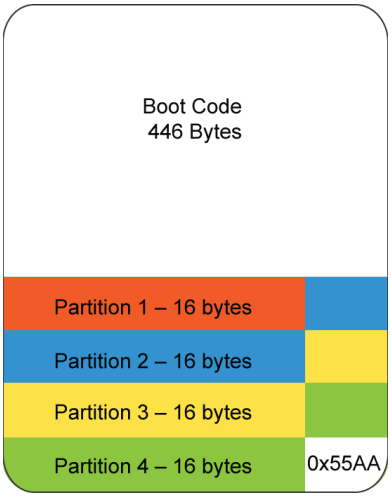


図 20.2: **MBR ディスクパーティション テーブル**

ディスクパーティション テーブルは、ディスクのマスター ブートレコード（**MBR**）に含まれており、446 バイトのブートレコードに続く 64 バイトがそれにあたります。

ディスク上の 1 つのパーティションが、アクティブとしてマークされることがあります。システム起動時にマスター ブートローダーは、アクティブとしてマークされたパーティションからロードするアイテムを見つけます。

拡張パーティションを 1 つ定義でき、そこには複数の論理パーティションを含めることができることを思い出してください。

MBR の構造は、オペレーティング システム非依存に記述されます。最初の 446 バイトはプログラムコード用に予約されています。通常、ここにブートローダー プログラムの一部を保持します。次の 64 バイトは、最大 4 エントリのパーティション テーブルです。オペレーティング システムは、このテーブルを使ってハードディスクを処理します。

Linux システムでは **CHS** の開始アドレスと終了アドレスは無視されます。

（参考：**MBR** の末尾 2 バイトにはマジックナンバー、シグネチャワード、または、セクターマーカーという名前で呼ばれる領域があり、必ず `0x55AA` の値を持っています）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

GPT パーティション テーブル

- 最新のハードウェアは **GPT** をサポート、**MBR** サポートは徐々に減っている
- GPT の Protective MBR には、UEFI システムに対する後方互換性があり、以前の方法でシステムを起動できる
- ディスクの先頭と末尾に **GPT** ヘッダーを保存、以下のメタデータを記述する
  - 使用可能なブロックのリスト
  - パーティションの数
  - パーティション エントリのサイズ
- 各パーティションエントリの最小サイズは 128 バイトである

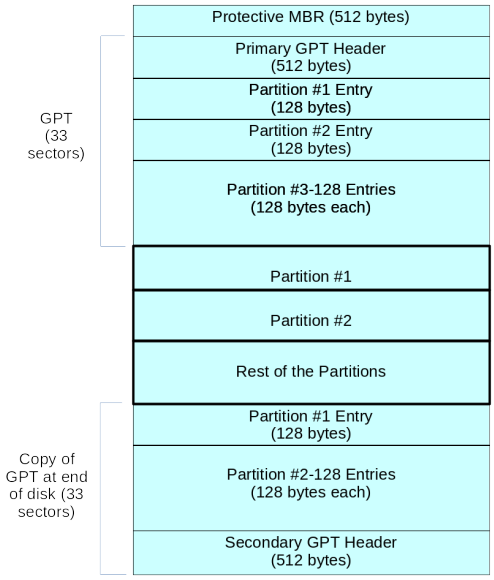


図 20.3: **GPT パーティション テーブル**

blkid ユーティリティ（後で説明します）は、パーティションに関する情報を表示します。

- 最新の UEFI/GPT システムの場合::

```
ROOT@x7:/root>blkid /dev/sda6
/dev/sda6: LABEL="CENTOS7" UUID="77461ee7-c34a-4c5f-b0bc-29f4feecc743" TYPE="ext4" \
    PARTUUID="1f361af4-81e6-4a81-8271-7b5dc305fc1e"
ROOT@x7:/root>
```

- レガシー MBR システムの場合:

```
c7:/teaching/LFCW/LFS301>sudo blkid /dev/sdb1
/dev/sdb1: LABEL="RHEL7" UUID="471dfeba-3ec7-4529-8069-2afe50762c57" TYPE="ext4"
```

どちらの例も、パーティション自体ではなく、パーティション上のファイルシステムを表す一意の UUID を提供しています。ファイルシステムが再フォーマットされると、この UUID も変更されます。

GPT パーティションには、パーティションを識別する PARTUUID も含まれます。この値は、ファイルシステムが再フォーマットされても変わりません。

ハードウェアがサポートしていれば、**MBR** システムを **GPT** に移行可能ですが、その際マシンが操作不能なることもありえます。

それを考えると、リスクを犯してまで挑戦するメリットはないでしょう。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.5 ディスクデバイスの命名

ディスクデバイス と デバイスノードの命名

- デバイスファイルは /dev に置かれている
- ファイル名のフォーマット: xxy[z]
  - xx はデバイスタイプを示す (通常 sd)
  - y はデバイス番号を示す (a,b,c、など)
  - z はパーティション番号
- 例
  - /dev/sda は最初に見つかったドライブである
  - /dev/sdb2 は 2 番目のドライブの 2 番目のパーティションである

Linux カーネルは、/dev ディレクトリにある **デバイスノード** を介して、ディスクとの低レベルでアクセスを行います。通常デバイスノードは、カーネルの **仮想ファイルシステム** のインフラを介してだけアクセスされます。直接デバイスノードを操作する低水準 (raw) アクセスを行うと、ファイルシステムは簡単に破壊されてしまいます。

例えば、次のようにパーティションをフォーマットするとそうなります。

```
$ sudo mkfs.ext4 /dev/sda9
```

SCSI と **SATA** ディスクのデバイスノードは、単純な命名規則を採用しています。

- 最初のハードディスクは /dev/sda となります。
- 2 番目のハードディスクは /dev/sdb となります。

パーティションも、次のように簡単に番号付けされます。

- /dev/sdb1 は、2 番目のディスクの最初のパーティション
- /dev/sdc4 は、3 番目のディスク上の 4 番目のパーティション

上記で sd は **SCSI** または **SATA** ディスクを意味します。**IDE** ディスクがあった時代には /dev/hda3、/dev/hdb などと命名されていました。

ls -l /dev を実行すると、現在利用可能なディスクデバイスノードが表示されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.6 blkid と lsblk

blkid

- ブロックデバイスを見つけて、その属性をレポートするユーティリティである
- ブロックデバイスの一覧を表示する
- ブロックデバイスのアトリビュート（属性）を表示する
- コマンドの例:

```
$ sudo blkid
$ sudo blkid /dev/sda*
$ sudo blkid -L root
```

blkid は、ブロックデバイスを見つけてその属性をレポートするユーティリティ **libblkid** ライブラリで動作します。特定デバイス、またはデバイスのリストを引数に指定します。

blkid は、ブロックデバイスの種類（ファイルシステム、スワップなど）やコンテンツメタデータ（LABEL または UUID フィールドなど）から、デバイスの属性（トークン、NAME=value ペア）を特定します。

blkid は、フィンガープリント可能なデータを含むデバイスでだけ動作します。たとえば、空のパーティションではブロックを識別する **UUID** は生成されません。

blkid には、 2 種類のコマンド形式があります。特定の `NAME=value` を持つデバイスを検索するか `NAME=value` ペアを持つ複数のデバイスを表示します。

引数がなければ、全てのデバイスについてレポートします。表示デバイスの指定とレポート属性の指定には、かなりの数のオプションがあります。

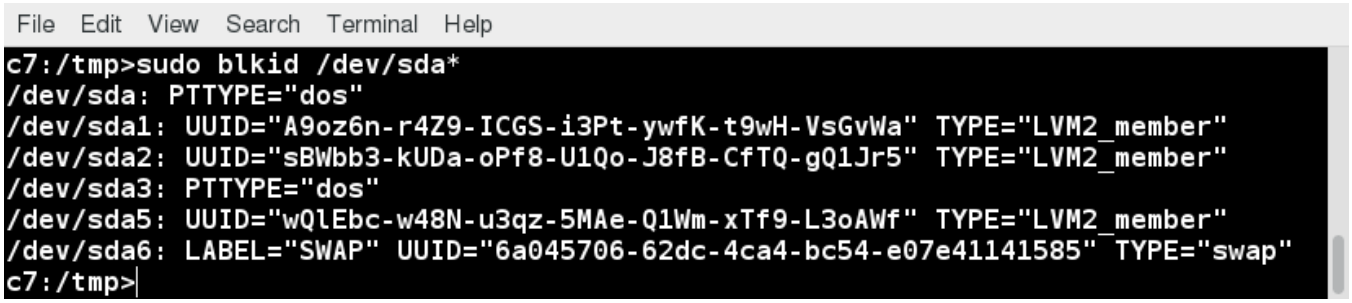


図 20.4: **blkid の使用例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

lsblk

- ツリー形式で、ブロックデバイスを表示する

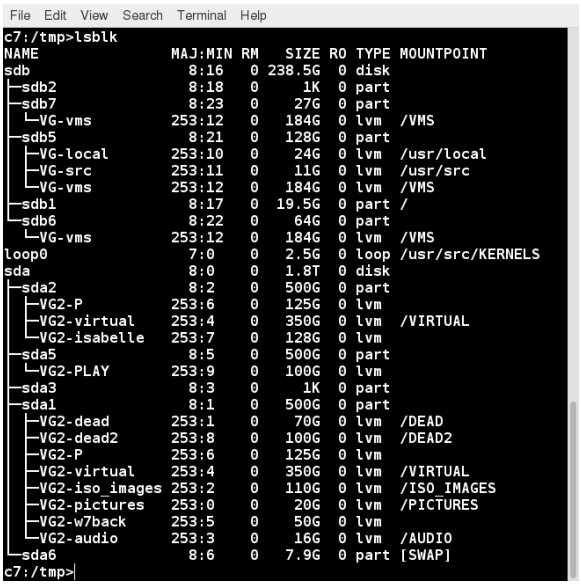


図 20.5: **lsblk の使用例**

```
$ lsblk -h
```

```
Usage:
 lsblk [options] [<device> ...]

List information about block devices.

Options:
 -a, --all            print all devices
 -b, --bytes          print SIZE in bytes rather than in human readable format
 -d, --nodeps         don't print slaves or holders
 -D, --discard        print discard capabilities
 -z, --zoned          print zone model
 -e, --exclude <list> exclude devices by major number (default: RAM disks)
 -f, --fs             output info about filesystems
 -i, --ascii          use ascii characters only
 -I, --include <list> show only devices with specified major numbers
 -J, --json           use JSON output format
 -l, --list           use list format output
 -T, --tree           use tree format output
 -m, --perms          output info about permissions
 -n, --noheadings     don't print headings
 -o, --output <list>  output columns
 -O, --output-all     output all columns
 -p, --paths          print complete device path
 -P, --pairs          use key="value" output format
 -r, --raw            use raw output format
 -s, --inverse        inverse dependencies
 -S, --scsi           output info about SCSI devices
 -t, --topology       output info about topology
 -x, --sort <column>  sort output by <column>

 -h, --help           display this help
 -V, --version        display version

 Available output columns:
 ......
 For more details see lsblk(8).
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION | Training & Certification

20.7 パーティションのサイズ変更

パーティションのサイズ変更

- 多くの **Linux** システムは、少なくとも 2 つのパーティションを使用する

  root (/)
  - ファイルシステムが利用する
  - ほとんどの **Linux** インストールは、複数のファイルシステムを持つ
  - サイズ変更はかなり困難である（可能ならば **LVM** を利用する）

  Swap
  - 物理メモリーの拡張として使用される
  - 物理メモリー（**RAM**）のサイズと同じにするのが一般的である
  - 複数のスワップパーティションやスワップファイルもありえる
  - ディスクが一台のシステムなら、スワップ パーティションは集約する
  - 複数ディスクのシステムなら、スワップ パーティションは分散させる
- **Ubuntu** を含む最近のディストーションの中には、デフォルトでスワップをパーティションではなくファイルの中に配置している
  - フレキシブル (例えば、リサイズが簡単にできる)
  - ただし、エラーやバグが破壊を拡散させた場合は、より危険な状態になる可能性がある

ほとんどのインストールでは、複数のパーティションに複数のファイルシステムがあり、**マウントポイント**で結ばれています。

ほとんどのファイルシステムでは、ルート パーティション サイズの変更は困難です。後述する **LVM** では、これが簡単にできます。

ルート パーティションだけで **Linux** を実行することも可能なのですが、ほとんどのシステムはより多くのパーティションを使用してバックアップの容易化、ディスクドライブのより効率的な使用、セキュリティの強化を図っています。

スワップサイズの推奨値は物理メモリーのサイズと同じですが、場合によっては 2 倍にすることが推奨されることもあります。正しい選択は、システムの利用シナリオ と ハードウェアの性能 などに依存します。

スワップ領域をどれだけ追加しても、どこかで限界はあるので必ずしも万能ではありません。メモリーの追加やシステムセットアップの再検討が必要な場合もあるでしょう。

古い回転式ハードディスクドライブでは、スワップパーティションを別に持つことがより理にかなっているかもしれませんが、**SSD** タイプのメディアでは、これは重要ではありません。しかし、それでも、より低速でおそらく安価なハードウェアにスワップを置きたいと思うかもしれません。これは、パーティションでもファイルでも同じことで、より一般的な選択肢になりつつあります。

いくつかのディストリビューションは（オプションで）**zram**（www.kernel.org/doc/html/latest/adminguide/blockdev/zram.html）を使用するようになっています。これは、スワップにディスクストレージの代わりに、圧縮メモリを使用します。これはメモリー不足に陥りやすいですが、専門家の手にかかればパフォーマンスを向上させることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.8 パーティション テーブル の バックアップと復元

パーティション テーブル の バックアップと復元

- **MBR**
  - **dd** プログラムを利用する
  - 次のコマンドで（パーティション テーブルと一緒に）**MBR** をバックアップする

    ```
    $ dd if=/dev/sda of=mbrbackup  bs=512 count=1
    ```
  - **MBR** を復元する

    ```
    $ sudo dd if=mbrbackup of=/dev/sda bs=512 count=1
    ```
- **GPT**
  - **sgdisk** を利用する（**sgdisk** は **MBR** システムに対しても利用可能）

    ```
    $ sudo sgdisk --backup=gptbackup /dev/sda
    $ sudo sgdisk --load-backup=gptbackup /dev/sda
    ```

ディスクパーティション テーブルの変更によって、ファイルシステムの全データが削除されてしまう可能性があることを常に想定してください。（削除されるべきではありませんが、注意は必要です！）

したがって、変更などの作業を行う前に（まだバックアップされていない）全てのデータをバックアップするのが賢明です。**MBR** システムでは **dd** を使って変換やコピーができます。**dd** では、単純な入力ミスやオプションの誤用によってディスク全体を破壊する可能性があるので注意してください。**dd** はプライマリーパーティションテーブルだけをコピーします。他のパーティション（例えば拡張パーティションなど）に含まれるパーティション テーブルは対象にしません。

GPT システムの場合次のように **sgdisk** ツールを使用するのが最適です。

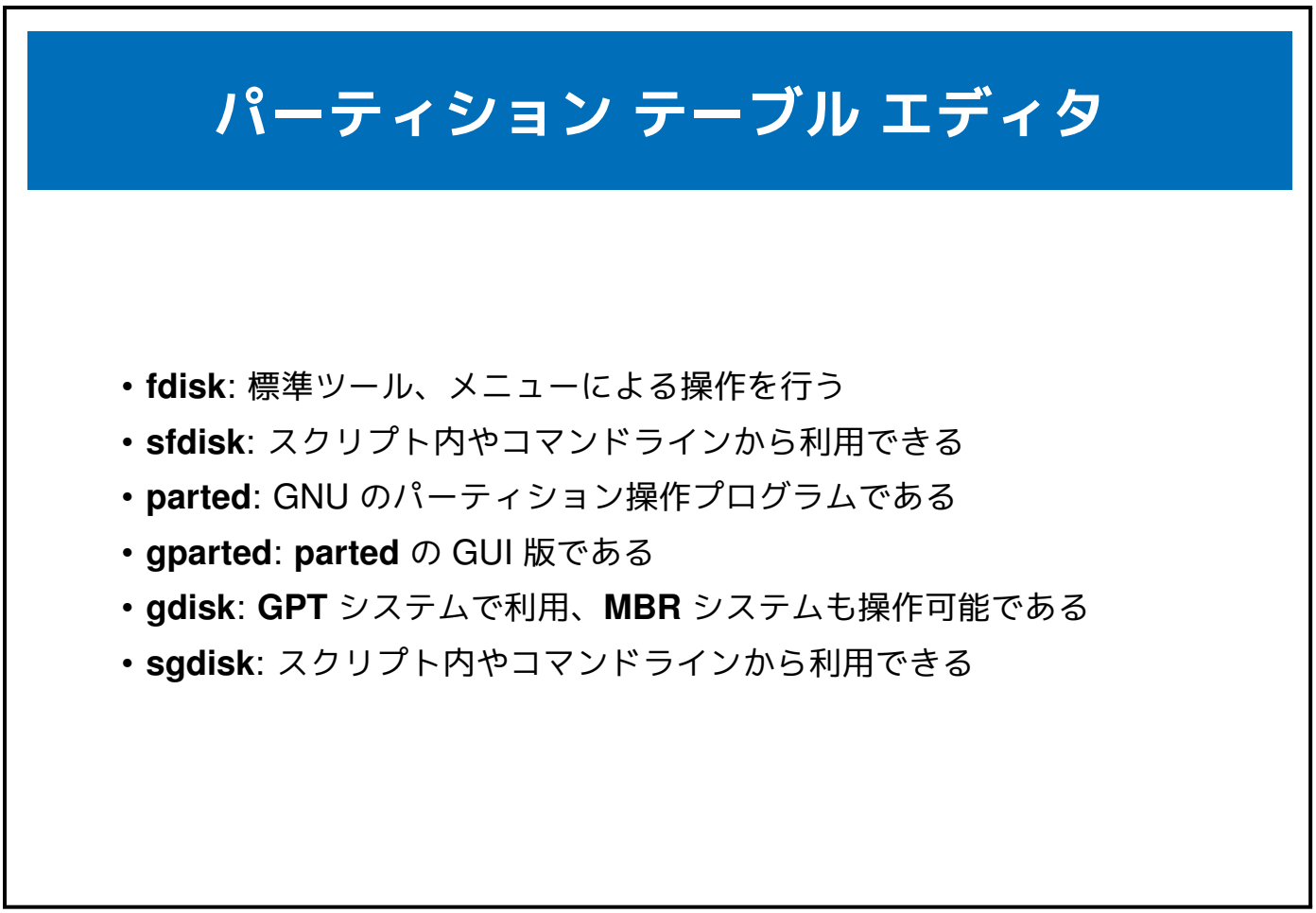
```
x8:/tmp>sudo sgdisk -p /dev/sda
Disk /dev/sda: 1000215216 sectors, 476.9 GiB
Model: SAMSUNG MZNLN512
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): DBDBE747-8392-4B62-97AA-6214490073DC
Partition table holds up to 128 entries
....
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048          534527   260.0 MiB   EF00  EFI System Partition
   2          534528          567295   16.0 MiB    0C01  Microsoft reserved ...
....
```

純粋な **MBR** システムの場合には例のようなメッセージが表示されます。

```
c8:/tmp>sudo sgdisk -p /dev/sda
****************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format
in memory.
****************************************************************
Disk /dev/sda: 500118192 sectors, 238.5 GiB
Model: Crucial_CT256MX1
Sector size (logical/physical): 512/4096 bytes
....
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.9 パーティション テーブル エディタ



fdisk は、メニュー方式のパーティション テーブルエディタです。これは、最も一般的で最も柔軟なパーティション テーブルエディタの１つです。他のパーティションテーブルエディタと同様、変更を行う前に現在のパーティション テーブル設定を書き留めるか、現在の設定のコピーを作成してください。

sfdisk は、非対話型の **Linux** ベースのパーティション エディタ プログラムで、スクリプトで役立ちます。**sfdisk** ツールは注意して使用してください！

GPT システムでは、**gdisk** と **sgdisk** が同様の役割を果たします。どちらも **MBR** システムも操作できます。

parted は、GNU のパーティション操作プログラムです。パーティション（特定のファイルシステムを含む）の作成、削除、サイズ変更、移動 ができます。**parted** コマンドの GUI インターフェイス版が **gparted** です。

多くの **Linux** ディストリビューションには、CD-ROM または USB スティックから実行できる **ライブ／インストールバージョン** があります。通常これらのメディアには **gparted** が含まれているので、パーティション プログラムを実行していないディスク（＝システムのライブ起動に使っていない元々の起動ドライブなど）をグラフィカルパーティション ツールを使って簡単に操作できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.10 fdisk

fdisk の使用例

- どの Linux インストールにも常に含まれ、簡単に利用できる
- メニュー操作方式となっている、プロンプトに答えていけばよい
  - m: メニューを表示
  - p: パーティション テーブルの一覧を表示
  - n: 新しいパーティションを作成
  - d: パーティションを削除
  - t: パーティションの種類を変更
  - w: 新しいパーティション テーブル情報を書き込んで終了
  - q: 変更せずに終了
- パーティション テーブルの編集

```
$ sudo fdisk /dev/sda
```

- パーティション テーブルの表示だけで変更は何もしない

```
$ sudo fdisk -l /dev/sda
```

fdisk の実行には、root 権限が必要です。処理は少し複雑になる可能性があり、注意が必要です。**fdisk** はメニュー駆動式のインターフェイスを持っています。まず、特定のディスクで起動します。

```
$ sudo fdisk /dev/sdb
```

上に列挙した一文字のコマンドを使って操作します。

幸いなことに、w を入力してパーティション テーブルをディスクに書き込むまで、実際の変更は行われません。したがって、w を使用してディスクに書き込む前に（p を使用して）パーティション テーブルが正しいことを確認することが重要です。何かが間違っていた場合 q を使用すれば安全に終了できます。

システムを再起動するまで新しいパーティション テーブルは使用されません。ただし、次のコマンドで再読み込みできます。

```
$ sudo partprobe -s
```

を実行して、修正されたパーティション テーブルを試してください。ただし、これは常に確実に機能するとは限りません。パーティションの混在は壊滅的なものになる可能性があるため、新しいパーティションのフォーマットなどの前には再起動をお勧めします。

オペレーティング システムが、現在どのパーティションを認識しているかを確認する次のコマンドは、いつでも利用可能です。

```
$ cat /proc/partitions
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

20.11 演習

デモ教材ビデオ

using_fdisk_demo.mp4

課題 20.0: ループバックデバイスを使って、パーティションのエミュレーションをする

本セクション中の演習では **mkfs** を使いファイルシステムのフォーマットと、**mount** を使いルートファイルシステムの階層構造中にマウントを行います。これらのコマンドの詳細は次のセクションで説明します。

最初の 3 つの演習では **parted** プログラムを使う場合と使わない場合の両方で **ループデバイス** 機構を利用します。

重要

本コースの後半の演習では、パーティション分けされていないディスク領域が必要になります。1〜2GB 程度とそれほど大きな領域は必要ありません。

自分のマシンを使っている場合、その領域があるか確認してください。もし無ければ **gparted** を使うか、説明した/する方法で、パーティションとファイルシステム (こちらを先に！) を縮小して、空き領域を用意してください。

注目

もしディスク領域が物理的に空いている場合は、以下の手順を踏む **必要はありません**。しかし実習しておくことは有益です。

課題 20.1: ファイルをディスク パーティション イメージとして使う

最初の演習では、ハードディスクの完全なパーティション イメージを持つファイルを作成し、実際のハード パーティションと同様に扱えるようにします。その後の演習で、複数のパーティションを格納したり、完全なディスクのように振る舞えるようにします。

1. 大きさが 1GB の、ゼロで満たされたファイルを作成します。

   ```
   $ dd if=/dev/zero of=imagefile bs=1M count=1024
   ```

   もしパーティションに余裕が無ければ、小さいファイルでも構いません。

2. ファイルシステムを作成します:

   ```
   $ mkfs.ext4 imagefile
   ```

   ```
   mke2fs 1.42.9 (28-Dec-2013)
   imagefile is not a block special device.
   Proceed anyway? (y,n) y
   Discarding device blocks: done
   .....
   ```

   mkfs.ext3、**mkfs.vfat**、**mkfs.xfs** などを使って、別のファイルシステムにフォーマットしても構いません。

3. 適当な場所にマウントします。

   ```
   $ mkdir mntpoint
   $ sudo mount -o loop imagefile mntpoint
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

これで、ファイルを格納したりするなど普通に利用できるようになりました。

4. 終了したらアンマウントします。

```
$ sudo umount mntpoint
```

loop オプションを用いる別の方法でマウントします。

```
$ sudo losetup /dev/loop2 imagefile
$ sudo mount /dev/loop2 mntpoint
 ....
$ sudo umount mntpoint
$ sudo losetup -d /dev/loop2
```

この後の演習で **losetup** について説明します。/dev/loop[0-7] を使えるようになりますが、これらはすでにあまり使われなくなっているので注意が必要です。

実パーティションの代わりにループデバイスファイルを用いる方法は便利ですが、性能測定やベンチマーキングにはふさわしくありません。なぜなら、ファイルシステム層を別のファイルシステム上に配置することになるからです。そのため性能面では悪い影響があり、ファイルシステムイメージが置かれた下位のファイルシステムの動作の影響を受けるからです。

課題 20.2: ディスクイメージファイルをパーティション分割する

次のステップではイメージファイルを複数のパーティションに分割し、各々のパーティションをファイルシステムやスワップとして利用できるようにします。

先の演習で作成したイメージファイルを使うか、新規に作成してもかまいません。

1. イメージファイルに対して **fdisk** を実行します。

```
$ sudo fdisk -C 130 imagefile
```

```
Device does not contain a recognized partition table
Building a new DOS disk label with disk identifier 0x6280ced3.
Welcome to fdisk (util-linux 2.23.2).

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

Command (m for help):
```

-C 130 により、ドライブ中の疑似シリンダ数を 130 に設定します。これは **RHEL 6** に含まれるような古いバージョンの **fdisk** にだけ必要です。しかし（この設定をしても）他のディストリビューションには悪影響を与えません。

2. m を入力してコマンド一覧を表示します。

```
m
```

```
Command (m for help): m
Command action
a   toggle a bootable flag
b   edit bsd disklabel
c   toggle the dos compatibility flag
d   delete a partition
g   create a new empty GPT partition table
G   create an IRIX (SGI) partition table
l   list known partition types
m   print this menu
n   add a new partition
o   create a new empty DOS partition table
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
p   print the partition table
q   quit without saving changes
s   create a new empty Sun disklabel
t   change a partition's system id
u   change display/entry units
v   verify the partition table
w   write table to disk and exit
x   extra functionality (experts only)

Command (m for help):
```

3. 大きさ 256MB（他でも構いません）のプライマリ パーティションを作成します。

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

```
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): +256M
Partition 1 of type Linux and of size 256 MiB is set
```

4. もうひとつ 256MB のプライマリ パーティションを追加します。

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

```
Partition type:
p   primary (1 primary, 0 extended, 3 free)
e   extended
Select (default p): p
Partition number (2-4, default 2): 2
First sector (526336-2097151, default 526336):
Using default value 526336
Last sector, +sectors or +size{K,M,G} (526336-2097151, default 2097151): +256M
Partition 2 of type Linux and of size 256 MiB is set

Command (m for help): p

Disk imagefile: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x6280ced3

Device Boot      StartEnd      Blocks   Id  System
imagefile1          2048      526335      262144   83  Linux
imagefile2          526336     1050623      262144   83  Linux
```

5. ディスクにパーティション テーブルを書き込み、終了します。

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

```
The partition table has been altered!

Syncing disks.
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

この演習はうまく行きましたが、ここで作成した 2 つのパーティションを利用する方法がまだわかりません。これを次の演習で説明します。

課題 20.3: losetup と parted を使う

ここでは、さらに次のことを実験します。

- ループ デバイスと **losetup**
- 非インタラクティブ（バッチ式）でパーティション分割するために **parted** を使う。

以降の手順を実行する前に **losetup** と **parted** の **man ページ**を読んでおいてください。

以前作成したイメージ ファイルそのままか、ゼロクリアしたもの、または新規に作成しても構いません。

1. これから使う **ループ** デバイスを認識します。

```
$ sudo losetup -f
```

```
/dev/loop1
```

```
$ sudo losetup /dev/loop1 imagefile
```

最初のコマンドは、**空いている** 最初のループデバイスを探します。すでにいくつかのループデバイスを利用している可能性があるからです。たとえば、上記のコマンドを実行する前にシステムが使っているかもしれません。

```
$ losetup -a
```

```
/dev/loop0: []: (/usr/src/KERNELS.sqfs)
```

圧縮された **squashfs** が /dev/loop0 を使って、読み出し専用でマウントされています。（コマンドの出力は、ディストリビューションによって異なります）この結果を無視して **losetup** を使って /dev/loop0 を操作すると、ファイルは破壊されてしまいます。

2. ループデバイス（イメージファイル）に、ディスク パーティション ラベルを作成します。

```
$ sudo parted -s /dev/loop1 mklabel msdos
```

3. ループデバイスに 3 つのプライマリ パーティションを作成します。

```
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 0 256
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 256 512
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 512 1024
```

4. パーティション テーブルを確認します。

```
$ fdisk -l /dev/loop1
```

```
Disk /dev/loop1: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x00050c11

      Device Boot      Start         End      Blocks   Id  System
/dev/loop1p1               1      500000      250000   83  Linux
/dev/loop1p2          500001     1000000      250000   83  Linux
/dev/loop1p3         1000001     2000000      500000   83  Linux
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5. 使用しているディストリビューションによって多少異なります。たとえば **RHEL** と **Ubuntu** では新しくデバイスノードが作成されます。

```
$ ls -l /dev/loop1*
```

```
brw-rw---- 1 root disk   7, 1 Oct  7 14:54 /dev/loop1
brw-rw---- 1 root disk 259, 0 Oct  7 14:54 /dev/loop1p1
brw-rw---- 1 root disk 259, 3 Oct  7 14:54 /dev/loop1p2
brw-rw---- 1 root disk 259, 4 Oct  7 14:54 /dev/loop1p3
```

次のように使います。

6. パーティションにファイルシステムを格納します。

```
$ sudo mkfs.ext3 /dev/loop1p1
$ sudo mkfs.ext4 /dev/loop1p2
$ sudo mkfs.vfat /dev/loop1p3
```

7. ３つのファイルシステムを全てマウントし、使えることを確認します。

```
$ mkdir mnt1 mnt2 mnt3

$ sudo mount /dev/loop1p1 mnt1
$ sudo mount /dev/loop1p2 mnt2
$ sudo mount /dev/loop1p3 mnt3

$ df -Th
```

```
Filesystem              Type      Size  Used Avail Use% Mounted on
/dev/sda1               ext4       29G  8.5G   19G  32% /
....
/dev/loop1p1            ext3      233M  2.1M  219M   1% mnt1
/dev/loop1p2            ext4      233M  2.1M  215M   1% mnt2
/dev/loop1p3            vfat      489M     0  489M   0% mnt3
```

8. ファイルシステムの利用が終了したら、元に戻します:

```
$ sudo umount mnt1 mnt2 mnt3
$ rmdir mnt1 mnt2 mnt3
$ sudo losetup -d /dev/loop1
```

課題 20.4: 実ハードディスクのパーティションを分割する

パーティションを作成していない、空き領域を持つハードディスクがあれば **fdisk** を使って、プライマリ パーティション、拡張パーティションを作成してロジカルパーティションを新規に作成してください。作成したパーティション テーブルをディスクに書き込み、フォーマット、マウントしてください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 21 章

ファイルシステム機能：属性, 作成, 検査, マウント

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- **lsattr** と **chattr** を使用した拡張ファイル属性の使用方法について説明する。
- **mkfs** と **fsck** でファイルシステムを作成し、フォーマットする。
- ファイルシステムの問題をトラブルシュートする。
- ファイルシステムのエラーをチェックし、修正する。
- ユーティリティの **df** と **du** を使用してディスクスペースの使用状況を確認する。
- ファイルシステムのマウントとアンマウントを行う。
- **NFS** などのネットワークファイルシステムを利用する。
- ファイルシステムを **automount（自動マウント）**する方法を理解し、ブート時にマウントされる。
- **ネットワーク ブロック デバイス（NBD）**を理解し、使用する。

21.1 拡張属性

lsattr と chattr

- ファイルに拡張属性を設定できる
  - `i`: 変更不可
  - `a`: 追加だけが許可
  - `d`: ダンプなし
  - `A`: atime の更新なし
- 属性フラグの値は **lsattr** で表示する
- 属性フラグの値は **chattr** で設定する
- 属性フラグの値は、ファイルの inode フラグに記録される
- ユーザーだけが変更および設定できる

拡張属性は、ファイルシステムが直接解釈しないメタデータをファイルに結びつけるものです。この拡張属性の管理には、4 種類の **namespaces (名前空間)** を使っています。それは、user, trusted, security, system です。system 名前空間は **アクセス制御リスト（ACL）**に使用され、security 名前空間は **SELinux** によって使用されます。

属性フラグの値は、ファイルの inode に保存され root ユーザーだけが変更および設定できます。それらは **lsattr** で表示され **chattr** で設定されます。属性フラグはファイルに設定できます。

- `i` (**変更不可 (immutable)**): immutable 属性を持つファイルは変更できません。（root ユーザでも変更できません）削除や名前の変更もできません。ハードリンクも作成できず、ファイルにデータを書き込むこともできません。この属性を設定または削除できるのはスーパーユーザーだけです。
- `a` (**追加だけが許可 (append-only)**): Append-only 属性が設定されたファイルは、書き込みする場合は追加モードでだけ開くことができます。この属性を設定または削除できるのはスーパーユーザーだけです。
- `d` (**ダンプなし (no-dump)**): No-dump 属性が設定されたファイルはダンプの対象外となります。バックアップが無意味なスワップファイルとキャッシュファイルに指定すると、無駄なダンプ処理を回避できます。
- `A` (**atime の更新なし (No atime update)**): No-atime-update 属性が設定されたファイルは、ファイルがアクセスされても変更されなかったときは **atime**（アクセス時間）レコードを変更しません。これにより、システム上のディスク I/O の量が減少するため、一部のシステムのパフォーマンスが向上します。

設定できるフラグは他にもあります。`man chattr` と入力するとリスト全体が表示されます。

chattr の書式は次のとおり:

```
$ chattr [+|-|=mode] filename
```

lsattr ファイルの属性を表示します:

```
$ lsattr filename
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.2 ファイルシステムの作成とフォーマット

mkfs

- **mkfs** の一般的なコマンド形式

```
mkfs [-t fstype] [options] [device-file]
```

- [device-file] は /dev/sda3 や /dev/vg/lvm1 等デバイス名
- 各ファイルシステム毎に固有なフォーマットオプションがある
- 各ファイルシステムは独自の **mkfs** プログラムを持っている
- 等価なコマンドの例

```
$ sudo mkfs -t ext4 /dev/sda10
$ sudo mkfs.ext4 /dev/sda10
```

全てのファイルシステムは、パーティション上のファイルシステムを **フォーマット**（作成）するためのユーティリティを持っています。そのようなユーティリティの一般的な名前は、**mkfs** ですが **mkfs** は各ファイルシステムに固有なフォーマットプログラムのフロントエンドです。ファイルシステムに固有なフォーマットプログラムには、特有なオプションがある場合があります。

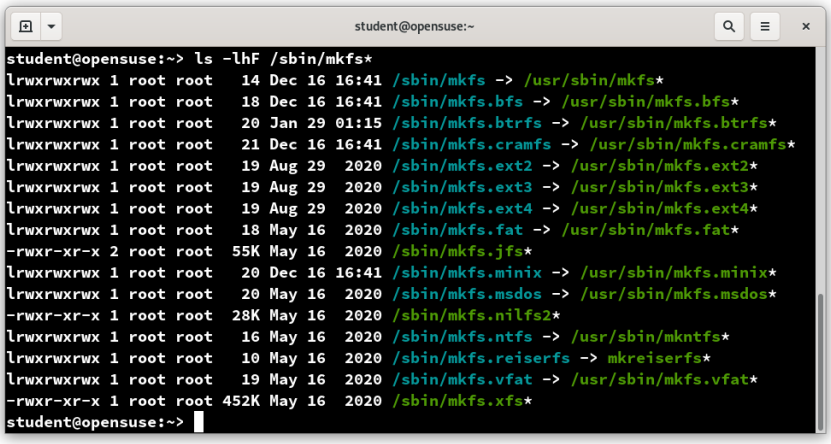


図 21.1: **mkfs**

それぞれの **mkfs.*** プログラムの詳細は **man** ページを参照してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.3 ファイルシステムのトラブルシューティング

ファイルの整合性をチェックする

- 設定ファイルやバイナリが破損していないか確認する
- 多数のメソッドを使用できる
  - **rpm**
    `$ rpm -V some_package`
    特定のひとつのパッケージをチェックする
    `$ rpm -Va`
    全てのパッケージをチェックする
  - **aide**: 侵入検知
    `$ aide --check`
  - **debsums**: **Debian**
    `$ debsums options some_package`

破損した設定ファイルやバイナリをチェックする方法はいくつかあります。1 つの方法は、特定のパッケージをチェックするために `rpm -V` コマンドを使用することです。また、`rpm -Va` を使用して、すべてのパッケージの完全性をチェックすることができます。

`aide` を使うのも、ファイルの変更をチェックする方法の一つです。`aide -check` というコマンドは、ファイルのスキャンを実行し、前回のスキャンと比較します。

`Debian` では、整合性チェックを行う唯一の方法は、`debsums` を使うことです。`debsums somepackag` を実行すると、そのパッケージのファイルのチェックサムをチェックします。しかし、すべてのパッケージのチェックサムを保持しているわけではないので、この方法は常に有効とは限りません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ファイルシステムの破損と復旧

- **fsck** は破損したファイルシステムの修正に使用されることがある
- /etc/fstab に設定ミスがないか確認する
- 修復を試みるために、以下を試す必要があるかもしれない

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

  to attempt fixes
- ファイルシステムの手動マウントを試みる

fsck を使って修復を試みたいと思うかもしれません。しかし、その前に **/etc/fstab** に設定ミスや破損がないことを確認する必要があります。実行中のカーネルがそのファイルシステムタイプを認識できないという可能性も忘れずに考慮してください。

ルートファイルシステムがマウントされていれば、このファイルを調べることができますが、/ は読み取り専用にマウントされている可能性があります。ファイルを編集して修正するには、次のコマンドを実行して書き込み許可で再マウントします。

```
$ sudo mount -o remount,rw /
```

もし /etc/fstab に間違いが無いようであれば **fsck** を試すことができます。まず以下のコマンドを実行して全てのファイルシステムのマウントを試してください。

```
$ sudo mount -a
```

全てのファイルシステムの自動マウントが成功しない場合は、問題のあるファイルシステムを手動マウントしてみることができます。**fsck** は、最初はエラー内容の確認のためだけに実行し、そこで見つかったエラーを修正するために（修正のオプションを付けて）再実行してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.4 ファイルシステムのチェックと修復

fsck

- 設定した累積利用時間、または、マウント回数に到達すると、自動的に実行する
- クリーンにアンマウント出来なかった時は、次回ブート時に実行する
- マウントされているオンラインのファイルシステムはチェックできない
- **fsck** の一般的なコマンド形式

```
fsck [-t fstype] [options] [device-file]
```

[device-file] は通常 /dev/sda3 や /dev/vg/lvm1 等のデバイス名となる

- パーティションのスーパーブロックを検証するので、通常はファイルシステムの指定は必要ない
- 等価なコマンドの例

```
$ sudo fsck -t ext4 /dev/sda10
$ sudo fsck.ext4 /dev/sda10
```

全てのファイルシステムは、エラーチェック（と可能なら検出されたエラーを修復する）ユーティリティを持っています。
これらのユーティリティの一般名は **fsck** ですが、これはファイルシステムに固有のチェックプログラムのフロントエンドです。

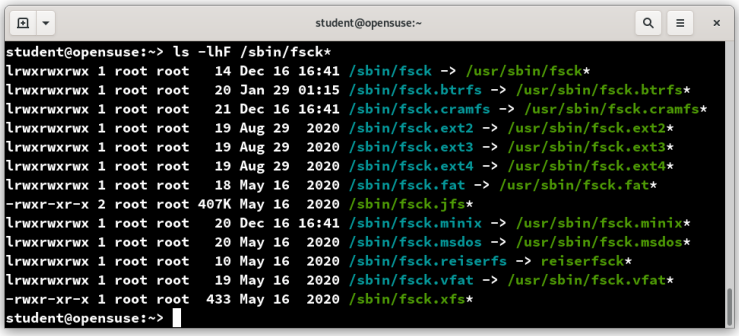


図 21.2: **fsck**

次回ブート時に、マウントされた全ファイルシステムを強制チェックするには

```
$ sudo touch /forcefsck
$ sudo reboot
```

チェックでエラーがでなければ、作成されたファイル /forcefsck は自動的に削除されます。

-r オプションを指定して、見つかったエラーを 1 つずつ手動で修正するか、-a オプションを使用して可能な限り最善の方法で自動修正を試みるかを選択できます。さらに各ファイルシステムには、独自のチェックを実行する **fsck** のオプションがある場合があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.5 ファイルシステムの使用量

df: ファイルシステムの使用量

- **df** (**d**isk **f**ree) は、ファイルシステムの使用量の確認に利用される
- ファイルシステムの使用量を表示する（デフォルトではキロバイト単位）

  `$ df`

  ファイルシステムの使用量を **人が読みやすい単位** で表示する

  `$ df -h`
- ファイルシステムのタイプを表示する

  `$ df -T`
- inode 情報を表示する

  `$ df -i`

```
x8:/tmp>df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  7.8G     0  7.8G   0% /dev
tmpfs          tmpfs     7.8G     0  7.8G   0% /dev/shm
tmpfs          tmpfs     7.8G  9.6M  7.8G   1% /run
tmpfs          tmpfs     7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/sda8      ext4       26G   10G   15G  42% /
/dev/sda7      vfat      200M   14M  187M   7% /boot/efi
/dev/sda5      ext4       15G  9.6G  4.1G  71% /UBUNTU
/dev/sda6      ext4      389G  252G  118G  69% /ALL
/dev/sda1      vfat      256M   43M  214M  17% /Cbootefi
tmpfs          tmpfs     1.6G  1.2M  1.6G   1% /run/user/42
tmpfs          tmpfs     1.6G  4.0K  1.6G   1% /run/user/1000
/dev/loop0     squashfs  2.5G  2.5G     0 100% /usr/src/KERNELS
x8:/tmp>
```

図 21.3: **df の使用例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.6 ディスクの使用量

du: ディスクの使用量

- du (**d**isk **u**sage) は、ディスクの使用量と用途を表示する
- カレントディレクトリの使用量を表示する

```
$ du
```

- ディレクトリだけではなく全てのファイルをリストする

```
$ du -a
```

- 人が読みやすい単位で表示する

```
$ du -h
```

- ディレクトリを指定してディスク使用量を表示する

```
$ du -h somedir
```

以下のコマンドを試してみてください。

```
$find . -maxdepth 1 -type d -exec du -shx {} \; | sort -hr
```

```
c8:/home/coop/.cache>find . -maxdepth 1 -type d -exec du -shx {} \; | sort -hr
338M    .
229M    ./google-chrome
86M     ./spotify
7.7M    ./tracker
6.8M    ./thunderbird
4.9M    ./gnome-software
2.5M    ./mesa_shader_cache
956K    ./gstreamer-1.0
436K    ./samba
52K     ./evolution
12K     ./fontconfig
8.0K    ./vmware
8.0K    ./Slack
8.0K    ./skypeforlinux
4.0K    ./libgweather
4.0K    ./googleearth_CACHE
4.0K    ./gnome-shell
4.0K    ./gnome-screenshot
c8:/home/coop/.cache>
```

図 21.4: **du の使用例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.7 ファイルシステムのマウント

ファイルシステムのマウント

- ファイルシステムを、指定したディレクトリにマウントする

```
$ mount -t ext4 /dev/sdb4 /home
```

  - **ext4** ファイルシステムをマウントする
  - 通常、-t でファイルシステムのタイプを指定する必要は無い
  - ファイルシステムはハードディスク上の決められたパーティション (/dev/sdb4) 上に配置される
  - ファイルシステムは、現在のディレクトリツリーの /home という場所にマウントされる
  - /home ディレクトリに最初からファイルがあった場合、それらのファイルはパーティションがアンマウントされるまで見えなくなる

Linux でアクセス可能な全てのファイルは、先頭をルートディレクトリ (/) とする１つの大きな階層ツリー構造になっています。２つ以上のパーティションが（各パーティションに別のファイルシステムを指定することも可能です）、同じファイルシステムツリーに統合されることも珍しくありません。これらのパーティションは、別の物理デバイス（＝ドライブ）上や、更にはネットワーク上にあっても良いのです。

mount プログラムを使用すると、ツリー構造の任意の場所にファイルシステムをマウントできます。**umount** は、それらのマウントを解除（アンマウント）します。

マウントポイント とは、ファイルシステムがアッタチされるディレクトリです。**mount** を実行する前にそのディレクトリが存在している必要があります。**mkdir** を使用して空のディレクトリを作成できます。既存のディレクトリをマウントポイントとして使用しようとした時に、もしそのディレクトリに先に何かファイルがあった場合には、そのファイルはマウント後に非表示となります。それらのファイルは削除されるのではなく、ファイルシステムがアンマウントされると再び表示されます。

スーパーユーザーだけが、ファイルシステムを マウント／アンマウント できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

マウント

- **mount** の一般的なコマンド形式

```
mount [options] <source> <directory>
```

- **デバイスノード**、**ラベル**、**UUID** を使った
  マウントが可能です

- ファイルシステム独自のオプションがある、
  共通オプションの例

```
$ sudo mount -o remount,ro /myfs
```

  リードオンリーでリマウント

```
File  Edit  View  Search  Terminal  Help
c:/tmp> mount --help
Usage:
 mount [-lhV]
 mount -a [options]
 mount [options] [--source] <source> | [--target] <directory>
 mount [options] <source> <directory>
 mount <operation> <mountpoint> [<target>]

Options:
 -a, --all               mount all filesystems mentioned in fstab
 -c, --no-canonicalize   don't canonicalize paths
 -f, --fake              dry run; skip the mount(2) syscall
 -F, --fork              fork off for each device (use with -a)
 -T, --fstab <path>      alternative file to /etc/fstab
 -h, --help              display this help text and exit
 -i, --internal-only     don't call the mount.<type> helpers
 -l, --show-labels       lists all mounts with LABELs
 -n, --no-mtab           don't write to /etc/mtab
 -o, --options <list>    comma-separated list of mount options
 -O, --test-opts <list>  limit the set of filesystems (use with -a)
 -r, --read-only         mount the filesystem read-only (same as -o ro)
 -t, --types <list>      limit the set of filesystem types
     --source <src>      explicitly specifies source (path, label, uuid)
     --target <target>   explicitly specifies mountpoint
 -v, --verbose           say what is being done
 -V, --version           display version information and exit
 -w, --rw, --read-write  mount the filesystem read-write (default)

 -h, --help     display this help and exit
 -V, --version  output version information and exit
...
c7:/tmp>
```

図 21.5: **mount**

以下は、全て同じマウント結果になります。

```
$ sudo mount /dev/sda2   /home
$ sudo mount LABEL=home  /home
$ sudo mount    -L home  /home
$ sudo mount UUID=26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
$ sudo mount   -U 26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
```

ラベルは、**e2label** などのファイルシステムの種類に固有のユーティリティによって割り当てられます。**UUID** は、パーティションがファイルシステムのコンテナとして作成され、**mkfs** でフォーマットされる時に割り当てられます。

これら３つのデバイス指定方法はいずれも使用できますが、最新のシステムではデバイスノード形式の使用は推奨していません。その理由は、システムの起動方法によってどのハードドライブが最初に見つかるかは一定でないため、デバイスの名前が変わってしまう可能性があるためです。

ラベルは改善されていますが、まれに２つのパーティションが同じラベルで作成される場合があります。一方で **UUID** はパーティションの作成時に作成され、常にユニークな値をとります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

現在マウントされているファイルシステム

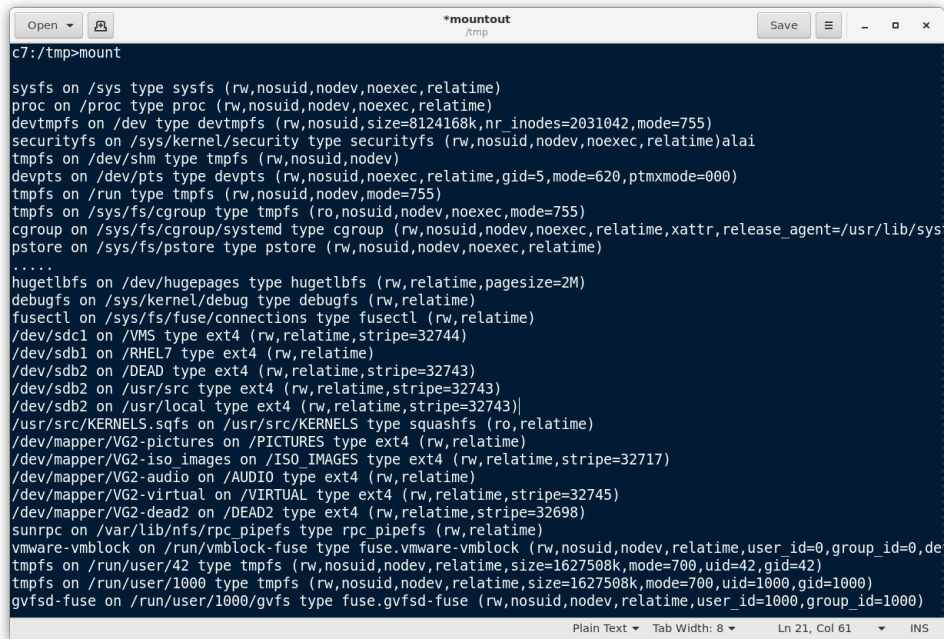


図 21.6: **現在マウントされているファイルシステム**

上の画面には 3 つの基本的なファイルシステムが表示されています。

- カーネル空間 - ブロック デバイスとして扱います

  ext4

- ユーザー空間 - ブロック デバイスとして扱います

  vmware-vmblock - VMWare のボリュームを共有するために使われる **FUSE** ファイルシステム
  gvfsd-fuse - **GNOME** のマウントを扱うための **FUSE** ファイルシステム

- 疑似 - ブロック デバイスではありません（しかし、一般には カーネル空間 に配置されています）

  sysfs、**procfs**、**devtmpfs**、**devtmpfs** - コア機能を提供しています
  debugfs、**hugetlbfs**、**cgroup**、**sunrpc** - ネットワーク ファイルシステムを含め、追加機能を提供しています

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

アンマウント

- ファイルシステムのアンマウントに使用する
- **umount** の一般的なコマンド形式

```
umount [device-file | mount-point]
```

- 例:
  - /home ファイルシステムのアンマウント

```
$ umount /home
```

  - /dev/sda3 デバイスのアンマウント

```
$ umount /dev/sda3
```

ファイルシステムをアンマウントするコマンドは **umount** です。(**unmount** ではありません！)

mount と同様に **umount** にも多くのオプションがあり、その多くはファイルシステムの種類に固有のものです。繰り返しになりますが、**man** ページは特定のオプションに関する最適な情報源です。

ファイルシステムをアンマウントするときに発生する最も一般的なエラーは、現在使用中のファイルシステムに対してアンマウントしようとすることです。つまり、アプリケーションがファイルシステム内のファイル、または他のエントリを使用している場合です。

このエラーは、マウントされたファイルシステムのディレクトリでターミナルウィンドウを開くのと同じくらい、簡単に解決できます。そのウィンドウで **cd** と打つか、アプリケーションを強制終了するだけで、デバイスがビジー状態であるというエラーが解消されアンマウントが可能になります。

ただし、このエラーを引き起こすプロセスが他にある場合、ファイルシステムをアンマウントする前にそれらを強制終了する必要があります。**fuser** でどのユーザーがファイルシステムを使用しているか確認して、そのユーザーを強制終了できます。(この操作には注意が必要です、強制終了する前にユーザーに警告すべきかもしれません) **lsof** ("list open files") を使用すれば、どのファイルが使用中でアンマウントをブロックしているか確認することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.8 NFS

ネットワーク共有 (NFS)

- 最も一般的なネットワーク共有は、**NFS** である
- それ以外には、**AFS**、**SMB** (**CIFS**) がある
- **mount** でネットワーク共有を指定できる

```
$ sudo mount -t nfs myserver.com:/shdir /mnt/shdir
```

- `/etc/fstab` に書くと起動時にマウントされる、それ以降は `mount -a` を使う:

```
myserver.com:/shdir /mnt/shdir nfs rsize=8192,wsize=8192,timeo=14,intr 0 0
```

- ネットワーク接続が有効になる前にシステムは **NFS** をマウントするかもしれない
  この調整に `netdev` と `noauto` オプションが利用可能である
- システムは、ネットワークが立ち上がる前に NFS ファイルシステムをマウントしようとする場合がある
  これを回避するために `netdev` と `noauto` オプションが使用可能 (詳細は `man nfs` を見て **mount** の
  オプションを調査する)
- **autofs** または **automount** を使って解決することも可能である
- **mount** には極めて多彩なオプションがあり、いくつかは **nfs** 専用である
  nfs と **mount** それぞれの詳細は **man** ページを参照

ネットワーク共有を介してリモート ファイルシステムをマウントするのは一般的で、(マウントされたファイルシステムは) ローカルマシン上にあるかのように見えます。

おそらく、今までに使用されている最も一般的な方法は **NFS** (**N**etwork **F**ile **S**ystem) です。

NFS は 1989 年に **Sun Microsystems** によって最初に開発され、継続的に更新されてきました。最新のシステムでは 2000 年以降継続的に更新されている **NFSv4** を使用しています。

他のネットワークファイルシステムには **AFS** (**A**ndrew **F**ile **S**ystem) と **SMB** (**S**erver **M**essage **B**lock) があります。
SMB は **CIFS** とも呼ばれます。

ネットワークファイルシステムは、ネットワーク共有上にそれが存在しない、もしくは、ネットワーク自体が利用できないなどの要因で、利用できない可能性があるためシステムはその可能性を想定しておく必要があります。

このような状況になっても、システムが指定された時間より長く待っている間にハングアップ、またはブロックしないようにする必要があります。これらの設定は、mount コマンドか `/etc/fstab` で指定できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.9 ブート時のマウントと/etc/fstab

ブート時のマウント

- システム初期化中に次のコマンドを実行すると、/etc/fstab 内にリストされている全てのファイルシステムがマウントされる

```
$ mount -a
```

- 起動時には、ローカルと、ネットワークマウントされるリモート ファイルシステムの、どちらもマウントするこが出来る

/etc/fstab には、ブート時にマウント可能なファイルシステムを定義します。どのファイルシステムがマウントできるか、ローカルのどの位置にマウントするか、誰がどのような権限でマウントするか、などが記載されています。/etc/fstab にファイルやディレクトリが書かれている場合、起動時にマウントされることもあります。

ここには、ローカルおよび NFS や samba のようなリモートにあるネットワーク共有ファイルの両方を含めることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

/etc/fstab

- ファイルシステム、標準的なマウントポイント、マウント時のオプションの情報が含まれる
- マウントするファイルシステムに関するファイルの情報が、スペースで区切られて記載される
  - デバイスファイル、ラベルまたは UUID
  - マウントポイント
  - ファイルシステムタイプ
  - カンマで区切られたオプションのリスト
  - ダンプ周期（または 0）
  - **fsck** パス番号（または 0）
- **mount** と **umount** コマンドは `/etc/fstab` 内の情報を参照する

このファイルの各レコードには、起動時にマウントされるファイルシステムに関する情報が含まれます。レコードの最初のアイテムは、(`/dev/sda1` などの) デバイスファイル名、ラベルまたは UUID のどれかになります。次にファイルシステムのマウントポイント（ツリー構造のどこに組み込まれるか）が続きます。次にカンマで区切られたオプションをもった、ファイルシステムタイプが記述されます。続いてダンプ周期（`dump -w` コマンドで利用）かゼロ（ **dump** の対象外とするというの意味)、その次に **fsck** パス回数かゼロ（このパーティションを **fsck** の対象外とするという意味）が記述されます。

```
x8:/tmp> cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Tue Aug 27 15:29:19 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=53ea9807-fd58-1234-9460-d03ec36f73a3 /     ext4    defaults 1 1
UUID=29C4-7777          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
LABEL=ALL               /ALL                    ext4    defaults 1 2
LABEL=UBUNTU            /UBUNTU                 ext4    defaults 1 2

/ALL/usr/local          /usr/local              none    bind 0 0
/ALL/usr/src            /usr/src                none    bind 0 0
/ALL/VMS                /VMS                    none    bind 0 0
/ALL/RESOURCES          /RESOURCES              none    bind 0 0

LABEL=Sam128            /SAM                    ext4    noauto 0 0
/usr/src/KERNELS.sqfs   /usr/src/KERNELS        squashfs loop  0 0

/dev/sda1               /Cbootefi       vfat    umask=0077,shortname=winnt 0 2
x8:/tmp>
```

図 21.7: **/etc/fstab の例**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.10 自動マウント

ファイルシステムの自動マウント

- ファイルシステムの使用時に利用可能 (**mount**) となり、アイドル状態になると切り離される (**umount**)
- **autofs**:
  - 何年も前から利用可能な技術である
  - **autofs** パッケージのインストレーションが必要となる
  - 柔軟性があり、設定ファイルは `/etc` に書く
- **automount**:
  - **systemd** の機能
  - `/etc/fstab` を修正し再起動するか、**systemd** を再スタートする:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart local-fs.target
```

必要なときにだけファイルシステムをマウントできる機能は、**Linux** システムには長い間存在しています。

このユーティリティを使うには、適切なパッケージマネージャーを使用してインストールされた **autofs** パッケージと `/etc` 内の構成ファイルが必要になります。

autofs は非常に柔軟でよく知られていますが、一方で **systemd** ベースのシステム（全てのエンタープライズ向け **Linux** ディストリビューションが導入しています）には、**systemd** のフレームワークに組み込まれた **自動マウント (automount)** 機能が備わっています。これを利用するための設定は、適切なデバイス、マウントポイント、マウントオプションを指定する行を **/etc/fstab** に追加するだけです。具体的には以下の指定をして

```
LABEL=Sam128 /SAM ext4 noauto,x-systemd.automount,x-systemd.device-timeout=10,x-systemd.idle-timeout=30 0 0
```

