Linuxシステムの運用管理の基本を学べる

Linux
システム管理標準教科書
Ver. 2.0.0

LinuC
Linux技術者認定
https://linuc.org

open your NEXT future
LPI-JAPAN
Linux Professional Certification
https://lpi.or.jp

Linuxシステム管理標準教科書

2025年9月16日　Ver.2.0.0
LPI-Japan　発行

まえがき

このたび、特定非営利活動法人エルピーアイジャパンは、Linux 技術者教育に利用していただくことを目的とした教材、「Linux システム管理標準教科書」をインターネット上にて公開し、提供することとなりました。

この「Linux システム管理標準教科書」は、既に公開し、多くの教育機関や学習者に活用されている「Linux 標準教科書」「Linux サーバー構築標準教科書」の続編として作成されました。本教科書では、Linux を扱うシステム管理者が知っておきたい運用管理の基本を、実習を交えて理解することを目的にしています。2024 年から始まった全面改訂プロジェクトに従い、本教科書も最新の Linux ディストリビューションに対応する形で 2015 年にリリースされたバージョン 1 の改訂を行いました。

公開にあたっては、「Linux システム管理標準教科書」に添付されたライセンス（クリエイティブ・コモンズ・ライセンス）の下に公開されています。

本教科書の最新情報は以下の Web ページをご参照ください。

`https://linuc.org/textbooks/admin/`

本教科書の目的

本教科書の目的は、実習を通して LinuC 試験に含まれる Linux システムの運用管理に関わる知識を学習することにあります。実際にユーザーやファイル、ディレクトリを作成し、Web サーバーなどを動作させながら、運用管理に関する基本的な技術を修得できます。

本教科書でカバーされない範囲

本教科書は実務で必要となる Linux システム管理の実践的なスキルに絞って解説しており、LinuC レベル 1、レベル 2 の試験範囲に含まれるすべてのシステム管理に関わる事項について解説していません。本教科書で触れられていない範囲については、別途調べてみて理解を深めてください。

想定している実習環境

本教科書での実習環境として、以下の環境を想定しています。

学習者

学習者は 1 名で実習を進めることができます。

実習環境

「Linux 標準教科書」「Linux サーバー構築標準教科書」同様、仮想マシンを使った実習環境を想定しています。実習環境の準備については、両教科書を参照して行ってください。

本教科書で使用しているソフトウェア

仮想マシンは VirtualBox for Windows を使用しています。使い慣れているソフトウェアがあれば、そちらを使っても構いません。

Linux ディストリビューションは、AlmaLinux 9.6（x86 64 ビット）を使用しています。

本教科書が想定しているネットワーク

ネットワークは、インターネットに接続できることを前提にしています。

インターネットに接続できない場合、ソフトウェアの追加インストールなどいくつかの実習が行えない場合があります。その場合の対処方法については 5 章で解説しています。

全体の流れ

本教科書では、以下の通りに実習を進めます。

第 1 章ユーザーとグループの管理

Linux システムの基本であるユーザーとグループの管理について基本をしっかりと確認します。セキュリティにも関わるため、SSH によるリモートログインと root ユーザーについても解説します。

第 2 章ネットワークの管理

ネットワークの管理について解説します。セキュリティとしてパケットフィルタリングについても解説します。

第 3 章サービスの管理

systemd のサービス管理について解説します。また、NTP による時刻設定についても触れています。

第 4 章ファイルシステムの管理

ファイルシステムのアクセス権管理について解説します。アクセス制御の観点から、SELinux についても解説します。ファイルシステムの基盤であるストレージ管理、LVM（Logical Volume Manager）も解説します。

第 5 章システムのメンテナンス

システムのメンテナンスの基本であるパッケージ管理について解説します。また、システム監視についても解説します。

第 6 章トラブルシューティング

基本的なトラブルシューティングの手法について解説します。

後の章では、それよりも前の章での実習を前提に実習が進められている場合があります。実習を省略した場合、実習結果が異なる場合があります。また、ファイアーウォールや SELinux などセキュリティに関する設定について理解ができていることを想定しています。正常に実習が行えない場合には、「第 6 章トラブルシューティング」の内容を参考にしてみてください。

利用上の注意

本教科書は、Markdown 形式で執筆を行い、Pandoc を使用して PDF 形式および EPUB 形式に変換しています。変換ツールの使用上、以下のような問題が判明しています。可能な限り対応していますが、技術的な制約のため問題が残ってしまっている点に注意してご利用ください。

-（ハイフンが）2 つ続く場合の問題

本文中で-が 2 つ続くと、長い – になってしまう問題が確認されています。Web ブラウザ上での取り扱いでは回避方法が確認されていますが、Pandoc による変換では正常に動作しないため、コマンドのオプションやパーミッション表記などで-を 2 つ記述できません。この問題を回避するため、-と-の間にスペースを入力しています。見た目が少しおかしくなっていますが、この回避方法によるものです。

コマンド実行例などで空白が入ってしまう問題

コマンド実行例や URL などで空白が入ってしまう問題が確認されています。-や/などの記号が含まれているとその前に空白が入るため、コピー＆ペーストで実行しようとすると正常に実行できない、URL をクリックして Web ブラウザにアクセスできないなどの問題が発生します。発生する理由が不明のため、問題を解決、回避できていません。このような問題が起きた場合には、コピー＆ペーストする際に手動で空白を取り除いてください。

執筆者・制作者紹介

本教科書は、オープンなプロジェクト形式で開発を行っています。企画段階から意見交換を行い、事前の技術的な調査、執筆、レビューなどをプロジェクトのメンバーで分担して行っています。

宮原徹（バージョン 2 全体執筆担当）

本教科書は、Linux/オープンソースソフトウェアをこれから勉強する皆さんと、熱心に指導に当たられている先生方の一助になればと思い、作成いたしました。バージョン 2 の改訂にあたっては、新しいディストリビューションへの対応だけでなく、より分かりやすい実習になるように調整を行ってみました。また、バージョン 1 の記載から実務で不要と思われる項目については思い切って削除し、見通しのよい構成に変更してみました。このレベルでは技術が複合的になるため、一本道で解説するのが難しい面があります。一度通しで目を通した後、2 回、3 回と実際に手を動かしてみることで技術相互の関係性が理解できるでしょう。

バージョン 1 執筆者

バージョンアップに伴う変更がなかった解説は、以下の方々の執筆によるものです。

- 平 愛美（バージョン 1 : 1 章〜3 章執筆担当）
- 面 和毅（バージョン 1 : 4 章〜6 章執筆担当）

バージョン 2 の開発にご協力をいただいた方々（50 音順）

- 秋庭 雅明（データプロセス株式会社）
- 榎本 浩義（弓削商船高等専門学校）
- 上堂薗 健（DaaS 合同会社）
- 河原木 忠司
- 坂井 麻衣
- 島ノ江励（フューチャー株式会社）
- 竹本 季史（インターノウス株式会社）
- 平川 雄一（株式会社テクノプロ　テクノプロ・エンジニアリング社）
- 福永 昭臣（株式会社ボールド）
- 宮本 清正（日本アイ・ビー・エム株式会社）
- 山口昌幸（NEC ソリューションイノベータ株式会社）

著作権

使用に関する権利

図 1: CC BY-NC-ND 4.0

表示

本教科書は、特定非営利活動法人エルピーアイジャパンに著作権が帰属するものであることを表示してください。

非営利

本教科書は、非営利目的で教材として自由に利用することができます。

商業上の利得や金銭的報酬を主な目的とした営利目的での利用は、特定非営利活動法人エルピーアイジャパンによる許諾が必要です。ただし、本教科書を利用した教育において、本教科書自体の対価を請求しない場合は、営利目的の教育であっても基本的に使用できます。その場合も含め、LPI-Japan 事務局までお気軽にお問い合わせください。

＊営利目的の利用とは以下のとおり規定しております。営利企業または非営利団体において、商業上の利得や金銭的報酬を目的に本教科書の印刷実費以上の対価を受講生に請求して本教科書の複製を用いた研修や講義を行うこと。

改変禁止

本教科書は、改変せず使用してください。本教科書に対する改変は、特定非営利活動法人エルピーアイジャパンまたは特定非営利活動法人エルピーアイジャパンが認める団体により行われています。

フィードバック

フィードバックは誰でも参加できる Slack で受け付けていますので、積極的にご参加ください。Slack 参加の詳細は以下の本教科書の Web ページを参照してください。

`https://linuc.org/textbooks/admin/`

図 2: https://linuc.org/textbooks/admin/

本教科書の使用に関するお問合せ先

特定非営利活動法人エルピーアイジャパン（LPI-Japan）事務局
〒 100-0011 東京都千代田区内幸町 2-1-1 飯野ビルディング 9 階
TEL：03-6205-7025
E-Mail：info@lpi.or.jp

Linux 技術者認定「LinuC（リナック）」のご紹介

Linux 技術者認定「LinuC（リナック）」とは、クラウド／ DX 時代の IT エンジニアに求められるシステム構築から運用管理に必要なスキルを証明できる技術者認定です。アーキテクチャ設計からシステム構築、運用管理までの技術領域を広くカバーしており、４つのレベルの認定取得を通じて一歩ずつ確実に求められるスキルを習得し、それを証明することができます。

LinuC の出題範囲策定や試験開発は、実際に現場で活躍しているハイレベルな IT エンジニアが参加するコミュニティによって行われています。そのため、グローバルで業界標準として利用されている技術領域をカバーし、システム開発や運用管理の現場で本当に必要とされる知識や実践的なスキルを問う内容になっています。その結果として従来型の Linux 領域にとどまった技術認定とは異なり、国内・海外を問わず活躍を目指す IT エンジニアにとって、実践的かつ有用な技術者認定となっています。

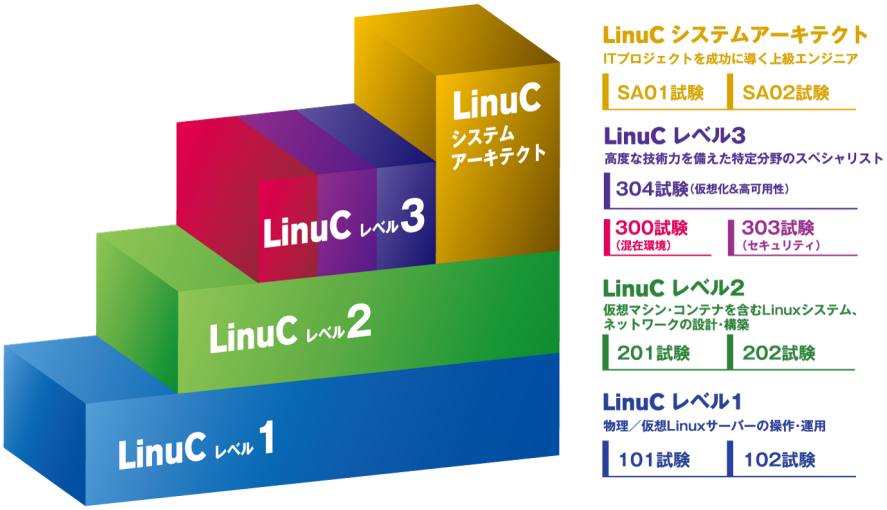

図 3: LinuC の体系図

LinuC レベル 1

コンピュータシステムを理解し、仮想環境を含む Linux システムの基本操作とシステム管理が行える即戦力エンジニアの証明（ITSS レベル 1）

LinuC レベル 2

仮想環境を含む Linux のシステム設計・ネットワーク構築において、アーキテクチャに基づいた設計・導入・保守・問題解決ができるエンジニアの証明（ITSS レベル 2）

LinuC レベル 3

異種混在環境の運用スキル、Linux ベースのセキュアなシステム設計・構築スキル、大規模な仮想化システムや高可用性システムの構築スキルといったスペシャリストの証明（ITSS レベル 3）

LinuC システムアーキテクト

オンプレ／クラウド、物理／仮想化を含むシステムのライフサイクル全体を俯瞰して最適なアーキテクチャを設計・構築ができる上級エンジニアの証明（ITSS レベル 4 相当のスキルレベル）

LinuC の詳細については、以下の Web サイトを参照してください。

`https://linuc.org/about/01.html`

図 4: https://linuc.org/about/01.html

LinuC の認定取得に向けた学習方法

LinuC の特徴の 1 つは学習環境が充実していることです。LinuC の認定取得に必要となるスキルや知識を身につけるには　体系的な学習として技術解説書を読むだけではなく、実際に手を動かして確認し問題集を繰り返し解いて理解を定着させる方法を推奨しています。

体系的にしっかりとした教育を実施している認定校や認定教材のほか、様々な学習コンテンツや無料で参加できるセミナーなどが豊富に用意されています。

詳しくは以下のリンク先「学習のすすめ方」をご覧ください。

`https://linuc.org/measures/`

図 5: https://linuc.org/measures/

LinuC の出題範囲を網羅した LPI-Japan 認定教材については以下のリンク先「認定教材」をご覧ください。

`https://linuc.org/measures/textbook/`

図 6: https://linuc.org/measures/textbook/

目次

1 ユーザーとグループの管理

1.1 ユーザーの管理

Windows や macOS などの「デスクトップ OS」では、基本的に 1 人のユーザーが 1 台のマシンを占有して利用します。

それに対して、Linux などの「サーバー OS」では、複数のユーザーが同時に利用できるように最適化された環境になっています。そのため、管理者は利用者一人一人にユーザーアカウントを作成し、ユーザーは自分専用のユーザーアカウントでログインする仕組みとなっています。

1.1.1 ユーザーアカウントの種類

ユーザーアカウントの種類には、一般アカウントの「一般ユーザー」と、管理者権限である「root ユーザー」(スーパーユーザー)、そしてアプリケーションで利用される「システムユーザー」が存在します。

ユーザーの種類	用途
一般ユーザー	ユーザー個人のアカウント
root ユーザー (スーパーユーザー)	管理者権限を持つユーザー
システムユーザー	システムやアプリケーションで使用するユーザー

1.1.2 一般ユーザー

一般ユーザーは、通常のユーザーがサーバーにログインして利用するためのアカウントです。

ユーザー情報の確認は、id コマンドを使います。以下の例では、OS インストール時に作成したユーザー linuc でログインした後、id コマンドでユーザー情報を確認しています。

```
$ id
uid=1000(linuc) gid=1000(linuc) groups=1000(linuc),10(wheel)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

uid はユーザーを識別するためのユーザー ID、gid は主グループ ID を示しています。所属グループ（groups）には所属している主グループ ID およびサブグループ ID が表示されます。

一般ユーザーの uid は、AlmaLinux では 1000〜65535 が使用されます。値はディストリビューションによって範囲が異なります。

1.1.3 root ユーザー

root ユーザーは、管理者権限を持っている特別なユーザーで uid には 0 が付与されています。スーパーユーザーとも呼ばれます。

root ユーザーになるには、主に以下の方法があります。

Linux のローカルコンソールから root ユーザーでログインする　Linux の動作しているマシンを直接操作できる時には、ローカルコンソールから root ユーザーでログインできます。

一般ユーザーでログインした後、su コマンドを実行する　一般ユーザーから root ユーザーになるには、su コマンドを使います。su コマンドを実行するときに-（ハイフンのみ）オプションを付けると、そのユーザーでログインしたのと同じことになります。ただし、root ユーザーにパスワードが設定されていないと行えません。あるいは、sudo コマンド（後述）を使って su コマンドを実行することも可能です。

以下の例は、sudo コマンドを使って su コマンドを実行した例です。

```
$ sudo su -
[sudo] linuc のパスワード:
#
```

プロンプトが root ユーザー用の「#」に変わったのが分かります。id コマンドで、ユーザー識別子を表示します。

```
# id
uid=0(root) gid=0(root) groups=0(root)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

root ユーザーの uid は必ず 0 になります。

root ユーザーは一般ユーザーのように常用してはいけません。何故なら、入力したコマンドの間違い等のオペレーションミスで、システム障害に繋がる危険性があるからです。root ユーザーの権限が必要なコマンドの実行は、次に解説する sudo コマンドを使用します。

1.1.4 一般ユーザーが sudo コマンドを実行する

sudo コマンドを使うと、一般ユーザーはコマンドを root 権限で実行できます。特定のユーザーやグループに対して、特定のコマンドのみ実行できるように設定するなど細かい制御ができます。su コマンドと異なり、sudo コマンドの実行には、実行したユーザーのパスワードを入力して認証を行う必要があります。

sudo コマンドは実行履歴がログに残るので、後からいつ、誰が、どのコマンドを実行したのか調査できる点にメリットがあります。また、実行時の認証が実行ユーザーのパスワードなので、root ユーザーのパスワードを共有したり、管理したりする必要がなくなります。

本教科書では、原則 sudo コマンドを使って実習を進めていきます。

1.1.5 システムユーザー

システムユーザーは、バックグラウンドで動くサービスが使用するユーザーです。AlmaLinux では uid の 1 から 999 がシステムユーザー用に予約されています。一般ユーザーのようにログインして利用するユーザーではありません。

たとえば、SSH を動作させるためには、「sshd」ユーザーがシステム内部で sshd デーモンを実行しています。root ユーザーは id コマンドを使って、他のユーザーの情報を確認できます。sshd サービスのシステムユーザーの情報を確認しましょう。

```
$ id sshd
uid=74(sshd) gid=74(sshd) groups=74(sshd)
```

1.1.6 useradd コマンドによる一般ユーザーの作成

新規に一般ユーザーを作成するには、root ユーザーで useradd コマンドを実行します。useradd コマンドの引数にユーザー名を指定します。

ユーザーを作成後、必要に応じて passwd コマンドでパスワードを設定します。

以下の例では、-c オプションでコメントを入れています。

```
$ sudo useradd -c "Ichiro Suzuki" suzuki
$ id suzuki
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki)
```

useradd コマンドの主なオプションは以下の通りです。

オプション	説明
-u	ユーザー ID を指定する
-g	主グループ名または主グループ ID を指定する
-G	サブグループを指定する。複数指定するときはカンマ (,) で区切る
-s shell	ユーザーのログインシェルを指定する
-c コメント	コメントを入れる（ユーザーのフルネームなど）
-d ディレクトリ名	ホームディレクトリを指定する
-e YYYY-MM-DD	ユーザーアカウントが無効になる年月日を指定する

1.1.7 パスワードの設定

passwd コマンドを使って、引数で指定したユーザーのパスワードを設定します。

```
$ sudo passwd suzuki
ユーザー suzuki のパスワードを変更。
新しい パスワード:
新しい パスワードを再入力してください:
passwd: すべての認証トークンが正しく更新できました。
```

管理者である root ユーザーがパスワードを設定した場合、該当のユーザーがログインしてパスワードを変更することが推奨されます。この時、既存のパスワードと新しく設定するパスワードの両方を聞かれます。新しく設定するパスワードは、誕生日や名前など推測されやすいパスワードにしないように注意しましょう。

以下の例では、ユーザー suzuki でログインした後、パスワードを変更しています。

```
[suzuki@vbox ~]$ passwd
ユーザー suzuki のパスワードを変更。
Current password: ※ユーザーsuzukiの現在のパスワードを入力
新しい パスワード: ※ユーザーsuzukiの新しいパスワードを入力
新しい パスワードを再入力してください:
    ※ユーザーsuzukiの新しいパスワードを再入力
passwd: すべての認証トークンが正しく更新できました。
```

1.1.8 ユーザー情報の確認

ユーザー情報は/etc/passwd ファイルに保管されています。ファイルの内容を閲覧する cat コマンドを使って /etc/passwd ファイルを確認します。

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
（略）
linuc:x:1000:1000:LinuC:/home/linuc:/bin/bash
suzuki:x:1001:1001:Ichiro Suzuki:/home/suzuki:/bin/bash
```

/etc/passwd は左から順に以下の情報が入っていて、コロン (:) で区切られています。

項目	意味
ユーザー名	ユーザーアカウント名
パスワード	x はシャドウパスワードが設定されている
ユーザー ID	ユーザー ID
グループ ID	グループ ID
コメント	ユーザーに関するコメント
ホームディレクトリ	ユーザーのホームディレクトリ
シェル	ログインした時に起動するシェル

1.1.9 シャドウパスワードについて

かつての UNIX では、ユーザーのパスワードを暗号化（ハッシュ化）して/etc/passwd に記録していました。しかし、/etc/passwd は誰でも内容を読むことができるファイルのため、一般ユーザーにパスワードを解析されてしまう危険性がありました。

そのため、root ユーザーのみ読み取れるシャドウファイル (/etc/shadow) を用意し、暗号化（ハッシュ化）したパスワードを別途格納しています。暗号化は元に戻すこと（復号）ができますが、ハッシュ化は元に戻すことができない（困難）な仕組みです。パスワードがシャドウファイルに格納されていると、/etc/passwd のパスワード部分には x が入るようになっています。

/etc/shadow のパーミッションは 000、あるいは 400 に設定されています。root ユーザーはパーミッションに関係なく読み取ることができますが、その他のユーザーは読み書きすることができません。パーミッションの詳細については後述します。

```
$ ls -l /etc/shadow
----------. 1 root root 1157  7月 20 20:48 /etc/shadow
```

root ユーザーで、ユーザー suzuki のシャドウパスワードを確認してみましょう。

```
$ sudo cat /etc/shadow | grep suzuki
suzuki:$6$rounds=100000$LJ9X08ZYN/zXIRaf$MlFDUV8wemu1kcBDCmfR0kt0d7thdFo7Y9dZ9sK
HlG6ua8TrIcMmrHkN3IGdSwFtHaW1sKdNINZtAXNAlmzl.:20289:0:99999:7:::
```

ユーザー名の後にある文字列がパスワードをハッシュ化したものです。$で区切られており、それぞれの項目の意味は以下の通りです。

項目	意味
6	ハッシュ方式。6 は SHA-512
rounds=100000	パスワードのハッシュ化を繰り返す回数。総当たり攻撃の速度を遅らせる役割を持っている
LJ9X08ZYN/zXIRaf	ソルトと呼ばれる入力されたパスワードの文字数を増やす値
MlF…mzl.	ハッシュ化されたパスワード

1.2 グループの管理

ユーザーを管理するひとつの単位として、グループ機能があります。このグループでは、所属する部署やユーザーの役割などによってユーザーにグループを割当て、グループ単位での適切な管理が行えます。

たとえば、特定のディレクトリ配下へのアクセスや、特定のファイルの読み書きを特定のグループのみに制限したいとき、「パーミッション」という機能を使って管理できます。

1.2.1 主グループとサブグループ

ユーザーは、1 つ以上のグループに所属している必要があります。ユーザーが最初に所属するグループを「主グループ」(またはプライマリーグループ) といいます。所属できる主グループは一つのみで、複数のグループにユーザーを所属させたい場合はサブグループを割り当てます。

```
$ id suzuki
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki)
```

gid が主グループです。所属グループはユーザーが所属しているすべてのグループが表示されます。

1.2.2 /etc/group を確認する

グループの設定は/etc/group ファイルに格納されていますので、確認してみましょう。

```
$ cat /etc/group
root:x:0:
bin:x:1:
 （略）
linuc:x:1000:
suzuki:x:1001:
```

useradd コマンドは、主グループの指定が無かった場合には作成するユーザーと同じ名前のグループを作成し、そのグループをユーザーの主グループに指定します。グループ ID も、作成するユーザーの uid と同じになります。

1.2.3 グループの作成

groupadd コマンドでグループを作成します。グループ ID を指定したい時は-g オプションを使います。グループ ID が指定されなかった時には、自動的に割り当てられます。

groupadd コマンドの書式は以下の通りです。

```
groupadd [-g グループID] グループ名
```

グループ ID 5000 を指定して grouptest というグループを作成します。

```
$ sudo groupadd -g 5000 grouptest
```

/etc/group ファイルを確認します。

```
$ cat /etc/group | grep grouptest
grouptest:x:5000:
```

1.2.4 グループ名の変更

groupmod コマンドでグループ名を変更します。

groupmod コマンドの書式は以下の通りです。

```
groupmod [-n 新しいグループ名] グループ名
```

グループ名を grouptest から eigyou に変更します。

```
$ sudo groupmod -n eigyou grouptest
```

/etc/group ファイルを確認します。

```
$ cat /etc/group | grep eigyou
eigyou:x:5000:
```

同じグループ ID で、グループ名が eigyou に変更になりました。

1.2.5 ユーザーをサブグループに所属させる

ユーザーをサブグループに所属させるときは、usermod コマンドを使います。-G（大文字）オプションで所属させたいサブグループをカンマ区切りですべて指定します。所属していたサブグループの指定を忘れると、そのサブグループの所属から外れてしまうので注意してください。所属しているサブグループが多い場合には、-a オプションを使って所属サブグループを追加する指定を行うか、後述する gpasswd コマンドを使用してください。

usermod コマンドの書式は以下の通りです。

```
usermod [-G サブグループ名 [,...]] [-a] ユーザー名
```

ユーザー suzuki の所属するサブグループとして eigyou グループを指定します。

```
$ sudo usermod -G eigyou suzuki
```

id コマンドでサブグループが追加されているか確認します。

```
# id suzuki
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki),5000(eigyou)
```

所属グループに eigyou サブグループが追加されているのが分かります。

1.2.6 サブグループの所属ユーザーを確認する

グループにサブグループとして所属しているユーザーは、/etc/group に記述されています。

```
$ cat /etc/group
（略）
suzuki:x:1001:
eigyou:x:5000:suzuki
```

ユーザー suzuki が、eigyou グループにサブグループとして所属しているのが分かります。

1.2.7 gpasswd コマンドを使ってサブグループを管理する

ユーザーの所属しているサブグループが多い場合、gpasswd コマンドを使うのが便利です。gpasswd コマンドは、追加や除外したいサブグループを 1 つだけ指定できます。

gpasswd コマンドの書式は以下の通りです。

```
gpasswd -a 追加するユーザー名 グループ名
gpasswd -d 除外するユーザー名 グループ名
```

ユーザー suzuki の所属するサブグループから eigyou グループを除外します。

```
$ sudo gpasswd -d suzuki eigyou
ユーザ suzuki をグループ eigyou から削除
$ id suzuki
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki)
```

/etc/group を確認して、eigyou グループからユーザー suzuki が削除されていることを確認します。

```
$ cat /etc/group
（略）
suzuki:x:1001:
eigyou:x:5000:
```

eigyou グループからユーザー suzuki が除外されているのが分かります。

再度、ユーザー suzuki の所属するサブグループに eigyou グループを指定します。

```
$ sudo gpasswd -a suzuki eigyou
ユーザ suzuki をグループ eigyou に追加
$ id suzuki
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki),5000(eigyou)
```

ユーザー suzuki の所属グループに eigyou グループが追加されました。

1.3 パーミッションを使ったファイルシステムのアクセス管理

ファイルシステムのアクセス管理を行うには、「パーミッション」という機能を利用します。パーミッション (permission) とは、日本語で「許可」を意味します。

許可されたユーザーやグループのみが特定のファイルやディレクトリにアクセスでき、読み取り・書き込み・スクリプト等の実行ができるように設定できます。

1.3.1 パーミッションの確認

ユーザーのホームディレクトリに移動します。cd コマンドにオプションや引数を付けなければ、ユーザーのホームディレクトリに移動できます。ホームディレクトリとは、ユーザー個人に割り当てられた領域になり、ここにユーザーが作ったファイルやプログラムを保存したり、ユーザー独自の設定ファイルを格納したりしています。

pwd コマンドで現在のディレクトリがホームディレクトリになっていることを確認します。

```
$ cd
$ pwd
/home/linuc
```

次に touch コマンドで、空のファイルを作ります。

```
$ touch test.txt
```

ls コマンドに、-l オプションを付けて実行し、ファイルの詳細を表示します。

```
$ ls -l test.txt
-rw-r--r--. 1 linuc linuc 0  7月 20 21:54 test.txt
```

ファイル表示の一番左にある「-rw-r- -r- -」がパーミッションです。

なお、AlmaLinux の環境では、ll コマンドでも「ls -l」と同様の結果が得られます。「ll」は「ls -l」のエイリアスとなるように設定されています。

```
$ ll test.txt
-rw-r--r--. 1 linuc linuc 0  7月 20 21:54 test.txt
```

設定されているエイリアスは alias コマンドで確認できます。

```
$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias xzegrep='xzegrep --color=auto'
alias xzfgrep='xzfgrep --color=auto'
alias xzgrep='xzgrep --color=auto'
alias zegrep='zegrep --color=auto'
alias zfgrep='zfgrep --color=auto'
alias zgrep='zgrep --color=auto'
```

1.3.2 パーミッションの表記方法

パーミッションは rwx の 3 つの文字で表されます。それぞれの意味は以下の通りです。また、各パーミッションは数値で表記することもできます。数値表記は chmod コマンドなどで使用されます。

意味	文字表記	数値表記
読取許可 (Readable)	r	4
書込許可 (Writable)	w	2
実行許可 (eXecutable)	x	1
何も許可しない	-	0

上記 test.txt のアクセス権限は以下のようになっています。

	所有ユーザー	所有グループ	その他
文字表記	rw-	r- -	r- -
数値表記	4+2+0=6	4+0+0=4	4+0+0=4

先頭から、所有ユーザー（user）、グループ (group)、どちらにも該当しないその他のユーザー (other) のアクセス権限を示しています。

- 先頭の「rw-」が所有ユーザー、すなわちユーザー linuc の権限で、読取・書込ができます。
- 次の「r- -」が所有グループ、すなわち linuc グループの権限で、読み取りのみ可能です。
- 最後の「r- -」がユーザーにもグループにも該当しないその他のユーザーの権限で、読み取りのみ可能です。

1.3.3 パーミッションの数値表記

パーミッションの数値表記は、設定したい権限に対応する数値の合計値を、所有ユーザー、所有グループ、その他の順に並べた 3 桁の数値で表されます。

たとえば、上記の例の「rw-r- -r- -」というパーミッションを数字で表記すると「644」になります。

1.3.4 ディレクトリのパーミッション

ディレクトリのパーミッションも、基本的な考え方はファイルのパーミッションと同じです。異なる点として、実行権限が無いとそのディレクトリに移動してカレントディレクトリにすることができません。

mkdir コマンドで testdir という新規ディレクトリを作成し、アクセス権限を変更してみます。

