Linux
Professional
Institute

LPIC-1

バージョン 5.0
日本語

102

Table of Contents

Linux Professional Institute

課題 105: シェル、スクリプト

Linux Professional Institute

105.1 シェル環境をカスタマイズして使用する

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 105.1

総重量

4

主な知識分野

- ログイン時や新しいシェルの作成時に環境変数(PATHなど)を設定する。
- 頻繁に使用されるコマンドシーケンスのためのBash関数の作成。
- 新しいユーザーアカウントのスケルトンディレクトリを維持する。
- 適切なディレクトリでコマンド検索パスを設定する。

用語とユーティリティ

- `.`
- `source`
- `/etc/bash.bashrc`
- `/etc/profile`
- `env`
- `export`
- `set`
- `unset`
- `~/.bash_profile`
- `~/.bash_login`
- `~/.profile`
- `~/.bashrc`
- `~/.bash_logout`
- `function`
- `alias`

105.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	105 シェルとシェルスクリプト
Objective:	105.1 シェル環境をカスタマイズして使用する
Lesson:	1 of 3

はじめに

シェルは間違いなくLinuxシステムで最も強力なツールであり、ユーザーとカーネルとの間のインターフェースとして、ユーザーが入力したコマンドを解釈します。つまり、すべてのシステム管理者は、シェルの使い方に習熟していなければなりません。ご存じのように、大部分のLinuxディストリビューションでは、Bourne Again Shell（Bash）がシェルとして使われています。

Bash（やその他のシェル）が起動するとまず、一連の起動スクリプトを実行します。起動スクリプトは、セッションの環境をカスタマイズします。システム全体のスクリプトと、ユーザー毎に固有のスクリプトがあります。これらのスクリプトでは、変数、エイリアス、関数などを用いて、ユーザーのニーズや好みに応じた設定を行えます。

シェルの種類によって、読み込まれる起動ファイルが異なります。いくつかのパターンを見てみましょう。

シェルの種類: 対話型 vs 非対話型　と　ログイン vs 非ログイン

まず始めに、シェルおける 対話型 と ログインシェル の概念を明確にします。

対話型シェルと非対話型シェル

ユーザーとシェルの間で、対話が行われるか否かを意味します:　　対話型シェルでは、シェルがプロンプトを表示し、ユーザーがキーボードからコマンドを入力し、シェルは画面にメッセージを出力します。

ログインシェルと非ログインシェル

ユーザーがアカウント名やパスワードを入力してコンピューターシステムにログインした直後に実行

されるシェルを、ログインシェル と呼びます。それ以外のシェルは、非ログインシェル です。

ログインシェルと非ログインシェルのいずれにも、対話型と非対話型があり、その組み合わせによって役割が異なります。

ユーザーがシステムにログインしたときに実行される　対話型のログインシェル　は、ユーザーの必要に応じて設定（環境）をカスタマイズします。例えば、ある部署に属するユーザーが利用するアプリケーション用に、特別な変数を定義するといった処理を行います。

対話型の非ログインシェル　　は、ログイン後にユーザーが実行するシェルです。一時的に別のユーザーになったり、ログインシェルとは異なるシェルを利用するために明示的にサブ（子）シェルを起動することがあります。このタイプのシェルでは、ログインシェルと同様に、ファイルのコピーやスクリプトの作成といった作業を行うことができますし、独自の変数を定義することもできます。

非対話型のシェル　　は、ユーザーとのやりとりを行いません。自動実行されるシェルスクリプトが、このタイプのシェルを使用します。つまりこのタイプのシェルは、プロンプトを表示して入力を求めることはありませんし、出力が必要な場合はファイルに書き込みます。

非対話型のログインシェル　　はほとんど使われません。このタイプのシェルを使うことはまずありませんが、シェルの挙動を説明するために触れておきます。例えば、`/bin/bash --login <スクリプト>` を実行すると、ログインシェルと同じ処理を行ってからスクリプトを実行しますし、あるいは、sshコマンドの標準入力（stdin）にスクリプトをパイプすれば、そのスクリプトが（リモートマシンの）ログインシェルで実行されます。

```
cat script.sh | ssh <ユーザー名>@<サーバー名>
```