再起動するか、以下のコマンドを使います。

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart local-fs.target
```

次ページで例を示し、オプションについて説明します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

自動マウントの例

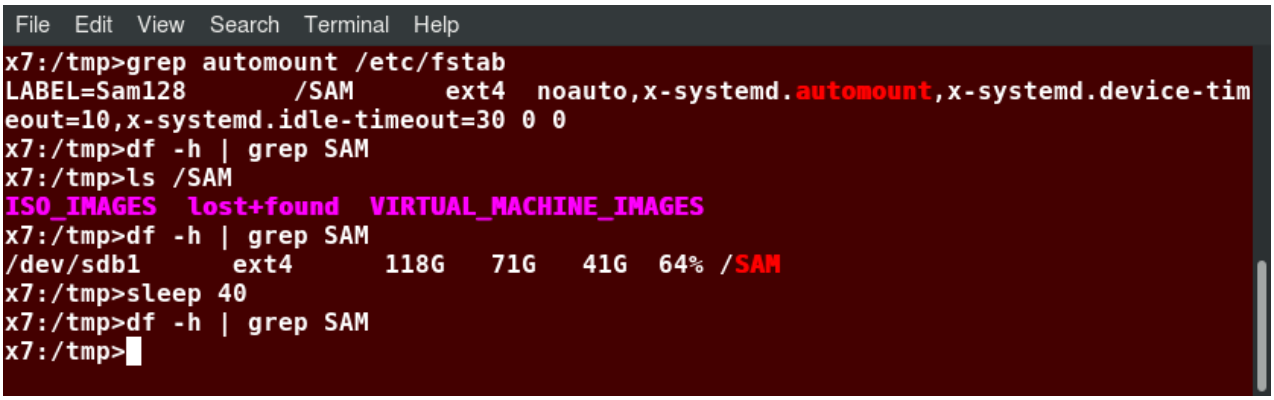


図 21.8: **automount の例**

上の例では、使用時にだけシステムに接続される **USB** ペンドライブをマウントします。
`/etc/fstab` のオプションは以下の通りです。

- `noauto` 起動時にマウントしません。`auto` は **automount** の意味ではありません。
- `x-systemd.automount` **systemd** の **自動マウント (automount)** 機能を使用します。
- `x-systemd.automount.device-timeout=10` このデバイスが使用できない場合、すなわち **NFS** を介してアクセス可能なネットワークデバイスの場合は、ハングアップする代わりに 10 秒後にタイムアウトします。
- `x-systemd.automount.idle-timeout=30` デバイスが 30 秒間使用されない場合は、アンマウントします。

デバイスはブート中にマウントされる **かも** しれませんが、指定タイムアウト時間にはアンマウントされることに注意してください。
スクリーンショットは、デバイスを使いたい時にだけ利用する（= マウントする）方法を示しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.11 ネットワーク ブロック デバイス

NBD の紹介

- ブロックデバイスをソースコンピュータ（サーバー）からターゲットコンピュータ（クライアント）にエクスポートするために設計された Linux プロトコル
- **NBD** は、通信に **UNIX ソケット**または **TCP/IP** を使用できる
- サーバーからエクスポートされる単位は、1 つ以上のファイル、イメージ、またはブロックデバイスである
- クライアント側では、サーバーから提示されたデータの塊は、**nbd カーネル モジュール**を介してマッピングされ、ブロックデバイスとしてアクセスされる
- クライアント側のブロックデバイスは `/dev/nbd0/`, `/dev/nbd1` などの名前で認識される
- 最も単純な構成ではデータストリームは暗号化されないが、暗号化も NBD の仕様に含まれている

以下の場所にいくつかの追加情報および参考資料があります。

NBD プロトコルの情報
https://github.com/NetworkBlockDevice/nbd/blob/master/doc/proto.md
NBD のユーザランドクライアント及びサーバーアプリケーション
https://github.com/NetworkBlockDevice/nbd

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ネットワーク ブロック デバイス

nbd のクライアントとサーバーのペアを構成するための一般的な手順

- サーバーからエクスポートするものを定義する
- サーバーに共有する項目を定義する
- クライアントを接続する
- **/dev/nbd0, /dev/nbd1** といったデバイスを使って、他のブロックデバイスと同様にパーティションの作成やフォーマットができる
- パーティションで区切られた nbd デバイスでは、ほぼすべてのタイプのファイルシステムを使用可能

今回使用するステップは

1. **dd** を使用して、空のファイルを作成します。
2. サーバーを使って共有する項目を設定ファイルに定義する。
3. **nbd** カーネルモジュールを有効化する。
4. エクスポート名、IP アドレス、ポートを使って、クライアントからサーバーに問い合わせる
5. **nbd-client** コマンドでローカルの `/dev/nbd` ブロックドライバとサーバを関連付ける。
6. `fdisk` を使って nbd にパーティションを作る。
7. nbd にファイルシステムを追加してマウントする。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

NBD ユーザー ユーティリティ

以下のような **nbd** のクライアント及びサーバー用のパッケージが利用可能

- **nbdkit**: **CentOS**, **Fedora**, **Debian**, **Ubuntu**
- **nbd-client** 及び **nbd-server**: **Ubuntu**, **Debian**
- **nbd** (**GitHub** から): **CentOS**, **Fedora**, **Debian**, **Ubuntu**
- **xNBD-client** 及び **xNBD-server**: **Debian**
- **qemu-img**: **CentOS**

一般的にクライアントとサーバーは混在可能なので、ユースケースで慎重にテストすることを推奨する

私たちが利用するパッケージ

演習では **GitHub** にある **nbd** パッケージを利用します。これはパッケージの中で最も複雑なものではありませんが、ほとんどの **Linux** バージョンで利用できます。

ubuntu22.04 のユーザーユーティリティと管理者ユーティリティの例です:

- **nbd-server.conf** は、含むサーバーの設定ファイルの一例です。
  - 接続を待機する IP アドレスとポート
  - ディスクとして書き出す記憶装置
  - いくつかのオプションの制御要素
- **nbd-server** は、接続要求や通信に応答するサーバー側のコンポーネントです。
- **nbd-server man ページ**には、サーバーの設定に関する詳細が記載されています。man ページの情報は、ディストリビューションによって異なる場合があります。
- **nbd-client** は、サーバーへの問い合わせと接続を行うために使用されます。
- **nbd-client man ページ**には、サーバーへの接続を行うためのクライアント関連情報が記載されています。

これらのユーティリティは、ディストリビューション毎にどのようにパッケージ化されているかによって、異なる名前や異なる機能を含む場合があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ネットワーク ブロック デバイスの例

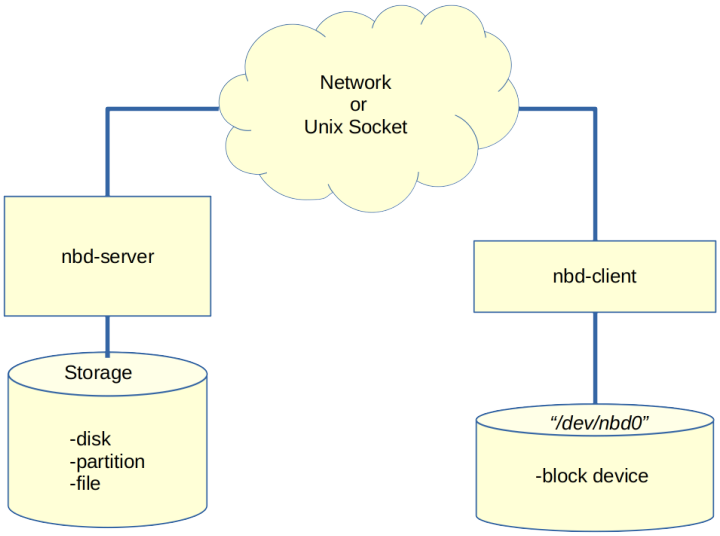


図 21.9: **ネットワーク ブロック デバイス**

クライアントとサーバーのコマンドの例をいくつか紹介します。
サーバーは **CentOS-8-Stream** で、**GitHub** の **nbd** パッケージを使用しました。
クライアントは **CentOS-8-Stream** で、**GitHub** の **nbd** パッケージを使用しました。

- nbd カーネルモジュールがロードされていることを確認する。

```
$ sudo  modprobe -i nbd
```

- 192.168.242.160 の export foo とローカルデバイス /dev/nbd10 を接続します。

```
$ sudo  nbd-client -N foo 192.168.242.160 /dev/nbd10
```

Ubuntu インストール時のコマンドの例です。

- nbd サーバーのプロセスを開始する。

```
$ sudo nbd-server -C nbd-server.conf
```

- クライアントからサーバー上のエクスポートをリストアップする

```
$ sudo nbd-client -l 127.0.0.1 10042
```

- エクスポート「foo」をローカルデバイス /dev/nbd0 に接続する。

```
$ sudo nbd-client -N foo 127.0.0.1 10042 /dev/nbd0
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

21.12 演習

デモ教材ビデオ

using_filesystems_demo.mp4

課題 21.1: ファイル属性について

1. 自分のユーザーアカウントで、空の /tmp/appendit ファイルを **touch** で作成してください。
2. **cat** を利用して、/etc/hosts の内容を /tmp/appendit に追加してください。
3. /tmp/appendit を /etc/hosts と比較してください。違いは無いはずです。
4. **chattr** コマンドで /tmp/appendit に追加だけ（append-only）属性を追加してください。エラーが出るはずです。なぜでしょうか。
5. root ユーザーで追加だけ（append-only）属性を追加してください。今度は成功するはずです。**lsattr** コマンドで、ファイルの拡張属性を見てください。
6. 一般のユーザーで **cat** コマンドを使って /etc/passwd の内容を /tmp/appendit にコピーしてください。エラーがでるはずです。なぜでしょうか。
7. 同じことを root ユーザーで実行してください。これもエラーになります。なぜでしょうか。
8. 一般のユーザーで追加のリダイレクション（>>）を使って、/etc/passwd ファイルを /tmp/appendit に追加してください。成功します。結果を調べてください。
9. root ユーザーで /tmp/appendit の変更不可（immutable）属性をセットし、拡張属性を確認してください。
10. 一般のユーザーと root ユーザーの両方で /tmp/appendit への追加、ファイル名称の変更、ファイルへのハードリンク、ファイルの削除を実行してください。
11. 拡張属性を削除することでファイルを削除できるようになります。実行してください。

解 21.1

1. ```
   $ cd /tmp
   $ touch appendit
   $ ls -l appendit
   ```

   ```
   -rw-rw-r-- 1 coop coop 0 Oct 23 19:04 appendit
   ```

2. ```
   $ cat /etc/hosts > appendit
   ```

3. ```
   $ diff /etc/hosts appendit
   ```

4. ```
   $ chattr +a appendit
   ```

   ```
   chattr: Operation not permitted while setting flags on appendit
   ```

5. ```
   $ sudo chattr +a appendit
   $ lsattr appendit
   ```

   ```
   -----a-------e-- appendit
   ```

6. ```
   $ cat /etc/passwd > appendit
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
bash: appendit: Operation not permitted
```

7.
```
$ sudo su
$ cat /etc/passwd > appendit
```

```
bash: appendit: Operation not permitted
```

```
$ exit
```

8.
```
$ cat /etc/passwd >> /tmp/appendit
$ cat appendit
```

9.
```
$ sudo chattr +i appendit
$ lsattr appendit
```

```
----ia-------e- appendit
```

10.
```
$ echo hello >> appendit
```

```
-bash: appendit: Permission denied
```

```
$ mv appendit appendit.rename
```

```
mv: cannot move `appendit' to `appendit.rename': Operation not permitted
```

```
$ ln appendit appendit.hardlink
```

```
ln: creating hard link `appendit.hardlink' => `appendit': Operation not permitted
```

```
$ rm -f appendit
```

```
rm: cannot remove `appendit': Operation not permitted
```

```
$ sudo su
$ echo hello >> appendit
```

```
-bash: appendit: Permission denied
```

```
$ mv appendit appendit.rename
```

```
mv: cannot move `appendit' to `appendit.rename': Operation not permitted
```

```
$ ln appendit appendit.hardlink
```

```
ln: creating hard link `appendit.hardlink' => `appendit': Operation not permitted
```

```
$ rm -f appendit
```

```
rm: cannot remove `appendit': Operation not permitted
```

```
$ exit
```

11.
```
$ sudo su
$ lsattr appendit
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
----ia-------e- appendit
```

```
$ chattr -ia appendit
$ rm appendit
```

```
rm: remove regular file `appendit'? y
```

```
$ ls appendit
```

```
ls: cannot access appendit: No such file or directory
```

課題 21.2: マウントのオプション

新しいパーティション、またはループバックファイル

本演習では、新しいパーティションを作成するか、ループバックファイルを利用するか、のどちらかをします。大きな違いはありませんが、両方のやり方を説明します。

1. **fdisk** を使って 250MB の新パーティションを作成します。通常 /dev/sda に作成します。または、ループバックファイルを利用して、新しいパーティションをシミュレートするために、0（zero）で満たされたファイルを作成します。
2. **mkfs** を使って、新しいパーティションまたはループバックファイル上のファイルシステムをフォーマットします。これを、ブロックサイズを変えながら 3 回行います。スーパーブロックの位置、ブロックグループの数、その他の必要な情報をメモしておきます。
3. 新たにディレクトリ (/mnt/tempdir) を作成し、新しいファイルシステムをマウントします。結果を確認します。
4. マウントしたファイルシステムをアンマウントし、読み取り専用でマウントします。
5. マウントしたディレクトリにファイルを作成します。エラーになります。なぜでしょうか。
6. ファイルシステムをアンマウントします。
7. ブート時にマウントされるように、/etc/fstab ファイルに行を追加します。
8. ファイルシステムをマウントします。
9. ファイルシステム上のバイナリ ファイルを実行不可にするようにファイルシステムを属性を変更します。(/mnt/tempdir エントリの noexec をデフォルトに変更します）ファイルシステムを再マウントし、実行ファイル（例えば /bin/ls ）を /mnt/tempdir にコピーします。実行してみます。エラーになります。なぜでしょうか。

演習修了後クリーンアップするためには、/etc/fstab から当該エントリを削除してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

解 21.2

物理的なパーティションによる方法

1. 以前に説明したので **fdisk** の詳細は省略します。パーティションを /dev/sda11 で作成したものとします。

```
$ sudo fdisk /dev/sda
```

```
.....
w
$ partprobe -s
```

partprobe がうまく実行できない場合があります。システムに間違いなく新パーティションを認識させるため、リブートしたほうが良いでしょう。

2.
```
$ sudo mkfs -t ext4         -v /dev/sda11
$ sudo mkfs -t ext4 -b 2048 -v /dev/sda11
$ sudo mkfs -t ext4 -b 4096 -v /dev/sda11
```

-v フラグ（verbose）で情報を表示します。今回のような小さいパーティションでは、ブロックサイズの標準は 1024 バイトです。

3.
```
$ sudo mkdir /mnt/tempdir
$ sudo mount /dev/sda11 /mnt/tempdir
$ mount | grep tempdir
```

4.
```
$ sudo umount /mnt/tempdir
$ sudo mount -o ro /dev/sda11 /mnt/tempdir
```

アンマウント時にエラーが発生したときは、当該ディレクトリにいないことを確認してください。

5.
```
$ sudo touch /mnt/tempdir/afile
```

6.
```
$ sudo umount /mnt/tempdir
```

7. /etc/fstab にこの行を追記してください。

```
/dev/sda11 /mnt/tempdir ext4 defaults 1 2
```

8.
```
$ sudo mount /mnt/tempdir
$ sudo mount | grep tempdir
```

9. /etc/fstab の行を次のように変更してください。

```
/dev/sda11 /mnt/tempdir ext4 noexec 1 2
```

そして次を実行します。

```
$ sudo mount -o remount /mnt/tempdir
$ sudo cp /bin/ls /mnt/tempdir
$ /mnt/tempdir/ls
```

エラーになります。なぜでしょうか。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ループバックファイルによる方法

1. `$ sudo dd if=/dev/zero of=/imagefile bs=1M count=250`

2. ```
   $ sudo mkfs -t ext4         -v
   $ sudo mkfs -t ext4 -b 2048 -v /imagefile
   $ sudo mkfs -t ext4 -b 4096 -v /imagefile
   ```

   パーティションではなくファイルであるという警告がでますが、そのまま進んでください。

   -v フラグ（verbose）で情報を表示します。今回のような小さいパーティションでは、ブロックサイズの標準は 1024 バイトです。

3. ```
   $ sudo mkdir /mnt/tempdir
   $ sudo mount -o loop /imagefile /mnt/tempdir
   $ mount | grep tempdir
   ```

4. ```
   $ sudo umount /mnt/tempdir
   $ sudo mount -o ro,loop /imagefile /mnt/tempdir
   ```

   アンマウント時にエラーが発生したときは、当該ディレクトリにいないことを確認してください。

5. `$ sudo touch /mnt/tempdir/afile`

6. `$ sudo umount /mnt/tempdir`

7. `/etc/fstab` にこの行を追記してください。

/etc/fstab の追加内容

```
/imagefile /mnt/tempdir ext4 loop 1 2
```

8. ```
   $ sudo mount /mnt/tempdir
   $ sudo mount | grep tempdir
   ```

9. `/etc/fstab` の行を次のように変更してください。

/etc/fstab の変更内容

```
/imagefile /mnt/tempdir ext4 loop,noexec 1 2
```

次を実行してください:

```
$ sudo mount -o remount /mnt/tempdir
$ sudo cp /bin/ls /mnt/tempdir
$ /mnt/tempdir/ls
```

エラーになります。なぜでしょうか。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 21.3: NBD の設定とテスト

概要

このラボでは、**ループバックデバイス**を介して通信する **NBD** を設定し、テストする予定です。クライアントとサーバーの両方が同じシステム上で実行されます。

NBD エクスポートの場合、空のファイルが使用されます。

`opt` ディレクトリは、エクスポートされたデバイスとコンフィギュレーションファイルのために使用されます。本番環境では、異なる場所が指定される場合があります。

CentOS-stream-8 の問題

講座作成時には **CentOS-8-Stream** は正常に動作していません。**CentOSStream-9** または **Fedora** をご利用ください。解決待ちです。

1. 最初のステップ

Red Hat, Centos, Fedora では

git でダウンロードしたソースコードを格納するディレクトリを用意する。**NBD** プログラムをクローンし、ビルドし、インストールします。

Debian, Ubuntu, Linux Mint では

NBD のクライアントとサーバーパッケージをインストールします。

2. `nbd-server.conf` ファイルを `/opt` ディレクトリに作成します。最小限の、コメントなしのファイルで十分です。（もっと面白い設定ファイルは、SOLUTIONS ディレクトリにあります。）
3. エクスポート用のファイルを 2 つ作成し、所有者を `student` に設定します。
4. **nbd-server** を起動します。
5. nbd サーバーがクライアントからのクエリに応答することを確認する。
6. **nbd** カーネルモジュールをインストールし、ブロックデバイスが存在することを確認する。
7. サーバーから提供されたイメージを、ip アドレス、ポートを使って、ローカルブロックデバイスに接続します。
8. **NBD** の既存情報を消去する。
9. **NBD** を使用して **GPT** を作成します。
10. **NBD** にパーティションを作成します。
11. **NBD** に **ext3** ファイルシステムを追加してテストします。

解 21.3

1. 最初のステップ

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Red Hat, Centos, Fedora では

```
$ mkdir ~/src
$ cd ~/src
$ git clone https://github.com/NetworkBlockDevice/nbd.git
$ cd nbd/
```

docbook-utils と autoconf-archive パッケージは、あなたのシステムにはインストールされていないかもしれません、インストールしてください。

```
$ sudo dnf install docbook-utils
$ sudo dnf install autoconf-archive
```

プログラムをビルドします。

```
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

Debian, Ubuntu, Linux Mint では

Debian, **Ubuntu**, or **Linux Mint** では deb パッケージが利用可能なので **NBD** ユーティリティをインストールします。

```
$ sudo apt install nbd-client nbd-server
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nbd-client nbd-server
0 upgraded, 2 newly installed, 0 to remove and 43 not upgraded.
\ldots output truncated
^^I
```

2. **nbd-server** の設定ファイルを /opt ディレクトリに作成します。/opt ディレクトリのファイルの編集には **sudo** が必要です。

```
$ sudo /opt/nbd-server.conf
```

/opt/nbd-server.conf

```
[generic]
    allowlist = 1
    listenaddr = 0.0.0.0
    port = 10042
[foo]
    exportname = /opt/dsk1
    readonly = false
[bar]
    exportname = /opt/dsk2
^^I
```

3.
```
sudo dd if=/dev/zero of=/opt/dsk1 status=progress bs=100M count=5
sudo dd if=/dev/zero of=/opt/dsk2 status=progress bs=100M count=5
sudo chmod 777 /opt
sudo chown student.student /opt/*
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

4. `$ sudo nbd-server -C /opt/nbd-server.conf`

5. `$ sudo nbd-client -l 127.0.0.1 -p 10042`

```
Negotiation: ..
foo
bar
^^I
```

6.
```
$ sudo modprobe -i nbd
$ ls /dev/nbd*
```

```
/dev/nbd0   /dev/nbd11  /dev/nbd14  /dev/nbd3  /dev/nbd6  /dev/nbd9
/dev/nbd1   /dev/nbd12  /dev/nbd15  /dev/nbd4  /dev/nbd7
/dev/nbd10  /dev/nbd13  /dev/nbd2   /dev/nbd5  /dev/nbd8
^^I
```

7. `$ sudo nbd-client -N foo  127.0.0.1 10042 /dev/nbd0`

8. NBD の既存情報を消去します。

```
$ sudo dd if=/dev/zero of=/dev/nbd0 bs=1M count=5
```

```
5+0 records in
5+0 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.00161564 s, 3.2 GB/s
^^I
```

9. NBD に GPT ラベルを作成します。

```
$ sudo su -c "echo 'label: gpt' | sfdisk /dev/nbd0"
```

```
Checking that no-one is using this disk right now ... OK

Disk /dev/nbd0: 4 GiB, 4294967296 bytes, 8388608 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

>>> Script header accepted.
>>> Done.
Created a new GPT disklabel (GUID: 000C20C1-6929-424F-93E1-28319DE1FF1F).

New situation:
Disklabel type: gpt
Disk identifier: 000C20C1-6929-424F-93E1-28319DE1FF1F

The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
^^I
```

10. NBD にパーティションを追加します。

```
$ sudo su -c "echo ';' | sfdisk /dev/nbd0"
```

```
Checking that no-one is using this disk right now ... OK

Disk /dev/nbd0: 4 GiB, 4294967296 bytes, 8388608 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 000C20C1-6929-424F-93E1-28319DE1FF1F

Old situation:

>>> Created a new GPT disklabel (GUID: EFFC087B-A15C-7D42-809C-4BDED58EE238).
/dev/nbd0p1: Created a new partition 1 of type 'Linux filesystem' and of size 4 GiB.
/dev/nbd0p2: Done.

New situation:
Disklabel type: gpt
Disk identifier: EFFC087B-A15C-7D42-809C-4BDED58EE238

Device      Start     End Sectors Size Type
/dev/nbd0p1  2048 8388574 8386527   4G Linux filesystem

The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
^^I
```

11. **NBD** に **ext3** ファイルシステムを追加してテストします。

```
$ sudo mkfs.ext3 -L nbd-foo /dev/nbd0
```

```
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 128000 4k blocks and 128000 inodes
Filesystem UUID: 6a01ff5a-9ad5-4f43-8050-9f2b8df26c14
Superblock backups stored on blocks:
   32768, 98304

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
^^I
```

```
$ sudo mount LABEL=nbd-foo /mnt
$ sudo touch /mnt/file1
$ ls -l /mnt
```

```
total 16
-rw-r--r-- 1 root root     0 Jun  9 10:58 file1
drwx------ 2 root root 16384 Jun  9 10:56 lost+found
^^I
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 22 章

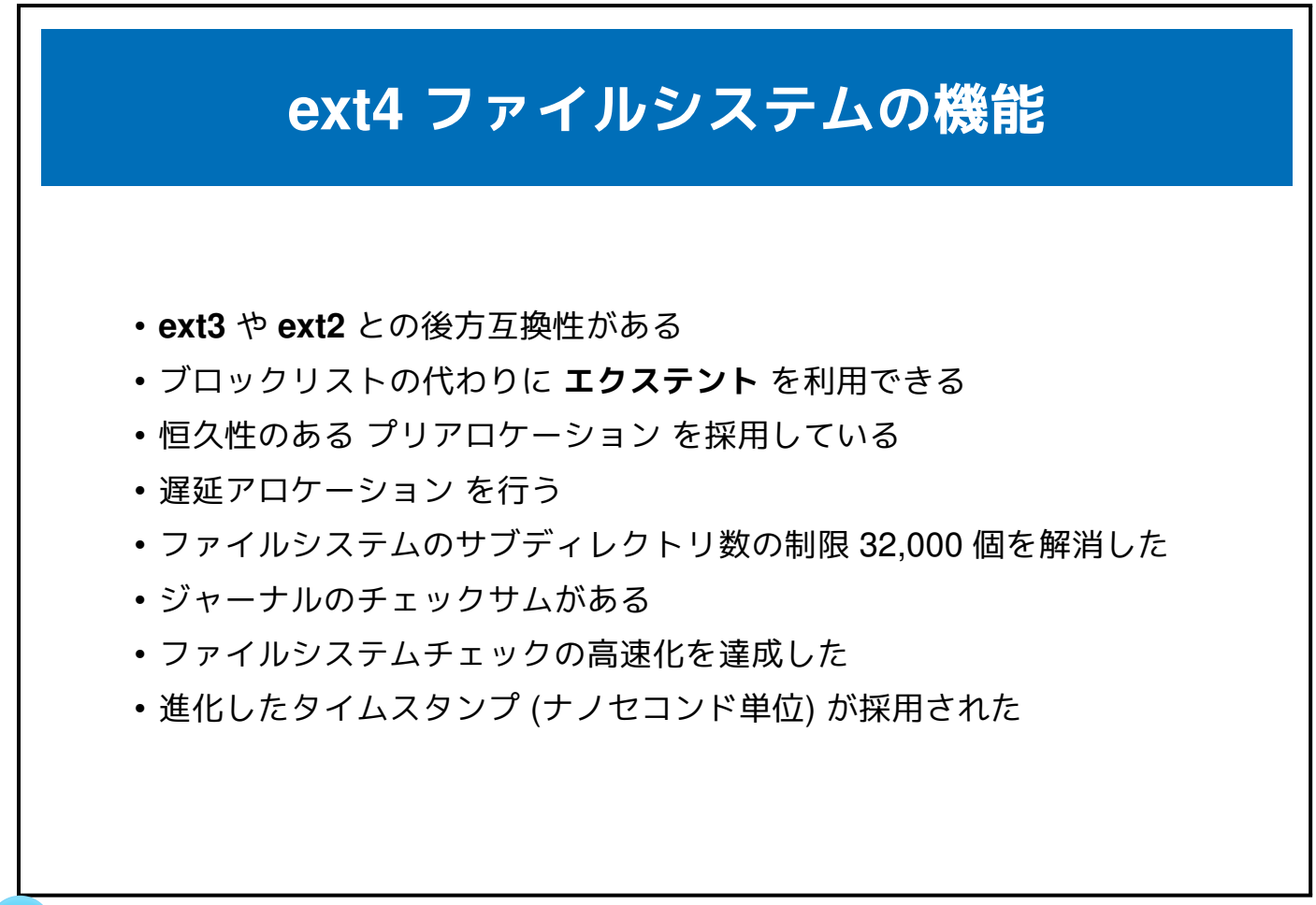
ext4 ファイルシステム

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- **ext4** ファイルシステムの主な機能とディスク上のレイアウトを説明できます。
- ブロックグループ、スーパーブロック、データブロック、inode の概念について説明できます。
- **dumpe2fs** および **tune2fs** ユーティリティを使用できます。

22.1 ext4 の機能



エクステント

エクステント とは連続したブロックのグループです。エクステントを利用すると大きなファイルのパフォーマンスを向上させることが可能でフラグメンテーションも削減されます。

ext4 は 1EB までのボリュームと 16TB までのファイルをサポートします。**エクステント** は標準的なブロックマップのメカニズムを置き換えるものです。

ext4 は **ext3** と **ext2** との後方互換性を持っています。ファイルに対して領域を事前に確保することができます。確保された領域は、通常は連続領域となることが保証されます。また、**allocate-on-flush** と呼ばれる性能向上のテクニック (実際のディスクに書くタイミングまで、ブロックアロケーションを保留する）も採用されています。**ext4** では **ext3** にあったサブディレクトリ数の上限 32,000 個の制約も解消されました。

ext4 はジャーナルデータにチェックサムを使っているので、信頼性も向上しています。これにより、ジャーナル期間中のディスク I/O 待ちが安全に回避できるので性能も若干改善しています。

この他、タイムスタンプも強化されています。**ext4** はナノセコンド単位のタイムスタンプを記録します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

22.2 ext4 のレイアウト、スーパー ブロック、ブロック グループ

ext4 の スーパー ブロック と ブロック グループ

- 先頭の **スーパー ブロック** にファイルシステムの全ての情報が含まれる
- 以降に **ブロック グループ** とよばれる連続ブロックのセットがある
  - ここには管理情報が含まれている
  - 複数のブロックグループには、冗長性のためのスーパーブロックのコピーが記録される
  - その他のブロックには、ファイルのデータが書かれる
- ブロックサイズは、ファイルシステム作成時に決まる
  - 512, 1K, 2K, 4K, 8K バイトなど、但しメモリーの**ページ サイズ** (**x86** では 4k バイト) よりは大きくは出来ない

ext4 ファイルシステムは、ブロックグループのセットに分割されます。ブロックアロケータは、シーク時間を減らすためにひとつのファイルのブロックを同じブロックグループ内に配置しようとします。デフォルトのブロックサイズ 4 KB の場合、ブロックグループのサイズは 128MB になります。

ext4 ではジャーナルを除く全てのフィールドは **リトル エンディアン** でディスクに書き込まれます。

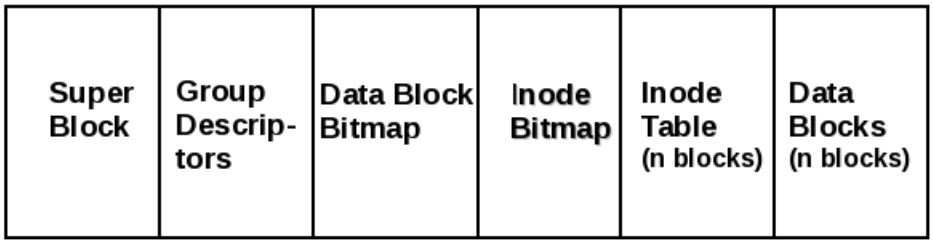


図 22.1: **ext[3,4] ファイルシステムレイアウト**

標準的なブロックグループのレイアウトは単純です。先頭の 1024 バイト（ブロック 0）は使いません。(ここはブートセクターなどの用途のために空けておく）スーパーブロックは、ブロックグループ 0 を除く先頭（つまりブロック 1）から始まります。その後ろには、グループディスクリプタといくつかの **GDT** ブロックが続きます。その後ろにデータブロックビットマップ、inode ビットマップ、inode テーブルそしてデータブロックが格納されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ブロック グループ

- スーパー ブロック と グループ ディスクリプタコピー は、多重化されていてファイルシステムのリカバリー時にはコピーが利用される

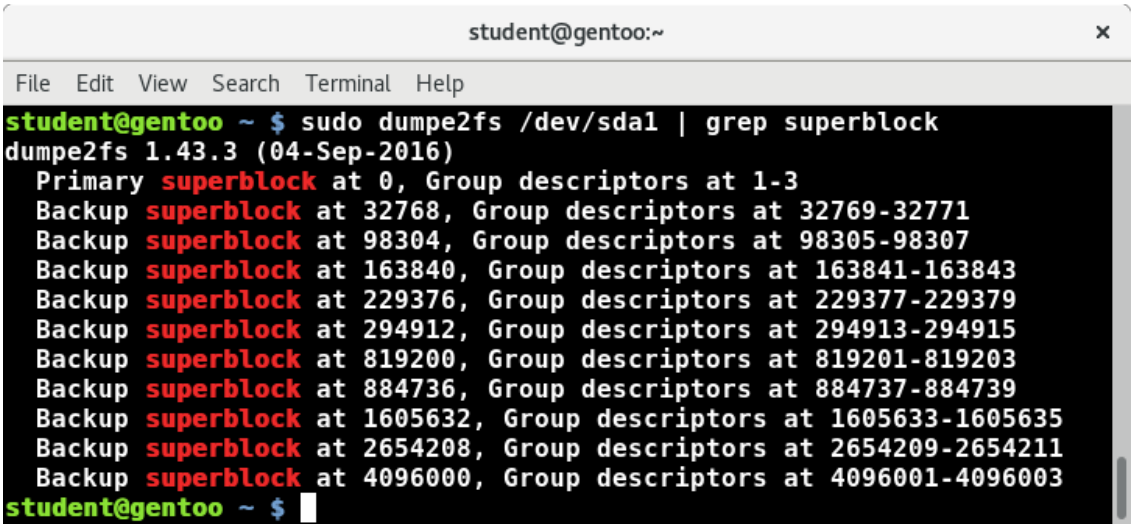


図 22.2: **ブロック グループ**

全ブロック共通で、一番目と二番目のブロックには **スーパー ブロック** と **グループ ディスクリプタ** が書かれています。

カーネルは、通常は最初のブロック グループを参照します。多重化されたコピーは、ファイルシステムチェック時だけ参照されます。何も問題が見つからなければ、カーネルは単純に最初のブロックグループの内容を多重化コピーに上書きします。

マスターコピーに問題が見つかった場合には、健全なデータをつかってファイルシステム構造がリビルドできるまで、順に次のデータを参照していきます。この冗長性により、ファイルシステムチェックが定期的に実行されている限り、ext2/ext3/ext4 ファイルシステムが完全に壊されることはありません。

ext ファイルシステム ファミリーの初期には、全てのブロック グループに全てのブロック グループのグループ ディスクリプタとスーパーブロックのコピーが含まれていました。しかし現在では、最適化のため全てのブロック グループにスーパー ブロックとグループ ディスクリプタのコピーがあるわけではありません。自分の環境がどうなっているかを確認するには、添付のスクリーンショットのコマンドを（適切なデバイス ノードを指定して）利用して、正確な配置を確認することができます。スーパー ブロックの複製の配置は、ファイルシステム作成時に `sparse_super` オプションで設定されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

スーパー ブロックの詳細

- スーパー ブロックには、ファイルシステムのグローバル情報が書かれる
  - マウント回数と最大マウント回数
  - このファイルシステムのブロックサイズ
  - グループ毎のブロック数
  - 未使用のブロック数
  - 未使用の inode 数
  - オペレーティング システム ID
- スーパーブロックは、冗長性のため複数のブロックグループに保存されている

マウント回数は、ディスクが正常にマウントされるたびに増えます。最大マウント回数、もしくは 180 日のどちらか先に発生したタイミングで、ファイルシステムがチェックされます。

ブロックサイズは **mkfs** コマンドによって設定されます。

ファイルシステムのスーパー ブロックは、ディスクのブロック 0 に保存されています。スーパー ブロックには、ファイルシステム自体に関する情報が格納されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

22.3 dumpe2fs

ext4 のブロックと inode 情報の確認 dumpe2fs

- ブロックサイズは以下のデータの最大値を決めるのに使われる
  - ブロック
  - inode
  - スーパーブロック
- **dumpe2fs** でファイルシステムの次の情報が参照可能
  - 制限
  - 機能とフラグ
  - その他の属性

dumpe2fs を使って特定のパーティションの情報を取得することができます。

```
$ sudo dumpe2fs /dev/sdb1

dumpe2fs 1.45.6 (20-Mar-2020)
Filesystem volume name:   VMS
Last mounted on:          /VMS
Filesystem UUID:          fce521c7-e2ce-414a-8a7e-e2311640802f
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent 64bit flex_bg \
sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linuxxf .
Inode count:              14352384
Block count:              57388288
Reserved block count:     2869413
Free blocks:              22270800
Free inodes:              14352217
First block:              0
Block size:               4096

  Block bitmap at 1056 (bg #0 + 1056)
  Inode bitmap at 1072 (bg #0 + 1072)
  Inode table at 1599-2110 (bg #0 + 1599)
  415 free blocks, 8192 free inodes, 0 directories, 8192 unused inodes
  Free blocks: 33822-33985, 34550-34691, 38803-38911
  Free inodes: 8193-16384
Group 2: (Blocks 65536-98303) csum 0xdde9 [INODE_UNINIT, ITABLE_ZEROED]
  Block bitmap at 1057 (bg #0 + 1057)
  Inode bitmap at 1073 (bg #0 + 1073)
  Inode table at 2111-2622 (bg #0 + 2111)
  0 free blocks, 8192 free inodes, 0 directories, 8192 unused inodes
  Free blocks:
  Free inodes: 16385-24576
....
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

22.4 tune2fs

tune2fs: ファイルシステムのパラメータ変更

tune2fs を使うと、ファイルシステムのパラメータの変更が可能となる

- ファイルシステムのチェック間隔を決める最大マウント回数を変更する (max-mount-count):

  ```
  $ sudo tune2fs -c 25 /dev/sda1
  ```

- チェックインターバルを変更する (interval-between-checks):

  ```
  $ sudo tune2fs -i 10 /dev/sda1
  ```

- 変更可能な値を含む、現在のスーパーブロックのコンテンツのリスト

  ```
  $ sudo tune2fs -l /dev/sda1
  ```

  基本的には **dumpe2fs** からのグローバル情報を表示する

```
$ sudo tune2fs -l /dev/sdb1

tune2fs 1.45.6 (20-Mar-2020)
Filesystem volume name:   VMS
Last mounted on:          /VMS
Filesystem UUID:          fce521c7-e2ce-414a-8a7e-e2311640802f
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent 64bit flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              14352384
Block count:              57388288
Reserved block count:     2869413
Free blocks:              22270800
Free inodes:              14352217
First block:              0
Block size:               4096
Fragment size:            4096
.....
Filesystem created:       Mon Sep 25 14:14:57 2017
Last mount time:          Mon Mar  8 06:05:03 2021
Last write time:          Mon Mar  8 06:05:03 2021
Mount count:              2003
Maximum mount count:      -1
Last checked:             Wed Feb 28 14:24:15 2018
Check interval:           0 (<none>)
Lifetime writes:          14 TB
....
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

22.5 演習

課題 22.1: デフラグ（Defragmentation）

Linux の初心者の中には **Windows** で良く使われているファイルシステムの **デフラグツール** についての説明が欠けていることに驚く人がいます。

しかし **Linux** を含む **UNIX** タイプのオペレーティング システムでは、ファイルシステムのフラグメンテーションが原因で重大な問題が発生することはありません。

この主な理由はディスクの内周に多くのファイルを詰め込むようなことはしていないからです。その代わりディスク全体にフリースペースが多くばらまかれることになります。そのため、１つのファイルを作るときに十分なサイズのフリーブロックを１個かまたは少ない数のブロックで対応できるというより良いやりかたで作れるのです

最近のハードウェアでは、ディスクの内周かどうかはハードウェアが隠蔽してしまいます。

これは、止めてください

デフラグは 読み/消去/書き込み を繰り返し実行するので、**SSD** の寿命を短くします。最新のオペレーティング システムでは、**SSD** については標準でデフラグが使われないようになっています。

さらに、新しい **journalling** ファイルシステム（**ext4** を含む）は、設計上 **extents** （連続した大きな領域）と呼ばれる領域を持っています。

とはいえ **ext4** ファイルシステムをデフラグするツールも存在しています。

```
$ sudo e4defrag
```

```
  Usage   : e4defrag [-v] file...| directory...| device...
          : e4defrag  -c  file...| directory...| device...
```

e2fsprogs パッケージに含まれる **e4defrag** は、最新の **Linux** ディストリビューションには必ず含まれています。

２つのオプションがあります。

- -v: 詳細な出力を出力します。
- -c: 解析してレポートを出力しますが、他は何もしません。

引数には以下が指定できます。

- ファイル
- ディレクトリ
- デバイス

例:

```
$ sudo e4defrag -c /var/log
```

```
  <Fragmented files>                            now/best       size/ext
  1. /var/log/lastlog                             5/1             9 KB
  2. /var/log/sa/sa24                             3/1            80 KB
  3. /var/log/rhsm/rhsm.log                       2/1           142 KB
  4. /var/log/messages                            2/1          4590 KB
  5. /var/log/Xorg.1.log.old                      1/1            36 KB

   Total/best extents                           120/112
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
Average size per extent                         220 KB
Fragmentation score                             1
[0-30 no problem: 31-55 a little bit fragmented: 56- needs defrag]
This directory (/var/log) does not need defragmentation.
Done.
```

```
$ sudo e4defrag  /var/log
```

```
ext4 defragmentation for directory(/var/log)
[2/152]/var/log/Xorg.2.log:     100%    [ OK ]
[3/152]/var/log/Xorg.0.log.old: 100%    [ OK ]
[4/152]/var/log/messages-20141019.gz:   100%    [ OK ]
[5/152]/var/log/boot.log:       100%    [ OK ]
[7/152]/var/log/cups/page_log-20140924.gz:      100%    [ OK ]
[8/152]/var/log/cups/access_log-20141019.gz:    100%    [ OK ]
[9/152]/var/log/cups/access_log:        100%    [ OK ]
[10/152]/var/log/cups/error_log-20141018.gz:    100%    [ OK ]
[11/152]/var/log/cups/error_log-20141019.gz:    100%    [ OK ]
[12/152]/var/log/cups/access_log-20141018.gz:   100%    [ OK ]
[14/152]/var/log/cups/page_log-20141018.gz:     100%    [ OK ]
...
[152/152]/var/log/Xorg.1.log.old:       100%    [ OK ]

        Success:                        [ 112/152 ]
        Failure:                        [ 40/152 ]
```

いろいろなファイル、ディレクトリ、デバイス全体に対して **e4defrag** を試してください、その際常に最初に -c を試しましょう

Linux ファイルシステムで、デフラグが必要になるのは、使用率が 90%以上になったときや、**boot** パーティションのように小さいファイルシステムに比較的大きなファイルを格納しようとした時だけだと気付くでしょう。

課題 22.2: tune2fs コマンドで、ファイルシステムのパラメータを変更する

フォーマット済の **ext4** ファイルシステムのプロパティを扱います。これは、ファイルシステムのアンマウントは不要です。

以下のコマンドで作成したイメージファイルを利用します。または、`imagefile` の代わりに `/dev/sdaX`（変更したいファイルシステムがマウントされている）を利用します。

```
$ dd if=/dev/zero of=imagefile bs=1M count=1024
$ mkfs.ext4 imagefile
```

1. **dumpe2fs** を利用して、変更したいファイルシステムのプロパティを得ます。
   表示:
   - 最大マウント回数（回数を超えたら、ファイルシステムチェックを強制実行する）
   - チェック間隔 （指定間隔を超えたら、ファイルシステムチェック実行）
   - 予約ブロック数とブロック数の合計
2. 変更:
   - 最大マウント回数を 30 に
   - チェック間隔を 3 週間に
   - 予約ブロックの割合を 10%
3. **dumpe2fs** を使ってファイルシステムの情報を得たら、オリジナルの出力と比較します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

解 22.2

1. `$ dumpe2fs imagefile > dumpe2fs-output-initial`

```
dumpe2fs 1.42.9 (28-Dec-2013)
         \end{response}
         \begin{out}
$ grep -i  -e "Mount count" -e "Check interval" -e "Block Count" dumpe2fs-output-initial
         \end{cmd}
         \begin{out}[]
Block count:              262144
Reserved block count:     13107
Mount count:              0
Maximum mount count:      -1
Check interval:           0 (<none>)
```

2. `$ tune2fs -c 30 imagefile`

```
tune2fs 1.42.9 (28-Dec-2013)
Setting maximal mount count to 30
```

`$ tune2fs -i 3w imagefile`

```
tune2fs 1.42.9 (28-Dec-2013)
Setting interval between checks to 1814400 seconds
```

`$ tune2fs -m 10 imagefile`

```
tune2fs 1.42.9 (28-Dec-2013)
Setting reserved blocks percentage to 10% (26214 blocks)
```

3. `$ dumpe2fs imagefile > dumpe2fs-output-final`

```
dumpe2fs 1.42.9 (28-Dec-2013)
```

`$ grep -i  -e "Mount count" -e "Check interval" -e "Block Count" dumpe2fs-output-final`

```
Block count:              262144
Reserved block count:     26214
Mount count:              0
Maximum mount count:      30
Check interval:           1814400 (3 weeks)
```

`$ diff dumpe2fs-output-initial dumpe2fs-output-final`

```
14c14
< Reserved block count:     13107
---
> Reserved block count:     26214
29c29
< Last write time:          Wed Oct 26 14:26:19 2016
---
> Last write time:          Wed Oct 26 14:26:20 2016
31c31
< Maximum mount count:      -1
---
> Maximum mount count:      30
33c33,34
< Check interval:           0 (<none>)
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
---
> Check interval:           1814400 (3 weeks)
> Next check after:         Wed Nov 16 13:26:16 2016
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 23 章

論理ボリューム管理 (LVM)

学習目標

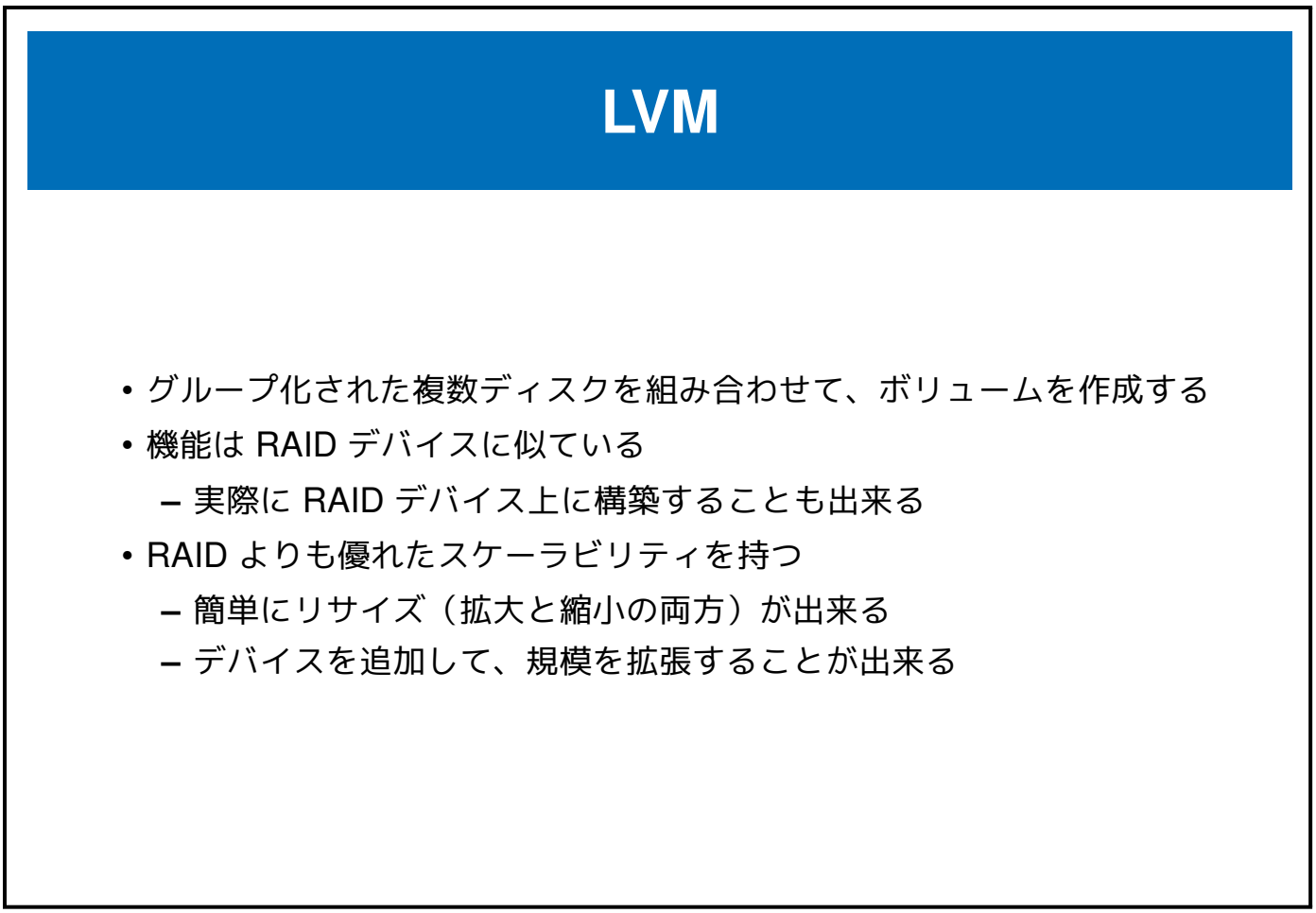
このセッションが終わるころには、次のことができるようになっているはずです

- **LVM** の概念を説明する。
- 論理ボリュームの使い方を理解する。
- 論理ボリュームを作成する。
- 論理ボリュームを表示する。
- 論理ボリュームをリサイズする。
- LVM スナップショットを使用する。

注目

** これらのセクションの一部または全体をオプション扱いとする場合があります。これらには補足資料、専門トピック、または高度な話題が含まれインストラクターが教室の状況や時間制約に応じてこれらの内容を紹介するかを判断します。

23.1 論理ボリューム管理 (LVM)



LVM (**L**ogical **V**olume **M**anagement) は、１つの仮想パーティションを複数のチャンクに分割します。各チャンクは、異なるパーティションやディスクに配置できます。

LVM には多くの利点があります。特に、論理パーティションとファイルシステムのサイズ変更や、ストレージ領域の追加やデータの再配置などは非常に簡単にできます。

１つ以上の **物理ボリューム**（ディスク パーティション）を、**ボリューム グループ** にグループ化します。次に、ボリューム グループを **論理ボリューム** に再分割します。論理ボリュームはシステムの物理ディスク パーティションを模倣したもので、通常のパーティションとしてフォーマットしてファイルシステムにマウントできます。

物理ボリュームと論理ボリュームの 作成、削除、サイズ変更 などを行う、さまざまなコマンドライン ユーティリティがあります。大半の **Linux** ディストリビューションは **system-config-lvm** と呼ばれるグラフィカルツールを使用します。しかしながら **RHEL** がこのツールのサポートを止めたので、最新のファイルシステムタイプで利用可能な信頼性が高いグラフィカルツールがありません。しかし、コマンドライン ユーティリティは使いにくいものではなく、むしろ柔軟なのは幸いなことです。

LVM はパフォーマンスに影響します。**LVM** レイヤーのオーバーヘッドから生じる、明らかな追加コストがあります。ただし、非 **RAID** システムであっても、**ストライピング**（複数のディスクへのデータの分割）を使用すれば、ある程度の並列化の向上を実現できます。

論理ボリュームには **RAID** に似た機能があります。実際に **RAID** デバイス上に構築することが出来ます。この構成では **LVM** のスケーラビリティを持った論理ボリュームに **RAID** デバイスの冗長性を付加することが出来ます。

LVM は **RAID** よりも優れたスケーラビリティを備えています。論理ボリュームは簡単にサイズ変更できます。つまり、必要に応じて拡大または縮小できます。さらに領域が必要な場合は、いつでも論理ボリュームにデバイスを追加できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

23.2 ボリューム と ボリューム グループ

ボリューム と ボリューム グループ

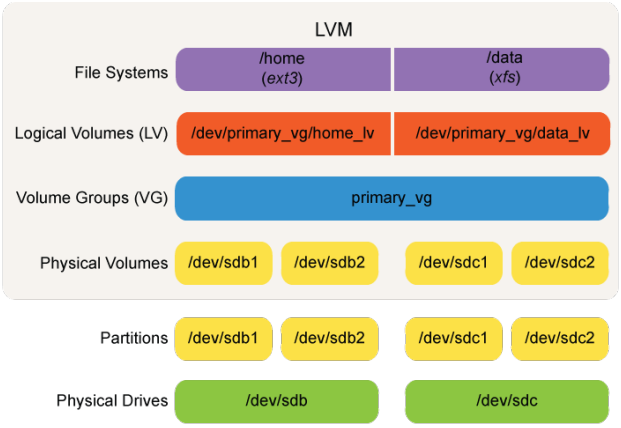


図 23.1: **LVM のコンポーネント**

- デバイスは物理ボリュームに変換される
- 複数の物理ボリュームがグループ化され、ボリューム グループになる
  - ボリューム グループは、エクステントに分割される
  - エクステントのサイズは設定できる (デフォルトは 4MB)
- ボリュームグループから、論理ボリュームが配置される
  - エクステントのサイズと数は、変更することができる
  - ファイルシステムは論理ボリューム上に構築される
  - 任意の名前を付けることが出来る

パーティションは物理ボリュームに変換され、複数の物理ボリュームはボリューム グループにグループ化されます。システム上に複数のボリュームグループが存在する場合があります。

ボリューム グループ内の領域は **エクステント** に分割されます。**エクステント** のサイズはデフォルトで 4MB ですが、割り当て時に簡単に変更できます。

ボリュームグループの生成と操作には、以下のように必ず **vg** で始まるコマンドライン ユーティリティが用意されています。

- **vgcreate**: ボリューム グループの生成します。
- **vgextend**: ボリューム グループに物理ボリュームを追加します。
- **vgreduce**: ボリューム グループを縮小します。

ボリューム グループに参加または離脱させる物理パーティションを操作するユーティリティは **pv** で始まり、次のものがあります。

- **pvcreate**: 指定パーティションを物理ボリュームとして管理できるようにします。
- **pvdisplay**: 使用されている物理ボリュームの詳細な情報を表示します。
- **pvmove**: ボリューム グループ内の 1 つの物理ボリュームから、他の物理ボリュームにデータを移動します。これは何らかの理由でディスクまたはパーティションが削除された場合に、必要になることがあります。これは次のコマンドに続きます。
- **pvremove**: 指定したパーティションにある物理ボリュームを削除します。

`man lvm` と入力すると **LVM** ユーティリティの説明が表示されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

23.3 論理ボリュームの利用

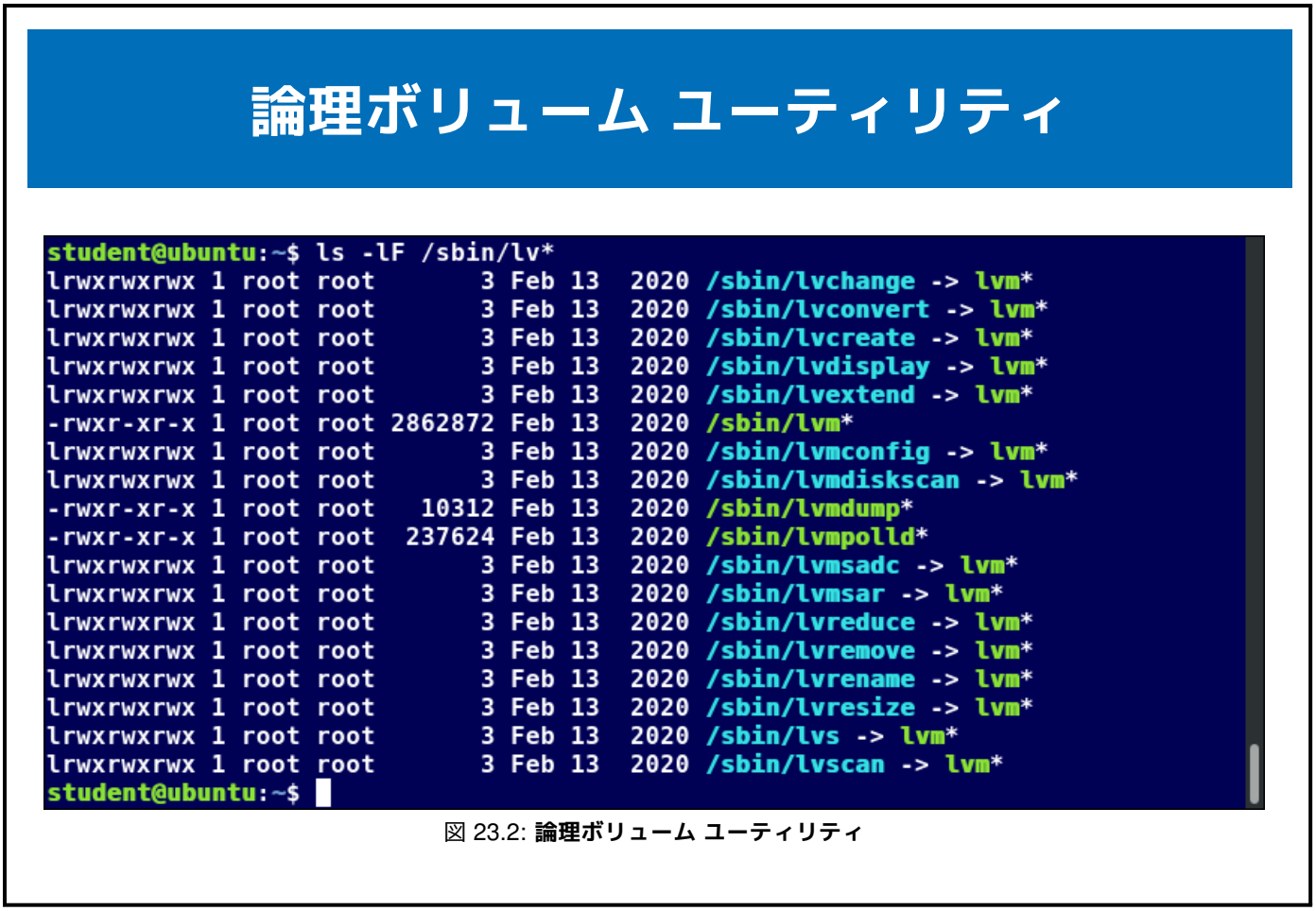


図 23.2: **論理ボリューム ユーティリティ**

論理ボリュームの操作には、多くのユーティリティがあります。当然のことながらそれらは全て **lv** で始まります。
ここでは最も一般的に使用されるものについて説明しますが、次のコマンドで短いコマンドリストを取得できます。

```
$ ls -lF /sbin/lv*
```