```
$ mkdir testdir
$ ls -ld testdir
drwxr-xr-x. 2 linuc linuc 6  7月 20 22:01 testdir
```

ls コマンドの-d オプションは、ディレクトリそのもののパーミッションなどを確認します。-d オプションが無いと、引数で指定されたディレクトリ内のファイルを表示しようとします。

```
$ ls -l testdir
合計 0
```

testdir ディレクトリのパーミッション表記の先頭にディレクトリを識別する d が付いていることが確認できます。このディレクトリのパーミッションは、rwx(4+2+1)、r-x(4+0+1)、r-x(4+0+1) で 755 になっています。suzuki グループ所属のユーザーや、その他のユーザーはこのディレクトリにアクセスできますが、書き込みは行えません。

パーミッションの変更は chmod コマンドを使用します。chmod コマンドの書式は以下の通りです。

```
chmod モード ファイル
```

モードの指定は文字表記、数値表記の両方が行えます。数値表記は指定された値に設定しますが、文字表記は+と-でパーミッションの付与、または解除を指定します。

以下、文字表記でのモード指定の例です。

モード指定	意味
ug+x	ユーザーとグループに実行権限を付与
a+x	すべてのユーザーに実行権限を付与
g-w	グループの書き込み権限を解除

以下の例では、chmod コマンドでディレクトリのパーミッションからユーザー自身の実行権限を解除することで、カレントディレクトリにできなくなることを確認しています。

```
$ chmod u-x testdir
$ ls -ld testdir
drw-r-xr-x. 2 linuc linuc 6  7月 20 22:01 testdir
$ cd testdir
-bash: cd: testdir: 許可がありません
$ chmod u+x testdir
$ cd testdir
$ pwd
/home/linuc/testdir
```

1.3.5 ユーザーアカウントの有効期限を設定する

ユーザーアカウントが使用できる有効期限を設定できます。たとえば、期限が決まっているプロジェクトなど、ユーザーアカウントの使用が期間限定の場合に有効期限を設定します。

新規アカウント追加時には useradd コマンド、すでに存在するアカウントの場合には usermod コマンドに-e オプションを付与して有効期限を指定できます。

ユーザーアカウントの有効期限設定の書式は以下の通りです。

```
useradd -e YYYY-MM-DD ユーザー名
usermod -e YYYY-MM-DD ユーザー名
```

usermod コマンドで既存ユーザーアカウントの有効期限を設定します。ユーザーアカウントに有効期限を設定すると、設定日にアカウントがロックされて使用できなくなります。

以下の例では、動作確認のために有効期限を本日の日付で設定しています。

```
$ sudo usermod -e 2025-7-21 suzuki
```

アカウント有効期限を確認するために、chage コマンドを使います。-l オプションを付与して実行することで、引数で指定したユーザーの各種有効期限などが表示されます。

```
$ sudo chage -l suzuki
最終パスワード変更日                    : 7月 20, 2025
パスワード期限:                          : なし
パスワード無効化中                       : なし
アカウント期限切れ                       : 7月 21, 2025
パスワードが変更できるまでの最短日数      : 0
パスワードを変更しなくてよい最長日数      : 99999
パスワード期限が切れる前に警告される日数  : 7
```

確認のため、アカウント有効期限を設定したユーザーアカウントでログインします。「Your account has expired」と表示され、アカウントがロックされている状態になっています。

```
login: suzuki
Password: ※ユーザーsuzukiのパスワードを入力
Your account has expired; please contact your system administrator
```

有効期限をリセットして無期限有効にするには、以下のように「''」（シングルクォート 2 つ）で空の有効期限を指定します。アカウント期限切れがなしに設定されます。

```
$ sudo usermod -e '' suzuki
$ sudo chage -l suzuki
最終パスワード変更日                    : 7月 20, 2025
パスワード期限:                          : なし
パスワード無効化中                       : なし
アカウント期限切れ                       : なし
パスワードが変更できるまでの最短日数      : 0
パスワードを変更しなくてよい最長日数      : 99999
パスワード期限が切れる前に警告される日数  : 7
```

1.3.6 パスワードの有効期限を設定する

ユーザーのパスワードの有効期限を設定したいときは、chage コマンドを使います。-M オプションでパスワードの有効な日数を指定します。

以下の例ではパスワードの有効日数を 30 に設定しているので、30 日毎にパスワードを再設定する必要があります。

```
$ sudo chage -M 30 suzuki
```

パスワードの有効期限を確認します。パスワード期限で表示された日付の翌日以降になると、ユーザーログイン時、強制的にパスワードの変更要求を行います。

```
$ sudo chage -l suzuki
最終パスワード変更日                    : 7月 21, 2025
パスワード期限:                          : 8月 20, 2025
パスワード無効化中                       : なし
アカウント期限切れ                       : なし
パスワードが変更できるまでの最短日数      : 0
パスワードを変更しなくてよい最長日数      : 30
パスワード期限が切れる前に警告される日数  : 7
```

パスワードの有効期限を即座に失効させるには、-d オプションで 0 を指定します。このオプションは、パスワードが最後に変更された日付の値を 1970 年 1 月 1 日に設定し、即座にパスワードを失効させ、ユーザーログイン時に強制的にパスワード変更を要求できます。

```
$ sudo chage -d 0 suzuki
```

chage コマンドでアカウントの情報を確認してみると、最終パスワード変更日、パスワード期限、パスワード無効化中の値が「パスワードは変更しなければなりません」になっていることが分かります。

```
$ sudo chage -l suzuki
最終パスワード変更日                    :パスワードは変更しなければなりません
パスワード期限:                          : パスワードは変更しなければなりません
パスワード無効化中                       : パスワードは変更しなければなりません
アカウント期限切れ                       : なし
パスワードが変更できるまでの最短日数      : 0
パスワードを変更しなくてよい最長日数      : 30
パスワード期限が切れる前に警告される日数  : 7
```

確認のため、設定したユーザーアカウントでログインします。即座にパスワード再設定が要求されます。以下の実行例では、ログインの代わりに su コマンドを使用しています。

```
[linuc@vbox ~]$ su - suzuki
You are required to change your password immediately (administrator enforced).
Current password: ※ユーザーsuzukiの現在のパスワードを入力
新しい パスワード: ※ユーザーsuzukiの新しいパスワードを入力
新しい パスワードを再入力してください:
    ※ユーザーsuzukiの新しいパスワードを再入力
[suzuki@vbox ~]$
```

パスワード変更後、再度ログインすると、今度はログインできます。パスワードは 30 日間有効になります。

```
[suzuki@vbox ~]$ chage -l suzuki
最終パスワード変更日                    : 7月 21, 2025
パスワード期限:                          : 8月 20, 2025
パスワード無効化中                       : なし
アカウント期限切れ                       : なし
パスワードが変更できるまでの最短日数      : 0
パスワードを変更しなくてよい最長日数      : 30
パスワード期限が切れる前に警告される日数  : 7
```

1.3.7 ユーザーの削除

ユーザーを削除します。ユーザーの削除後はログインできなくなります。

以下の例では、ユーザー testuser を追加し、削除しています。

```
$ sudo useradd testuser
$ id testuser
```

```
uid=1002(testuser) gid=1002(testuser) groups=1002(testuser)
$ sudo userdel testuser
$ id testuser
id: `testuser': no such user
```

userdel コマンドをオプション無しで実行すると、ユーザーのホームディレクトリや受信したメールを格納するメールスプールは削除されません。ユーザー削除と同時にホームディレクトリなども削除したい場合には、userdel コマンドに-r オプションをつけて実行する必要があります。

```
$ ls -l /home
合計 4
drwx------. 15 linuc  linuc  4096  7月 20 22:01 linuc
drwx------.  3 suzuki suzuki   99  7月 20 21:35 suzuki
drwx------.  3   1002   1002   78  7月 21 12:03 testuser
$ ls -l /var/spool/mail
合計 0
-rw-rw----. 1 linuc  mail 0  7月 19 11:44 linuc
-rw-rw----. 1 suzuki mail 0  7月 20 20:46 suzuki
-rw-rw----. 1   1002 mail 0  7月 21 12:03 testuser
```

このように、所有ユーザーが削除されたディレクトリやファイルは、パーミッションを確認すると所有ユーザーが元のユーザー ID で表示されるようになります。

再度ユーザー testuser を作成します。

```
$ sudo useradd testuser
useradd: warning: the home directory /home/testuser already exists.
useradd: Not copying any file from skel directory into it.
メールボックスファイルを作成します: ファイルが存在します
$ ls -l /home
合計 4
drwx------. 15 linuc    linuc    4096  7月 20 22:01 linuc
drwx------.  3 suzuki   suzuki     99  7月 20 21:35 suzuki
drwx------.  3 testuser testuser   78  7月 21 12:03 testuser
$ ls -l /var/spool/mail
合計 0
-rw-rw----. 1 linuc    mail 0  7月 19 11:44 linuc
-rw-rw----. 1 suzuki   mail 0  7月 20 20:46 suzuki
-rw-rw----. 1 testuser mail 0  7月 21 12:03 testuser
```

同じユーザー ID（上記の例では 1002）でユーザーが追加されたので、ホームディレクトリとメールスプールはユーザー testuser が再度所有ユーザーになっています。もし、削除後に追加された別のユーザーに同じユーザー ID（1002）が割り当てられると、ファイルやディレクトリの所有権が別のユーザーに移ってしまうので注意が必要です。

ホームディレクトリとメールスプールを同時に削除するには、userdel -r コマンドでユーザーを削除します。

```
$ sudo userdel -r testuser
$ ls -l /home
合計 4
drwx------. 15 linuc  linuc  4096  7月 20 22:01 linuc
drwx------.  3 suzuki suzuki   99  7月 20 21:35 suzuki
$ ls -l /var/spool/mail
合計 0
-rw-rw----. 1 linuc  mail 0  7月 19 11:44 linuc
-rw-rw----. 1 suzuki mail 0  7月 20 20:46 suzuki
```

1.3.8 グループの削除

グループを削除するには、groupdel コマンドを使用します。ユーザーが所属している主グループは削除できませんが、サブグループは警告無しに削除されます。グループを削除する前に/etc/group を参照して、そのグループ

に所属しているユーザーがいないか確認しておきます。

以下の例ではユーザー testuser（主グループ testuser）を作成し、グループ testgroup にサブグループとして所属させています。主グループは削除できませんが、サブグループは削除できます。

```
$ sudo useradd testuser
$ sudo groupadd testgroup
$ sudo gpasswd -a testuser testgroup
ユーザ testuser をグループ testgroup に追加
$ id testuser
uid=1002(testuser) gid=1002(testuser) groups=1002(testuser),5001(testgroup)
$ sudo groupdel testuser
groupdel: ユーザ 'testuser' のプライマリグループは削除できません。
$ sudo groupdel testgroup
$ id testuser
uid=1002(testuser) gid=1002(testuser) groups=1002(testuser)
```

1.4 SSH によるリモートログイン

SSH (Secure Shell) とは、リモート (遠隔) のサーバーにログインしてサーバーを操作するためのプロトコルです。SSH は、外部へ通信の内容が漏れないように通信が暗号化されています。また、パスワード認証よりもセキュリティレベルの高い公開鍵認証ができます。

Linux では、OpenSSH のサーバーおよびクライアントが利用できます。また、Linux サーバーに対して Windows クライアントから SSH でリモートログインすることもできます。

まずローカルホスト上で公開鍵認証の設定方法を実習し、ホスト OS の Windows から仮想マシン上で動作するゲスト OS の Linux に SSH でログインする方法を解説します。

1.4.1 SSH サービスの状態確認と開始

AlmaLinux では OpenSSH サーバーはデフォルトでインストールされて自動的に起動しています。sshd デーモンが起動していることを確認しておきます。SSH プロトコルはポート番号 22 番を使用しています。

```
$ sudo lsof -i:22
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd     890   root    3u  IPv4  22943      0t0  TCP *:ssh (LISTEN)
sshd     890   root    4u  IPv6  22945      0t0  TCP *:ssh (LISTEN)
```

1.4.2 SSH のログイン認証方法

SSH のログイン認証方法には、以下の方法があります。

パスワード認証による接続　サーバーに登録済みのユーザー名と、ユーザーのログインパスワードを使ってログイン認証を行います。簡単で分かりやすい認証方式ですが、ユーザー名とパスワードが分かれば誰でもログインできてしまうので、インターネットに接続するサーバーなどでは使用しません。デフォルトで有効になっているので、SSH サーバーの設定を変更して無効にしておきます。

公開鍵認証による接続　事前に作成した公開鍵をログインしたいサーバーに登録しておきます。公開鍵に対応した秘密鍵を持っているユーザーだけがログインできます。パスワード認証に比べて事前の設定が必要になりますが、ログインするためには秘密鍵が必要になるので、パスワード認証より安全な認証の仕組みです。

1.4.3 パスワード認証による接続

パスワード認証を使って、SSH サーバーに接続してログイン認証を行います。

サーバーに事前にログイン用のユーザー sshuser を作成します。

```
$ sudo useradd sshuser
$ sudo passwd sshuser
ユーザー sshuser のパスワードを変更。
新しいパスワード: ※ユーザーsshuserの新しいパスワードを入力
新しいパスワードを再入力してください:
    ※ユーザーsshuserの新しいパスワードを再入力
passwd: すべての認証トークンが正しく更新できました。
```

ローカルホストに SSH で接続します。接続するには ssh コマンドを使用します。ユーザー名を省略すると、ssh コマンドを実行したユーザーのユーザー名が指定されたことになります。

ssh コマンドの書式は以下の通りです。

```
$ ssh [ユーザー名@]接続先
```

接続先には IP アドレス、又は名前解決できるホスト名を指定します。今回はローカルホスト（localhost）にローカルループバックで接続してみます。ローカルループバックは、IPv4 のアドレスでは 127.0.0.1、IPv6 では::1 を宛先アドレスとしたホスト自身に接続する方法です。

```
$ ssh sshuser@localhost
```

SSH サーバーに接続すると、サーバーから「SSH サーバー証明書」が送られてきます。初回の接続時には以下のように尋ねられるので、yes と入力し、作成したユーザー sshuser のパスワードを入力します。ログインすると、コマンドプロンプトの表示が変わって sshuser でログインしたことが分かります。

```
[linuc@vbox ~]$ ssh sshuser@localhost
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:mVq+9GENHHauxEaOHKosLF8VcGW3rqny2PNy1Yio/zk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
    ※yesと入力
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
sshuser@localhost's password:
    ※サーバーに作成したユーザーsshuserのパスワードを入力
[sshuser@vbox ~]$ id
uid=1003(sshuser) gid=1003(sshuser) groups=1003(sshuser)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

ログアウトするには、exit コマンドを使用します。

```
[sshuser@vbox ~]$ exit
ログアウト
Connection to localhost closed.
[linuc@vbox ~]$
```

1.4.4 ssh コマンドの冗長モードによるトラブルシューティング

もし、ログインがうまくいかない場合は ssh コマンドに-v オプション (冗長モード) を付けてデバッグ用のメッセージを表示させ、詳細を確認します。

```
[linuc@vbox ~]$ ssh -v sshuser@localhost
OpenSSH_8.7p1, OpenSSL 3.2.2 4 Jun 2024
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Reading configuration data /etc/ssh/ssh_config.d/50-redhat.conf
debug1: Reading configuration data /etc/crypto-policies/back-ends/openssh.config
debug1: configuration requests final Match pass
debug1: re-parsing configuration
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Reading configuration data /etc/ssh/ssh_config.d/50-redhat.conf
```

```
debug1: Reading configuration data /etc/crypto-policies/back-ends/openssh.config
debug1: Connecting to localhost [::1] port 22.
debug1: Connection established.
（略）
Last login: Mon Jul 21 14:50:00 2025 from ::1
[sshuser@vbox ~]$ exit
[linuc@vbox ~]$
```

SSH プロトコルのやり取りが表示されるので、問題になっているやり取りを探すことができます。

1.4.5 SSH サーバー証明書による「なりすまし」の防止

一度接続したことのあるサーバーの SSH サーバー証明書は、クライアントのホームディレクトリに作られた.ssh ディレクトリの中に作成された known_hosts ファイルに保存されます。2 回目以降の接続時には、初回に尋ねられた表示は出ず、すぐに認証のためのパスワード入力が要求されます。

```
[linuc@vbox ~]$ ssh sshuser@localhost
sshuser@server's password:
Last login: Mon Sep  1 07:21:42 2025 from ::1
[sshuser@vbox ~]$ exit
[sshuser@vbox ~]$
```

cat コマンドでクライアントにある~/.ssh/known_hosts ファイルの中身を確認してみましょう。

```
[linuc@vbox ~]$ cat .ssh/known_hosts
localhost ssh-ed25519
    AAAAC3NzaC1lZDI1NTE5AAAAIF67wXfBQsFqTo7nM1aPX0yQh4DbQwvYXyBmZPepW6b9
localhost ssh-rsa
    AAAAB3NzaC1yc2EAAAADAQABAAABgQDPPbQ64LradMJ8nv+A5zIe5VKbawLPNylJznt61j3BUNz6
（略）
localhost ecdsa-sha2-nistp256
    AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBpjYw71x2i6
QU0WQa+MWhs3k7G/lJZlDsHA55siz79uQJh7d5Ya5GH8X5AmO1POOX4e3N8/t/3XTyLHwy/RA1A=
```

SSH サーバー証明書は、SSH サーバーに接続した際にサーバーからクライアントに対して送られてきます。初回接続時は~/.ssh/known_hosts に保存されている SSH サーバー証明書が無いので、接続してもよいか確認されます。yes と答えるとサーバー証明書は~/.ssh/known_hosts に保存されます。

2 回目以降の接続では、送られてきた SSH サーバー証明書と known_hosts に保存してある SSH サーバー証明書を比較して、同一であれば同じサーバーであることが分かります。もし異なる SSH サーバー証明書が送られてきた場合には、別のサーバーが「なりすまし」をしている可能性があるので、ssh コマンドは警告を表示して接続を中断します。

また、仮にコピーした SSH サーバー証明書を送ってきてサーバーなりすましをしようとしても、その後の接続手順の中で確認作業を行っているので、やはり接続は中断され、サーバーなりすましは失敗します。SSH サーバーには、SSH サーバー証明書（公開鍵）とサーバー秘密鍵の、ペアになった 2 つの鍵が必要だからです。公開鍵と秘密鍵については後述します。

サーバーの再インストールなどを行うと、サーバーの SSH サーバー証明書は再作成され、変更されてしまいます。その場合には、クライアントの~/.ssh/known_hosts ファイルに登録されている SSH サーバー証明書を削除して下さい。~/.ssh/known_hosts ファイルは単なるテキストファイルなので、vi エディタなどで該当する SSH サーバー証明書を削除するか、後述する ssh-keygen コマンドで削除します。

以下の例では、既存の SSH サーバー証明書や対となる秘密鍵を削除して、SSH サーバーを再起動してサーバー証明書を再作成します。再度 SSH サーバーに接続することで、接続が拒否されることや、クライアント側に保存されたサーバー証明書を削除することで再度接続できるようになることを確認します。

サーバー証明書は、SSH サーバーの動作しているシステムの/etc/ssh ディレクトリ内にある ssh_host で始まるファイルです。

```
$ ls /etc/ssh/ssh_host*
/etc/ssh/ssh_host_ecdsa_key      /etc/ssh/ssh_host_ed25519_key
    /etc/ssh/ssh_host_rsa_key
```

```
/etc/ssh/ssh_host_ecdsa_key.pub  /etc/ssh/ssh_host_ed25519_key.pub
    /etc/ssh/ssh_host_rsa_key.pub
```

ファイル名が_key で終わるファイルが秘密鍵、_key.pub で終わるファイルがサーバー証明書です。これらを削除してみます。

```
[linuc@vbox ~]$ sudo rm /etc/ssh/ssh_host*
[linuc@vbox ~]$ ls /etc/ssh/ssh_host*
ls: '/etc/ssh/ssh_host*' にアクセスできません:
    そのようなファイルやディレクトリはありません
```

SSH サーバーを再起動すると、秘密鍵とサーバー証明書が再作成されます。

```
[linuc@vbox ~]$ sudo systemctl restart sshd
[linuc@vbox ~]$ ls /etc/ssh/ssh_host*
/etc/ssh/ssh_host_ecdsa_key      /etc/ssh/ssh_host_ed25519_key
    /etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_ecdsa_key.pub  /etc/ssh/ssh_host_ed25519_key.pub
    /etc/ssh/ssh_host_rsa_key.pub
```

再度、ローカルホストに SSH でログインしてみます。

```
[linuc@vbox ~]$ ssh sshuser@localhost
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:9t/Wd3Arnr427qUAcRCvwMYOEVguhsz4T4cfvLACx5w.
Please contact your system administrator.
Add correct host key in /home/linuc/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/linuc/.ssh/known_hosts:3
Host key for localhost has changed and you have requested strict checking.
Host key verification failed.
```

known_hosts に保存されたサーバー証明書と、再作成されて送られて来たサーバー証明書が異なるため、警告が表示されて接続できなくなりました。

known_hosts に保存してあるサーバー証明書を削除します。ssh-keygen コマンドに-R オプションをつけて実行すると、サーバー証明書を削除できます。

```
[linuc@vbox ~]$ ssh-keygen -R localhost
# Host localhost found: line 1
# Host localhost found: line 2
# Host localhost found: line 3
/home/linuc/.ssh/known_hosts updated.
Original contents retained as /home/linuc/.ssh/known_hosts.old
```

再度接続してみます。新しいサーバー証明書が送られてくるので、保存するために yes と入力すると正常に接続できます。

```
[linuc@vbox ~]$ ssh sshuser@localhost
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:9t/Wd3Arnr427qUAcRCvwMYOEVguhsz4T4cfvLACx5w.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
    ※yesと入力
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
```

```
sshuser@localhost's password:
Last login: Mon Jul 21 13:57:28 2025 from ::1
[sshuser@vbox ~]$
```

実際のシステム管理では、OS を再インストールすることでサーバー証明書が再作成されることがあります。事前にバックアップを取っておき、再インストール後に書き戻すか、クライアント側で以前のサーバー証明書を削除するか、いずれかの方法を採るようにしてください。

1.4.6 公開鍵認証による接続

パスワード認証では「ユーザー名」と「パスワード」で認証しますが、もしこのパスワードが漏れてしまうと非常に危険です。また、セキュリティ攻撃用のプログラムを使って手当たり次第にパスワードを試す「総当たり攻撃」の可能性もあります。そこで、より安全な認証方法として公開鍵認証による接続が利用できます。インターネット上に公開するサーバーの場合には、パスワード認証を禁止し、この公開鍵認証で接続します。

公開鍵認証は、SSH 接続用の公開鍵と秘密鍵のキーペアを生成し、接続先のサーバーに公開鍵を登録して認証します。これを「公開鍵認証」といいます。

公開鍵認証の大まかな手順としては、以下のようになります。

1. 公開鍵と秘密鍵のキーペアを生成する
2. クライアントに公開鍵と秘密鍵を設置する
3. サーバーに公開鍵を設置する

1.4.7 SSH 公開鍵・秘密鍵の作成

SSH 公開鍵認証に使用する公開鍵と秘密鍵を作成します。Linux では ssh-keygen コマンドを使用します。

公開鍵と秘密鍵の設置場所は、デフォルトでは ssh-keygen コマンドを実行したユーザーのホームディレクトリにある.ssh ディレクトリに作成されます。

ssh-keygen コマンドを実行すると、鍵の設置場所とパスフレーズの入力が求められます。パスフレーズは、秘密鍵を有効にするための合言葉のようなものです。万一秘密鍵を盗まれたとしても、パスフレーズが分からなければ鍵の所有ユーザーになりすまして SSH サーバーに接続することはできません。

ユーザー sshuser で公開鍵・秘密鍵を作成します。

```
[linuc@vbox ~]$ sudo su - sshuser
[sshuser@vbox ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sshuser/.ssh/id_rsa):
    ※Enterキーを押す
Created directory '/home/sshuser/.ssh'.
Enter passphrase (empty for no passphrase): ※秘密鍵のパスフレーズを入力
Enter same passphrase again: ※秘密鍵のパスフレーズを再入力
Your identification has been saved in /home/sshuser/.ssh/id_rsa
Your public key has been saved in /home/sshuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:43Ociwq8FL/su9I+G2awYgv9f4O/YzQz27tnwtuxES4 sshuser@vbox
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|                 |
|                 |
|    o   S   .    |
| . . = .=o o .   |
|. + =.=ooBE +    |
| o =.*+o*+++o+   |
|  . o=@X++=Bo    |
+----[SHA256]-----+
```

~/.ssh ディレクトリに作成された公開鍵（id_rsa.pub）と秘密鍵 (id_rsa) を確認します。.ssh ディレクトリは ssh コマンド実行時、または ssh-keygen コマンド実行時に自動的に作成されます。

```
[sshuser@vbox ~]$ ls -ld .ssh
drwx------. 2 sshuser sshuser 38  7月 21 14:20 .ssh
[sshuser@vbox ~]$ ls -l .ssh
合計 8
-rw-------. 1 sshuser sshuser 2655  7月 21 14:20 id_rsa
-rw-r--r--. 1 sshuser sshuser  566  7月 21 14:20 id_rsa.pub
```

鍵の中身はテキストファイルになっているので、cat コマンドで確認できます。

```
[sshuser@vbox ~]$ cat .ssh/id_rsa.pub
ssh-rsa
    AAAAB3NzaC1yc2EAAAADAQABAAABgQCiiI9Fsn0CRxaoO7Xq0q4gEsADAmJNQaWNPOj/licSYrgZ
（略）
sU13lsxJ8= sshuser@vbox
```

公開鍵は 1 行のテキストですが、秘密鍵は何行かに分かれたフォーマットになっています。

```
[sshuser@vbox ~]$ cat .ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABAFbcvD8X
iKLcV42cDca6ITAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQCiiI9Fsn0C
（略）
T8kWVid7AggT/yyOXtxlygEAmC+7g1+NKAkxoqGaVBUOqveGCo+Qn5vLTw5kRHD7dSg3Xo
D7B+VNMSitiB2+f8HwfZJoVBA6I=
-----END OPENSSH PRIVATE KEY-----
```

1.4.8 SSH クライアントの.ssh ディレクトリおよび公開鍵・秘密鍵のパーミッション

公開鍵認証に使用する公開鍵・秘密鍵、およびそれらを格納する.ssh ディレクトリはセキュリティを守るためパーミッションが厳密に決められています。

ssh-keygen コマンドを使用して鍵を作成した際にはパーミッションは適切に設定されていますが、別のマシンで作成した鍵をコピーしてくる場合には、パーミッションを自分で設定する必要があります。

また、所有ユーザーは ssh コマンドを実行するユーザーである必要があります。初期設定時などにユーザーの作成から公開鍵・秘密鍵の設置までを root ユーザーで行っていると、所有ユーザーが root になってしまうので注意が必要です。

設定するパーミッションは以下の通りです。

ディレクトリおよびファイル	パーミッション
~/.ssh ディレクトリ	rwx------(700)
id_rsa（秘密鍵）	rw-------(600)
id_rsa.pub（公開鍵）	rw-r--r--(644)

```
[sshuser@vbox ~]$ ls -ld .ssh
drwx------. 2 sshuser sshuser 38  7月 21 14:20 .ssh
[sshuser@vbox ~]$ ls -l .ssh
合計 8
-rw-------. 1 sshuser sshuser 2655  7月 21 14:20 id_rsa
-rw-r--r--. 1 sshuser sshuser  566  7月 21 14:20 id_rsa.pub
```

1.4.9 サーバーへの公開鍵の設置

次に、クライアントの SSH 公開鍵 (id_rsa.pub) をサーバーに設置します。以下の手順で設置を行います。

1. クライアントからサーバーに公開鍵をコピー
2. ~/.ssh ディレクトリを作成
3. ~/.ssh/authorized_keys ファイルを作成
4. 公開鍵の内容を~/.ssh/authorized_keys に追加
5. 公開鍵認証でログインできることを確認

これらの手順を自動的に行ってくれる ssh-copy-id コマンドもありますが、ファイル名やパーミッションなどの確認も兼ねて手動で実行してみます。

本来であれば、現在使用している仮想マシンとは別の仮想マシンをサーバーとして用意して行いますが、以下の例ではその作業を省略して同一のホストをサーバーと見立てて作業を行っています。別途サーバーを用意できる場合には、あらかじめサーバーにユーザー sshuser を作成し、localhost と記述している箇所をサーバーの IP アドレスに置き換えて実行してみてください。

サーバーに公開鍵（id_rsa.pub）をコピーします。ここでは SSH プロトコルを使ったファイルコピーを行う scp コマンドを使います。

scp コマンドの書式は以下の通りです。

```
scp コピー元ファイル ユーザー名@接続先:コピー先ファイル
```

以下のように scp コマンドを実行して~/.ssh/id_rsa.pub を、サーバーのユーザー sshuser のホームディレクトリにリモートコピーします。

```
[sshuser@vbox ~]$ scp ~/.ssh/id_rsa.pub sshuser@localhost:~
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:9t/Wd3Arnr427qUAcRCvwMYOEVguhsz4T4cfvLACx5w.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
sshuser@localhost's password:
id_rsa.pub

    100%  566     1.5MB/s   00:00
合計 4
-rw-r--r--. 1 sshuser sshuser 566  7月 21 14:27 id_rsa.pub
```

sshuser として SSH 接続するのが初めてだったので、サーバー証明書の保存が要求されました。ホームディレクトリ内に公開鍵がコピーされています。

~/.ssh ディレクトリを作成し、公開鍵の設置を行います。

ホームディレクトリに.ssh ディレクトリを作成し、chmod コマンドでパーミッションを 700 に変更します。

```
[sshuser@vbox ~]$ mkdir .ssh
mkdir: ディレクトリ `.ssh' を作成できません: ファイルが存在します
[sshuser@vbox ~]$ chmod 700 .ssh
[sshuser@vbox ~]$ ls -ld .ssh
drwx------. 2 sshuser sshuser 80  7月 21 14:27 .ssh
```