前述のように、自動化されたスクリプト（主に定期的に繰り返し実行する保守管理タスクを実行するために、cronから起動される）では、非対話型の非ログインシェル　　が使われます。このタイプのシェルでは、ログイン時に読み込まれる起動ファイルの一部が読み取られません。

ターミナルを開く

デスクトップ環境では、ターミナル（エミュレーター）アプリケーションを開くか、システムコンソールの1つに切り替えることで、シェルにアクセスします。GUIからアクセスした場合には、端末装置（`tty`）ではなく、端末エミュレータ（`pty`（仮想端末））を扱うことになります。GUIセッションで使用する　gnome-terminal　や　konsole　などの端末エミュレーターは、テキストベースの端末装置よりも機能が豊富でユーザーフレンドリーですが、XTerm　や　sakura　といったシンプルなターミナルエミュレーターも存在します。

キーボードから `Ctrl`+`Alt`+`F1` から `Ctrl`+`Alt`+`F6` を使用することで、テキストベースで対話型のログインシェルを開くコンソールログインに移動できます。`Ctrl`+`Alt`+`F7` で、デスクトップセッションに戻ります。（訳注：PCアーキテクチャの場合のみですし、ファンクションキーの割り当てはディストリビューションによって異なります）。

NOTE

`tty`　　はテレタイプライター（訳注:テキストベースの端末装置。GUIが登場する以前の1990年代によく使用されていた）を、`pts`　　は仮想端末インターフェイスを表します。詳細については、`man tty` および `man pts` を参照してください。

bash でシェルを起動する

ログイン後、ターミナルに bash と入力して新しいシェルを開きます。技術的には、このシェルはログインシェルの子プロセスになります。

子プロセスとして bash を起動するときに、さまざまなオプションで起動するシェルのタイプを指定できます。bash 起動時のいくつかのオプションを示します。

bash -l または bash --login
ログインシェルを起動します。

bash -i
対話型シェルを起動します。

bash --noprofile
ログインシェルを起動する際に読み込まれるシステム全体の起動ファイル /etc/profile と、ユーザーレベルの起動ファイル ~/.bash_profile, ~/.bash_login, ~/.profile をすべて無視します。

bash --norc
対話型シェルを起動する際に読み込まれるシステム全体の起動ファイル /etc/bash.bashrc ファイルと、ユーザーレベルの起動ファイル ~/.bashrc の両方を無視します。

bash --rcfile <file>
対話型シェルを起動する際に、システム全体の起動ファイル /etc/bash.bashrc やユーザーレベルの起動ファイル ~/.bashrc を無視して、代わりに <file> を起動ファイルとして実行します。

次に、さまざまな起動ファイルについて説明します。

su と sudo によるシェルの起動

これら2つの似通ったコマンドを使用することで、任意のタイプのシェルを実行できます。

su
ユーザーIDを変更するか、スーパーユーザー（root）になります。このコマンドでは、ログインシェルと非ログインシェルのどちらも呼び出すことができます。

- su - user2、su -l user2、su --login user2 はいずれも、user2 として対話型のログインシェルを起動します。
- su user2 は、user2 として対話型の非ログインシェルを起動します。
- su - root または su - は、root として対話型のログインシェルを起動します。
- su root または su は、root として対話型の非ログインシェルを起動します。

なお、su コマンドを実行すると、変更後のユーザー（例では user2 ないし root）のパスワードが尋ねられます。

sudo
別のユーザー（スーパーユーザーを含む）としてコマンドを実行します。sudoコマンドは主に、一時的にroot権限を持ってコマンドを実行するために使用し、このコマンドを使用するユーザーは sudoers ファイルに記載されている必要があります。ディストリビューションによって異なります

が、Debian系ディストリビューションではグループ `sudo` が、Redhat系ディストリビューションではグループ `wheel` が、デフォルトで `sudoers` ファイルに記載されています。sudoコマンドを実行するユーザーを、それらのグループに追加すればsudoコマンドを使用できるようになります。`user2` を `sudo` グループに所属させるには、次のようにします。（訳注: インストール時に作成した「最初のユーザー」は、sudoコマンドを実行できるグループに所属しているのが一般的です。）

```
root@debian:~# usermod -aG sudo user2
```