これらのユーティリティは `/usr/sbin` ではなく `/sbin` にあります。これらユーティリティが、ブート、または修復とリカバリのいずれかに必要な場合があるためです。

そのほとんどは、全ての作業を行う Swiss army knife（＝万能）プログラムである **lvm** にシンボリックリンクされており、呼び出される名前から、何のためのユーティリティなのか想像できます。この命名ルールはほとんどの **pv*** と **vg*** ユーティリティにも当てはまり、とてもわかりやすくなっています。

```
student@ubuntu:~$ ls -lF /sbin/pv*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvchange -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvck -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvcreate -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvdisplay -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvmove -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvremove -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvresize -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvs -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/pvscan -> lvm*
student@ubuntu:~$
```

図 23.3: **物理ボリューム ユーティリティ**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

論理ボリュームの作成

1. ディスクドライブに、パーティションを作成する
   (**fdisk** で指定するディスクタイプ番号は 8e)
2. パーティションに、物理ボリュームを作成する
3. ボリュームグループを作成する
4. ボリュームグループに、論理ボリュームを割り当てる
5. 論理ボリュームをフォーマットする
6. 論理ボリュームをマウントする（必要に応じ `/etc/fstab` も更新する）

lvcreate は、ボリューム グループから論理ボリュームを割り当てます。サイズは、バイト単位、またはエクステント数で指定できます。（デフォルトは 4MB です）任意の名前を使用できます。

lvdisplay は使用可能な論理ボリュームの情報を表示します。

ファイルシステムを論理ボリュームに割り当て、いつもと同じように **mkfs** でフォーマットします。

たとえば、既に 2 つのパーティション `/dev/sdb1` と `/dev/sdc1` があり、これらのパーティションのファイルタイプが 8e に設定済みでなら、以下のコマンドで論理ボリュームを構成できます。

```
$ sudo pvcreate /dev/sdb1
$ sudo pvcreate /dev/sdc1
$ sudo vgcreate -s 16M vg /dev/sdb1
$ sudo vgextend vg /dev/sdc1
$ sudo lvcreate -L 50G -n mylvm vg
$ sudo mkfs -t ext4 /dev/vg/mylvm
$ mkdir /mylvm
$ sudo mount /dev/vg/mylvm /mylvm
```

マウント設定を恒久化（= 再起動してもマウントされるように）するには、`/etc/fstab` に以下の行を追加してください。

```
/dev/vg/mylvm /mylvm ext4 defaults 1 2
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

論理ボリュームの表示

- **pvdisplay** - 物理ボリュームを表示する

```
$ pvdisplay
$ pvdisplay /dev/sda5
```

- **vgdisplay** - ボリュームグループを表示する

```
$ vgdisplay
$ vgdisplay /dev/vg0
```

- **lvdisplay** - 論理ボリュームを表示する

```
$ lvdisplay
$ lvdisplay /dev/vg0/lvm1
```

pvdisplay は物理ボリュームの情報を表示します。物理ボリューム名を省略すると全ての物理ボリュームがリストされます。
vgdisplay はボリュームグループを表示します。ボリュームグループ名を省略すると全てのボリュームグループがリストされます。
lvdisplay は論理ボリュームの情報を表示します。論理ボリューム名を省略すると全ての論理ボリュームがリストされます。

```
student@ubuntu:~$ ls -lF /sbin/vg*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgcfgbackup -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgcfgrestore -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgchange -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgck -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgconvert -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgcreate -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgdisplay -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgexport -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgextend -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgimport -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgimportclone -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgmerge -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgmknodes -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgreduce -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgremove -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgrename -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgs -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgscan -> lvm*
lrwxrwxrwx 1 root root 3 Feb 13  2020 /sbin/vgsplit -> lvm*
student@ubuntu:~$
```

図 23.4: **ボリューム グループ ユーティリティ**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

23.4 論理ボリュームのサイズ変更

論理ボリュームのサイズ変更

- 物理パーティションのサイズ変更より、はるかに簡単である
- どのボリュームグループのエクステントを使うこともできる
- ファイルシステムがオンライン状態のまま、縮小ないし拡大ができる
- サイズは、エクステント数でもバイト数 (M,G, 等) で指定しても良い
- 一番確実なのは、以下のように **lvresize** を使う方法である

  ```
  $ sudo lvresize  -r -L 20 GB  /dev/VG/mylvm
  ```

  -r オプションを使うと、ボリュームサイズ変更とファイルシステムサイズ変更が一緒に行われる
- **lvextend**、**lvreduce** を **resize2fs** と組み合わせて設定もできる
- サイズ変更ユーティリティは、ファイルシステムに依存する

論理ボリューム上に構築された **ext4** ファイルシステムを拡大するには

```
$ sudo lvresize -r -L +100M /dev/vg/mylvm
```

プラス記号はサイズを増やすという意味です。サイズを拡大する時には、アンマウントさせる必要はありません。
ファイルシステムを縮小するには

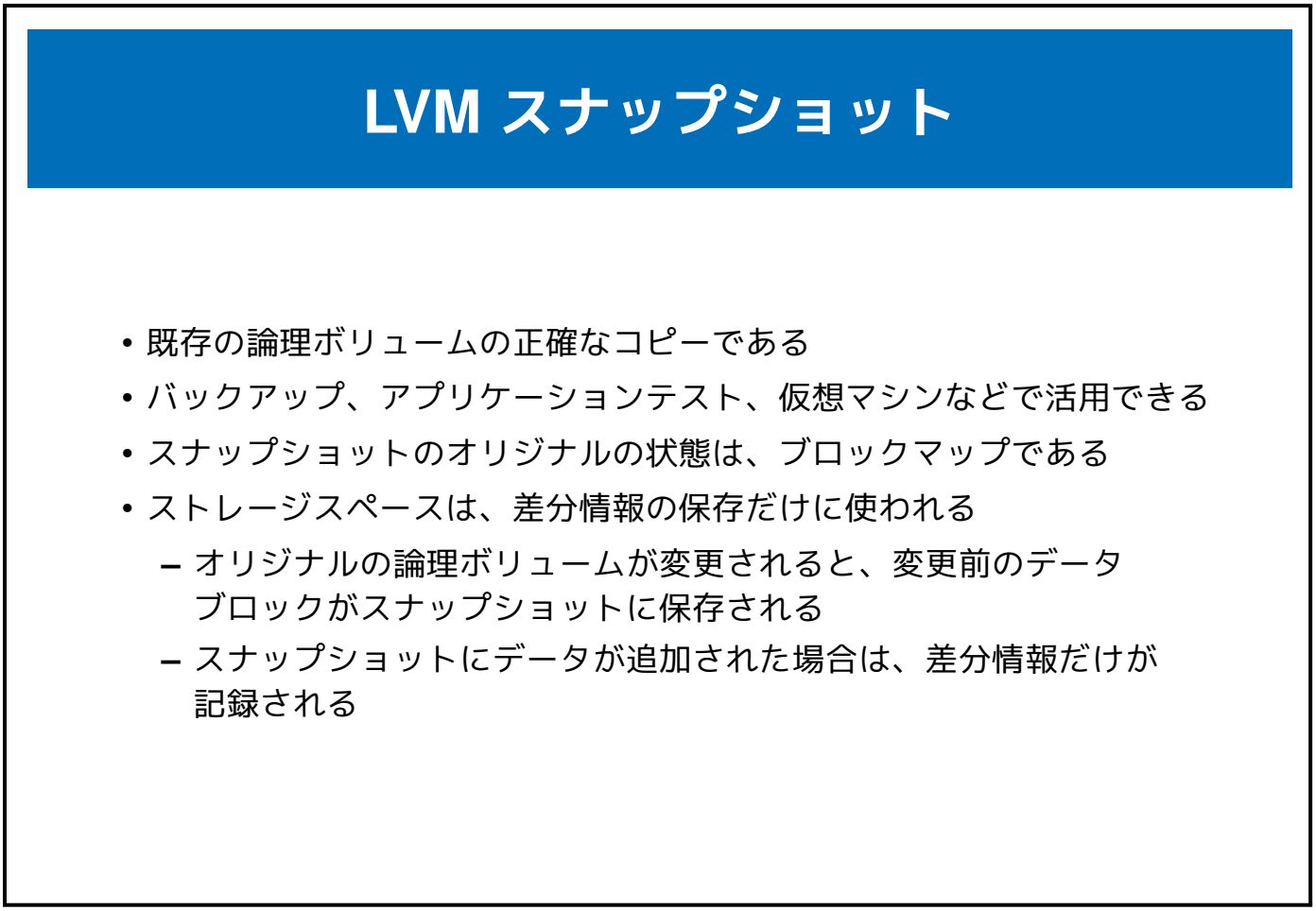
```
$ sudo lvresize -r -L 200M /dev/vg/mylvm
```

ボリュームグループのサイズ縮小をしたい場合は

```
$ sudo pvmove /dev/sdc1
$ sudo vgreduce vg /dev/sdc1
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

23.5 LVM スナップショット **



LVM スナップショットは、既存の論理ボリュームの正確なコピーを作成します。

スナップショットは、バックアップ、アプリケーションのテスト、**VM** (仮想マシン) をデプロイする時などに有効活用できます。スナップショットのオリジナルの状態は、ブロックマップとして保存されます。スナップショットは、差分情報の記録スペースだけを消費します。

- オリジナルの論理ボリュームが変更されると、変更前のデータブロックがスナップショットに保存されます。
- スナップショットにデータが追加された場合は、差分情報だけが記録されます。

既存の論理ボリュームにスナップショットを作るには:

```
$ sudo lvcreate -l 128 -s -n mysnap /dev/vg/mylvm
```

マウントポイントを作成し、スナップショットをマウントするには

```
$ mkdir /mysnap
$ mount -o ro /dev/vg/mysnap   /mysnap
```

スナップショットの利用とスナップショットの削除方法

```
$ sudo umount /mysnap
$ sudo lvremove /dev/vg/mysnap
```

スナップショットを使い終わったら、必ずスナップショットを削除してください。スナップショットを削除せず、変更の情報でいっぱいになってしまうと、スナップショット自体が自動的に無効になります。オリジナルのサイズのスナップショットは、オーバーフローすることはありません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

23.6 演習

デモ教材ビデオ

using_lvm_demo.mp4

課題 23.1: 論理ボリューム

2 つの 250MB パーティションを使って論理ボリュームを作成します。ディスク上に実パーティションを確保できると想定します。

1. 2 つの 250MB の論理ボリューム（8e）のパーティションを確保します。
2. パーティションを物理ボリュームに変更します。
3. myvg という名称のボリュームグループを作成し、2 つの物理ボリュームを標準のエクステントサイズで追加します。
4. ボリュームグループ myvg から、mylvm という名称で 300MB の論理ボリュームを割り当てます。
5. フォーマット後、論理ボリューム mylvm を /mylvm にマウントします。
6. **lvdisplay** を使って、論理ボリュームに関する情報を見ます。
7. 論理ボリュームと対応するファイルシステムを 350MB まで大きくします。

解 23.1

1. 以下を実行します。

   ```
   $ sudo fdisk /dev/sda
   ```

   どのようなハードディスクでも構いませんが、パーティションを 2 つ作成します。**fdisk** の t コマンドでパーティションタイプを 8e とします。タイプを設定しなくても問題ありませんが、設定した方が混乱が少ないです。w コマンドで、パーティション テーブルに書き込み終了します。次に

   ```
   $ sudo partprobe -s
   ```

   を実行するか、新パーティションが確実に有効になるようにリブートします。

2. ここでは、新パーティションを /dev/sdaX と /dev/sdaY とします。

   ```
   $ sudo pvcreate /dev/sdaX
   $ sudo pvcreate /dev/sdaY
   $ sudo pvdisplay
   ```

3. ```
   $ sudo vgcreate myvg /dev/sdaX /dev/sdaY
   $ sudo vgdisplay
   ```

4. ```
   $ sudo lvcreate -L 300M -n mylvm myvg
   $ sudo lvdisplay
   ```

5. ```
   $ sudo mkfs.ext4 /dev/myvg/mylvm
   $ sudo mkdir /mylvm
   $ sudo mount /dev/myvg/mylvm /mylvm
   ```

   継続的にマウントしたいときは、/etc/fstab に以下のように記入してください。

/etc/fstab の記述内容

```
/dev/myvg/mylvm /mylvm ext4 defaults 0 0
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
```

6. `$ sudo lvdisplay`

7.
```
$ df -h
$ sudo lvresize -r -L 350M /dev/myvg/mylvm
$ df -h
```

   または

```
$ sudo lvresize -r -L +50M /dev/myvg/mylvm
```

   または、古い方法ですが

```
$ df -h
$ sudo lvextend -L 350M /dev/myvg/mylvm
$ sudo resize2fs /dev/myvg/mylvm
$ df -h
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 24 章

カーネル サービスと設定

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- カーネルが果たすべき主な責任と、それをどのように達成するかを把握する。
- カーネルコマンドラインで設定できるパラメータと、そのパラメータを 1 回のシステム起動時だけ、または持続的に有効にする方法について説明する。
- ブートプロセスの不具合をトラブルシューティングする。
- これらのパラメータに関する詳細なドキュメントはどこにあるのかを把握する。
- **sysctl** を使用して、システム起動後、またはシステム再起動後も持続的にカーネルパラメータを設定する。

24.1 カーネルの概要

カーネルの概要

- カーネルは、オペレーティング システムのコアである
- カーネルは以下の役割を担う
  - システム（＝ハードウェア）の初期化と起動を行う
  - プロセスのスケジューリングを制御する
  - メモリーを管理する
  - ハードウェアアクセスを制御（仲介）する
  - アプリケーションとストレージデバイス間の I/O (Input/Output)
  - ローカルとネットワーク越しのファイルシステムをインプリする
  - セキュリティ管理、（ファイルシステムの権限管理などの）ローカルとネットワーク接続の両方をカバーする
  - ネットワークを制御する
- オペレーティング システムには、カーネル以外にも多くのコンポーネントが含まれている

狭義の **Linux** はオペレーティング システムの **カーネル** だけを指します。**オペレーティング システム** にはカーネルとやりとりするライブラリやアプリケーションなど、他の多くのコンポーネントが含まれています。

カーネルはハードウェアとソフトウェアとを接続し、競合するアプリケーションやサービス間でメモリーや CPU 時間の割り当てなどのシステムリソースを管理（調停）する重要な中枢コンポーネントです。接続された全てのデバイスを **デバイスドライバ** を使って処理し、デバイスをオペレーティング システムで使用できるようにします。

カーネル **だけ** を実行するシステムの場合、機能はかなり制限されます。特定用途向けに特化した **組み込みデバイス** だけで、このようなケースが見受けられます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

24.2 カーネル ブート パラメータ

カーネル ブート パラメータ

- grub.cfg 内のカーネル行で指定する
- 起動時に対話的に指定することもできる
- パラメータは以下のいずれかの方法で指定する
  - 単純な引数として指定
  - param=value の形式で value には文字列、整数、整数の配列などを指定できる

```
vmlinuz root=/dev/sda6 .....  noapic .... crashkernel=256M
```

- パラメータを意図的に秘密にしたり、隠蔽したりすることはできない
- カーネルソース内の kernel-parameters.txt 、またはネット上のドキュメントが https://kernel.org/doc/html/latest/admin-guide/kernel-parameters.html にある

カーネル オプションは、カーネルを指定する行の最後に、項目をスペースで区切りで記述します。
カーネル ブートパラメータの例

```
linux /boot/vmlinuz-5.19.0 root=/dev/sda5 ro crashkernel=512M quiet selinux=0
```

- root: ルート ファイルシステム（root=UUID=... や root=/dev/sda5、root=LABEL=CentOS9 などの形式がある。
- ro: ブート時にルート デバイスをリード オンリーでマウントします。
- crashkernel=512M: **kdump** 機能による**カーネル クラッシュ ダンプ**のために確保するメモリー量
- quiet: ほとんどログメッセージを出さない
- selinux=0: **selinux** を無効にする

マシンにインストールされたカーネルのドキュメントの **man bootparam** と kernel-parameters.txt を見てください。これはカーネルソースか、**kernel-doc** のような名前のパッケージにあります。さらに簡単なのは、https://kernel.org/doc/html/latest/adminguide/kernel-parameters.html をオンラインで見ることができます。

Linux ディストリビューションでは、そのディストリビューション特有のパラメータが追加されることが多いので注意が必要です。（上の例の rhgb は **Red Hat Graphical Boot** の略です）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

24.3 カーネル コマンドライイン

カーネル コマンドライン

- システム起動時には **カーネル コマンドライン** でパラメータが渡される
- /boot 下にある grub.cfg という **GRUB** の構成ファイルの中の linux か linux16 で示される行に書かれる。
- 起動時に対話的な変更（= パラメータの上書き）ができる
- 実際の設定方法は **Linux** ディストリビューションとバージョンに依存

```
linux /boot/vmlinuz-4.15.0-58-generic \
      root=UUID=5cec328b-bcd6-46d2-bcfa-8430257cd7a5 \
      ro find_preseed=/preseed.cfg auto noprompt \
      priority=critical locale=en_US quiet crashkernel=512M-:192M
```

または

```
$ cat /proc/cmdline
```

```
BOOT_IMAGE=/boot/vmlinuz-4.15.0-58-generic \
     root=UUID=5cec328b-bcd6-46d2-bcfa-8430257cd7a5\
     ro find_preseed=/preseed.cfg auto noprompt \
     priority=critical locale=en_US quiet crashkernel=512M-:192M
```

カーネル コマンドラインのサンプルはディストリビューションによりますが、以下のようなものです。

```
linux /boot/vmlinuz-5.19.0 root=UUID=7ef4e747-afae-48e3-90b4-9be8be8d0258 ro quiet

linuxefi /boot/vmlinuz-5.2.9 root=UUID=77461ee7-c34a-4c5f-b0bc-29f4feecc743 ro crashkernel=auto rhgb quiet
↪  crashkernel=384M-:128M
```

そしてこの行は /boot/grub などの boot 下のサブディレクトリの grub.cfg または /boot/efi/EFI/centos/grub.cfg などのファイルの中にあります。

vmlinuz ファイル以降で指定されているものは全てオプションです。カーネルが理解できないオプションは、全てシステムで最初に実行されるユーザープロセスの **init** (pid=1) に渡されます。システムの起動に使用したコマンドラインを確認するには

```
$ cat /proc/cmdline
```

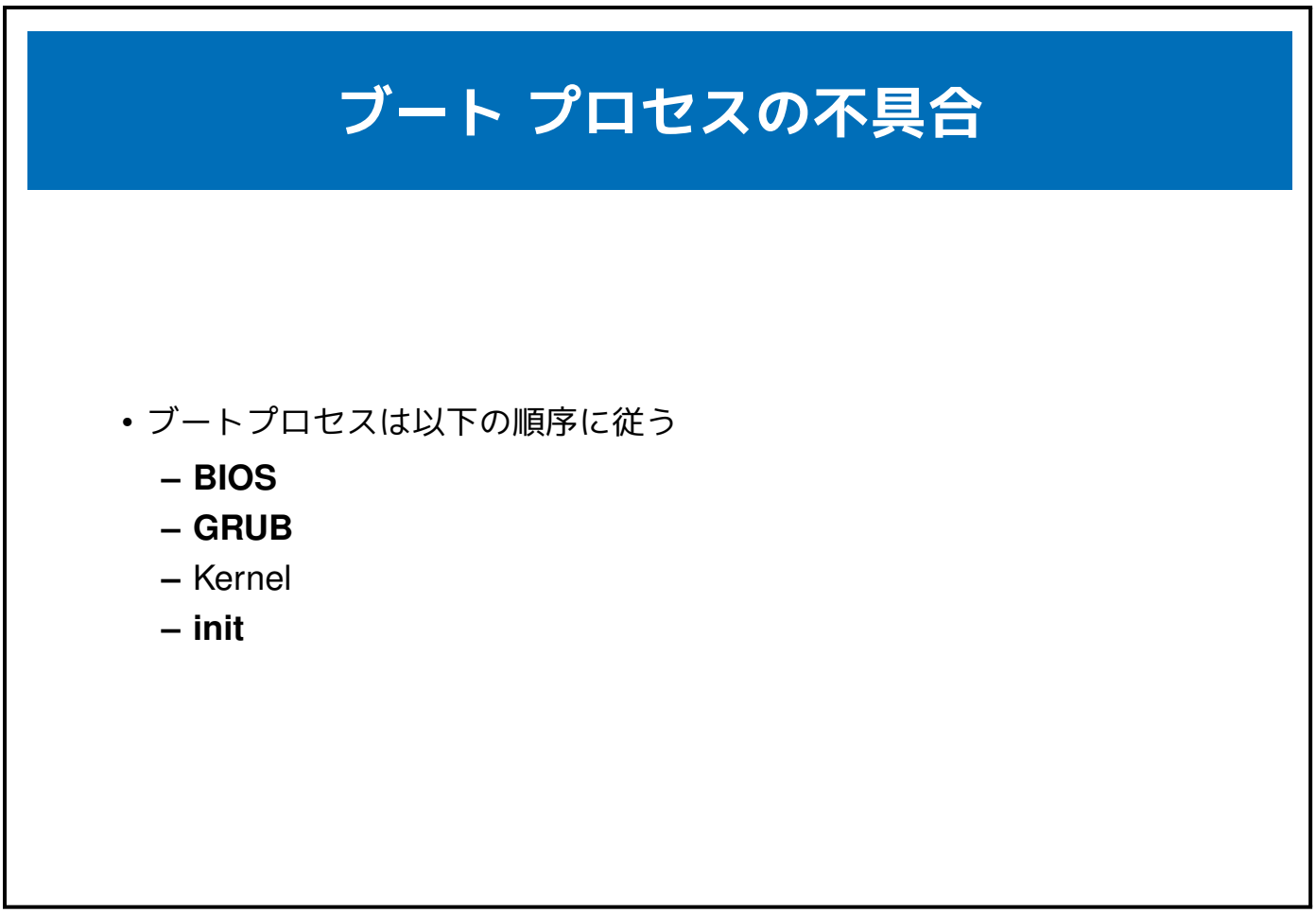
```
BOOT_IMAGE=(hd0,msdos2)/boot/vmlinuz-5.19.0 root=UUID=7f7221b8-60d8-41b9-b643-dfcc80527c37 ro rhgb quiet
↪  crashkernel=512M
```

BLSFG で状況が変わっていく

現在 **Fedora** と **RHEL/CentOS** がカーネル コマンドラインを置き換える **BLSFG** (Boot Loader Specification Configuration) を使用しています。情報は /boot/grub2/grubenv または /boot/loader/entries 内のファイルで見ることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

24.4 ブート プロセスの不具合



システムが正しく起動しない、または完全に起動しない場合には、各ステージで何が起こるかを熟知していることが重要です。**BIOS** の処理までは正常に実行されたと仮定すると、以下のような状態になる可能性があります。

ブートローダーの画面が表示されません:
GRUB の設定ミスやブートセクターが壊れていないか確認する。 ブートローダーを再インストールする必要がありますか？

カーネルの読み込みに失敗しました:
ブートプロセス中にカーネルが **パニック**になる場合、カーネルの設定ミスや破損、**GRUB** 設定ファイルのカーネルブートパラメータが正しくない可能性が高いです。過去にカーネルが正常に起動したことがある場合は、カーネルが破損しているか、不正なパラメータが供給されたかのどちらかです。カーネルを再インストールするか、起動時にインタラクティブな **GRUB** メニューに入り、最小限のコマンド ライン パラメータを使用して、その方法で修正することができます。あるいは、レスキューイメージでブートしてみるのもいいでしょう。

カーネルはロードするが、ルートファイルシステムのマウントに失敗した:
想定されるおもな原因としては

1. **GRUB** の設定ファイルの記述に間違いがある
2. `/etc/fstab` の設定ファイルの記述に間違いがある
3. ルート ファイル システムの形式がカーネルや **initramfs** サポートされていない

init の処理中に失敗した:
init 処理の中では色々な問題が発生する可能性があります。もし以前に別のカーネルで正常動作していたなら、それは大きな手がかりとなります。破損したファイルシステム、スタートアップスクリプトのエラーなどにも注意してください。3（グラフィックなし）や 1（シングルユーザモード）など、より低いランレベルで起動してみてください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

24.5 sysctl

sysctl

- /proc/sys 内の設定が **sysctl** で実行される
- 変数名は /proc/sys のパスに対応している
- カーネルパラメータの表示、設定、変更は

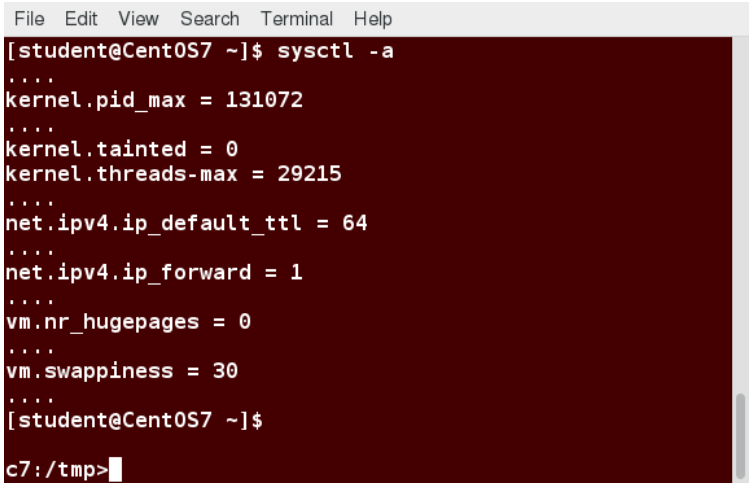


図 24.1: **sysctl**

sysctl インターフェイスは、実行中のカーネルパラメータを読み取って調整するために使用します。以下を実行することで現在の値を表示できます。

```
$ sysctl -a
```

以下の 2 つのコマンドは同じことをします。

```
$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
$ sudo sysctl net.ipv4.ip_forward=1
```

最初のコマンド文では **echo** 付きの単純な **sudo** を使用できないことに注意してください。コマンドは、ここに示した複雑な方法で実行するか root として実行する必要があります。

設定が /etc/sysctl.conf に書かれている場合、ブート時に設定を修正できます。以下のように入力します。

```
$ sudo sysctl -p
```

ファイルを即座に読み込み、全てのパラメータを指定されたとおりに設定します。これもブートプロセスの一部です。

/usr/lib/sysctl.d と /usr/lib/sysctl.d

systemd の登場により状況はもう少し複雑になりました。ベンダーは /usr/lib/sysctl.d/ ディレクトリ内に設定ファイルを置きます。これらのファイルは /etc/sysctl.d にあるファイルに追加または置換できます。ただし以前からある自己文書化されている（＝見れば分かるよう具体的に書かれている）ファイル (/etc/sysctl.conf) も引き続きサポートされています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

24.6 演習

デモ教材ビデオ

using_sysctl_demo.mp4

課題 24.1: sysctl によるシステムの調整

1. 自分のシステムに **ping** ができるかチェックします。
2. `net.ipv4.icmp_echo_ignore_all` の値を表示します。これはシステムが **ping** に応答するかしないかの指定です。0 の場合 ping に応答します。
3. **sysctl** コマンドを使って `net.ipv4.icmp_echo_ignore_all` に 1 を設定します。ping に応答するか確認します。
4. `net.ipv4.icmp_echo_ignore_all` の値を 0 に戻して、もとの動作に戻るか確認します。
5. `/etc/sysctl.conf` を修正して値を変更することで、リブートしなくても本設定を有効にできます。
6. これらが正しく動作することを確認します。

全てを終えたら、システムが元に戻るようにリセットします。

解 24.1

ping の宛先は `localhost`、`127.0.0.1`（ループバックアドレス）のどちらか、または実際の IP アドレスでもかまいません。

1.
```
$ ping localhost
```

2.
```
$ sysctl net.ipv4.icmp_echo_ignore_all
```

3.
```
$ sudo sysctl net.ipv4.icmp_echo_ignore_all=1
$ ping localhost
```

4.
```
$ sudo sysctl net.ipv4.icmp_echo_ignore_all=0
$ ping localhost
```

5. 以下の行を `/etc/sysctl.conf` に追加してください：

in /etc/sysctl.conf の追加内容

```
net.ipv4.icmp_echo_ignore_all=1
```

そして以下を実行してください。

```
$ sysctl -p
```

6.
```
$ sysctl net.ipv4.icmp_echo_ignore_all
$ ping localhost
```

`/etc/sysctl.conf` への変更は恒久的なので、前の状態に戻したいと思うでしょう。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 24.2: プロセス ID の最大値を変更する

通常 **Linux** システムでは、最初のユーザープロセスの **init** プロセスがプロセス番号（PID）PID=1 となりその後、プロセスが生成されるごとに１つずつ番号が大きくなっていきます。（プロセスが死ぬときも同様です）

しかし、PID が /proc/sys/kernel/pid_max（32768 : 32K）に達したとき、また小さい番号に戻ります。つまり 32K 個以上のプロセスは実行できないということです。

1. PID の最大値を取得します。
2. 現在発行済の PID を取得します。
3. pid_max を現在の発行値より小さくします。
4. 新しいプロセスを生成して PID の値を確認します。

解 24.2

重要

以下では２つの方法を演習します。一つは **sysctl** を実行する方法、もう一つは直接 /proc/sys/kernel/pid_max に echo する方法です。**echo** する方法は root になる必要があります; **sudo** は使えません。なぜかがわからないときは、自分で調べてください。

1.
```
$ sysctl kernel.pid_max
$ cat /proc/sys/kernel/pid_max
```

2. 入力します。

```
$ cat &
```

```
[1] 29222
```

```
$ kill -9 29222
```

3.
```
$ sudo sysctl kernel.pid_max=24000
$ echo 24000 >  /proc/sys/kernel/pid_max # This must be done as root
$ cat /proc/sys/kernel/pid_max
```

4.
```
$ cat &
```

```
[2] 311
```

```
$ kill -9 311
```

注目

（pidmax となって）PID 番号の振り直しを行う時カーネルは PID = 300 で始まり、それより低い値ではないことに注意してください。プロセスに対して PID を新しい割り当てることは実際には簡単ではないことに気付いたでしょうか。システムは（pidmax となって）番号がすでに一回転している可能性があるため、カーネルは新しい PID を生成するときに、その PID がまだ使用されていないことを常に確認する必要があるのです。この確認方法について、**Linux** カーネルはシステム上のプロセスの数に依存しない非常に効率的なを持っています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 25 章

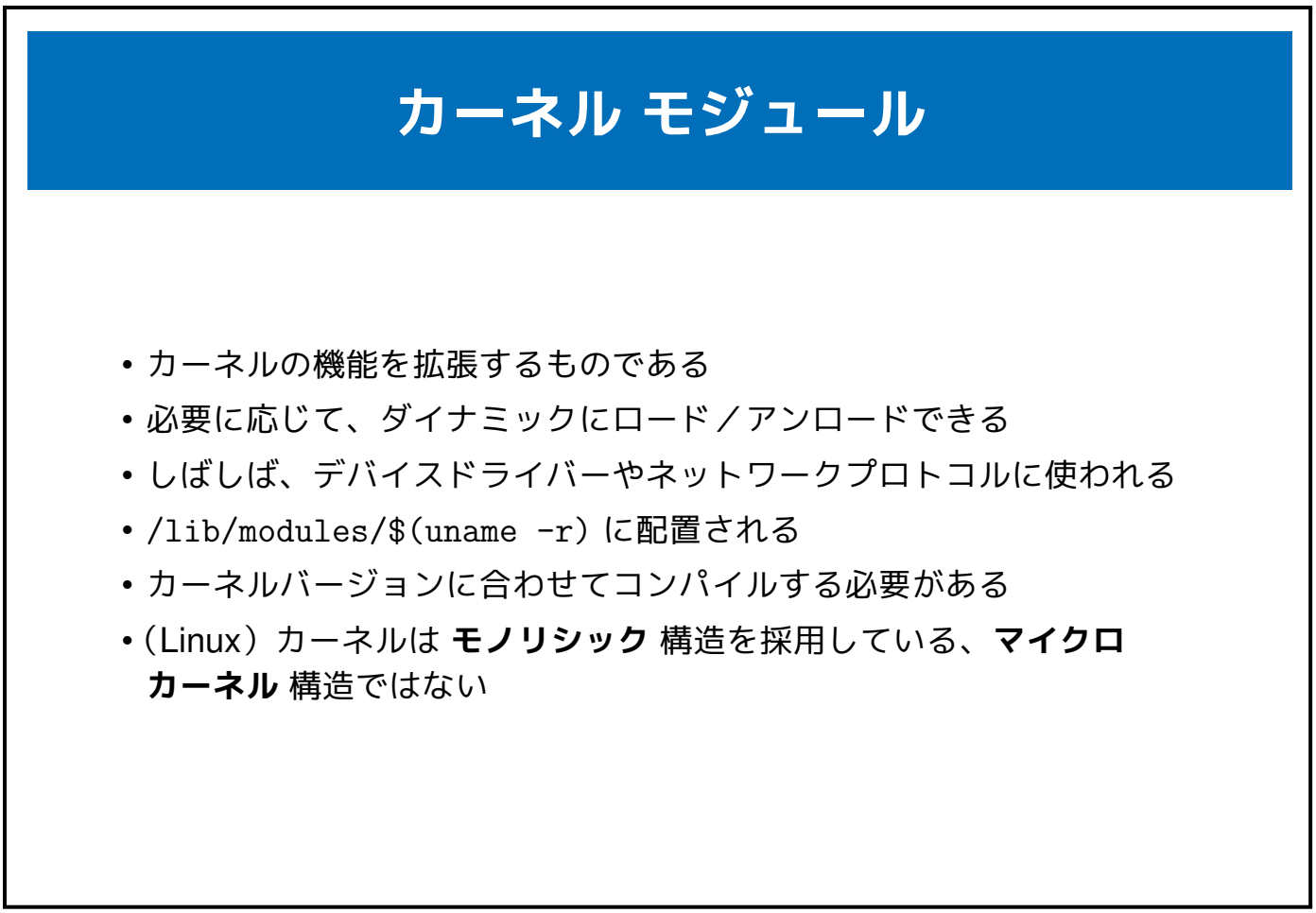
カーネル モジュール

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- カーネルモジュールを活用する利点を列挙する。
- カーネルモジュールのロード、アンロードには **insmod**、**rmmod**、**modprobe** を使用する。
- カーネルモジュールに関する情報を調べるには **modinfo** を使用する。
- モジュールの設定方法について説明する。

25.1 カーネル モジュール



Linux カーネルは広範囲に **モジュール** を使用します。**モジュール** には重要なソフトウェアが含まれ、システム起動後であっても、必要に応じてロードおよびアンロードができます。

システム内や周辺機器として接続された、ハードウェアを制御するための **デバイスドライバ** の多くは、モジュールとして組み込まれています。ネットワークプロトコルの制御、さまざまなファイルシステムサポート、その他さまざまな目的のためにもモジュールが利用されます。

モジュールをロードする時には、モジュールの挙動を制御するパラメータを指定することが可能です。パラメータを変更することによって、条件や要件の変化に対して極めてフレキシブルかつ迅速に適合できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

lsmod でモジュール一覧を表示

```
c8:/tmp>lsmod
Module                  Size  Used by
vmnet                  57344  13
vmmon                 126976  0
rfcomm                 90112  4
nft_fib_inet           16384  1
bnep                   28672  2
vmw_vsock_vmci_transport    32768  0
vmw_vmci               81920  1 vmw_vsock_vmci_transport
kvm_intel             270336  0
kvm                   958464  1 kvm_intel
irqbypass              16384  1 kvm
ext4                  905216  5
mbcache                16384  1 ext4
jbd2                  147456  1 ext4
e1000e                290816  0
c8:/tmp>
```

図 25.1: **lsmod の利用**

ほとんどのカーネル機能は、常に使用される可能性がある機能も含め、モジュールとして構成できます。この柔軟性によって、開発やデバッグ中のテストでシステムを再起動する必要がが殆どなくなるので、新機能の開発効率が向上します。

他のオペレーティング システムにもモジュールのような方法はありますが、**Linux** は他のオペレーティング システムよりもはるかに積極的にモジュールを活用しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

25.2 モジュール ユーティリティ

モジュール ユーティリティ

- **lsmod** ロードされたモジュールを一覧表示する
- **insmod** モジュールを直接ロードする
- **rmmod** モジュールを直接削除する
- **modprobe** 事前に構築された、依存関係と場所の情報をもつデータベースを使用して、モジュールをロードまたはアンロードする
- **depmod** モジュールの依存関係データベースを更新する
- **modinfo** モジュールに関する情報を表示する

modprobe を使用してモジュールをロードします。

```
$ modprobe e1000e
```

`modprobe -r` を使用してモジュールをアンロードまたは削除します。

```
$ modprobe -r e1000e
```

modprobe は、モジュール依存関係のデータベースの更新を要求します。
depmod を使用してファイル、`/lib/modules/$(uname -r)/modules.dep` を生成または更新します。
insmod を使用してモジュールを直接ロードします。(この時は、モジュールの完全修飾名を指定する必要があります)

```
$ insmod /lib/modules/$(uname -r)/kernel/drivers/net/ethernet/intel/e1000e.ko
```

rmmod を使用してモジュールを直接削除します。

```
$ rmmod e1000e
```

lsmod を使用して、ロードされたモジュールをリストします。

```
$ lsmod
```

modinfo を使用して、モジュールに関する情報(パラメータを含む)を表示します。

```
$ modinfo e1000e
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

モジュール に関する留意事項

- 以下の状況では、モジュールがアンロードできない
  - アンロードしようとしたものが、他のモジュールに使われている
  - 実行中のプロセスが使っている
- 以下の状況では、モジュールがロードできない
  - 別のカーネルバージョン向けにコンパイルされている、または コンパイルオプションが異なる
  - 現在実行中のモジュールと競合する
- 通常モジュールは **modprobe** でロードする、**insmod** を使わない
- 許容できないオープンソース ライセンスをもったモジュールがロードされた時には、カーネルは **tainted** とマークする

モジュールをロードおよびアンロードする際に、留意すべき重要な点がいくつかあります。

- １つ以上の他の **モジュール** が参照しているものはアンロードできません。これらは **lsmod** で確認できます。
- １つ以上の **プロセス** で使用されているモジュールもアンロードできません。これも **lsmod** で確認できます。ただし、ネットワークデバイスドライバモジュールなど、参照カウントを記録しないモジュールがあります。このため、ネットワークスタック全体の大部分をシャットダウン再起動することなく、特定のモジュールだけを一時的に置き換えるのは非常に困難です。
- **modprobe** でモジュールをロードした場合、先にロードする必要がある他のモジュールが自動的にロードされます。
- **modprobe -r** でモジュールをアンロードする場合、そのモジュールが使用していた他のモジュールも、別のモジュールがそれを参照していなければシステムが自動的にアンロードします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

25.3 modinfo

modinfo の使用例

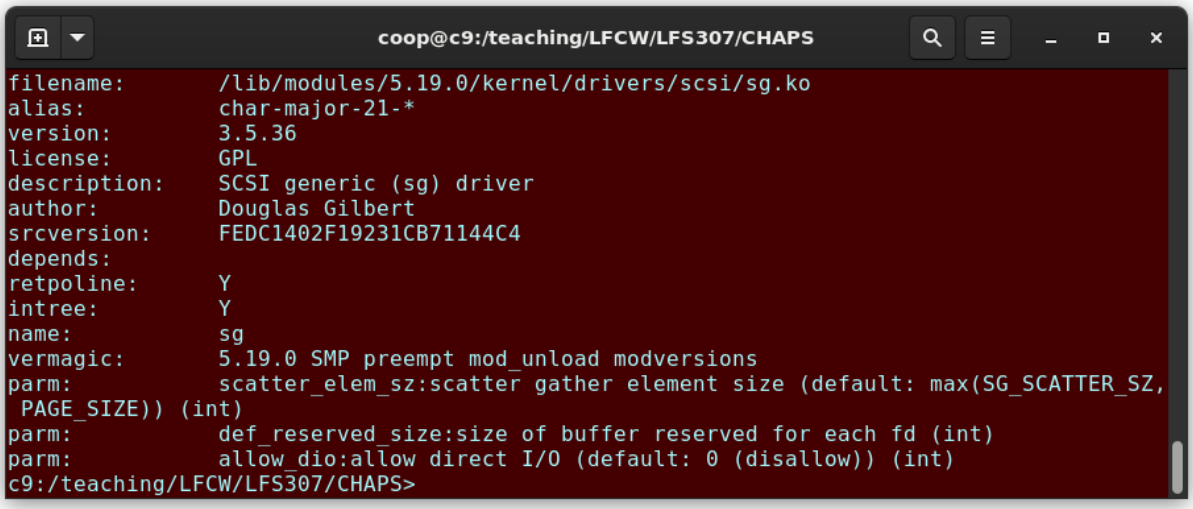


図 25.2: **modinfo**

上のスクリーンショットのように、**modinfo** はカーネル モジュールに関する情報を（現在ロードされているかどうかに関係なく）表示します。これには、バージョン、ファイル名、デバイスドライバモジュールが処理できるハードウェアデバイス、読み込み時に指定可能なパラメータに関する情報が含まれます。

さらに、多くのモジュールに関する多くの情報が `/sys` 疑似ファイルシステムのディレクトリツリーで確認できます。上の例では `/sys/module/sg` を見ていて、いくつかのパラメータは `/sys/module/sg/parameters` で読み書きできます。
次に、それらを設定する方法を示します。

多くのモジュールは、以下に示すようにロード時にパラメータ値の指定が可能です。

```
$ sudo /sbin/insmod <pathto>/e1000e.ko debug=2 copybreak=256
```

または、モジュールがシステムが想定する場所に置かれている場合は、より簡単に

```
$ sudo /sbin/modprobe e1000e debug=2 copybreak=256
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

25.4 モジュール の コンフィギュレーション

/etc/modprobe.d/

- /etc/modprobe.d サブディレクトリツリー内の .conf 拡張子で終わる全てのファイルは **modprobe** を使用してモジュールをロード、およびアンロードするときに参照（スキャン）される
- モジュールのエイリアス名（別名）を定義する
- 特定のモジュールを **blacklist** 化できる
- ロード時に渡されるパラメータを指定できる
- モジュールのロード／アンロード時に実行するコマンドを指定できる

/etc/modprobe.d 内のファイルは、**modprobe** を使用してロードするときに使ういくつかのパラメータを直接設定します。これらのパラメータには、モジュール名の **エイリアス（別名）**と自動的に提供されるオプションが含まれます。特定のモジュールを **blacklist** ファイルに登録して、それらがロードされないように指定できます。

設定はモジュールのロードまたはアンロード時に適用され、必要に応じて構成を変更できます。

/etc/modprobe.d 内のファイル形式はシンプルです。１行に１つのコマンドを書き、空行と # で始まる行は無視されます。（これはコメントの追加に便利です）行の終わりにバックスラッシュがあると次の行に継続するため（＝途中で改行できるので）ファイルが少し見やすくなります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

25.5 演習

デモ教材ビデオ

using_kernel_modules.mp4

課題 25.1: カーネルモジュール

1. 現在ロードされているモジュール一覧を表示します。
2. 現在ロードされていないモジュールをロードします。

   ディストリビューション標準のカーネルを実行してる場合には、モジュールは /lib/modules/<kernel-version>/kernel/drivers/net ディレクトリに見つけることができます。(ディストリビューションのカーネルには、システムが必要とする可能性のある、全てのデバイス、ファイルシステム、ネットワークプロトコルなどのドライバが含まれています) しかし、独自のカーネルを実行している場合は、コンパイル済のモジュールがロードされない状態で多数存在しているケースは希でしょう。

   一般的なギガビットイーサネットドライバの `e1000.ko` または、`e1000e.ko` を利用します。尚、この 2 つが同時にロードされることはありえません。
3. ロードされたカーネルモジュール一覧を再表示し、本当にロードされたか確認します。
4. ロードされたモジュールを削除します。
5. 一覧を再表示し、モジュールが削除されたか確認します。

解 25.1

1. `$ lsmod`
2. これから e1000e には、自分が利用するモジュール名を入れてください。これらの方法は問題なく実行できますが、もちろん 2 番目の方法が楽です。

   ```
   $ sudo insmod /lib/modules/$(uname -r)/kernel/drivers/net/ethernet/intel/e1000e.ko
   $ sudo /sbin/modprobe e1000e
   ```
3. `$ lsmod | grep e1000e`
4. あらためて、どちらの方法も問題なく動きます。

   ```
   $ sudo rmmod e1000e
   $ sudo modprobe -r e1000e
   ```
5. `$ lsmod | grep e1000e`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 26 章

デバイスと udev

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- デバイスノードの役割と、メジャー番号とマイナー番号の使い方を説明する。
- udev メソッドの必要性を理解し、その主要なコンポーネントをリストアップする。
- udev デバイスマネージャーの機能を説明する。
- udev ルールファイルを確認し、カスタムルールを作成する方法を学ぶ。

26.1 udev とデバイス管理

udev

- **udev** は **U**ser **Dev**ice management の略である
- **udev** は動的に、内蔵ハードウェアと周辺デバイスを検出
  - システム起動中
  - **ホットプラグ** イベント発生随時
- **udev** は必要に応じ、適切なコンフィギュレーションでデバイスドライバーのロードとアンロードを行う
- **デバイスノード** は自動的に生成され、アプリケーションと OS のサブシステムから利用される
- システム管理者は **udev** の挙動と **スペシャル udev ルール** 作成をコントロールできる

udev アクションには以下が含まれます。

- デバイス名の命名
- デバイスファイルとシンボリックリンクの生成
- ファイル属性の設定
- 必要なアクションの実行

システムにデバイスが追加ないし削除されると、**udev** はカーネルからメッセージを受け取ります。それを元に `/etc/udev/rules.d` ディレクトリにあるルールファイル（.files）を構文解析して、それらのデバイスが追加ないし削除された時のルールがあるかを確認します。

これらのルールは完全にカスタマイズ可能で、デバイスファイル名、デバイスファイルの生成、シンボリックリンクの生成、デバイスファイルの属性設定（ユーザー、グループ所有者の設定を含む）、更に起動されるアクション（指定したプログラムを実行させる）などを設定できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

26.2 デバイス ノード

デバイス ノード

- **デバイス ノード** は /dev ディレクトリに配置される
- キャラクター デバイスとブロック デバイスはデバイス ノードを持ち、ネットワーク デバイスは持たない
- プログラムはデバイス ノードを介して通常の I/O でデバイスと通信する
- 一つのデバイスドライバーが複数のデバイス ノードを使うことがある
- デバイス ノードの生成方法

  ```
  $ sudo mknod [-m mode] /dev/name <type> <major> <minor>
  ```

  例えば

  ```
  $ sudo mknod -m 666 /dev/mycdrv c 254 1
  ```

```
$ ls -l /dev
```

```
total 0
crw------- 1 coop audio      14,   4 Jul  9 01:54 audio
crw------- 1 root root       10,  62 Jul  9 01:54 autofs
lrwxrwxrwx 1 root root             4 Jul  9 01:54 cdrom -> scd0
lrwxrwxrwx 1 root root             4 Jul  9 01:54 cdrw -> scd0
crw------- 1 coop root        5,   1 Jul  9 06:54 console
....
lrwxrwxrwx 1 root root             4 Jul  9 01:54 dvd -> scd0
lrwxrwxrwx 1 root root             4 Jul  9 01:54 dvdwriter -> scd0
....
brw-r----- 1 root disk        8,   0 Jul  9 01:53 sda
brw-r----- 1 root disk        8,   1 Jul  9 01:53 sda1
brw-r----- 1 root disk        8,   2 Jul  9 06:54 sda2
....
brw-r----- 1 root disk        8,  16 Jul  9 01:53 sdb
brw-r----- 1 root disk        8,  17 Jul  9 01:53 sdb1
brw-r----- 1 root disk        8,  18 Jul  9 01:53 sdb2
....
crw-rw-rw- 1 root tty         5,   0 Jul  9 01:54 tty
crw-rw---- 1 root root        4,   0 Jul  9 14:54 tty0
crw------- 1 root root        4,   1 Jul  9 06:54 tty1
cr--r--r-- 1 root root        1,   9 Jul  9 01:53 urandom
....
crw-rw-rw- 1 root root        1,   5 Jul  9 01:54 zero
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

デバイス ノードによるハードウエア アクセス

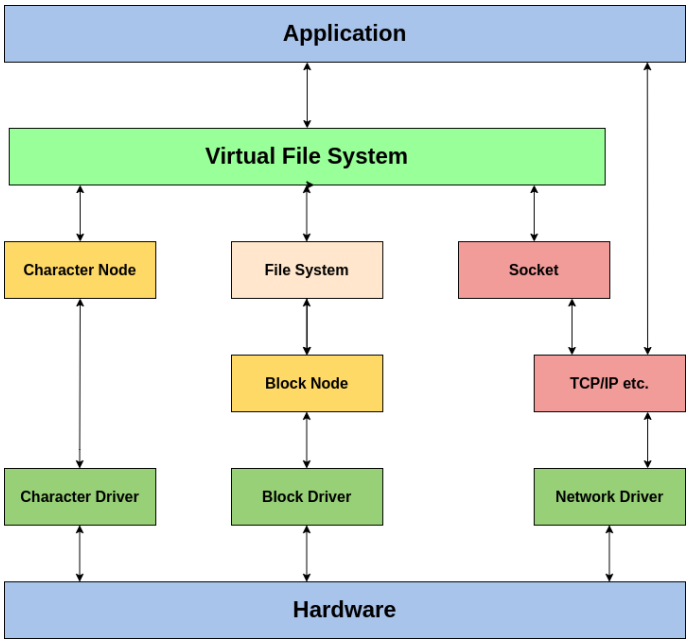


図 26.1: **デバイス ノード**

udev を使わずに静的に生成された **疑似デバイス**（`/dev/null` や `/dev/zero` を含む）に加え、`/dev` にある **デバイス ノード** がマシン上のハードウエアとの主要な、あるいは最初の接点となります。

一般的なデバイスには、以下のようなものがあります。

- /dev/sd* – ハードディスクとして見えるデバイス
- /dev/ttyS* – シリアルポート、コンソールに使われることが多い
- /dev/snd/* – いろいろなオーディオ デバイス

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

udev コンポーネント

- **libudev** ライブラリ
  デバイスに関する情報への
  アクセスを許可する
- **udevd** または **systemd-udevd**
  デーモン
  `/dev` ディレクトリを管理する
- **udevadm** ユーティリティ
  制御および診断を行う

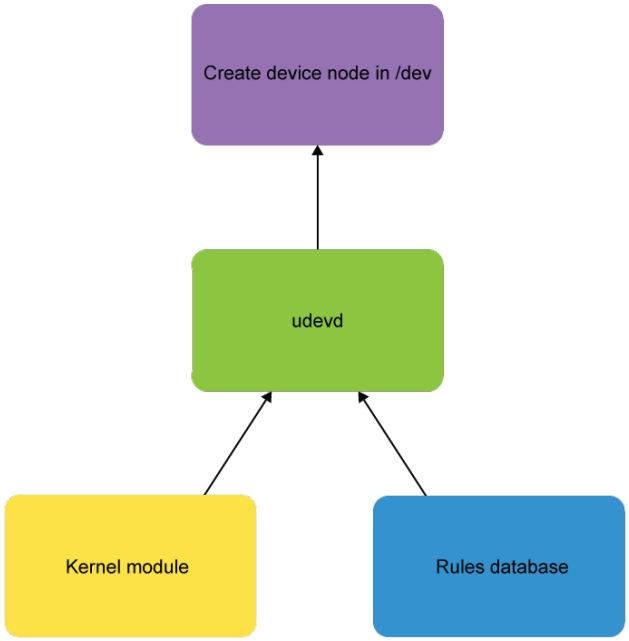


図 26.2: **udev ルール**

udev は **デーモン**（**udevd** または **systemd-udevd**）として実行され **netlink** ソケットを監視します。新しいデバイスが初期化されたり削除されると、**uevent** カーネル機構はソケットを介してメッセージを送信します。**udev** はそれを受信し、ルールに従って正しい名前とプロパティのデバイスノードを作成したり削除したりします。

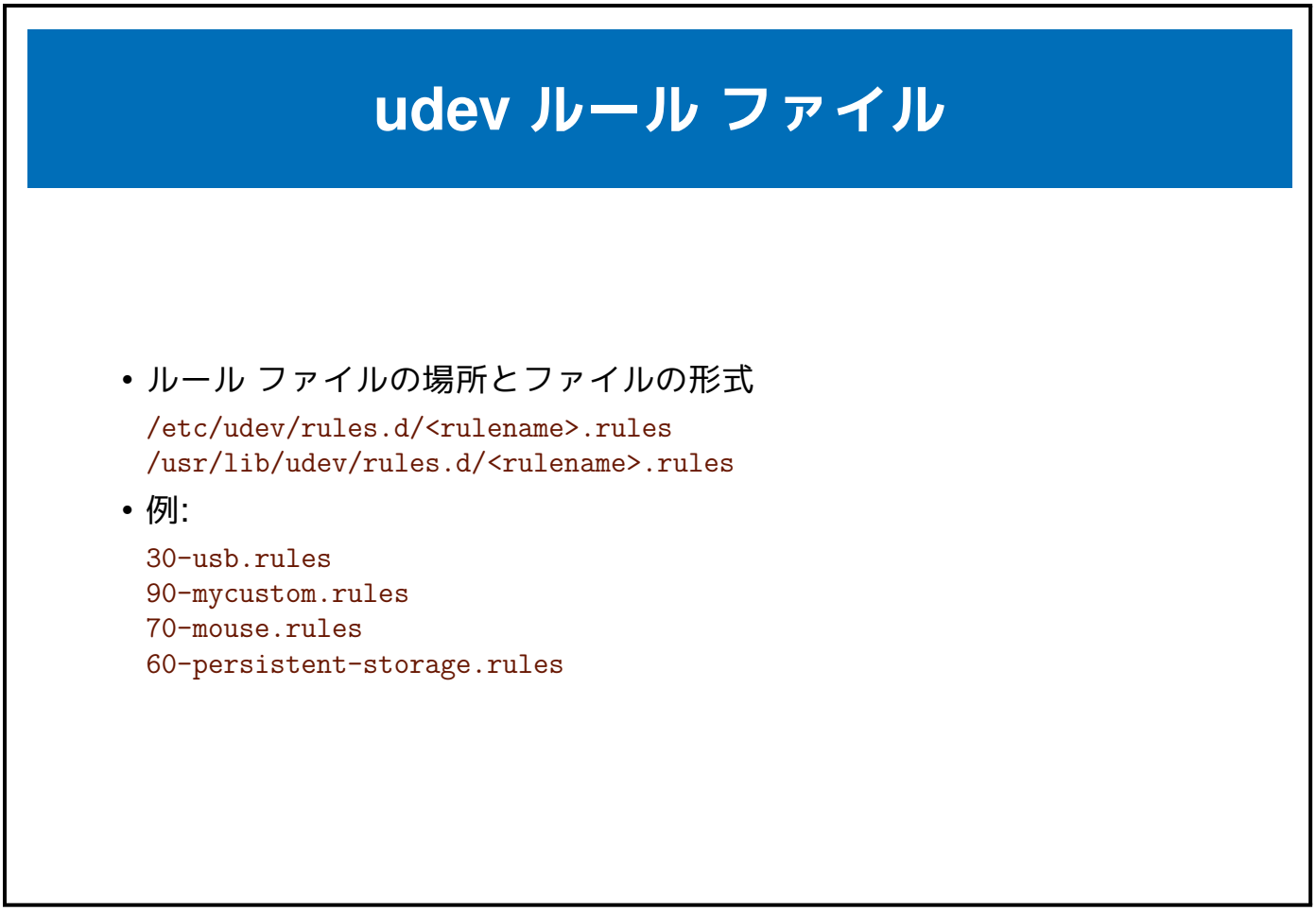
udev は、以下の 3 つのコンポーネントで構成されます。

1. デバイスに関する情報へのアクセスを許可する **libudev** ライブラリ
2. `/dev` ディレクトリを管理する **udevd** デーモン
3. 制御および診断を行う **udevadm** ユーティリティ

最もクリーンな **udev** の使い方は、プレーンなシステムを使うことです。最初のカーネル起動時には、空の `/dev` ディレクトリに必要な分だけ（＝実際に認識されたデバイスに対してだけ）デバイス ノードが生成されます。この方法を使うには、暫定的なデバイス ノード セットと **udev** インフラストラクチャが含まれた **initramfs** イメージを使ってブートする必要があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

26.3 ルール



1 行に 2 つの情報を定義します。

- 最初の部分は == で示される 1 つ以上の一致するペアで構成します。これらは、デバイスの属性や特性もしくはその両方がここに指定した値に一致するかをみます。
- 2 番目の部分はファイル名、グループの割り当て、さらにはファイルのパーミッションなど名前に値を割り当てる 1 つ以上の割り当てキーと値のペア（= KVS、キーバリューストア）で構成します。

一致するルールが見つからない場合、デフォルトのデバイスノード名とその他の属性が使用されます。

```
$ cat /etc/udev/rules.d/99-fitbit.rules
```

```
SUBSYSTEM=="usb", ATTR{idVendor}=="2687", ATTR{idProduct}=="fb01", SYMLINK+="fitbit", MODE="0666"
```