既に.ssh ディレクトリが作成されているのでエラーが表示されましたが、新規のサーバーの場合には存在しないのでエラーは発生しません。パーミッションの設定も忘れずに行います。

.ssh ディレクトリの中に authorized_keys ファイルを作成し、パーミッションを変更します。公開鍵はこのファイルの中に追加していきます。

```
[sshuser@vbox ~]$ touch .ssh/authorized_keys
[sshuser@vbox ~]$ chmod 600 .ssh/authorized_keys
[sshuser@vbox ~]$ ls -l .ssh/authorized_keys
-rw-------. 1 sshuser sshuser 0  7月 21 14:31 .ssh/authorized_keys
```

公開鍵を authorized_keys に追加します。cat コマンドで出力をリダイレクトします。出力で“»”を使うと既存ファイルの authorized_keys ファイルを上書きせずに追記する事ができます。authorized_keys ファイルを作成する作業では、cp コマンドや mv コマンドは使用しないでください。authorized_keys ファイルを上書きする危険性がある他、SELinux が有効になっている場合、正常に動作しないことがあります。

```
[sshuser@vbox ~]$ cat id_rsa.pub >> .ssh/authorized_keys
[sshuser@vbox ~]$ cat .ssh/authorized_keys
ssh-rsa
    AAAAB3NzaC1yc2EAAAADAQABAAABgQCiiI9Fsn0CRxaoO7Xq0q4gEsADAmJNQaWNPOj/licSYrgZ
（略）
sU13lsxJ8= sshuser@vbox
```

sshuser でログインします。公開鍵が正しく設置されていれば、秘密鍵のパスフレーズの入力が求められます。もしパスワード認証を求められるような場合には、ファイル名やパーミッションなど、設置の手順を再確認してみてください。

```
[sshuser@vbox ~]$ ssh sshuser@localhost
Enter passphrase for key '/home/sshuser/.ssh/id_rsa':
    ※秘密鍵のパスフレーズを入力
Last login: Mon Jul 21 14:32:36 2025 from ::1
```

ssh コマンドを実行するのは sshuser であり、linuc ユーザーではないことに注意してください。公開鍵認証を行うには ssh コマンドを実行したユーザーのホームディレクトリ以下の.ssh ディレクトリ内に秘密鍵 id_rsa が存在していることが必要です。何度もログインを繰り返していたりすると、秘密鍵が無い別のユーザーから SSH ログインしようとしているために接続できないミスがよく発生します。接続できない時は必ず~/.ssh ディレクトリに秘密鍵があることを確認してください。

```
[sshuser@vbox ~]$ ls .ssh/
authorized_keys  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
[sshuser@vbox ~]$ exit
ログアウト
Connection to localhost closed.
[sshuser@vbox ~]$ exit
ログアウト
[linuc@vbox ~]$ ls .ssh
known_hosts  known_hosts.old
[linuc@vbox ~]$ ssh sshuser@localhost
sshuser@localhost's password:
Last login: Mon Jul 21 14:33:24 2025 from ::1
```

ここまでの実習では、su コマンドで sshuser に変更し、さらに SSH でログインしています。2 回 exit コマンドを実行し、linuc ユーザーに戻ると、秘密鍵も公開鍵も持っていないことがわかります。そして、linuc ユーザーでは公開鍵認証は行えず、パスワード認証になります。

1.4.10 ssh-copy-id コマンドを使った公開鍵の設置

SSH 公開鍵を手動で登録する方法の他に、ssh-copy-id コマンドを使った公開鍵の登録方法があります。ssh-copy-id コマンド一つで、ホストの authorized_keys に公開鍵を自動的に登録できます。

ssh-copy-id コマンドの書式は以下の通りです。

```
$ ssh-copy-id ユーザー名@接続先
```

linuc ユーザーで公開鍵と秘密鍵を作成した後、ssh-copy-id コマンドを実行してサーバーに公開鍵を登録します。

```
[linuc@vbox ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/linuc/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/linuc/.ssh/id_rsa
```

```
Your public key has been saved in /home/linuc/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:M3PenKR+5kG9lyRh8Zprpkn99gBEGUNyUutksZRwpjI linuc@vbox
The key's randomart image is:
+---[RSA 3072]----+
|          ++%=   |
|           Xo=o  |
|        E . *o . |
|         o =..+  |
|        S . =+.. |
|         * *.o+..|
|          o.==o..|
|          .. B..+ |
|           .*. ..o|
+----[SHA256]-----+
[linuc@vbox ~]$ ls .ssh
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

ssh-copy-id コマンドを実行します。

```
[linuc@vbox ~]$ ssh-copy-id linuc@localhost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
    "/home/linuc/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
    out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
    prompted now it is to install the new keys
linuc@localhost's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'linuc@localhost'"
and check to make sure that only the key(s) you wanted were added.
```

公開鍵認証で SSH 接続できるか確認します。

```
[linuc@vbox ~]$ ssh linuc@localhost
Enter passphrase for key '/home/linuc/.ssh/id_rsa':
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Jul 20 22:12:53 2025
[linuc@vbox ~]$ exit
ログアウト
Connection to localhost closed.
[linuc@vbox ~]$
```

公開鍵認証で接続できるようになった後は、SSH サーバーの設定を変更してパスワード認証による接続を禁止します。設定方法は後述します。

1.4.11 scp コマンドを使ったファイル転送

scp コマンドを使うと、SSH プロトコルを使用してファイル転送ができます。クライアントで作成した公開鍵をサーバーにコピーするために既に使用しましたが、ディレクトリ内の複数のファイルを再帰的に転送することもできます。

クライアントのホームディレクトリに testdir ディレクトリを作成し、その中に複数のファイルを作ります。次に、scp コマンドに-r オプションを付与して実行し、ディレクトリの中身をすべて scptestdir ディレクトリという新しいディレクトリに再帰的に転送します。

```
[linuc@vbox ~]$ ls
```

```
test.txt  testdir  ダウンロード  テンプレート  デスクトップ  ドキュメント
    ビデオ  音楽  画像  公開
[linuc@vbox ~]$ touch testdir/file1
[linuc@vbox ~]$ touch testdir/file2
[linuc@vbox ~]$ ls testdir
file1  file2
[linuc@vbox ~]$ scp -r testdir linuc@localhost:~/scptestdir
Enter passphrase for key '/home/linuc/.ssh/id_rsa':
file1                                                 100%    0     0.0KB/s
    00:00
file2                                                 100%    0     0.0KB/s
    00:00
[linuc@vbox ~]$ ls
scptestdir  test.txt  testdir  ダウンロード  テンプレート  デスクトップ
    ドキュメント  ビデオ  音楽  画像  公開
[linuc@vbox ~]$ ls scptestdir/
file1  file2
```

scptestdir ディレクトリが作成され、2 つのファイルが転送されていることが確認できます。

1.4.12 sftp コマンドを使ったファイル転送

SFTP (SSH File Transfer Protocol) とは、SSH でファイルを送受信できるプロトコルです。動作は FTP に似ています。

SFTP の主なコマンドは以下の通りです。

コマンド	動作
pwd	カレントディレクトリの表示
ls	ファイルの表示
cd [パス]	カレントディレクトリの移動
put [-P] ローカルパス [リモートパス]	ファイルをリモートに転送。-P オプションは所有権やパーミッションを維持
get [-P] リモートパス [ローカルパス]	ファイルをローカルに転送。-P オプションは所有権やパーミッションを維持
rm パス	ファイルを削除
mkdir パス	ディレクトリを作成
rmdir パス	ディレクトリを削除
lpwd	ローカルのカレントディレクトリの表示
lls [ls コマンドのオプション] [パス]	ローカルのファイルの表示
lcd パス	ローカルのカレントディレクトリの移動
lmkdir パス	ローカルのディレクトリを作成

接続先でのリモート操作と、接続元でのローカル操作がある点に注意してください。

あらかじめ転送用のファイルを作成した後、sftp コマンドでクライアントからサーバーにログインします。ログインできると、”sftp>”というプロンプトが表示されます。

```
[linuc@vbox ~]$ touch sftptestfile
[linuc@vbox ~]$ ls
scptestdir  sftptestfile  test.txt  testdir  ダウンロード  テンプレート
    デスクトップ  ドキュメント  ビデオ  音楽  画像  公開
[linuc@vbox ~]$ sftp linuc@localhost
Enter passphrase for key '/home/linuc/.ssh/id_rsa': ※秘密鍵のパスフレーズを入力
Connected to localhost.
sftp>
```

put コマンドでサーバーにファイルを転送できます。

```
sftp> ls
scptestdir            sftptestfile          test.txt              testdir
                  ダウンロード    テンプレート
デスクトップ    ドキュメント    ビデオ             公開                画像
                  音楽
sftp> cd testdir
sftp> ls
file1          file2
sftp> lls
scptestdir  sftptestfile  test.txt  testdir  ダウンロード  テンプレート
    デスクトップ  ドキュメント  ビデオ  音楽  画像  公開
sftp> put sftptestfile
Uploading sftptestfile to /home/linuc/testdir/sftptestfile
sftptestfile                                          100%    0     0.0KB/s
    00:00
sftp> ls
file1          file2          sftptestfile
sftp> exit
```

1.5 Windows からの SSH 接続

Windows でも ssh コマンドが標準で使用できるため、追加のソフトウェア無しで SSH 接続が行えます。ホスト OS の Windows から仮想マシン上のゲスト OS である Linux に公開鍵認証でリモートログインできるようにするまでの手順を解説します。

1.5.1 仮想マシン上の Linux の IP アドレスを確認

Linux の IP アドレスを ip a コマンドで確認します。

```
[linuc@vbox ~]$ ip a
 （略）
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:40:b7:96 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic
        noprefixroute enp0s8
       valid_lft 124875sec preferred_lft 124875sec
 （略）
```

Linux の IP アドレスは 192.168.56.101 です。

1.5.2　コマンドプロンプトの起動

Windows の検索ウインドウに「cmd」と入力し、コマンドプロンプトを起動します。

図 7: コマンドプロンプトの起動

1.5.3　ssh コマンドでリモートログインできることを確認

パスワード認証で Linux にリモートログインできることを確認します。

```
>ssh linuc@192.168.56.101
```

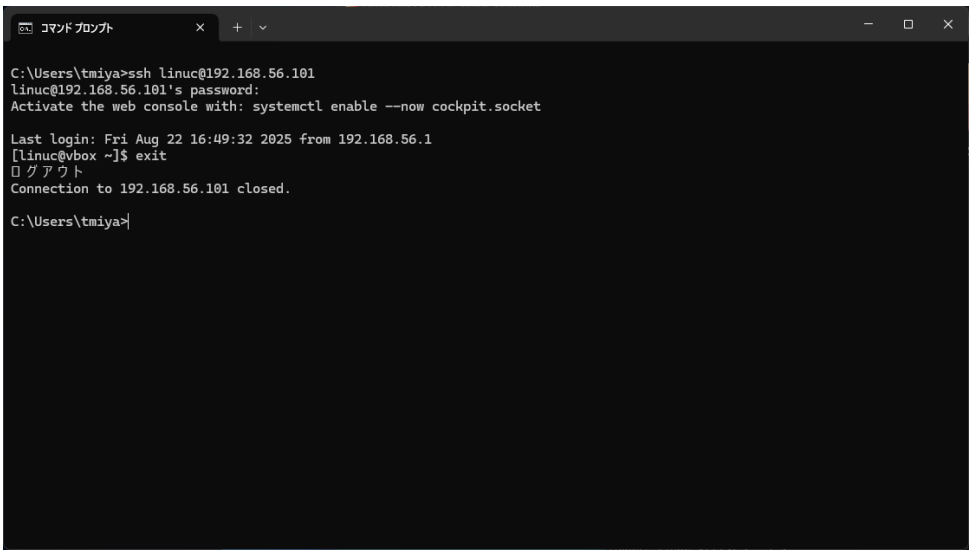

図 8: パスワード認証で Linux にリモートログイン

1.5.4　公開鍵・秘密鍵を作成

ssh-keygen コマンドで公開鍵・秘密鍵を作成します。

```
>ssh-keygen
```

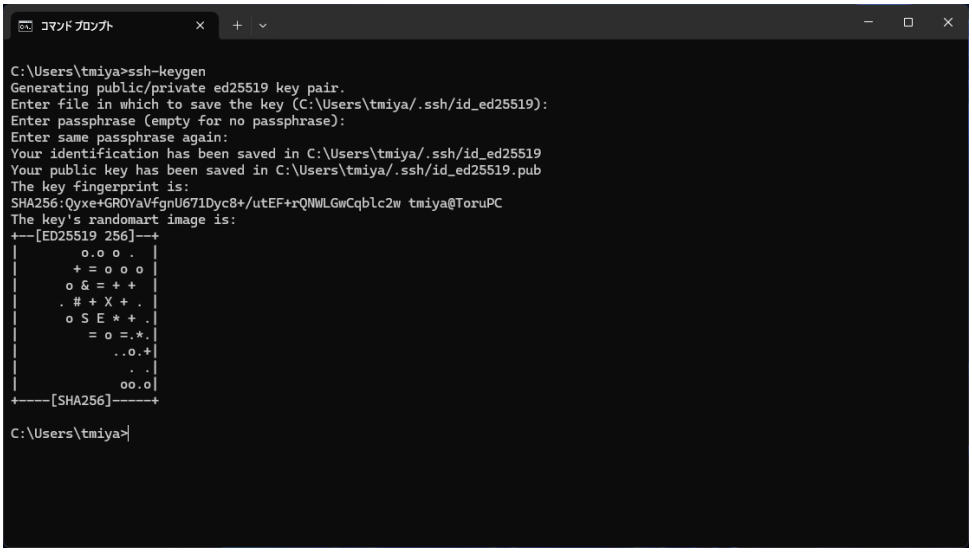

図 9: ssh-keygen コマンドで公開鍵・秘密鍵を作成

1.5.5　scp で公開鍵を Linux にコピー

scp で公開鍵を Linux にコピーします。

```
>cd .ssh
.ssh>dir
.ssh>scp id_ed25519.pub linuc@192.168.56.101:~/
```

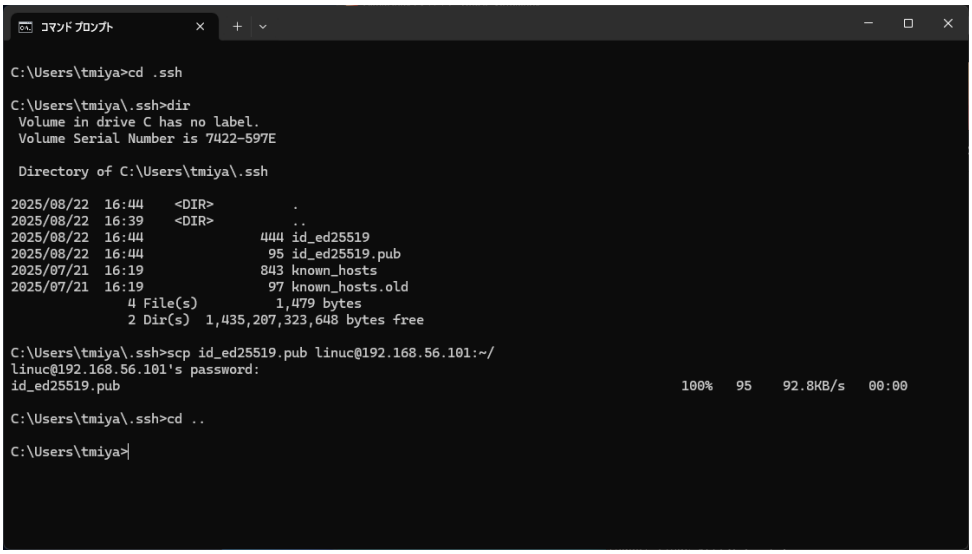

図 10: scp で公開鍵を Linux にコピー

1.5.6　Linux に公開鍵を設置

再度 Linux にリモートログインして公開鍵を設置します。

```
>ssh linuc@192.168.56.101
```

以下はログイン後の Linux での作業です。

```
[linuc@vbox ~]$ ls -l id_ed25519.pub
[linuc@vbox ~]$ cat id_ed25519.pub >> .ssh/authorized_keys
[linuc@vbox ~]$ exit
```

cat コマンドの出力を authorized_keys に追加リダイレクトすることで、追加で公開鍵が書き込めます。

1.5.7　公開鍵認証によるリモートログインの確認

再度 Linux にリモートログインして、公開鍵認証が行われることを確認します。

```
>ssh linuc@192.168.56.101
```

パスフレーズを入力してリモートログインできたら、公開鍵認証の設定は完了です。

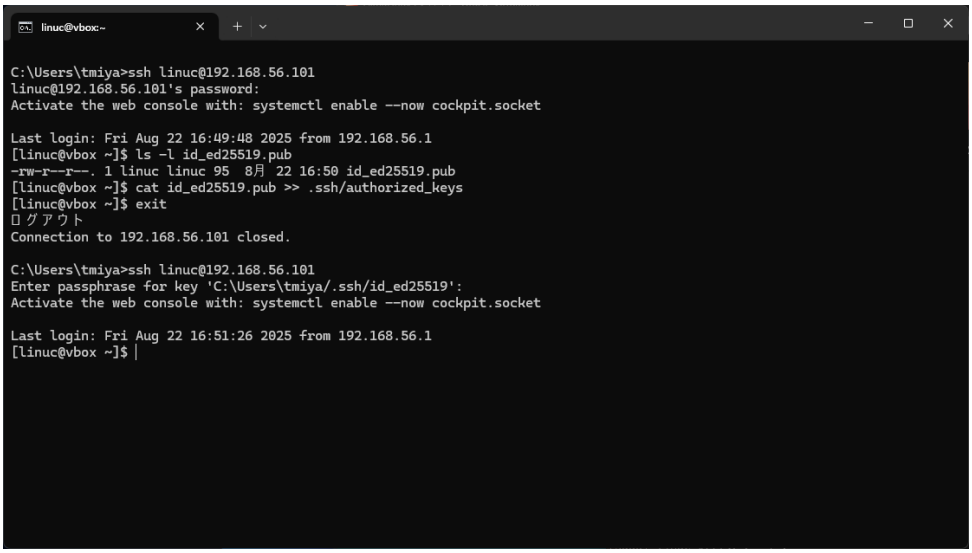

図 11: 公開鍵認証で Linux にリモートログイン

1.6　パスワード認証の禁止と管理者ユーザー root のログインの禁止

公開鍵認証による接続ができるようになったら、SSH サーバーの設定を変更してパスワード認証による接続を禁止しておきます。

SSH サーバーの設定ファイル/etc/ssh/sshd_config を以下のように設定変更します。

```
$ sudo vi /etc/ssh/sshd_config
```

```
#PasswordAuthentication yes
PasswordAuthentication no ※noに設定
```

また、管理者ユーザー root の外部からの直接ログインを禁止することもできます。root ユーザーの直接ログインを許すかどうかは後述します。root ユーザーの SSH ログインを禁止するには、以下のように変更します。

```
#PermitRootLogin prohibit-password
PermitRootLogin no ※noに設定
```

設定を保存したら、systemctl コマンドで sshd を再起動します。

```
$ sudo systemctl restart sshd
```

これで、外部からのパスワード認証を使ったログインが禁止され、かつ root ユーザーでのリモートログインが禁止されました。

1.7 root 権限の管理

管理者ユーザーである root は最も高い権限を持っているアカウントとなるため、管理方法には注意が必要です。root 権限を取得するには、以下の 3 つの方法があります。

- root で直接ログインする
- 一般ユーザーでログインした後、su コマンドを実行して管理者ユーザー root に切り替える
- 一般ユーザーでログインした後、sudo コマンドを使って root 権限でコマンドを実行する

どの方法も一長一短がありますが、root で直接ログインするのは許さず、一般ユーザーでログインした後に su コマンドか sudo コマンドを使用させることが多いようです。

1.7.1 su コマンドを実行できるユーザーを制限する

su コマンドは root パスワードを知っているユーザーなら誰でも root になれる事が問題になる場合があります。PAM（Pluggable Authentication Modules）の設定を変更して、su コマンドを実行できるユーザーを制限します。

wheel グループに所属しているユーザーのみ su コマンドを実行して root に切り替えられるように設定します。

PAM の設定ファイル/etc/pam.d/su を vi エディタで開いて、行頭のコメントアウトを 2 カ所外します。上の設定行は、wheel グループに所属しているユーザーはパスワード無しで su コマンドを実行できる、という設定です。下の設定行は、wheel グループに所属しているユーザーのみ su コマンドを実行できる、という設定です。

```
$ sudo vi /etc/pam.d/su

#%PAM-1.0
auth            required        pam_env.so
auth            sufficient      pam_rootok.so
# Uncomment the following line to implicitly trust users in the "wheel" group.
auth           sufficient      pam_wheel.so trust use_uid ※行頭の#を削除
# Uncomment the following line to require a user to be in the "wheel" group.
auth           required        pam_wheel.so use_uid ※行頭の#を削除
（略）
```

設定変更は修正した設定ファイルを保存するとすぐに反映されるので、システムの再起動などは必要ありません。

確認のため、一般ユーザー linuc で su コマンドを実行してみます。wheel グループに所属しているため、パスワード無しで root ユーザーに変更できます。

```
$ id
uid=1000(linuc) gid=1000(linuc) groups=1000(linuc),10(wheel)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
$ su -
# id
uid=0(root) gid=0(root) groups=0(root)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

1.7.2 sudo コマンドを実行できるユーザーを制限する

AlmaLinux では、デフォルトでは wheel グループに所属しているユーザーのみ sudo コマンドを実行でき、一般ユーザーは sudo コマンドを実行する権限が与えられていません。

sudo コマンドの設定は、/etc/sudoers に記述されていますが、パーミッションが 440 という書き込み権限の無い特殊な設定となっています。

```
$ ls -l /etc/sudoers
-r--r-----. 1 root root 4328  7月  1 07:46 /etc/sudoers
```

/etc/sudoers を編集するには、sudo コマンドを使って visudo コマンドを実行します。vim エディタで/etc/sudoers が編集可能な状態で開かれます。

```
$ sudo visudo
```

wheel グループに所属しているユーザーは sudo コマンドを実行できるように設定されている場所を確認します。下記の行がその設定です。

```
%wheel  ALL=(ALL)       ALL
```

「%wheel」は wheel グループを指定し、「ALL=(ALL) ALL」ですべてのコマンドを実行可能としています。

visudo コマンドは vi エディタを呼び出しただけですので、「:q」と入力して編集を終了します。

1.7.3 sudo で実行できるコマンドの制限

sudo コマンドでは、ある特定のグループに対して、一部のコマンドのみ実行できるように制限できます。

たとえば、グループ power に所属しているユーザーに対してシステムの再起動やシャットダウンができるような権限を付与します。

```
$ sudo visudo
```

実行可能なコマンドをカンマ区切りで記述していきます。追加する場所はどこでも構いませんが、ファイルの最後尾に追加しておくと後から追加したと分かってよいでしょう。NOPASSWD はパスワード無しで実行できる権限です。

```
%power        ALL=NOPASSWD:  /usr/sbin/reboot, /usr/sbin/poweroff
```

power グループを作成し、suzuki ユーザーのサブグループに追加します。

```
$ sudo groupadd power
$ sudo usermod -G power suzuki
```

su -コマンドで作成したユーザー suzuki に切り替えます。

```
$ sudo su - suzuki
$ id
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki),5001(power)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

sudo コマンドを使ってシステムを再起動します。

```
[suzuki@vbox ~]$ sudo reboot
```

システムがシャットダウンされ、再起動するのを確認します。

2 ネットワークの管理

2.1 ネットワークインターフェイスの設定

ネットワークインターフェースには、IP アドレスなどネットワーク通信のための各種設定が必要となります。ネットワーク環境に合わせて設定を変更します。

2.1.1 ip コマンドを使ったネットワークインターフェースの設定

ip コマンドは、ネットワークインターフェースの状態の確認や設定、ルーティングテーブルの表示や追加、削除、ARP テーブルの確認や削除など、ネットワークにおける操作全般を行うことができます。

2.1.2 ネットワーク設定の確認

IP アドレスや MAC アドレスの確認には ip address show コマンドを実行します。ip a コマンドに省略可能です。

```
$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:e6:fe:4c brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 73888sec preferred_lft 73888sec
    inet6 fd17:625c:f037:2:a00:27ff:fee6:fe4c/64 scope global dynamic
        noprefixroute
       valid_lft 86313sec preferred_lft 14313sec
    inet6 fe80::a00:27ff:fee6:fe4c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:40:b7:96 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe40:b796/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

enp0s3 が VirtualBox の NAT に設定されたアダプター、enp0s8 がホストオンリーに設定されたアダプターです。

2.1.3 ルーティングテーブル、デフォルトゲートウェイの確認

ルーティングテーブルの確認を行うには ip route show コマンドを実行します。ip r コマンドに省略可能です。これは従来の route コマンドに相当します。

```
$ ip route show
default via 10.0.2.2 dev enp0s3 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.56.0/24 dev enp0s8 proto kernel scope link src 192.168.56.101 metric 101
```

default になっている行がデフォルトゲートウェイの設定です。この例では、10.0.2.2 がネットワークインターフェース enp0s3 から通信する場合のデフォルトゲートウェイとして設定されています。192.168.56.0/24 は、ホストオンリーで外部とは通信できないネットワークなのでゲートウェイが設定されていません。

2.1.4 ARP テーブルの確認

ARP テーブルの確認を行うには ip neighbor show コマンドを実行します。neighbor は neigh に省略できます。

```
$  ip neigh show
10.0.2.3 dev enp0s3 lladdr 52:55:0a:00:02:03 STALE
10.0.2.2 dev enp0s3 lladdr 52:55:0a:00:02:02 REACHABLE
fe80::2 dev enp0s3 lladdr 52:56:00:00:00:02 router STALE
```

2.2 ss コマンドを使った設定確認

ss コマンドは Linux の通信ソケットの状態を確認するコマンドです。確認する対象をネットワークインターフェースに指定することで、ネットワークの各種状態を確認できます。よく使用するオプションは以下の通りです。

オプション	説明
-t	TCP の情報を表示
-u	UDP の情報を表示
-l	LISTEN しているポートの情報を表示
-a	すべての接続を表示
-n	コンピューター名の名前解決をせずに IP アドレスで表示
-i	ネットワークインターフェースの統計情報を表示
-4	IPv4 だけを表示
-6	IPv6 だけを表示

2.2.1 TCP 通信の状態の表示

すべての TCP 通信の状態を表示したい場合には、ss -tan コマンドを実行します。TCP、すべて、名前解決をしない、を指定するオプションです。

```
$ ss -tan
State      Recv-Q     Send-Q         Local Address:Port        Peer Address:Port
LISTEN     0          128                  0.0.0.0:22               0.0.0.0:*
LISTEN     0          4096             127.0.0.1:631                0.0.0.0:*
LISTEN     0          25                   0.0.0.0:514              0.0.0.0:*
ESTAB      0          0            192.168.56.101:22         192.168.56.1:50396
LISTEN     0          128                     [::]:22                  [::]:*
LISTEN     0          4096                   [::1]:631                 [::]:*
LISTEN     0          25                      [::]:514                 [::]:*
```

いくつかの待ち受けポートの他、SSH で接続しているのが分かります。

2.2.2 待ち受け TCP ポートの表示

待ち受けているすべての TCP のポートを表示したい場合には、ss -tl コマンドを実行します。TCP、待ち受けです。ポート番号は/etc/services を参照して書き換えられています。

```
$ ss -tl
State      Recv-Q     Send-Q         Local Address:Port        Peer Address:Port
LISTEN     0          128                  0.0.0.0:ssh              0.0.0.0:*
LISTEN     0          4096             127.0.0.1:ipp                0.0.0.0:*
LISTEN     0          25                   0.0.0.0:shell            0.0.0.0:*
LISTEN     0          128                     [::]:ssh                 [::]:*
LISTEN     0          4096                   [::1]:ipp                 [::]:*
LISTEN     0          25                      [::]:shell               [::]:*
```

2.2.3 待ち受け UDP ポートの表示

待ち受けているすべての UDP のポートを表示したい場合には、ss -ul コマンドを実行します。

```
$ ss -ul
State      Recv-Q     Send-Q         Local Address:Port        Peer Address:Port
UNCONN     0          0                    0.0.0.0:43085            0.0.0.0:*
UNCONN     0          0                    0.0.0.0:mdns             0.0.0.0:*
UNCONN     0          0                    127.0.0.1:323            0.0.0.0:*
UNCONN     0          0                    0.0.0.0:syslog           0.0.0.0:*
UNCONN     0          0                       [::]:mdns                [::]:*
UNCONN     0          0                     [::1]:323                  [::]:*
UNCONN     0          0                       [::]:syslog              [::]:*
UNCONN     0          0                       [::]:57956               [::]:*
```

2.3 ping コマンドを使用した疎通の確認

リモートのホストに IP のパケットが到達できるか確認するためには ping コマンドを実行します。Ctrl+C を実行するまで無制限に実行されます。

```
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 バイト応答 送信元 8.8.8.8: icmp_seq=1 ttl=255 時間=8.47ミリ秒
64 バイト応答 送信元 8.8.8.8: icmp_seq=2 ttl=255 時間=8.29ミリ秒
64 バイト応答 送信元 8.8.8.8: icmp_seq=3 ttl=255 時間=8.95ミリ秒
^C
--- 8.8.8.8 ping 統計 ---
送信パケット数 3, 受信パケット数 3, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 8.294/8.572/8.954/0.279 ms
```