`su` と同様に `sudo` でも、ログインシェルと非ログインシェルのどちらかを呼び出すことができます。

- `sudo su - user2`、`sudo su -l user2`、`sudo su --login user2` はいずれも、 `user2` として対話的なログインシェルを起動します。※注
- `sudo su user2` は、`user2` として対話的な非ログインシェルを起動します。※注
- `sudo -u user2 -s` は、`user2` として対話的な非ログインシェルを起動します。
- `sudo su - root`、`sudo su -` は、`root` として対話的なログインシェルを起動します。※注
- `sudo -i` は、 `root` として対話的なログインシェルを起動します。
- `sudo -i <some_command>` は、`root` として対話的なログインシェルを起動してコマンドを実行し、終了すると今のシェルに戻ります。
- `sudo su root`、`sudo su` は、`root` として対話的な非ログインシェルを起動します。※注
- `sudo -s`、`sudo -u root -s` は、`root` として対話的な非ログインシェルを起動します。

`su` や `sudo` を使用する場合、どのタイプのシェルを起動するのかを考慮します。切替先ユーザーの作業環境に切り替える必要があるときは、ログインシェルを呼び出します。オプションを指定しなければ、非ログインシェルを呼び出します。

※**訳注**: `su` は Switch User に由来し、切替先のユーザーのシェルを起動するコマンドで、特定のコマンドを実行する際は `-c` オプションを指定します。対して `sudo` は Switch User DO に由来し、指定したコマンドだけを実行するコマンドで、切替先のユーザーとしてシェルを実行するには `-i` ないし `-s` オプションを指定します。また、`su` コマンドを実行すると変更後のユーザー（例では user2 ないし root）のパスワードが尋ねられますが、`sudo` コマンドでは自分（現在のユーザー）のパスワードが尋ねられます。`su` コマンドの利用はパスワードの共有に繋がるため、セキュリティの面から推奨されていません。

実行中のシェルはどのタイプ？

作業中のシェルのタイプを確認するためには、`echo $0` コマンドを実行します。

対話型のログインシェル

`-bash`

対話型の非ログインシェル

`bash` または `/bin/bash`

非対話型の非ログインシェル（スクリプト）

`<スクリプト名>`

いくつのシェルがあるか？

システムで稼働中の `bash` シェルの数を確認するには、コマンド `ps aux | grep bash` を使用します。

```
user2@debian:~$ ps aux | grep bash
user2       5270  0.1  0.1  25532  5664 pts/0    Ss   23:03   0:00 bash
user2       5411  0.3  0.1  25608  5268 tty1     S+   23:03   0:00 -bash
user2       5452  0.0  0.0  16760   940 pts/0    S+   23:04   0:00 grep --color=auto bash
```

`ps` コマンドの `TTY` と `COMMAND` 欄などから、実行中のシェルのタイプが分かります。 1行目では `TTY` が `pts`（仮想端末）であることから、`user2` はGUIセッションにログインしてターミナルを開いたものと推測されます（GUIのターミナル内で実行されるシェルは、対話型の非ログインシェルになります）。 2行目では `TTY` が `tty1` であることから、`user2` は`Ctrl`+`Alt`+`F1`を入力して、1番目の仮想コンソールからシェルセッションに入ったのでしょう（コンソールからのログインでは、対話型のログインシェルが起動します）。 3行目の出力は、`grep` コマンドの引数に `bash` が含まれているために表示されたものです。`TTY` 欄が1行目の `bash` と同じ `pts/0` ですから、`user2` は`Ctrl`+`Alt`+`F7`を入力してGUIセッションに戻り、ターミナルでコマンド `ps aux | grep bash` を実行したものと推測されます。GUI環境のターミナルエミュレーターでは、ログインシェルではなく非ログインシェルが起動されることに注意してください。ただし、GUI環境が起動するときにログイン時に必要な起動ファイルを読み込んでいますから、シェル環境としては同じです。

シェル環境を整える 起動ファイル

Linuxシステムにおけるシェルの種類を理解したので、それぞれのタイプのシェルが実行する起動ファイル（スタートアップファイル）を見ていきましょう。システム全体で有効となるスクリプトは `/etc/` ディレクトリに置かれて、ユーザー毎に固有のスクリプトはユーザーのホーム（`~`）ディレクトリに置かれます。ユーザー固有の起動ファイルは、所定の順序で検索して見つかった最初のもののみが読み込まれて、残りのファイルには進みません。お気に入りのテキストエディタか、`less` を使って、以下に示す起動ファイルを自分で調べてください。