```
$ cat /etc/udev/rules.d/60-vboxdrv.rules
```

```
KERNEL=="vboxdrv", NAME="vboxdrv", OWNER="root", GROUP="vboxusers", MODE="0660"
KERNEL=="vboxdrvu", NAME="vboxdrvu", OWNER="root", GROUP="root", MODE="0666"
KERNEL=="vboxnetctl", NAME="vboxnetctl", OWNER="root", GROUP="vboxusers", MODE="0660"
SUBSYSTEM=="usb_device", ACTION=="add", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh $major $minor
↪  $attr{bDeviceClass}"
SUBSYSTEM=="usb", ACTION=="add", ENV{DEVTYPE}=="usb_device", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh $major
↪  $minor $attr{bDeviceClass}"
SUBSYSTEM=="usb_device", ACTION=="remove", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh --remove $major $minor"
SUBSYSTEM=="usb", ACTION=="remove", ENV{DEVTYPE}=="usb_device", RUN+="/usr/lib/virtualbox/VBoxCreateUSBNode.sh --remove
↪  $major $minor"
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

udev ルールの作成

- ルールの形式:

```
<match><op>value [, ...] <assignment><op>value [, ... ]
```

- サンプル

```
KERNEL=="sdb", NAME="my-spare-disk"
KERNEL=="sdb", DRIVER=="usb-disk", SYMLINK+="sparedisk"
KERNEL=="sdb", RUN+="/usr/bin/my-program"
KERNEL=="sdb", MODE="0660", GROUP="mygroup"
```

udev ルールの書式は簡単です。１行に２つの情報を定義します。最初の部分には１つ以上の一致ペアで構成します。（二重等号 == で示されます）これらはデバイスの属性や特性、もしくはその両方が、ここに指定した値に一致するか確認します。２番目の部分にはファイル名、グループの割り当て、さらにはファイルのパーミッションなど名前に値を割り当てる１つ以上の割り当てキーと設定値のペアを含みます。

一致するルールが見つからない場合、デフォルトのデバイス ノード名が使用されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ルール ファイルの例

- **Fitbit** デバイス

```
$ cat /etc/udev/rules.d/99-fitbit.rules
SUBSYSTEM=="usb", ATTR{idVendor}=="2687", ATTR{idProduct}=="fb01", \
                SYMLINK+="fitbit", MODE="0666"
```

- **kdump**/**kexec** によるクラッシュ ダンプと高速カーネル ロード設定

```
$ cat /usr/lib/udev/rules.d/98-kexec.rules
SUBSYSTEM=="cpu", ACTION=="add", PROGRAM="/bin/systemctl try-restart kdump.service"
SUBSYSTEM=="cpu", ACTION=="remove", PROGRAM="/bin/systemctl try-restart kdump.service"
SUBSYSTEM=="memory", ACTION=="online", PROGRAM="/bin/systemctl try-restart kdump.service"
SUBSYSTEM=="memory", ACTION=="offline", PROGRAM="/bin/systemctl try-restart kdump.service"
```

- **kvm** 仮想マシン ハイパーバイザーの設定

```
$ cat /usr/lib/udev/rules/.d/80-kvm.rules
KERNEL=="kvm", GROUP="kvm", MODE="0666"
```

以下のようにすると、ハードウエアにルールが適用される状況を逐次確認できます。

```
$ sudo udevadm monitor
```

USB デバイスをすばやく抜き差しして、コマンドの出力を試してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

26.4 演習

デモ教材ビデオ

using_udev_demo.mp4

課題 26.1: udev

1. **USB** デバイスが挿入されたときに、myusb というシンボリックリンクを生成するルールを作成、インプリメントします。
2. **USB** デバイスを挿入します。それは、ペン、マウス、ウェブカメラでも何でもかまいません。
   注意：ハイパーバイザーのもとで仮想マシンを実行中ならば、ゲスト OS が **USB** デバイスを見えるようにしておく必要があります。マウスのクリックで（= ハイパーバイザーの設定ユーティリティで）ホスト OS から切り離すことができます。
3. /dev ディレクトリのリストを表示し、シンボリックリンクが生成されたことを確認します。
4. **USB** を取り外します。（USB ストレージドライブの場合は、安全のためにまず **umount** してください）
5. /dev にシンボリックリンクがあるか、確認してください。

解 26.1

1. 内容が次の 1 行だけのファイル /etc/udev/rules.d/75-myusb.rules を作成します。

   ```
   $ cat /etc/udev/rules.d/75-myusb.rules
   ```

   ```
   SUBSYSTEM=="usb", SYMLINK+="myusb"
   ```

   すでに使われていないキー値である BUS を SUBSYSTEM の代わりに使用しないでください。**udev** の最近のバージョンではすでに無くなっています。

   注意: ファイル名は問題ではありません。ルール中に ACTION コンポーネントがあれば、システムはそれを実行します; 他のルールを参考にしてください。

2. デバイスを挿入します。
3. `$ ls -lF /dev | grep myusb`
4. デバイスがマウントされていれば

   ```
   $ umount /media/whatever
   ```

   ここで /media/whatever が意味しているのはマウントポイントです。これで、安全にデバイスを抜くことができます。

5. `$ ls -lF /dev | grep myusb`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 27 章

ネットワーク アドレス

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- IPv4 アドレスと IPv6 アドレスの種類を区別する。
- IP アドレスクラスについて理解する。
- ネットマスクの役割を理解する。システムのホスト名を取得、設定、変更する。
- **NTP** を使用して、時刻の設定、同期、補正を行う。

27.1 IP アドレス

IP アドレス

- IP アドレスはインターネット上のノードを一意に識別するもの
- IP アドレスは **RIP (Routing Information Protocol)** で登録する
- **IPv4** は **4 オクテット**で構成される 32 ビットアドレスである

  `148.114.252.10`

  (1 オクテット = 1 バイト)
- **IPv6** は 8 つの 16 ビットオクテットペアで構成される 128 ビットアドレスである

  `2003:0db5:6123:0000:1f4f:0000:5529:fe23`
- いくつか予約済アドレスがある
- ここではまだ主流と言える **IPv4** に焦点をあてる

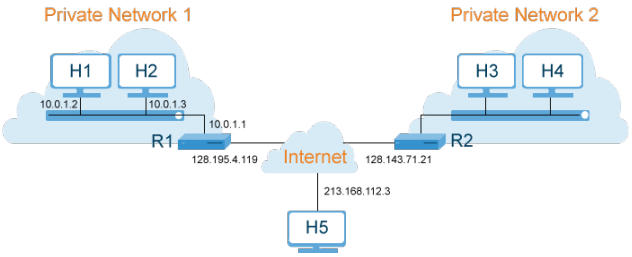


図 27.1: **IP アドレス**

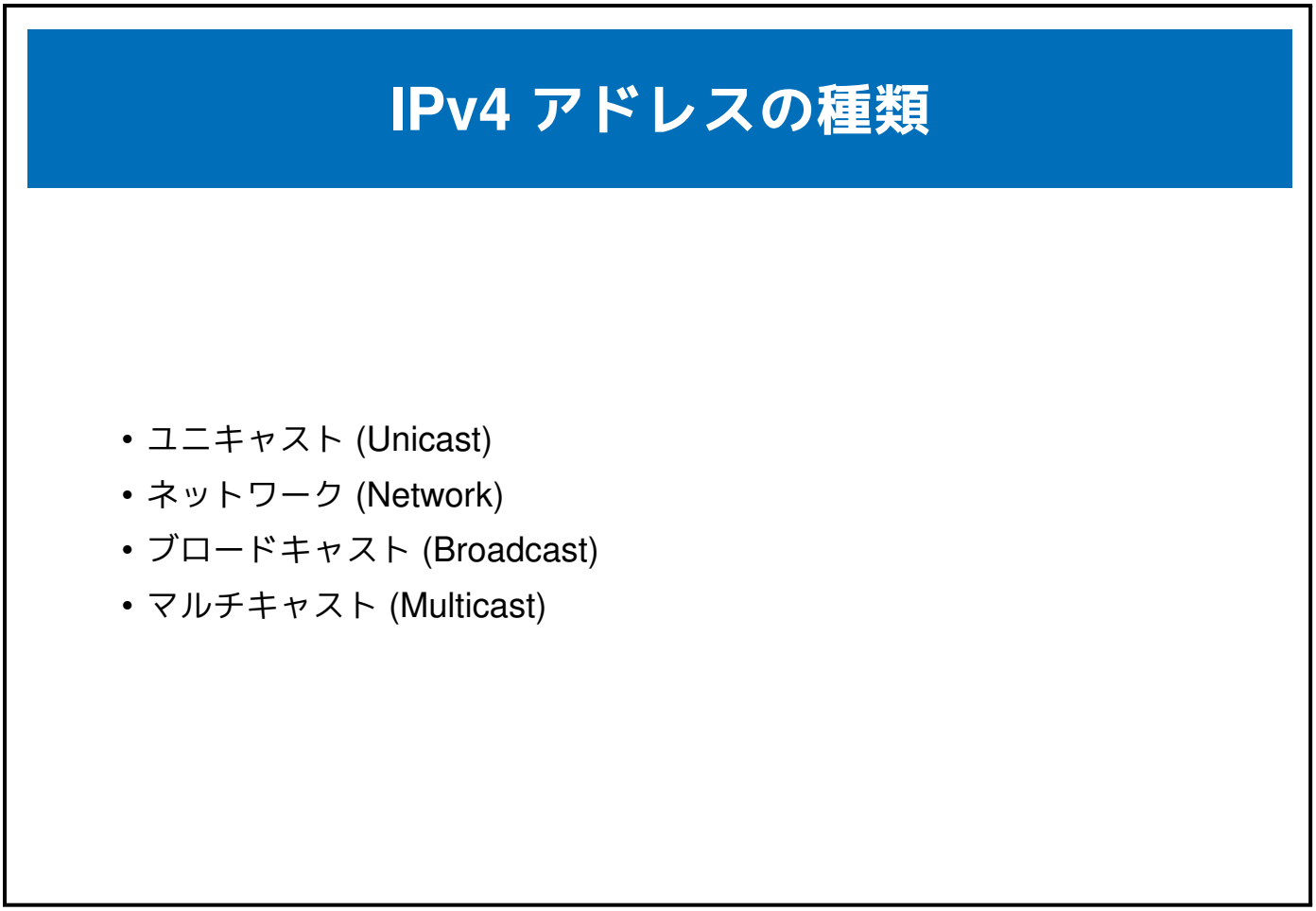
IP アドレスの利用や参照に関する推奨は、ハードウェアに起因する制限から地域起因や政治起因の境界まで、あらゆるものに関わる慣例によって、時代とともに劇的に変化してきました。

IPv4 アドレスは今でも最も広くサポートされていますが、その（相対的に）小さなアドレス空間サイズから、もう何年も前から数が不足しています。コストが重視される公衆向けサービスでは、これらの IP アドレスを少量取得することはまだ可能ですが、**IPv4** の大規模展開には、今となってはかなりの立ち上げコストがかかる可能性があります。

現時点でのベストプラクティス提案は、**IPv4** と **IPv6** の両方がサービス用に設定された **デュアル ホーム** 構成です。数十年来の長期目標は、最終的に **IPv6** を上級市民として扱い、インターネットサービスに **IPv4** アドレスが全く必要ないようにすることです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.2 IPv4 アドレスの種類



IPv4 アドレスの種類には、以下のものがあります。

- **ユニキャスト (Unicast)**:
  特定のホストに関連付けられたアドレスです。`140.211.169.4` や `64.254.248.193` のようになります。
- **ネットワーク (Network)**:
  ホスト部分 が全て 2 進数のゼロに設定されているアドレスです。例えば：`192.168.1.0`（後で説明しますが、ホスト部分は最後の 1〜3 オクテットです。この例では最後のオクテットだけが該当します）
- **ブロードキャスト (Broadcast)**:
  同一ネットワーク（= IP アドレスのホスト部分を除いたネットワーク部分が共通の意味）上にいる全てのメンバーに対して一斉通報するためのアドレスです。`172.16.255.255` や `148.114.255.255` や `192.168.1.255` の例のように、ホスト部分はビットを全て立てるのでバイト表現では 255 となります。（最初の 2 つの例ではホスト部分は最後の 2 オクテットで、3 番目の例では最後のオクテットだけです）
- **マルチキャスト (Multicast)**:
  特定の複数のノードに対して通報するアドレスです。アドレス `224.0.0.2` はマルチキャストアドレスの例です。マルチキャストアドレスを認識するように設定された特定のノードだけが、そのマルチキャストグループへ投げられたパケットを解釈します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

予約済みアドレス

- ループバックインターフェイス

  `127.x.x.x`
- アドレスが与えられていない状態

  `0.0.0.0`
- 汎用 ブロードキャスト プライベートアドレス

  `255.255.255.255`
- その他

  `10.0.0.0 - 10.255.255.255`
  `172.16.0.0 - 172.31.255.255`
  `192.168.0.0 - 192.168.255.255`

特定のアドレスとアドレス範囲は、特別な目的のために予約されています。

- `127.x.x.x`

  ループバック（ローカルシステム）インターフェイス用に予約されています。`0 <= x <= 254` の値を取ります。
  通常は `127.0.0.1` と表されます。

- `0.0.0.0`

  まだ自分のアドレスが確定していない時に設定されます。**DHCP** や **BOOTP** などのプロトコルが（IP アドレスを取得するために）サーバーと通信しようとするときにこのアドレスを使用します。

- `255.255.255.255`

  内部使用のために予約されている、汎用のブロードキャストプライベートアドレスです。これらのアドレスは、誰にも割り当てられたり登録されたりすることはありません。通常、これらはルーティングできません。

- その他の予約済みアドレスには、以下のものがあります。

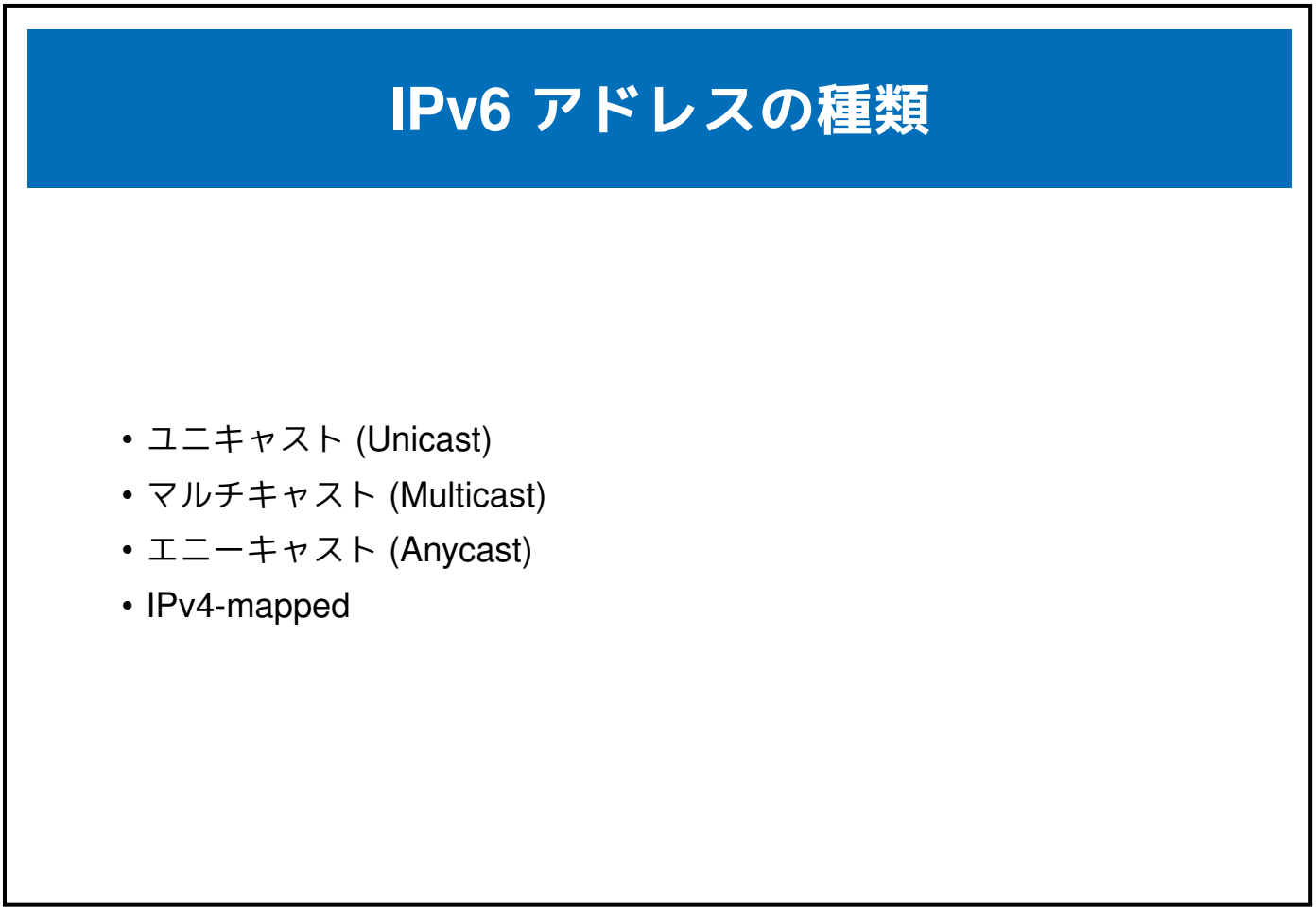
  `10.0.0.0 - 10.255.255.255`
  `172.16.0.0 - 172.31.255.255`
  `192.168.0.0 - 192.168.255.255`

  これらにはそれぞれ目的があります。たとえば、なじみのあるアドレス範囲である `192.168.x.x` は、プライベートネットワーク内のローカル通信にだけ使用されます。

IPv4 と **IPv6** の予約済みアドレスの長いリストが https://en.wikipedia.org/wiki/Reserved_IP_addresses にあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.3 IPv6 アドレスの種類



IPv6 アドレスの種類には、以下のものがあります。

- **ユニキャスト (Unicast)**:
  パケットは 1 つのインターフェイスに配信されます。
  - **Link-local**: 全てのインターフェイスに自動的に割り当てられます、ルーティングはできません。
  - **Global**: 動的、または手動で割り当てられるアドレスで、ルーティングも可能です。
  - ドキュメントの閲覧用などに予約されているアドレスです。
- **マルチキャスト (Multicast)**:
  パケットが複数のインターフェイスに対して配信されます。
- **エニーキャスト (Anycast)**:
  複数ノードのインターフェイスに同じアドレスを割り付けておき（ルーティング距離の観点から）、最も近いノードにパケットが転送されます。
- **IPv4-mapped**:
  IPv4 アドレスが埋め込まれた **IPv6** アドレスです。例えば `::FFFF:a.b.c.d/96` のようなものです。

さらに、**IPv6** には `::1/128` のように、`lo` インターフェイスに割り当てられるループバックなどの特別なアドレスがあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.4 IPv4 アドレス クラス

IPv4 アドレス クラス

- Class A -- 最上位のオクテットレンジが 1-127
  - ネットワーク数は 128、1 ネットワークあたり最大 16,777,214 のホストに対応、127.x.x.x はループバック用に予約されている
- Class B -- 最上位のオクテットレンジが 128-191
  - ネットワーク数 16,384、最大 65,534 ホスト／ネットワーク
- Class C -- 最上位のオクテットレンジが 192-223
  - ネットワーク数 2,097,152、最大 254 ホスト／ネットワーク
- Class D -- 最上位のオクテットレンジが 224-239
  - マルチキャストアドレス
- Class E -- 最上位のオクテットレンジが 240-254
  - 予約済みアドレス範囲である

歴史的に IP アドレスはいくつかの定義クラスに分類されています。クラス A、B、C はアドレスのネットワーク部分とアドレスのホスト部分を区別するために使用されます。これはルーティングの目的で使用されます。

クラス A のアドレスは、アドレスのネットワーク部分に 8 ビット、ホスト部分に 24 ビットを使用します。クラス B のアドレスは、ネットワーク部分に 16 ビット、ホスト部分に 16 ビットを使用し、クラス C のアドレスは、ネットワーク部分に 24 ビット、ホスト部分に 8 ビットを使用します。

クラス D のアドレスはマルチキャストに使用されます。クラス E のアドレスは現在使用されていません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.5 ネットマスク

ネットマスク

- Class A ネットマスク
  - 255.0.0.0 (10 進数表記)
  - ff:00:00:00 (16 進数表記)
  - 11111111 00000000 00000000 00000000 (2 進数表記)
- Class B ネットマスク
  - 255.255.0.0 (10 進数表記)
  - ff:ff:00:00 (16 進数表記)
  - 11111111 11111111 00000000 00000000 (2 進数表記)
- Class C ネットマスク
  - 255.255.255.0 (10 進数表記)
  - ff:ff:ff:00 (16 進数表記)
  - 11111111 11111111 11111111 00000000 (2 進数表記)

これまで見てきたように、**ネットマスク** はネットワーク部分に使用されるアドレスの値と、ホスト部分に使用される値を決定するために使用されます。また、ネットワークとブロードキャストアドレスの決定にも使用されます。

- Class A アドレスは 8 ビットをネットワークのアドレス指定に、24 ビットをホストのアドレス指定に使います。
- Class B アドレスは 16 ビットをネットワークに、16 ビットをホストに使います。
- Class C アドレスは 24 ビットをネットワークに、8 ビットをホストに使います。
- Class D アドレスはマルチキャスト用に使われます。
- Class E アドレスは現在使われていません。

ネットワークアドレスは、ネットマスクと IP アドレスの論理積（AND 演算- &）を計算することで取得されます。同じメディアを介して接続し、同じネットワークアドレスを共有するノード群を構成するローカルネットワークを定義できる点でも、ネットワークアドレスは興味深いです。同じネットワーク上の全てのノードは、お互いを直接見ることができます。

```
ex.  172.16.2.17  ip address
   &255.255.0.0  netmask
  -----------------
    172.16.0.0   network address
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.6 ホスト名

ホスト名の取得と設定

- 他ノードから見た、ネットワークデバイスを特定するためのラベル
- 以前は **ノード名 (nodename)** とも呼ばれていた
- インストール時に設定され、後からいつでも変更できる
- 誰でも、以下の方法でホスト名を取得できる

```
$ hostname
wally
```

- ホスト名の変更には、root 権限が必要である

```
$ sudo hostname lumpy
lumpy
```

- 大部分の **Linux** ディストーションでは、現在のホスト名が `/etc/hostname` に保存されている
- リブート後にも変更が残るように **恒久化**させるには **systemd** インフラの一つである **hostnamectl** を使う

```
$ sudo hostnamectl set-hostname lumpy
```

DNS のために、ホスト名にはピリオド (dot) とドメイン名が追加されます。そのためホスト名が `antje` のマシンの完全修飾ドメイン名 **fully qualified domain name** (**FQDN**) は `antje.linuxfoundation.org` となります。

歴史的には、ホスト名の恒久的な変更には `etc` ディレクトリツリーにある設定ファイルの変更が必要でした。設定するファイルは **Red Hat** ベースのシステムでは `/etc/sysconfig/network`、**Debian** ベースのシステムでは `/etc/hostname` 、そして **SUSE** ベースのシステムでは `/etc/HOSTNAME` です。しかし、最近のシステムなら **hostnamectl** を使うべきです。

```
$ hostnamectl
   Static hostname: c8
         Icon name: computer-desktop
           Chassis: desktop
        Machine ID: ce0c82382a8a4c80bbd6931a917a2f1c
           Boot ID: 94207b3fbd9b4891b9a94e21762a47cb
  Operating System: Red Hat Enterprise Linux 8.2
  ↪  (Ootpa) \
            dracut-049-70.git20200228.el8 (Initramfs)
             Kernel: Linux 5.11.6
       Architecture: x86-64
```

そして、使い方を参照するには

```
$ hostnamectl --help
hostnamectl [OPTIONS...] COMMAND ...
Query or change system hostname.
  -h --help              Show this help
     --version           Show package version
     --no-ask-password   Do not prompt for password
  -H --host=[USER@]HOST  Operate on remote host
  -M --machine=CONTAINER Operate on local container
     --transient         Only set transient hostname
     --static            Only set static hostname
     --pretty            Only set pretty hostname
Commands:
  status                 Show current hostname
  ↪  settings
  set-hostname NAME      Set system hostname
  set-icon-name NAME     Set icon name for host
  set-chassis NAME       Set chassis type for host
  set-deployment NAME    Set deployment environment
  ↪  for host
  set-location NAME      Set location for host
  set-location NAME      Set location for host
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.7 NTP

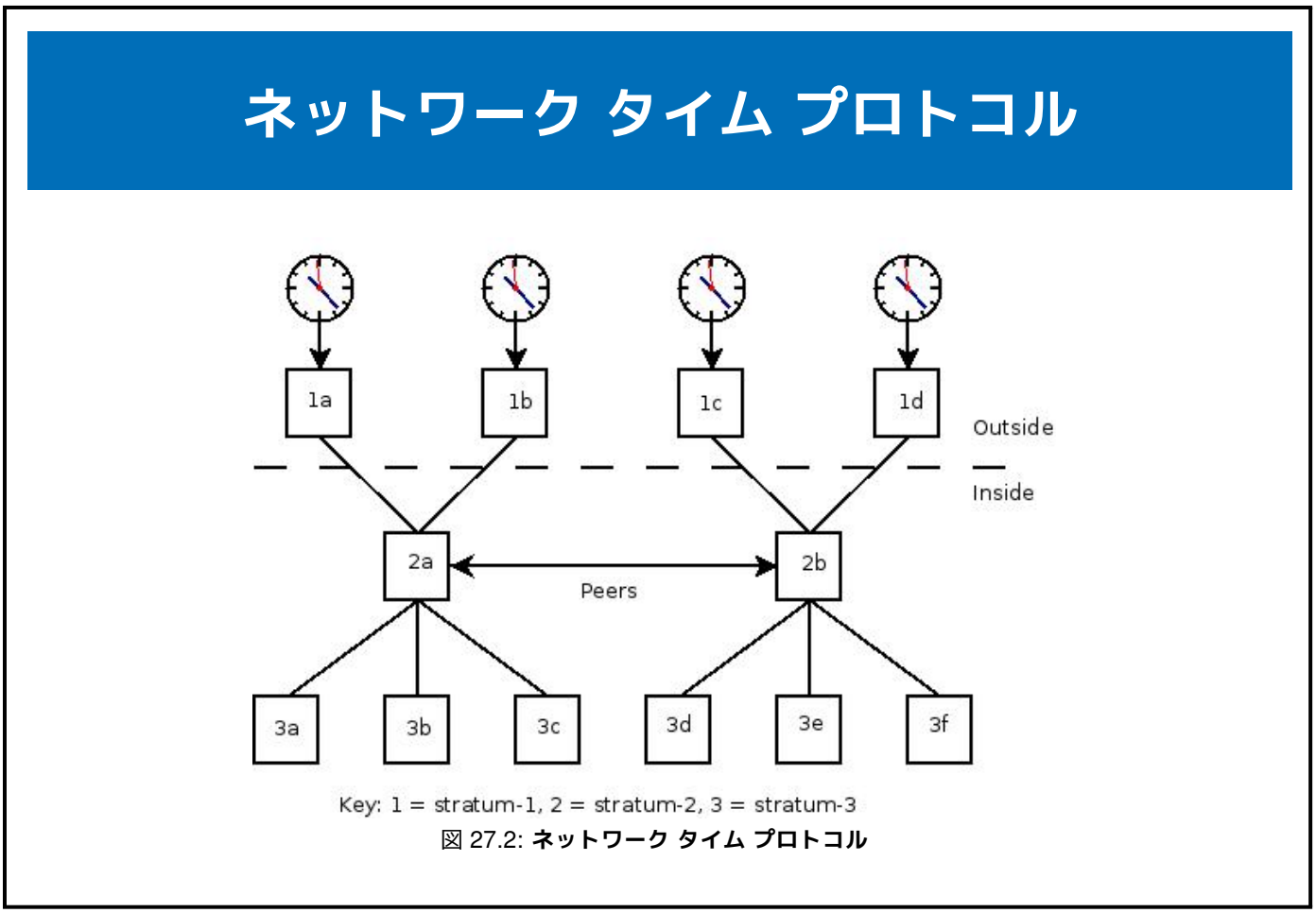


図 27.2: **ネットワーク タイム プロトコル**

多くのプロトコルは、正しく機能するために（時刻が正確ではないとしても）一貫した時間を必要とします。

多くの暗号化システムのセキュリティは、適切な時刻に大きく依存しています。また、商品取引や株式取引などの業界では、わずか数秒の違いが数百ドル、いや数千ドルを意味することがあるため、非常に正確な時間が必要となります。

ネットワークタイムプロトコルは、システムの時間を更新し、同期させるための方法です。NTP は、デーモンとプロトコルで構成されています。NTP のタイムソースは"strata " に分けられています。

- Strata 0 クロックは、特別な目的の時間デバイスです。（原子時計、GPS ラジオなど）
- strata 1 サーバーは、strata 0 ソースに直接（シリアルなどで）接続された NTP サーバーです。
- Strata 2 サーバは、NTP を使用して Strata 1 サーバを参照する全ての NTP サーバです。
- strata 3 サーバーは、NTP を使用して strata 2 サーバーを参照する NTP サーバーで

NTP may function as a **client**, **server** or a **peer**. **NTP** は、**クライアント**、**サーバー**、**ピア**として機能します。

- **クライアント**は、**サーバー**または **ピア**から時刻を取得する。
- **サーバー**は、**クライアント**に時刻を提供する。
- **ピア**は、指定された **サーバー**に関わらず他のピアとの間で時刻を同期する。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

NTP アプリケーション

NTP アプリケーションには色々ありますが、最も有名なものは

- **ntp** ,http://www.ntp.org, これがデフォルトです。
- **chrony**, http://chrony.tuxfamily.org, 一時的なネットワーク接続や仮想マシンを含めた幅広い環境での利用を想定しています。
- **systemd-timesyncd**, **systemd** パッケージに含まれている唯一の ntp クライアントです。

NTP サービスの実装は、ディストリビューションによって大きく異なります。設定の最も一般的な要素は、ほとんどの一般的な時間同期アプリケーションで類似しています。以下は、**ntp** アプリケーションの設定ファイルの一般的な場所です。

- **ntp**; `/etc/ntp.conf`
- **chrony**; `/etc/chrony.conf`
- **systemd-timesyncd**; `/etc/systemd/timesyncd.conf`

設定の詳細については、関連するマニュアルページを参照してください。

様々な ntp アプリケーションは互いに競合する可能性があるため、一度に 1 つの ntp アプリケーションだけをアクティブにしてください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ntpd クライアントの設定

- NTP ソースをサーバーまたはピアとして選択します。
- NTP デーモンを起動します。
- 時刻が同期していることを確認します。

優れた NTP サーバーは、優れた時間ソース以上のものではありません。NTP プールプロジェクト (http://www.pool.ntp.org/en/) は、数少ないの NTP サーバーに（接続が集中し）過剰になっていた負荷を軽減するために作られました。**pool** ディレクティブは、ラウンドロビン DNS サーバーを認識し、リクエストごとに異なる IP アドレスを許可します。接続が **ピア**を許可している場合、**pool** コマンドはそれ自体でピアを確立します。

必要であれば、以前の **server** ディレクティブを使用することができます。

NTP プールを使用するように NTP サーバーを設定するには、`/etc/ntp.conf` を編集して、次の設定を追加または編集します：

/etc/ntp.conf

```
driftfile /var/lib/ntp/ntp.drift

pool 0.pool.ntp.org
pool 1.pool.ntp.org
pool 2.pool.ntp.org
pool 3.pool.ntp.org
```

ntpdc -c peers コマンドは、ローカルシステムと設定されたタイムサーバー間の時間差を表示することができます。timedatectl コマンドは多くのディストリビューションにあり、システムの時刻と日付を照会し、制御するために使用されることがあります。**timedatectl** は **systemd** の一部であり、**systemd-timesyncd** と一緒に使用すると、追加のオプションをサポートします。

```
# timedatectl
```

```
               Local time: Sun 2023-03-05 16:13:50 UTC
           Universal time: Sun 2023-03-05 16:13:50 UTC
                 RTC time: Sun 2023-03-05 16:13:50
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ntpd サーバーの設定

ntp サーバー の設定要素は、file/etc/ntp.conf ファイルに記載されています。ここでは、**ntp サーバー** の関連項目を紹介します。

- 基準時間ソースとなるローカルマシンの宣言
  - **server** と **fudge** ディレクティブ
- **ntpq** と **ntpdc** コマンド (CVE-2013-5211 を参照) を使ってタイムサーバーに問い合わせができるマシンを制限する。
  - **restrict** ディレクティブ
- Declaration of which systems are **ntp ピア**となるシステムの宣言
  - **peers** は `/etc/ntp.conf` ファイル内に記述される
- NTP デーモンの起動

以下は `/etc/ntp.conf` 内のアクセス コントロールの記述の例

/etc/ntp.conf: access control

```
# Default policy prevents queries
restrict default nopeer nomodify notrap noquery
# Allow queries from a particular subnet
restrict 123.123.x.0 mask 255.255.255.0 nopeer nomodify notrap
# Allow queries from a particular host
restrict 131.243.1.42 nopeer nomodify notrap noquery
# Unrestrict localhost
restrict 127.0.0.1
```

ntp ピアの設定の記述例

/etc/ntp.conf

```
peer 128.100.49.12
peer 192.168.0.1
```

これは、**ntp** が自分自身を **NTP** サーバーとして宣言している例である。2 行目の **fudge** エントリで、このサーバーが stratum 10 サーバーであることを指定しています。

/etc/ntp.conf

```
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

27.8 演習

課題 27.1: stratum 3 NTP サーバーを設定し、有効にする。サーバーをクライアントとして NTP プールに接続する

解 27.1

1. お使いのインストーラーで NTP デーモンがインストールされていることを確認します。

```
# yum install ntp
# zypper install ntp
# apt install ntp
```

2. NTP サーバーが NTP プールに問い合わせるように設定し、匿名トラフィックを許可します。

`/etc/ntp.conf` を編集し、`server X.X.X.X` の行を以下に合わせます。match these:

/etc/ntp.conf

```
server 0.pool.ntp.org
server 1.pool.ntp.org
server 2.pool.ntp.org
server 3.pool.ntp.org
```

3. 適切なコマンドで **ntp** サーバーを再起動します。

```
# systemctl restart ntpd
# systemctl restart ntp
```

4. 設定した新しいタイムサーバーに問い合わせします。

```
$ ntpq -p
```

注目

タイムサーバーと同期して問い合わせ結果が何か帰ってくるまでには、何分か待たなければならないかもしれません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 28 章

ネットワーク デバイスと構成

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- ネットワークデバイスを特定し、オペレーティングシステムのデバイス命名法と特定の役割への紐付けを理解する。
- **ip** ユーティリティで、デバイス、ルーティング、ポリシーベースルーティング、トンネリングの表示と制御ができる。
- **PNIDN**（Predictable Network Interface Device Names）を理解する。
- `etc` ディレクトリにある主要なネットワーク設定ファイルを把握する。
- **ネットワークマネージャー**（**nmtui** および **nmcli**）を使用して、ディストリビューションに依存しない方法でネットワークインターフェイスを設定する。
- デフォルトルートと、その他のスタティックルートの設定方法を把握する。
- 仮想ネットワークインターフェイスを理解する。
- DNS と名前解決を理解する。
- 有効な検査ユーティリティを実行してネットワークの問題を解決する。

28.1 ネットワーク デバイス

ネットワーク デバイス

- ネットワーク名には、タイプ識別子の後ろに番号がつけられている
  - **Ethernet**: eth0、eth1、eno1、eno2 など
  - **Wireless**: wlan0、wlan1、wlp3s0、wlp3s2 など
  - **Bridged**: br0、br1 など
  - **Virtual**: vmnet1、vmnet8 など
- ひとつのデバイスが、複数の IP アドレスを持つことができる
  - 古いシステムでは eth0:4 → eth0 といった関連付けを行った
  - 最近は、**ifconfig** の代わりに **ip** を使って一つのインターフェイス名に複数のアドレスを関係付ける

ネットワーク デバイスは特別な **デバイス ファイル** とは関連付けられず、**デバイス ノード** として名前が付けられます。/dev ディレクトリのエントリを関連付けるのではなく、名前で認識されています。

以前は、複数の仮想デバイスを単一の物理デバイスに関連付けることができました。これらは、コロンと数字を使って命名されていました。具体的には eth0:0 は一番目の eth0 デバイスのいう意味のエイリアス（別名）です。これは 1 つのネットワークカードで、複数の IP アドレスをサポートするために行われました。ただし **ifconfig** の代わりに **ip** を使用する場合この命名法は利用できないので、これ以上は説明しません。さらに、このやりかたは **IPv6** とは互換性がありません。

古典的な方法によるデバイスの命名は、特に同じ種類のインターフェイスが複数存在する場合に問題を発生させます。たとえば、同じネットワーク カードが 2 枚あるとします。古典的な命名法では 1 つが eth0、もう 1 つは eth1 という名前になります。
では、それぞれの物理デバイスと名前の対応関係はどうなるでしょうか？

最も簡単な方法は、最初のデバイスを eth0 に、 2 番目を eth1 にすることです。ところが残念ながら、最新のシステムのデバイス
検出は確定的ではなく、デバイスが予測できない順序で配置されたり接続されたりする場合があるのです。そのため、ローカル インターフェイスとインターネット インターフェイスの順序が入れ替わってしまう可能性がでてきます。カーネルのバージョンと構成によって、ハードウェアが変更されていなくてもインターフェイスの配置順序が変わることがわかっています。

多くのシステム管理者は、システム構成ファイルと起動スクリプトの中でハードウェア（MAC）アドレスとデバイス名との関連付けをハードコーディングするという方法で、この問題を解決しています。この方法は長年にわたって使われてきましたが、手動調整が必要なだけでなく、MAC アドレスが決まっていない場合には関連付けを定義できません。MAC アドレスが決まっていないケースは、組み込みシステムと仮想化で発生する可能性があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.2 予測可能な ネットワーク インターフェイス デバイス名

予測可能な ネットワーク インターフェイス デバイス名

- **予測可能な ネットワーク インターフェイス デバイス名 (PNIDN)** (**P**redictable **N**etwork **I**nterface **D**evice **N**ames) は **udev** と連携し **systemd** と統合されている
- デバイスに指定できる名前は 5 種類ある
  1. **ファームウェア** や **BIOS** がオンボード デバイスに対して提供するインデックス番号を組み入れた名前　例：eno1
  2. **ファームウェア** や **BIOS** が提供している **PCI Express** ホットプラグ スロットのインデックス番号を組み込んだ名前　例：ens1
  3. ハードウェア接続の物理的か地理的、もしくは、その両方の場所を組み込んだ名前　例：enp2s0
  4. MAC アドレスを組み込んだ名前　例：enx7837d1ea46da
  5. 従来型のメソッドを使用した名前　例：eth0

かつては eth0 と eth1 と呼ばれていた 2 つのオンボード **PCI** ネットワーク インターフェイスをもったマシンの例を示します。

```
$ ip link show | grep enp
2: enp4s2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT qlen 1000
3: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000

$ ifconfig | grep enp
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
enp4s2: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
```

これらの名前は **PCI** システム上で、ボードがどこに接続されているかで決まります。

```
$ lscpi | grep Ethernet
02:00.0 Ethernet controller: Marvell Technology Group Ltd. 88E8056 PCI-E Gigabit Ethernet Controller (rev 12)
04:02.0 Ethernet controller: Marvell Technology Group Ltd. 88E8001 Gigabit Ethernet Controller (rev 14)
```

lspci 出力の各行の先頭の 3 組の数字は、バス番号、デバイス（またはスロット）番号、デバイス機能の番号です。これは、物理的な位置を反映したものです。同様に、従来 wlan0 という名前で呼ばれていた無線デバイスを調べます。

```
$ ip link show | grep wl
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORMANT qlen 1000

$ lspci | grep Centrino
03:00.0 Network controller: Intel Corporation Centrino Advanced-N 6205 [Taylor Peak] (rev 34)
```

同じパターンが見られます。新しい命名スキームをオフにすれば簡単に古典的な名前に戻すことができます。古典的な命名は研究プロジェクトとして残す予定です。これ以降は、明確かつ単純に説明するためにおもに古典的な名前を使っていきます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.3 ネットワーク設定ファイル

ネットワーク設定ファイル

- 各ディストリビューションには、独自のファイルやディレクトリがある
  - **Red Hat**
    `/etc/sysconfig/network`
    `/etc/sysconfig/network-scripts/ifcfg-ethX`
    `/etc/sysconfig/network-scripts/ifcfg-ethX:Y`
    `/etc/sysconfig/network-scripts/route-ethX`
  - **Debian** `/etc/network/interfaces`
  - **SUSE** `/etc/sysconfig/network`
- **systemd** が標準的になっている
  - **Network Manager** を利用することが推奨される
  - 最近のディストリビューションでは、これらの設定ファイルは存在しないか中身がないことがある

新しい **Linux** ディストリビューションでは、これらの設定ファイルは存在しないか、空か、もしくはずっと小さくなっています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ネットワーク マネージャー 設定ファイル

- **Network Manager** は、いくつかのディストリビューションから、従来のネットワークファイルを使用できるネットワークプロファイルは、**プラグイン** でサポートされている
- **key-value** 形式が望ましい
- 従来の構成と互換性を持たせるためのオプションの **プラグイン** がいくつか用意されている
  - **ifupdown** は `/etc/network/interfaces`
  - **ifcfg-rh** は `/etc/sysconfig/network-scripts`
  - **ifcfg-suse** は、SUSE と openSUSE の単純な互換性のため
  - **key-file** システム固有の設定ファイルを汎用的な設定に置き換える

追加情報については、以下を参照
https://wiki.gnome.org/Projects/NetworkManager/SystemSettings/

`/etc/NetworkManager/NetworkManager.conf`の **[main]** セクションに、設定処理のためのプラグインをカンマ区切りで使用する設定オプションがあります。
その他の情報については、こちらをご覧ください。
https://developer-old.gnome.org/NetworkManager/stable/NetworkManager.conf.html

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.4 ネットワーク マネージャー

ネットワーク マネージャー

- ネットワーク インターフェイスは、ソフトウェアもハードウェアも、基本的に（設定後には変更されない）スタティックなものだった
- 最新システムの多くは、ダイナミックなコンフィギュレーションに対応している:
  - デバイスの場所が変わると、ネットワークも変更されることがある
  - 無線デバイスは、より多くのネットワークに接続される可能性がある
  - ハードウェア（無線デバイスなど）は、接続されたりオン／オフされるとデバイス名が変わることがある
- これまで説明してきた設定ファイルは、スタティックな環境で使われることを想定したもので、ディストリビューション依存性も大きかった
- **ネットワーク マネージャー** はシステムが異なってもほぼ共通である

昔はネットワーク接続はほぼ全て有線（イーサネット）で、大規模なシステム変更が無い限り（デバイス名やアドレスが）変更されることはありませんでした。

システムが起動すると、システムは `/etc` ディレクトリ サブツリーのネットワーク構成ファイルを調べて、静的または動的（DCHP）アドレス構成やデバイスをブート時に開始すべきかどうかなどを決定しました。

複数のネットワーク デバイスが存在する場合には、どの順序で起動するか、どのネットワークに接続するか、呼び名をどうするかなどポリシーを決定する必要がありました。

ワイヤレス接続（および、**USB** アダプターなどのホットプラグ ネットワークデバイス）がより一般的になるにつれ、ハードウェアの一時的な性質と、接続されている特定のネットワークの性質の両方が原因で、設定ははるかに複雑になりました。

ディストリビューションに依存したインターフェイスやコンフィギュレーションの排除には、大きなアドバンテージがありました。

ネットワーク マネージャー には構成ファイルがありますが、それらを直接編集するのではなく通常は用意されているユーティリティ群を利用して操作や更新する方が良いでしょう。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ネットワーク マネージャー インターフェイス

- **GUI**
  全ての **Linux** デスクトップ (**GNOME**、**KDE**、**XFCE** 等) に備わる、通常はトップバーのネットワークアイコンをクリックして起動する
- **nmtui**
  network **m**anager **t**ext **u**ser **i**nterface は **ncurses** を利用したシンプルなメニュー駆動式のコマンドラインインターフェイスを持つ
- **nmcli**
  network **m**anager **c**ommand **l**ine **i**nterface は、抽象度の高いインターフェースから利用可能な低水準メソッドで、スクリプトに組み込む場合にも利用しやすい

ホテルの部屋やコーヒーショップでラップトップを使用している場合は、**Linux** ディストリビューションが提供する何らのグラフィカル インターフェイスを使用していると思います。これを使用すれば、異なるネットワークを選択したり、セキュリティとパスワードを構成したり、デバイスのオン／オフを切り替えたりできます。

しばらく使い続けることを前提として構成を設定変更する時には、直感的に設定ファイルの編集ができる **nmtui** を使うと良いです。

スクリプトを使ってネットワーク構成を変更する必要がある場合は **nmcli** を使用します。またコマンドライン操作の方が好きな人も **nmtui** の代わりにこの **nmcli** を使用することもできます。

GUI が適切に設定されていれば、これらの３つの方法のいずれかを使用してもタスクを実行できるはずです。しかし、ここではディストリビューション中立で、設定ファイルに起因する違いを意識する必要がない **nmtui** と **nmcli** に焦点を当てます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

nmtui

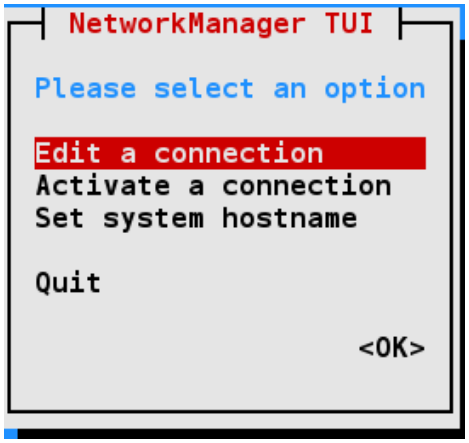


図 28.1: **nmtui メイン画面**

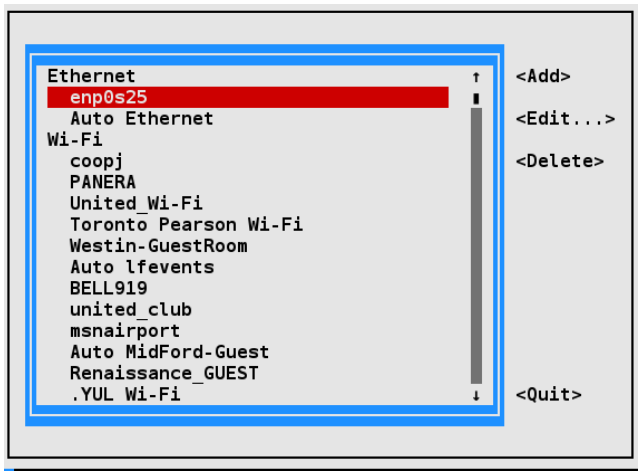


図 28.2: **nmtui 編集画面**

nmtui の利用はより直感的です。矢印キーや TAB キーを使ってメニューを操作します。

ネットワーク接続の編集や有効化以外に、システムのホスト名を設定することもできます。しかしこのような操作は、通常のユーザー権限では行えないので root パスワードの入力を求められます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

nmtui ワイヤレス接続設定

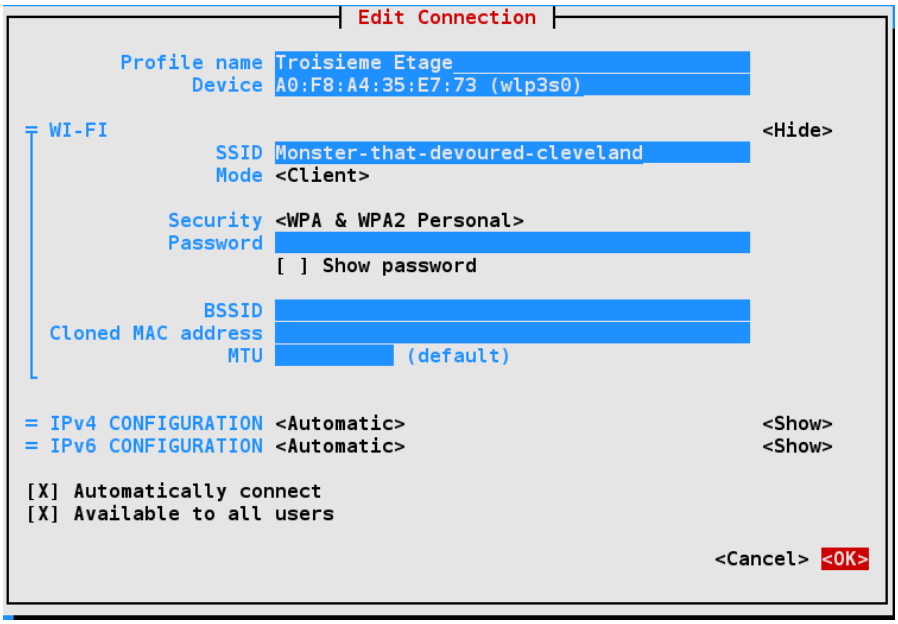


図 28.3: **nmtui ワイヤレス接続設定**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

nmcli

- **ネットワーク マネージャー**に対するコマンドライン インターフェイスである（**n**etwork **m**anager **c**ommand **l**ine **i**nterface）
- 対話モードも用意されている
- `man nmcli-examples` も参照
- 演習で、いくつかの事例を紹介する

更に知りたい？

以下を参照
https://fedoraproject.org/wiki/Networking/CLI

```
$ nmcli --help
```

```
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }
OPTIONS
  -a, --ask                                ask for missing parameters
  -c, --colors auto|yes|no                 whether to use colors in output
  -e, --escape yes|no                      escape columns separators in values
  -f, --fields <field,...>|all|common      specify fields to output
  -g, --get-values <field,...>|all|common  shortcut for -m tabular -t -f
  -h, --help                               print this help
  -m, --mode tabular|multiline             output mode
  -o, --overview                           overview mode
  -p, --pretty                             pretty output
  -s, --show-secrets                       allow displaying passwords
  -t, --terse                              terse output
  -v, --version                            show program version
  -w, --wait <seconds>                     set timeout waiting for finishing operations

OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager
  a[gent]         NetworkManager secret agent or polkit agent
  m[onitor]       monitor NetworkManager changes
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

28.5 ルーティング

ルーティング

- ネットワーク内、およびネットワーク間のパケットの経路を設定する
- ルーティングテーブル
  - 全てのネットワークとホストへのパスを定義する
  - （同一ネットワーク内の）ローカルパケットは直接送信する
  - リモート ネットワーク向けのパケットはルーターに送る

ルーティング は、ネットワーク トラフィックを送信するネットワーク内のルート（パス）を選択するプロセスです。**ルーティングテーブル**は、システムが管理する他のネットワークへのルートのリストです。全てのネットワークとホストへのルートを定義し、リモート トラフィックをルーターに送信します。

現在のルーティング テーブルを表示するには **route**（非推奨）、または **ip** を使用します。

```
student@ubuntu: /etc
student@ubuntu:/etc$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 ens33
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 ens33
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 ens33
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
student@ubuntu:/etc$ ip route
default via 192.168.1.1 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
192.168.1.0/24 dev ens33  proto kernel  scope link  src 192.168.1.24  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown

student@ubuntu:/etc$
```

図 28.4: **route と ip route の利用**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

デフォルト ルート

- ルーティング テーブル エントリーにマッチしない時に使用される
- DHCP を使って動的に取得できる
- 手動で設定することもできる（スタティック設定）
  - **Red Hat** 系のシステム: `GATEWAY=x.x.x.x`
    `/etc/sysconfig/network` または
    `/etc/sysconfig/network-scripts/ifcfg-ethX` に含まれる
  - **Debian** 系のシステム: `gateway=x.x.x.x`
    `/etc/network/interfaces` に含まれる
- `/etc` 内のファイルの手による編集を避けるため **nmcli** を利用する

```
$ sudo  nmcli con mod virbr0 ipv4.routes 192.168.10.0/24 \
  +ipv4.gateway 192.168.122.0
$ sudo nmcli con up virbr0
```

デフォルト ルート は、ネットワークのパスがルーティング テーブルに明示的に指定されていない場合の、パケットの送信先です。デフォルト ゲートウェイは、システム稼働中にも設定が可能です。

```
$ sudo ip route add default via 192.168.1.10 dev enp2s0
$ ip route
```

```
default via 192.168.1.10 dev enp2s0
default via 192.168.1.1 dev enp2s0 metric 1024
192.168.1.0/24 dev enp2s0 proto kernel scope link src 192.168.1.100
```

この変更により、ネットワーク接続が切れる可能性があるので注意してください！　ネットワークをリセットするか、上記の例で次のように変えれば復元できます。

```
$ sudo ip route add default via 192.168.1.1 dev enp2s0
```

この変更は恒久的なものではなく、システムを再起動した時には消失します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

スタティック ルート

- ルーター、又はルートが複数ある場合にパケットのフローを制御する
- インターフェイス毎に設定する
- 恒久的な設定も、一時的な設定もできる
- **ip** を利用する (一時的な設定となる)
- 恒久的な設定にするには
  - **Red Hat** 系のシステム:
    `/etc/sysconfig/network-scripts/route-ethX`
  - **Debian** 系のシステム:
    `/etc/network/interfaces`
  - **SUSE** 系のシステム:
    `/etc/sysconfig/network/ifroute-eth0`
- `/etc` 内のファイルの手修正を避けるため **nmcli** を使う (演習で試す)

システムに複数のルート、ないしルーターがある場合や複数のインターフェイスがある場合には、パケットの行き先に応じて選択的にルートをコントロールできると便利です。

一時的なルートの設定は **route** または **ip** コマンドを使って設定することができます。

```
$ sudo ip route add 10.5.0.0/16 via 192.168.1.100
```

Red Hat 系のシステムでは `/etc/sysconfig/network-scripts/route-ethX` を以下のように編集することで、恒久的なルートの設定が可能です。

```
$ cat /etc/sysconfig/network-scripts/route-eth0
  10.5.0.0/16 via 172.17.9.1
```

Debian 系のシステムでは以下のように `/etc/network/interfaces` に行を追加する必要があります。

```
iface eth1 inet dhcp
  post-up ip route add 10.1.2.51 dev eth1
  post-up ip route add 10.1.2.52 dev eth1
```

SUSE 系のシステムでは以下のように `/etc/sysconfig/network/ifroute-eth0` に以下の行を追加、ないし作成する必要があります。

```
# Destination Gateway Netmask Interface [Type] [Options]
  192.168.1.150 192.168.1.1 255.255.255.255 eth0
  10.1.1.150 192.168.233.1.1 eth0
  10.1.1.0/24 192.168.1.1 - eth0
```

各フィールドは TAB で区切ります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.6 仮想ネットワーク インターフェイス

Teaming と Bonding ネットワーク インターフェイス

- Teaming と Bonding インターフェイスを使うと、動的なネットワークインターフェイス構成が可能となり、ネットワークに高いスループットと更なる堅牢性が提供される
  - **Bonding**
    カーネルモジュールを使用して、ネットワークリンクのアグリゲーションを行い、フェイルオーバー、冗長性、が確保されパフォーマンスが向上する
  - **Teaming**
    teamed というユーザー空間サービスを使用する以外は **bonding** と同じ

Teaming は非推奨となっています

- **Teaming** 設定は **非推奨**としてリストアップされており、将来的はに削除予定です。
- **Teaming** から **Bonding** への変換をスムーズに行うために、設定ファイルを変換する **team2bond** ユーティリティが用意されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

Bonding ネットワーク インターフェイス

Bonding インターフェイスの設定にはいくつかのやり方がある

- **sysfs**
  直接 `/sys` 仮想ファイルを変更する。変更内容は保存されない
- **iproute2**
  `ip` コマンドを使って **bonding** を設定できる。詳細は `man 8 ip-link` を参照。変更内容は保存されない
- **ネットワーク マネージャー**
  全ての **ネットワーク マネージャー**ユーザーインターフェイスが **bonding** をサポートする。変更内容は保存される
  ネットワーク マネージャーの設定ファイルを、お好みのエディターで直接編集することも可能である

更に知りたい？

bonding の `sys` インターフェイスの詳細については、https://www.kernel.org/doc/Documentation/networking/bonding.txt を参照してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

nmcli を使った Bonding ネットワーク インターフェイスの設定

nmcli を使って何ステップかの操作で **bonding** インターフェイス設定を作る。最小限の手順は

- **アダプターの指定**: `nmcli device status` を利用する
- **bonding デバイスの作成**: `nmcli connection add` を利用する
- **インターフェイスを bond に接続**: `nmcli connection add` コマンドを使って、アダプターから **bond** デバイスに対する接続を作る
- **アダプターをオンラインに設定**: **bond adapter** に対して `nmcli connection up` を発行する
- **再起動 (リブート)**: 再起動は必須ではないが、再起動によっていかなる未解決の設定の混乱も解消されるので再起動が強く推奨される

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

nmcli を使った bonding 設定例

コマンドライン上で

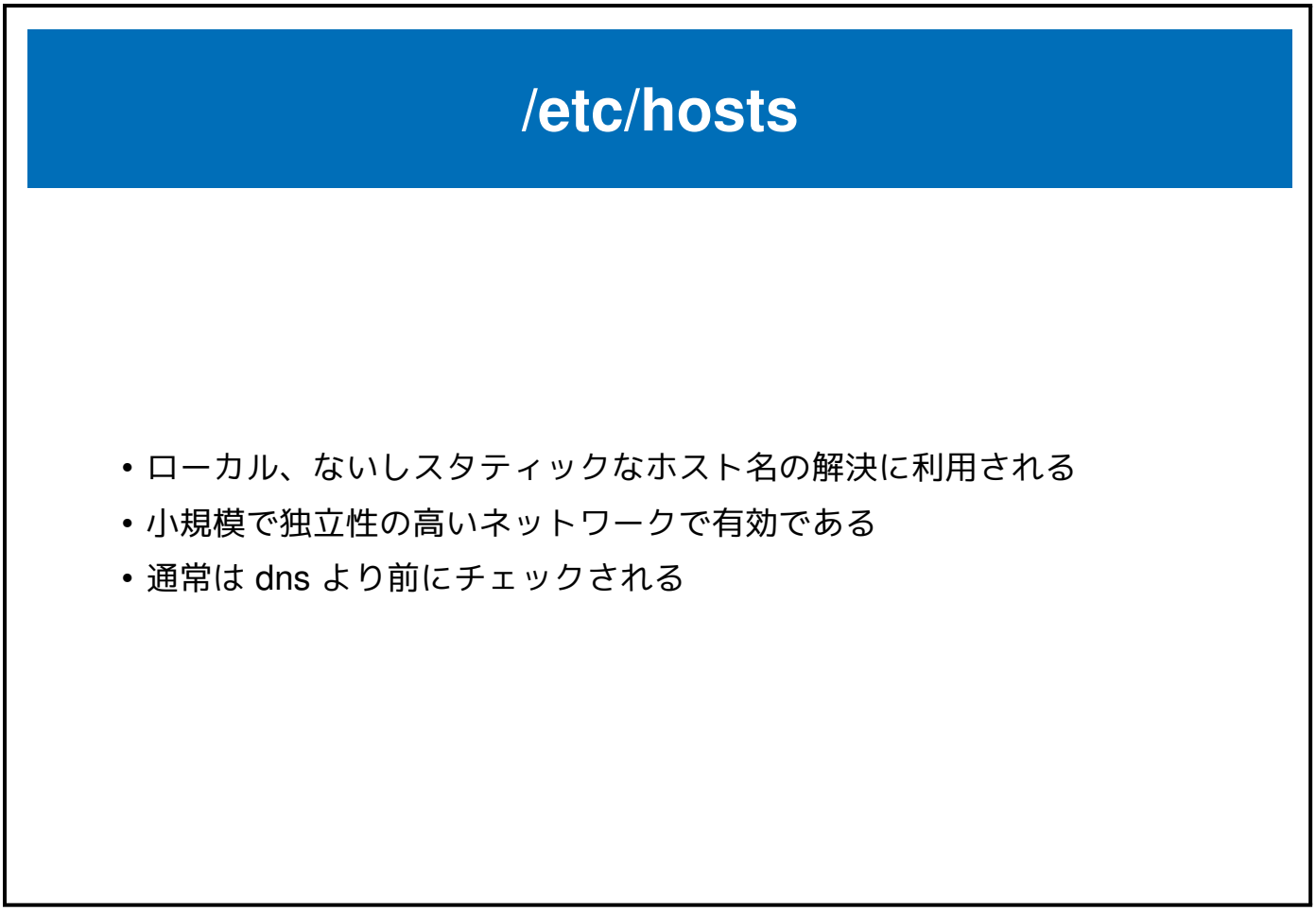
```
nmcli connection add type bond \
     con-name bond0 \
     ifname bond0 \
     bond.options "mode=active-backup,miimon=1000"
nmcli connection add type ethernet slave-type bond \
     con-name bond0-port1 \
     ifname enp0s20u1u1 \
     master bond0
nmcli connection add type ethernet slave-type bond \
     con-name bond0-port2 \
     ifname enp0s20u1u2 \
     master bond0
nmcli connection up bond0
nmcli device status
reboot
nmcli device status
```

更に知りたい？

nmcli に関する詳細は https://wiki.gnome.org/Projects/NetworkManager/SystemSettings/ を参照してください。
bonding と bonding オプションに関する詳細は https://www.kernel.org/doc/Documentation/networking/bonding.txt を参照してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.7 DNS と名前解決



/etc/hosts は、ホスト名と IP アドレスのローカル データベースを保持します。これには、IP アドレスを対応するホスト名と別名（エイリアス名）にマッピングするレコードのセット（それぞれ 1 行ずつ）が含まれています。

典型的な /etc/hosts ファイルは、次のようになります。

/etc/hosts