-c オプションで実行回数を指定できます。以下の例では、5 回だけ疎通確認を行っています。

```
$ ping -c 5 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 バイト応答 送信元 8.8.8.8: icmp_seq=1 ttl=255 時間=8.80ミリ秒
64 バイト応答 送信元 8.8.8.8: icmp_seq=2 ttl=255 時間=8.71ミリ秒
64 バイト応答 送信元 8.8.8.8: icmp_seq=3 ttl=255 時間=8.35ミリ秒
64 バイト応答 送信元 8.8.8.8: icmp_seq=4 ttl=255 時間=8.47ミリ秒
64 バイト応答 送信元 8.8.8.8: icmp_seq=5 ttl=255 時間=8.61ミリ秒

--- 8.8.8.8 ping 統計 ---
送信パケット数 5, 受信パケット数 5, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 8.351/8.589/8.798/0.161 ms
```

ping コマンドは ICMP プロトコルを使いますので、経路の途中にあるルーターやファイアーウォールなどで ICMP プロトコルがブロックされていると、ping コマンドを実行してもうまく結果が返ってこない場合があります。また、対象となるリモートのホストが ping コマンドに反応を返さない場合もあります。

レスポンス結果 (RTT) の目安としては、同じネットワークセグメント上のホストの場合は 1ms(ミリ秒) 以内、国内のインターネット上の他のホストの場合、10ms〜30ms、地球の裏側で 500ms 程度かかります。

2.4 ethtool コマンドを使ったネットワークインターフェース情報の確認

ethtool コマンドは、ネットワークインターフェースに対するハードウェアスペックやファームウェア、リンク状態、リンクスピードなどの確認、およびアクセラレーション機能の有効化・無効化を制御することができます。

ネットワークインターフェースに対するハードウェアスペックやファームウェア、リンク状態、リンクスピードなどの確認をするには ethtool コマンドを実行します。

```
$ ethtool enp0s3
Settings for enp0s3:
```

```
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                            100baseT/Half 100baseT/Full
                            1000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: Yes
    Supported FEC modes: Not reported
    Advertised link modes:  10baseT/Half 10baseT/Full
                            100baseT/Half 100baseT/Full
                            1000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Advertised FEC modes: Not reported
    Speed: 1000Mb/s
    Duplex: Full
    Auto-negotiation: on
    Port: Twisted Pair
    PHYAD: 0
    Transceiver: internal
    MDI-X: Unknown
netlink error: Operation not permitted
        Current message level: 0x00000007 (7)
                               drv probe link
    Link detected: yes
```

ドライバーなどの情報を確認するには ethtool -i コマンドを実行します。

```
$ ethtool -i enp0s3
driver: e1000
version: 5.14.0-570.26.1.el9_6.x86_64
firmware-version:
expansion-rom-version:
bus-info: 0000:00:03.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

2.5 各種ネットワーク設定ファイル

Linux のネットワーク関連の設定は、いくつかの設定ファイルに分散して設定されています。IP アドレスは Network Manager を使って管理するため設定ファイルを直接編集することなどはありませんが、その他のネットワーク設定ファイルは参照だけでなく編集することもあります。

2.5.1 静的な名前解決/etc/hosts

設定ファイル/etc/hosts には、静的な名前解決のために IP アドレスとホスト名が対になって列挙された情報が記述されています。

```
$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

2.5.2 参照 DNS 設定ファイル/etc/resolv.conf

設定ファイル/etc/resolv.conf には、DNS を使って名前解決を行う場合に参照する DNS サーバの情報が記述されています。先に記述されてある DNS サーバを優先 DNS サーバとして最初に参照します。その優先 DNS サーバが応答しない場合には、次に記述されている代替 DNS サーバを参照します。

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
search localdomain
nameserver 10.0.2.3
nameserver fd17:625c:f037:2::3
```

/etc/resolv.conf は直接編集しない /etc/resolv.conf ファイルを直接編集して、参照する DNS の設定を変更することは推奨されていません。なぜなら、/etc/resolv.conf の設定情報は NetworkManager よって自動的に変更されます。そのため、/etc/resolv.conf を手動で変更後にシステムや NetworkManager を再起動すると、/etc/resolv.conf の設定が古い設定で上書きされてしまい、名前解決ができなくなるなど予期せぬトラブルが発生してしまいます。

DNS サーバの設定は、NetworkManager で変更します。ネットワークインターフェースが有効になる際に、記述しておいた値に基づいて/etc/resolv.conf ファイルが設定されます。

2.5.3 名前解決設定ファイル/etc/nsswitch.conf

設定ファイル/etc/nsswitch.conf には、名前解決などを行う場合に参照する仕組みのリストと優先順位が記述されています。/etc/hosts を参照するのか、DNS や NIS を参照させるのかなどが細かく設定できます。

```
$ cat /etc/nsswitch.conf
 （略）
hosts:      files dns myhostname
 （略）
```

ホストの名前解決は、左から順に「files」、「dns」で優先度が記述されています。まず設定ファイル/etc/hosts を参照し、次に DNS を参照して名前解決を行います。

この設定ファイルには他に、ユーザ認証を行う際の参照先の指定なども記述されます。

2.5.4 ポート番号とサービスの対応リスト/etc/services

設定ファイル/etc/services には、各種 TCP/UDP のポート番号と対応するサービスの名前が記述されています。

たとえば、HTTP プロトコルは以下のように記述されています。

```
$ cat /etc/services
 （略）
http            80/tcp          www www-http    # WorldWideWeb HTTP
http            80/udp          www www-http    # HyperText Transfer Protocol
http            80/sctp                         # HyperText Transfer Protocol
 （略）
```

HTTP は基本的に TCP を使っていますが、UDP や SCTP（Stream Control Transmission Protocol）も定義されています。

各種コマンドがポート番号を表示する際、TCP のポート番号 80 番を表示する時にはプロトコル名に置き換えて http と表示します。ss コマンドの-n オプションは表示を数値で表示し、-n オプションが指定されないとプロトコル名などを名前で表示します。

この設定ファイルは、あくまで各種コマンドがポート番号をプロトコル名に置き換えて表示するために参照されます。実際には他のプロトコルがポートを使用している場合もあります。

```
$ ss -tln | grep 22
LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
LISTEN 0      128             [::]:22           [::]:*
$ ss -tl | grep ssh
LISTEN 0      128          0.0.0.0:ssh        0.0.0.0:*
LISTEN 0      128             [::]:ssh          [::]:*
```

ポート番号 22 が ssh に置き換えられているのが分かります。ポート番号が数字のまま表示される場合、/etc/services に記述が無いためです。また、2 つ目の例は表示が IPv6 での表示になっていますが、これは IPv6 が有効な場合の SSH サーバーの動作によるものです。IPv4 でも IPv6 でも、どちらでも接続可能になっています。

2.5.5 プロトコル定義ファイル/etc/protocols

設定ファイル/etc/protocols には各種プロトコルの名前とプロトコル番号が記述されています。たとえば、よく使用しているプロトコル番号は以下の通りです。

```
ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
tcp     6       TCP             # transmission control protocol
udp     17      UDP             # user datagram protocol
```

2.6 firewalld によるパケットフィルタリング

firewalld は Linux カーネルに実装されたパケットフィルタリングを使用する仕組みです。パケットフィルタリングとは、ネットワークに対してどのようなパケットの通過を許可および拒否するか判定する機能のことです。ファイアーウォールの基本的な機能であるため、ファイアーウォール機能と説明される場合もあります。firewalld は、カーネル内の netfilter 機能のフロントエンドとして実装されており、ユーザーランドのパケットフィルタリングのルール定義を行うコマンドとして firewall-cmd コマンドが用意されています。

2.6.1 firewalld の NAT 機能

firewalld にはパケットフィルタリング機能の他に、NAT(Network Address Translation) というパケットの送信元または宛先の IP アドレスを変換する機能があります。NAT には以下の種類があります。ここでは内部ネットワークをプライベート IP アドレスが割り振られた LAN、外部ネットワークをグローバル IP アドレスが割り振られたインターネットを想定して説明します。

スタティック NAT 内部ホストの IP アドレスと外部向け IP アドレスを 1 対 1 で結びつけます。内部から外部へアクセスするパケットの送信元 IP アドレスを、内部ホストの IP アドレスから結びつけられている外部向け IP アドレスに書き換えて通信を行います。外部と通信できるホストは用意された外部向け IP アドレスの数とあらかじめ決まっています。外部から内部への通信を許可して、外部から内部のホストへアクセスさせることもできます。

ダイナミック NAT 複数の内部ホストの IP アドレスと複数の外部向け IP アドレスを N 対 N で結びつけます。内部から外部へアクセスするパケットの送信元 IP アドレスを、内部ホストの IP アドレスから外部向け IP アドレスのプールから選択された IP アドレスに書き換えて通信を行います。IP アドレスの対応付けは通信開始時に行われるので、外部向け IP アドレスが余っていないと通信が行えませんが、他のホストが通信を終了して外部向け IP アドレスが解放されると通信が行えるようになります。内部のホストの数が多い場合には、外部向け IP アドレスが不足することになるので、次の NAPT を使用した方が外部との通信が行いやすいでしょう。

NAPT(IP マスカレード) 複数の内部ホストの IP アドレスと 1 つの外部向け IP アドレスを結びつけます。内部向け IP アドレスは外部に出る際に外部向け IP アドレスに書き換えられて通信を行います。その際に NAPT ではポート番号の変換も行います。ポート番号は最大 65535 番まであるので、1 つの外部 IP アドレスで沢山の内部ホストを外部と通信させることができます。

2.6.2 firewalld のステータス確認

firewall-cmd コマンドで firewalld のステータスを確認します。

```
$ sudo firewall-cmd --list-all
public (default, active)
  target: default
  ingress-priority: 0
  egress-priority: 0
  icmp-block-inversion: no
  interfaces: ens160
  sources:
```

```
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

services で、通信を許可するサービスを確認できます。cockpit はポート番号の確認で websm（9090 番）と表示されていたものです。そして DHCP、SSH が許可されています。

2.6.3 パケットの通過を一時的に許可する

パケットの通過を許可は firewall-cmd - -add-service コマンドで設定します。コマンドの書式は以下の通りです。

```
firewall-cmd --add-service=サービス名
```

たとえば、Web サーバーへの HTTP 通信を許可するには、以下のように実行します。

```
$ sudo firewall-cmd --add-service=http
success
$ sudo firewall-cmd --list-service
cockpit dhcpv6-client http ssh
```

2.6.4 パケットの通過を永続的に許可する

上記の方法では、システムを再起動すると許可設定は消えてしまいます。永続的に許可を設定をしたい場合には、- -permanent オプションをつけて設定を変更し、- -reload オプションをつけて実行して設定を適用します。以下の例は、上記の HTTP 通信の許可を行っていない状態を想定したものです。

```
$ sudo firewall-cmd --add-service=http --permanent
success
$ sudo firewall-cmd --list-service
cockpit dhcpv6-client ssh
```

この段階では適用されていませんので- -reload をつけて実行します。

```
$ sudo firewall-cmd --reload
success
$ sudo firewall-cmd --list-service
cockpit dhcpv6-client http ssh
```

設定が反映されました。

2.6.5 設定できるサービスを確認する

どのようなサービスが値として指定できるのか、一覧を確認するには- -get-services オプションを使います。

```
$ sudo firewall-cmd --get-services
0-AD RH-Satellite-6 RH-Satellite-6-capsule afp alvr amanda-client
    amanda-k5-client amqp amqps anno-1602 anno-1800 apcupsd aseqnet audit
    ausweisapp2 bacula
（略）
wsmans xdmcp xmpp-bosh xmpp-client xmpp-local xmpp-server zabbix-agent
    zabbix-java-gateway zabbix-server zabbix-trapper zabbix-web-service zero-k
    zerotier
```

非常に多くの設定が可能です。必要な設定は許可したいソフトウェア側のマニュアルなども参照してください。

2.6.6 許可サービスの取り消し

許可されているサービスを取り消しすることもできます。

```
$ sudo firewall-cmd --remove-service=cockpit
success
```

この設定も一時的なもので、システムの再起動時に許可したくない場合には、--permanent オプションをつけて設定を変更し、--reload オプションをつけて実行して設定を適用します。

```
$ sudo firewall-cmd --remove-service=cockpit --permanent
success
$ sudo firewall-cmd --reload
success
```

3　サービスの管理

3.1　OS が起動するまでのプロセス

マシンに電源を入れた後、以下のような順番でシステムの初期化が行われ、OS が起動します。

1. 電源オン
2. UEFI/BIOS 起動とハードウェアの初期化
3. ブートローダー（GRUB）の起動
4. Linux カーネルイメージの読み込み
5. systemd プロセスの起動
6. 各種サービスの起動
7. OS 起動完了

3.2　ブートローダー GRUB の起動

マシンの電源をオンにすると、UEFI/BIOS が起動してハードウェアの初期化が行われ、起動に使用するブートデバイス（ハードディスクなど）が決定します。ブートデバイスからブートローダーである GRUB が読み込まれ、起動処理が引き継がれます。GRUB は、Linux カーネルのイメージをメモリ上にロードする役割を持っています。

GRUB は自動的にデフォルトのカーネルをロードしますが、起動時に ESC キーを押していると GRUB の起動メニューを呼び出すことができます。

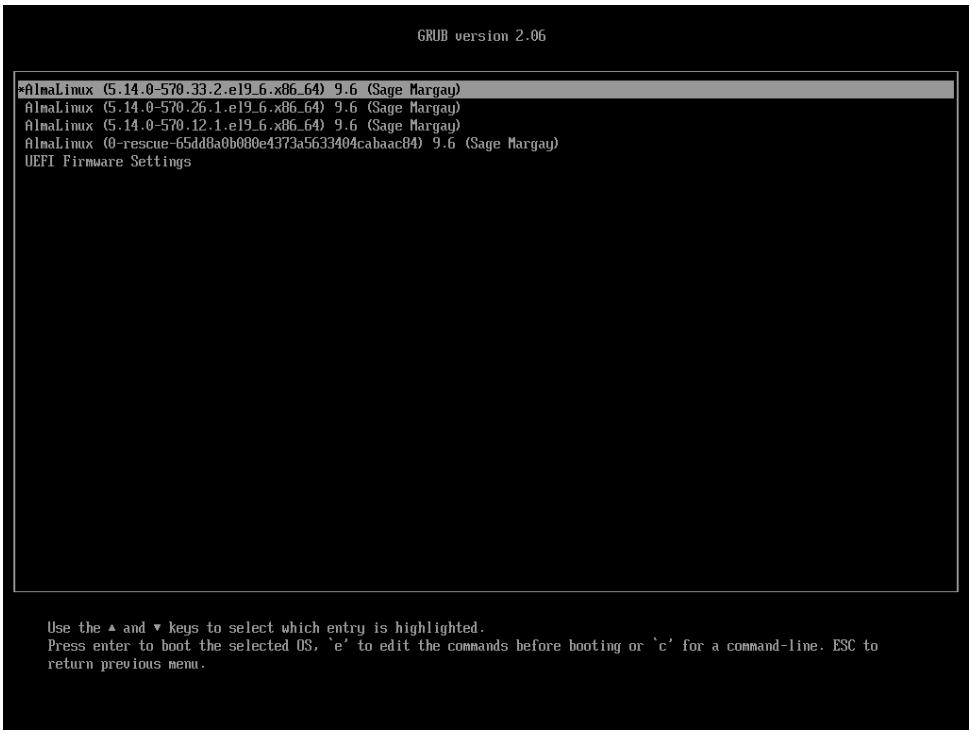

図 12: GRUB メニュー画面

Linux カーネルイメージが複数ある場合は、GRUB のメニュー画面でロードしたいイメージを選択して、Enter キーを押します。

3.2.1　GRUB の設定確認

GRUB の設定を確認したい場合には、grubby コマンドに- -info=ALL オプションをつけて実行します。

```
$ sudo grubby --info=ALL
index=0
kernel="/boot/vmlinuz-5.14.0-570.26.1.el9_6.x86_64"
args="ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
    resume=/dev/mapper/almalinux_vbox-swap rd.lvm.lv=almalinux_vbox/root
    rd.lvm.lv=almalinux_vbox/swap rhgb quiet $tuned_params"
```

```
root="/dev/mapper/almalinux_vbox-root"
initrd="/boot/initramfs-5.14.0-570.26.1.el9_6.x86_64.img $tuned_initrd"
title="AlmaLinux (5.14.0-570.26.1.el9_6.x86_64) 9.6 (Sage Margay)"
id="65dd8a0b080e4373a5633404cabaac84-5.14.0-570.26.1.el9_6.x86_64"
index=1
kernel="/boot/vmlinuz-5.14.0-570.12.1.el9_6.x86_64"
args="ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
    resume=/dev/mapper/almalinux_vbox-swap rd.lvm.lv=almalinux_vbox/root
    rd.lvm.lv=almalinux_vbox/swap rhgb quiet $tuned_params"
root="/dev/mapper/almalinux_vbox-root"
initrd="/boot/initramfs-5.14.0-570.12.1.el9_6.x86_64.img $tuned_initrd"
title="AlmaLinux (5.14.0-570.12.1.el9_6.x86_64) 9.6 (Sage Margay)"
id="65dd8a0b080e4373a5633404cabaac84-5.14.0-570.12.1.el9_6.x86_64"
index=2
kernel="/boot/vmlinuz-0-rescue-65dd8a0b080e4373a5633404cabaac84"
args="ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
    resume=/dev/mapper/almalinux_vbox-swap rd.lvm.lv=almalinux_vbox/root
    rd.lvm.lv=almalinux_vbox/swap rhgb quiet"
root="/dev/mapper/almalinux_vbox-root"
initrd="/boot/initramfs-0-rescue-65dd8a0b080e4373a5633404cabaac84.img"
title="AlmaLinux (0-rescue-65dd8a0b080e4373a5633404cabaac84) 9.6 (Sage Margay)"
id="65dd8a0b080e4373a5633404cabaac84-0-rescue"
```

3 つの設定が存在しているのがわかります。各設定項目の意味は以下の通りです。

項目	意味
index	ブートメニューの選択肢のインデックス番号です。
kernel	起動に使用するカーネルです。
args	カーネル起動時に引き渡される値です。
root	起動時に参照されるルートデバイスです。
initrd	起動時に使用される起動 RAM ディスクです。
title	ブートメニューの選択肢に表示される文字列です。
id	マシンのユニーク ID とカーネルバージョンを組み合わせた値です。

3.2.2 GRUB のデフォルト起動の確認

GRUB がデフォルトで起動するカーネルの確認は、grubby コマンドに- -default-kernel オプション、または- -default-index オプションをつけて実行すると確認できます。

```
$ sudo grubby --default-kernel
/boot/vmlinuz-5.14.0-570.26.1.el9_6.x86_64
$ sudo grubby --default-index
0
```

3.2.3 GRUB のデフォルト起動カーネルの変更

GRUB のデフォルト起動カーネルを変更したい場合には、grubby コマンドに- -set-default オプションでカーネル、または index の番号を指定することで変更できます。

```
$ sudo grubby --set-default /boot/vmlinuz-5.14.0-570.12.1.el9_6.x86_64
The default is
    /boot/loader/entries/65dd8a0b080e4373a5633404cabaac84-5.14.0-570.12.1.el9_6
.x86_64.conf with index 1 and kernel /boot/vmlinuz-5.14.0-570.12.1.el9_6.x86_64
$ sudo grubby --set-default 0
The default is
    /boot/loader/entries/65dd8a0b080e4373a5633404cabaac84-5.14.0-570.26.1.el9_6
.x86_64.conf with index 0 and kernel /boot/vmlinuz-5.14.0-570.26.1.el9_6.x86_64
```

3.3 カーネルの起動

GRUB で指定された Linux カーネルイメージがメモリに読み込まれて、カーネルが起動します。カーネルはハードウェアを初期化し、カーネルの各種機能を有効にしていきます。

カーネルは必要に応じてモジュールを読み込みますが、モジュールは初期化 RAM ディスク（initramfs）に含まれています。カーネルは初期化 RAM ディスクをメモリに読み込み、仮のルートファイルシステムとして利用可能にすることで、必要となるモジュールのファイルが読み込めるようになります。

3.3.1 dmesg によるカーネル起動時の動作の確認

カーネルが起動する際の動作の様子は、dmesg コマンドで確認できます。

```
# dmesg
[    0.000000] Linux version 5.14.0-570.26.1.el9_6.x86_64
    (mockbuild@x64-builder03.almalinux.org) (gcc (GCC) 11.5.0 20240719 (Red Hat
    11.5.0-5), GNU ld version 2.35.2-63.el9) #1 SMP PREEMPT_DYNAMIC Wed Jul 16
    09:12:04 EDT 2025
[    0.000000] The list of certified hardware and cloud instances for Red Hat
    Enterprise Linux 9 can be viewed at the Red Hat Ecosystem Catalog,
    https://catalog.redhat.com.
[    0.000000] Command line:
    BOOT_IMAGE=(hd0,gpt2)/vmlinuz-5.14.0-570.26.1.el9_6.x86_64
    root=/dev/mapper/almalinux_vbox-root ro
    crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
    resume=/dev/mapper/almalinux_vbox-swap rd.lvm.lv=almalinux_vbox/root
    rd.lvm.lv=almalinux_vbox/swap rhgb quiet
[    0.000000] [Firmware Bug]: TSC doesn't count with P0 frequency!
[    0.000000] BIOS-provided physical RAM map:
（略）
```

特定のデバイスが動作しないなどのトラブルは、カーネル起動時のメッセージで発生している問題を特定できることがあります。デバイス名等で検索してみるとよいでしょう。

3.4 systemd について

systemd は、カーネル起動後、最初に起動されるプロセスです。Linux のシステムを構成する各種サービスの起動などを行う役目を担っています。

3.4.1 ユニットでの管理

systemd では「ユニット」という単位でシステムを管理します。ユニットには、「ターゲット」（ランレベルに相当）ユニットや「サービス」ユニットがあり、それぞれのユニットは依存関係の定義ができるようになっています。

依存関係とは、たとえば「このサービスを実行するにはあらかじめこのサービスが実行されていなければならない」という関係です。systemd では依存関係にないサービスを並列処理で実行するため、高速にシステムを起動できるという利点があります。

主なユニットの種類は以下の通りです。

ユニット	役割
service	従来のサービスと同様
target	サービスを取りまとめるためのユニット
mount	マウントポイント
swap	スワップ領域
device	デバイス

mount ユニット、swap ユニット、device ユニットは直接操作することはありません。主に service ユニットを操作することになります。target ユニットは、システムを GUI で起動するか CUI で起動するかなどで操作する程度にしか使用しません。

3.4.2 サービスの操作

systemd では、サービスの起動や停止を行うのに systemctl コマンドを使用します。

firewalld の起動や停止、再起動、そして状態の確認を行うには、以下の systemctl コマンドを使用します。

サービスの起動と停止、再起動 systemctl start コマンドで、サービスを起動します。systemctl stop コマンドでサービスを停止します。systemctl restart コマンドで、サービスを再起動します。

```
$ sudo systemctl start firewalld
$ sudo systemctl stop firewalld
$ sudo systemctl restart firewalld
```

サービスのステータス確認 systemctl status コマンドで、サービスのステータスを確認できます。

```
$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
         preset: enabled)
     Active: active (running) since Sat 2025-07-26 15:13:03 JST; 2s ago
       Docs: man:firewalld(1)
   Main PID: 3400 (firewalld)
      Tasks: 2 (limit: 10816)
     Memory: 23.1M
        CPU: 367ms
     CGroup: /system.slice/firewalld.service
             └─3400 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

 7月 26 15:13:03 vbox systemd[1]: Starting firewalld - dynamic firewall
     daemon...
 7月 26 15:13:03 vbox systemd[1]: Started firewalld - dynamic firewall daemon.
```

3.4.3 ユニット一覧の取得

systemd で管理されているユニットの一覧を取得するには、systemctl list-unit-files コマンドを実行します。

```
$ systemctl list-unit-files
UNIT FILE                                  STATE           PRESET
proc-sys-fs-binfmt_misc.automount          static          -
-.mount                                    generated       -
boot-efi.mount                             generated       -
boot.mount                                 generated       -
dev-hugepages.mount                        static          -
dev-mqueue.mount                           static          -
proc-sys-fs-binfmt_misc.mount              disabled        disabled
（略）
systemd-sysupdate-reboot.timer             disabled        disabled
systemd-sysupdate.timer                    disabled        disabled
systemd-tmpfiles-clean.timer               static          -

427 unit files listed.
```

すべての種類のユニットが表示されてしまうので、ユニットの種類を絞り込むには-t オプションを付与して実行します。

たとえば、service ユニットだけを表示するには以下の systemctl コマンドを実行します。

```
$ systemctl list-unit-files -t service
UNIT FILE                                  STATE           PRESET
accounts-daemon.service                    enabled         enabled
```

```
alsa-restore.service                          static          -
alsa-state.service                            static          -
（略）
```

表示されるステータス（STATE）の意味は以下の通りです。

ステータス	意味
enabled	システム起動時に実行される
disabled	システム起動時に実行されない
static	システム起動時の実行の有無は設定できない
generated	systemd が生成したユニット

3.4.4 現在のユニットの状況を確認

現在のユニットの状況を確認するには、systemctl list-units コマンドを実行します。systemctl コマンドのデフォルトはこのサブコマンドの指定になっています。

以下の例は同じ結果を返します。

```
$ systemctl list-units
$ systemctl
```

-t オプションを使って、service ユニットだけに絞り込むこともできます。

```
$ systemctl -t service
  UNIT                    LOAD   ACTIVE SUB     DESCRIPTION
  accounts-daemon.service loaded active running Accounts Service
  alsa-state.service      loaded active running Manage Sound Card State
      (restore and store)
  atd.service             loaded active running Deferred execution scheduler
  auditd.service          loaded active running Security Auditing Service
（略）
● mcelog.service          loaded failed failed  Machine Check Exception Logging
    Daemon
（略）

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.
58 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

表示の意味は以下の通りです。

項目	意味
UNIT	ユニット名
LOAD	systemd への設定の読み込み状況
ACTIVE	実行状態の概要。active か inactive で表される
SUB	実行状態の詳細。running（実行中）や exited（実行したが終了した）などで表される。
DESCRIPTION	ユニットの説明

デフォルトでは、項目 ACTIVE の実行状態が active になっているもののみが表示されています。inactive のユニットも表示するには- -all オプションをつけて実行します。

項目 LOAD は、systemctl mask コマンドで無効化されると masked に変わります。詳細は後述します。

項目 ACTIVE が failed になっていると、何らかの原因で起動失敗しているということになります。上記の例では、mcelog サービスの起動に失敗しています。

3.4.5 デバイス一覧の確認

-t device オプションを付与して、デバイス一覧を表示します。

```
$ systemctl list-units -t device
  UNIT

      LOAD   ACTIVE SUB     DESCRIPTION                                  >
  sys-devices-pci0000:00-0000:00:01.1-ata2-host2-target2:0:0-2:0:0:0-block-sr0
.device      loaded active plugged VBOX_CD-ROM
  sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device
                                          loaded active plugged 82540EM Gigabit
      Ethernet Controller (PRO/>
  sys-devices-pci0000:00-0000:00:05.0-sound-card0-controlC0.device
                             loaded active plugged
      /sys/devices/pci0000:00/0000:00:05.0/soun>
  sys-devices-pci0000:00-0000:00:08.0-net-enp0s8.device
                                          loaded active plugged 82540EM Gigabit
      Ethernet Controller (PRO/>
 （略）
```

3.4.6 マウント状況の確認

-t mount オプションを付与して、マウントの状況一覧を表示します。

```
$ systemctl list-units -t mount
  UNIT                    LOAD   ACTIVE SUB     DESCRIPTION
  -.mount                 loaded active mounted Root Mount
  boot-efi.mount          loaded active mounted /boot/efi
  boot.mount              loaded active mounted /boot
  dev-hugepages.mount     loaded active mounted Huge Pages File System
 （略）
```

3.4.7 スワップ状況の確認

-t swap オプションを付与して、スワップの状況一覧を表示します。

```
$ systemctl list-units -t swap
  UNIT                                    LOAD   ACTIVE SUB    DESCRIPTION
  dev-mapper-almalinux_vbox\x2dswap.swap loaded active active
      /dev/mapper/almalinux_vbox-swap
 （略）
```

3.4.8 サービスの自動起動の設定

システム起動時にサービスを自動起動するには、systemctl enable コマンドを実行します。自動起動しないようにするには systemctl disable コマンドを実行します。

例として、firewalld をシステム起動時に自動起動するように設定します。すでにデフォルトで自動起動する設定のため、一度 systemctl disable コマンドを実行した後、systemctl enable コマンドを実行します。

```
$ sudo systemctl disable firewalld
Removed "/etc/systemd/system/multi-user.target.wants/firewalld.service".
Removed "/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service".
$ sudo systemctl enable firewalld
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service
    → /usr/lib/systemd/system/firewalld.service.
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service
    → /usr/lib/systemd/system/firewalld.service.
```

/usr/lib/systemd/system/firewalld.service が firewalld の起動スクリプトです。systemctl enable コマンドを実行すると、/etc/systemd/system/multi-user.target.wants ディレクトリにシンボリックリンクが作成されます。

この動作は、multi-user.target ターゲットユニットが呼び出された時に、シンボリックリンクの起動スクリプトが実行されるように設定しています。

systemctl disable コマンドを実行すると、作成されたシンボリックリンクが削除され、起動スクリプトは呼び出されなくなります。

3.4.9 サービスの systemd からの除外

systemctl mask コマンドを実行すると、指定したサービスが systemd の管理から除外され、手動での起動も行えなくなります。

動作としては、/etc/systemd/system/firewalld.service が/dev/null へのシンボリックリンクとして作成され、この起動スクリプトが呼び出されても何も行われなくなります。

firewalld を systemd から除外します。