NOTE 起動ファイルには、Bashのみが読み込むものと、大部分のシェルが利用する一般的なものがあります。

対話型ログインシェルの場合

グローバル（システム全体）レベル

`/etc/profile`

Bourneシェル（`sh`）とその互換シェル（`bash` など）が利用する、システム全体の（すべてのユーザーが対象の）起動ファイルです。このファイルでは、`if` 文で要件を確認しながら、`PATH` や `PS1` などいくつかの変数を設定します。ディストリビューションによって異なりますが、`/etc/profile` スクリプトは、追加の起動ファイル（`/etc/bashrc` や `/etc/profile.d` 内のファイル）があれば、それらを追加で読み込んで実行するものが多いようです。

`/etc/profile.d/*`

このディレクトリが存在するディストリビューションでは、`/etc/profile` がこのディレクトリにある起動ファイルを追加で読み込みます。

ユーザーレベル

シェルはグローバルレベルの起動ファイルを処理した後に、ユーザーレベルの起動ファイルを1つだけ読み込みます。ログインシェルでは、以下の順にファイルを探して最初に見つかったもののみを実行します。

`~/.bash_profile`
: Bashがログインシェルであれば、まず最初にこの `~/.bash_profile` を探しますが、他の（bash互換ではない）シェルは、このファイルを無視します。つまり、この起動ファイルでは、bashに固有の命令を使うことができます。ディストリビューションによって処理内容は異なりますが、`~/.profile` を追加で読み込むものが多いようです。

`~/.bash_login`
: Bashがログインシェルである場合、`~/bash_profile` が存在しないときに、次にこのファイルを探します。有れば実行しますし、無い場合には次のファイルに進みます。

`~/.profile`
: このファイルはBashに固有ではありません（つまり他のシェルでも実行されます）。このファイルでは、すべてのシェルで必要となる `PATH` などの重要な環境変数を設定します（例えば、`~/bin` ディレクトリがあれば、それを `PATH` に追加します）。ログインシェルがBashの場合は、後述の `~/.bashrc` ファイルを追加で読み込むことが多いようです。

`~/.bash_logout`
: このファイルは `起動ファイル` ではなく、ログインシェルが終了するときに実行される、いわば `終了ファイル` です。リモートセッションのクリーンアップや、コンソール画面を消去するなどの処理を行うために利用されます。

対話型ログインシェルの起動ファイルを探索する

`/etc/profile` と `/home/user2/.profile` を変更して、これらのファイルがどのように実行されるかを見てみましょう。それぞれのファイルに、実行中のファイル名を表示するコマンドを追加します。

```
root@debian:~# echo 'echo Hello from /etc/profile' >> /etc/profile
root@debian:~# echo 'echo Hello from ~/.profile' >> ~/.profile
```

NOTE リダイレクト演算子 `>>` は、コマンドの出力を既存のファイルに追記します（上書きしません）。ただし、ファイルが存在しない場合は新しく作成します。

`echo` で追加したコマンドの出力で、これらのファイルがいつ読み取られて実行されるかがわかります。確認するために、別のマシンから `ssh` を使って `user2` としてログインしてみます。

```
user2@debian:~$ ssh user2@192.168.1.6
user2@192.168.1.6's password:
Linux debian 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 27 19:57:19 2018 from 192.168.1.10
Hello from /etc/profile
Hello from /home/user2/.profile
```

追加したechoコマンドによって、最後の2行が表示されました。次の3点に注意してください。

- グローバルレベルのファイル（/etc/profile）が最初に実行された。
- 次にユーザーレベルのファイル（`~/.profile` ）が実行された（`~/.bash_profile` ないし `~/.bash_login` から、明示的に読み込まれた可能性もあります）。
- チルダ（~）がファイルの絶対パス（`/home/user2/.profile`）に展開されている。