```
127.0.0.1      localhost localhost.localdomain localhost4 localhost4.localdomain4
::1            localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.100  hans hans7 hans64
192.168.1.150  bethe bethe7 bethe64
192.168.1.2    hp-printer
192.168.1.10   test32 test64 oldpc
```

このような静的な名前解決は、主にローカルで小規模で分離されたネットワークで使用されます。通常 **DNS** によるアドレス解決の前に参照されます。参照の優先順位は /etc/nsswitch.conf で制御できます。（これは、最近ではあまり使われていません）

/etc 内の他のホスト関連ファイルには、以下のものがあります。
/etc/hosts.deny、/etc/hosts.allow.

これらは自己文書化されており、その目的は名前を見れば分かります。allow ファイルが最初に検索され、クエリがそこに見つからない場合にだけ deny ファイルが検索されます。

/etc/host.conf には、一般的な構成情報が含まれています。ただし、これは殆ど使用されません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

DNS と名前解決

- 名前解決: ホスト名を IP アドレスに変換すること
  - スタティックな名前解決
  - ダイナミックな名前解決 (**DNS**)
- 逆引き: IP アドレスからホスト名に変換すること
- コマンドラインから:

```
$ dig linuxfoundation.org
$ host linuxfoundation.org
```

名前解決 とは、ホスト名をホストの IP アドレスに変換することです。
たとえば、ブラウザまたは電子メール クライアントは `training.linuxfoundation.org` との間で送受信を行うために、`training.linuxfoundation.org` にサービスを提供しているサーバー群の名前を解決して IP アドレスに変換します。

名前解決は `/etc/hosts` を利用して **スタティック** に行うことも、**DNS** サーバーを利用してダイナミックに行うこともできます。
ホスト名の IP アドレス解決に使用するコマンドライン ツールとして:

- **dig**: 情報を最も詳細に取得できるツールで、多くのオプションがあります。
- **host**: 簡潔に情報を取得できます。
- **nslookup**: 昔からあるツールですが推奨しません。特に `DNSSEC` を使っている時には、誤った答えを出すことがあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

dig の利用

- 利用可能なツールの中で、最も正確な答えを提供する
- デフォルトの出力は、単純なタスクやスクリプトには冗長なことが多い
- 名前解決の全体像を見ることができる
- DNSSEC の動作の表示と検証ができる
- 有効なオプションの例

  +trace （デフォルトでは）ルートサーバーからのパス全体をチェックする

  -t [レコード タイプ] レコードを指定する（A, TXT, CNAME, MX などを指定できる）

以下に、**dig** によるアドレス参照のデフォルトの挙動の例を示します。

```
$ dig linuxfoundation.org
```

```
; <<>> DiG 9.16.27-Debian <<>> linuxfoundation.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54765
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;linuxfoundation.org.^^I^^IIN^^IA

;; ANSWER SECTION:
linuxfoundation.org.^^I524^^IIN^^IA^^I3.13.31.214

;; Query time: 3 msec
;; SERVER: 10.139.1.1#53(10.139.1.1)
;; WHEN: Sun May 07 13:40:00 EDT 2023
;; MSG SIZE  rcvd: 64
```

この例では、（IPv4）の A レコード、フラグ、トランスポート、タイミング情報が表示されています。

Let's try with reverse resolution and make it much less verbose: 逆引きもやってみましょう。こちらは表示内容が少ないです。

```
$ dig -x 1.1.1.1 +short
```

```
one.one.one.one.
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

DNS

- 動的な名前解決ができる
- /etc/resolv.conf
  - 検索対象とするドメインを指定できる
  - 利用するネームサーバーの厳密な参照順序を指定する
  - 手動で構成を設定することも、**DHCP** (**D**ynamic **H**ost **C**onfiguration **P**rotocol) などのサービスから更新させることもできる

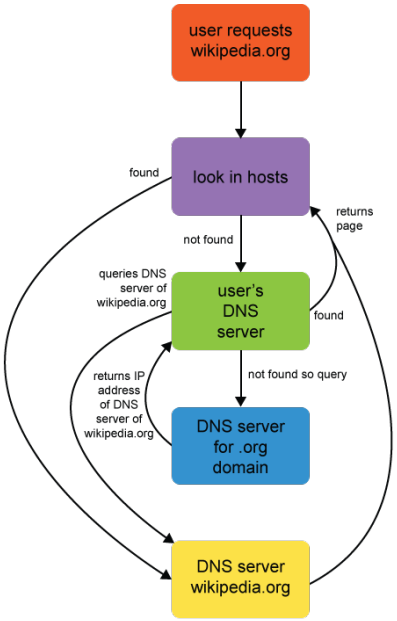


図 28.5: **DNS**

/etc/hosts を利用したローカルな名前解決が成功しない時、システムは **DNS** (**D**omain **N**ame **S**erver) に問い合わせます。

DNS は、動的に名前解決を行いクライアントが名前を検索するために使用するサーバーのネットワークを構成します。サービスは分散されます。各々の **DNS** サーバーは、そのサーバーの **権限ゾーン (zone of authority)** に帰属する情報だけを持っていますが、複数のサーバーが連携することでどんな名前でも解決できます。

DNS は /etc/resolv.conf で設定されます。これは伝統的に以下のようになっていました。

/etc/resolv.conf

```
search example.com aps.org
nameserver 192.168.1.1
nameserver 8.8.8.8
```

最近のほとんどのシステムでは、次のように /etc/resolv.conf ファイルは自動生成されます。

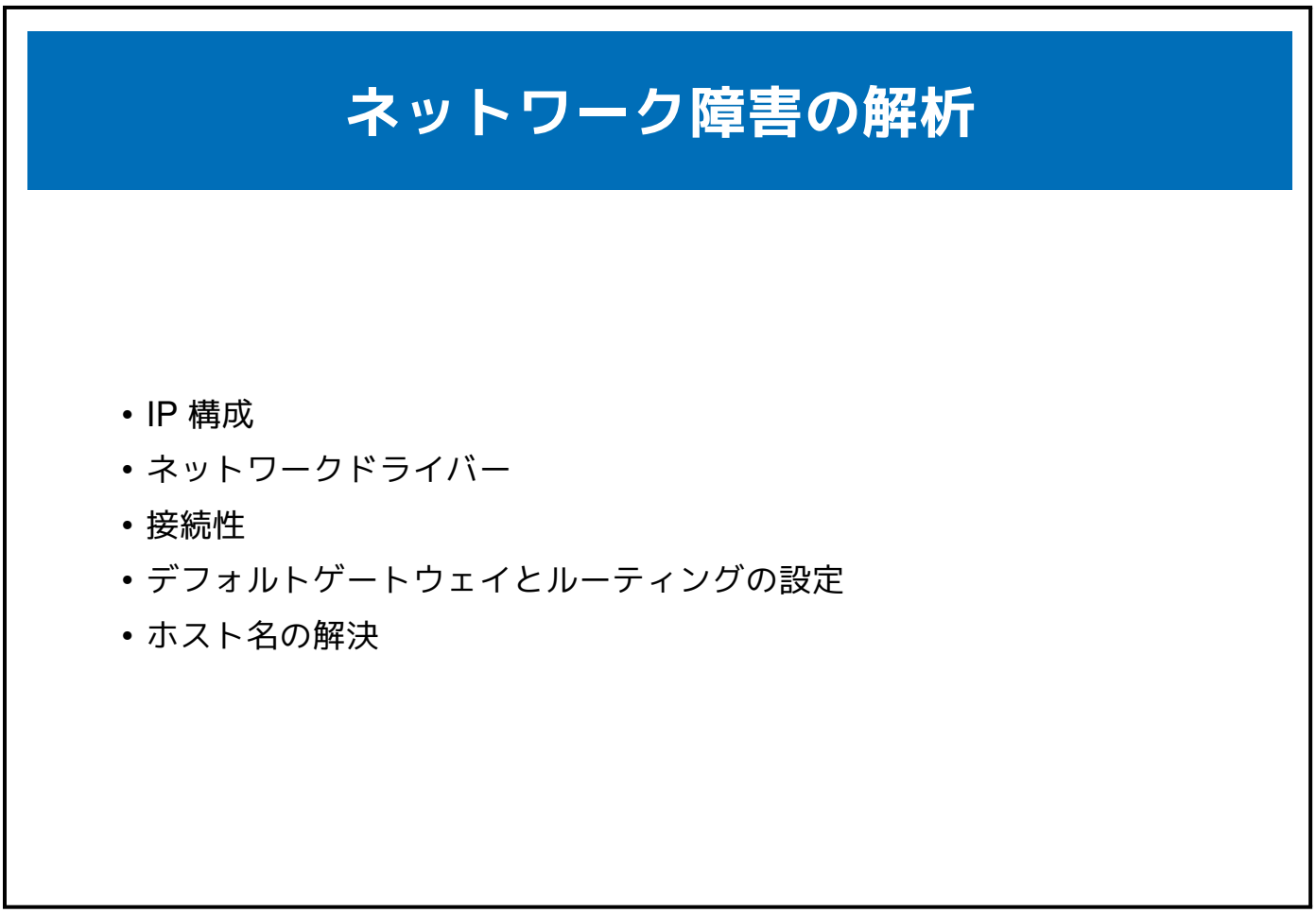
```
# Generated by NetworkManager
```

```
    192.168.1.1
```

これは、プライマリ ネットワーク インターフェイスで **DHCP** を呼び出している **NetworkManager** が生成しています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

28.8 ネットワークに関する障害解析



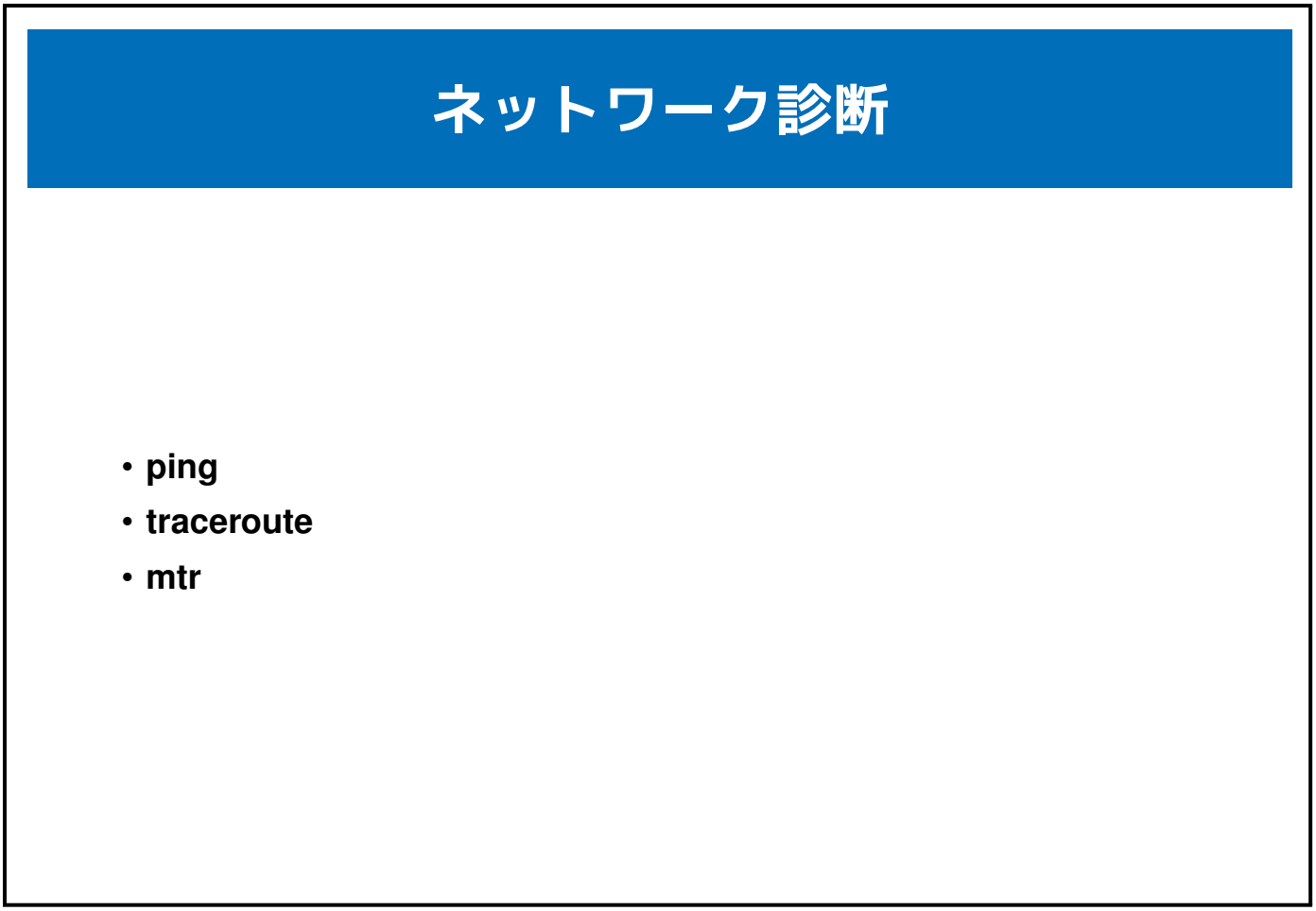
ネットワークの問題は、ソフトウェア起因またはハードウェア起因のいずれの可能性もあり、デバイスドライバの読み込みや、ネットワークのケーブル接続など、簡単な原因もありえます。ネットワークがアップ（＝活性化）しているけれどもパフォーマンスが悪い場合は、トラブルシューティングではなくパフォーマンス チューニングの対象になります。問題の原因がマシン以外に起因する場合や、バッファサイズなどのさまざまなネットワーク パラメータの調整が必要な場合があります。

ネットワークに問題がある時には、以下の項目を確認する必要があります。

- **IP 構成:**
  ifconfig または **ip** を使用して、インターフェイスがアップ（活性化）しているか、アップしているなら設定されているかを確認します。
- **ネットワークドライバー:**
  インターフェイスを起動できない時には、ネットワークカードに正しいデバイスドライバがロードされていない可能性があります。**lsmod** でネットワークドライバがカーネルモジュールとしてロードされているかを確認します。もしくは `/proc` や `/sys` など、関連する疑似ファイルで `/proc/interrupts` や `/sys/class/net` などの値を確認します。
- **接続性:**
  ping を使用して対象のネットワークに到達可能かどうかを確認し、応答時間とパケット損失があるかを確認します。**traceroute** を使うとネットワーク内でパケットが通過した経路を追跡できますが、**mtr** を使えばこれを連続的に実施できます。これらのユーティリティを使用すると問題がローカル側にあるのかインターネット側にあるのかの切り分けが可能です。
- **デフォルト ゲートウェイとルーティング構成:**
  route -n を実行してルーティングテーブルが適切か確認します。
- **ホスト名の解決:**
  対象 URL に対して **dig** または **host** を実行し、**DNS** が正しく機能しているかどうかを確認します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.9 ネットワーク診断



全てのシステム管理者のツール ボックスには、多くの基本的なネットワーク ユーティリティがあります。具体的には

ping:
64 バイトのテストパケットを指定されたネットワークホストに送信し、(見つかった場合) 到達に必要だった時間 (ミリ秒単位)、失われたパケット、およびその他のパラメーターについての情報を報告します。正確な出力は対象となるホストによって異なりますが、少なくともネットワークが機能しておりホストまで到達できていることがわかります。

traceroute:
このユーティリティは、宛先へのネットワーク経路を調べるために使用されます。ホストに到達するために通過したルーターパケットフロー (経路) と、各 **ホップ** への到達にかかった時間を示します。

mtr:
ping と **traceroute** の機能を組み合わせて、**top** のように継続的に更新する表示を作成します。

RHEL などいくつかの **Linux** ディストリビューションでは、最初の 3 つの診断ユーティリティを実行するために root 権限 (**sudo** でもよい) を要求します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ping の例

```
[student@fedora ~]$ ping -c 10 linuxfoundation.org
PING linuxfoundation.org (23.185.0.4) 56(84) bytes of data.
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=1 ttl=128 time=21.3 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=2 ttl=128 time=20.2 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=3 ttl=128 time=19.2 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=4 ttl=128 time=20.7 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=5 ttl=128 time=19.8 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=6 ttl=128 time=19.6 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=7 ttl=128 time=19.5 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=8 ttl=128 time=20.3 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=9 ttl=128 time=21.0 ms
64 bytes from 23.185.0.4 (23.185.0.4): icmp_seq=10 ttl=128 time=20.9 ms

--- linuxfoundation.org ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 19.247/20.259/21.282/0.670 ms
[student@fedora ~]$
```

図 28.6: **ping**

$ **ping**

パケットの到達時間を見ます。

-c **10**

合計 **10** パケットについて確認します。

linuxfoundation.org

設定されたリゾルバーから **LINUXFOUNDATION.ORG** に返された IP アドレス（複数ある場合はランダム）を表示します。

最後に結果のサマリーを表示します。（デフォルトの動作）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

traceroute の例

```
c8:/tmp>traceroute google.com
traceroute to google.com (172.217.9.78), 30 hops max, 60 byte packets
 1  router (192.168.1.1)  0.340 ms  0.425 ms  0.544 ms
 2  * * *
 3  096-034-031-180.biz.spectrum.com (96.34.31.180)  19.900 ms  19.553 ms  18.011 ms
 4  acr06ftbgwi-gbe-8-46.ftbg.wi.charter.com (96.34.30.244)  19.682 ms  20.144 ms  19.848 ms
 5  cts03alxnmn-tge-3-0-0.alxn.mn.charter.com (96.34.25.137)  26.237 ms  26.465 ms  26.214 ms
 6  crr01rochmn-bue-2.roch.mn.charter.com (96.34.25.166)  26.445 ms  23.845 ms  25.653 ms
 7  crr01stcdmn-bue-6.stcd.mn.charter.com (96.34.25.0)  27.146 ms  21.003 ms  28.613 ms
 8  bbr02slidla-tge-0-1-0-6.slid.la.charter.com (96.34.1.188)  35.569 ms  33.988 ms  31.121 ms
 9  bbr02chcgil-bue-1.chcg.il.charter.com (96.34.1.149)  46.788 ms  46.397 ms  39.472 ms
10  prr01chcgil-bue-4.chcg.il.charter.com (96.34.3.11)  44.474 ms  44.081 ms  43.731 ms
11  096-034-152-159.biz.spectrum.com (96.34.152.159)  42.068 ms 096-034-152-155.biz.spectrum.co
m (96.34.152.155)  42.878 ms 096-034-152-159.biz.spectrum.com (96.34.152.159)  44.781 ms
12  108.170.243.193 (108.170.243.193)  40.572 ms 108.170.243.174 (108.170.243.174)  39.112 ms 1
08.170.243.193 (108.170.243.193)  42.198 ms
13  72.14.239.123 (72.14.239.123)  42.209 ms  39.873 ms 72.14.232.192 (72.14.232.192)  45.645 m
s
14  72.14.239.123 (72.14.239.123)  43.850 ms 108.170.244.2 (108.170.244.2)  43.267 ms 108.170.2
44.15 (108.170.244.15)  42.362 ms
15  ord38s09-in-f14.1e100.net (172.217.9.78)  42.122 ms  41.888 ms  40.801 ms
c8:/tmp>
```

図 28.7: **traceroute**

$ **traceroute**

ネットワークパスを検索します。（デフォルトでは UDP ポートを利用します）

google.com

設定されたリゾルバーが **GOOGLE.COM** に返した IP アドレス（複数ある場合はランダム）を表示します。

隠れた複雑性

DNS とインターネットルーティングは、あなたがネットワーク上のどこからアクセスするかに依存するので **google.com** のホスト名検索を行った時に返される IP アドレスが異なる場合があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

mtr の例

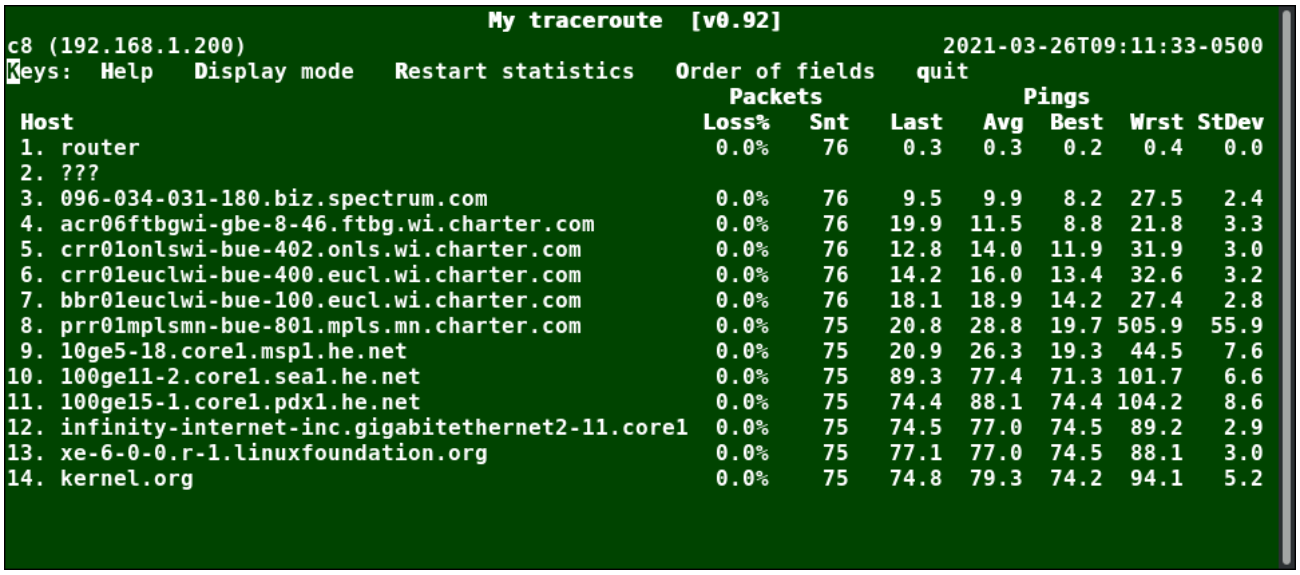


図 28.8: **mtr**

$ **mtr**

ネットワークの経路を検索します。(UDP ポートか ICMP を使用します)

-t

GUI ではなく、ターミナルを利用します。

kernel.org

設定されたリゾルバーが **KERNEL.ORG** に返した IP アドレスを（複数ある場合はランダムに）表示します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

28.10 演習

課題 28.1: 静的にホスト名を追加する

本演習では、ローカルのホストデータベースにエントリを追加します。

1. /etc/hosts を開き mysystem.mydomain を追加します。その際、ネットワークカードに割り当てられた IP アドレスを指すようにします。
2. 次に ad.doubleclick.net への参照が、127.0.0.1 を示すようなエントリを追加します。
3. 追加の演習として http://winhelp2002.mvps.org/hosts2.htm または http://winhelp2002.mvps.org/hosts.txt から、host ファイルをダウンロードしインストールします。新旧の host ファイルで、ブラウザの動作に違いがありましたか？

解 28.1

1. ```
   $ sudo sh -c "echo 192.168.1.180    mysystem.mydomain >> /etc/hosts"
   $ ping mysystem.mydomain
   ```

2. ```
   $ sudo sh -c "echo 127.0.0.1       ad.doubleclick.net >> /etc/hosts"
   $ ping ad.doubleclick.net
   ```

3. ```
   $ wget http://winhelp2002.mvps.org/hosts.txt
   ```

   ```
   --2014-11-01 08:57:12--  http://winhelp2002.mvps.org/hosts.txt
   Resolving winhelp2002.mvps.org (winhelp2002.mvps.org)... 216.155.126.40
   Connecting to winhelp2002.mvps.org (winhelp2002.mvps.org)|216.155.126.40|:80... connected.
   HTTP request sent, awaiting response... 200 OK
   Length: 514744 (503K) [text/plain]
   Saving to: hosts.txt

   100%[======================================>] 514,744      977KB/s   in 0.5s

   2014-11-01 08:57:13 (977 KB/s) - hosts.txt saved [514744/514744]
   ```

   ```
   $ sudo sh -c "cat hosts.txt >> /etc/hosts"
   ```

課題 28.2: dig を使って DNS 情報を確認する

この演習では、**dig** を使って IPv4 A レコードのフォワード（＝順引き）が、メールサーバーの PTR レコードのリバース（＝逆引き）と一致することを検証します。

1. bind がインストールされていることを確認します。もしされていなければ

   On Debian, Ubuntu, or Linux Mint

   apt install bind9-dnsutils

   On Red Hat, Centos, or Fedora

   dnf install bind-utils

2. **protonmail.com** のメールサーバーの IP アドレスを見つけます。
3. IP アドレスの逆引きが一致することを確認します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
   ```

解 28.2

1. `$ dig +short -t mx protonmail.com`

```
10 mailsec.protonmail.ch.
5 mail.protonmail.ch.
```

`$ dig +short -t A mail.protonmail.ch`

```
185.70.42.128
185.205.70.128
176.119.200.128
```

2. `$ dig +short -x 185.205.70.128`

```
mail.protonmail.ch.
```

課題 28.3: nmcli を使い、ネットワークインターフェイスに別名/アドレスを付与する

システムに恒久的に別 IPv4 アドレスを追加します。`/dev` を直接編集するのではなく **nmcli** を利用します。

1. まず、インターネットアドレスとインターフェイス名を取得します。

`$ sudo nmcli con`

```
NAME           UUID                                  TYPE            DEVICE
Auto Ethernet  1c46bf37-2e4c-460d-8b20-421540f7d0e2  802-3-ethernet  ens33
virbr0         a84a332f-38e3-445a-a377-4363a8eb963f  bridge          virbr0
```

コネクション名称は `Auto Ethernet` であるとわかります。

`$ sudo nmcli con show "Auto Ethernet" | grep IP4.ADDRESS`

```
IP4.ADDRESS[1]:                         172.16.2.135/24
```

アドレスは 172.16.2.135 となっています。このコマンドはコネクションに関する全ての情報を表示し、以下のように `NAME` の代わりに `UUID` を指定することもできることもできます。

`$ nmcli con show 1c46bf37-2e4c-460d-8b20-421540f7d0e2`

2. あなたのマシンに割り振られる新しいアドレスを追加します。

`$ sudo nmcli con modify "Auto Ethernet" +ipv4.addresses 172.16.2.140/24`

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

3. インタフェースを活性化し、存在を確認します。

```
$ sudo nmcli con up "Auto Ethernet"
```

```
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/6)

$ ping -c 3 172.16.2.140
PING 172.16.2.140 (172.16.2.140) 56(84) bytes of data.
64 bytes from 172.16.2.140: icmp_seq=1 ttl=64 time=0.038 ms
64 bytes from 172.16.2.140: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 172.16.2.140: icmp_seq=3 ttl=64 time=0.032 ms

--- 172.16.2.140 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.032/0.034/0.038/0.007 ms
```

4. 別名を削除してクリーンアップします。

```
$ sudo nmcli con modify  "Auto Ethernet" -ipv4.addresses 172.16.2.140/24
...
$ sudo nmcli con up "Auto Ethernet"
...
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 28.4: nmcli を使い、静的ルートを追加する

システムに、恒久的に静的 IPv4 ルートアドレスを追加します。/dev を直接編集するのではなく **nmcli** を利用します。

1. **ip route** を使って現在の経路表（ルーティングテーブル）を確認します。

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
```

2. **nmcli** を使って新しいルートを追加します。

```
$ sudo nmcli conn mod "Auto Ethernet" +ipv4.routes "192.168.100.0/24 172.16.2.1"
```

3. まだ、有効でないことに注意してください。

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
```

4. インターフェイスをリロードして有効化してください。結果を見ます。

```
$ sudo nmcli conn up "Auto Ethernet"
```

```
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/25)
```

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
192.168.100.0/24 via 172.16.2.1 dev ens33
```

5. リブートしてルートが有効であることを確認してください。（つまり **恒久的** になっているということです）
確認したら削除します。

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
192.168.100.0/24 via 172.16.2.1 dev ens33
```

```
$ sudo nmcli conn mod "Auto Ethernet" -ipv4.routes "192.168.100.0/24 172.16.2.1"
```

```
$ sudo nmcli conn up "Auto Ethernet"
```

```
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/3)
$ ip route
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
```

6. 以下のように、コマンドラインから **ip** を使いルートを設定できますが、これはリブート後には無効になります。

```
$ sudo ip route add 192.168.100.0/24 via 172.16.2.1
....
```

この方法で設定したルートが恒久的でないことを確認できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 29 章

LDAP **

リソース 要件

この演習では、追加の仮想マシンをインストールする必要があります。これは、クライアント構成をテストするために使用される小型の **LDAP サーバー** です。**LDAP** アプライアンスは、約 512MB のメモリーと 2GB のディスクスペースを使用します。この演習は、授業外で行ってください。

学習目標

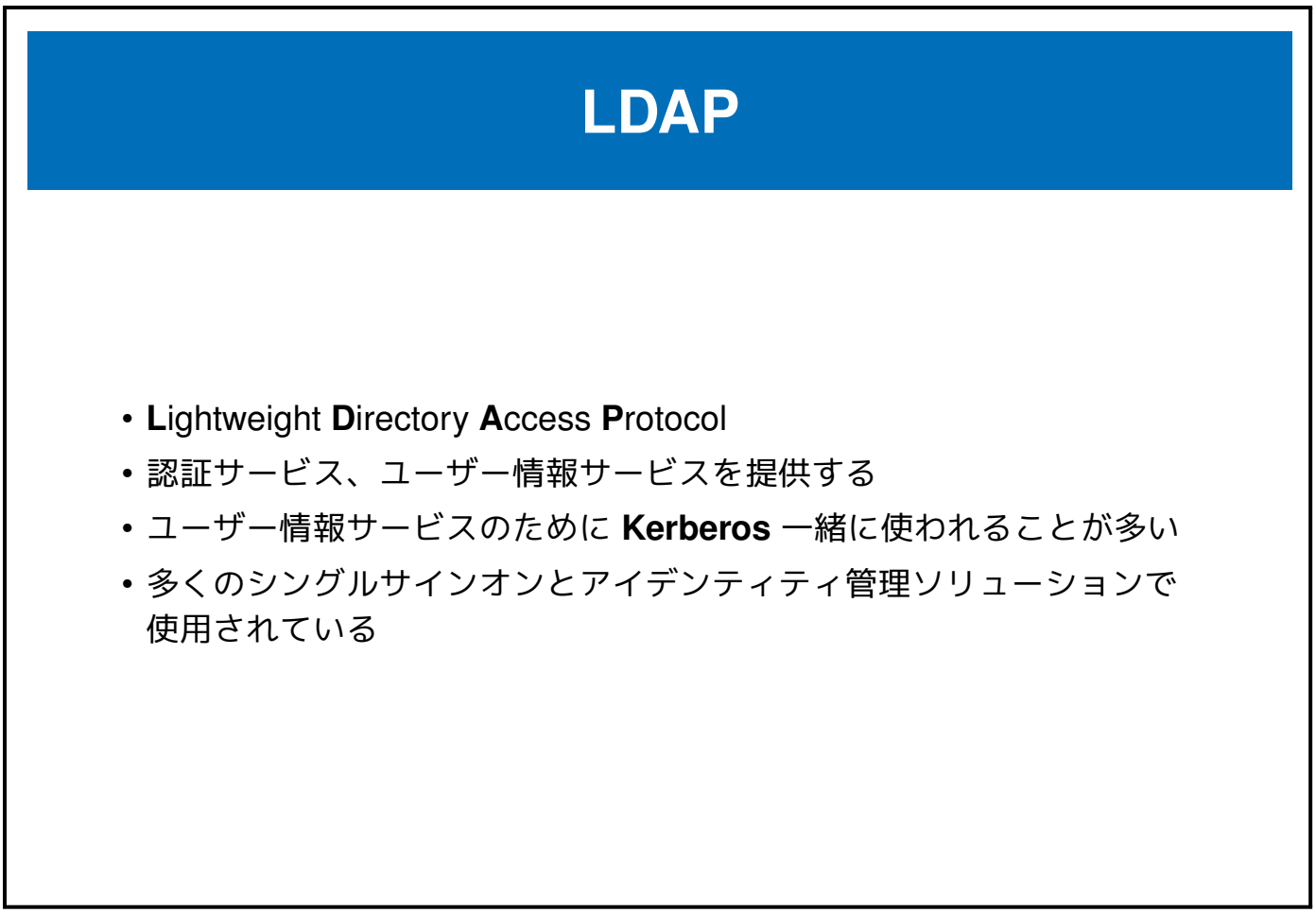
このセッションが終わるころには、次のことができるようになっているはずです

- **LDAP** 認証の目的を理解する。
- **LDAP** に関わる `/etc` の重要なファイルを把握する。
- これらのファイルを操作するために利用できるユーティリティを把握する。
- **LDAP** サーバーアプライアンスを使用した演習に取り組む。

注目

** これらのセクションの一部または全体をオプション扱いとする場合があります。これらには補足資料、専門トピック、または高度な話題が含まれインストラクターが教室の状況や時間制約に応じてこれらの内容を紹介するかを判断します。

29.1 LDAP 認証



LDAP は、IP プロトコルでディレクトリサービス（電話帳のようなもの）を提供するための x.500 仕様の派生版です。**LDAP** は、特に高速で、計算が軽く、堅牢であるように設計されたデータベーススタイルの機能を提供します。**LDAP** アプリケーションプロトコルは、トランスポート、ディレクトリのレイアウト（構成）、データへのアクセス方法（読み取り、書き込み、削除など）を詳細に説明します。

LDAP は認証と認可の両方のデータを提供することができますが、主に認可だけに使用され、認証には **Kerberos**（信頼できる第三者認証）がよく使われています。興味深いことに、**Microsoft** の **Active Directory** サービス（少なくとも認証・認可用）は、**LDAP** と **Kerberos** が結合したものと大差ないのです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

LDAP 認証

- 認証の一元化
- マルチドメイン
- マルチプラットフォーム
- マルチベンダー
- シングルサインオン
- **PAM** と **sssd** を使用

LDAP (**L**ightweight **D**irectory **A**ccess **P**rotocol) は、ネットワーク上で分散型ディレクトリサービスを利用・管理するための業界標準プロトコルであり、オープンかつベンダーニュートラルであることを目的としています。

LDAP を集中認証に使用する場合、各システム（またはクライアント）は、ユーザー認証のために集中型 **LDAP** サーバーに接続します。

LDAP は **TLS** (**T**ransport **L**ayer **S**ecurity) と **SSL** **S**ecure **S**ockets **L** レイヤーを認識（サポート）します。**TLS**/**SSL** を使うと安全に暗号化された接続が可能になります。

クライアント構成は、**LDAP** サーバーに接続するための最小限の情報だけを指定する必要があります。**LDAP** サーバーの IP アドレス、検索ベース **DN**（ドメイン名）、オプションで **TLS** などの要素が認証に必要です。

追加のユーザ空間プログラムは、**Openldap-clients** パッケージによって提供されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

LDAP 認証プロセス

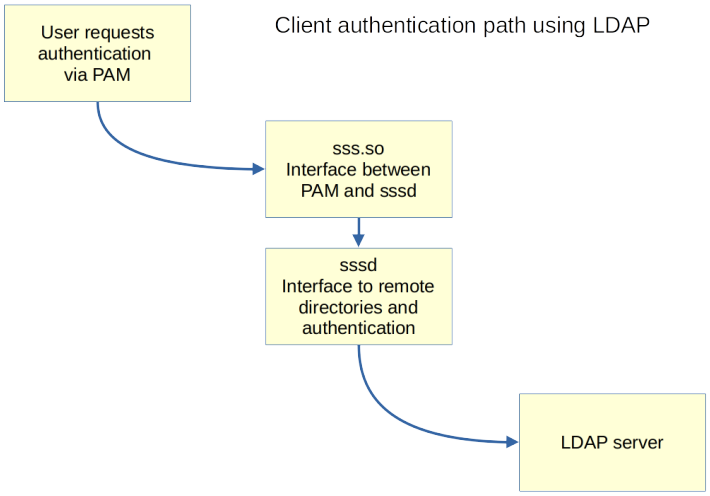


図 29.1: **LDAP クライアント認証プロセス**

上の図は、**LDAP/SSSD** が使用する情報経路を示したものです。**LDAP** サーバーへの接続は、**PAM**（Pluggable authentication module）から始まります。PAM は Linux のデフォルトの認証機構で、LDAP と SSSD はデフォルトの認証構成に追加される拡張モジュールです。

- **PAM** は **pam.sss.so モジュール**を使うように構成されます
- **sss.so** と **sssd** は **sssd** と **LDAP** が統合された設定ファイルを使います

/etc/sssd/conf.d/00-sssd.conf

```
[sssd]
config_file_version = 2
domains = example.com
services = nss, pam,autofs

[domain/example.com]
enumerate = true
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://192.168.122.154/
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
cache_credentials = True
ldap_tls_reqcert =allow
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

LDAP と sssd パッケージ

- **sssd** システム セキュリティ サービス デーモン
- **sssd-ldap** は、LDAP と併用する場合の sssd の設定機構
- **sssd-tools** sssd で使用する各種ツール
- **oddjob-mkhomedir** ホームディレクトリを作成する PAM モジュール
- **ldap-utils** ユーザー スペース ユーティリティ- で ldap からユーザを追加・削除するツールが含まれる、**Ubuntu** 専用
- **openldap-clients** ldap-utils と同様に **CentOS** 専用

クライアント システムを **LDAP** 認証用に構成した場合には、以下のファイルが変更されます。

```
/etc/sssd/conf.d/00-sssd.conf
/etc/pam.d/common-session.conf
```

/etc/sssd/conf.d/00-ssd.conf ファイルには複数のコンフィギュレーションがある場合、順番に並べられるようにするための２桁のプレフィックスが使用されています。

/etc/pam.d/ 設定ファイルはお使いのディストリビューションによって異なる場合があります。

- **CentOS**-8 /etc/pam.d/system-auth
- **Ubuntu**-22.04 /etc/pam.d/common-session.conf

これらのファイルは手編集することが出来ます。

以前のディストリビューションリリースでは、設定に使用する特定のユーティリティがありました。**sssd** とその関連の採用により、クライアント設定の複雑さが大幅に軽減され、専用の設定ツールは不要になりました。テキストファイルの設定情報は、お好みのテキストエディターで設定することができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

29.2 演習 **

課題 29.1: LDAP と TLS による集中認証 - LDAP サーバー アプライアンスのインストール

リソース要件

この演習では、追加の仮想マシンをインストールする必要があります。これは、クライアント構成をテストするために使用される小型の **LDAP サーバー** です。**LDAP** アプライアンスは、約 512MB のメモリーと 2GB のディスクスペースを使用します。この演習は、授業外で行ってください。

この演習では、集中型ネットワーク認証のための **LDAP** サーバーアプライアンスをインストールします。**ターンキーの openldap** アプライアンスサーバーは、さまざまなディストリビューションで利用可能であることから選択されました。このアプライアンスをインストールすると、新しい **VM** が作成されます。この **アプライアンス VM** は、メモリーは 512MiB、ディスクは 1GB 以下とかなり小型です。ディスクスペースは 20GB まで動的に拡張可能ですが、ここでの使用率はかなり低いままです。アプライアンスを入手する方法はいろいろありますが、その一つの選択肢として、textbfVMWare と **Virtual-Box** と互換性のある OVA イメージ ファイルを以下のディレクトリに用意しています。また別の方法として、**VMWare Player** と **VMWare Workstation** で動作する **vmdk** イメージファイルも以下のディレクトリから利用可能です。

https://training.linuxfoundation.org/cm/LFS207

利用可能なオプションは以下を参照してください
https://www.turnkeylinux.org/openldap

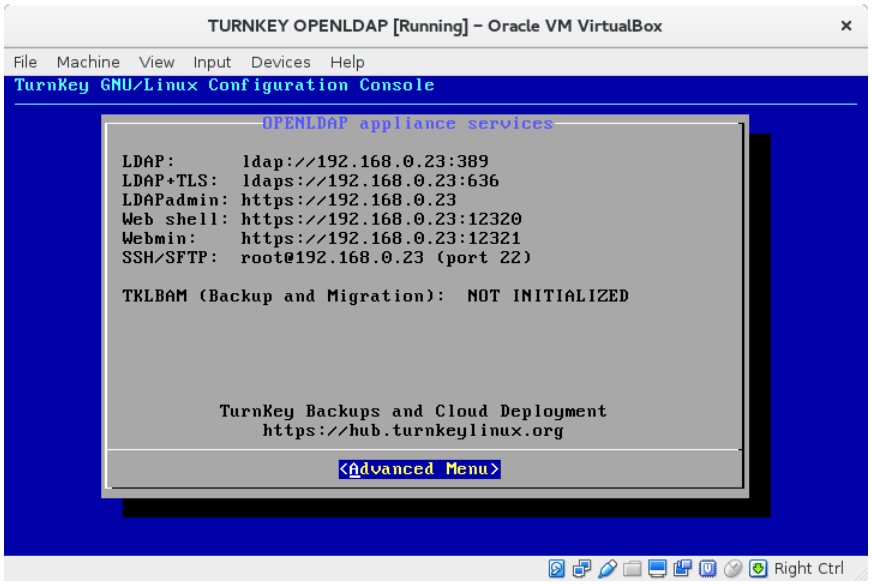


図 29.2: **ターンキー OpenLDAP アプライアンス サービス**

1. アプライアンスのインポート
   - `turnkey-openldap-14.1-jessie-amd64.ova` が利用できることを確認する（バージョンは異なる可能性がある）
   - **Virtual-Box Manager** を開く
   - `File -> Import Appliance` を選択する
   - **OpenLADP appliance** の場所を入力（または選ぶ）
   - **Next** を選択し指示にしたがう
2. 今作られた **TURNKEY OPENLDAP** 仮想マシンが起動する
3. LDAP アプライアンスの初回起動時に、以下のパラメータで構成する

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

- Root のパスワードは "pa$$w0rd"
- Admin のパスワードは "pa$$w0rd"
- ドメイン は "example.com"
- "HUB","System Notification" や "Update" services は無い

Turnkey OpenLDAP Appliance Summary が見えるはずです、これがアプライアンスが **正常動作**している状態の画面です.

課題 29.2: LDAP と TLS による集中認証 - LDAP サーバーのテストとユーザーの追加

この演習では、i**LDAP** サーバーの基本的な動作を確認します。具体的には、サーバーがクエリに応答する機能、および、**POSIX** グループとユーザーを追加する機能を確認します。ユーザーを追加したら、**wireshark** で接続を監視し、**LDAP** サーバーとの間でプレーンテキストの情報転送を観察することができます。平文のユーザー情報を持つことはセキュリティポリシー上好ましくないので、データストリームを暗号化することにします。

非常に重要

このラボでは、**LDAP** サーバーとの通信にクリアテキストを使用します。暗号化キーの使用は、このコースの範囲外です。クリアテキストは、どのような種類の本番環境でも **推奨されません。**ラボ環境に対応するため、証明書の検証を無効にします。キーを検証しない機能はラボでは便利ですが、本番システムでこの技術を **使用しないで**ください。

この演習では、以前に **LDAP** サーバーとの通信に使ったマシンを利用します。

`ready-for.sh` スクリプトが実行されて、ソリューションとリソースファイルはディレクトリにロードされ展開されているものとします。

1. 適切なコマンドで openldap-clients パッケージを検証まはたインストールします
   Ubuntu

   ```
   # apt install sssd sssd-ldap sssd-tools oddjob-mkhomedir ldap-utils
   ```

   CentOS

   ```
   # yum install sssd sssd-ldap sssd-tools oddjob-mkhomedir openldap-clients
   ```

2. 前項の **Turnkey OpenLDAP Appliance Summary** 画面に表示されている **LDAP** サーバーの ip アドレスを使用します。演習では、**example.com** ドメインの簡単な **LDAP** 検索を実行します。ドメイン「example.com」のベースレコードに関する情報が表示されるはずです。

   ```
   # ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub"objectclass=*"
   ```

   ```
   # extended LDIF
   #
   # LDAPv3
   # base <dc=example,dc=com> with scope subtree
   # filter: (objectclass=*)
   # requesting: ALL
   #

   # example.com
   dn: dc=example,dc=com
   objectClass: top
   objectClass: dcObject
   objectClass: organization
   o: example.com
   dc: example

   # admin, example.com
   dn: cn=admin,dc=example,dc=com
   objectClass: simpleSecurityObject
   objectClass: organizationalRole
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
cn: admin
description: LDAP administrator

# Groups, example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Groups

# users, Groups, example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

# Users, example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Users

# Hosts, example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Hosts

# Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Idmaps

# samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: SAMBA Access Account

# Aliases, example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Aliases

# nsspam, example.com
dn: cn=nsspam,dc=example,dc=com
cn: nsspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: NSS/PAM Access Account

# search result
search: 2
result: 0 Success

# numResponses: 11
# numEntries: 10

^^I
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

3. **solutions** ディレクトリにある groups.ldif ファイルを使って新しいグループ情報を追加します。

```
# ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f groups.ldif
```

```
 Enter LDAP Password:
adding new entry "cn=luser1,ou=Groups,dc=example,dc=com "
^^I
```

4. **ldapsearch** コマンドを使うと新しいグループが見えるはずです。

```
# ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub"objectclass=*"
```

```
 extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: example

# admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# Groups, example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Groups

# users, Groups, example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

# Users, example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Users

# Hosts, example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Hosts

# Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
objectClass: top
ou: Idmaps

# samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: SAMBA Access Account

# Aliases, example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Aliases

# nsspam, example.com
dn: cn=nsspam,dc=example,dc=com
cn: nsspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: NSS/PAM Access Account

# luser1, Groups, example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
gidNumber: 999001
memberUid: luser1

# search result
search: 2
result: 0 Success

# numResponses: 12
# numEntries: 11
^^I
```

新しいグループを追加すると上の出力 **numResponses** と **numEntries** が増えている点に注目してください。

5. **solutions** ディレクトリにある `users.ldif` ファイルを使って新しいユーザー情報を追加します。

```
# ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f users.ldif
```

```
Enter LDAP Password:
adding new entry "cn=luser1,dc=example,dc=com"
^^I
```

6. **ldapsearch** コマンドを使うと新しいユーザーが見えるはずです。

```
# ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub"objectclass=*"
```

```
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: example

# admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# Groups, example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Groups

# users, Groups, example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

# Users, example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Users

# Hosts, example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Hosts

# Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Idmaps

# samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: SAMBA Access Account

# Aliases, example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Aliases

# nsspam, example.com
dn: cn=nsspam,dc=example,dc=com
cn: nsspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
description: NSS/PAM Access Account

# luser1, Groups, example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
gidNumber: 999001
memberUid: luser1

# luser1, example.com
dn: cn=luser1,dc=example,dc=com
uid: luser1
cn: luser1
givenName: luser1
sn: linux
homeDirectory: /home/users/luser1
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
uidNumber: 999001
gidNumber: 999001

# search result
search: 2
result: 0 Success

# numResponses: 13
# numEntries: 12
^^I
```

7. wireshark をインストール、またはインストールされていることを確認します。

```
# yum install wireshark-gnome
# wireshark
```

8. wireshark を使って ldapsearch コマンドをキャプチャーします, これはテキストデータです。

   Wireshark には優秀なフィルター機能があり、フィルターに "tcp and (tcp.port==389) or (tcp.port==636)" と指定すると **wireshark** は **LDAP** 通信だけを表示します。

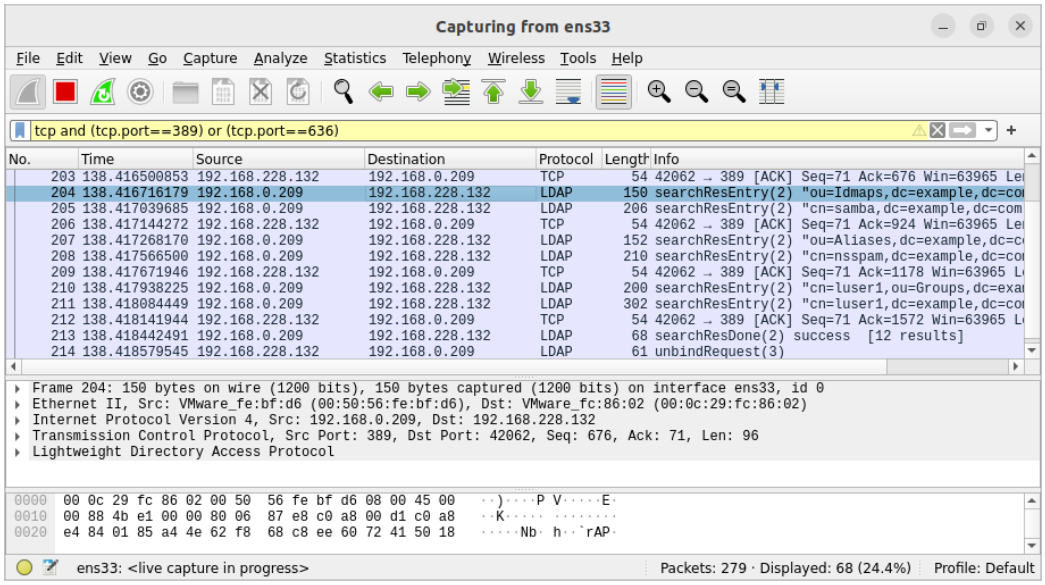


図 29.3: **Wireshark による LDAP パケットのトレース**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 29.3: LDAP による集中認証によるユーザー認証

この演習では、クライアントが **LDAP** を使った集中認証を使うように設定します。

非常に重要

CentOS-8 や **Ubuntu 20.04** から、リモートディレクトリへのアクセスや認証の管理に **System Security Services Daemon** や **sssd** が使用されるようになりました。**sssd** の使い方や設定方法は、利用するディストリビューションで共通です。

1. 必要なパッケージをインストールします

```
# apt install sssd sssd-ldap ldap-utils oddjob-mkhomedir
```

2. **ldap_uri** フィールドに **ldap** サーバーの IP アドレスを代入して、リストにある要素を含むように **sssd** 設定を作成または更新します。

/etc/sssd/conf.d/00-sssd.conf

```
[sssd]
config_file_version = 2
domains = example.com
services = nss, pam,autofs

[domain/example.com]
enumerate = true
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://192.168.122.154/
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
cache_credentials = True
ldap_tls_reqcert =allow
```

3. **sssd.conf** ファイルのパーミッションを確認して設定します。

```
# chmod 600 /etc/sssd/conf.d/00-sssd.conf
# chown root.root /etc/sssd/conf.d/00-sssd.conf
```

4. ホームディレクトリを自動的に作成するように **oddjob** を設定する。以下の例のように `/etc/pam.d/common-session` に **oddjob** を追加します。

/etc/pam.d/common-session.conf

```
session required        pam\_unix.so

session optional        pam\_oddjob\_mkhomedir.so

session optional        pam\_sss.so
```

5. 変更を反映するためにサービスを再起動します。

```
# systemctl restart sssd oddjobd
# systemctl enable sssd oddjobd
```

6. **ldap** サーバーをテストします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

7. **luser1** について利用可能なユーザー情報を確認します。

```
# getent passwd luser1
```

以下のような応答が返ることを確認してください。

```
luser1:*:999001:999001:luser1:/home/users/luser1:
```

8. Test the user authorization is functioning for user **luser1** に対するユーザー認証が行われることを確認してください。
パスワードは password です。ホームディレクトリは利用可能になっていないかもしれません。

```
# ssh luser1@localhost
```

9. データストリームは（暗号化されていない）クリアテキストであることを確認してください。
   - ポート 389 と 636 を取得するようにフィルター設定して **wireshark** のセッションを開きます。
   - **wireshark** がパケットをキャプチャーしている間、luser1 として localhost に SSH 接続して log をオン／オフします。
   - **wireshark** の出力を観測します。
10. 証明書の確認を無効にします。

非常に重要

プロダクション システムで証明書の確認を無効化することは **推奨されません。**

お好みのエディターで `/etc/sssd/sssd.conf` ファイルを編集します。
`[domain/default]` セクションに以下の行を追加します。

```
ldap_tls_reqcert = never
```

`sssd` サービスを再起動します。

```
# systemctl restart sssd
```

11. **LDAP** によって **luser1** のユーザー情報が提供されていることを確認してください。