```
$ sudo systemctl mask firewalld
Created symlink /etc/systemd/system/firewalld.service → /dev/null.
$ sudo systemctl start firewalld
Failed to start firewalld.service: Unit firewalld.service is masked.
```

systemctl is-enabled コマンドで、サービスの状態が確認できます。firewalld サービスの状態は masked となっています。

```
$ sudo systemctl is-enabled firewalld
masked
```

systemctl unmask コマンドを実行すると、シンボリックリンクが削除されて、指定したサービスが systemd で管理されるようになります。firewalld サービスの状態は disabled になります。

```
$ sudo systemctl unmask firewalld
Removed "/etc/systemd/system/firewalld.service".
$ sudo systemctl is-enabled firewalld
enabled
```

3.4.10 systemd のサービスに関連するディレクトリとシステム起動の仕組み

systemd が内部的にどのような仕組みになっているのか、関連するディレクトリを解説します。

systemctl enable コマンドの動作を見ても分かる通り、systemd の仕組みにおいて、関連するディレクトリは以下の 2 つです。

ディレクトリ	役割
/usr/lib/systemd/system	サービス起動スクリプトを格納
/etc/systemd/system	サービス起動スクリプトに対するシンボリックリンクを配置

システム起動時の systemd の動作は、/etc/systemd/system ディレクトリ以下のサブディレクトリ内に作成されたサービス起動スクリプトへのシンボリックリンクが順次実行されてサービスが起動されます。シンボリックリンクの作成される場所は、役割別のターゲットユニット毎にディレクトリが分けられています。

/etc/systemd/system ディレクトリにあるターゲット毎のディレクトリの実行順とその役割は以下の通りです。

実行順	ディレクトリ	役割
1	sysinit.target.wants	システム初期に実行
2	basic.target.wants	システム共通に実行

実行順	ディレクトリ	役割
3	multi-user.target.wants	CUI 起動の状態
4	graphical.target.wants	GUI 起動の状態

systemd では multi-user.target を実行後に graphical.target が実行されるようになっています。

どこまで実行するかは、次に説明するデフォルトターゲットの設定によって決められています。

3.4.11 デフォルトターゲットの変更

systemd ではサービス起動スクリプトを順番に実行していき、デフォルトターゲットで指定されたターゲットまで実行します。デフォルトターゲットを変更することで、CUI 起動をするか、GUI 起動にするかを選択できます。

systemctl get-default コマンドで、現在のデフォルトターゲットを確認します。

```
$ systemctl get-default
graphical.target
```

デフォルトターゲットの変更は、systemctl set-default コマンドを実行します。デフォルトターゲットを multi-user.target に変更し、再起動します。CUI で起動してくることを確認します。

```
$ sudo systemctl set-default multi-user.target
Created symlink /etc/systemd/system/default.target →
    /usr/lib/systemd/system/multi-user.target.
$ sudo reboot
```

システムが再起動すると、CUI で起動し、ログインプロンプトが表示されます。

GUI での起動に戻すには、以下の systemctl set-default コマンドを実行します。

```
$ sudo systemctl set-default graphical.target
Created symlink /etc/systemd/system/default.target →
    /usr/lib/systemd/system/graphical.target.
$ sudo reboot
```

システムが再起動すると、GUI で起動し、ユーザー選択画面が表示されます。

3.4.12 現在のターゲットの一時的な変更

systemd での現在のターゲットを一時的に変更するには、systemctl isolate コマンドを実行します。

GUI から CUI に変更します。GUI ログインしている場合、ログアウトします。

```
$ sudo systemctl isolate multi-user.target
```

CUI から GUI に変更します。

```
$ sudo systemctl isolate graphical.target
```

3.5 systemd のタイマーによるジョブのスケジュール実行

システムのメンテナンスなどで定期的にプロセスを実行するには、systemd のタイマーを使います。

3.5.1 有効なタイマーの一覧

systemd の有効なタイマーの一覧を表示するには、systemctl list-timers コマンドを実行します。

```
$ systemctl list-timers
NEXT                         LEFT       LAST                          PASSED
    UNIT                         ACTIVATES
Sat 2025-07-26 16:11:27 JST 13min left -                             -
    systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.service
Sat 2025-07-26 16:48:06 JST 50min left -                             -
    dnf-makecache.timer          dnf-makecache.service
Sun 2025-07-27 00:00:00 JST 8h left    Sat 2025-07-26 14:58:27 JST 59min ago
    logrotate.timer              logrotate.service
Sun 2025-07-27 00:00:00 JST 8h left    Sat 2025-07-26 14:58:27 JST 59min ago
    mlocate-updatedb.timer       mlocate-updatedb.service

4 timers listed.
Pass --all to see loaded but inactive timers, too.
```

3.5.2 タイマーの詳細の確認

タイマーの詳細を確認をするには、systemctl status コマンドを使います。ログのローテーションを行う logrotate.timer の状態を確認してみます。

```
$ systemctl status logrotate.timer
● logrotate.timer - Daily rotation of log files
     Loaded: loaded (/usr/lib/systemd/system/logrotate.timer; enabled; preset:
         enabled)
     Active: active (waiting) since Sat 2025-07-26 15:56:31 JST; 1min 50s ago
      Until: Sat 2025-07-26 15:56:31 JST; 1min 50s ago
    Trigger: Sun 2025-07-27 00:00:00 JST; 8h left
   Triggers: ● logrotate.service
       Docs: man:logrotate(8)
             man:logrotate.conf(5)

 7月 26 15:56:31 localhost systemd[1]: Started Daily rotation of log files.
```

3.5.3 タイマーの設定ファイルの内容の確認

タイマーの設定ファイルの内容を確認するには、systemctl cat コマンドを使います。

```
$ systemctl cat logrotate.timer
# /usr/lib/systemd/system/logrotate.timer
[Unit]
Description=Daily rotation of log files
Documentation=man:logrotate(8) man:logrotate.conf(5)

[Timer]
OnCalendar=daily
AccuracySec=1h
Persistent=true

[Install]
WantedBy=timers.target
```

最初に、このタイマーで実行されるサービスの定義ファイルが/usr/lib/systemd/system/logrotate.timer であることが分かります。

[Timer] セクションがスケジュール実行を定義しています。OnCalendar が実行するタイミングを指定しています。daily は毎日 0 時を意味しています。このようなキーワードでの指定の他、時間や実行間隔などを指定することもできます。

AccuracySec は、タイマーの精度を指定しています。精度を高くすると、頻繁にタイマーを制御するため CPU 実行が必要となり消費電力が上がります。デフォルトは 1m（1 分）ですが、ここではあまり高い精度は必要ない設定を行っています。

Persistent は、このタイマーを実行するタイミングにシステムが停止していた時、再度システムが起動した際の挙動を定義しています。true に設定されていると、再度のシステム起動時にこのタイマーを実行します。

3.6 NTP による時刻合わせ

NTP（Network Time Protocol）はマシンなどの時刻を合わせるためのプロトコルです。Linux では、Linux が動作しているマシンの時刻を合わせる NTP クライアントとしての動作と、その他のマシンなどに時刻同期を提供する NTP サーバーとしての動作があります。クライアントの数が多い場合、外部の NTP サーバーに大きな負荷をかけないよう、ローカルの NTP サーバーを用意してこのサーバーのみ外部の NTP サーバーと時刻同期し、内部のクライアントはローカル NTP サーバーと時刻同期させるようにするとよいでしょう。

3.6.1 Chrony の動作確認

AlmaLinux では、NTP サーバー/NTP クライアントとして Chrony が使われており、chronyd というサービスとして扱われます。Chrony はデフォルトで起動しているので、動作している様子を確認します。

```
$ systemctl status chronyd
● chronyd.service - NTP client/server
     Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; preset:
         enabled)
     Active: active (running) since Sat 2025-07-26 16:20:42 JST; 8s ago
       Docs: man:chronyd(8)
             man:chrony.conf(5)
    Process: 750 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited,
        status=0/SUCCESS)
   Main PID: 785 (chronyd)
      Tasks: 1 (limit: 10816)
     Memory: 4.1M
        CPU: 47ms
     CGroup: /system.slice/chronyd.service
             └─785 /usr/sbin/chronyd -F 2

 7月 26 16:20:42 localhost chronyd[785]: chronyd version 4.6.1 starting
     (+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS
     +SECHASH +I>
 7月 26 16:20:42 localhost chronyd[785]: Loaded 0 symmetric keys
 7月 26 16:20:42 localhost chronyd[785]: Using right/UTC timezone to obtain
     leap second data
 7月 26 16:20:42 localhost chronyd[785]: Frequency -8.088 +/- 0.231 ppm read
     from /var/lib/chrony/drift
 7月 26 16:20:42 localhost chronyd[785]: Loaded seccomp filter (level 2)
 7月 26 16:20:42 localhost systemd[1]: Started NTP client/server.
 7月 26 16:20:48 vbox chronyd[785]: Selected source 139.162.81.45
     (2.almalinux.pool.ntp.org)
 7月 26 16:20:48 vbox chronyd[785]: System clock wrong by -1.974025 seconds
 7月 26 16:20:46 vbox chronyd[785]: System clock was stepped by -1.974025
     seconds
 7月 26 16:20:46 vbox chronyd[785]: System clock TAI offset set to 37 seconds
```

ログを見てみると、起動時の時間が約 2 秒ほどずれており、時間が 2 秒ほど巻き戻されているのがタイムスタンプでも分かります。

3.6.2　参照している NTP サーバーの確認

次に時刻同期のために参照している NTP サーバーを確認します。クライアントとしての動作は chronyc コマンドを使って操作します。

```
$ chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* old-tok2.jp.ntp.li            3   6    77    45  -1598us[-1688us] +/-   26ms
^+ time.cloudflare.com           3   6    77    46    +24us[  -67us] +/-   63ms
^+ time.cloudflare.com           3   6    77    44  +1973us[+1973us] +/-   61ms
^+ pred-374.chl.la               3   6    77    44   +873us[ +873us] +/-   24ms
```

2 桁目にある記号のうち、「*」となっているのが現在参照している NTP サーバーです。NTP サーバーは様々な組織がサービスを提供しており、複数のサーバーが参照可能（記号「+」）になっているのがわかります。

ステータスの読み方は以下の通りです。1 列目の M 列はソースのモードを示します。

記号	意味
^	サーバー
=	ピア
#	ローカル

2 列目の S 列はソースの状態を示します。

記号	意味
*	同期しているソース
+	同期候補のソース
-	同期候補から外れたソース
?	切断されたソース、パケットが全てのテストをパスしないソース
x	偽の時計と判断したもの
~	時刻の変動性が大きすぎるソース

3.6.3　参照する NTP サーバーの設定を確認、変更する

Chrony の設定は「/etc/chrony.conf」に記述されています。参照する NTP サーバーを NICT が提供している「ntp.nict.jp」に変更してみます。

```
$ sudo vi /etc/chrony.conf
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (https://www.pool.ntp.org/join.html).
# pool 2.almalinux.pool.ntp.org iburst
pool ntp.nict.jp iburst
```

デフォルトでは ntp.org が運営している NTP サーバープールからランダムに参照するようになっています。この設定をコメントアウトして無効にし、NICT の NTP サーバーを参照するように設定を追加します。

設定変更を適用する　変更を適用するには、chronyd サービスを再起動してみます。

```
$ sudo systemctl restart chronyd
```

chronyc コマンドで時刻同期の様子を確認します。

```
# chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? ntp-a3.nict.go.jp             1   6     3     2   -441us[ -441us] +/- 4866us
^? 2001:ce8:78::2                1   6     3     1  +2926ns[+2926ns] +/- 9113us
```

```
^? ntp-b3.nict.go.jp              1   6     3     2    -33us[  -33us] +/- 4379us
^? ntp-a2.nict.go.jp              1   6     3     1   -285us[ -285us] +/- 4682us
```

記号「?」なので、まだ同期していない状態です。

```
# chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- ntp-a3.nict.go.jp              1   6     7     1   +356us[  -68us] +/- 4295us
^- 2001:ce8:78::2                 1   6     7     0   -827us[-1251us] +/-   10ms
^+ ntp-b3.nict.go.jp              1   6     7     1   +281us[ -143us] +/- 4552us
^* ntp-a2.nict.go.jp              1   6     7     0   -105us[ -529us] +/- 4624us
```

ntp-a2 と同期し、ntp-b3 も時刻同期可能な状態です。

```
# chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ ntp-a3.nict.go.jp              1   6    17    53   +369us[ +534us] +/- 4276us
^+ 2001:ce8:78::2                 1   6    17    52  -1092us[ -927us] +/-   10ms
^+ ntp-b3.nict.go.jp              1   6    17    53   -238us[  -73us] +/- 4827us
^* ntp-a2.nict.go.jp              1   6    17    52   +268us[ +433us] +/- 4123us
```

4 つすべての NTP サーバーが時刻同期可能となりました。

Linux が正しい時刻で動作していることは、ログの記録やファイルなどのデータの保存にとって重要なので、正しい時刻になっているように NTP クライアントとしてきちんと動作していることを確認しておきましょう。

3.7 NTP サーバーとして時刻を提供する

NTP サーバーは、他のクライアントに対して自身の時刻との同期をサービスとして提供します。

3.7.1 NTP サーバー機能を有効にする

Chrony はデフォルトでは NTP サーバー機能が無効になっています。設定を変更して NTP サーバーを有効にします。有効にするには、接続を許可するクライアントの IP アドレスを指定します。

```
$ sudo vi /etc/chrony.conf
（中略）
# Allow NTP client access from local network.
#allow 192.168.0.0/16
allow 192.168.56.0/24
```

allow ディレクティブはデフォルトではコメントアウト状態ですが、接続を許可する IP アドレス、ここではネットワークアドレスで一括して許可するように設定しています。

設定を有効にするため、chronyd サービスを再起動します。

```
$ sudo systemctl restart chronyd
```

3.7.2 ファイアーウォールの設定を変更して接続を許可する

NTP はネットワークプロトコルのため、ファイアーウォールで接続の許可をする必要があります。

```
$ sudo firewall-cmd --add-service=ntp --permanent
success
$ sudo firewall-cmd --reload
success
$ sudo firewall-cmd --list-services
cockpit dhcpv6-client ntp ssh
```

3.7.3　クライアントで NTP サーバーにアクセスする

次にクライアントから NTP サーバーにアクセスしてみます。

以下は別途クライアント用として用意した Almalinux での操作です。NTP サーバーとして設定したサーバーの IP アドレスを指定します。

```
$ sudo vi /etc/chrony.conf
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (https://www.pool.ntp.org/join.html).
# pool 2.almalinux.pool.ntp.org iburst
server 192.168.56.101 iburst
```

NTP サーバーの参照は server と pool の 2 種類があります。server は単一のサーバーを指定する場合、pool は名前解決で複数の結果が返る場合、最大 4 つまでを参照先 NTP サーバーとする、という違いがあります。

設定を適用します。

```
$ sudo systemctl restart chronyd
```

動作を確認します。

```
$ chronyc sources
MMS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* 192.168.56.101                2   6    17     1  +1842ns[  +11us] +/- 4560us
```

時刻が同期しました。

NTP サーバーを 1 つだけ設定しましたが、NTP サーバーが停止するとクライアントは時刻同期ができなくなります。時刻設定がシビアな環境においては、NTP サーバーを複数用意する、用意した NTP サーバーが参照できない場合は一時的に外部 NTP サーバーを参照させるなどいくつかの方法が考えられます。システム環境に合った設定を検討、実施するようにしてください。

4 ファイルシステムの管理

4.1 アクセス権の管理

Linux は POSIX で示されているアクセス制御に準拠しています。POSIX とは「Portable Operating System Interface for UNIX」の略で、IEEE（Institute of Electrical and Electronics Engineers、アイ・トリプル・イー）によって定められた、UNIX ベースの OS の仕様セットです。ユーザー ID（uid）/グループ ID（gid）とパーミッションの組み合わせでファイルに対するアクセス権を管理しています。

4.1.1 UID と GID

ユーザー ID（uid : User Identifier）は Linux システムでユーザーを識別するためのユニークな番号です。Linux で追加されたユーザーカウントには、それぞれ個別に uid が割り振られます。uid は 0 から 65535 までの値をとります。0 は特別なユーザー ID で、管理者権限を持つ root ユーザーに付与されています。

グループ ID（gid: Group Identifier）はグループを識別するためのユニークな番号です。Linux のユーザーは、1 つ以上のグループに所属することができます。gid は 0 から 65535 までの値をとります。

4.1.2 検証用ユーザー、グループの確認

アクセス制御の動作確認のため、検証用のユーザーを用意します。すでにユーザー suzuki は作成しているので、ユーザー sato を追加します。

```
$ sudo useradd sato
$ id sato
uid=1004(sato) gid=1004(sato) groups=1004(sato)
$ id suzuki
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki),5001(power)
```

4.1.3 別々のユーザーとして作業する

ユーザー sato とユーザー suzuki での操作をスムーズに行うため、それぞれ別々のユーザーでログインします。

Linux サーバーとは別の端末から SSH でリモートログインして操作を行っている場合には、それぞれのユーザーでリモートログインします。

Linux サーバー上の GUI で操作を行っている場合には、適当なユーザーでログインした後、別々のターミナルを起動し、su コマンドを使ってユーザーを切り替えるとよいでしょう。

ターミナル A でユーザー suzuki に切り替えます。

```
$ sudo su - suzuki
$ id
uid=1001(suzuki) gid=1001(suzuki) groups=1001(suzuki),5001(power)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

ターミナル B でユーザー sato に切り替えます。

```
$ sudo su - sato
$ id
uid=1004(sato) gid=1004(sato) groups=1004(sato)
    context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

4.2 プロセスの実行権の管理

Linux では、root ユーザーを除いて他のユーザーが起動したプロセスを停止させることはできません。

以下の例では、ユーザー sato で vi エディタ（vim）を起動して/tmp にファイルを作成しようとしているプロセスをユーザー suzuki が kill コマンドで停止しようとしますが、停止できません。

ユーザー sato が vi エディタで/tmp/sato を作成します。

```
[sato@vbox ~]$ vi /tmp/sato
```

ユーザー suzuki が vim プロセスを確認します。

```
[suzuki@vbox ~]$ ps aux | grep vim
sato        2351  0.0  0.5 229720  9472 pts/1    S+   18:15   0:00 /usr/bin/vim
    /tmp/sato
suzuki      2355  0.0  0.1 221676  2304 pts/0    S+   18:15   0:00 grep
    --color=auto vim
```

ユーザー suzuki がユーザー sato が実行中の vim エディタのプロセスを kill コマンドで停止しようとしますが、停止できません。指定するプロセス ID は、ps コマンドの 2 番目の表示項目です。

```
[suzuki@vbox ~]$ kill 2351
-bash: kill: (2351) - 許可されていない操作です
```

ユーザー sato は、エディタで「sato」と記述し、「:wq」と入力して vim エディタを終了します。

4.3 ファイルのアクセス権の管理

ユーザー sato が作成したファイル/tmp/sato を使って、アクセス権の動作を検証します。

ユーザー sato でファイル/tmp/sato のアクセス権を確認します。その他のユーザーへのアクセス権は読み取りのみ与えられています。

```
[sato@vbox ~]$ ls -l /tmp/sato
-rw-r--r--. 1 sato sato 5  7月 26 18:17 /tmp/sato
```

ユーザー suzuki で cat コマンドを実行し、ファイル/tmp/sato の内容を確認します。その他のユーザーへの読み取りは許可されているので、内容を確認できます。

```
[suzuki@vbox ~]$ cat /tmp/sato
sato
```

ユーザー suzuki でファイル/tmp/sato に追記してみます。書き込みのアクセス権は与えられていないのでエラーとなります。

```
[suzuki@vbox ~]$ echo "suzuki" >> /tmp/sato
-bash: /tmp/sato: 許可がありません
```

4.4 umask とデフォルトのパーミッションの関係

umask とは、ファイルやディレクトリが新規に作成される際にデフォルトのパーミッションを決定するための値です。umask コマンドで確認できます。

```
$ umask
0022
```

umask の設定値には、新しくファイルを作成する際に設定しない（許可しない）パーミッションを 8 進数で指定します。

	読み取り	書き込み	実行
パーミッション	r	w	x
8 進数値	4	2	1

ファイルとディレクトリでは設定されるデフォルトのパーミッションが変わるので、それぞれ確認してみましょう。

4.4.1 ファイル作成のパーミッションと umask

ファイルが新規作成される際にはファイルの実行パーミッション (eXecute) は設定しないので、0666(rw-rw-rw-) に対して umask の値が適用されます。

umask が 0022 と設定されていると、グループとその他のユーザーの書き込みのパーミッション（w）が設定されていないファイル（-rw-r--r--、0644）が作成されます。

```
$ umask
0022
$ touch testfile
$ ls -l testfile
-rw-r--r--. 1 sato sato 0  7月 26 18:19 testfile
```

4.4.2 ディレクトリ作成のパーミッションと umask

ディレクトリが新規作成される際には、実行パーミッション (eXecute) が必要になるので、0777(rwxrwxrwx) に対して umask の値が適用されます。実行パーミッションが必要になるのは、1 章でも説明したとおり、そのディレクトリをカレントディレクトリにするためには実行パーミッションが必要になるからです。

umask が 0022 と設定されていると、グループとその他のユーザーの書き込みのパーミッション（w）が設定されないディレクトリ（-rwxr-xr-x、0755）が作成されています。

```
$ umask
0022
$ mkdir testdir
$ ls -ld testdir
drwxr-xr-x. 2 sato sato 6  7月 26 18:20 testdir
```

4.4.3 umask が 4 桁の理由

パーミッションは通常、ユーザー、グループ、その他のユーザーの 3 つに対するアクセス権が設定されますが、umask の値は 4 桁になっています。これは、通常のパーミッションの先頭に、setUID/setGID/スティッキービットを表す桁が含まれるためです。setUID などについては後述します。また、通常 setUID などをデフォルトパーミッションとして設定することはないので、umask は先頭を省略して 3 桁で設定することもできます。以下の例では、umask を 022 と 3 桁で設定していますが、umask コマンドの結果は 0022 になっています。

```
$ umask 022
$ umask
0022
```

4.4.4 umask を変更する

umask を変更したい場合には、umask コマンドで設定した umask 値を引数として与えます。以下の例では、umask の値を 0002 に変更したので、新規に作成したファイルのアクセス権は 664(-rw-rw-r--) に設定されています。

```
$ umask 0002
$ touch umasktest
$ ls -l umasktest
-rw-rw-r--. 1 sato sato 0  7月 26 18:21 umasktest
```

4.4.5 デフォルトの umask

デフォルトの umask の値は 0022 ですが、これは/etc/login.defs に UMASK として定義されています。

```
$ cat /etc/login.defs
 （略）
# Default initial "umask" value used by login(1) on non-PAM enabled systems.
```

```
# Default "umask" value for pam_umask(8) on PAM enabled systems.
# UMASK is also used by useradd(8) and newusers(8) to set the mode for new
# home directories if HOME_MODE is not set.
# 022 is the default value, but 027, or even 077, could be considered
# for increased privacy. There is no One True Answer here: each sysadmin
# must make up their mind.
UMASK       022
 （略）
```

また、ログインシェル以外での umask の設定は、bash シェルを起動する際に読み込まれるシェルスクリプト/etc/bashrc の中で umask が設定されています。

```
$ cat /etc/bashrc
 （略）
    # Set default umask for non-login shell only if it is set to 0
    [ `umask` -eq 0 ] && umask 022
 （略）
```

4.5 setUID の確認

setUID が実行ファイルに設定されていると、その実行ファイルは所有ユーザーの権限で実行されます。setUID が設定されている場合、ls コマンドの出力で所有ユーザーの実行パーミッションが「s」と表示されます。

setUID が設定されている例として、passwd コマンドがあります。一般ユーザーがパスワードを変更するには、root ユーザーだけが書き込める/etc/shadow ファイルに対する変更が必要です。パスワードを変更する passwd コマンドは、所有ユーザーが root ユーザーで setUID が設定されているので、一般ユーザーが passwd コマンドを実行すると、root ユーザーの権限で実行されて/etc/shadow ファイルに変更を加えることができます。

コマンドを実行したユーザーを「実行ユーザー」、setUID で権限が変更されたユーザーを「実効ユーザー」と呼びます。

以下の例では、passwd コマンドを一時停止して、ps コマンドで実効ユーザーを確認しています。

setUID が設定されていることを確認します。

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 32656  4月 14  2022 /usr/bin/passwd
```

passwd を実行し、Ctrl+Z キーで一時停止します。一時停止後、シェルプロンプトに戻すためには Enter キーを押す必要があります。

```
$ passwd
ユーザー sato のパスワードを変更。
Current password: ※Enterキーを入力

[1]+  停止                  passwd
$
```

ps コマンドで実効ユーザーを確認します。passwd コマンドの実効ユーザーが root であることが確認できます。

```
$ ps aux | grep passwd
root        2377  0.0  0.4 233672  7296 pts/1    T    18:22   0:00 passwd
sato        2380  0.0  0.1 221676  2304 pts/1    S+   18:23   0:00 grep
    --color=auto passwd
```

fg コマンドで一時停止した passwd コマンドをフォアグラウンドプロセスに戻します。

```
$ fg
passwd
passwd: 認証トークン操作エラー
$
```

4.6 setGID の確認

setGID が設定されていると、所有グループの権限で実行されます。setGID は所有グループの実行パーミッションが「s」と表示されます。

setGID が設定されている例として、write コマンドがあります。

```
$ ls -l /usr/bin/write
-rwxr-sr-x. 1 root tty 23984  3月 13 15:30 /usr/bin/write
```

write コマンドは、ログインしている他のユーザーに対してメッセージを送るコマンドです。以下の例では、write コマンドを一時停止して、ps コマンドで実効グループを確認しています。

2 つのユーザーアカウントでログインします。同じユーザーアカウントでも構いません。su コマンドで変更しているユーザーには write コマンドでメッセージを送ることはできないので、ログインユーザーを確認して write コマンドを実行し、Ctrl+Z キーで一時停止します。

```
[sato@vbox ~]$ write linuc
^Z
[1]+  停止                  write linuc
[sato@vbox ~]$
```

ps コマンドで実効グループを確認します。

```
$ ps a -eo "%p %u %g %G %y %c" | grep write
     37 root     root     root     ?        kworker/R-write
   2388 sato     sato     tty      pts/1    write
```

表示は左から、プロセス ID（%p）、実行ユーザー（%u）、実行グループ（%g）、実効グループ（%G）、実行端末（%y）、コマンド（%c）となっています。実行したのはユーザー sato ですが、setGID されているため tty グループとして動作していることが確認できます。

tty とは「Tele-TYpewriter」の意味で、端末を表します。write コマンドはログインしている他のユーザーの端末にメッセージを表示するために setGID を行って実効グループを tty グループにしているわけです。

一時停止している write コマンドをフォアグラウンドに戻し、終了しておきます。

```
[sato@vbox ~]$ fg
write linuc
^C[sato@vbox ~]$
```

4.7 スティッキービット

スティッキービットが設定されたファイルやディレクトリは、「すべてのユーザーが書き込めるが、所有者しか削除できない」というアクセス権限が設定されます。

たとえば/tmp ディレクトリに対してスティッキービットが設定されています。/tmp ディレクトリは全てのユーザーやアプリケーションが書き込めるディレクトリとして、一時ファイルの作成などに使用されています。しかし/tmp ディレクトリのパーミッションを 777（rwxrwxrwx）に設定すると、作成したファイルを他のユーザーが削除できてしまいます。そこで/tmp ディレクトリにスティッキービットを設定すると、そのファイルを削除できるのは作成したユーザーのみとなります。

スティッキービットが設定されていると、ls コマンドの出力でその他のユーザーの実行パーミッションが「t」と表示されます。

```
$ ls -ld /tmp
drwxrwxrwt. 17 root root 4096  7月 26 18:17 /tmp
```

ユーザー sato で/tmp/sbittest を作成し、パーミッションを 666 に設定します。

```
[sato@vbox ~]$ touch /tmp/sbittest
[sato@vbox ~]$ chmod 666 /tmp/sbittest
```

```
[sato@vbox ~]$ ls -l /tmp/sbittest
-rw-rw-rw-. 1 sato sato 0  7月 26 18:29 /tmp/sbittest
```

ユーザー suzuki で/tmp/sbittest に書き込みをします。その他のユーザーに対する書き込みのパーミッションが付与されているので書き込みが行えます。

```
[suzuki@vbox ~]$ echo "suzuki" >> /tmp/sbittest
[suzuki@vbox ~]$ cat /tmp/sbittest
suzuki
```

ユーザー suzuki で/tmp/sbittest を削除しようとしますが、スティッキービットが働いて削除できません。

```
[suzuki@vbox ~]$ rm /tmp/sbittest
rm: '/tmp/sbittest' を削除できません: 許可されていない操作です
```

ユーザー sato で/tmp/sbittest を削除します。所有ユーザーは削除が行えます。

```
[sato@vbox ~]$ rm /tmp/sbittest
[sato@vbox ~]$ ls -l /tmp/sbittest
ls: '/tmp/sbittest' にアクセスできません:
    そのようなファイルやディレクトリはありません
```

4.8 SELinux

SELinux は Linux カーネル 2.6 から実装された、root ユーザーの特権に対しても制限を掛けることができる強制アクセス制御（MAC、Mandatory Access Control）の仕組みです。

本教科書では、SELinux の基本的な管理について解説します。SELinux のより詳しい説明については、『Linux セキュリティ標準教科書』を参照してください。