対話型非ログインシェルの場合

グローバル（システム全体）レベル

`/etc/bash.bashrc`
: `bash` 用のグローバルレベルの起動ファイルです。このスクリプトでは、`bash` が対話的に実行されていることを確認し、画面サイズ（ウィンドウサイズ）を環境変数（`LINES` と `COLUMNS`）にセットしたり、シェルプロンプトの書式をセットするといった処理を行います。

ユーザーレベル

`~/.bashrc`
: `bash` 用のユーザーレベルの起動ファイルです。`/etc/bash.bashrc` と同様のタスク（対話的な実行や画面サイズの確認）に加えて、ヒストリーに係わる環境変数をセットしたり、`~/.bash_aliases` ファイルを読み込んでエイリアスや関数を定義します。

対話型非ログインシェルの起動ファイルを探索する

今度は、`/etc/bash.bashrc` と `/home/user2/.bashrc` を変更してみます。

```
root@debian:~# echo 'echo Hello from /etc/bash.bashrc' >> /etc/bash.bashrc
root@debian:~# echo 'echo Hello from ~/.bashrc' >> ~/.bashrc
```

`user2` が新しいシェルを起動したときはこうなります。

```
user2@debian:~$ bash
Hello from /etc/bash.bashrc
Hello from /home/user2/.bashrc
```

ここでも、2つのファイルが読み取られて実行されました。

WARNING グローバルファイルが先に実行されるので、ユーザーレベルのファイル内容が優先されます。

非対話型ログインシェルの場合

bash の起動時に、-l または --login オプションを指定すると、ログインシェルとして動作します。すなわち、対話型のログインシェルの場合と同じ起動ファイルが読み取られます。

確認するために、簡単なスクリプトを書いて実行してみます。コマンドラインから ログインオプションを指定した /bin/bash を直接呼び出すので、スクリプトファイルに shebangs（#!）は含めません。

1. test.sh というスクリプトを作成して、実行されたことが確認できるように echo 'Hello from a script' という行を入れておきます。

```
user2@debian:~$ echo "echo 'Hello from a script'" > test.sh
```

2. スクリプトを実行可能にします。

```
user2@debian:~$ chmod +x ./test.sh
```

3. -l オプションを指定した bash を呼び出して、test.sh を実行します。

```
user2@debian:~$ bash -l ./test.sh
Hello from /etc/profile
Hello from /home/user2/.profile
Hello from a script
```

結果はこうなります！ スクリプトを実行する前に、ログイン時と同様に /etc/profile と ~/.profile の両方が実行されました。

NOTE シェルスクリプトと shebangs については、後のレッスンで説明します。

次に、echo コマンドの標準出力（stdout）を、パイプで ssh の標準入力（stdin）に繋いでみましょう。

```
user2@debian:~$ echo "Hello-from-a-noninteractive-login-shell" | ssh user2@192.168.1.6
Pseudo-terminal will not be allocated because stdin is not a terminal.
user2@192.168.1.6's password:
Linux debian 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Hello from /etc/profile
Hello from /home/user2/.profile
-bash: line 1: Hello-from-a-noninteractive-login-shell: command not found
```

今度も `/etc/profile` と `~/.profile` が実行されました。出力の最初の行（stdinがターミナルではないので仮想端末は割り当てられない）と、と最後の行（リモートマシンの `-bash` からのエラーメッセージ）から、非対話型のログインシェルが動いていることが分かります。

非対話型の非ログインシェルの場合

非対話型の非ログインシェルでは、ここまでに述べた起動ファイルのいずれも読み取りません。その代わりに、環境変数 `BASH_ENV` を探して、その値をファイル名とするファイルを起動ファイルとして使用します。環境変数 については、次のレッスンで詳しく学習します。

まとめると、ログインした直後には通常、`/etc/profile` と `~/.profile` が実行されて、それらが `/etc/bash.bashrc` と `~/.bashrc` を実行します。次のコマンドの出力は、その様子を示しています。

```
root@debian:~# su - user2
Hello from /etc/bash.bashrc
Hello from /etc/profile
Hello from /home/user2/.bashrc
Hello from /home/user2/.profile
```

`su - user2` は、対話型のログインシェルを呼び出します。それぞれの起動スクリプトの最後にメッセージを出力する行を追加したことに注意してください。4行の出力は次のように説明できます。