```
# getent passwd luser1
```

12. **luser1** に対するユーザー認証が機能していることを確認してください。

```
# ssh luser1@localhost
```

13. **LDAP** サーバーからの入出力ストリームデータ（双方向）が暗号化されていることを確認してください。
   - ポート 389 と 636 を取得するようにフィルター設定して **wireshark** のセッションを開きます。
   - **wireshark** がパケットをキャプチャーしている間、luser1 として localhost に SSH 接続して log をオン／オフします。
   - **wireshark** の出力を観測します。
14. 終了したら元の設定に戻してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 30 章

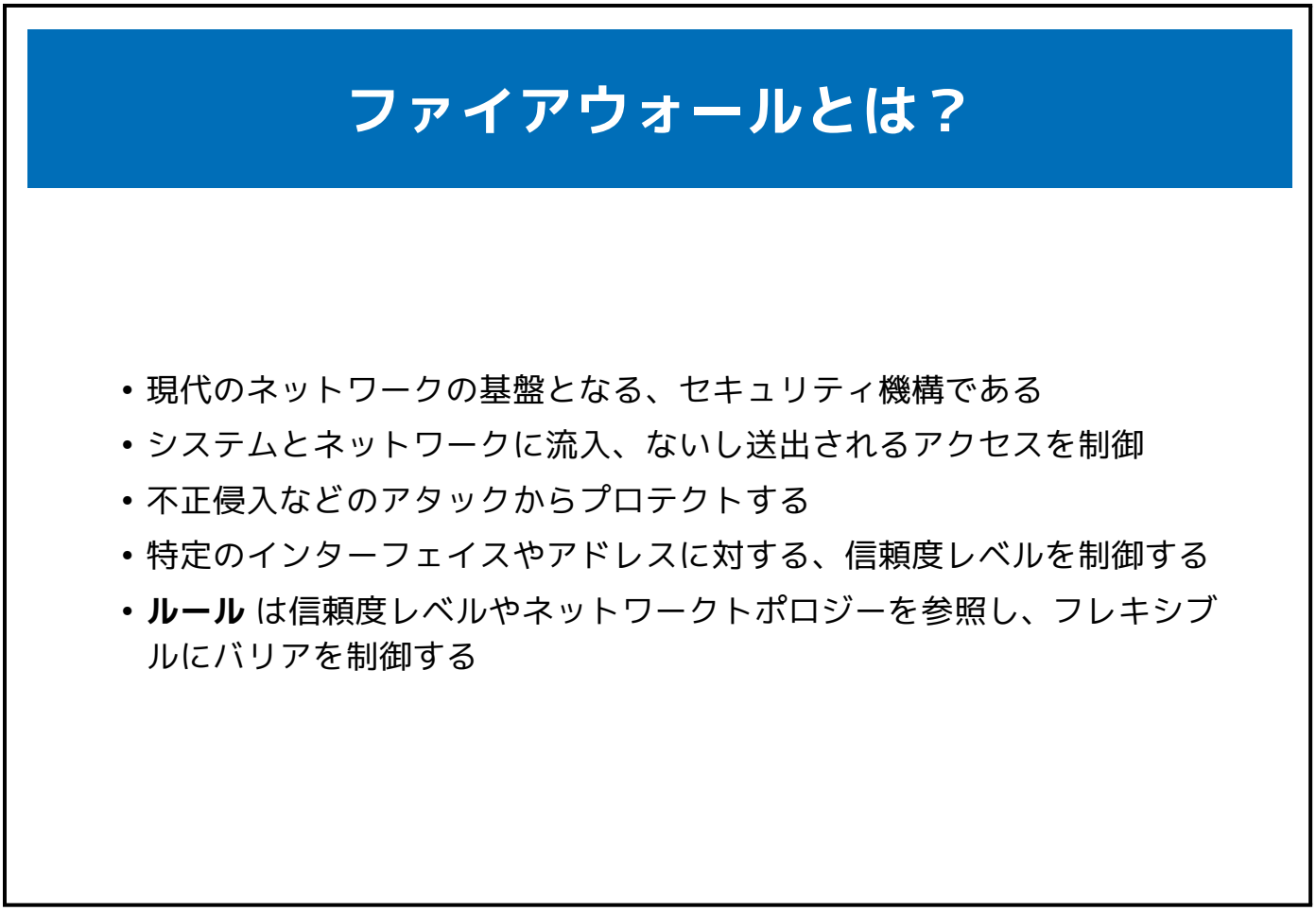
ファイアウォール

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- ファイアウォールとは何か、なぜ必要なのかを理解する。
- コマンドラインとグラフィカルインターフェイスのそれぞれで利用できるツールを理解する。
- **firewalld** と **firewall-cmd** プログラムについて説明する。
- ゾーン、ソース、サービス、ポートの扱い方を理解する。

30.1 ファイアウォール



ファイアウォール は、全てのネットワーク トラフィックの監視と制御を行うネットワーク セキュリティ システムです。流入と送出の両方のネットワーク接続とパケットに **ルール** を適用し、特定の接続の信頼レベルとネットワークトポロジに応じた柔軟なバリア（つまり、ファイアウォール）を構築します。

ファイアウォールはハードウェア ベースでもソフトウェア ベースでもかまいません。ファイアウォールはネットワークルーターだけでなく、個々のコンピュータやネットワーク ノード内に構築することもできます。多くのファイアウォールにはルーティング機能もあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

パケット フィルタリング

- ファイアウォール構築の基礎となる技術である
- ネットワーク ストリーム上の全パケットがチェック対象となる
- フィルター後に、システムで設定した **ルール** が適用される
- アクションには 拒否、破棄、削除、リダイレクト などがある

ほとんど全てのファイアウォールは **パケット フィルタリング** を行っています。

情報はネットワークを介してパケット形式で送信されます。これらのパケットには次のものが含まれます。

- ヘッダー (Header)
- ペイロード (Payload)
- フッター (Footer)

ヘッダーとフッターには宛先と送信元のアドレス、パケットの種類、準拠するプロトコルの種類、さまざまなフラグ、ストリーム内のパケット番号、そして転送に関するその他のあらゆるメタデータに関する情報が含まれています。

実際のデータは、ペイロードに格納されます。

パケット フィルタリングは、アプリケーション、トランスポート、ネットワーク、データリンクなど、ネットワーク伝送の複数の階層でパケットをインターセプトします。

ファイアウォールは、各パケットが次のことを行うための一連の **ルール** を確立します:

- 内容、アドレスなどに基づいて、承認または拒否する
- 何らかの方法で破棄する
- 別のアドレスにリダイレクトさせる
- セキュリティに関して調査する
- その他

パケット フィルタリングの結果を踏まえて実行されるルールとアクションを決定する、さまざまなユーティリティが存在します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ファイアウォールの世代

- **パケット フィルタリング**:
  パケットを監査し、パケットを破棄、拒否、転送する
- **ステートフル フィルター**:
  接続ステート を検査し新規接続か、既存接続の延長か、それ以外かを判別する
- **アプリケーションレイヤー ファイアウォール**:
  アプリケーションが利用するプロトコルパケットを認識する

初期のファイアウォール（1980 年代後半に遡ります）は、**パケット フィルタリング** に基づくものでした。各ネットワーク パケットのコンテンツを検査し破棄、拒否、または転送していました。パケットが含まれるストリームの通信状態に関する **接続ステート (connection state)** は考慮されていませんでした。

次の世代のファイアウォールは **ステートフル フィルタ** に基づいており、パケットの **接続ステート** も調査して、新しい接続であるか 既存の接続の延長なのか を判定材料に利用します。DoS 攻撃はこの種のファイアウォールを攻撃して過剰な負荷をかけようとします。

第３世代ファイアウォールは **アプリケーションレイヤー ファイアウォール** と呼ばれ、アプリケーションがどんなプロトコルで接続しているかを認識します。通常のフローと異なるものは全て拒否できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.2 インターフェイス

ファイアウォールのインターフェイス と ツール

- コマンドライン ツール
  - **iptables**
  - **firewall-cmd**
  - **ufw**
- グラフィカル インターフェイス
  - **system-config-firewall**
  - **firewall-config**
  - **gufw**
  - **yast**
- **shorewall** など、これ以外にもユーティリティが存在する
- コマンドライン ツールは、ディストリビューションによる差異が少ない

システムへのファイア ウォール設定には、次の２つ進め方があります。

- コマンドラインから比較的低レベルのツールを利用し、併せて `/etc` 配下にある **iptables**、**firewall-cmd**、**ufw** などの各種構成ファイルを編集する。
- 堅牢なグラフィカル インターフェイスを利用する。**system-config-firewall**、**firewall-config**、**gufw**、**yast** など

私達は以下の理由から、低レベルのツールを使用することにします。

- 低レベルツールは、グラフィカルなツールほど頻繁に変更されない。
- 低レベルツールの方が、より多くの機能を持っていることが多い。
- 低レベルツールは、ディストリビューション毎の違いが少ない。**GUI**（のデザイン）はそれぞれのディストリビューションに固有なもので、ディストリビューション ファミリー毎にかなり異なる傾向があります。

欠点は、最初は習得が難しく感じることです。

firewall-cmd と **firewall-config** の両方を含む最新の **firewalld** パッケージに絞って説明します。デフォルトでこれらが含まれていないディストリビューションでも、ソースから比較的簡単にインストールすることができます。演習で必要になった時には我々もそうします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

iptables を使用しない理由

- 最近のファイアウォールではユーザー側で **iptables** パッケージを利用しているケースが実際には多い
- **iptables** は、これから説明していく **firewalld** と共通のカーネル ファイアウォール実装インターフェイスを利用している
- **iptables** を使って有用な設定ができるまで理解するにはとても時間がかかることから、今回は取り上げないことにした

LFS311: Linux for System Engineers

以下を参照してください
https://training.linuxfoundation.org/linux-courses/system-administration-training/linux-for-system-engineers

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.3 firewalld

firewalld と firewall-cmd

- **Dynamic Firewall Manager** (**firewalld**)
- **zones** と協調動作する
- **IPv4** と **IPv6** の両方をサポートする
- **一時的な変更** と **恒久的な変更** の設定を分離できる
- メインのツールは **firewall-cmd** である
- **iptables** ユーティリティの置き換えとなる
- **systemd** の構成要素である
- 徐々に主要なディストリビューションに採用されはじめた、ソースからのインストールも可能である

firewalld は **Dynamic Firewall Manager** です。ネットワーク インターフェイスや接続の信頼レベルが定義されているネットワーク／ファイアウォールの **ゾーン** と協調動作します。これは **IPv4** と **IPv6** の両方のプロトコルをサポートします。

さらに構成情報の **ランタイム変更 (＝一時的な変更)** と **恒久的な**（持続的な）変更を分け、ファイアウォール ルールを追加するためのサービス、またはアプリケーションとのインターフェイスも持っています。

設定ファイルは `/etc/firewalld` と `/usr/lib/firewalld` に保存されます。`/etc/firewalld` 内のファイルは他のディレクトリの設定ファイルを上書きするものなので、システム管理者が編集する必要があります。

コマンドラインツールは **firewall-cmd** です。この先で説明していきます。

次に進む前に、以下を実行してみることをお勧めします。

```
$ firewall-cmd --help
```

```
  Usage: firewall-cmd [OPTIONS...]
  ....
  Status Options
    --state                 Return and print firewalld state
    --reload                Reload firewall and keep state information
    --complete-reload       Reload firewall and loose state information
    --runtime-to-permanent  Create permanent from runtime configuration
  ....
```

全部で約 200 行あるのでここに全部は掲載できません。しかし、殆どのオプションは名前から何をするものか理解できるでしょう。

firewalld は **iptables** を置き換えるサービスです。両方のサービスを同時に実行するとエラーになります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

firewalld サービスの状態

- **firewalld** は他サービスと同様に 有効化／無効化、起動／停止 ができる

```
$ sudo systemctl [enable/disable] firewalld
$ sudo systemctl [start/stop] firewalld
```

- ステータの表示

```
$ sudo systemctl status firewalld
$ sudo firewall-cmd --state
```

- ２つ以上のネットワーク インターフェイスがある場合は **IP フォワード** を有効化する必要がある

```
$ sudo sysctl net.ipv4.ip_forward=1
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

firewalld はファイア ウォールの使用と構成を実行するサービスで、上に示した一般的なやり方で 有効化／無効化、開始／停止ができます。次のいずれかの方法で、現在の状態を表示できます:

```
$ sudo systemctl status firewalld
```

```
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
   Active: active (running) since Tue 2015-04-28 12:00:59 CDT; 5min ago
 Main PID: 777 (firewalld)
...
```

```
$ sudo firewall-cmd --state
```

```
running
```

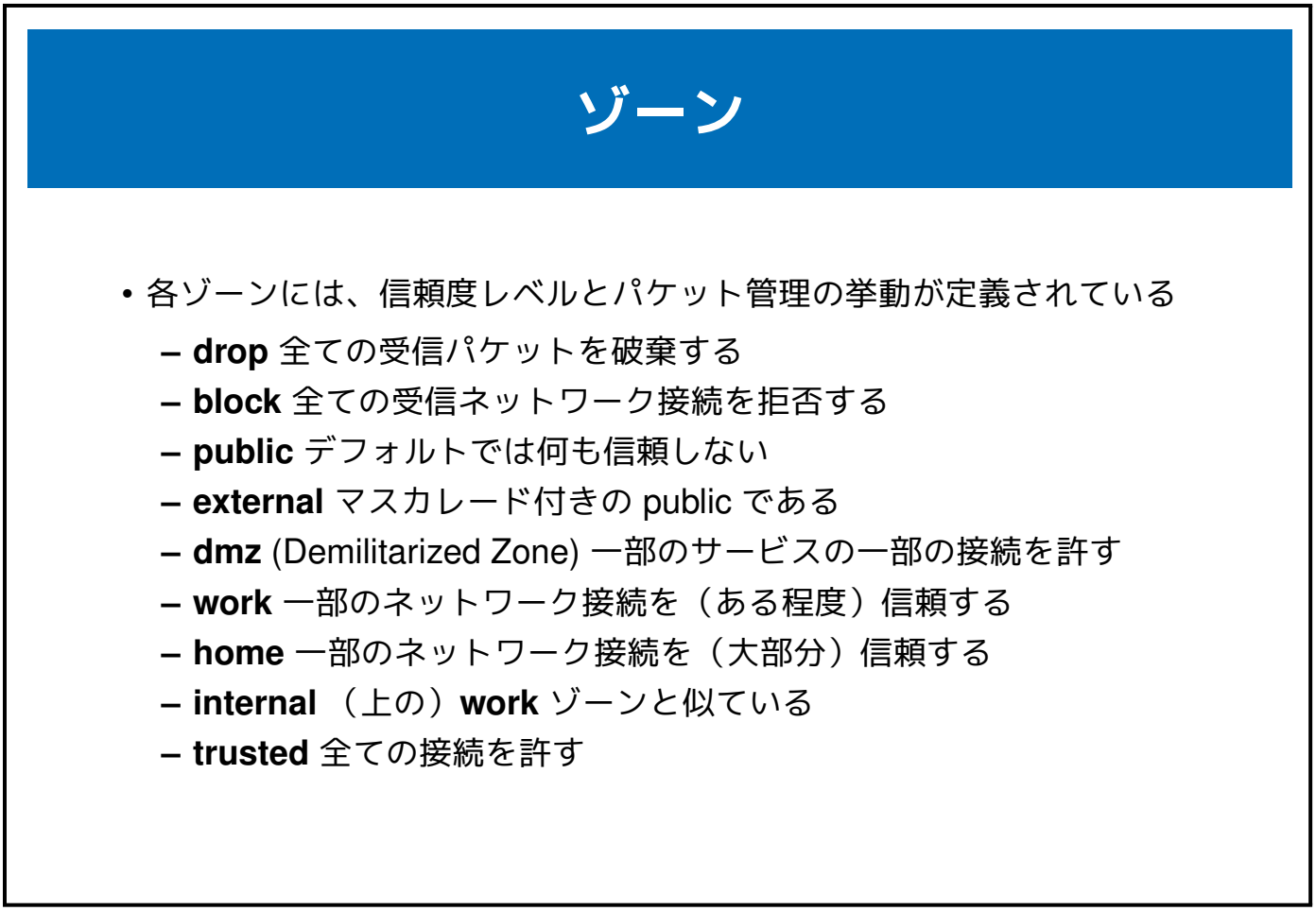
IPv4 を使っている時に、２つ以上のネットワーク インターフェイスがある場合には **IP フォワード** を有効にする必要があります。ランタイムで（動作中に）これを行うには、次のいずれかを実行します。

```
$ sudo sysctl net.ipv4.ip_forward=1
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

(**echo** を正しく実行させるには２番目のコマンドを root として実行する必要があります) ただし、これは恒久的ではありません。恒久化するには `net.ipv4.ip_forward=1` という行を `/etc/sysctl.conf` に追加して再起動するか、`sudo sysctl -p` とタイプして、再起動無しで設定変更を読み込ませる必要があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.4 ゾーン



firewalld は **ゾーン** と連携して動作します。それぞれのゾーンには、信頼レベルと受信／発信パケットに対する特定の動作が定義されています。各インターフェイスは特定のゾーンに属します。（通常は **NetworkManager** が該当するゾーンを **firewalld** に通知します）ゾーンは **firewall-cmd** または **firewall-config** GUI で変更できます。ゾーンには:

- **drop**: 全ての受信パケットは、応答せずに破棄されます。発信接続だけが許可されます。
- **block**: 全ての受信ネットワーク接続が拒否されます。許可される接続は、システム内からの接続だけです。
- **public**: ネットワーク上のコンピュータを信頼しません。意識的に選択された特定の着信接続だけが許可されます。
- **dmz**: (Demilitarized Zone) 一部の（全てではない）サービスへのアクセスを公開する場合に使用します。特定の受信接続だけが許可されます。
- **external**: ルーターなどで **マスカレード** が使用されている場合に使用されます。信頼レベルは **public** ゾーンと同じです。
- **work**: 接続されたノードを（完全ではありませんが）無害であると信頼します。特定の受信接続だけが許可されます。
- **home**: ほとんどの場合、他のネットワークノードを信頼しますが、許可する受信接続を選択します。
- **internal**: **work** ゾーンに似ています。
- **trusted**: 全てのネットワーク接続が許可されます。

システムインストール時に、全てではないにしてもほとんどの **Linux** ディストリビューションは、全てのインターフェイスのデフォルトに **public** ゾーンを選択します。説明したいくつかのゾーンの違いは明確ではありません。ここでは詳細に説明しませんが、必要以上にオープンなゾーンの使用は避けてください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ゾーン管理の例 1

- デフォルトゾーンの取得する

```
$ sudo firewall-cmd --get-default-zone
public
```

- 現在使用されているゾーンのリストを取得する

```
$ sudo firewall-cmd --get-active-zones
public
  interfaces: eno16777736
```

- 利用可能な全てのゾーンをリストする

```
$ sudo firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

```
$ firewall-cmd --help
```

```
....
Zone Options
  --get-default-zone   Print default zone for connections and interfaces
  --set-default-zone=<zone>
                       Set default zone
  --get-active-zones   Print currently active zones
  --get-zones          Print predefined zones [P]
  --get-services       Print predefined services [P]
  --get-icmptypes      Print predefined icmptypes [P]
  --get-zone-of-interface=<interface>
                     Print name of the zone the interface is bound to [P]
  --get-zone-of-source=<source>[/<mask>]
                 Print name of the zone the source[/mask] is bound to [P]
  --list-all-zones  List everything added for or enabled in all zones [P]
  --new-zone=<zone>    Add a new zone [P only]
  --delete-zone=<zone> Delete an existing zone [P only]
  --zone=<zone>  Use this zone to set or query options, else default zone
                       Usable for options maked with [Z]
  --get-target         Get the zone target [P] [Z]
  --set-target=<target>
                       Set the zone target [P] [Z]
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ゾーン管理の例 2

- デフォルトゾーンを **trusted** 変更し、更に変更を元に戻す

```
$ sudo firewall-cmd --set-default-zone=trusted
success
$ sudo firewall-cmd --set-default-zone=public
success
```

- 一時的にインターフェイスを特定のゾーンに割り当てる

```
$ sudo firewall-cmd --zone=internal --change-interface=eno1
success
```

- それを恒久化する:

```
$ sudo firewall-cmd --permanent --zone=internal --change-interface=eno1
success
```

`/etc/firewalld/zones/internal.xml` が作成される

全ての **firewalld** の制御は **firewall-cmd** プログラムを介して行います。より詳細な情報は、次の方法で取得できます。

```
man firewalld-cmd
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ゾーン管理の例 3

- 特定のインターフェイスに関連付けられたゾーンを突き止めるには:

```
$ sudo firewall-cmd --get-zone-of-interface=eno1
public
```

- 特定のゾーンに関する、全ての詳細情報を取得するには

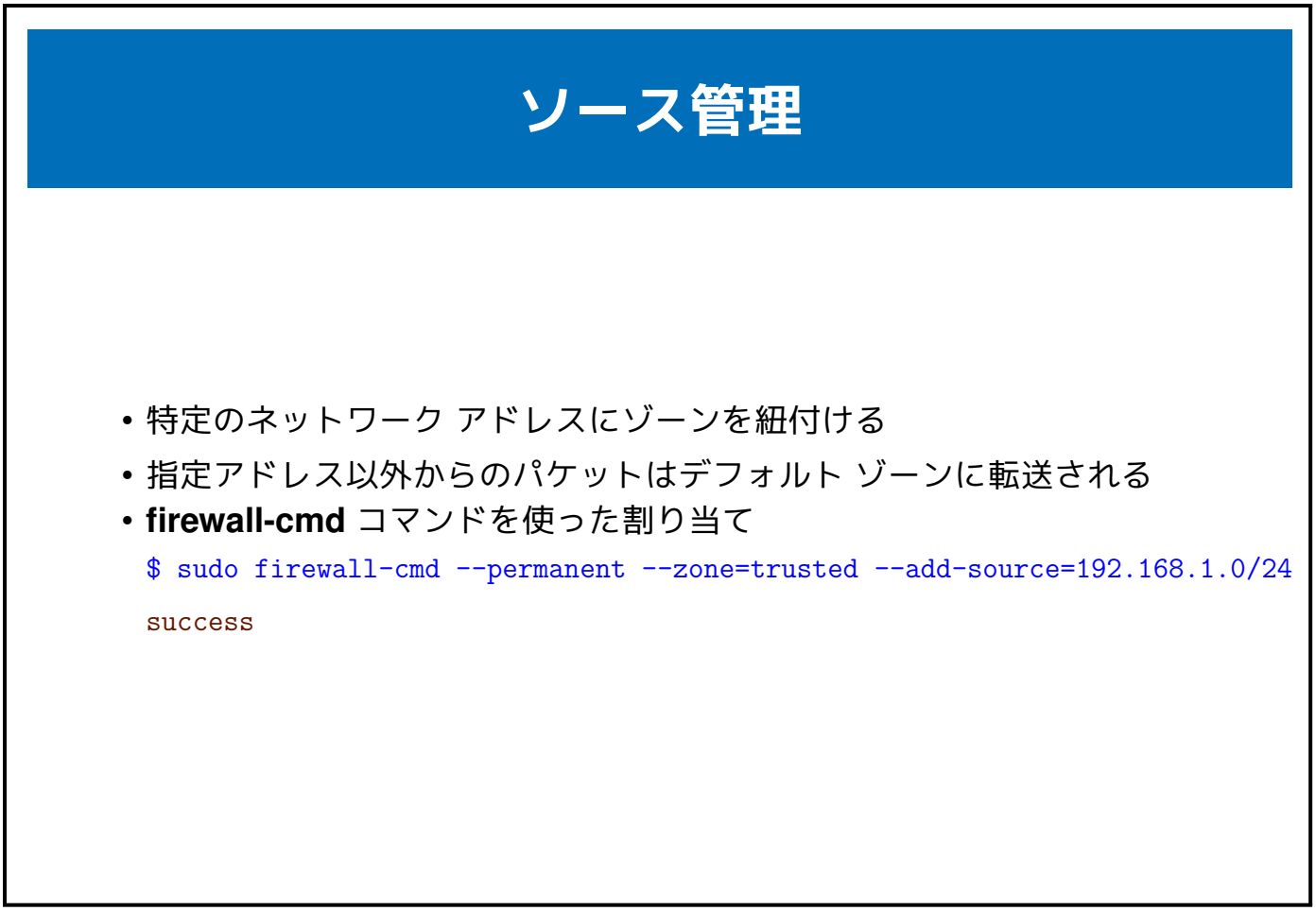
```
$ sudo firewall-cmd --zone=public --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eno1
  sources:
  services: chromecast libvirt libvirt-tls nfs nfs3 rsyncd ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

更に知りたい？

firewall-cmd に関する包括的なレファレンスは以下を参照してください。
https://firewalld.org/documentation/man-pages/firewall-cmd.html

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.5 ソース管理



どのゾーンも、ネットワーク インターフェイスだけでなく、特定のネットワーク アドレスに結び付けることもできます。
次の場合、パケットはそのゾーンに割り当てられます。

- 既にそのゾーンに結び付けられているアドレスからパケットを取得している場合。もしそうでなければ、
- ゾーンに結び付けられたインターフェイスから取得している場合

上記の取得方法に当てはまらないパケットは、デフォルトゾーン（つまり、通常は **public**）に割り当てられます。
ソースをゾーンに（恒久的に）設定するには

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24
   success
```

これは 192.168.1.x の IP アドレスを持つソースを全て **trusted** ゾーンに追加することを意味します。--remove-source オプションを使用すれば、以前に割り当てられたソースをゾーンから削除することができます。
また --change-source を使用すれば、ゾーンを変更できることも覚えておいてください。
ゾーンに結び付けられたソースをリストするには

```
$ sudo firewall-cmd --permanent --zone=trusted --list-sources 192.168.1.0/24
```

上記のどちらのコマンドも --permanent オプションを省略すると、現在のランタイム動作だけに適用されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.6 サービスとポートの管理

サービス管理

- 利用可能なサービスを見るには

```
$ sudo  firewall-cmd --get-services
```

- ゾーンに含まれるものを見るには

```
$ sudo firewall-cmd --list-services --zone=public
```

- ゾーンにサービスを追加するには

```
$ sudo firewall-cmd --permanent --zone=home --add-service=dhcp
success
$ sudo firewall-cmd --reload
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

これまで、特定のインターフェイスやアドレスをゾーンに割り当てる方法を説明しましたが、ゾーン内でアクセスできる **サービス** や **ポート** については説明していませんでした。利用可能な全てのサービスを表示するには

```
$ sudo  firewall-cmd --get-services
```

```
RH-Satellite-6 amanda-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns ftp high-availability http\
   https imaps ipp ipp-client ipsec kerberos kpasswd ldap ldaps libvirt libvirt-tls mdns mountd ms-wbt\
    mysql nfs ntp openvpn pmcd pmproxy pmwebapi pmwebapis pop3s postgresql proxy-dhcp radius rpc-bind
     ↪  samba\
    samba-client smtp ssh telnet tftp tftp-client transmission-client vnc-server wbem-https
```

特定のゾーン内で現在利用可能なサービスを見るには:

```
$ sudo firewall-cmd --list-services --zone=public
```

```
dhcpv6-client ssh
```

ゾーンにサービスを追加するには

```
$ sudo firewall-cmd --permanent --zone=home --add-service=dhcp
```

```
success
\end{cmd}
\begin{out}[]
$ sudo firewall-cmd --reload
```

２番目のコマンドによる変更を有効にするためには `--reload` の指定が必要です。`/etc/firewalld/services` のファイルを編集して新しいサービスを追加することもできます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ポート管理

- ポート管理

```
$ sudo firewall-cmd --zone=home --list-ports
$ sudo firewall-cmd --zone=home --add-port=21/tcp
```

ポート管理は、サービス管理に非常に似ています。

```
$ sudo firewall-cmd --zone=home --add-port=21/tcp
```

```
success
```

```
$ sudo firewall-cmd --zone=home --list-ports
```

```
21/tcp
```

`/etc/services` を見ると port 21 が **ftp** に対応していることが確認できます。

```
$ grep " 21/tcp" /etc/services
```

```
ftp             21/tcp
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.7 ポートの付け替え（リダイレクト）

ポートの付け替えと NAT

パケット受信時に、場合によっては **宛先ポート番号**を変更したいケースがあります。

- 80 番ポート向けの受信パケットの宛先を 8080 番に変更したい
- --zone=external で指定された外部からの接続にゾーンを指定する

```
$ sudo firewall-cmd --zone=external \
                --add-forward-port=port=80:proto=tcp:toport=8080
$ sudo firewall-cmd --zone=external --list-all
```

ポート番号の付け替え時には、可能な限りルールを明確にするために、ゾーンとできれば IP アドレスを指定することが望ましいのです。

```
$ sudo firewall-cmd --zone=external --add-forward-port=port=80:proto=tcp:toport=8080
```

```
success
```

```
$ sudo firewall-cmd --zone=external --list-all
```

```
external
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
        port=80:proto=tcp:toport=8080:toaddr=
  source-ports:
  icmp-blocks:
  rich rules:
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ネットワーク アドレス変換（NAT）

ネットワーク アドレス変換（NAT）は、ファイアウォールにパケットが出入りする際に、ファイアウォールがパケットの送信元アドレスまたは宛先アドレスを変更するために使用する方法である。**NAT** にはいくつかの種類がある

- **DNAT** (**Destination Network Address Translation**)
  宛先のネットワーク アドレス変換
- **SNAT** (**Source Network Address Translation**)
  送信元のネットワーク アドレス変換
- **MASQUERADE（マスカレード）**, **SNAT** の一種

DNAT、**SNAT** と **マスカレード** については次ページ以降で説明します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

DNAT

DNAT (**Destination Network Address Translation**) は、パケットがインターネットからシステムに入る際に、エンドデスティネーション（「宛先」IP アドレス）、場合によっては宛先ポートを変更する。

- パケットを送信時には、**DNS** から取得したパブリック IP アドレスが付与される
- ファイアウォールにはパブリック IP アドレスが渡される
- パケットがファイアウォールを経由して内部ネットワークに到達し、目的のサービスに接続できるようにするには、パケットの宛先を変更する必要がある

DNAT ファイアウォール ルールは、通常 **NAT** テーブルの **pre-routing（ルーティング前）**チェーンに配置されます。これは **インターネット**に接続されたアダプターや、ウェブサーバーなどのサービスで使用されるパブリック アドレスになります。

ファイアウォールは通常（パケットを通過させるだけの）パススルー機器で（ローカルに実行しているサービスはほとんどない）、パケットを検査し、宛先ポートを調べて適用できるルールがあるかどうか確認します。私たちはパケットを以下のリールで検査します。

- 宛先 IP アドレス：ファイアウォールの IP アドレスである必要があります。
- 宛先ポート番号：サービスのポート番号になります。（http なら 80 番）
- **DNAT** ルールにより、宛先 IP アドレスはファイアウォール内のウェブサーバーのアドレスに変更されます。
- パケットはファイアウォールの向こうのウェブサーバーに **forwarded（フォワード）**されます。
- ウェブサーバーはパケットを受け入れて処理し、応答を返信します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION | Training & Certification

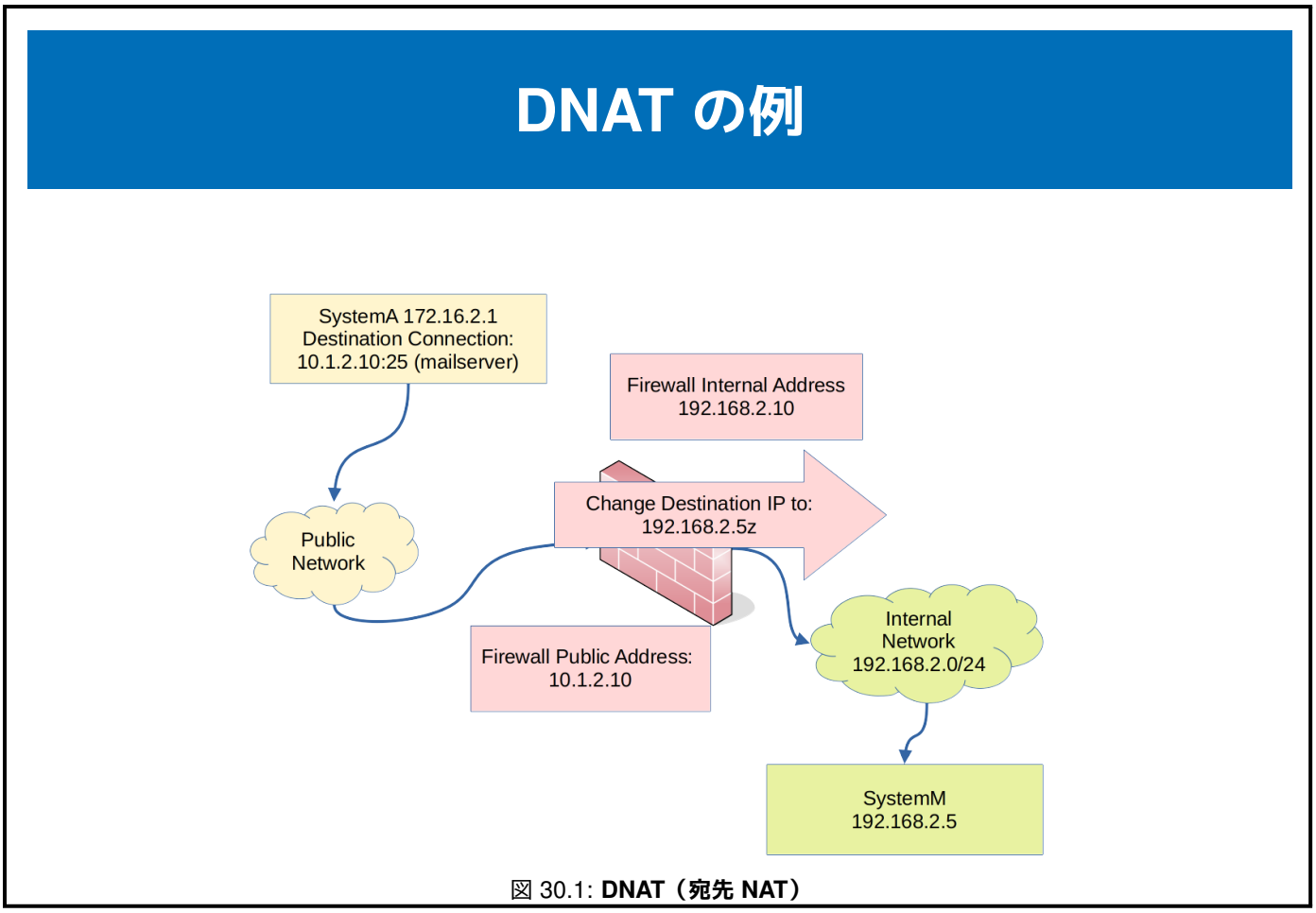


図 30.1: **DNAT（宛先 NAT）**

この図は、パケットがユーザーシステムからファイアウォールで保護されたネットワーク内のシステム上のサービスに至る典型的な経路を表しています。

- `SystemA`（IP 172.16.2.1）は、メール配信のために `systemM`（IP 10.1.2.25）のアドレスを調べます。
- 宛先ポート番号は、サービスのポート番号（smtp=25）になります。
- ファイアウォールの **DNAT** ルールは、宛先をファイアウォールの背後にあるメールサーバーのアドレス（IP 192.168.2.5）に変更します。
- パケットはファイアウォールの向こうのメールサーバーに **forwarded（フォワード）**されます。
- メールサーバーは、パケットを受け入れて処理し、応答を返信します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SNAT とマスカレード

Source Network Address Translation (**SNAT**) は、パケットがインターネットに向かう途中でシステムを離れる際に、パケット上のソース（「From」）IP アドレスを変更します。

- 内部ネットワークで使用されるアドレスがインターネットに公開されないようにするため、トランスレーションが必要
- ほとんどの内部ネットワークはプライベート ネットワークであり、ルーティング出来ない（＝インターネットにそのまま流せない）

- 外向きのインターフェイスの IP アドレスが DHCP で提供されている場合、**SNAT** を実装するために **MASQUERADE（マスカレード）**オプションが使用されます。
- マスカレードは **SNAT** よりも多くの計算資源を必要とします。
- 外向きのインターフェイスが静的（＝固定 IP）の場合は、SNAT を使ってパブリック IP アドレスを指定することができます。
- **SNAT** とマスカレードは **NAT** テーブルの **post-routing（後ルーティング）**チェーンにあります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

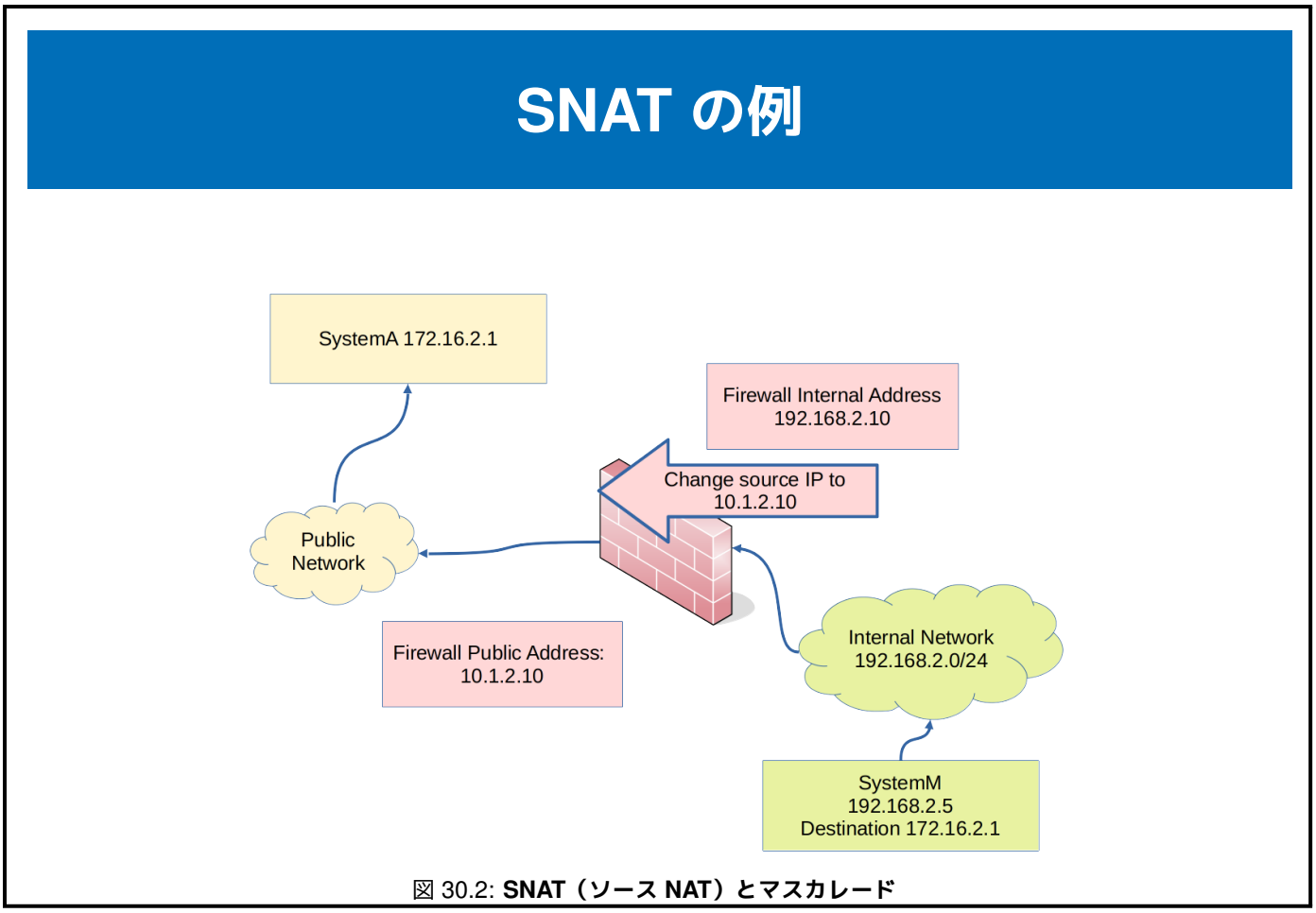


図 30.2: **SNAT（ソース NAT）とマスカレード**

SNAT または **マスカレード** ファイアウォール ルールは、通常、NAT テーブルのポスト ルーティング チェーンに配置されます。通常、ポスト ルーティングは、パケットがインターネットに向かう途中でシステムを離れる際の最後の検査または修正ポイントです。

- パケットの宛先アドレスは、接続を開始したリモート システムのアドレスになります。
- パケットの送信元アドレスは、サービスが使っている内部アドレスからファイアウォールの外向きアダプターのアドレスに変更されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

30.8 演習

課題 30.1: firewalld のインストール

ほとんどの **Linux** ディストリビューションには **firewalld** パッケージ（多目的に使える **firewall-cmd** ユーティリティを含む）が含まれています。まだインストールされていない場合もあります。

まず、次のコマンドでインストール済みかチェックします。

```
$ which firewalld firewall-cmd
```

```
/usr/sbin/firewalld
/usr/bin/firewall-cmd
```

プログラムが見つからなかったら、ディストリビューションに応じた以下のいずれかの方法でインストールします。

```
$ sudo dnf install firewalld
$ sudo zypper install firewalld
$ sudo apt install firewalld
```

インストールできない場合には **firewalld** パッケージがディストリビューションに含まれていません。ソースからインストールする必要があります。

ソースからインストールするために https://fedorahosted.org/firewalld/ に行き、**git** ソースのレポジトリから入手するか、最新の tar ボールををダウンロードします。

以下の手順でソースからインストールします。（最新バージョンを利用してください）

```
$ tar xvf firewalld-0.3.13.tar.bz2
$ cd firewalld-0.3.13
$ ./configure
$ make
$ sudo make install
```

このソースには **アンインストール** 機能も含まれています: 不要になった場合は、

```
$ sudo make uninstall
```

`./configure` を実行してライブラリなどで不足するものを見つけたら、パッケージからのインストールの場合はディストリビューションが対応してくれますが、ソースからのインストールの場合は少々面倒です。**Linux Foundation** の **ready-for.sh** スクリプトを実行すれば、ほとんど問題はおきません。

課題 30.2: firewall-cmd を使う

firewalld の使用方法の表面的なことだけを見てみます。わかりやすいオプションを持ち、さまざまなタスクを実行してくれる **firewall-cmd** を実行することで大半のことが行えます。

これを確認するためには以下を実行します。

```
$ firewall-cmd --help
```

```
Usage: firewall-cmd [OPTIONS...]
....
Service Options
  --new-service=<service>
                        Add a new service [P only]
  --delete-service=<service>
                        Delete and existing service [P only]
....
```

RHEL 8 システムでは 409 行もあるので省略します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

もっと詳細な説明に興味がある場合は、**man firewall-cmd** を実行すれば、概要、/etcにある構成ファイル、**zones** と **services** に関する **man** ページを見つけることができます。

課題 30.3: ゾーンにサービスを追加する

パブリックゾーンに **http** と **https** サービスを追加し、正しく表示されるか確認します。

解 30.3

```
$ sudo firewall-cmd  --zone=public --add-service=http
```

```
success
```

```
$ sudo firewall-cmd  --zone=public --add-service=https
```

```
success
```

```
$ sudo firewall-cmd --list-services --zone=public
```

```
dhcpv6-client http https ssh
```

注意：以下を実行していれば、

```
$ sudo firewall-cmd --reload
$ sudo firewall-cmd --list-services --zone=public
```

```
dhcpv6-client ssh
```

新たに追加したサービスがリストには表示されていないです！　ちょっと奇妙ですが、これは新しいサービスを追加した時に --permanent フラグを指定しなかったためです。--reload オプションは恒久化されたサービスだけを表示するものだからです。

課題 30.4: firewall GUI を使う

各ディストリビューションは、それぞれのファイアウォール用にグラフィカルなインターフェイスを持っています。**Red Hat** ベースのシステムでは **firewall-config**、**Ubuntu** では **gufw**、そして **openSUSE** では **yast** に組み込まれています。

ディストリビューションに依存しないように、コマンドラインについて説明しました。しかし、比較的容易なファイアウォールを構成する場合は、GUI を利用することで効率的に進めることができるでしょう。

ファイアウォールの GUI 構成ツールを使って、**パブリック** ゾーンに **http** と **https** を追加してみてください。その効果を実感してください。

GUI の理解にも時間を割いてください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 30.5: 自分の実際の IP アドレスを特定する

この演習では、**NAT** の最も一般的な使い方として、**ファイアウォール**の背後にあるデバイスの 内部アドレスまたは プライベートアドレスを隠蔽する方法を説明します。

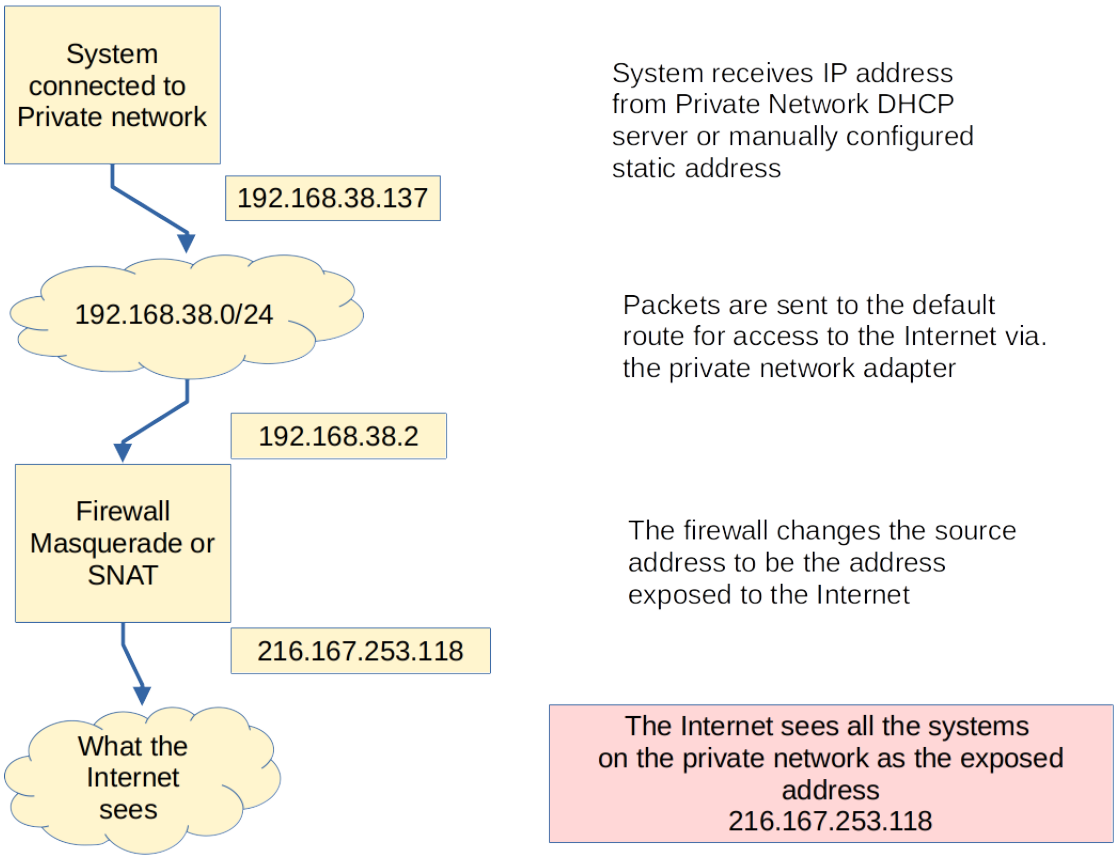


図 30.3: **SNAT ないしマスカレードの例**

上図を参考に、システムで使用されている IP アドレスを決定してください。

1. Linux VM 上でネットワーク アドレスを取得します。**ip** コマンドで取得できます。そして必要のない余分な情報は以下に示したように表示しないようにできます。

   ```
   $ sudo ip a | grep " inet "
   ```

   ```
   inet 127.0.0.1/8 scope host lo
   inet 192.168.38.137/24 brd 192.168.38.255 scope global dynamic noprefixroute ens33
   ^^I
   ```

   表示されるアドレスは、**ループバック**と仮想マシンの IP アドレスです。

2. VM のアドレスは **192.168.24.0/24** のサブネットにあり、**インターネット**上には流せない **ルート不可能** なアドレスとなります。**プライベート インターネットのアドレス割り当て**に関する追加情報は、以下のリンクを参照してください。

   https://datatracker.ietf.org/doc/html/rfc1918

   クライアントを次のホップに接続するための IP アドレスを見つけるには **デフォルト ルート**を調べます。

   ```
   $ sudo ip r
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
default via 192.168.38.2 dev ens33 proto dhcp metric 100
^^I^^I
```

次のホップは **192.168.38.2** という IP アドレスを持つシステムであることがわかります。

3. **192.168.38.2** の実体はこの環境のファイアウォールなので、私たちがログインすることはできません。外部から私たちのシステムがどのように見えているか確認できます。

   コマンドラインから

   ```
   $ sudo lynx -dump http://whatismyip.akamai.com  | awk '{print $1}'
   ```

   ```
   216.167.253.118
   ```

   または、Goolge で **what is my ip address** を検索して、多くのグラフィカルなオプションから 1 つを選択します

   外部との通信に使用されるアドレスは、プライベート アドレスのどれでもなく、プライベート ネットワーク上のすべてのマシンが、外部からは同じ IP アドレスに見えていることに注意してください。
   プライベート IP アドレスを単一のアドレスに見せているのは **Source Network Translation（ソース ネットワーク 変換）**または **マスカレード**の利用例です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 31 章

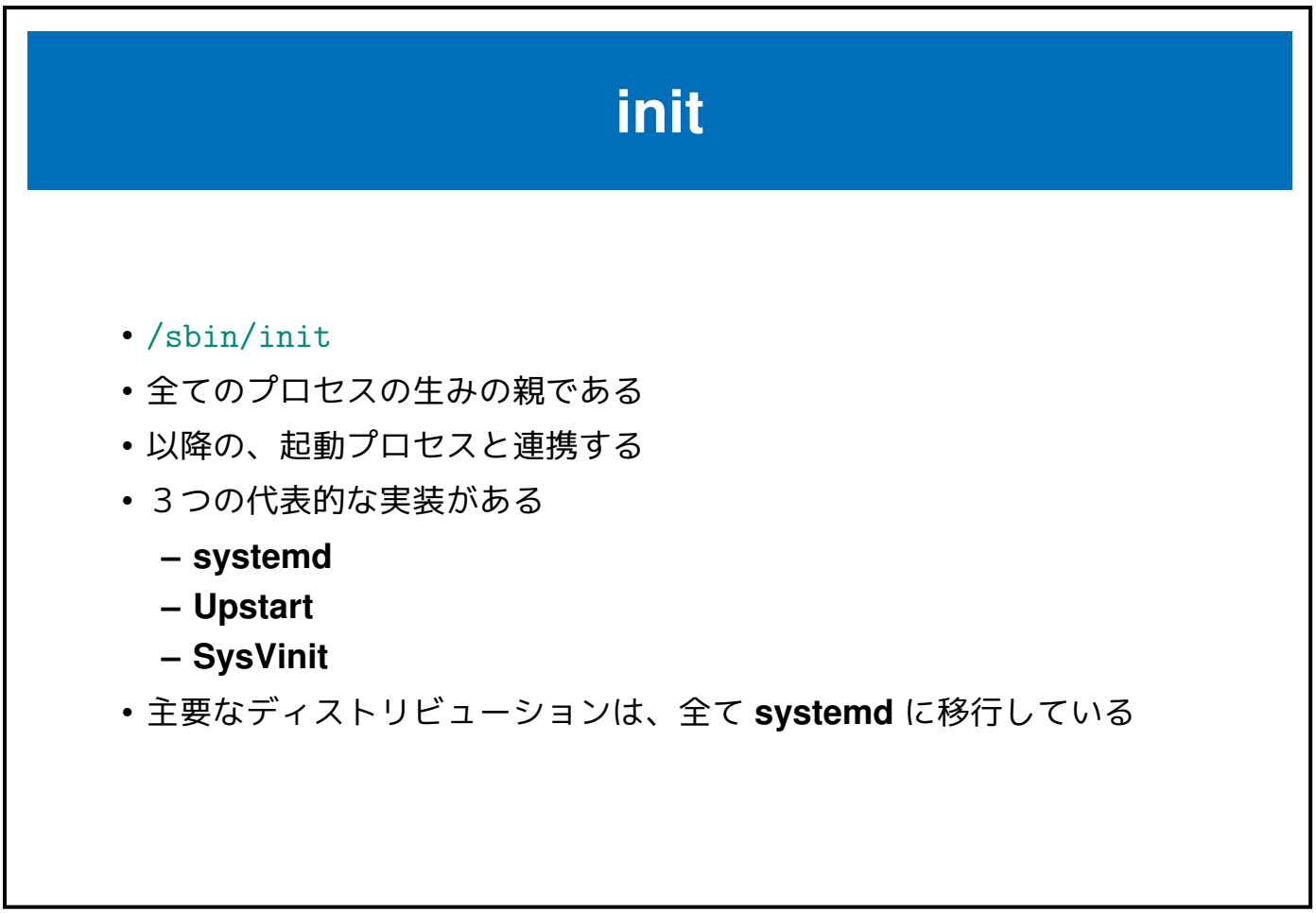
System の初期化: systemd の歴史とカスタマイズ

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- **SysVinit** と **Upstart** を置き換えるために **systemd** が誕生した経緯を理解する。
- **systemd** の主な機能を説明する。
- **systemctl** を使って、**systemd** でサービスの設定や制御を行う。
- ユーザーの **systemd units** と drop-in ファイルがどのように機能するかを見る。

31.1 init プロセス



`/sbin/init`（通常は、単に **init** と呼ばれます）は、システムで実行される最初のユーザーレベルのプロセス（またはタスク）で、システムがシャットダウンされるまで実行を続けます。歴史的には、全てのユーザープロセスの **親** と見なされてきましたが、一部のプロセスはカーネルから直接開始されるため、技術的に厳密に言えばこれは正しくありません。

init はブートプロセスの後段と連携し、環境のあらゆる面を設定し、システムにログインするために必要なプロセスを開始します。また、**init** はカーネルと密接に連携して、プロセスが終了したときの後始末を行います。

歴史的には、ほぼ全てのディストリビューションが **UNIX** の由緒ある **SysVinit** の **init** プロセスをベースにしていました。しかしながら **SysVinit** は、数十年前に、今とはかなり異なる状況下で開発されたものでした。

- マルチユーザー メインフレームシステムがターゲットでした。（PC、ラップトップなどのデバイスではありませんでした）
- シングルプロセッサ システムがターゲットでした。
- そのため起動（およびシャットダウン）時間は重要ではありませんでした。物事を正しく動作させる方がはるかに重要でした。

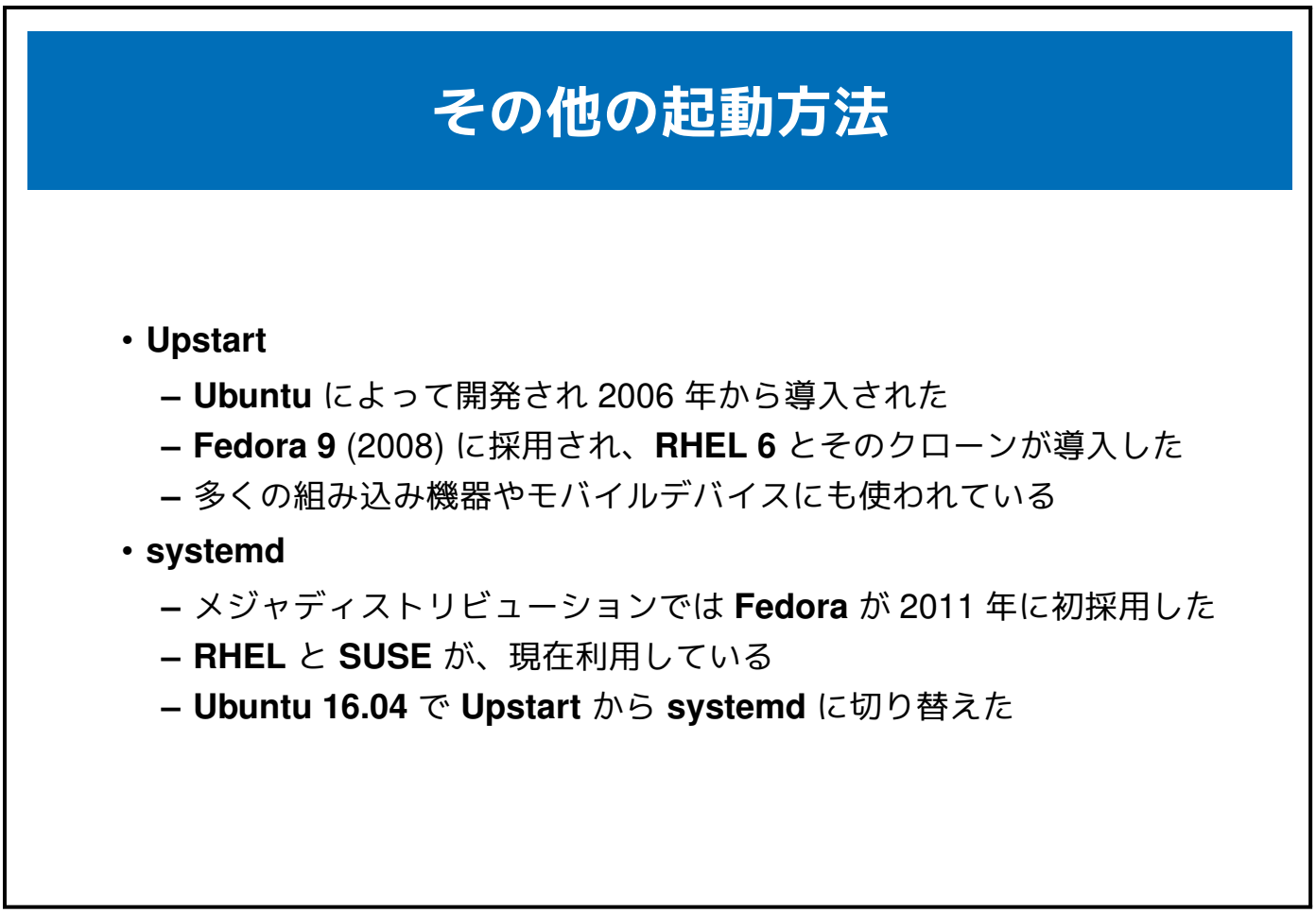
起動は、一連のシーケンシャルなステージに分割された **連続プロセス** と見なされます。次のステージに進むためには、前のステージを完結させる必要があります。このため、起動処理に複数プロセッサまたはコアによる並列処理を活用するのは簡単ではありませんでした。

さらに、シャットダウン／再起動 は比較的まれなイベントと見なされており、正確にどのくらい時間がかかるかは重要ではありませんでした。

最新のシステムには、機能を強化した新しいメソッドが求められました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

31.2 その他の起動方法



SysVinit 固有の制約に対処するため、新しいシステム起動制御メカニズムが開発されました。他にもありますが、エンタープライズディストリビュータは **Upstart** と **systemd** という 2 つの主要なスキームを採用しました。

RHEL が **systemd** を採用し、今では他の主要な **Linux** ディストリビューションも全てそれを採用し、デフォルト（の起動方法）として設定しています。主要な **systemd** の開発者達は **Linux** カーネル開発コミュニティと密接に連携しています。**Upstart** を選択した **Ubuntu** だけが例外でした。

systemd への移行はかなり複雑で、バグや欠落している機能が無効にされる可能性もあるため、レガシーソフトウェアとの基本互換レイヤーが導入され、これは今でも残っています。このため、中身は全く別物なのですが **SysVinit** ユーティリティとメソッドは現在まで存続しています。

systemd の開発と採用の歴史は一筋縄ではありませんでした。開発者のさまざまな個性がぶつかったので、必ずしも全ての議論が純粋に技術的な雰囲気の中で進められたわけではありませんでした。

以降の章では **systemd** に説明を絞ります。**SysVinit** と **Upstart** ついては殆ど触れません。（それらは、もはや重要な部分では使われていません）

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

31.3 systemd

systemd の機能

- 以前の init ベースのシステムより、起動が高速である
- 積極的に並列実行を活用する
- サービスの起動に **socket** と **D-Bus** アクティベーションを利用する
- 従来のシェルスクリプトを、プログラムに置き換えている
- デーモンのオンデマンド起動機能を提供している
- **cgroups** を使用して、プロセスを追跡する
- マウントと自動マウントのポイントを保持する
- 精巧なトランザクション依存関係に基づく、サービス 制御 ロジックを実装している
- **cron** と同様な機能を含む、詳細な開始と停止の機能を提供する

systemd と **Linux** のセッション マネージャーは、今やほとんどの主要なディストリビューションが採用しています。
bash スクリプトの代わりに **systemd** は `.service` ファイルを使用します。
きめ細かなロギングは、デフォルトでは **ユニット** 名を用いて提供されます。
root 権限を使わずにユーザープロセスを作成・追跡することも **systemd** の強力な機能です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

systemd の構成ファイルの場所

systemd は **unit** ファイルを複数の場所に配置可能で、root 以外のユーザーとして実行したり、システム **ユニット** ファイルから継承したりできる。
以下に典型的な場所を示す。
（ディストリビューションによって異なる）

システム ユニット ファイル

- /etc/systemd/system
- /lib/systemd/system

ユーザー ユニット ファイル

- /etc/systemd/user
- ~/.config/systemd/user

注目

殆どのシステムでは /lib/systemd/ は /usr/lib/systemd/ へのシンボリック リンクである

Debian: の標準的な **システム** ユニット ファイルの例を示します。

/lib/systemd/system/dbus.service