4.8.1 SELinux の仕組み

SELinux では、プロセスやファイルなど Linux の全てのリソースに対して「コンテキスト」（contexts）と呼ばれるラベルを付加し、「サブジェクト」（subject。アクセスする側。主にプロセス）が「オブジェクト」（object。アクセスされる側。主にファイルやディレクトリ、プロセス）に対してアクセスを行う際に、そのコンテキストを比較することによりアクセス制御を行います。

複数のコンテキストを組み合わせて、アクセスの可否を行うルールを SELinux では「ポリシー」と呼びます。ポリシーの詳細な説明と修正に関しては、『Linux セキュリティ標準教科書』を参照してください。

4.8.2 SELinux の有効、無効の確認

SELinux の状態は getenforce コマンドで確認できます。

```
$ getenforce
Enforcing
```

getenforce コマンドの結果は以下の通りです。

結果	状態
Enforcing	SELinux によるアクセス制御が有効
Permissive	SELinux は有効であるが動作拒否は行わない
Disabled	SELinux によるアクセス制御が無効

SELinux の状態は、setenforce コマンドによる動的な変更か、設定ファイル/etc/selinux/config による永続的な変更のいずれかで変更できます。

4.8.3 SELinux の強制

最近のディストリビューションでは、SELinux を無効（Disabled）にするのは推奨されず、また動的変更、設定ファイルによる変更も行えなくなっているので、無効化の方法については解説しません。SELinux を無効化するのではなく、正しく設定する方法、また正常に動作しない場合には Permissive に一時的に設定してログを確認し、適切に設定する方法を学んでください。

4.8.4 setenforce コマンドによる SELinux の動的な変更

setenforce コマンドで SELinux の状態を動的に変更できます。変更は root ユーザーで実行する必要があります。

ただし、動的に変更できるのは Enforcing と Permissive の切り替えのみで、SELinux を有効から無効（Disabled）に、あるいは無効から有効に変更することはできません。

```
setenforce [ Enforcing | Permissive | 1 | 0 ]
```

たとえば、システムの SELinux によるアクセス制御を一時的に適用しないようにしたいときには状態を Permissive に変更します。SELinux によるアクセス制御での動作の拒否は行われなくなりますが、デバッグなどの用途のために SELinux のポリシー違反が発生するとログは出力されます。システムが思ったように動作せず、SELinux が原因と思われる時などに Permissive に設定して、SELinux が原因かどうかの切り分け作業を行います。

```
$ sudo setenforce permissive
$ getenforce
Permissive
```

4.8.5 SELinux の永続的な変更

SELinux の設定を永続的に変更するには、SELinux の設定ファイル/etc/selinux/config の設定を変更します。システムを再起動すると、設定が反映されます。

/etc/selinux/config を編集し、設定項目 SELINUX の値を permissive に変更します。

```
$ sudo vi /etc/selinux/config
```

設定を変更します。

```
#SELINUX=enforcing
SELINUX=permissive
```

システムを再起動し、再度ログインして getenforce コマンドで SELinux が Permissive になったことを確認します。

```
$ getenforce
Permissive
```

/etc/selinux/config を編集し、設定項目 SELINUX の値を enforcing に変更します。

```
$ sudo vi /etc/selinux/config
```

```
SELINUX=enforcing
#SELINUX=permissive
```

システムを再起動し、再度ログインして getenforce コマンドで SELinux が有効（Enforcing）になったことを確認します。

```
$ getenforce
Enforcing
```

4.8.6 SELinux のコンテキスト

コンテキストはファイルなどに設定され、SELinux のアクセス制御に利用されます。コンテキストは、次の 4 つの識別子で構成されています。

- ユーザー (user)
- ロール (role)
- タイプ (type)：プロセスの場合には特に「ドメイン」とも言います
- MLS：高度な Multi Level Security を提供できますが、通常のシステムではあまり使われません

コンテキストは、これらの識別子を組み合わせて、以下の形式で表されます。

```
ユーザー:ロール:タイプ:MLSレベル
```

SELinux でのアクセス制御は、タイプ／ドメインに対して許可する動作を定義した「ポリシー」に基づいて行われます。タイプ／ドメインの名前は、役割やプロセス名からつけられています。たとえば、Apache Web サーバーのプロセスである httpd には「httpd_t」というドメインがつけられています。

4.8.7 Apache Web サーバーのインストール

この後の SELinux の動作確認では Apache Web サーバーを使います。まだインストールされていない場合には、dnf コマンドを使ってインストールします。dnf コマンドについては第 5 章で詳しく解説します。

```
$ sudo dnf install httpd -y
```

4.8.8 ファイルのコンテキストの確認方法

SELinux のアクセス制御で用いられるコンテキストは、プロセスやファイルを参照するコマンドに-Z オプションをつけて実行することで確認できます。

たとえば、ファイルやディレクトリに付与されているコンテキストを確認するには ls -lZ コマンドを実行します。例として、Apache Web サーバー（httpd）に関するファイルを確認してみます。

```
$ ls -lZ /var/www
合計 0
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_script_exec_t:s0  6  3月 13
    03:17 cgi-bin
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0      6  3月 13
    03:17 html
```

Web サーバーのコンテンツを含む/var/www/html ディレクトリには「httpd_sys_content_t」というタイプが付与されています。この/var/www/html ディレクトリ内にファイルを作成すると、親ディレクトリのコンテキストに従ってファイルにコンテキストが付与されます。

確認のために、/var/www/html ディレクトリ以下に index.html ファイルを作成してみます。親ディレクトリからコンテキストを継承し、index.html ファイルに「httpd_sys_content_t」というタイプが付与されています。

```
$ sudo touch /var/www/html/index.html
$ ls -lZ /var/www/html/index.html
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0  7月 26
    18:40 /var/www/html/index.html
```

また、プロセスのコンテキストの情報を確認するには、ps axZ コマンドを実行します。

以下の例では、httpd のプロセスを確認すると、httpd_t ドメインが付与されていることが分かります。

```
$ sudo systemctl start httpd
$ ps axZ | grep httpd
system_u:system_r:httpd_t:s0       2412 ?        Ss     0:00 /usr/sbin/httpd
    -DFOREGROUND
system_u:system_r:httpd_t:s0       2413 ?        S      0:00 /usr/sbin/httpd
    -DFOREGROUND
（略）
```

SELinux のポリシーでは、httpd プロセスに付与されている httpd_t ドメインが、「httpd_sys_content_t」などのタイプが付与されているファイルに read（読み取り）などが行えるように権限が設定されています。

4.8.9 Boolean を使った SELinux の制御

SELinux を有効にしてアプリケーションがうまく動作しない場合には、SELinux のアクセス制御によってプロセスがファイルやディレクトリにアクセスできないことが原因の場合があります。そのような時には、SELinux のポリシーを設定する必要があります。

一般的なポリシーの設定は「Boolean」（ブーリアン）と呼ばれる設定の有効、無効で対応できます。Boolean は、パッケージをインストールすると、そのソフトウェア用に用意された Boolean が追加される場合があります。

もし、独自のアプリケーションを使用したり、アプリケーションの設定を大幅に変更した場合には、ポリシーを追加、修正する必要があります。ポリシーの追加、修正方法については『Linux セキュリティ標準教科書』を参照してください。

以下の例では、Apache Web サーバー (httpd) に関するポリシーを設定しています。

getsebool コマンドで Boolean の設定状況一覧を確認します。Boolean 名には関係するプロセス名が含まれているので、grep コマンドで「httpd」をキーワードにして検索します。

```
$ getsebool -a | grep httpd
httpd_anon_write --> off
httpd_builtin_scripting --> on
（略）
httpd_enable_homedirs --> off
（略）
```

後の作業で httpd_enable_homedirs の Boolean を設定します。この Boolean は、Apache Web サーバーのユーザーホームディレクトリ機能に関する設定です。ユーザーホームディレクトリ機能は、各ユーザーのホームディレクトリに作成された public_html ディレクトリ内を Web コンテンツとして公開する仕組みです。

Apache Web サーバーの設定ファイル/etc/httpd/conf.d/userdir.conf を修正し、UserDir ディレクティブを設定してユーザーホームディレクトリ機能を有効にします。

```
$ sudo vi /etc/httpd/conf.d/userdir.conf

（略）
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    #UserDir disabled ※ 行頭に#を追加してコメントアウト

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disabled" line above, and uncomment
    # the following line instead:
    #
    UserDir public_html ※ 行頭の#を削除
（略）
```

httpd サービスを再起動します。

```
$ sudo systemctl restart httpd
```

ユーザー linuc のホームディレクトリに public_html ディレクトリを作成します。

```
$ pwd
/home/linuc
$ mkdir public_html
```

/home/linuc ディレクトリ、/home/linuc/public_html ディレクトリのパーミッションを 711 に設定します。

```
$ chmod 711 /home/linuc
$ chmod 711 /home/linuc/public_html/
```

public_html ディレクトリに index.html ファイルを作成します。

```
$ echo "SELinux test" > /home/linuc/public_html/index.html
```

ブラウザを起動し、「http://localhost/~linuc/」にアクセスします。SELinux のアクセス制御が有効になっているため、「Forbidden」が表示されます。

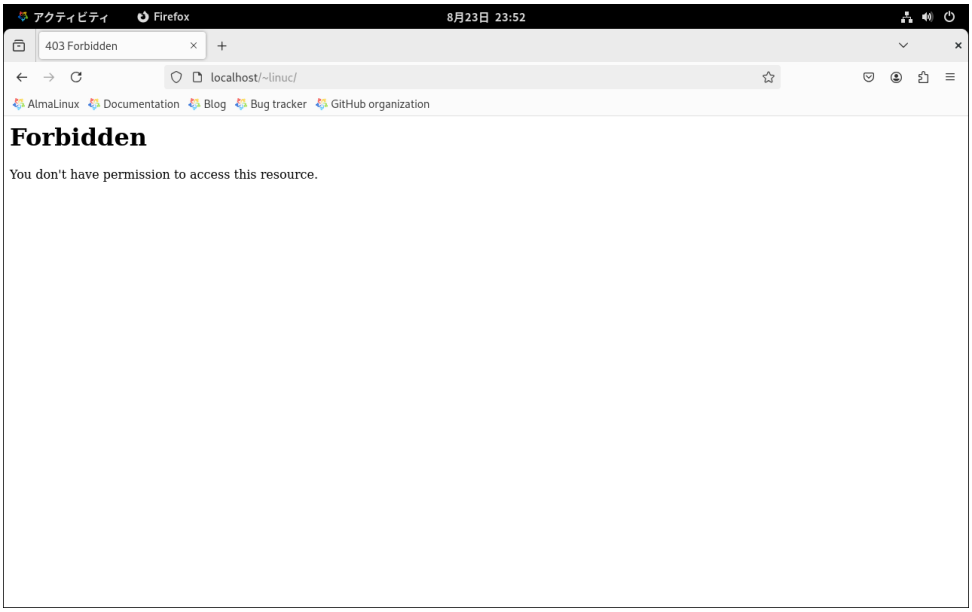

図 13: Forbidden の表示

ログファイル/var/log/audit/audit.log を確認します。httpd(httpd_t) がユーザーホームディレクトリ(user_home_dir_t) にアクセスできなかったというログが出力されています。

```
$ sudo cat /var/log/audit/audit.log | grep index.html
type=AVC msg=audit(1753523214.725:248): avc:  denied  { getattr } for  pid=2624
    comm="httpd" path="/home/linuc/public_html/index.html" dev="dm-0"
    ino=51579776 scontext=system_u:system_r:httpd_t:s0
    tcontext=unconfined_u:object_r:httpd_user_content_t:s0 tclass=file
    permissive=0
type=AVC msg=audit(1753523214.725:249): avc:  denied  { getattr } for  pid=2624
    comm="httpd" path="/home/linuc/public_html/index.html" dev="dm-0"
    ino=51579776 scontext=system_u:system_r:httpd_t:s0
    tcontext=unconfined_u:object_r:httpd_user_content_t:s0 tclass=file
    permissive=0
```

setsebool コマンドを実行して、Boolean「httpd_enable_homedirs」を有効に設定します。

```
$ getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
$ sudo setsebool httpd_enable_homedirs on
$ getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

再度ブラウザで「http://localhost/~linuc/」にアクセスします。Boolean でアクセスが許可されたので、作成したページが表示されます。

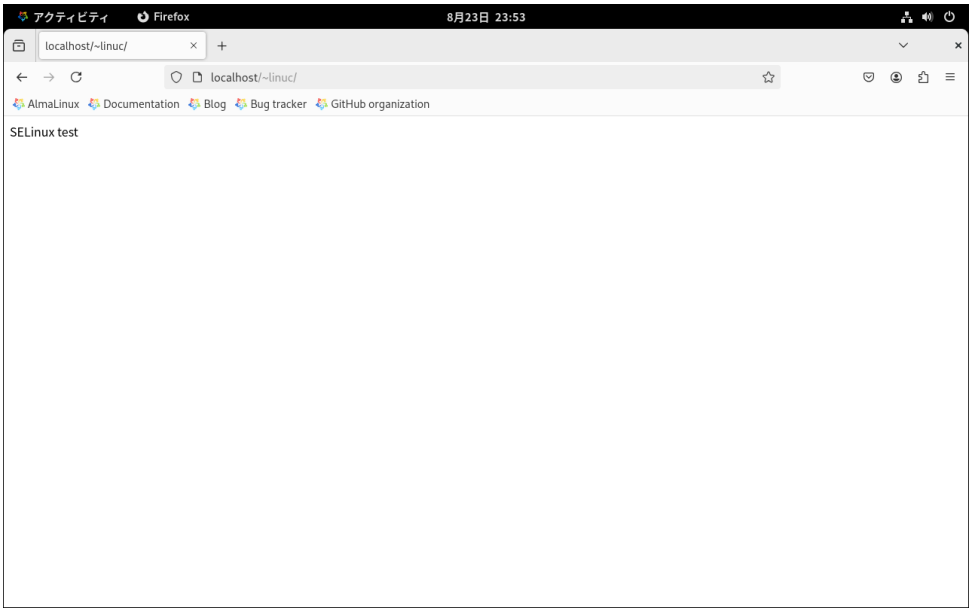

図 14: SELinux test の表示

4.9　LVM の設定

LVM（Logical Volume Manager）は、ハードディスクなどの記憶媒体の物理的な状態を隠蔽し、論理的なイメージで管理するための技術です。

LVM を使うことで、複数のハードディスクにまたがったボリュームが作成できるようになり、ファイルシステムの容量が足りなくなった場合の容量の追加が簡単になります。ボリュームの操作は、システムを再起動することなく行うことができます。

また、ハードディスクに障害が発生した時には、新しい HDD を追加して、壊れている HDD を外すなどの障害対応が容易になります。他にも、スナップショットを取ることができるなどのメリットがあります。

現在の一般的な Linux ディストリビューションでは、インストール時に LVM でパーティションを作成できます。AlmaLinux では、インストール時に自動パーティション設定を選択すると、デフォルトで LVM を使用してストレージを設定します。

LVM の詳しい説明に関しては、『高信頼システム構築標準教科書』を参照してください。

4.9.1　実習用ディスクの追加

以下の実習では、ディスクを追加し、LVM で管理します。そのための仮想ハードディスクを仮想マシンに追加します。

1. ゲスト OS をシャットダウンし、仮想マシンを停止する
2. VirtualBox の仮想マシンの「設定」で「ストレージ」を開く
3. 「コントローラー:SATA」で仮想ハードディスク追加のボタンをクリックする
4. 「ハードディスク選択」ダイアログで「作成」をクリックする
5. 「仮想ハードディスクの作成」ダイアログで「完了」をクリックする
6. 作成された仮想ハードディスクを選択し、「選択」をクリックする
7. 「コントローラー:SATA」に仮想ハードディスクが追加されたことを確認して、「OK」をクリックする
8. 仮想マシンを起動する

4.9.2　デバイス名の確認

仮想ハードディスクのデバイス名を確認します。ストレージデバイスの確認には lsblk コマンドを実行します。

```
$ lsblk
NAME                    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                       8:0    0   20G  0 disk
├─sda1                    8:1    0  600M  0 part /boot/efi
├─sda2                    8:2    0    1G  0 part /boot
```

```
└─sda3                      8:3    0 18.4G  0 part
  ├─almalinux_vbox-root 253:0    0 16.4G  0 lvm  /
  └─almalinux_vbox-swap 253:1    0    2G  0 lvm  [SWAP]
sdb                       8:16   0   20G  0 disk
sr0                      11:0    1 1024M  0 rom
```

sdb が新たに追加した仮想ハードディスクのデバイス名です。

/dev ディレクトリにもデバイスファイルが作成されています。

```
$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sdb
```

/dev/sdb が新たに追加した仮想ハードディスクのデバイスファイルです。

4.9.3 物理ボリューム（PV）

LVM は、物理ボリューム（PV: Physical Volume）、ボリュームグループ（VG: Volume Group）、論理ボリューム（LV: Logical Volume）の 3 つで構成されています。

物理ボリューム (PV) は、物理ディスクのパーティション単位で扱われます。一つの物理ディスクすべてを一つの PV として扱うこともできますし、一つの物理ディスク内にパーティションを複数作成し、それぞれのパーティションを別々の PV として扱うこともできます。

PV を作成するには、パーティションを作成し、パーティションタイプを 8E に設定します。

以下の例では、Linux マシンに新規に追加した/dev/sdb として認識されているハードディスクを LVM で使用できるよう、fdisk でパーティションを作成して PV として設定しています。同時に、後の作業で領域拡張を行うための追加パーティションも作成しておきます。

```
$ sudo fdisk /dev/sdb

fdisk (util-linux 2.37.4) へようこそ。
ここで設定した内容は、書き込みコマンドを実行するまでメモリのみに保持されます。
書き込みコマンドを使用する際は、注意して実行してください。

デバイスには認識可能なパーティション情報が含まれていません。
新しい DOS ディスクラベルを作成しました。識別子は 0x3370db49 です。

コマンド (m でヘルプ): n ※ 新規パーティション作成のnを入力
パーティションタイプ
   p   基本パーティション (0 プライマリ, 0 拡張, 4 空き)
   e   拡張領域 (論理パーティションが入ります)
選択 (既定値 p): p ※ 基本パーティションのpを入力
パーティション番号 (1-4, 既定値 1): 1 ※ パーティション番号1を入力
最初のセクタ (2048-41943039, 既定値 2048): ※ デフォルトを使うのでEnterを入力
最終セクタ, +/-セクタ番号 または +/-サイズ{K,M,G,T,P} (2048-41943039, 既定値
    41943039): +10GB ※ +10GBを入力

新しいパーティション 1 をタイプ Linux、サイズ 9.3 GiB で作成しました。

コマンド (m でヘルプ): n ※ 新規パーティション作成のnを入力
パーティションタイプ
   p   基本パーティション (1 プライマリ, 0 拡張, 3 空き)
   e   拡張領域 (論理パーティションが入ります)
選択 (既定値 p): p ※ 基本パーティションのpを入力
パーティション番号 (2-4, 既定値 2): 2 ※ パーティション番号2を入力
最初のセクタ (19533824-41943039, 既定値 19533824): ※
    デフォルトを使うのでEnterを入力
最終セクタ, +/-セクタ番号 または +/-サイズ{K,M,G,T,P} (19533824-41943039,
    既定値 41943039): ※ デフォルトを使うのでEnterを入力
```

```
新しいパーティション 2 をタイプ Linux、サイズ 10.7 GiB で作成しました。

コマンド (m でヘルプ): p ※ 設定を確認するのでpを入力
ディスク /dev/sdb: 20 GiB, 21474836480 バイト, 41943040 セクタ
ディスク型式: VBOX HARDDISK
単位: セクタ (1 * 512 = 512 バイト)
セクタサイズ (論理 / 物理): 512 バイト / 512 バイト
I/O サイズ (最小 / 推奨): 512 バイト / 512 バイト
ディスクラベルのタイプ: dos
ディスク識別子: 0x3370db49

デバイス    起動 開始位置 終了位置   セクタ サイズ Id タイプ
/dev/sdb1           2048 19533823 19531776   9.3G 83 Linux
/dev/sdb2       19533824 41943039 22409216  10.7G 83 Linux

コマンド (m でヘルプ): t ※ パーティションのタイプを変更するtを入力
パーティション番号 (1,2, 既定値 2): 1 ※ パーティション番号1を入力
16 進数コード または別名 (L で利用可能なコードを一覧表示します): 8e ※ Linux
    LVMのコード8eを入力

パーティションのタイプを 'Linux' から 'Linux LVM' に変更しました。

コマンド (m でヘルプ): t ※ パーティションのタイプを変更するtを入力
パーティション番号 (1,2, 既定値 2): 2 ※ パーティション番号2を入力
16 進数コード または別名 (L で利用可能なコードを一覧表示します): 8e ※ Linux
    LVMのコード8eを入力

パーティションのタイプを 'Linux' から 'Linux LVM' に変更しました。

コマンド (m でヘルプ): p ※ 設定を確認するのでpを入力
ディスク /dev/sdb: 20 GiB, 21474836480 バイト, 41943040 セクタ
ディスク型式: VBOX HARDDISK
単位: セクタ (1 * 512 = 512 バイト)
セクタサイズ (論理 / 物理): 512 バイト / 512 バイト
I/O サイズ (最小 / 推奨): 512 バイト / 512 バイト
ディスクラベルのタイプ: dos
ディスク識別子: 0x3370db49

デバイス    起動 開始位置 終了位置   セクタ サイズ Id タイプ
/dev/sdb1           2048 19533823 19531776   9.3G 8e Linux LVM
/dev/sdb2       19533824 41943039 22409216  10.7G 8e Linux LVM

コマンド (m でヘルプ): w ※ パーティション情報を書き込むwを入力
パーティション情報が変更されました。
ioctl() を呼び出してパーティション情報を再読み込みします。
ディスクを同期しています。
```

デバイスファイルを確認します。

```
$ lsblk /dev/sdb
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sdb      8:16   0   20G  0 disk
├─sdb1   8:17   0  9.3G  0 part
└─sdb2   8:18   0 10.7G  0 part
$ ls /dev/sdb*
/dev/sdb  /dev/sdb1  /dev/sdb2
```

デバイスファイルに/dev/sdb1 と/dev/sdb2 が追加されたのが確認できます。

4.9.4 ボリュームグループ（VG）

ボリュームグループ (VG) は、1 つ以上の物理ボリューム（PV）をひとまとめにしたものです。これは仮想的なディスクに相当します。

ボリュームグループは vgcreate コマンドで作成します。

```
vgcreate ボリューム名 PVデバイス名 [PVデバイス名 ...]
```

たとえば、物理ボリューム（PV）として作成した/dev/sdb1 を使って Volume00 という名前のボリュームグループを作成するには、以下の vgcreate コマンドを実行します。

```
$ sudo vgcreate Volume00 /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
  Volume group "Volume00" successfully created
```

また、ボリュームグループの情報は vgscan コマンドで確認できます。

```
$ sudo vgscan
  Found volume group "almalinux_vbox" using metadata type lvm2
  Found volume group "Volume00" using metadata type lvm2
```

4.9.5 論理ボリューム（LV）

論理ボリューム（LV）は、ボリュームグループ（VG）上に作成する仮想的なパーティションです。Linux からはデバイスとして認識されます。ハードディスクに物理パーティションを作成する場合と同様に、ボリュームグループをすべて一つの論理ボリュームとすることもできますし、一つのボリュームグループを複数の論理ボリュームに分割して使用することもできます。

論理ボリュームは lvcreate コマンドを使って作成します。

```
lvcreate -L サイズ -n 論理ボリューム名 ボリュームグループ名
```

たとえば、ボリュームグループ Volume00 にサイズ 1GB、論理ボリューム名「LogVol01」の論理ボリュームを作成するには、以下の lvcreate コマンドを実行します。

```
$ sudo lvcreate -L 1024M -n LogVol01 Volume00
  Logical volume "LogVol01" created.
```

4.9.6 論理ボリュームにファイルシステムの作成

作成した論理ボリュームを利用するには、通常のパーティションと同じく論理ボリューム上にファイルシステムを作成します。論理ボリュームは、以下のようなデバイスとして扱うことができます。

```
/dev/ボリュームグループ名/論理ボリューム名
```

/dev/Volume00/LogVol01 上に ext4 ファイルシステムを作成するために、mkfs コマンドを実行します。

```
$ sudo mkfs -t ext4 /dev/Volume00/LogVol01
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 5ca94cf6-2d63-4ce2-89da-c2dc251d9e42
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

mount コマンドを使って、/dev/Volume00/LogVol01 をマウントします。

```
$ sudo mkdir /mnt/LVMtest
$ sudo mount -t ext4 /dev/Volume00/LogVol01 /mnt/LVMtest
$ mount | grep /mnt/LVMtest
/dev/mapper/Volume00-LogVol01 on /mnt/LVMtest type ext4 (rw,relatime,seclabel)
```

4.9.7 ボリュームグループへのディスクの追加

既存のボリュームグループ Volume00 に物理ボリューム/dev/sdb2 を追加します。

vgextend コマンドを実行して、物理ボリューム/dev/sdb2 をボリュームグループ Volume00 に追加します。

```
$ sudo vgextend Volume00 /dev/sdb2
  Physical volume "/dev/sdb2" successfully created.
  Volume group "Volume00" successfully extended
```

vgdisplay コマンドを実行して、ボリュームグループ Volume00 の情報を確認します。PV（Physical volume）の数が 2 となっており、/dev/sdb2 が加わっていることが分かります。

```
$ sudo vgdisplay Volume00
  --- Volume group ---
  VG Name               Volume00
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               <20.00 GiB
  PE Size               4.00 MiB
  Total PE              5119
  Alloc PE / Size       256 / 1.00 GiB
  Free  PE / Size       4863 / <19.00 GiB
  VG UUID               HsR4Jt-H94l-Ulsw-rvaz-Jebs-6my2-eGtfcD
```

4.9.8 論理ボリュームの拡張

LVM では、論理ボリュームのサイズを変更できます。また、LVM の論理ボリューム上に作成された ext4 ファイルシステムは、ファイルシステムをマウントしたまま拡張できます。

df コマンドを実行して、現在のファイルシステムの容量を確認します。現在の容量は 1GB です。

```
$ df -h /mnt/LVMtest
ファイルシス                  サイズ  使用  残り 使用% マウント位置
/dev/mapper/Volume00-LogVol01   974M   24K  907M    1% /mnt/LVMtest
```

lvextend コマンドを実行して、論理ボリューム LogVol01 のサイズを 2G まで拡大します。

```
$ sudo lvextend -L 2G /dev/Volume00/LogVol01
  Size of logical volume Volume00/LogVol01 changed from 1.00 GiB (256 extents)
      to 2.00 GiB (512 extents).
  Logical volume Volume00/LogVol01 successfully resized.
```

resize2fs コマンドを実行して、ファイルシステムを拡大します。

```
$ sudo resize2fs /dev/Volume00/LogVol01
resize2fs 1.46.5 (30-Dec-2021)
Filesystem at /dev/Volume00/LogVol01 is mounted on /mnt/LVMtest; on-line
    resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/Volume00/LogVol01 is now 524288 (4k) blocks long.
```

df コマンドで再度容量を確認します。容量が 2GB に増えていることが確認できます。

```
$ df -h /mnt/LVMtest
ファイルシス                    サイズ  使用  残り 使用% マウント位置
/dev/mapper/Volume00-LogVol01   2.0G   24K  1.9G    1% /mnt/LVMtest
```

4.9.9 論理ボリュームの縮小

運用上、他の論理ボリュームを拡大したい等の理由で使用率の低い論理ボリュームの縮小を行う場合があります。論理ボリュームを縮小するには、先にファイルシステムを縮小し、その後に論理ボリュームを縮小する必要があります。ファイルシステムの縮小はマウントしたままでは行えないので、作業中は一度アンマウントしておく必要があります。

縮小したいボリュームをアンマウントします。umount コマンドを実行して、/mnt/LVMtest をアンマウントします。

```
$ sudo umount /mnt/LVMtest
```

縮小したい論理ボリューム/dev/Volume00/LogVol01 に対して fsck コマンドを実行します。強制的にチェックを行うために-f オプションを付与して実行します。

```
$ sudo fsck -f /dev/Volume00/LogVol01
fsck from util-linux 2.37.4
e2fsck 1.46.5 (30-Dec-2021)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/mapper/Volume00-LogVol01: 11/131072 files (0.0% non-contiguous),
    17196/524288 blocks
```

resize2fs コマンドを実行して、ファイルシステムを縮小します。例として、1GB まで縮小します。

```
$ sudo resize2fs /dev/Volume00/LogVol01 1G
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/Volume00/LogVol01 to 262144 (4k) blocks.
The filesystem on /dev/Volume00/LogVol01 is now 262144 (4k) blocks long.
```

lvreduce コマンドを実行して、論理ボリューム/dev/Volume00/LogVol01 を縮小します。

```
$ sudo lvreduce -L 1G /dev/Volume00/LogVol01
  File system ext4 found on Volume00/LogVol01.
  File system size (1.00 GiB) is equal to the requested size (1.00 GiB).
  File system reduce is not needed, skipping.
  Size of logical volume Volume00/LogVol01 changed from 2.00 GiB (512 extents)
      to 1.00 GiB (256 extents).
  Logical volume Volume00/LogVol01 successfully resized.
```

/mnt/LVMtest に再マウントして、容量を確認します。

```
$ sudo mount -t ext4 /dev/Volume00/LogVol01 /mnt/LVMtest
$ df -h /mnt/LVMtest
ファイルシス                  サイズ  使用  残り 使用% マウント位置
/dev/mapper/Volume00-LogVol01   974M   24K  912M    1% /mnt/LVMtest
```