1. `Hello from /etc/bash.bashrc` は、`/etc/profile` が `/etc/bash.bashrc` を読み取って実行したことを示します。
2. `Hello from /etc/profile` は、`/etc/profile` が最後まで実行されたことを示します。
3. `Hello from /home/user2/.bashrc` は、`~/.profile` が `~/.bashrc` を読み取って実行したことを示します。
4. `Hello from /home/user2/.profile` は、`~/.profile` が最後まで実行されたことを示します。

`su - <username>`（`su -l <username>` と `su --login <username>` も同じ意味です）でログインシェルを呼び出したときには、これらの起動ファイルが実行されることを見てきました。対して `su <username>` では、`/etc/bash.bashrc` と `~/.bashrc` が実行されます。（訳注: 実行される起動スクリプトが違うことが、最も重要な「ログインシェル」と「非ログインシェル」の違いです。）

ファイルの読み込み（sourcing）

ここまでの説明で、一部の起動スクリプトから、他のスクリプトが読み込まれて実行されることがあることを説明しました。このセクションでは、このソーシングと呼ばれる仕組みを説明します。

`.` でファイルを読み込む

多くの起動ファイルには、他のファイルを読み込むドット（`.`）コマンドが含まれています。

例えば、Debianの `.profile` を見てみましょう。

```
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
    . "$HOME/.bashrc"
```

```
    fi
```

この `if` ブロックは、`$HOME/.bashrc` が存在する場合（`-f`）に、それを読み込んで実行することを意味しています。以下のステートメントが、ファイルを読み込んでいる部分です:

```
. "$HOME/.bashrc"
```

NOTE `$HOME` はユーザーのホームディレクトリの絶対パスに展開される環境変数です。次のレッスンで説明します。

また、起動ファイルを変更した時に、再起動や再ログインせずにその変更を有効にしたい場合にも、`.` を使用します。例を示しましょう:

- `~/.bashrc` にエイリアスを追加します。

```
user2@debian:~$ echo "alias hi='echo We salute you.'" >> ~/.bashrc
```

WARNING 1つのコマンドを起動ファイルに追加する時には、上書き（`>`）と追加（`>>`）を間違えないように注意しましょう。

- `~/.bashrc` の末尾を表示して、コマンドが追加されたことを確認します。

```
user2@debian:~$ tail -n 1 !$
tail -n 1 ~/.bashrc
alias hi='echo We salute you.'
```

NOTE `!$` は直前のコマンドの最後の引数に展開されます（ヒストリー機能の利用例です）。この場合は `~/.bashrc` です。

- 手動でファイルをソースします。

```
user2@debian:~$ . ~/.bashrc
```

- エイリアスを呼び出してみて、動作を確認します。

```
user2@debian:~$ hi
We salute you.
```

NOTE エイリアス と 変数 については、次のレッスンで説明します。

`source` でファイルを読み込む

`source` コマンドは `.` コマンドの同義語です。`~/.bashrc` を読み込むには、次のようにすることもできます。

```
user2@debian:~$ source ~/.bashrc
```

起動ファイルの起源: SKEL

skel ディレクトリ（skeletalに由来）は、新しいユーザーアカウントを作成したときに、そのホームディレクトリに置かれるファイル構造のテンプレートです。その中には、新しいユーザーアカウントを作成した際に、そのホームディレクトリに置かれるシェルの起動ファイルなどが含まれています。システムにおける skel ディレクトリの絶対パスは、adduser コマンドの設定ファイルである /etc/adduser.conf の中にある SKEL 変数に定義されています。

```
user2@debian:~$ grep SKEL /etc/adduser.conf
# The SKEL variable specifies the directory containing "skeletal" user
SKEL=/etc/skel
# If SKEL_IGNORE_REGEX is set, adduser will ignore files matching this
SKEL_IGNORE_REGEX="dpkg-(old|new|dist|save)"
```

SKEL は /etc/skel に設定されているのが一般的です。つまり、ほとんどの場合、シェルを設定する起動スクリプトは /etc/skel にあります。

```
user2@debian:~$ ls -a /etc/skel/
.  ..  .bash_logout  .bashrc  .profile
```