```
[Unit]
Description=D-Bus System Message Bus
Documentation=man:dbus-daemon(1)
Requires=dbus.socket

[Service]
ExecStart=/usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
↪  --syslog-only
ExecReload=/usr/bin/dbus-send --print-reply --system --type=method_call --dest=org.freedesktop.DBus /
↪  org.freedesktop.DBus.ReloadConfig
OOMScoreAdjust=-900
```

ユニットファイルが非常に複雑になることがあります。この例では、ドキュメントへのリンクを含める方法や、サービスを開始したり再起動したりするための、さまざまな方法が紹介されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

31.4 systemctl

systemctl

```
$ systemctl [options] command [name]
```

- **systemd** がコントロールする全てのステータスを表示する

  ```
  $ systemctl
  ```

- 利用可能な全てのサービスを表示する

  ```
  $  systemctl list-units -t service --all
  ```

- 現在アクティブなサービスだけを表示する

  ```
  $ systemctl list-units -t service
  ```

- 一つ以上の **units** の起動（活性化）する

  ```
  $ sudo systemctl start foo
  $ sudo systemctl start foo.service
  $ sudo systemctl start /path/to/foo.service
  ```

  unit は、サービスでもソケットでも良い

同様な方法で、**ユーザー** ユニットも見ることができます。

```
$ systemctl list-units --user 'xdg*'
```

```
  UNIT                            LOAD   ACTIVE SUB     DESCRIPTION
  xdg-desktop-portal-gtk.service loaded active running Portal service (GTK+/GNOME implementation)
  xdg-desktop-portal.service     loaded active running Portal service
  xdg-document-portal.service    loaded active running flatpak document portal service
  xdg-permission-store.service   loaded active running sandboxed app permission store

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
4 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

注目

ユーザー ユニットも多数存在するので、上記のコマンドでは、デスクトップ Xorg セッションで *xdg* で始まるユニットのみに出力を限定しました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

31.5 演習

デモ教材ビデオ

using_systemctl_demo.mp4

課題 31.1: systemd を使って既存のサービスを変更する

systemd はビルトインサービスを完全に置き換えたり無効にしたりするだけでなく、`.override` ファイルや `.config` ディレクトリを経由して **ドロップイン** と呼ばれる既存のサービスの特定のパラメータだけを変更することもできます。この方法を使うと、パッケージのアップグレードによって変更が戻されないというメリットがあります。

最初のコンソール (tty1) でログイン (getty) を担当する **ユニット** の Description を変更し、それを元に戻します。

ここでは、グラフィカル・インターフェースまたは ssh 経由でワークステーションにログインしていると仮定します。安全のため、コンソール ターミナルからログインしていないことを確認してください。

```
$ tty
```

重要

このコマンドの結果が `/dev/tty1` と表示されたら、この演習はスキップした方が良いでしょう。

1. はじめに、getty@tty1.service のステータスを確認します。

```
$ systemctl status getty@tty1.service
```

```
$ systemctl status getty@tty1.service
* getty@tty1.service - Getty on tty1
     Loaded: loaded (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-03-09 12:58:42 EST; 2 months 7 days ago
       Docs: man:agetty(8)
             man:systemd-getty-generator(8)
             http://0pointer.de/blog/projects/serial-console.html
   Main PID: 621 (agetty)
      Tasks: 1 (limit: 4656)
     Memory: 200.0K
        CPU: 2ms
     CGroup: /system.slice/system-getty.slice/getty@tty1.service
             `-621 /sbin/agetty -o -p -- \u --noclear tty1 linux

Warning: some journal files were not opened due to insufficient permissions.
```

2. 次に、サービスの説明を変更してみましょう。**Description=** の行をコピーして、記述内容を変更してみてください。この例では、動的出力を **最初のターミナル** に変更しています。

```
$ sudo systemctl edit getty@tty1.service
```

```
### Editing /etc/systemd/system/getty@tty1.service.d/override.conf
### Anything between here and the comment below will become the new contents of the file

[Unit]
Description=Getty on the first terminal

### Lines below this comment will be discarded

### /lib/systemd/system/getty@.service
# #  SPDX-License-Identifier: LGPL-2.1-or-later
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
# #
# #  This file is part of systemd.
# #
# #  systemd is free software; you can redistribute it and/or modify it
# #  under the terms of the GNU Lesser General Public License as published by
# #  the Free Software Foundation; either version 2.1 of the License, or
# #  (at your option) any later version.
```

3. 次に /etc/systemd/system/getty@tty1.service.d/override.conf の内容を確認します。
ここに今行った変更が表示されしているはずです。- 2 行だけで、最後の改行はありません。

```
$ cat /etc/systemd/system/getty@tty1.service.d/override.conf
```

```
[Unit]
Description=Getty on the first terminal<YOUR PROMPT HERE>
```

4. デーモンをリロードして変更を取り込み、結果をチェック確認します。

```
$ sudo systemctl daemon-reload
$ systemctl status getty@tty1.service
```

```
* getty@tty1.service - Getty on the first terminal
     Loaded: loaded (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/getty@tty1.service.d
             `-override.conf
     Active: active (running) since Thu 2023-03-09 12:58:42 EST; 2 months 7 days ago
       Docs: man:agetty(8)
             man:systemd-getty-generator(8)
             http://0pointer.de/blog/projects/serial-console.html
   Main PID: 621 (agetty)
      Tasks: 1 (limit: 4656)
     Memory: 200.0K
        CPU: 2ms
     CGroup: /system.slice/system-getty.slice/getty@tty1.service
             `-621 /sbin/agetty -o -p -- \u --noclear tty1 linux

Warning: some journal files were not opened due to insufficient permissions.
```

5. 最後に、作成したファイルを削除して元の通常の状態に戻します。

```
$ sudo rm  /etc/systemd/system/getty@tty1.service.d/override.conf
$ systemctl status getty@tty1.service
```

```
* getty@tty1.service - Getty on tty1
     Loaded: loaded (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-03-09 12:58:42 EST; 2 months 7 days ago
       Docs: man:agetty(8)
             man:systemd-getty-generator(8)
             http://0pointer.de/blog/projects/serial-console.html
   Main PID: 621 (agetty)
      Tasks: 1 (limit: 4656)
     Memory: 200.0K
        CPU: 2ms
     CGroup: /system.slice/system-getty.slice/getty@tty1.service
             `-621 /sbin/agetty -o -p -- \u --noclear tty1 linux

Warning: some journal files were not opened due to insufficient permissions.
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 32 章

バックアップとリカバリーの方法

学習目標

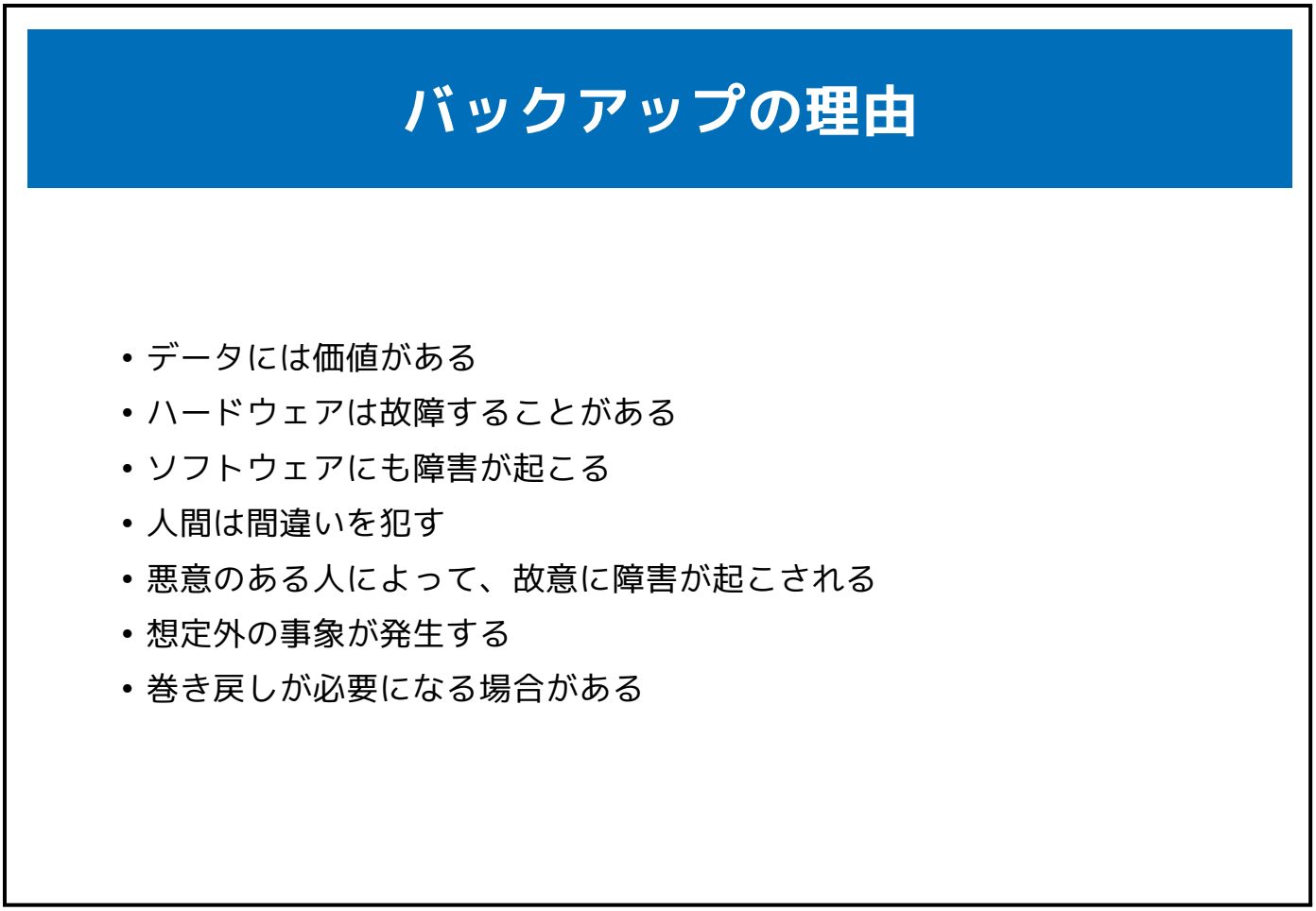
このセッションが終わるころには、次のことができるようになっているはずです

- バックアップが必要なデータを特定し、優先順位をつける。
- データのバックアップとアーカイブの違いを理解する。
- 状況に応じて、さまざまな種類のバックアップとリストアの方法を効率的に採用する。
- **xz** や **gzip** など様々な圧縮方式を使い、アーカイブプログラムである **tar** と効率よく組み合わせる。
- 生データのコピーには **dd** を、複数マシンの効率的なバックアップや同期には **rsync** を使用する。
- 最も有名ないくつかのバックアッププログラムについて説明する。

注目

** これらのセクションの一部または全体をオプション扱いとする場合があります。これらには補足資料、専門トピック、または高度な話題が含まれインストラクターが教室の状況や時間制約に応じてこれらの内容を紹介するかを判断します。

32.1 バックアップの基本



１つのパーソナルシステムだけを管理している場合でも、多数のマシンのネットワークを管理している場合でも、システムバックアップは非常に重要です。

ディスク上のデータは、重要な作業成果物であるため、保護すべきです。失われたデータを再度作成するのは、時間と費用がかかります。一部のデータは、他にない唯一のものであり、再度作成する方法がない場合があります。

ストレージメディアの信頼性は向上していますが、ドライブ容量も増大しています。１バイトあたりの故障率は低下していますが、それでも予測できない故障は発生します。ドライブには「故障したドライブ」と「故障する可能性のあるドライブ」の２種類しかない、という言い方は悲観的かもしれませんが真実です。**RAID** はデータ保護には役立ちますが、それでもバックアップは必要です。

完璧なソフトウェアはありません。バグによってデータが破壊または破損される可能性があります。長期間使用している安定したプログラムにも、問題が発生する場合があります。

誰もが、隣のパーティションから（または、自分の口から）OOPS！(しまった !)、または、もっとひどい言葉をを聞いたことがあるはずです。時には単純な入力ミスだけで、ファイルとデータが大規模に破壊されることがあります。

それは、普通に不満を抱く従業員か、もしくは損害を起こすポイントを知っている外部のハッカーかもしれません。セキュリティの問題とバックアップ機能は非常に強く関連しています。

ファイルは、どのように、誰が、いつ起こったかを知らなくても、ただ消えてしまう可能性があります。

場合によっては、システムの全て、または一部を以前の **スナップショット** に復元することが要求されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

バックアップが必要なものは何か？

- 絶対に必要なもの
  - 業務に関連したデータ
  - システム設定ファイル
  - ユーザーのファイル (通常は /home 以下のファイル).
- 多分必要なもの
  - スプーラーディレクトリ (印刷、メールなど)
  - ログファイル (/var/log やそれ以外の場所にある).
- おそらく必要がないもの
  - 簡単に再インストールできるソフトウェア、適切に管理されたシステムでは大部分がこの範疇に該当するだろう
  - /tmp ディレクトリ
- 絶対に必要ないもの
  - /proc、/dev、/sys や swap など疑似ファイルシステム

組織に不可欠なファイルのバックアップが必要なのは明らかです。頻繁に変更される構成ファイルも、各個人ユーザーのファイルと共にバックアップする必要があります。

システムの履歴を調査する必要がある場合、ログファイル（の保全）は重要です。侵入やその他のセキュリティ違反を検出するためには、ログファイルは特に重要です。

簡単に再インストールできるものをバックアップする必要はありません。また、スワップ パーティション（またはファイル）と /proc ファイルシステムは（ /tmp ディレクトリと同様に）、データが基本的に一時的なものなのでバックアップの必要がないか、バックアップしても無意味です。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.2 バックアップ vs. アーカイブ

バックアップ vs. アーカイブ

- 全てのバックアップメディアに、読み取り不能となる寿命の限界がある
- 標準的な想定寿命は:
  - 磁気テープ: 10-30 年
  - CD や DVD: 3-10 年
  - ハードディスク: 2-5 年
- 実際の寿命は以下に大きな影響を受ける
  - 環境条件（温度、湿度など）
  - 記録メディアの品質
  - 現在のオペレーティング システムとハードウェアで、データを読み出せるソフトが存在するか
- バックアップには十分だが、恒久的なデジタルアーカイブには使えない

通常のバックアップ周期よりも長く保存させたい場合には、複数のコピーを作成したり時々新しいメディアにコピーすることで対応できます。

超長期間（つまり数十年、数世紀など）の保存は、以下のような標準的な保存手段が全て時代遅れになってしまうので簡単ではありません。

- ハードウェア
- ソフトウェアとドキュメントのフォーマット（書式）
- メディア

安価なデジタル記録メディアの寿命は、どれも（ワインのように、適切に保管され継続的に手入れされていたとしても）紙やフィルムほどは長くはありません。

これは、人々が深刻に懸念している問題で、全てが失われる前に良い解決策を考えるべきです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

テープ ドライブ

- 比較的低速である
- シーケンシャル アクセスしかできない
- 以前のようには、一般的ではなくなった
- 今日では、メインのバックアップには使われない
- それでも「オフサイトストレージ」や「アーカイブ」には有効である

テープドライブは、以前ほどは一般的ではなくなりました。比較的低速で順次（シーケンシャル）アクセスしかできません。最近は、プライマリバックアップにテープを使用することはほとんどありません。長期間参照できるような、アーカイブ目的のオフサイトストレージとしてテープが使用されることがあります。ただし、磁気テープドライブには物理的劣化やデータ損失がなくても、その寿命には限界があります。

最近のテープドライブはほとんどが **LTO** (**L**inear **T**ape **O**pen) で、1990 年代後半にオープンスタンダードの代替として最初のバージョンが登場しました。初期には、ほとんどが独自フォーマットのものでした。初期バージョンの記録容量は最大で 100GB でした。新しいバージョンでは、同じサイズのカートリッジで 2.5TB 以上を記録できます。

日々のバックアップには、通常は何らかの **NAS** (**N**etwork **A**ttached **S**torage)、または **クラウドベースのソリューション** が使われるため、新しいテープベースのインストールはますます魅力的ではなくなっています。それでもまだテープを使う場面もあるので、システム管理者は対処する必要があるかもしれません。

以降は、バックアップメディアの物理的な形式ではなく、抽象度の高い（記録メディアに依存しない）テーマを説明をします。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.3 バックアップ方法と戦略

バックアップ方法

- **完全バックアップ (Full)**:
  システム上の全ファイルをバックアップする
- **増分バックアップ (Incremental)**:
  最後に行った増分、または完全バックアップ以降に変更された全ファイルをバックアップする
- **差分バックアップ (Differential)**:
  最後の完全バックアップ以降に変更された全ファイルをバックアップ
- **複数のレベルの増分バックアップ (Multiple level incremental)**:
  同じレベル、または以前のレベルにおいて、最後に取られたバックアップ以降に変更された全ファイルをバックアップする
- **ユーザー (User)**:
  特定のユーザーのディレクトリ内のファイルだけをバックアップする

多くの場合、複数の異なるバックアップ方法を互いに連携させて利用できます。

全てのバックアップは、保護対象システムと同じ物理的な場所に保管しないでください。そうでないと、火災またはその他の物理的損傷により全てが失われる可能性があるからです。過去には「バックアップとは磁気テープを安全な場所に物理的に輸送すること」を意味していました。現在は「バックアップファイルをインターネット経由で別の物理的な場所に転送すること」を意味します。適切な暗号化とその他のセキュリティ予防措置を使用して、安全な方法でデータを転送する必要があるのは明らかです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

バックアップ戦略

テープを使った有用な戦略（「テープ」を他のメディアに置き換えられる）

1. 金曜日の完全バックアップにはテープ１を使用
2. 月曜日から木曜日の増分バックアップにはテープ２～５を使用
3. 第 2 金曜日の完全バックアップにはテープ６を使用
4. 2 番目の月曜日から木曜日の増分バックアップにはテープ２～５を使用
5. テープ６の完全バックアップが完了するまでテープ１を上書きしない
6. テープ６に完全バックアップした後、災害復旧時のためにテープ１を外部の場所に移動
7. 次の完全バックアップ（次の金曜日）にはテープ６の代わりにテープ１を使用

バックアップ方法は、対応する **レストア (復元)** 方法がないと役に立たないことに注意してください。バックアップの戦略を選択する際には、両方向（つまり、バックアップとレストア）の堅牢性、明瞭性、および使いやすさを考慮する必要があります。

目安として、少なくとも２週間のバックアップを用意するようにしてください。

最も単純なバックアップ戦略は、全てを一度完全バックアップしてから、その後変更される全ての増分バックアップを実行することです。完全バックアップには多くの時間がかかる場合がありますが、増分バックアップからの復元はもっと困難で、時間がかかる場合があります。したがって、両方を組み合わせれば時間と労力を最適化できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

バックアップ関連ユーティリティ

- **cpio**
- **tar**
- **gzip, bzip2, xz**
- **dd**
- **rsync**
- **dump/restore**
- **mt**

cpio と **tar** はファイルのアーカイブの作成と抽出を行います。

ほとんどの場合、アーカイブは **gzip**、**bzip2**、**xz** で圧縮されています。アーカイブファイルはディスク、磁気テープ、またはファイルを保持できる他のデバイスに書き込むことができます。アーカイブは、ひとつのファイルシステムやマシンから別のファイルシステムやマシンにファイルを転送するのに非常に便利です。

dd は強力なユーティリティで、メディア間で生データを転送する時によく使用します。パーティション全体、またはディスク全体をコピーできます。

rsync も強力なユーティリティでネットワーク全体で、またはローカルマシン上の異なるファイルシステムの間でディレクトリサブツリーまたはファイルシステム全体を同期できます。

dump と **restore** は古くからあるユーティリティで、バックアップ専用に設計されました。ファイルシステムから直接読み込みます。（それが、より効率的な方法です）ただし、元のファイルシステムと同じ種類のファイルシステムに復元しなければなりません。今はもっと新しい選択肢があります。

mt はバックアップと復元を実行する前に、テープの状態を問合せをして位置調整するのに役立ちます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.4 tar

tar を使用したバックアップ

- アーカイブの作成: -c または単に c を利用する

```
$ tar cvf  /dev/st0 /root
$ tar -cvf /dev/st0 /root
```

- マルチボリュームオプションで作成: -M を利用する

```
$ tar -cMf /dev/st0 /root
```

テープを挿入するよう促される

- 比較オプションでファイルを検証: -d または --compare を利用する

```
$ tar --compare --verbose --file /dev/st0
$ tar -dvf /dev/st0
```

- 各オプションには短縮形 (- と一文字) と、長い形式 (-- で始まる) がある
- **tar** のデフォルトは再帰的である

tar アーカイブを作成する場合、引数としてディレクトリを指定すると、そのディレクトリの全てのファイルとサブディレクトリがアーカイブに含まれます。復元するときに、必要に応じてディレクトリを再構成します。

さらに、増分バックアップを実行できる --newer オプションもあります

Linux で使用される **tar** のバージョンは、１本のテープまたは使用するデバイスに納まらない量のバックアップも処理できます。

-f または --file オプションを付加してデバイスやファイルを指定できます。

バックアップ作成後、検証オプションを使用して完全かつ正確であることを確認できます。

デフォルトでは **tar** は、全てのサブディレクトリを再帰的にアーカイブ内に格納します。

アーカイブを作成すると **tar** は絶対パス名から先頭のスラッシュを削除する旨のメッセージを出力します。これにより、ファイルをどこにでも復元できますが、デフォルトのこの挙動を変更することもできます。

ほとんどの **tar** オプションは、ダッシュを１つ付けた短い形式、または２つ付けた長い形式で指定できます。-c は --create と完全に同じです。

また（短い表記を使用する場合）、オプションを組み合わせることができますが、その時には全てのダッシュを入力する必要がないことに注意してください。

さらにシングルダッシュの **tar** オプションは、ダッシュ付きでもダッシュなしでも使用できます。
例えば、tar cvf file.tar dir1 と tar -cvf file.tar dir1 は同じ結果になります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

tar を使用したファイルの復元

- アーカイブからの復元: -x 、または --extract を利用する

```
$ tar --extract --same-permissions --verbose --file /dev/st0
$ tar -xpvf /dev/st0
$ tar  xpvf /dev/st0
```

- 復元するファイル名の指定ができる

```
$ tar -xvf /dev/st0 somefile
```

- tar バックアップのコンテンツをリストする

```
$ tar --list --file /dev/st0
$ tar -tf /dev/st0
```

-x 、または --extract オプションは、デフォルトでアーカイブからファイルを抽出します。特定のファイルだけを指定すると、抽出するファイルを絞り込むことができます。ディレクトリを指定すると、そこに含まれる全てのファイルとサブディレクトリも抽出されます。

-p または --same-permissions オプションは、ファイルを元の権限で復元します。

-t または --list オプションは、アーカイブ内のファイルをリストしますが抽出はしません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

tar を使用した増分バックアップ

- `--newer` または `--after-date` オプションを利用する
- 日付（だけ）をファイル抽出の基準にする
- 指定した日付以降に変更されたファイルをバックアップし圧縮する

```
$ tar --create --newer '2011-12-1' -vzf backup1.tgz /var/tmp
$ tar --create --after-date '2011-12-1' -vzf backup1.tgz /var/tmp
```

どちらも、2011 年 12 月 1 日以降に変更された `/var/tmp` 内の全てのファイルのバックアップアーカイブを作成する

`-N` (もしくは同じ意味の `--newer`) 、または、`--after-date` オプションを使用すれば **tar** で増分バックアップを実行できます。どのオプションでも、日付または対象（基準）のファイル名のいずれかを指定する必要があります。

tar は、ファイルの日付だけを調べパーミッションやファイル名などファイルに対する他の変更は考慮しません。これらが変更されたファイルを増分バックアップに含めるには、**find** コマンドで対象となるファイルを検索してバックアップするファイルのリストを作成します。

Note: `--newer` などのオプションが後に続く場合、`-vzf` などのオプションにはダッシュを使用する必要があります。そうしない場合 **tar** が混乱します。この種のオプション指定の混乱は、**ps** や **tar** のような古い **UNIX** ユーティリティで発生することがあります。そこには、さまざまな **UNIX** の系統が関係する複雑な歴史があります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.5 圧縮: gzip、bzip2、xz とバックアップ

アーカイブの圧縮方法

- ファイル **圧縮** はディスクスペース／ネットワーク転送時間を節約する
- 圧縮率を高めて効率を稼ごうとすると、圧縮時間は余計にかかる
  - **gzip**: Lempel-Ziv コーディング（LZ77）を使用し .gz ファイルを生成する
  - **bzip2**: Burrows-Wheeler ブロックソートテキスト圧縮アルゴリズムと Huffman コーディングを使用し .bz2 ファイルを生成する
  - **xz**: .xz ファイルを生成、従来の .lzma 形式もサポートする
- 現代のマシンではしばしば **圧縮** → **転送** → **解凍** した方が、単純に非圧縮のファイルを転送（またはコピー）するより時間が短くなる
- https://www.kernel.org からの **Linux** カーネルのダウンロードには今は **xz** 形式だけが使われる

圧縮ユーティリティは、非常に簡単に（そして頻繁に）**tar** と組み合わせて使用されます。

```
$ tar zcvf source.tar.gz  source
$ tar jcvf source.tar.bz2 source
$ tar Jcvf source.tar.xz  source
```

これらは圧縮アーカイブを作成します。最初のコマンドは、以下のコマンドとまったく同じ結果になります。

```
$ tar cvf source.tar source ; gzip -v source.tar
```

このコマンドは、中間ファイルストレージを使わないのでより効率的です。アーカイブと圧縮は並列にパイプライン処理されます。
解凍する場合

```
$ tar xzvf source.tar.gz
$ tar xjvf source.tar.bz2
$ tar xJvf source.tar.xz
```

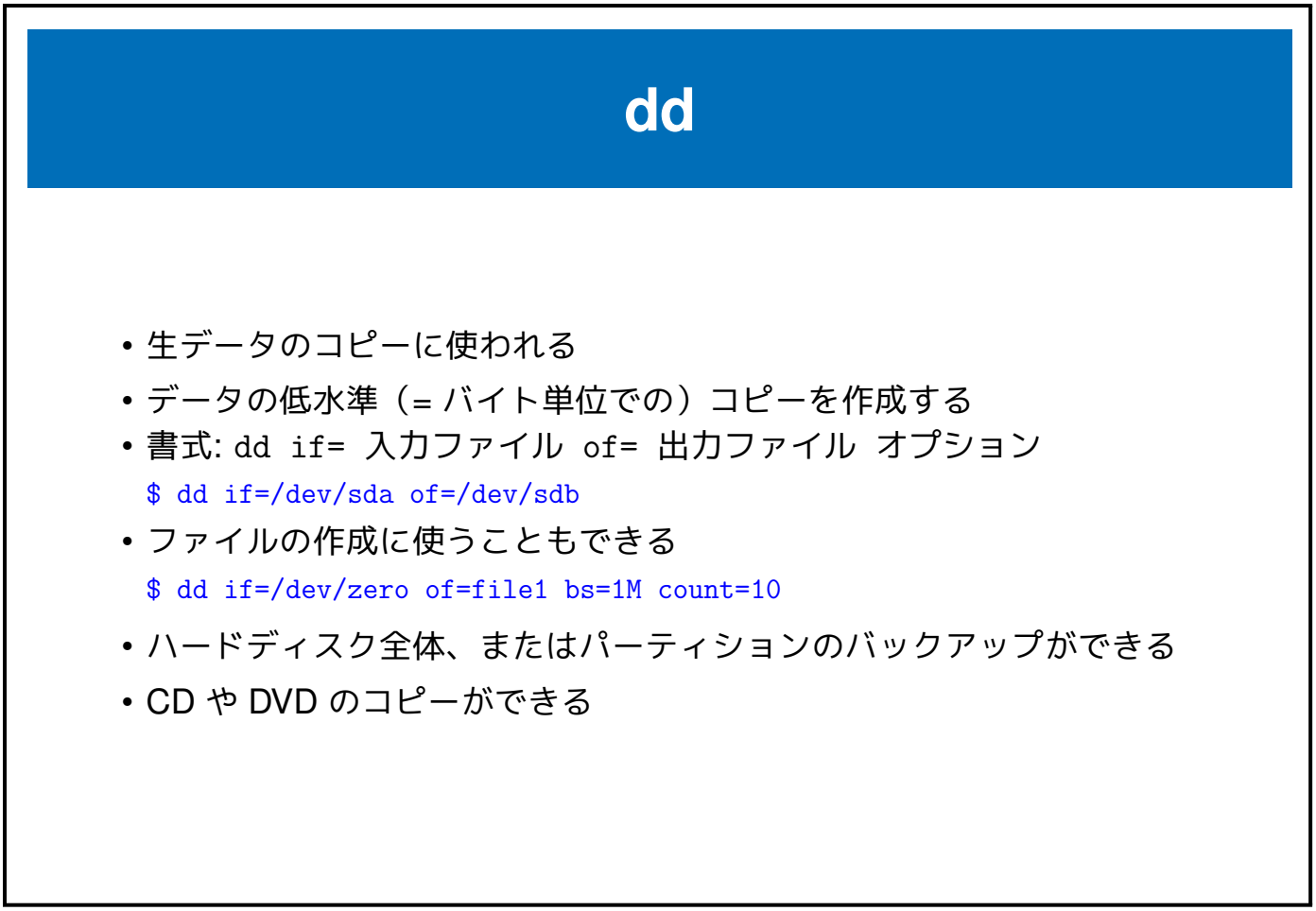
または単純に

```
$ tar xvf source.tar.gz
```

tar の最新バージョンは、圧縮の方法を検出して自動的に処理します。
.jpg 画像や .pdf ファイルなど、既に圧縮されているアーカイブを更に圧縮する価値は無いのは明らかです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.6 dd



dd は一般的な **UNIX** ベースのプログラムで、主な目的は生データの変換や低レベルのコピーや生データの変換です。**dd** は、指定した数のバイトやブロックをコピー、バイトオーダー変換（＝エンディアン変換）の逐次実行、さらにデータの変換にも使われます。さらにデバイスファイル領域へのコピーにも使われます。例えば、ハードディスクのブートセクターのバックアップ、もしくは `/dev/zero` や `/dev/random` のようなスペシャルファイルから決まった量のデータを読み出すのにも使われます。

ハードドライブ全体を、別のディスクにバックアップします。

```
$ dd if=/dev/sda of=/dev/sdb
```

ハードディスクイメージの作成します。

```
$ dd if=/dev/sda of=sdadisk.img
```

パーティションをバックアップします。

```
$ dd if=/dev/sda1 of=partition1.img
```

CD-ROM をバックアップします。

```
$ dd if=/dev/cdrom of=tgsservice.iso bs=2048
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.7 rsync

rsync を使ったバックアップ

- 使い方

```
$ rsync [options] sourcefile destinationfile
```

- ローカルマシンとネットワークターゲット間でバックアップするには

```
$ rsync file.tar someone@backup.mydomain:/usr/local
```

- ネットワーク（上の）２台のマシンの間でバックアップするには

```
$ rsync -r amachine:/usr/local bmachine:/usr/
```

- 本番前に --dry-run でテスト実行する

```
$ rsync -r --dry-run /usr/local /BACKUP/usr
```

rsync (**r**emote **sync**hronize) （リモート同期）は、次のようにネットワークを介して（または同じマシンの異なる場所の間で）ファイルを転送するために使用されます。

同期元（ソース）と 同期先（デスティネーション）は、target:path の形式を取ることができます。ここで target は [user@]host の形式にすることができます。user@ の部分はオプションであり、リモートユーザーがローカルユーザーと異なる場合に使用されます。したがって、以下の **rsync** コマンドは、全て実行可能です:

```
$ rsync file.tar someone@backup.mydomain:/usr/local
$ rsync -r a-machine:/usr/local b-machine:/usr/
$ rsync -r --dry-run /usr/local /BACKUP/usr
```

rsync を使う時には（特に --delete オプションを使用する場合には）、場所の正確な指定について、特に注意する必要があります。そのため、最初は --dry-run オプションを使用し、期待通りの動作になっていることを確認してから本番を実行することを強くお勧めします。

rsync は非常に賢いです。ローカルファイルとリモートファイルを小さなチャンクで照合します。１つのディレクトリを類似のディレクトリにコピーする場合、差分だけがネットワーク経由でコピーされるので非常に効率的です。こうすることで、最初のディレクトリと二番目のディレクトリが同期されます。多くの場合 **rsync** に -r オプションを併用し、ディレクトリツリーを再帰的にたどりソースファイル としてリストされているディレクトリから下の、全てのファイルとディレクトリをコピーします。このため、以下のようなプロジェクトディレクトリのバックアップに非常に適しています。

```
$ rsync -r project-X archive-machine:archives/project-X
```

明快な（そして、非常に効果的で高速な）バックアップ戦略は、**rsync** コマンドを使用してネットワーク全体でディレクトリ、または、パーティションを単純に複製し、そして、頻繁に更新することです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.8 バックアッププログラム **

独自のアプリケーションやストレージベンダーが提供するアプリケーション、オープンソースアプリケーションなど **Linux** で利用可能なバックアッププログラムスイートがいろいろあります。特に良く知られているものには

- **Amanda** (**A**dvanced **M**aryland **A**utomatic **N**etwork **D**isk **A**rchiver) は、ネイティブユーティリティ（**tar** と **dump** を含む）を使用しますが、はるかに堅牢で制御しやすいです。

  Amanda は、一般的に標準リポジトリから入手してエンタープライズ **Linux** システムで利用できます。詳細な情報は http://www.amanda.org で確認できます。

- **Bacula** は、異種ネットワーク上の自動バックアップ用に設計されています。かなり複雑なので、経験豊富な管理者だけが使用することを（作成者は）推奨しています。

  Bacula は、通常標準リポジトリを通じてエンタープライズ **Linux** システムで利用できます。詳細な情報は https://www.bacula.org/7.0.x-manuals/en/main/Main_Reference.html で確認できます。

- **Clonezilla** は、非常に堅牢な **ディスククローンプログラム** です。ディスクのイメージを作成して展開しバックアップを復元したり、**ゴースト化（＝クローン作成）**に利用して多くのマシンのインストールに使用できるイメージを提供できます。詳細な情報は https://clonezilla.org で確認できます。

  プログラムには２つのバージョンがあります。単一マシンのバックアップとリカバリに適した **Clonezilla Live** と、同時に多くのコンピュータにクローンを作成できるサーバーエディションの **Clonezilla SE, server edition** です。

  Clonezilla の使用はそれほど難しくなく非常に柔軟で、（**Linux** だけでなく）多くのオペレーティング システム、ファイルシステムの種類、ブートローダーをサポートしています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

32.9 演習

課題 32.1: tar を使ったバックアップ

1. backup ディレクトリを作成、その下に /usr/include 下の全ファイルを圧縮し、上位ディレクトリの include を含む **tar** アーカイブを作成します。圧縮形式には **gzip**、**bzip2** または **xzip** のどれを使っても構いません。
2. アーカイブ中のファイルをリストします。
3. restore というディレクトリを作成し、その下にアーカイブを解凍します。
4. オリジナルのディレクトリ下の内容とアーカイブを解凍した内容を比較します。

解 32.1

1. ```
   $ mkdir /tmp/backup
   $ cd /usr ; tar zcvf /tmp/backup/include.tar.gz include
   $ cd /usr ; tar jcvf /tmp/backup/include.tar.bz2 include
   $ cd /usr ; tar Jcvf /tmp/backup/include.tar.xz include
   ```

   または

   ```
   $ tar -C /usr -zcf include.tar.gz include
   $ tar -C /usr -jcf include.tar.bz2 include
   $ tar -C /usr -Jcf include.tar.xz include
   ```

   3 つの圧縮方式の効率を比較します。

   ```
   $ du -sh /usr/include
   ```

   ```
   55M   /usr/include
   ```

2. ```
   $ ls -lh include.tar.*
   ```

   ```
   c8:/tmp/backup>ls -lh
   total 17M
   -rw-rw-r-- 1 coop coop 5.3M Jul 18 08:17 include.tar.bz2
   -rw-rw-r-- 1 coop coop 6.7M Jul 18 08:16 include.tar.gz
   -rw-rw-r-- 1 coop coop 4.5M Jul 18 08:18 include.tar.xz
   ```

3. ```
   $ tar tvf include.tar.xz
   ```

   ```
   qdrwxr-xr-x root/root         0 2014-10-29 07:04 include/
   -rw-r--r-- root/root     42780 2014-08-26 12:24 include/unistd.h
   -rw-r--r-- root/root       957 2014-08-26 12:24 include/re_comp.h
   -rw-r--r-- root/root     22096 2014-08-26 12:24 include/regex.h
   -rw-r--r-- root/root      7154 2014-08-26 12:25 include/link.h
   .....
   ```

   解凍の際は j、J または z オプションは不要です: **tar** は自動的に判断します。

4. ```
   $ cd .. ; mkdir restore ; cd restore
   $ tar xvf ../backup/include.tar.bz2
   ```

   ```
   include/
   include/unistd.h
   include/re_comp.h
   include/regex.h
   include/link
   .....
   $ diff -qr include /usr/include
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 32.2: cpio を使ったバックアップ

先の演習と同じですが **tar** の代わりに **cpio** を利用します。内容は、若干の変更を行うことで容易に利用できるようにします。

1. backup ディレクトリを作成、その下に /usr/include 下の全ファイルを圧縮し、上位ディレクトリの include を含む **tar** アーカイブを作成します。圧縮形式には **gzip**、**bzip2** または **xzip** のどれを使っても構いません。
2. アーカイブ中のファイルをリストします。
3. restore というディレクトリを作成し、その下にアーカイブを解凍します。
4. オリジナルのディレクトリ下の内容とアーカイブを解凍した内容を比較します。

解 32.2

1. `$ (cd /usr ; find include | cpio -c -o > /home/student/backup/include.cpio)`

   ```
   82318 blocks
   ```

   または、圧縮形式で作成します:

   `$ (cd /usr ; find include | cpio -c -o | gzip -c > /home/student/backup/include.cpio.gz)`

   ```
   82318 blocks
   ```

   ```
   $ ls -lh include*
   total 64M
   -rw-rw-r-- 1 coop coop  41M Nov  3 15:26 include.cpio
   -rw-rw-r-- 1 coop coop 6.7M Nov  3 15:28 include.cpio.gz
   -rw-rw-r-- 1 coop coop 5.3M Nov  3 14:44 include.tar.bz2
   -rw-rw-r-- 1 coop coop 6.8M Nov  3 14:44 include.tar.gz
   -rw-rw-r-- 1 coop coop 4.7M Nov  3 14:46 include.tar.xz
   ```

2. `$ cpio -ivt < include.cpio`

   ```
   drwxr-xr-x  86 root     root            0 Oct 29 07:04 include
   -rw-r--r--   1 root     root        42780 Aug 26 12:24 include/unistd.h
   -rw-r--r--   1 root     root          957 Aug 26 12:24 include/re_comp.h
   -rw-r--r--   1 root     root        22096 Aug 26 12:24 include/regex.h
   .....
   ```

   入力がリダイレクションされていることに注意してください。引数としてアーカイブを指定しないやり方もあります。

   ```
   $ cd ../restore
   $ cat ../backup/include.cpio | cpio -ivt
   $ gunzip -c include.cpio.gz | cpio -ivt
   ```

3. ```
   $ rm -rf include
   $ cpio -id < ../backup/include.cpio
   $ ls -lR include
   ```

   または、

   ```
   $ cpio -idv < ../backup/include.cpio
   ```

   ```
   $ diff -qr include /usr/include
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
   ```

課題 32.3: rsync を使ったバックアップ

1. **rsync** を利用して /usr/include のコピーをバックアップディレクトリに作成します。

```
$ rm -rf include
$ rsync -av /usr/include .
```

```
sending incremental file list
include/
include/FlexLexer.h
include/_G_config.h
include/a.out.h
include/aio.h
.....
```

2. ２回目のコマンド実行の結果を見ます。

```
$ rsync -av /usr/include .
```

```
sending incremental file list

sent 127398 bytes  received 188 bytes  255172.00 bytes/sec
total size is 41239979  speedup is 323.23
```

3. **rsync** で混乱しやすいのは、以下のコマンド形式が誤りであることです。

```
$ rsync -av /usr/include include
```

```
sending incremental file list
...
```

このコマンドを実行すると、新しいディレクトリ include/include が作成されます！

4. 不要なファイルを削除するためには --delete オプションを用います。

```
$ rsync -av --delete /usr/include .
```

```
sending incremental file list
include/
deleting include/include/xen/privcmd.h
deleting include/include/xen/evtchn.h
....
deleting include/include/FlexLexer.h
deleting include/include/

sent 127401 bytes  received 191 bytes  85061.33 bytes/sec
total size is 41239979  speedup is 323.22
```

5. 他のシンプルな演習として、バックアップコピーからサブディレクトリを削除し --dry-run オプションをつけた場合と、つけない場合で **rsync** を実行します。

```
$ rm -rf include/xen
$ rsync -av --delete --dry-run /usr/include .
```

```
sending incremental file list
include/
include/xen/
include/xen/evtchn.h
include/xen/privcmd.h

sent 127412 bytes  received 202 bytes  255228.00 bytes/sec
total size is 41239979  speedup is 323.16 (DRY RUN)
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
$ rsync -av --delete  /usr/include .
```

6. **rsync** を適切なオプションの組み合わせで使うスクリプトを紹介します。

rsync を利用したスクリプト

```
#!/bin/sh
set -x

rsync --progress -avrxH --delete $*
```

ローカルマシン上でも、ネットワーク越えでも同様に実行できます。-x オプションを用いると、ファイルシステム境界を越えた **rsync** の実行を抑止します。

追加課題

複数のマシンを利用できるならば、ソースとバックアップ先に異なるマシンを指定して実行します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 33 章

Linux Security Modules

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- Linux Security Module のフレームワークの仕組みと導入方法を理解する。
- 利用可能なさまざまな LSM の実装をリストアップできる。
- SELinux の主な機能を説明する。
- さまざまなモードと利用できるポリシーについて説明する。
- コンテキストの重要性とその取得・設定方法を把握する。
- 重要な SELinux ユーティリティプログラムの使い方を把握する。
- AppArmor の使い方に慣れる。

33.1 Linux Security Modules

Linux Security Modules（LSM）とは

- **LSM** (**L**inux **S**ecurity **M**odule) **mandatory access controls** は、カーネルへのリクエストに対して強制的なアクセス制御を行う
- **LSM** の使用:
  - カーネルへの変更が最少となること
  - カーネルへのオーバーヘッドが最小となること
  - 異なる実装を柔軟に選択でき、それぞれが自己完結した **LSM** として提示される

最新のコンピュータシステムはセキュリティで保護する必要がありますが、そのセキュリティ要件はデータの機密性、アカウントを持つユーザーの数、外部ネットワークへの露出、法的要件、その他の要因によって異なります。適切なセキュリティ制御を使用可能にする責任は、アプリケーション設計者と **Linux** カーネル開発者とメンテナーにあります。もちろんユーザーも適切な手順に従う必要があります。しかし適切に実行されているシステムでは、非特権ユーザーはシステムをセキュリティ侵害にさらすことがないように操作権限が非常に限られているはずです。このセクションでは **Linux** カーネルが **Linux Security Modules** フレームワークを使用してセキュリティを強化する方法、特に **SELinux** を中心に説明します。

基本的な考え方は **システムコールをフック** することです。強化された権限で作業を実行するために、アプリケーションがカーネル（システム）モードへの移行を要求する部分にコードを挿入します。このコードはパーミッションが有効であること、悪意のある意図から保護されていることなどの確認を行います。これは、システムコールがカーネルによって実行される前後にセキュリティ関連の機能ステップを呼び出す形で行われます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

主な LSM の選択肢

- **SELinux**: https://selinuxproject.org/page/Main_Page
- **AppArmor**: https://apparmor.net
- **Smack**: http://schaufler-ca.com
- **TOMOYO**: https://tomoyo.osdn.jp
- **Yama**: https://kernel.org/doc/html/latest/admin-guide/LSM/Yama.html

複数の LSM の同時利用

2019 年以降は、複数の **LSM** を特定の定められた順序で組み合わせる（積み上げる）ことが可能となっている

- https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules にはそれぞれの LSM メカニズムの評価と LSM の選択に関する優れたドキュメントがあるので参照するとよい

長い間、強化セキュリティモデル実装は **SELinux** だけでした。2001 年にこのプロジェクトが最初にアップストリームでカーネルに組み込まれた時、セキュリティ強化のアプローチの選択肢が 1 種類に限定されることに対して反論がありました。

その結果 **SELinux** 以外のモジュールも使用できる **LSM** アプローチが採用され、開発成果は 2003 年に Linux カーネルに組み込まれました。

Linux カーネルの同じ部分を変更する可能性があるため、当初は一度に使用できる **LSM** は 1 つだけでした。

しかし現在では複数 **LSM** のスタックも可能です。スタックさせる時はモジュールに対し **major** と **minor** を指定します。

ユーザーの利用度の多い順に最初に **SELinux** を説明し、次に **AppArmor** を説明します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

33.2 SELinux

SELinux の概要

- NSA によって開発された
- 保護のためのメソッドが追加された
- 利用するのは:
  - コンテキスト
  - ルール
  - ポリシー
- デフォルトでは、明示的に許可されていないアクションは全て拒否する
- オンライン上の追加情報
  SELinux User's and Administrator's Guide:
  https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_users_and_administrators_guide/index

SELinux は、元々は米国 **NSA** (**N**ational **S**ecurity **A**dministration) によって開発されたもので、非常に長い間 **RHEL** に統合され多くの利用事例がありました。

運用という観点から見ると **SELinux** はプロセスがシステム上のどのファイル、ディレクトリ、ポート、およびその他のアイテムにアクセスできるかを決定するセキュリティルールのセットです。

３つの基本概念

1. **コンテキスト:** ファイル、プロセス、ポートに付与されたラベル。例には **SELinux** ユーザー、ロール、タイプがあります。
2. **ルール: コンテキスト**、**プロセス**、**ファイル**、**ポート**、**ユーザー** 等のアクセス制御を記述したものです。
3. **ポリシー: SELinux** が行うシステム全体のアクセス制御の判断が何かを記述する **ルール** のセットです。

SELinux コンテキストは、ユーザー、プロセス、ファイル、およびポートが相互に対話する方法を定義するルールによって使用されるラベルです。デフォルトのポリシーはアクセス拒否なので、システムで許可させたいアクションをルールで定義します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

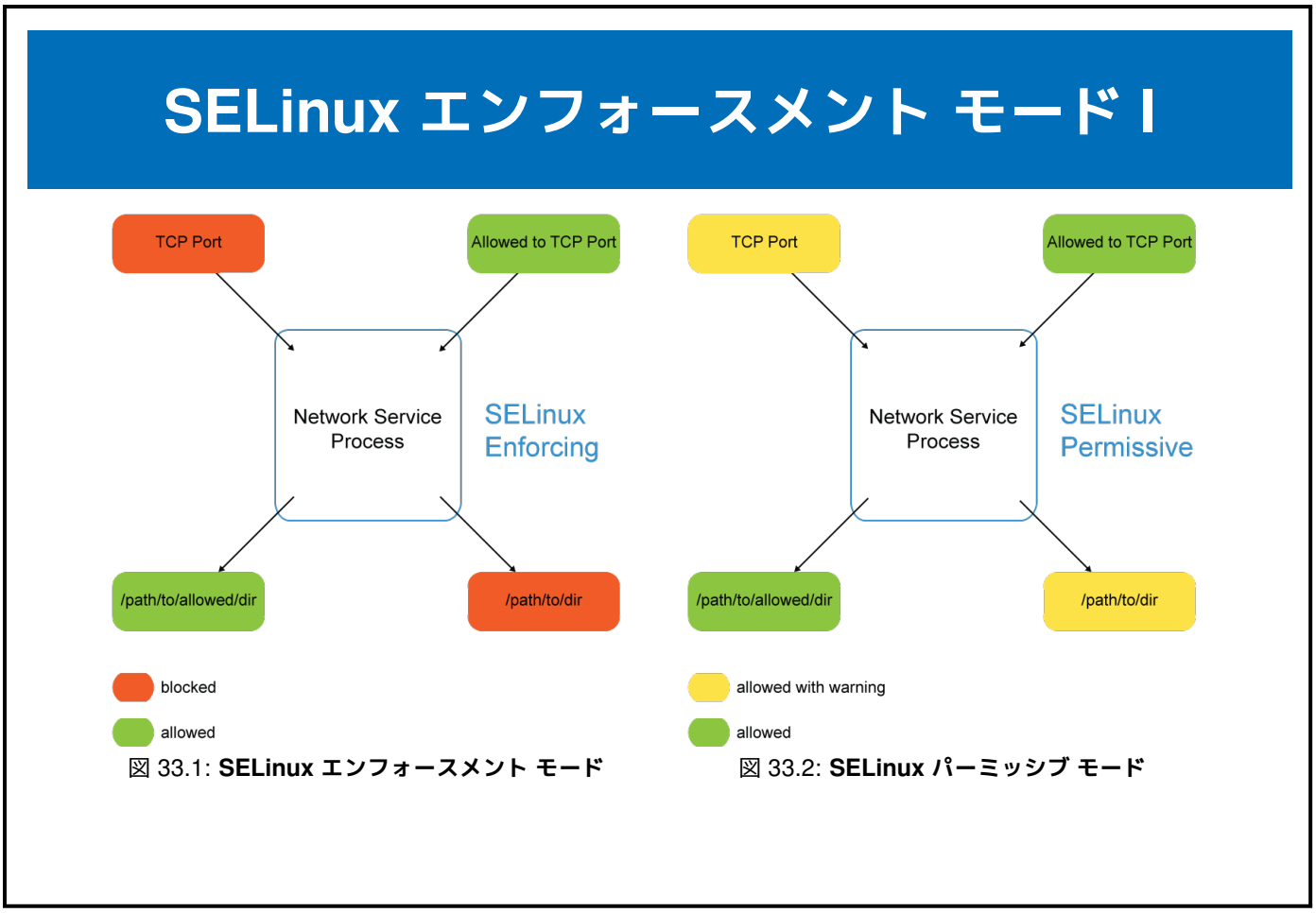


図 33.1: **SELinux エンフォースメント モード**

図 33.2: **SELinux パーミッシブ モード**

これらのモードは、ファイル（通常は `/etc/selinux/config`）で選択（および説明）され、その場所はディストリビューションによって異なります。（多くの場合 `/etc/sysconfig/selinux` もしくはそこからリンクされています）ファイルは読むだけで内容がわかるように自己文書化されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SELinux エンフォースメントモード II

- **SELinux** には３つのモードがある
  - エンフォーシング (Enforcing)
  - パーミッシブ (Permissive)
  - 無効 (Disabled)
- 定義ファイルは `/etc/sysconfig/selinux` (**CentOS** と **openSUSE**) または `/etc/selinux/config` (**Ubuntu**)
- **getenforce** と **setenforce** で、現在のモード確認、設定を行う

- **エンフォーシング: (Enforcing)** 全ての **SELinux** は有効で、ポリシーに従ってアクセスは拒否されます。全ての違反を監査しログに記録します。
- **パーミッシブ: (Permissive) SELinux** は有効ですが、アクセスを拒否せずエンフォーシングモードでは拒否される動作についての監査と警告だけが行われます。
- **無効: (Disabled) SELinux** カーネルとアプリケーションコードを完全に無効にして、システムを保護しません。

sestatus ユーティリティは、現在のモードとポリシーを表示します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

getenforce と setenforce

- 現在のモードの確認と設定
  - **getenforce**

```
$ getenforce
```

```
Enforcing
```

  - **setenforce**

```
$ sudo setenforce Permissive
$ getenforce
```

```
Permissive
```

setenforce を使用して、システムの稼働中に enforcing モードと permissive モードを切り替えることができます。ただし、この方法では disabled モードへの切り替えは行えません。

setenforce を使って **Permissive** と **Enforcing** モードを切り替えることができますが **SELinux** を完全に無効にすることはできません。**SELinux** を無効にする方法は少なくとも 2 つあります。

- **設定ファイル: SELinux** の設定ファイル (通常は `/etc/selinux/config`) を編集し `SELINUX=disabled` を指定する。これがデフォルトのやり方で **SELinux** を恒久的に無効化するために利用されます。
- **カーネルパラメータ:** 再起動時に `selinux=0` をカーネルパラメータに追加します。

ただし、**SELinux** を再度有効にする可能性のあるシステムでは **SELinux** を無効にすることはお勧めできません。**SELinux** を無効にするのではなく、パーミッシブモードを使用することをお勧めします。こうすれば、時間のかかるファイルシステム全体を対象としたラベルの再割り付けを回避できるからです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SELinux ポリシー

- **targeted:**
  SELinux の **デフォルトポリシー**で、プロセスを対象（=targeted）にかなりの制限を課す。ユーザープロセスと **init** プロセスは対象外だがネットワークサービスプロセスは対象となる。**SELinux** は **全てのプロセス** にメモリー制限を適用する。これにより、バッファオーバーフロー攻撃に対する脆弱性が軽減される。
- **minimum:**
  選択されたプロセスだけが保護される targeted ポリシーの一種である。
- **MLS:**
  マルチレベルセキュリティ（MLS: **M**ulti-**L**evel **S**ecurity）は、一番制限が厳しいポリシーである。全てのプロセスは、特定のポリシーを持つきめ細かいセキュリティ ドメインに置かれる。

同じ構成ファイル（通常は `/etc/sysconfig/selinux` です）に **SELinux ポリシー** を設定します。複数のポリシーを定義できますが、一度にアクティブにできるのは 1 つだけです。ポリシーを変更するには、システムの再起動とファイルシステムの内容の時間のかかる再ラベル付けが必要になる場合があります。各ポリシーは `/etc/selinux/[SELINUXTYPE]` の下にインストールされるファイルに設定されています。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

コンテキスト ユーティリティ

- コンテキストとコンテキスト ユーティリティ
  - ユーザー (User)
  - 役割 (Role)
  - タイプ (Type)
  - レベル (Level)
- コンテキストの参照と変更:
  - -Z オプションを使ってコンテストを見る

    ```
    $ ls -Z
    $ ps auZ
    ```
  - **chcon** を使って、コンテキストを変更する

    ```
    $ chcon -t etc_t somefile
    $ chcon --reference somefile someotherfile
    ```

前述したように、コンテキストはファイル、ディレクトリ、ポート、プロセスに適用されるラベルです。**SELinux** には **User**、**Role**、**Type**、**Level** という 4 つのコンテキストがあります。

ここでは、最も一般的に使用されるコンテキストである **type** に注目します。ラベルの命名規則により `kernel_t` のように type コンテキストのラベルは `_t` で終わる必要があります。

```
$ ls -Z
```

```
-rw-rw-r--. dog dog unconfined_u:object_r:user_home_t:s0 somefile
```

```
$ chcon -t etc_t somefile
$ ls -Z
```

```
-rw-rw-r--. dog dog unconfined_u:object_r:etc_t:s0   somefile
```

```
$ ls -Z
```

```
-rw-rw-r--. dog dog unconfined_u:object_r:etc_t:s0   somefile
-rw-rw-r--. dog dog unconfined_u:object_r:user_home_t:s0 somefile1
```

```
$ chcon --reference somefile somefile1
$ ls -Z
```

```
-rw-rw-r--. dog dog unconfined_u:object_r:etc_t:s0   somefile
-rw-rw-r--. dog dog unconfined_u:object_r:etc_t:s0   somefile1
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SELinux と標準コマンド

- **ls**、**ps**、**cp** など基本ユーティリティの多くが **SELinux** と連携するために拡張されている
- 通常は Z オプションを付加する:

```
$ ls -Z ...
$ ps axZ
```

- **SELinux** が無効化されていると、拡張オプションは機能しない

ls や **ps** などの多くの標準コマンドラインコマンドは **SELinux** をサポートするように拡張され、**man** ページに詳細な説明が追加されています。パラメータ Z が次のように標準のコマンドラインツールに渡されます:

```
$ ps axZ
```

```
LABEL PID TTY STAT TIME COMMAND
system_u:system_r:init_t:s0 1 ? Ss 0:04 /usr/lib/systemd/systemd --switched-root ...
system_u:system_r:kernel_t:s0 2 ? S 0:00 [kthreadd]
...
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2305 ? D 0:00 sshd: jimih@pts/0
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2306 pts/0 Ss 0:00 -bash
...
system_u:system_r:httpd_t:s0 7490 ? Ss 0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0 7491 ? S 0:00 /usr/sbin/httpd -DFOREGROUND
...
```

```
$ ls -Z /home/ /tmp/
```

```
/home/:
drwx------. jimih jimih unconfined_u:object_r:user_home_dir_t:s0 jimih
/tmp/:
-rwx------. root root system_u:object_r:initrc_tmp_t:s0 ks-script-c4ENhg
drwx------. root root system_u:object_r:tmp_t:s0 systemd-private-0ofSv0
-rw-------. root root system_u:object_r:initrc_tmp_t:s0 dnf.log
```

SELinux をサポートするために拡張されたツールには、他に **cp**、**mv**、**mkdir** があります。**SELinux** を無効にした場合、これらのユーティリティの関連フィールドには有用な情報は表示されないことに注意してください。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

コンテキストの継承

- 新しいファイルは、親ディレクトリからコンテキストを継承する
- ファイルを移動しても、元のディレクトリコンテキストが保存される
- コンテキストを変更しない場合には、問題が発生する場合がある

新しく作成されたファイルは親ディレクトリからコンテキストを継承しますが、ファイルを移動しても保存されているのは移動前のソースディレクトリのコンテキストであるため、移動後のディレクトリで問題を引き起こす可能性があります。

前の例で言えばファイルを `/tmp` から `/home/jimih` に移動しても `tmpfile` のコンテキストは変更されません。

```
$ cd /tmp/
$ touch tmpfile
$ ls -Z tmpfile
```

```
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_tmp_t:s0 tmpfile
```

```
$ cd
$ touch homefile
$ ls -Z homefile
```

```
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
```

```
$ mv /tmp/tmpfile .
$ ls -Z
```

```
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_tmp_t:s0 tmpfile
```

ファイルの移動で **SELinux** の問題が発生する典型的な例は **httpd** サーバーの `DocumentRoot` ディレクトリへのファイル移動です。**SELinux** が有効化されたシステムでは、ウェブサーバーは正しいコンテキストラベルを持つファイルにだけアクセスできます。ファイルを `/tmp` に作成し、それを `DocumentRoot` ディレクトリに移動した場合、そのファイルの **SELinux** コンテキストが調整されるまで **httpd** サーバーからファイルにアクセスできなくなります。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

restorecon

- 親ディレクトリと調和するように、ファイルコンテキストをリセット

```
$ restorecon -Rv /home/jimih
```

restorecon は、親ディレクトリの設定に基づいてファイルコンテキストをリセットします。
次の例では **restorecon** はホームディレクトリの全てのファイルのデフォルトラベルを再帰的にリセットします。

```
$ ls -Z
```

```
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_tmp_t:s0 tmpfile
```

```
$ restorecon -Rv /home/jimih
```

```
restorecon reset /home/jimih/tmpfile context \
          unconfined_u:object_r:user_tmp_t:s0->unconfined_u:object_r:user_home_t:s0ã
```

```
$ ls -Z
```

```
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_home_t:s0 homefile
-rw-rw-r--. jimih jimih unconfined_u:object_r:user_home_t:s0 tmpfile
```

`tmpfile` のコンテキストが、ホームディレクトリで作成されたファイルのデフォルトコンテキストにリセットされていることに注目してください。タイプが `user_tmp_t` から `user_home_t` に変更されました。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

semanage

- 新しいディレクトリに対する、デフォルトコンテキストを設定する
- デフォルト値の設定だけで、既存のディレクトリの設定は変更されない
- 既存コンテキストの設定変更は **restorecon** を続けて実行する

別の課題として、新しく作成されたディレクトリのデフォルトコンテキストの設定方法があります。**semanage fcontext** (**policycoreutils-python** パッケージによって提供される) は、ファイルおよびディレクトリのデフォルトコンテキストの変更と表示ができます。**semanage fcontext** はデフォルト設定だけを変更することに注意してください。既存のオブジェクトにすぐには適用されません。これをするには、その後に **restorecon** を実行する必要があります。例えば:

```
[root@rhel7 /]# mkdir /virtualHosts
[root@rhel7 /]# ls -Z
```

```
...
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 virtualHosts
```

```
[root@rhel7 /]# semanage fcontext -a -t httpd_sys_content_t /virtualHosts
[root@rhel7 /]# ls -Z
```

```
...
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 virtualHosts
```

```
[root@rhel7 /]# restorecon -RFv /virtualHosts
```

```
restorecon reset /virtualHosts context \
          unconfined_u:object_r:default_t:s0->system_u:object_r:httpd_sys_content_t:s0
```

```
[root@rhel7 /]# ls -Z
```

```
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 virtualHosts
```

したがって `default_t` から `httpd_sys_content_t` へのコンテキスト変更は **restorecon** 実行後に適用されます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SELinux のブール値の使い方

- **SELinux** ポリシーの挙動を変更する
- 有効化／無効化の設定は
  - **getsebool**: ブール値を表示する
  - **setsebool**: ブール値を設定する
  - **semanage boolean -l**: 恒久的にブール値を設定する

```
$ setsebool allow_ftpd_anon_write on
$ getsebool allow_ftpd_anon_write
  allow_ftpd_anon_write -> on
$ semanage boolean -l | grep allow_ftpd_anon_write
$ allow_ftpd_anon_write -> off
```

これは、恒久的な設定ではありません。

```
$ setsebool -P allow_ftpd_anon_write on
$ semanage boolean -l | grep allow_ftpd_anon_write
  allow_ftpd_anon_write          -> on
```

これで恒久化されました。

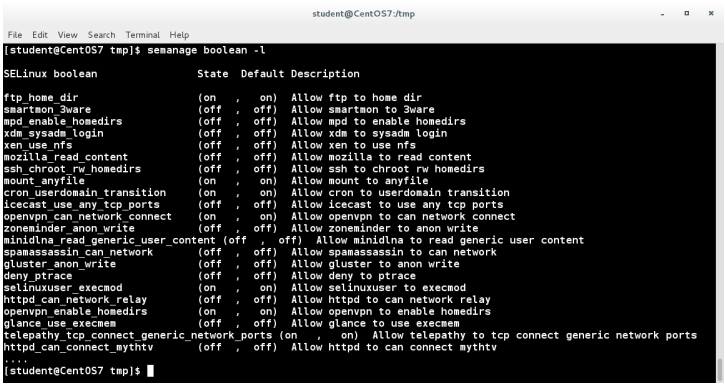


図 33.3: **semanage**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

SELinux アクセスの監視

- **setroubleshoot-server** を使って **SELinux** のエラーを追跡する
  - インストール後に **auditd** サービスを再起動する
- 生（= RAW）メッセージは `/var/log/audit/audit.log` に保存
- メッセージを `/var/log/messages` 移動する
- **sealert** を利用して詳細メッセージを見る

SELinux には、実行時に問題を収集し、これらの問題をログに記録し、同じ問題の再発を防ぐソリューションを提案するツール群が付属しています。これらのユーティリティは **setroubleshoot-server** パッケージで提供されます。それらの使用例を次に示します。

```
[root@rhel7 ~]# echo 'File created at /root' > rootfile
[root@rhel7 ~]# mv rootfile /var/www/html/
[root@rhel7 ~]# wget -O - localhost/rootfile
--2014-11-21 13:42:04-- ähttp://localhost/rootfile
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2014-11-21 13:42:04 ERROR 403: Forbidden.