5 システムのメンテナンス

5.1 パッケージ管理

Linux の各ディストリビューションでは、Linux カーネルやアプリケーションなどシステムに必要なソフトウェアをパッケージ形式でインストール、管理する仕組みを持っています。パッケージは、たとえば Linux カーネルであればカーネル本体およびカーネルモジュールを 1 つのパッケージファイルにまとめたものです。

Red Hat Enterprise Linux や AlmaLinux、Rocky Linux、SUSE Linux などでは、パッケージ管理に RPM(Red Hat Package Manager) が採用されています。また、アップデート時のバージョン管理等を行う為に開発された Yum（Yellowdog Updater Modified）が、システムの更新等で広く使われるようになり、現在では Yum の後継である DNF（Dandified Yum）が使われるようになっています。

Debian GNU/Linux や Ubuntu などのディストリビューションでは、パッケージ管理に Debian パッケージ（deb 形式）が採用されています。パッケージ管理ツールとして APT（Advanced Package Tool）が使われています。

本教科書では、AlmaLinux のパッケージ管理ツールである DNF の使用方法について解説します。

5.1.1 DNF とは

以前は RPM パッケージの管理には rpm コマンドが使用されていましたが、パッケージ間の依存関係を自動的に解決できなかったため、パッケージのインストールを行う際に管理者が自分で依存関係を確認しながら、手動で必要なパッケージを指定する必要がありました。そのため、依存関係で必要となるパッケージが多数あった場合、インストール作業が大変でした。

DNF では、dnf コマンドを使ったパッケージのインストール時に依存関係の解決を自動的に行い、必要となるパッケージも同時にインストールするため、パッケージ管理が簡単になっています。

DNF やその前に使われていた Yum の後継のため、現在でも設定ファイルなどは Yum の頃と同じものが使われています。また、dnf コマンドと yum コマンドの間で基本的なサブコマンドには互換性があります。

5.1.2 DNF のリポジトリの設定

DNF では、RPM パッケージをまとめて置いておく場所を「リポジトリ」と呼びます。パッケージのインストールの際には、リポジトリから必要な RPM パッケージを取得します。

リポジトリの設定ファイルは/etc/yum.repos.d ディレクトリにあります。

```
$ ls /etc/yum.repos.d
almalinux-appstream.repo  almalinux-extras.repo            almalinux-plus.repo
                  almalinux-sap.repo
almalinux-baseos.repo     almalinux-highavailability.repo
    almalinux-resilientstorage.repo  almalinux-saphana.repo
almalinux-crb.repo        almalinux-nfv.repo               almalinux-rt.repo
```

それぞれのリポジトリ設定ファイルに、リポジトリの参照先などが記述されています。

デフォルトで利用される almalinux-baseos.repo の中身は以下の通りです。

```
$ cat /etc/yum.repos.d/almalinux-baseos.repo
[baseos]
name=AlmaLinux $releasever - BaseOS
mirrorlist=https://mirrors.almalinux.org/mirrorlist/$releasever/baseos
# baseurl=https://repo.almalinux.org/almalinux/$releasever/BaseOS/$basearch/os/
enabled=1
gpgcheck=1
countme=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-AlmaLinux-9
metadata_expire=86400
enabled_metadata=1
（略）
```

設定項目 mirrorlist で指定した mirrors.almalinux.org は、リポジトリのミラーリストから返される、ネットワークで接続しやすいリポジトリのアドレスに置き換えられます。

設定項目 enabled の値が 0 に設定されていると、そのリポジトリは dnf コマンドの- -enablerepo オプションで指定されなければ参照しません。

5.1.3 DNF で PROXY サーバーを使う場合

dnf コマンドは、インターネットに接続し、HTTP でインストールするパッケージをダウンロードすることができる必要があります。もし、PROXY サーバーを経由する必要がある場合には、dnf コマンドの設定ファイル/etc/dnf/dnf.conf に PROXY サーバーに関する設定を記述する必要があります。設定項目は以下の通りです。

項目	設定する値
proxy	PROXY サーバーの URL
proxy_username	PROXY サーバーの認証ユーザ名
proxy_password	PROXY サーバーの認証パスワード

また、インターネットに接続できない場合には、インストール用の DVD メディアをリポジトリとして参照する方法が利用できます。具体的な手順は後述します。

5.1.4 dnf コマンドの基本的な使い方

dnf コマンドは、引数として様々なサブコマンドを指定して使用します。主なサブコマンドは以下の通りです。

5.1.5 パッケージのインストール

指定されたパッケージをインストールします。

```
dnf install パッケージ名
```

5.1.6 パッケージのアンインストール（削除）

指定されたパッケージをアンインストール（削除）します。

```
dnf remove パッケージ名
```

5.1.7 パッケージの更新の確認

インストールされているパッケージの更新があるかを確認します。

```
dnf check-update
```

5.1.8 パッケージの更新

インストールされているパッケージを更新します。パッケージ名を指定しなかった場合には、すべての更新可能なパッケージが対象となります。

```
dnf update [パッケージ名]
```

5.1.9 パッケージグループの一覧表示

利用可能なパッケージグループを表示します。

```
dnf grouplist
```

5.1.10 パッケージグループのインストール

指定されたパッケージグループに含まれるすべてのパッケージをまとめてインストールします。

```
dnf groupinstall パッケージグループ名
```

5.1.11 パッケージグループのアンインストール（削除）

指定されたパッケージグループに含まれるすべてのパッケージをまとめてアンインストール（削除）します。

```
dnf groupremove パッケージグループ名
```

5.1.12 パッケージグループを指定したインストール

dnf コマンドを使ってパッケージグループを指定したインストールを行います。

以下の例では、コンパイラーなどが含まれている「開発ツール」パッケージグループをインストールします。

利用可能なパッケージグループを表示します。

```
$ dnf grouplist
AlmaLinux 9 - AppStream                          3.6 kB/s | 4.2 kB     00:01
AlmaLinux 9 - AppStream                          6.9 MB/s |  14 MB     00:02
AlmaLinux 9 - BaseOS                             5.6 kB/s | 3.8 kB     00:00
AlmaLinux 9 - BaseOS                             7.1 MB/s |  15 MB     00:02
AlmaLinux 9 - Extras                             5.4 kB/s | 3.8 kB     00:00
AlmaLinux 9 - Extras                              27 kB/s |  20 kB     00:00
利用可能な環境グループ:
   サーバー
   最小限のインストール
（略）
インストール済みの環境グループ:
   サーバー (GUI 使用)
インストール済みのグループ:
   コンテナー管理
   ヘッドレス管理
利用可能なグループ:
   コンソールインターネットツール
   .NET Development
   RPM 開発ツール
   開発ツール
（略）
```

「開発ツール」パッケージグループをインストールします。

```
$ sudo dnf groupinstall "開発ツール" -y
メタデータの期限切れの最終確認: 2:33:22 前の 2025年07月27日 11時15分35秒
    に実施しました。
依存関係が解決しました。
================================================================================
 パッケージ                                          アーキテクチャー
     バージョン                                  リポジトリー
     サイズ
================================================================================
group/moduleパッケージをインストール:
 asciidoc                                            noarch
     9.1.0-3.el9                                 appstream                   237
     k
 autoconf                                            noarch
     2.69-39.el9                                 appstream                   665
     k
```

```
 automake                                              noarch
     1.16.2-8.el9                                   appstream                      662
     k
 （略）
  valgrind-1:3.24.0-3.el9.x86_64
      valgrind-devel-1:3.24.0-3.el9.x86_64
  xorg-x11-fonts-ISO8859-1-100dpi-7.5-33.el9.noarch
      xz-devel-5.2.5-8.el9_0.x86_64
  zlib-devel-1.2.11-40.el9.x86_64
      zstd-1.5.5-1.el9.x86_64

完了しました!
```

5.1.13 パッケージグループ名を英語表記で表示する

dnf コマンドはロケール（Locale）に対応しているため、環境変数 LANG の値が日本語に設定されているとパッケージグループ名が日本語で表示されます。このため、dnf groupinstall コマンドを実行する際に日本語でパッケージグループ名を指定しなければなりません。

日本語がうまく表示できない、あるいは日本語が入力できない環境の場合、dnf コマンドの前に「LANG=C」と入力することでパッケージグループ名を英語表記で表示できます。この方法は環境変数 LANG の値を一時的に変更した状態で、dnf コマンドを実行したことになります。

```
$ LANG=C dnf grouplist
Last metadata expiration check: 0:04:17 ago on Sun Jul 27 13:46:51 2025.
Available Environment Groups:
   Server
   Minimal Install
   Workstation
 （略）
```

インストール時には、環境変数 LANG を指定しないでも英語表記のままパッケージグループ名を指定できます。パッケージグループ名に空白が含まれている場合には「“」（ダブルクォート）でパッケージグループ名を括って下さい。

以下の例では、開発ツール（Development Tools）パッケージグループを指定して、コンパイラなどをインストールしています。

```
$ sudo dnf groupinstall "Development Tools"
```

5.1.14 インストール DVD メディアをリポジトリにする方法

インターネットに接続できない環境で dnf コマンドを利用する方法として、ISO イメージや DVD などのインストールメディアをリポジトリとして参照させる方法があります。

インストールメディアを/run/media/linuc ディレクトリにマウントし、リポジトリ設定ファイルを作成して、dnf コマンドにリポジトリ指定のオプションをつけて実行する必要があります。

以下の手順で、インストール DVD メディアをリポジトリとして参照できるようにします。

1. AlmaLinux に linuc ユーザーとしてグラフィカルログインします。
2. インストールメディアを DVD ドライブに挿入します。仮想マシンの場合には、ISO イメージファイルを仮想 DVD ドライブで参照します。VirtualBox の場合、「デバイス」メニューから「光学ドライブ」を選択し、ISO イメージファイルを選択します。
3. 自動マウントされることを確認します。
4. mount コマンドで確認します。インストールメディアは/run/media/linuc ディレクトリ内にマウントされています。
5. マウントされたディレクトリを参照するリポジトリ設定ファイルを作成します。

自動マウントされたディレクトリを確認します。

```
$ mount
（略）
/dev/sr0 on /run/media/linuc/AlmaLinux-9-6-x86_64-dvd type iso9660
    (ro,nosuid,nodev,relatime,nojoliet,check=s,map=n,blocksize=2048,
uid=1000,gid=1000,dmode=500,fmode=400,uhelper=udisks2)
```

インストールメディアをマウントした後、リポジトリの設定ファイル/etc/yum.repos.d/almalinux-media.repoを作成します。内容は以下の通りです。

```
$ sudo vi /etc/yum.repos.d/almalinux-media.repo
```

```
[media_BaseOS]
name=AlmaLinux 9 Media - BaseOS
baseurl=file:///run/media/linuc/AlmaLinux-9-6-x86_64-dvd/BaseOS/
enabled=0
gpgcheck=0
gpgkey=file:///run/media/linuc/AlmaLinux-9-6-x86_64-dvd/RPM-GPG-KEY-AlmaLinux-9

[media_AppStream]
name=AlmaLinux 9 Media - AppStream
baseurl=file:///run/media/linuc/AlmaLinux-9-6-x86_64-dvd/AppStream/
enabled=0
gpgcheck=0
gpgkey=file:///run/media/linuc/AlmaLinux-9-6-x86_64-dvd/RPM-GPG-KEY-AlmaLinux-9
```

dnfコマンドを実行します。--disablerepoオプションですべてのリポジトリを参照不要とし、--enablerepoオプションでmediaリポジトリのみ参照するように指定します。以下の例では、グループリストを取得しています。

```
$ sudo dnf --disablerepo=\* --enablerepo=media* grouplist
AlmaLinux 9 Media - BaseOS

    157 MB/s | 2.3 MB     00:00
AlmaLinux 9 Media - AppStream

    217 MB/s | 8.1 MB     00:00
メタデータの期限切れの最終確認: 0:00:01 前の 2025年07月27日 13時57分30秒
    に実施しました。
利用可能な環境グループ:
   サーバー
   最小限のインストール
（略）
```

5.2 システム監視

システムを適切に運用していく上で、システム上のリソースが常に有効に使われているか、極端にリソースを占有しているプロセスは無いかを監視することは重要な事です。

システム上のリソースを様々な角度で監視する方法を解説します。

5.2.1 topコマンドによるシステムリソース監視

topコマンドは、システムのどのプロセスがどの程度のCPUやメモリなどのリソースを消費しているかを簡単に示してくれる対話型のコマンドです。

topコマンドを実行すると、デフォルトでは先頭五行（サマリーエリア）にシステム全体の情報が表示されます。次の行が対話的にコマンドを入力するエリアです。その次の行から、プロセス毎の情報が表示されます。

```
top - 13:59:16 up  3:35,  3 users,  load average: 0.00, 0.05, 0.05
Tasks: 297 total,   1 running, 296 sleeping,   0 stopped,   0 zombie
```

```
%Cpu(s):  0.2 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   1700.3 total,    593.9 free,    895.9 used,    305.6 buff/cache
MiB Swap:   2048.0 total,   1916.2 free,    131.8 used.    804.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   4310 linuc     20   0  226132   3840   3072 R   0.3   0.2   0:00.71 top
      1 root      20   0  116536  15320   7328 S   0.0   0.9   0:01.13 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_wo+
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
     11 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
     14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
     15 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.06 ksoftir+
     17 root      20   0       0      0      0 I   0.0   0.0   0:00.93 rcu_pre+
     18 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_exp+
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_exp+
     20 root      rt   0       0      0      0 S   0.0   0.0   0:00.17 migrati+
```

先頭の 5 行には、以下の情報が表示されています。

行数	意味
1 行目	起動時間、ログインユーザ数、ロードアベレージ
2 行目	各状態のタスク数
3 行目	CPU の使用状況
4 行目	物理メモリの使用状況
5 行目	スワップ領域の使用状況

5.2.2 vmstat コマンドによるシステムリソース監視

vmstat コマンドは、メモリの使用状況や CPU の負荷などを表示するコマンドです。

vmstat コマンドを引数無しで実行すると、コマンドを実行した時点のメモリや CPU、ディスクの使用状況が表示されます。

```
$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0   7472 173780   2332 614608    0    0    50    56   86   90  1  1 98  0  0
```

表示される内容は以下の表のとおりです。

項目	意味
r	実行待ちプロセス数
b	割り込み不可のスリープ状態にあるプロセス数
swpd	スワップアウトされたメモリ量
free	空きメモリの容量
buff	バッファに使用されているメモリの容量
cache	キャッシュに使用されているメモリの容量
si	1 秒あたりのディスクからスワップインされているメモリの容量
so	1 秒あたりのディスクにスワップアウトされているメモリの容量
bi	1 秒あたりのブロックデバイスに送られたブロック
bo	1 秒あたりのブロックデバイスから受け取ったブロック

項目	意味
in	1 秒あたりの割り込みの回数
cs	1 秒あたりのコンテキストスイッチの回数
us	CPU 総時間当たりのユーザー時間の割合
sy	CPU 総時間当たりのシステム時間の割合
id	CPU 総時間当たりのアイドル時間の割合

vmstat コマンドに引数として数値を与えると、指定された数値の秒間隔でシステムのリソース情報を出力し続けます。終了するには Ctrl+C キーを入力します。

```
$ vmstat 5
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0   7472 173780   2332 614656    0    0    50    56   85   90  1  1 98  0  0
 0  0   7472 173532   2332 614744    0    0    18     0  180  126  2  2 96  0  0
 0  0   7472 173532   2332 614748    0    0     1     0  135   95  1  1 98  0  0
 ^C
```

5.2.3 sysstat によるシステムリソース監視

稼働中の Linux のシステム情報を継続して集めるには、sysstat パッケージに含まれている iostat コマンドや sar コマンドなどを使うと便利です。

sysstat パッケージをインストールします。

```
$ sudo dnf install sysstat
```

sysstat サービスを起動します。

```
$ sudo systemctl start sysstat
$ sudo systemctl enable sysstat
```

sysstat パッケージをインストールすると、タイマーが設定されます。

```
$ systemctl list-timers | grep sysstat
Sun 2025-07-27 14:10:00 JST 4min 15s left -                             -
               sysstat-collect.timer        sysstat-collect.service
Mon 2025-07-28 00:07:00 JST 10h left      -                             -
               sysstat-summary.timer        sysstat-summary.service
```

デフォルトで 10 分間隔でシステムのリソース情報が取得されるようにタイマーが設定されます。

```
$ systemctl cat sysstat-collect.timer
# /usr/lib/systemd/system/sysstat-collect.timer
 （略）
[Timer]
OnCalendar=*:00/10
 （略）
```

また、1 日に 1 回、取得した情報のサマリーを作成するようにタイマーが設定されます。

```
$ systemctl cat sysstat-summary.timer
# /usr/lib/systemd/system/sysstat-summary.timer
 （略）
[Timer]
OnCalendar=00:07:00
 （略）
```

まとめられた情報は、sar コマンドで参照できます。

```
$ sar
Linux 5.14.0-570.33.2.el9_6.x86_64 (vbox)   2025年08月24日      _x86_64_    (2
    CPU)

00時00分08秒      CPU      %user      %nice   %system   %iowait    %steal      %idle
00時10分05秒      all       0.41       0.00      0.46      0.05      0.00      99.08
00時20分07秒      all       0.17       0.01      0.22      0.01      0.00      99.60
00時30分08秒      all       0.33       0.00      0.35      0.01      0.00      99.31
00時40分00秒      all       0.12       0.00      0.22      0.01      0.00      99.65
15時44分12秒      all       0.65       0.00      0.57      0.17      0.00      98.62
15時50分03秒      all       0.35       0.00      0.38      0.07      0.00      99.19
16時00分05秒      all       0.28       0.00      0.36      0.02      0.00      99.35
16時10分06秒      all       0.73       0.00      0.70      0.07      0.00      98.50
16時20分08秒      all       0.25       0.00      0.25      0.02      0.00      99.48
16時30分00秒      all       0.03       0.01      0.12      0.02      0.00      99.82
16時40分01秒      all       0.01       0.00      0.09      0.00      0.00      99.89
16時50分03秒      all       0.01       0.00      0.10      0.01      0.00      99.89
17時00分05秒      all       0.01       0.00      0.10      0.01      0.00      99.88
17時10分06秒      all       0.01       0.00      0.09      0.00      0.00      99.89
17時20分08秒      all       0.01       0.00      0.08      0.01      0.00      99.90
17時30分09秒      all       0.01       0.00      0.08      0.00      0.00      99.91
17時40分01秒      all       0.01       0.00      0.08      0.01      0.00      99.90
17時50分02秒      all       0.01       0.00      0.08      0.00      0.00      99.90
18時00分04秒      all       0.05       0.00      0.13      0.01      0.00      99.81
18時10分05秒      all       0.04       0.01      0.11      0.01      0.00      99.83
平均値:       all       0.16       0.00      0.22      0.02      0.00      99.60
```

5.2.4 iostat コマンドによるシステムリソース監視

sysstat パッケージに含まれる iostat コマンドは、CPU の使用率や各種 I/O の利用状況を確認するためのコマンドです。I/O は、ハードディスクやテープドライブ、ネットワークマウントしたファイルシステム、端末入出力等の入出力性能を監視できます。

iostat コマンドを実行すると、システムが起動してから iostat コマンドを実行した時点までの間の CPU および I/O の状況が表示されます。

```
$ iostat
Linux 5.14.0-570.26.1.el9_6.x86_64 (vbox)   2025年07月27日      _x86_64_    (2
    CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.17    0.01    0.27    0.02    0.00   99.53

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read
    kB_wrtn    kB_dscd
dm-0              8.56        86.31        97.31         0.00    1176403
    1326269          0
dm-1              0.16         0.20         0.57         0.00       2712
    7752          0
dm-2              0.06         0.41         4.99         0.00       5550
    67953          0
sda               6.83        91.65        98.03         0.00    1249236
    1336119          0
sdb               0.19         3.93        10.18         0.00      53629
    138792          0
sr0               0.01         0.94         0.00         0.00      12758
             0          0
```

表示される内容は以下の表のとおりです。

項目	意味
%user	ユーザプロセスによる CPU の使用率
%nice	実行優先度（nice 値）を変更したユーザプロセスによる CPU の使用率
%system	システムプロセスによる CPU の使用率
%iowait	I/O 終了待ちとなった CPU の使用率
%steal	ハイパーバイザーによる他の仮想 CPU の実行待ちとなった CPU の使用率
%idle	CPU が何も処理をせずに待機していた時間の割合（ディスク I/O 以外）
tps	1 秒間の I/O 転送回数
kB_read/s	1 秒間のディスクの読み込み量 (ブロック数)
kB_wrtn/s	1 秒間のディスクの書き込み量 (ブロック数)
kB_dscd/s	1 秒間のディスクの廃棄された量 (ブロック数)
kB_read	ディスクの読み込み量 (ブロック数)
kB_wrtn	ディスクの書き込み量 (ブロック数)
kB_dscd	ディスクの廃棄された量 (ブロック数)

iostat コマンドに引数として数値を与えて実行すると、1 回目の表示はシステム起動から iostat コマンド実行時までの間の情報ですが、その後指定された秒間隔で全てのデバイスの I/O の利用状況が出力されます。終了するには Ctrl+C を入力します。

```
$ iostat 5
Linux 5.14.0-570.26.1.el9_6.x86_64 (vbox)   2025年07月27日      _x86_64_    (2
    CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.17    0.01    0.26    0.02    0.00   99.55

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read
    kB_wrtn    kB_dscd
dm-0              8.34        83.39        94.25         0.00    1177935
    1331277          0
dm-1              0.15         0.19         0.55         0.00       2712
    7752          0
dm-2              0.06         0.39         4.81         0.00       5550
    67953          0
sda               6.66        88.55        94.94         0.00    1250768
    1341127          0
sdb               0.18         3.80         9.83         0.00      53629
    138792          0
sr0               0.01         0.90         0.00         0.00      12758
             0          0


avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.00    0.00    0.20    0.00    0.00   99.80

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read
    kB_wrtn    kB_dscd
dm-0              0.00         0.00         0.00         0.00          0
             0          0
dm-1              0.00         0.00         0.00         0.00          0
             0          0
dm-2              0.00         0.00         0.00         0.00          0
             0          0
sda               0.00         0.00         0.00         0.00          0
             0          0
sdb               0.00         0.00         0.00         0.00          0
             0          0
```

```
sr0               0.00         0.00         0.00         0.00           0
             0           0


^C
```

iostat コマンドに-x オプションを付与して実行します。結果が拡張フォーマットで表示されます。

```
$ iostat -x
Linux 5.14.0-570.26.1.el9_6.x86_64 (vbox)   2025年07月27日      _x86_64_    (2
    CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.17    0.01    0.26    0.02    0.00   99.55

Device             r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s
    wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/s   drqm/s  %drqm
    d_await dareq-sz     f/s f_await  aqu-sz  %util
dm-0              2.66     83.05     0.00   0.00    0.31    31.23    5.65
    93.91     0.00   0.00    0.47    16.61    0.00       0.00     0.00   0.00
    0.00     0.00    0.00    0.00    0.00   1.36
dm-1              0.02      0.19     0.00   0.00    0.25    12.27    0.14
    0.55     0.00   0.00    4.15     4.00    0.00       0.00     0.00   0.00
    0.00     0.00    0.00    0.00    0.00   0.00
dm-2              0.02      0.39     0.00   0.00    0.12    19.82    0.04
    4.79     0.00   0.00    0.58   124.46    0.00       0.00     0.00   0.00
    0.00     0.00    0.00    0.00    0.00   0.00
sda               2.73     88.19     0.09   3.31    0.21    32.35    3.92
    94.61     1.90  32.69    0.39    24.14    0.00       0.00     0.00   0.00
    0.00     0.00    0.41    0.41    0.00   0.10
sdb               0.14      3.78     0.00   0.00    0.09    26.10    0.04
    9.79     0.07  65.54    0.66   272.68    0.00       0.00     0.00   0.00
    0.00     0.00    0.01    0.48    0.00   0.00
sr0               0.01      0.90     0.00   0.00    0.20    75.94    0.00
    0.00     0.00   0.00    0.00     0.00    0.00       0.00     0.00   0.00
    0.00     0.00    0.00    0.00    0.00   0.00
```

表示される内容は以下の表のとおりです。

項目	意味
r/s	1 秒間の読み込みリクエスト数
rkB/s	1 秒間の読み込みキロバイト（KB）数
rrqm/s	1 秒間デバイスへマージされた読み込みリクエスト数
%rrqm	1 秒間デバイスへマージされた読み込みリクエスト比率
r_await	デバイスへの読み込みリクエストの平均待ち時間
rareq-sz	デバイスへの読み込みリクエストの平均サイズ
w/s	1 秒間の書き込みリクエスト数
wkB/s	1 秒間の書き込みキロバイト（KB）数
wrqm/s	1 秒間デバイスへマージされた書き込みリクエスト数
%wrqm	1 秒間デバイスへマージされた書き込みリクエスト比率
w_await	デバイスへの書き込みリクエストの平均待ち時間
wareq-sz	デバイスへの書き込みリクエストの平均サイズ
d/s	1 秒間の廃棄リクエスト数
dkB/s	1 秒間の廃棄キロバイト（KB）数
drqm/s	1 秒間デバイスへマージされた廃棄リクエスト数
%drqm	1 秒間デバイスへマージされた廃棄リクエスト比率
d_await	デバイスへの廃棄リクエストの平均待ち時間
dareq-sz	デバイスへの廃棄リクエストの平均サイズ
f/s	1 秒間のフラッシュリクエスト数

項目	意味
f_await	デバイスへのフラッシュリクエストの平均待ち時間
aqu-sz	デバイスへのキューの平均サイズ
%util	デバイスへの IO リクエスト期間 CPU の使用率

5.2.5 廃棄（discard）の意味

iostat の表示にある廃棄（discard）とは、SSD の寿命を長くするため、ファイルが削除されて未使用になったブロックを廃棄し、再利用可能にする処理のことです。fstrim コマンド、あるいは fstrim サービスを実行することで行われます。

6 トラブルシューティング

6.1 トラブルシューティングの手法

サーバーに接続できないなどネットワークに起因する問題が発生した場合、以下のような手順で原因の調査を行います。

1. ログの確認
2. ping コマンドによる IP 通信の確認
3. 接続経路やポートの状況の確認
4. 通信内容の確認

6.2 ログ管理

システム障害の問題解決をはかるトラブルシューティングを行う場合に、もっとも有益な情報源がログです。

ログには、OS が出力するログ、アプリケーションが出力するログなど多くの種類が存在します。

ここでは、代表的なログの種類と確認方法、設定方法などを解説します。

6.2.1 ログの種類

AlmaLinux では、ログファイルは/var/log ディレクトリ以下に格納されています。

以下は代表的なログファイルです。

ファイル名	内容
messages	サービス起動時の出力など一般的なログ
secure	認証、セキュリティ関係のログ
maillog	メール関連のログ

また、systemd が記録する journald のログが存在します。こちらは別途解説します。

6.2.2 ログの確認

サーバーのログにサービス起動時、または動作時のエラーログが記録されていないかを確認します。また、クライアント側にもエラーログが記録されていないかを確認します。

- 一般的なトラブルであれば、まずは/var/log/messages を確認します。
- 認証関係やアクセス制限に関係するトラブルは/var/log/secure を確認します。
- メール関係であれば/var/log/maillog を確認します。
- Web サーバーであれば/var/log/httpd/error_log などを確認します。

6.2.3 dmesg に記録されるログ

dmesg コマンドは「display message」の略で、Linux カーネルがメッセージを出力するリングバッファ（循環バッファ）の内容を表示します。このリングバッファは一定のサイズ内で循環するようになっており、古いログは消えていきます。dmesg コマンドを用いることにより、システム起動時に出力されるカーネルメッセージの確認ができます。カーネルが正しくハードウェアを認識しているかどうかを確認する場合などに参照します。

```
$ dmesg
[    0.000000] Linux version 5.14.0-570.26.1.el9_6.x86_64
    (mockbuild@x64-builder03.almalinux.org) (gcc (GCC) 11.5.0 20240719 (Red Hat
    11.5.0-5), GNU ld version 2.35.2-63.el9) #1 SMP PREEMPT_DYNAMIC Wed Jul 16
    09:12:04 EDT 2025
[    0.000000] The list of certified hardware and cloud instances for Red Hat
    Enterprise Linux 9 can be viewed at the Red Hat Ecosystem Catalog,
    https://catalog.redhat.com.
```

```
[    0.000000] Command line:
    BOOT_IMAGE=(hd0,gpt2)/vmlinuz-5.14.0-570.26.1.el9_6.x86_64
    root=/dev/mapper/almalinux_vbox-root ro
    crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
    resume=/dev/mapper/almalinux_vbox-swap rd.lvm.lv=almalinux_vbox/root
    rd.lvm.lv=almalinux_vbox/swap rhgb quiet
[    0.000000] [Firmware Bug]: TSC doesn't count with P0 frequency!
[    0.000000] BIOS-provided physical RAM map:
（略）
```

6.2.4 syslog について

syslog は、カーネルやプログラムなどから出力されるログをまとめて記録する仕組みです。syslog を使うことで、各プログラムは独自にログを記録する仕組みを開発する必要が無くなります。また、syslog サーバーをネットワーク上で動作させることで、複数のホストからのログをまとめて記録することで、ログを一元管理することもできます。AlmaLinux では、syslog サーバーとして rsyslog が使用できます。

rsyslog は、従来の syslog デーモン（syslogd）に置き換わる、マルチスレッドの syslog デーモンです。rsyslog（Reliable syslog）という名前からも分かる通り、高い信頼性を実現するように開発されています。そのため、ログの転送に TCP を使用したり、データベースへのログ保存、暗号化したログの転送なども行うことができます。基本的な設定については、従来の syslogd と互換性があります。

6.2.5 ファシリティとプライオリティ

カーネルやプログラムが出力する syslog メッセージには、「ファシリティ」（facility）と「プライオリティ」（priority）と呼ばれる値が設定されています。

ファシリティは、何がそのログメッセージを生成したのかを指定します。たとえば、カーネルやメールといった値が指定されます。

また、プライオリティはメッセージの重要性を指定します。たとえば、単なる情報、非常に危険な状態などといった値が指定されます。

ファシリティには、以下の種類があります。

ファシリティ	意味
auth	セキュリティ・認証関連（login、su など）
authpriv	セキュリティ・認証関連（プライベート）
cron	cron や at のログ
daemon	一般的なデーモン（サーバープログラム）関連
kern	カーネル関連
lpr	プリンタ関連
mail	メール関連
news	NetNews 関連
security	auth と同じ
syslog	syslogd 自身のログ
user	ユーザアプリケーションのログ
uucp	uucp 転送を行うプログラムのログ
local0 から local7	独自のプログラムで利用可能な facility

プライオリティには、以下の種類があります。

プライオリティ	意味
debug	デバッグ用メッセージ
info	一般的な情報メッセージ
notice	通知メッセージ
warning	警告メッセージ
warn	warning と同じ
err	一般的なエラーメッセージ

プライオリティ	意味
error	err と同じ
crit	ハード障害などの危険なエラーメッセージ
alert	システム破損などの緊急事態
emerg	非常に危険な状態
panic	emerg と同じ
none	ファシリティを無効にする

6.2.6 syslog サーバーの設定

syslog サーバーの設定ファイルである/etc/rsyslog.conf には、受け取ったログメッセージをファシリティとプライオリティの組み合わせでどのファイルに出力するかの設定が記述されています。

記述は以下の形式となります。

```
ファシリティ.プライオリティ　アクション
```

syslog サーバーの設定ファイル中で、複数のファシリティを指定したい場合には、「,」（コンマ）で区切ります。たとえば、/var/log/messages には様々なファシリティからのログが記録されるように設定されています。この設定は、すべてのファシリティの info プライオリティ以上のログをすべて/var/log/messages に出力するようにしています。ただし、mail、authpriv、cron の 3 つのファシリティには none プライオリティが指定されているため、対象からは除外されています。

```
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
```

除外された各ファシリティの出力は、以下のように別途指定されています。

mail ファシリティのログは、メモリ上にある程度バッファリングした上でログファイルに書き込むように「-（ハイフン）」を指定しています。メールサーバーは一度に大量のログを書き込むことが多いからです。

```
authpriv.*                      /var/log/secure
mail.*                          -/var/log/maillog
cron.*                          /var/log/cron
```

6.2.7 プライオリティの動作

syslog 設定ファイル中でプライオリティを指定すると、そのプライオリティ以上の重要度のプライオリティがすべて当てはまります。たとえば、以下のように設定したとします。

```
mail.warning
```

mail ファシリティからの warning 以上（err、crit、alert、emerg）のすべてのプライオリティが当てはまります。

特定のプライオリティのみ指定したい場合には、「=プライオリティ」と指定します。

```
mail.=warning
```

この指定は mail ファシリティのプライオリティが warning のメッセージのみが当てはまります。

6.2.8 アクションの設定

ファシリティとプライオリティを記述した右側に、該当するログをどうするかを指定するアクションを記述します。

主なアクションは、以下の表のとおりです。

ファイル名　ログをファイルに書き込む。

-ファイル名　ログをファイルに書き込む際にバッファリングする。書き込み性能が向上するが、書き込まれていないデータがある時にシステム障害が発生するとログが失われる。

¥プログラム ログメッセージをプログラムに引き渡す。

***** すべてのユーザのコンソールにメッセージを表示する。今後、使用できなくなる可能性があります。

:omusrmsg:* *同様にすべてのユーザのコンソールにメッセージを表示する。今後はこちらを使用することが推奨されています。

[@ホスト名]（あるいは IP アドレス） UDP で syslog サーバーにログメッセージを送信する。

@[@ホスト名]（あるいは IP アドレス） TCP で syslog サーバーにログメッセージを送信する。

6.2.9 カーネルログの syslog 出力設定

カーネルのログを別途出力するように設定します。カーネルのログは、たとえば nftables によるパケットフィルタリングのようなカーネルの機能がログを出力します。

/etc/rsyslog.conf を編集し、ファシリティが kern、プライオリティが全てのメッセージを/var/log/kern.log に出力する設定を追加します。

```
$ sudo vi /etc/rsyslog.conf

# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console
kern.*                                                  /var/log/kern.log
```

rsyslog サービスを再起動して、新しい設定を読み込ませます。

```
$ sudo systemctl restart rsyslog
```

6.2.10 firewalld のログ設定変更

firewalld はカーネルの nftables を使用しているため、そのログはカーネルのログとして出力されます。

firewalld の設定を変更して、拒否した通信をすべてログ出力するように変更します。firewalld のログ出力設定を確認するには、firewall-cmd コマンドに--get-log-denied オプションをつけて実行します。

```
$ sudo firewall-cmd --get-log-denied
off
```

デフォルトでは off の状態ですが、--set-log-denied オプションで all に設定し、すべてのパケットのログを取得します。

```
$ sudo firewall-cmd --set-log-denied=all
success
$ sudo firewall-cmd --get-log-denied
all
```

6.2.11 firewalld のログ取得

ホスト OS 上で動作する Web ブラウザなど外部のクライアントから設定を行ったホストに対して、firewalld で許可されていないポート 443 番に Web ブラウザ等でアクセスしてみます。

firewalld の設定を確認します。

```
$ sudo firewall-cmd --list-service
cockpit dhcpv6-client http ssh
```

http のみ許可されており、https は許可されていません。

Web ブラウザに入力するアドレスは、プロトコルとして https で指定してポート番号 443 番にアクセスするようにします。

```
https://ホストのIPアドレス（192.168.56.101）
```

/var/log/kern.log にポート番号 443 番に対する通信を拒否した旨のログが出力されます。DPT が宛先のポートです。

```
$ sudo tail /var/log/kern.log
Jul 19 19:21:07 localhost kernel: filter_IN_public_REJECT: IN=enp0s8 OUT=
    MAC=08:00:27:40:b7:96:d0:11:e5:1a:ce:3b:08:00 SRC=192.168.56.1
    DST=192.168.56.101 LEN=64 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=TCP
    SPT=62291 DPT=443 WINDOW=65535 RES=0x00 CWR ECE SYN URGP=0
```

6.2.12 リモートホストのログを UDP で受け取る

syslog サーバーとしてリモートホストのログを受け取るための設定を行います。syslog のメッセージの送受信は、通常 UDP で行われます。

設定ファイル/etc/rsyslog.conf 内にある以下の 2 行から、行頭のコメントアウトを削除して設定を有効にします。

module(load= “imudp”) は、UDP 用のプロトコルモジュールのロードを設定しています。input(type= “imudp” port= “514”) は、UDP でログメッセージを受け取るポート番号を 514 番として指定しています。

```
$ sudo vi /etc/rsyslog.conf
```

```
（略）
# Provides UDP syslog reception
module(load="imudp") # needs to be done just once
input(type="imudp" port="514")
```

rsyslog サービスを再起動します。rsyslogd が UDP のポート番号 514 番で待ち受けるようになります。

```
$ sudo systemctl restart rsyslog
$ ss -uln | grep 514
UNCONN 0      0                                    0.0.0.0:514        0.0.0.0:*
UNCONN 0      0                                       [::]:514           [::]:*
```

設定後、firewalld の設定を変更し、外部からの UDP のポート番号 514 番へのパケットを許可するように設定を変更する必要があります。設定については後述します。

6.2.13 リモートホストのログを TCP で受け取る

ログメッセージの送受信に TCP を使用することにより、UDP で発生していたログの取りこぼしを防ぐことができます。UDP はセッションレスなプロトコルのため、送受信に失敗した時に再送信する仕組みが無いためです。

ただし、TCP はプロトコルの性質上 UDP よりも処理が重くなってしまうため、大量のログが送信されてくる環境では逆にボトルネックになってしまい、syslog サーバー側が高負荷で処理が滞ってしまう可能性があります。

そのため、TCP を使ったログメッセージの送受信は、ログの量がそれほど多くなくログ記録の信頼性が必要な場合に設定します。もし、大量のログが送信されてくる場合には、syslog サーバーを複数用意するか、UDP を使う必要があります。

設定ファイル/etc/rsyslog.conf 内にある以下の 2 行から、行頭のコメントアウトを削除して設定を有効にします。

module(load= “imtcp”) は、TCP 用のプロトコルモジュールのロードを設定しています。input(type= “imtcp” port= “514”) は、TCP でログメッセージを受け取るポート番号を 514 番として指定しています。

```
$ sudo vi /etc/rsyslog.conf
```

```
（略）
# Provides TCP syslog reception
module(load="imtcp") # needs to be done just once
input(type="imtcp" port="514")
```

rsyslog サービスを再起動します。rsyslogd が TCP のポート番号 514 番で待ち受けるようになります。

```
$ sudo systemctl restart rsyslog
$ ss -tln | grep 514
LISTEN 0      25           0.0.0.0:514       0.0.0.0:*
LISTEN 0      25              [::]:514          [::]:*
```

設定後、firewalld の設定を変更し、外部からの TCP のポート番号 514 番へのパケットを許可するように設定を変更する必要があります。

6.2.14 syslog サーバーのための firewalld の設定

firewalld の設定を変更し、TCP および UDP のポート番号 514 番の接続を許可しておきます。

```
sudo firewall-cmd --add-port=514/udp --permanent
sudo firewall-cmd --add-port=514/tcp --permanent
sudo firewall-cmd --reload
```

6.2.15 syslog クライアントの設定

ネットワークで接続された syslog サーバーに対してログメッセージを送信する syslog クライアントを設定します。

syslog クライアント側のホストでも rsyslog を設定し、アクションの設定でネットワーク上の syslog サーバーを指定します。

syslog クライアントの設定ファイル/etc/rsyslog.conf を修正します。

authpriv ファシリティに関するすべてのログを syslog サーバーに送信するように設定を追加します。[@送信先と指定することで UDP を使用した送信を指定できます]。

また、mail ファシリティに関するすべてのログを syslog サーバーに送信するように設定を追加します。@[@送信先と指定することで TCP を使用した送信を指定できます]。

```
$ sudo vi /etc/rsyslog.conf
```

```
# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure
authpriv.*                                              @192.168.56.101

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog
mail.*                                                  @@192.168.56.101
```

syslog クライアントの rsyslog サービスを再起動します。

```
$ sudo systemctl restart rsyslog
```

UDP でログを送信 syslog クライアントで logger コマンドを実行して、authpriv.debug プライオリティでログを出力します。

```
[linuc@client ~]$ logger -p authpriv.debug "This is auth log over UDP"
```

syslog サーバー上の/var/log/secure にログが出力されることを確認します。

```
[linuc@server ~]$ sudo tail -f /var/log/secure
（略）
Jul 27 15:16:44 vbox linuc[3207]: This is auth log over UDP
```

TCP でログを送信 syslog クライアントで logger コマンドを実行して、mail.debug プライオリティでログを出力します。

```
[linuc@client ~]$ logger -p mail.debug "This is mail log over TCP"
```

syslog サーバー上の/var/log/maillog にログが出力されることを確認します。

```
[linuc@server ~]$ sudo tail /var/log/maillog
Jul 27 15:17:48 vbox linuc[3209]: This is mail log over TCP
```

6.2.16 logrotate によるログローテーション

ログファイルは常に追記されていくため、ファイルサイズが次第に肥大化してディスク容量を圧迫し、後でログを確認する際に必要なログを見つけにくくなります。これらの問題を回避するため、ログを一定期間でローテーションする logrotate が使われています。

logrotate は、systemd timer から 1 日 1 回起動されます。/etc/logrotate.conf が logrotate の設定ファイルとなっており、ログファイルをローテーションするタイミングや、ログファイルを何世代まで残すかなどの設定が記述されています。サービス毎の詳細な設定は、/etc/logrotate.d ディレクトリに格納されています。

logrotate の設定で使用できるディレクティブは以下のとおりです。

create [モード] [所有ユーザ] [所有グループ] ローテーションを行った後、代わりに空の新規ログファイルを作成します。属性も指定できます。モードは 0755 のような数値書式。指定しない属性については元のファイルの属性が引き継がれます。

nocreate create をグローバルに設定した場合に、個別に create を無効にしたい際に使用します。

copy/nocopy 元のログファイルはそのままにして、コピーを保存します。

copytruncate/nocopytruncate copy の動作を行った後、元のログファイルの内容を消去します。見かけ的には create と同じ結果となります。これはログファイルをリロードする方法が無いプログラムへの対処法のひとつです。たとえば Oracle 10g R1/R2 の alert ログに対しては、この方法を行わないと以前のログファイル（例えば alert_xx.log.1）にログが書き込み続けられます。

rotate 世代数 世代ローテーションの世代数を指定します。たとえば元のログファイルが a.log の場合、num に 2 を指定すると、a.log → a.log.1 → a.log.2 →廃棄となります。0 の場合、a.log →廃棄となります。

start 数値 最初のローテーションファイルの末尾に付加する値を指定します。デフォルトは 1 です。たとえば num に 5 を指定すると、a.log → a.log.5 → a.log.6 となります。

extension 拡張子 ローテーションした旧ログファイルに付ける拡張子を指定します。指定には区切りのドットも必要です。たとえば拡張子に「.bak」と指定すると、some.log の初代ローテーションログは some.log.1.bak となります。圧縮も行う場合、圧縮による拡張子はさらにその後ろに付きます。

compress/nocompress ローテーションした後の旧ファイルに圧縮を掛けます。デフォルトは nocompress（非圧縮）です。

compresscmd コマンド ログファイルの圧縮に使用するプログラムを指定します。デフォルトは gzip です。

uncompresscmd コマンド ログファイルの解凍に使用するプログラムを指定します。デフォルトは gunzip です。

compressoptions オプション 圧縮プログラムへ渡すオプションを指定します。デフォルトは gzip に渡す「-9」（圧縮率最大）です。「-9 -s」のようにスペース入りで複数のオプションを指定することはできません。

compressext 拡張子 圧縮後のファイルに付ける拡張子（ドットも必要）を指定します。デフォルトでは、使用する圧縮コマンドに応じたものが付けられます。

delaycompress/nodelaycompress 圧縮処理を次のローテーションまで遅らせる、あるいは遅らせません。

olddir ディレクトリ/noolddir ローテーションした旧ログをディレクトリに移動します。移動先は元と同じデバイス上で指定します。元のログに対する相対指定も有効です。

mail address/nomail 旧ログファイルを address に送信します。どの段階のログを送るかは maillast などのオプションで決まります。

maillast 世代が終わって破棄されるログをメールします。

mailfirst 初代ローテーションログをメールします。

daily/weekly/monthly ログローテーションを日毎/週毎/月毎に行います。デフォルトは daily。たとえば weekly なら、毎日実行したとしても、週に 1 回だけローテーションが行われます。

size サイズ [K/M] ログのサイズがサイズバイトを超えていればローテーションを行います。この条件は daily,weekly などの条件より優先されます。キロバイト（K）、メガバイト（M）での指定もできます。

ifempty/notifempty 元のログファイルが空でもローテーションを行う、あるいは行いません。

missingok/nomissingok 指定のログファイルが存在しなかったとしてもエラーを出さずに処理を続行する、あるいはエラーを出力します。

firstaction ローテーションを行う前にスクリプトを実行します。prerotate よりも前に実行される個別定義内でのみ指定可能です。

prerotate ローテーションを行う前にスクリプトを実行します。firstaction の後に実行されます。個別定義内でのみ指定できます。

postrotate ローテーションが行われた後にスクリプトを実行します。lastaction より前に実行されます。個別定義内でのみ指定できます。

lastaction ローテーションが行われた後（よりも後）にスクリプトを実行します。postrotate の後に実行されます。個別定義内でのみ指定できます。

sharedscripts ローテーションするログが複数あった場合に、prerotate、postrotate のスクリプトを一度だけ実行します。

nosharedscripts ローテーションするログが複数あった場合に、prerotate、postrotate のスクリプトを各ログファイル毎に実行します。

include ファイル（ディレクトリ） include の記述のある位置に別の設定ファイルを読み込みます。ディレクトリを指定した場合、そのディレクトリ内から、ディレクトリおよび名前付きパイプ以外の通常ファイルがアルファベット順に読み込まれます。

tabooext [+] 拡張子 [, 拡張子,…] include でディレクトリを指定した場合に読み込むファイルから除外するファイルの拡張子を指定します。デフォルトで「.rpmorig」「.rpmsave」「,v」「.swp」「.rpmnew」「~」「.cfsaved」「.rhn-cfg-tmp-*」が指定されています。+を指定するとデフォルト指定に対して追加で拡張子を指定できます。+を指定しないとデフォルト指定を破棄して新規に拡張子を指定します。

ここに紹介した以外のディレクティブも多数あるので、マニュアル等を参照してみてください。

6.2.17 ログローテート設定ファイルの確認

/etc/logrotate.d/httpd を参考に、ローテートの設定を確認します。

```
$ cat /etc/logrotate.d/httpd
# Note that logs are not compressed unless "compress" is configured,
# which can be done either here or globally in /etc/logrotate.conf.
/var/log/httpd/*log {
    missingok
    notifempty
    sharedscripts
    delaycompress
    postrotate
        /bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
    endscript
}
```

この例では、以下の通りログローテーションの処理が行われます。

対象となるログファイルは/var/log/httpd ディレクトリ内の、ファイル名が log で終わるすべてのログファイルです。デフォルトでは access_log、error_log というファイル名のログファイルが作成されています。

- 1 行目の missingok でログファイルが実在しなかったとしてもエラーを出さずに処理を続行します。
- 2 行目の notifempty で元のログファイルが空ならばローテーションしません。
- 3 行目の sharedscripts で prerotate,postrotate のスクリプトを一度だけ実行します。
- 4 行目の delaycompress で圧縮処理を次のローテーションまで遅らせます。
- 5 行目の”postrotate”から”endscript”までが、ローテーションが行われた後に実行されるスクリプトです。systemctl コマンドを実行して httpd サービスを reload することで、新しいログファイルが生成されます。

6.3 journald のログの確認

journald のログを確認するには、journalctl コマンドを実行します。オプションを付与しないで実行すると、すべてのログが表示されます。

以下の例では、Linux カーネル起動時のログが記録されているのが分かります。

```
$ journalctl
 7月 19 21:48:50 localhost kernel: Linux version 5.14.0-570.12.1.el9_6.x86_64
     (mockbuild@x64-builder02.almalinux.org) (gcc (GCC) 11.5.0 >
 7月 19 21:48:50 localhost kernel: The list of certified hardware and cloud
     instances for Red Hat Enterprise Linux 9 can be viewed at th>
 7月 19 21:48:50 localhost kernel: Command line:
     BOOT_IMAGE=(hd0,gpt2)/vmlinuz-5.14.0-570.12.1.el9_6.x86_64
     root=/dev/mapper/almalinux_v>
 7月 19 21:48:50 localhost kernel: [Firmware Bug]: TSC doesn't count with P0
     frequency!
 7月 19 21:48:50 localhost kernel: BIOS-provided physical RAM map:
 （略）
```

特定のサービスのログに絞るには、-u オプションを付与して実行します。

以下の例では、httpd サービス起動時のログが確認できます。

```
# journalctl -u httpd
 7月 26 18:40:34 vbox systemd[1]: Starting The Apache HTTP Server...
```

```
7月 26 18:40:34 vbox httpd[2412]: AH00558: httpd: Could not reliably determine
    the server's fully qualified domain name, using fe80::a0>
7月 26 18:40:34 vbox systemd[1]: Started The Apache HTTP Server.
7月 26 18:40:34 vbox httpd[2412]: Server configured, listening on: port 80
（略）
```

6.3.1 journald のログの保存

journald のログは、再起動すると消えてしまう設定がデフォルトとなっています。journald の設定ファイル /etc/systemd/journald.conf の Storage 設定の値がデフォルトでは auto に設定されています。この設定は、以下のように動作します。

1. /var/log/journal ディレクトリが存在すれば書き込む
2. /var/log/journal ディレクトリが存在しないか、書き込めない場合には、/run/log/journal ディレクトリに書き込む

デフォルトでは/var/log/journal ディレクトリは存在しないため、/run/log/journal ディレクトリにログが書き込まれます。/run/log/journal ディレクトリは tmpfs でメモリ上に作られた一時領域なので、システム再起動時にログのファイルは消えてしまいます。

journald のログをシステム再起動時に消えないようにするには、以下のように/var/log/journal ディレクトリを作成して、システムを再起動します。

```
$ sudo mkdir /var/log/journal
$ sudo chmod 700 /var/log/journal
$ sudo reboot
```

ログファイルが作成されたことを確認します

```
$ ls -l /var/log/journal
合計 0
drwxr-sr-x+ 2 root systemd-journal 53  7月 27 15:24
    65dd8a0b080e4373a5633404cabaac84
$ ls -l /var/log/journal/65dd8a0b080e4373a5633404cabaac84
合計 16388
-rw-r-----+ 1 root systemd-journal 8388608  7月 27 15:24 system.journal
-rw-r-----+ 1 root systemd-journal 8388608  7月 27 15:24 user-1000.journal
```

6.4 ping コマンドによる IP 通信の確認

ping コマンドを使って、サーバーに対する通信が行えるかどうかを確認します。ping コマンドは ICMP を使った通信で IP 通信が可能か確認できます。サーバーに対する ping に応答が無い場合、以下のような問題が考えられます。

6.4.1 ping コマンドに応答しないサーバー自身の問題

IP アドレスやデフォルトゲートウェイが適切に設定されていなかったり、firewalld などのパケットフィルタリングで ICMP を通さない設定になっていることが考えられます。サーバーのネットワーク設定を再度確認してみます。また、サーバー側から他のホストへ ping コマンドを実行して、応答があるか確認してみます。

6.4.2 ネットワーク経路の問題

ネットワーク通信経路上にあるケーブルやスイッチ、ルーター、ファイアーウォールやロードバランサーなどのネットワーク機器に問題が無いかを確認します。ルーティングに問題があるかを確認するためには traceroute コマンドを使用しますが、traceroute コマンドは ICMP を使用しているため、途中のルーターで ICMP を通さない場合、すべての経路が確認できないことがあります。

6.5 サーバーに接続できない問題の解決

ping コマンドによる応答があるが、サーバーに接続できない場合には、ネットワーク経路上やサーバー自身のパケットフィルタリング、サーバー側でのポートバインディングの問題などが考えられます。

6.5.1 サーバーに接続できないネットワーク経路の問題

firewalld やネットワーク経路上のファイアーウォールなどで、指定されたポートへの通信が許可されていない。firewalld やファイアーウォールのポート許可設定を確認します。

6.5.2 サーバーに接続できないサーバー自身の問題

サービスが停止しており、指定されたポートを Listen していない。あるいは、ローカルループバックアドレス（127.0.0.1）のみ Listen しており、接続先に指定した IP アドレスにポートがバインドされていない。ss コマンドなどを使用して、ポートの状態を確認します。

6.5.3 ss コマンドでのポートの状況の確認

ss コマンドを使って、サービスプロセスとポート番号、さらに IP アドレスとのバインドの状況が確認できます。

ss コマンドに-p オプションを指定して実行します。

```
$ sudo ss -tlp | grep ssh
LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
    users:(("sshd",pid=968,fd=3))
LISTEN 0      128             [::]:22           [::]:*
    users:(("sshd",pid=968,fd=4))
```

この結果から、以下のことが分かります。

- sshd のプロセス ID が 968 であること
- TCP ポート番号 22 番で LISTEN していること
- ポート番号 22 番がサーバーのすべての IP アドレス（0.0.0.0:22/[::]:22）にバインドされていること
- 送信元制限を行っていないこと（0.0.0.0:/[::]:）

6.6 Wireshark を使ったパケットキャプチャによる通信内容の確認

サーバーとの接続が行えており、ログにも手がかりとなるエラーが無いが、サービスが正しく動作しないような場合には、通信パケットをキャプチャして、通信内容を確認します。パケットをキャプチャすることで、サーバーとクライアントの間でどのような通信が行われているかを確認できます。パケットキャプチャのツールとしては、GUI で操作できる Wireshark などがあります。

GUI を持つパケットキャプチャリングソフトである Wireshark を使えば、パケットキャプチャリングを行ったパケットの中身を見たり、フィルタリング機能で必要なパケットのみに絞り込んでパケットを確認することができます。

6.6.1 Wireshark のインストール

Wireshark をインストールします。

```
$ sudo dnf install wireshark
```

6.6.2 Wireshark を起動

AlmaLinux に GUI でログインし、端末から wireshark コマンドを実行します。パケットキャプチャには root 権限が必要なので、sudo コマンドで実行します。

```
$ sudo wireshark
```

6.6.3　キャプチャを行うデバイスの選択

インターフェースの一覧からパケットキャプチャを行いたいインターフェースをクリックして選択します。複数同時に選択したい場合には Ctrl キーを押しながら複数選択します。

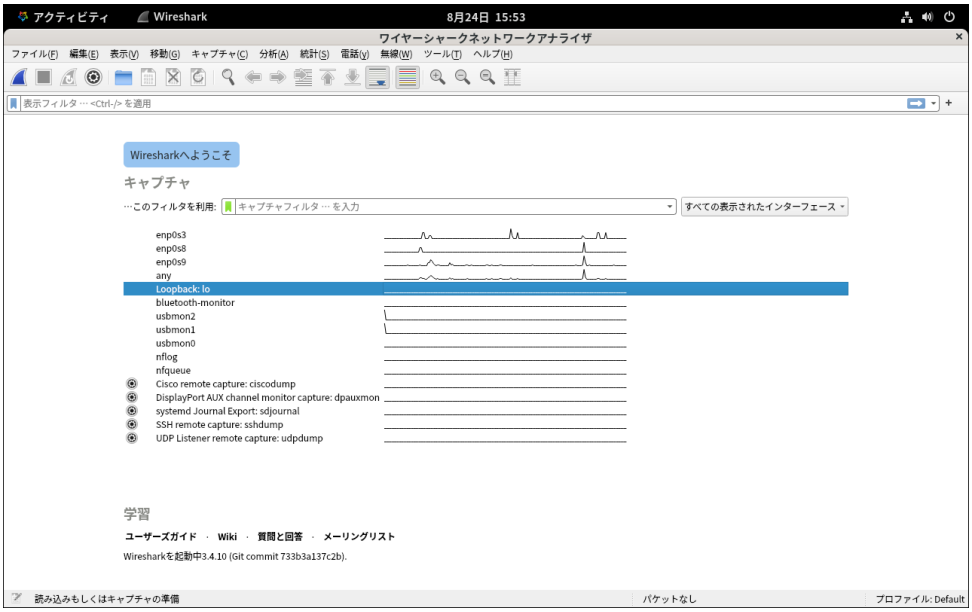

図 15: キャプチャするインターフェースの選択

Web サーバーが動作している Linux 上の Web ブラウザからローカル（localhost）にアクセスする場合、使用するインターフェースは「Loopback:lo」になる点に注意してください。

6.6.4　パケットキャプチャの開始

選択したインターフェースを右クリックして「キャプチャ開始」を選択します。インターフェースをダブルクリックでもキャプチャを開始できます。

6.6.5　Web サーバーにアクセス

サーバーと通信を行ってパケットキャプチャを行います。クライアントで Web ブラウザを起動し、サーバーの Web サーバーにアクセスします。

6.6.6　パケットキャプチャを停止

「キャプチャ」メニュー→「停止」を選択し、パケットキャプチャを停止します。左上の赤い四角い停止ボタンでも停止できます。

6.6.7　結果の絞り込み

「Filter:」のテキストボックスに「http」と入力して、Enter キーを押して絞り込みます。

参照したいパケットを選択し、ウインドウ真ん中の詳細情報で「Hypertext Transfer Protocol」をダブルクリックして、HTTP 通信の内容を確認します。

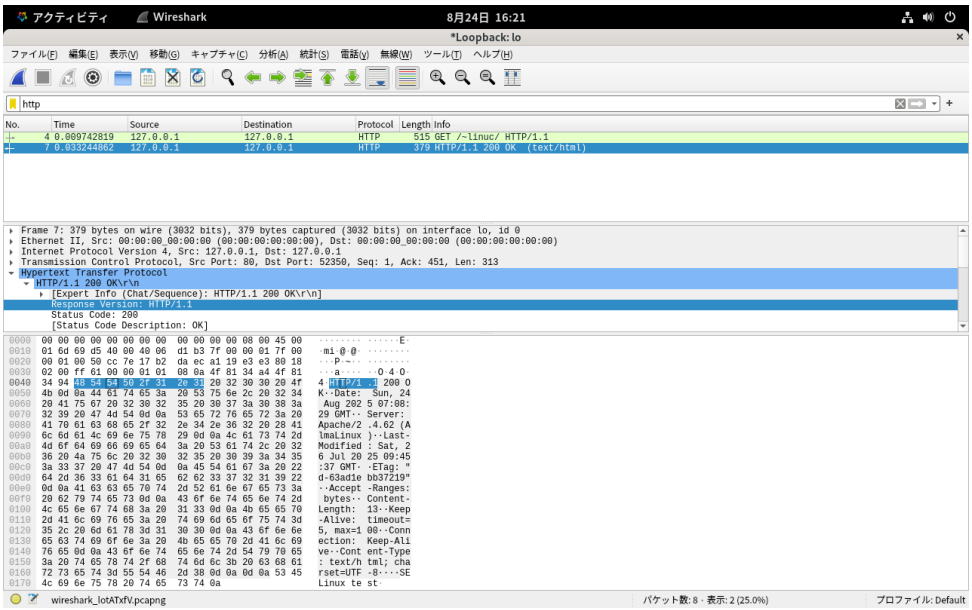

図 16: http で絞り込みを行います

Linuxシステム管理標準教科書

2025年9月16日　Ver.2.0.0

発行元　LPI-Japan