[root@rhel7 ~]# tail /var/log/messages
Nov 21 13:42:04 rhel7 setroubleshoot: Plugin Exception restorecon
Nov 21 13:42:04 rhel7 setroubleshoot: SELinux is preventing /usr/sbin/httpd from getattr access on the file .
For complete SELinux messages. run sealert -l d51d34f9-91d5-4219-ad1e-5531e61a2dc3
Nov 21 13:42:04 rhel7 python: SELinux is preventing /usr/sbin/httpd from getattr access on the file .
.....
Do allow this access for now by executing:
# grep httpd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp

Additional Information:
Source Context system_u:system_r:httpd_t:s0
Target Context unconfined_u:object_r:admin_home_t:s0
Target Objects [ file ]
Source httpd
Source Path /usr/sbin/httpd
......
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

33.3 AppArmor

AppArmor は **SELinux** の代替となる **LSM** です。2006 年以降 **Linux** カーネルにサポートが組み込まれています。**SUSE**、**Ubuntu** およびその他のディストリビューションで使用されています。

AppArmor は強制アクセス制御 (= **M**andatory **A**ccess **C**ontrol (**MAC**)) を提供して、従来の **UNIX** の任意アクセス制御 (=**D**iscretionary **A**ccess **C**ontrol (**DAC**)) モデルを補完します。

手動によるプロファイル指定に加えて **AppArmor** には学習モードがあります。このモードでは、プロファイルの違反はログに記録されますが、プログラムの実行は阻止されません。典型的なプログラム動作に基づくログからプロファイルに変換できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

状態の確認

- **AppArmor** はデフォルトでロードされ、有効化される
- **Linux** カーネルのコンパイル時に組み込んで有効化する必要がある
- 起動、停止など

  `$ sudo systemctl [start|stop|restart|status] apparmor`
- 起動時にロードするかどうかの設定

  `$ sudo systemctl [enable|disable] apparmor`
- 現在のステータスの詳細を見る

  `$ sudo apparmor_status`
- **Profiles** と **processes** には **enforce** と **complain** モードがある
  - **SELinux** における **enforcing** と **permissive** モードに相当する

AppArmor を採用しているディストリビューションは、デフォルトでそれを有効にしてロードすることが多いです。**Linux** カーネル（= **AppArmor** カーネルモジュール）も同様に有効にする必要があります。そして多くの場合一度に実行できる **LSM** は１つだけです。**AppArmor** カーネルモジュールが利用可能である場合 **systemd** が採用されたシステムでは次のことが可能です。

```
$ sudo systemctl [start|stop|restart|status] apparmor
```

現在の動作ステータスを変更、ないし問い合わせします。または、実行します。

```
$ sudo systemctl [enable|disable] apparmor
```

起動時にロードする／しないを指定します。
現在のステータスを見るには:

```
$ sudo apparmor_status
```

```
   apparmor module is loaded.
  25 profiles are loaded.
  25 profiles are in enforce mode.
     /sbin/dhclient
  ....
```

Profiles と **processes** には **enforce** ないし **complain** モードがあります。**SELinux** の **enforcing** と **permissive** に相当するものです。
プロセスのリストでは **PID** が与えられます。

```
$ ps aux | grep libvirtd
```

```
  root        787  0.0  0.9 527200 35936 ?        Ssl  10:54   0:00 /usr/sbin/libvirtd
  student    3346  0.0  0.0  13696  2204 pts/16   S+   11:42   0:00 grep --color=auto libvirtd
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

モード と プロファイル

- モード:
  - **エンフォース (Enforce) モード**　制限されたアクションは阻止され、試行が記録される。**aa-enforce** で設定する
  - **コンプレイン (Complain) モード**　ポリシーは実施されないが、違反の試行は記録される。**aa-complain** で設定する
- `/etc/apparmor.d` の中に設定された **ポリシー** によってユーティリティやソフトウェアパッケージの詳細設定ができる

プロファイルによって `/usr/bin/evince` などのパス名を持つシステム上の実行プログラムの使用方法を制限します。
プロセッサーは下記のどちらかのモードで実行されます。

- **エンフォース (Enforce) モード**
  アプリケーションにはポリシー遵守が強制され、制限された動作は阻止されます。違反の試行がシステムログファイルに記録されます。これがデフォルトのモードです。プロファイルをこのモードに設定するには **aa-enforce** を使用します。
- **コンプレイン (Complain) モード**
  ポリシー遵守の強制はありませんが、ポリシー違反の試行は記録されます。これは学習モードとも呼ばれます。プロファイルをこのモードに設定するには **aa-complain** を使用します。

Linux ディストリビューションには、あらかじめパッケージ化されたプロファイルが付属しており、通常は、その特定のパッケージがインストールされたときにプロファイルが一緒にインストールされます。もしくは **apparmor-profiles** などの **AppArmor** パッケージと一緒にプロファイルがインストールされます。これらのプロファイルは `/etc/apparmor.d` に保存されます。

新しいソフトウェアをインストールするときにはパッケージ内の実行可能ファイルに紐付いた新しいプロファイルを作成できます。

システムにインストールされる正確な **AppArmor** プロファイルは、ソフトウェアパッケージ選択によって異なります。
たとえばある **Ubuntu** システムの場合

```
student@ubuntu:/etc/apparmor.d$ ls
```

```
abstractions              usr.lib.dovecot.anvil          usr.lib.telepathy
apache2.d                 usr.lib.dovecot.auth           usr.sbin.avahi-daemon
bin.ping                  usr.lib.dovecot.config         usr.sbin.cups-brows
....
```

これらのファイルで何ができるかについての完全なドキュメントは `man apparmor.d` とすれば入手できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
THE LINUX FOUNDATION Training & Certification

AppArmor ユーティリティ

Table 33.1: **AppArmor ユーティリティ**

プログラム	用　途
apparmor_status	全てのプロファイルおよびプロファイルを含むプロセスのステータスを表示
apparmor_notify	**AppArmor** のログメッセージの概要を表示
complain	特定のプロファイルを complain モードに設定
enforce	指定したプロファイルを強制モードに設定します
disable	現在のカーネルから指定されたプロファイルをアンロードし、システム起動時にロードされないようにする
logprof	ログファイルをスキャンし既存のプロファイルでカバーされていない **AppArmor** イベントが記録されている場合は考慮に入れる方法を提案し、承認されている場合は制限するように変更して再ロードする
easyprof	プログラムの基本的な **AppArmor** プロファイルの生成を支援します。簡易なインターフェイスを提供

AppArmor には、監視と制御のための管理ユーティリティが数多くあります。たとえば **openSUSE** システムの場合

```
$ rpm -qil apparmor-utils | grep bin
```

```
/usr/bin/aa-easyprof
/usr/sbin/aa-audit
/usr/sbin/aa-autodep
/usr/sbin/aa-cleanprof
/usr/sbin/aa-complain
/usr/sbin/aa-decode
/usr/sbin/aa-disable
/usr/sbin/aa-enforce
/usr/sbin/aa-exec
......
/usr/sbin/complain
/usr/sbin/decode
/usr/sbin/disable
/usr/sbin/enforce
....
```

多くのユーティリティは、短縮された名前を使っても完全な長い名前でも起動することができます。例えば

```
linux-llgn:/etc/apparmor.d # ls -l /usr/sbin/*complain
```

```
-rwxr-xr-x 1 root root 1442 Oct 25 07:37 /usr/sbin/aa-complain*
lrwxrwxrwx 1 root root   11 Nov 11 13:02 /usr/sbin/complain -> aa-complain*
linux-llgn:/etc/apparmor.d #
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

33.4 演習 **

課題 33.1: SELinux：コンテキスト

注目

本演習は（**RHEL** のような）**SELinux** がインストールされているシステムでだけ実行可能です。**Ubuntu** のような **Debian** ベースのディストリビューションでもインストールは可能ですが容易ではなく、しばしば失敗します。

1. **getenforce** と **sestatus** を実行して **SELinux** がイネーブルになっていることと **enforcing** モードであることを確認してください。なっていなければ /etc/selinux/config を編集してからリブートして再確認してください。
2. **httpd** パッケージを（まだ、していなければ）インストールしてください。これは **Apache** ウェブサーバです。立ち上がっていることを確認してください。

   ```
   $ sudo dnf install httpd
   $ sudo systemctl start httpd
   $ elinks http:/localhost
   ```

   （ここから先のステップでは、ブラウザとして **lynx** や **elinks** なども使うことができますし、グラフィカル ブラウザとして **firefox** や **chrome** を利用することもできます）
3. スーパーユーザーで /var/www/html に小さなファイルを作成します。

   ```
   $ sudo sh -c "echo file1 > /var/www/html/file1.html"
   ```
4. このファイルが見えることを確認します:

   ```
   $ elinks -dump http://localhost/file1.html
   ```

   ```
   file1
   ```

   root のホーム ディレクトリに別の小さなファイルを作成し、それを /var/www/html に **move** します。（コピーではありません。移動してください）見てみます。

   ```
   $ sudo cd /root
   $ sudo sh -c "echo file2 > file2.html"
   $ sudo mv file2.html /var/www/html
   $ elinks -dump http://localhost/file2.html
   ```

   ```
                                  Forbidden

   You don't have permission to access /file2.html on this server.
   ```
5. セキュリティコンテキストを調べます。

   ```
   $ cd  /var/www/html
   $ ls -Z file*html
   ```

   ```
   -rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 file1.html
   -rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file2.html
   ```
6. offending コンテキストを変更し、あらためて調べます。

   ```
   $ sudo chcon -t httpd_sys_content_t file2.html
   $ elinks http://localhost/file2.html
   ```

   ```
   file2
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

課題 33.2: apparmor セキュリティを使う

注目

本演習は、（**Ubuntu** のような）**AppArmor** がインストールされているシステムでだけ実行可能です。

以下は **Ubuntu** 上でテストしています。しかし、他の **AppArmor** が実行可能なシステム、たとえば **openSUSE** などでも、**apt** コマンドを **zypper** に置き換えるだけでできるはずです。

Ubuntu では **/bin/ping** ユーティリティは、SUID されて実行されます。本演習では **ping** を **ping-x** にコピーして、プログラムが機能するようにケーパビリティを調整します。

次に **AppArmor** プロファイルをビルド、インストールして何も変わっていないことを確認します。**AppArmor** プロファイルを変更、ケーパビリティの追加を行いプログラムの機能性を高めます。

1. 必要なパッケージがインストールされていることを確認します。

```
student@ubuntu:~$ sudo apt install apparm*
```

2. **ping** のコピーを作成し（ここでは、**ping-x** とします）、特別なパーミッションやケーパビリティを持っていないことを確認します。さらに、**student**、一般ユーザーでは起動できないことを確認します。

```
student@ubuntu:~$ sudo cp /bin/ping /bin/ping-x
student@ubuntu:~$ sudo ls -l /bin/ping-x

-rwxr-xr-x 1 root root 64424 Oct 17 10:12 /bin/ping-x

student@ubuntu:~$ sudo getcap /bin/ping-x
student@ubuntu:~$

student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
ping: socket: Operation not permitted
```

3. **capabilities** をセットして **ping-x** を再度実行します。

```
student@ubuntu:~$ sudo setcap cap_net_raw+ep  /bin/ping-x

student@ubuntu:~$ ping-x  -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.093 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.086 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2034ms
rtt min/avg/max/mdev = 0.086/0.090/0.093/0.008 ms
```

変更した **ping-x** プログラムが動作するようになりました。

4. **ping-x** 用の **AppArmor** プロファイルが無く、**ping** 用のプロファイルしかなかったことを確認します。現在の **ping** プログラムの状態を調べます:

```
student@ubuntu:~$ sudo aa-status
```

aa-status の出力は長いため **grep** を使って関心のある部分を抜き出します。

```
student@ubuntu:~$ sudo aa-status | grep -e "^[[:alnum:]]" -e ping

apparmor module is loaded.
87 profiles are loaded.
51 profiles are in enforce mode.
   ping
36 profiles are in complain mode.
17 processes have profiles defined.
6 processes are in enforce mode.
11 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

ping がプロファイルを持ち、実行（enforce）されていることがわかります。

5. 次に **ping-x** 用にプロファイルを作成します。複数の端末を利用します。
一番目の端末（**端末 1**）で、**aa-genprof** コマンドを実行します。これは /var/log/syslog から **AppArmor** のエラーをスキャンすることで、**AppArmor** プロファイルを生成します。
二番目の端末（**端末 2**）では **ping-x** を実行します。（より詳細な情報は、**aa-genprof** の **man** ページを見てください）
端末 1：

```
student@ubuntu:~$ sudo aa-genprof /bin/ping-x
Writing updated profile for /bin/ping-x.
Setting /bin/ping-x to complain mode.

Before you begin, you may wish to check if a
profile already exists for the application you
wish to confine. See the following wiki page for
more information:
http://wiki.apparmor.net/index.php/Profiles

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

Profiling: /bin/ping-x

[(S)can system log for AppArmor events] / (F)inish
```

端末 2：

```
student@ubuntu:~$ ping-x  -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.099 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.120 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.114 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.099/0.111/0.120/0.008 ms
```

端末 1：
ping-x コマンドの実行が完了したら、**aa-genprof** を使ってプロファイルに必要な情報を追加します。必要な全ての情報を収集するためには何回か **スキャン** が必要です。
S を入力してスキャンを実行します。

```
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile:    /bin/ping-x
Capability: net_raw
Severity:   8

 [1 - capability net_raw,]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

A を入力して、ケーパビリティを許可してください。

```
Adding capability net_raw, to profile.

Profile:        /bin/ping-x
Network Family: inet
Socket Type:    raw
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
 [1 - network inet raw,]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

A を入力してネットワークファミリーを許可してください。

```
Adding network inet raw, to profile.

Profile:        /bin/ping-x
Network Family: inet
Socket Type:    dgram

 [1 - #include <abstractions/nameservice>]
  2 - network inet dgram,
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

A を入力してプロファイルにソケットタイプのデータグラムを追加してください。

```
Adding #include <abstractions/nameservice> to profile.

= Changed Local Profiles =

The following local profiles were changed. Would you like to save them?

 [1 - /bin/ping-x]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w (C)lean profiles / Abo(r)t
```

S を入力して新しいプロファイルを格納してください。

```
Writing updated profile for /bin/ping-x.

Profiling: /bin/ping-x

[(S)can system log for AppArmor events] / (F)inish
```

F を入力して終了してください。

```
Setting /bin/ping-x to enforce mode.

Reloaded AppArmor profiles in enforce mode.

                Please consider contributing your new profile!
See the following wiki page for more information:
http://wiki.apparmor.net/index.php/Profiles

Finished generating profile for /bin/ping-x.
```

6. /etc/appamor.d/bin.ping-x に格納されているプロファイルを見てみます。

```
student@ubuntu:~$ sudo cat /etc/apparmor.d/bin.ping-x
# Last Modified: Tue Oct 17 11:30:47 2017
#include <tunables/global>

/bin/ping-x {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability net_raw,

  network inet raw,

  /bin/ping-x mr,
  /lib/x86_64-linux-gnu/ld-*.so mr,

}
```

7. **aa-genproc** ユーティリティは、新しいポリシーをインストールし活性化します。すぐに使うことができ **systemctl reload apparmor** コマンドによって、必要に応じてリロードされます。余計な問題を避け、変更が保存されていることを確認するためシステムをリブートします。
システムを再起動したら、student として新しいプロファイルのもとで **ping-x** が機能することを確認します。ip アドレスを使ってローカルホストに ping します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

```
student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.095 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.043/0.065/0.095/0.021 ms
```

8. これは期待通りに動作するはずです。プロファイルは限定的で、**AppArmor** は指定されたパラメータ以上は機能的に許可しません。**AppArmor** がアプリケーションを保護していることを確認するため、ローカルホストの **IPV6** アドレス宛に ping してください。
   失敗します。

   ```
   student@ubuntu:~$ ping-x -c3  -6 ::1
   ping: socket: Permission denied
   ```

   (-6 オプションは **IPv6** だけの使用を意味し、::1 は、**IPv6** のローカルホストを意味します)
   出力を見るとソケットに関する問題があることがわかります。システムログを見ると **AppArmor** が **ping-x** プログラムに対して **IPv6** アクセスを許可しないことがわかります:

   ```
   766:104): apparmor="DENIED" operation="create" profile="/bin/ping-x"
   pid=2709 comm="ping-x" family="inet6" sock_type="raw" protocol=58
   requested_mask="create" denied_mask="create
   ```

9. これを解決するために以前のように **aa-genprof** を実行し、**端末 2** で **IPv6** ループバックにオプションを追加して ping します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

第 34 章

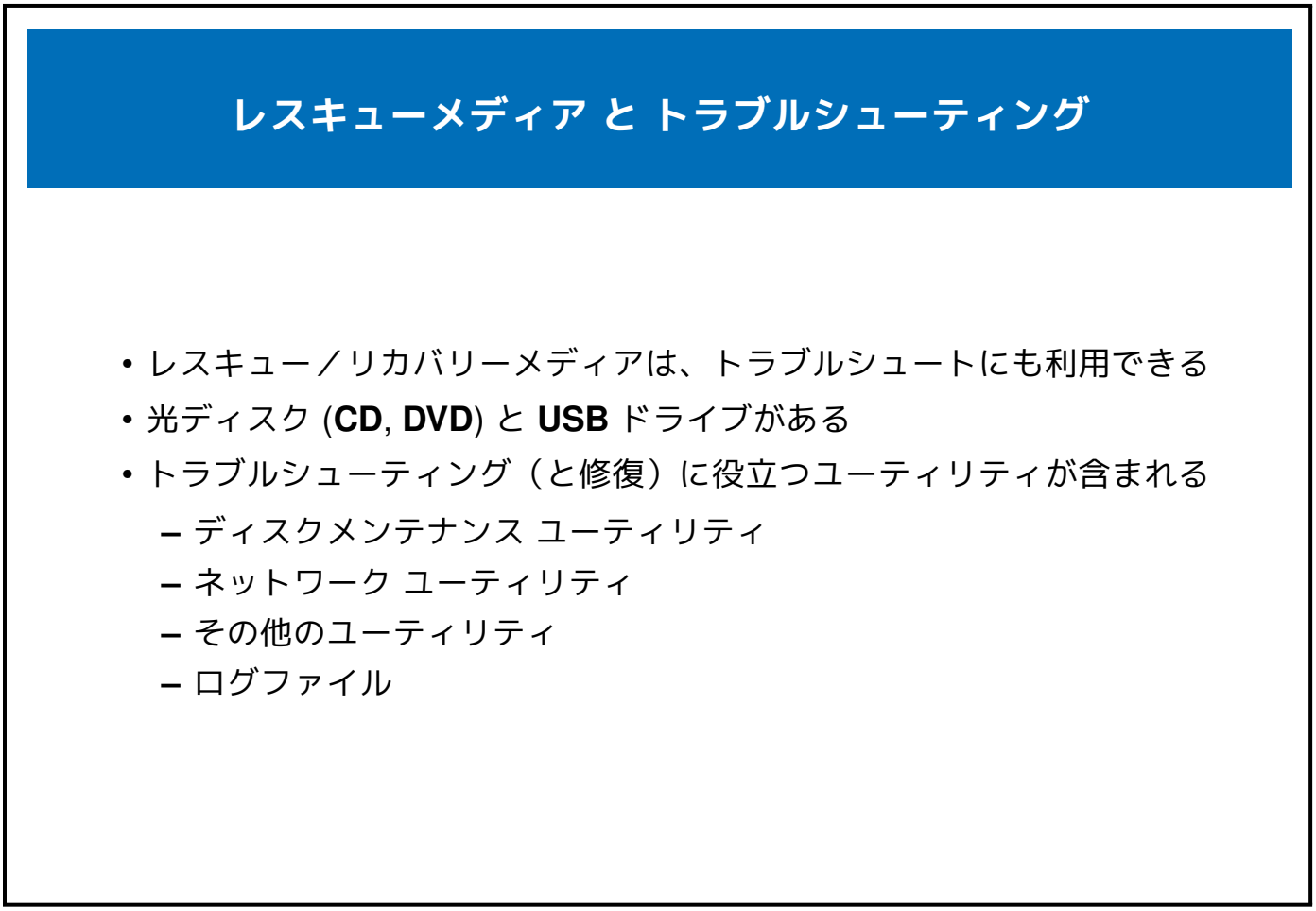
システム救出 (レスキュー)

学習目標

このセッションが終わるころには、次のことができるようになっているはずです

- システム レスキュー メディアはどのような形態で提供され、どう使うのか、または準備するかを説明する。
- レスキューメディアを使用して、システムの修復する。
- 緊急 モードへの移行方法、そこで何ができるかを説明する。
- シングル ユーザー モードへの移行方法、そこで何ができるか、緊急 モードとの違いを説明する。

34.1 レスキューメディア と トラブルシューティング



システムのレスキュー（救済）とリカバリー（修復）について説明しますが、ここではレスキューメディアを使ったトラブルシュートについて述べます。

何を選択するかは **Linux** ディストリビューションによって異なりますが、インストールメディアまたはライブ **CD**/**DVD** ないし **USB** ドライブから起動する時に `Rescue Installed System` などの名前がついた起動オプションの選択が可能でしょう。

レスキューディスクには、便利なプログラムが多く含まれています。

- パーティション作成、**RAID** デバイス管理、論理ボリューム管理、ファイルシステム作成のためのディスクユーティリティ: **fdisk**、**mdadm**、**pvcreate**、**vgcreate**、**lvcreate**、**mkfs**、これ以外にも多数あります。
- ネットワークのデバッグとネットワーク接続のためのネットワークユーティリティ **ifconfig**、**route**、**traceroute**、**mtr**、**host**、**ftp**、**scp**、**ssh**
- これ以外にも、利用可能なユーティリティが多数あります。 **bash**、**chroot**、**ps**、**kill**、**vi**、**dd**、**tar**、**cpio**、**gzip**、**rpm**、**mkdir**、**ls**、**cp**、**mv**、**rm** など

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

34.2 レスキュー／リカバリ メディアの使用

レスキュー／リカバリ メディア イメージの使用

- 可能な場合には、レスキュー対象システムのファイルシステムを（通常は /mnt/sysimage に）マウントするか聞いてくる
- **chroot** を使って、レスキュー対象システムの環境を復元できる
- （ローカル、ないしリモートから）ソフトウェアをインストールできるようにする

レスキューイメージを起動すると、いくつか質問をしてきます。その１つは（それが可能な場合に）ファイルシステムをマウントするかどうかです。

マウントを指定した場合どこかに（通常は /mnt/sysimage に）マウントされます。そのディレクトリに移動すればファイルにアクセスできます。もしくは次のコマンドでその環境に（＝リカバリー対象の元々のディレクトリレイアウトに）変更できます。

```
$ sudo chroot /mnt/sysimage
```

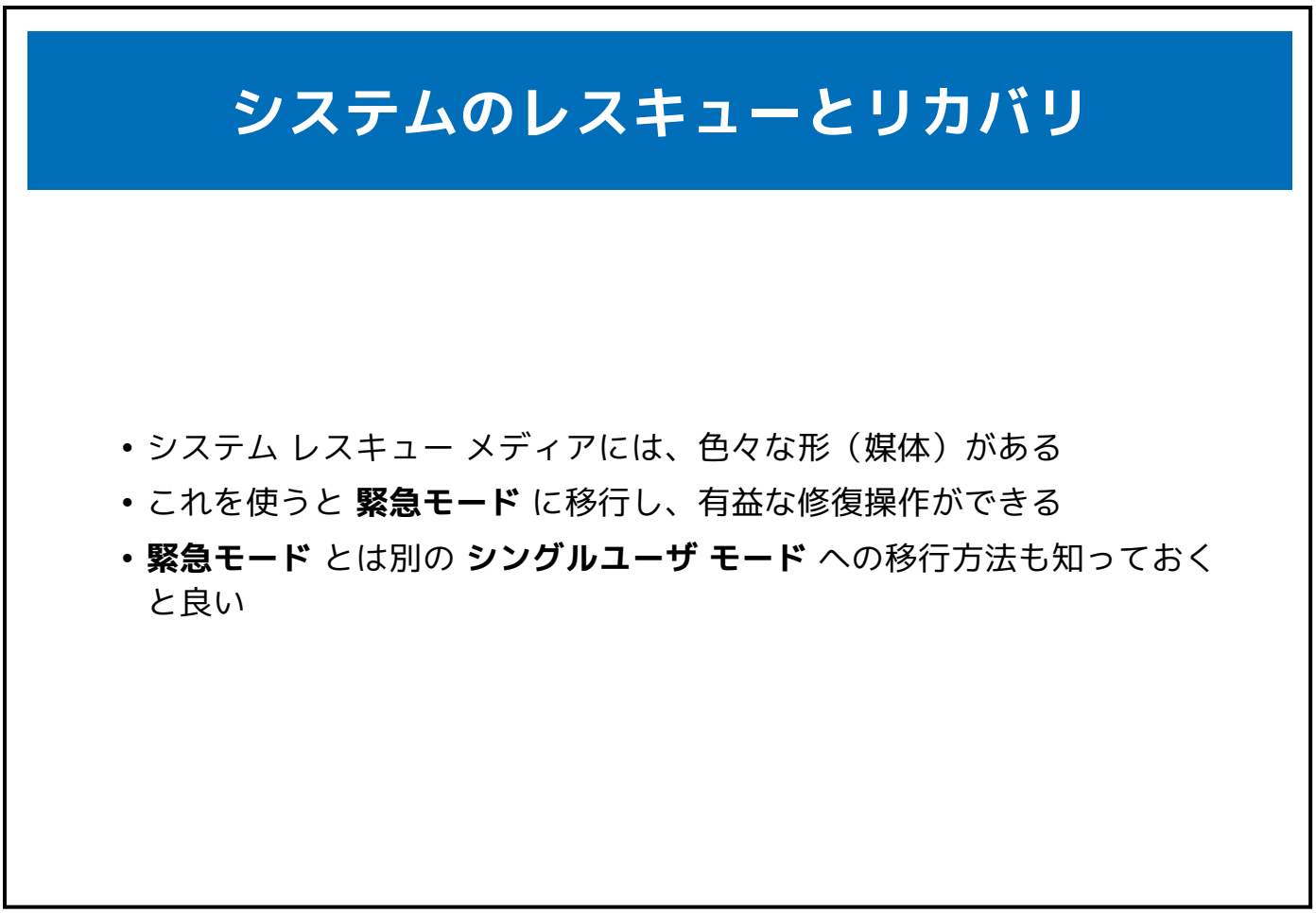
ネットワークベースのレスキューの場合 /mnt/source をマウントするよう求められる場合もあります。

chroot した環境の中からソフトウェアパッケージをインストールすることができます。または chroot した環境の外部からインストールすることもできます。たとえば **RPM** ベースのシステムでは --root オプションを使用してルートディレクトリの場所を指定できます。

```
$ sudo rpm -ivh --force --root=/mnt/sysimage /mnt/source/Packages/vsftpd-2*.rpm
```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

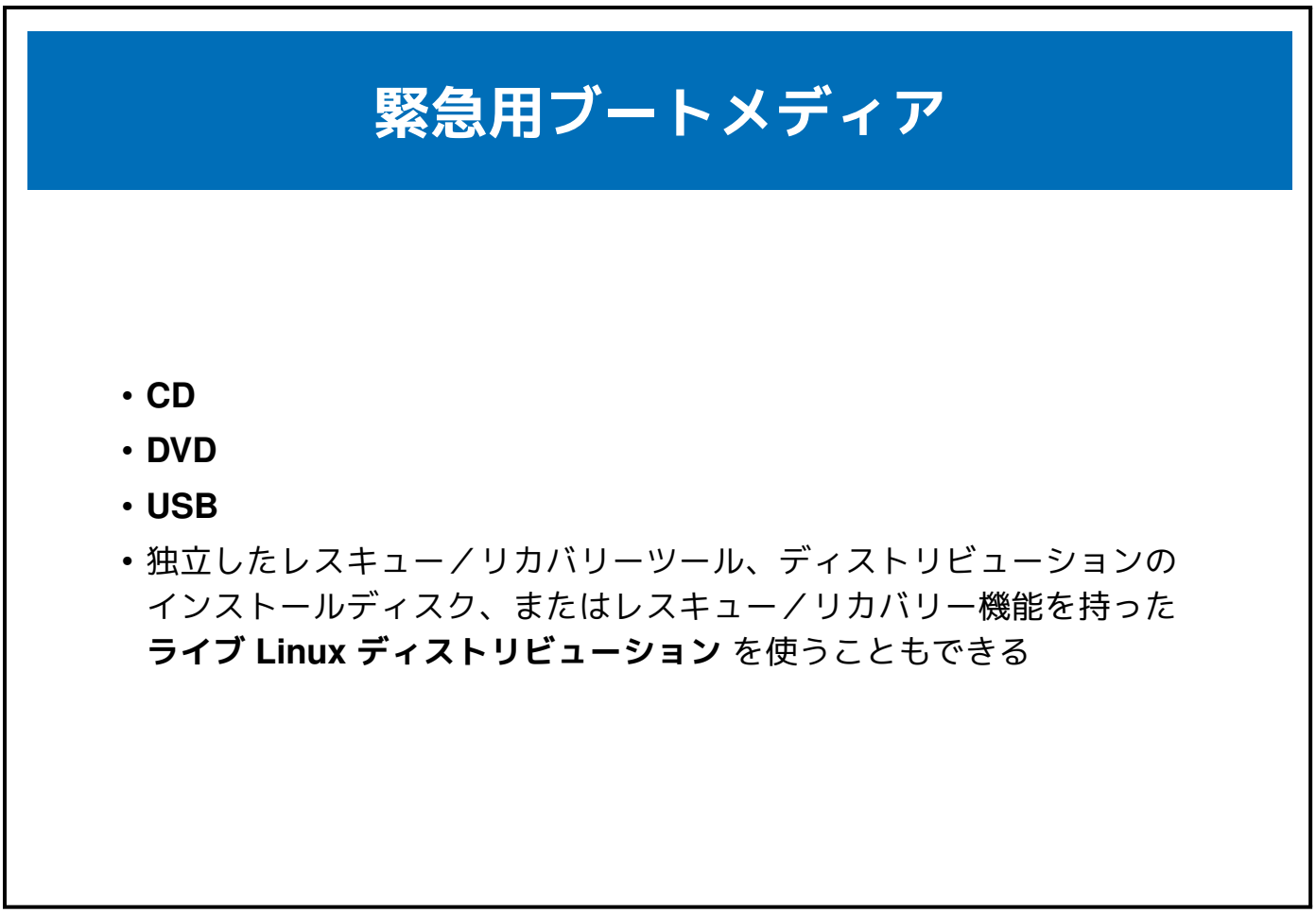
34.3 システムのレスキューとリカバリ



システムは、遅かれ早かれ適切にブートできないとか、ファイルシステムのマウントに問題がでるとか、デスクトップ表示の開始に問題があるなどといった重大な障害に遭遇してしまう可能性があります。光ディスク、またはポータブル **USB** ドライブ形式の **システム レスキューメディア** を使用してこれらの症状を解決できます。**緊急モード** または **シングルユーザー モード** でブートすると一連の **Linux** ツールを使用してシステムを通常の機能に修復できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

34.4 緊急用ブートメディア



緊急用ブートメディアは、ファイルの欠落、構成の誤り、ファイルの破損、サービスの構成の誤りなどが原因でシステムがブートしない時に役立ちます。

何らかの理由で root パスワードが失われたり、変に改ざんされたりしてリセットする必要がある場合にも、レスキューメディアが役立つことがあります。

ほとんどの **Linux** ディストリビューションのインストール用のメディア（**CD**, **DVD**, **USB**）や **ライブメディア** は、レスキューディスクとしても利用できるようになっています。これは非常に便利です。専用のレスキューディスクも利用できます。

（任意の形式の）ライブメディアは、メモリー内だけで完結した、完全で起動可能なオペレーティング システムを提供します。ディスクからはロードされません。ユーザーは、実際に（ディスクに）インストールしたり、コンピュータ上の既存のオペレーティング システムに変更を加えたりすることなく、オペレーティング システムや **Linux** ディストリビューションを体験、および評価できます。

ライブ リムーバブルメディアは、ハードディスクドライブなどのセカンダリ ストレージが欠けているコンピュータや、またはハードディスクドライブやファイルシステムが破損しているコンピュータでも実行できるという優れた点があり（ディスクがオフラインとなっている間に）ユーザーはデータをレスキューできます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

34.5 レスキュー メディアの使用

レスキュー メディアの使用

- 事実上全ての **Linux** ディストリビューションの **GRUB** のメニューにはレスキュー イメージの選択がある。もしあなたのシステムにこの選択肢がありレスキュー オプションが働くなら、レスキュー メディアをマウントする前にこれを試すべきである。
- システム起動時にメニュー選択から `rescue` を選ぶか、または `boot:` プロンプトに以下のようにタイプする必要がある。

  ```
  Linux rescue
  ```

- 言語選択やその他（ネットワークを初期化するか等）の質問を受けるかもしれない
- また、おそらくどのファイルシステムをマウントするかなども質問されるだろう

ライブ、インストール、レスキューのいずれのメディアを使用している場合でも、レスキューとリカバリのために特別なオペレーティング システムを開始する手順は同じで、紹介したように１つのメディアが３つの目的全てを果たします。

システムはリムーバブルメディアからブートすると、ブートメニューのオプションからレスキュー／リカバリモードにアクセスできます。多くの場合次のように `rescue` と入力する必要があります。

```
boot: Linux rescue
```

各ディストリビューションには多少の違いがありますが、手順を確認するのは簡単ですので全ての手順を説明しません。

次に、使用する言語などのいくつかの質問と、いくつかのディストリビューションに依存する選択があります。そして、正しいレスキューイメージの場所を選択するように求められます。**CD/DVD**、**ハードディスク**、**NFS**、**FTP**、**HTTP** など。

選択した場所には、正しいインストール ツリーが含まれている必要があり、インストール ツリーはレスキューディスクと同じ **Linux** バージョンのものである必要があります。リムーバブル メディアを使用している場合、インストール ツリーはメディアの作成元と同じものでなければなりません。ベンダーからダウンロードした `boot.iso` イメージを使用している場合は、ネットワークベースのインストールツリーも必要です。

ファイルシステムのマウントについても質問されます。それらが見つかると `/mnt/sysimage` の下にマウントされます。その後シェルプロンプトが表示され、システムに適切な修正を行うためのさまざまなユーティリティにアクセスできます。

chroot はルート（`/`）ファイルシステムへのアクセスを改善するために使用できます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

レスキュー用の USB キー

- 現在の全てのディストリビューションでは **USB** インストール／ライブ／レスキューディスクイメージが利用可能である
- **USB** ドライブへの転送（リムーバブル ドライブが `/dev/sdX` なら）：

```
$ dd if=boot.iso of=/dev/sdX
```

- **BIOS** で **USB** ドライブからの起動が有効になっていることを確認する
- **livecd-tools** や **liveusb-creator** などのツールを使うことで、この作業がとても簡単になり、ディスクのカスタマイズもできる

多くのディストリビューションは、ダウンロード用の `boot.iso` イメージファイルを提供しています。（名前は異なる場合があります）次のように **dd** を使用して、これを USB キードライブに置くことができます。

```
$ dd if=boot.iso of=/dev/sdX
```

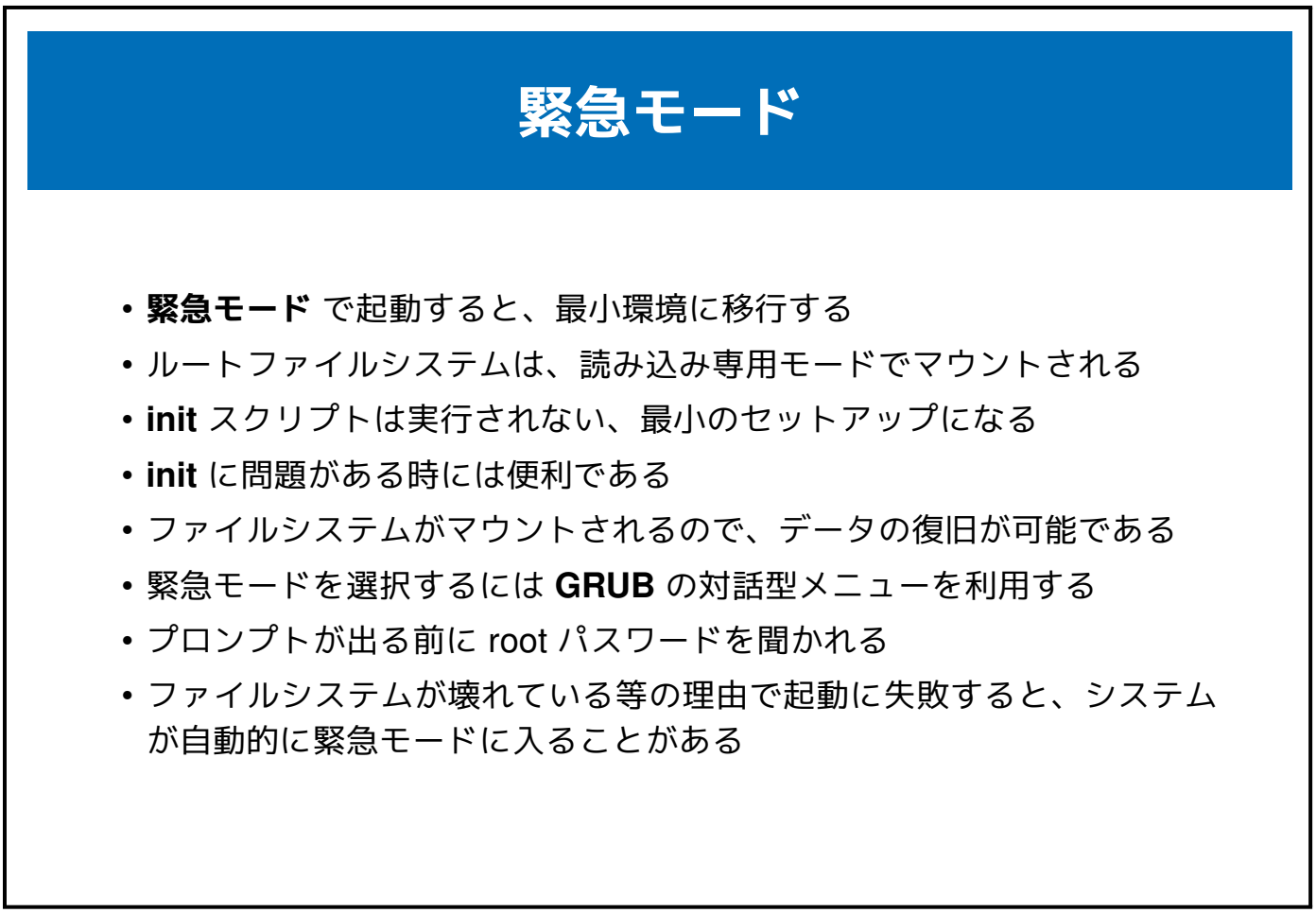
上記はシステムがリムーバブル ドライブを `/dev/sdX` として認識すると仮定しています。これによりドライブの既存のコンテンツが消去されることに注意してください！

システムに **USB** メディアから起動する機能があり、**BIOS** がそうするように構成されている場合、**USB** ドライブから起動できます。その後は、レスキュー **CD** や **DVD** と同じように機能します。ただし、**USB** ドライブにはインストールツリーが存在しないことに注意してください。したがってこの方法では、必要に応じてネットワークベースのインストールツリーが必要になります。

livecd-tools や **liveusb-creator** などの便利なユーティリティを使用すると、インストール イメージを取得する場所としてローカルドライブまたはインターネットのいずれかを指定できます。起動可能なイメージを作成してリムーバブル ドライブに書き込むという面倒な作業を全て実行できます。これは非常に便利であり、実際全ての **Linux** ディストリビューションで機能します。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

34.6 緊急モード



緊急モード では最小限の環境でブートします。ルート ファイルシステムは読み取り専用でマウントされ init スクリプトは実行されないのでほとんど何もセットアップされません。

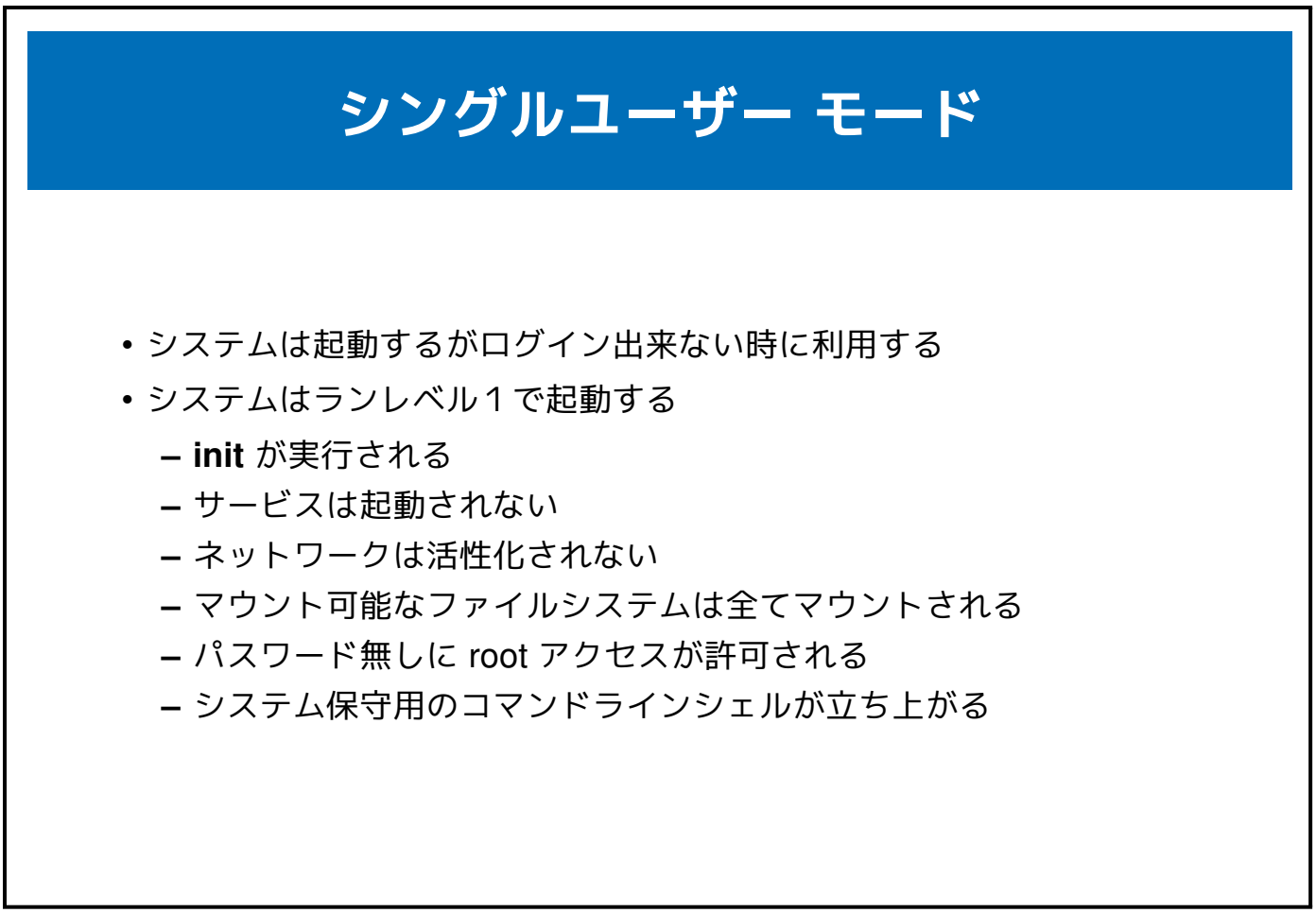
シングルユーザー モード（次で説明します）よりも緊急モードが優れている主な点は、**init** が破損しているか機能していない場合でもファイルシステムをマウントして再インストール中に失われたデータをリカバリできることです。

緊急モードに入るには **GRUB** ブートメニューからエントリを選択し `e` を押して起動メニューを編集する必要があります。そして、システムにブートを指示する前に `emergency` という単語をカーネルコマンドラインに追加します。シェルプロンプトを表示する前に root のパスワードの入力を求められます。

また、ファイルシステムの破損などさまざまな理由でブートが失敗した場合にも、緊急モードに入ることができます。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

34.7 シングルユーザー モード



システムブート時にブートが完了してもログインできない場合は **シングルユーザー モード** を試してください。

このモードではシステムは（**SysVinit** が採用されている場合には）**ランレベル** 1 で起動します。シングルユーザー モードは自動的にファイルシステムをマウントしようとするため、ルート ファイルシステムを正常にマウントできない場合、または **init** 構成が破損している場合は使用できません。

シングルユーザー モードで起動するには 1 つの例外を除いて緊急モードで説明したのと同じ方法を使用します。例外はキーワードを `emergency` から `single` に置き換えることです。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

34.8 演習

課題 34.1: 救出・復旧（レスキュー/リカバリ）メディアの準備

非常に重要

本演習では、意図的にシステムを破壊しレスキューメディアを使って復旧します。そのため、何か実行する前には必ずレスキューメディアから確実にブートできることを確認してください。そこで、まずレスキューメディアを確認します。それは、専用のレスキュー/リカバリメディアでも良いですし、光学ディスクや USB ドライブにインストールしたライブイメージでも構いません。

メディアからブートして、システムを強制的にレスキューメディアからブートする方法（**BIOS** の設定に詳しくなければなりません）、システムがブートした後にレスキューモードを選択できることを確認します。

注目

仮想マシンを利用しているときは、２つの違いを除いて論理的に同じ操作です。

- **BIOS** に入る方法は、使用しているハイパーバイザーによっては難しいものになります。あるものは素早くキー入力する必要があったりします。そのため、ドキュメントを読みやり方を確認しておきます。
- 仮想マシンがマウントできるように設定することで、物理的な光学ディスクやドライブを使用することもできます。**USB** を使用する場合は、もう１つやることがあります。仮想マシンが物理デバイスを認識できるようにすることです。`.iso` イメージ ファイルを直接仮想マシンに接続するのが容易な方法です。

仮想マシンで作業することで、危険性が少なくなります。修復不可能なやり方でシステムを壊す恐れがあるときは、本演習を始める前に仮想マシンのバックアップを取ることで、いつでもイメージを復元できます。

非常に重要

レスキュー/リカバリ メディアからブートできることを確認するまでは、以降の演習を始めないでください

課題 34.2: 破壊された GRUB 設定からの復旧

1. （ `/boot/grub/grub.cfg`、 `/boot/grub2/grub.cfg`、 `/boot/grub/grub.conf` ）の **GRUB** 設定ファイルを編集します。`kernel` 行の `UUID` フィールド中の最初の文字を削除します。レスキューモードで元に戻すため、消した文字を記録しておきます。（もし、ルート ファイルシステムをラベルまたはディスクデバイスノードで識別している場合も、同様の方法で行ってください）オリジナルのバックアップを取ってください。

   BLSCFG システムの場合

   かわりに `/etc/grub2/grubenv` を編集することで、コマンド行を破壊することもできます。

2. マシンをリブートしてください。システムは `No root device` was found メッセージを出して、ブートを停止します。`panic` occurred メッセージも出力されます。
3. マシンに **インストール** または **ライブ DVD CD** または **USB** ドライブを（または、動作しているインストール用のサーバーが利用できる場合にはネットワーク ブートメディアを）挿入ます。リブートします。ブートメニューが表示されたらレスキューモードを選択します。
4. 別の方法として、大半のディストリビューションが提供している **GRUB** メニューから、レスキューイメージを選択します。レスキューメディアと同様にできるでしょう。しかし、必ずしも動作するとは限りません。たとえば、ルート ファイルシステムが破壊されていれば、何もできません。

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5. レスキューモードでファイルシステム検索を求められたら、同意してください。プロンプトが出たら、シェルを開いて **mount** と **ps** のようなユーティリティを実行してレスキューシステムを詳しく調査します。
6. ファイルを編集したり、バックアップから戻したりして **GRUB** 設定を修正し、破壊されたシステムを修復します。
7. `exit` を入力してインストーラーに戻ります。ブートメディアを取り出します。リブートの手順を確認します。リブートします。通常通り立ち上がるはずです。

課題 34.3: パスワード失敗からの復旧

1. root（**sudo** ではなく）で、パスワードを変更します。新パスワードを知らないと仮定します。
2. ログアウトし、古いパスワードでログインします。当然、失敗します。
3. レスキューメディアでブートし、オプション選択時に `Rescue` を選択します。ファイルシステムをマウントし、コマンドラインシェルを起動します。
4. **chroot** します。（システムにいつも通りアクセスできます）

   ```
   $ chroot /mnt/sysimage
   ```

   root のパスワードを元にリセットします。
5. 終了し、レスキューメディアを取り出しリブートします。通常通りログインできます。

課題 34.4: パーティション テーブルの破壊から復旧

非常に重要

本演習はかなり危険で、システムを使用不能にします。内容を完全に理解してから、演習を開始してください。

注目

以下の方法は、**MBR** を採用したシステム用です。**GPT** を利用しているときは、パーティション分割の章で説明したように `--backup-file` と `--load-backup` オプションを指定した **sgdisk** を使います。

1. root でログインして **MBR** をセーブします。

   ```
   $ dd if=/dev/sda of=/root/mbrsave bs=446 count=1
   ```

   ```
   1+0 records in
   1+0 records out
   446 bytes (446 B) copied, 0.00976759 s, 45.7 kB/s
   ```

   注意してください: コマンドを正しく打ち込んだか確認してください、そして、セーブしたファイルの長さが正しいかも確認してください。

   ```
   $ sudo ls -l /root/mbrsave
   ```

   ```
   -rw-r--r-- 1 root root 446 Nov 12 07:54 mbrsave
   ```

2. **MBR** を次のように消去します。

   ```
   $ dd if=/dev/zero of=/dev/sda bs=446 count=1
   ```

   ```
   1+0 records in
   1+0 records out
   446 bytes (446 B) copied, 0.000124091 s, 3.6 MB/s
   ```

3. リブートします。失敗しますね。
4. レスキュー環境でリブートします。**MBR** を元に戻します。

   ```
   $ dd if=/mnt/sysimage/root/mbrsave of=/dev/sda bs=446 count=1
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

5. レスキュー環境を終了しリブートします。システムは元通りブートするはずです。

課題 34.5: インストール イメージを利用した復旧

注目

本演習は、特に **Red Hat** ベース システム用となっています。他のディストリビューション ファミリーでも、簡単に置き換えることができると思います。

1. **zsh** パッケージを削除します。(インストールされている場合)

   ```
   $ dnf remove zsh
   ```

   または、

   ```
   $ rpm -e zsh
   ```

   簡単にするために、他に依存しないパッケージを選択しました。もし他に依存するパッケージを選択した場合は、必要に応じて削除したパッケージを再インストールする必要があります。

2. レスキュー環境でブートします。
3. レスキュー環境内で、**zsh** を再インストール(またはインストール)します。まず、インストールメディアを `/mnt/source` にマウントします:

   ```
   $ mount /dev/cdrom /mnt/source
   ```

   パッケージを再インストールします:

   ```
   $ rpm -ivh --force --root /mnt/sysimage /mnt/source/Packages/zsh*.rpm
   ```

   `--force` オプションを指定し、**rpm** が依存情報他を指定ディレクトリから決定することを指定します。もし、システムが何度もアップデートされていてインストールイメージがかなり古くなっていた場合、本手順は失敗します!

4. 終了しリブートします。
5. **zsh** が再インストールされていることを確認します。

   ```
   $ rpm -q zsh
   ```

   ```
   zsh-5.0.2-7.el7.x86_64
   ```

6. ```
   $ zsh
   ```

   ```
   ....
   [coop@q7]/tmp/LFS201%
   ```


© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)
   ```

第 35 章

最後に

35.1 評価サーベイ

評価サーベイ

Linux Foundation の本講座を受講いただき、ありがとうございます。

皆さまのニーズを十分に満たすことができたかを知り、今後の講座を改善するため、皆さまのコメントは非常に重要であり、真摯に受け止めてまいります。

- 受講いただいた講座を評価してください。インストラクターがリンクを示します。
- お名前など入力する情報に間違えの無いようご注意ください。
- この入力情報を元に **修了書** を生成し、ご提示いただいたメールアドレスにお送りします。

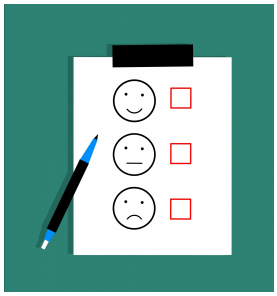


図 35.1: **サーベイ**

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)

© Copyright The Linux Foundation 2023. All rights reserved.(無断転載・無断複製禁止)