WARNING 名前が . で始まるファイルは隠しファイルですから、ディレクトリの内容を表示するには ls -a を使います。

新規ユーザーのホームディレクトリに、個人用のスクリプトを保存するディレクトリ my_personal_scripts が置かれるように、/etc/skel にディレクトリを作成してみましょう。

1. まず root として /etc/skel に移動します。

```
root@debian:~# cd /etc/skel/
root@debian:/etc/skel#
```

2. 内容を確認します。

```
root@debian:/etc/skel# ls -a
.  ..  .bash_logout  .bashrc  .profile
```

3. ディレクトリを作成して確認します。

```
root@debian:/etc/skel# mkdir my_personal_scripts
root@debian:/etc/skel# ls -a
.  ..  .bash_logout  .bashrc  my_personal_scripts  .profile
```

4. 次に `user2` をその `home` ディレクトリと共にいったん削除します。

```
root@debian:~# deluser --remove-home user2
Looking for files to backup/remove ...
Removing files ...
Removing user `user2' ...
Warning: group `user2' has no more members.
Done.
```

5. 再度 `user2` を追加して、新しいホームディレクトリを作ります。

```
root@debian:~# adduser user2
Adding user `user2' ...
Adding new group `user2' (1001) ...
Adding new user `user2' (1001) with group `user2' ...
Creating home directory `/home/user2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

6. 最後に `user2` としてログインして、`/home/user2` にあるすべてのファイルをリストして、期待通りになっていることを確認します:

```
root@debian:~# su - user2
user2@debian:~$ pwd
/home/user2
user2@debian:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  my_personal_scripts  .profile
```

`my_personal_scripts` ディレクトリがコピーされています。

演習

1. “シェルの起動方法” 列で指定した方法で起動したシェルのタイプを調べて、表を完成させてください。

シェルの起動方法	対話型？	ログインシェル？	`echo $0` の結果
`ssh user2@machine2`			
`Ctrl`+`Alt`+`F2`			
`su - user2`			
gnome-terminal			
konsole から sakura（端末エミュレータの一種）を起動			
`echo $0` を含む `test.sh` というスクリプト			

2. 指定されたシェルを起動するには、どのような `su` および `sudo` コマンドを実行しますか？

`user2` として、対話型のログインシェル

`su`:

`sudo`:

`root` として、対話型のログインシェル

`su`:

`sudo`:

`root` として、対話型の非ログインシェル

`su`:

`sudo`:

`user2` として、対話型の非ログインシェル

`su`:

`sudo`:

3. “タイプ” 列のシェルを起動した時に読み込まれる起動ファイルはどれですか？ 読み込まれるものをマークして表を完成させてください。

タイプ	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
`user2` の対話型ログインシェル				

タイプ	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
`root` の対話型ログインシェル				
`root` の対話型・非ログインシェル				
`user2` の対話型・非ログインシェル				

発展演習

1. 以下のコードは、`Hello world!` と表示するBash用の関数を定義します。このコードを含むファイルを作成してください。

```
function hello() {
     echo "Hello world!"
}
```

   - 現在のシェルでこの関数を使用できるようにするにはどうしますか？

   - 使用できるようになったら、関数をどうやって呼び出しますか？

   - `user2` がXウィンドウセッションでターミナルを開いたときに、この関数を自動的に定義してそれを呼び出せるようにするには、コードをどの起動ファイルに置きますか？ また、その時のシェルのタイプは何ですか？

   - ログインシェルであるかどうかに関係なく、`root` が新しい対話型シェルを起動したときに、関数を定義して実行するには、どのファイルにコードを置きますか？

2. 以下の単純な `Hello_world!` スクリプトがあります:

```
#!/bin/bash
#hello_world: a simple bash script to discuss interaction in scripts.
echo "Hello world!"
```

   - スクリプトを実行可能にして実行します。スクリプトを実行するシェルは対話型でしょうか？ またそれはどうしてですか？

   - どのようなときに非対話的なシェルが起動されますか？

3. `~/.bashrc` のいくつかの変数の値を変更してそれらを有効にしたいのですが、再起動や再ログインはしたくありません。その方法を2つ示して下さい。ホームディレクトリに居るとします。

4. Johnは、LinuxサーバーでX Windowセッションを開始しました。ターミナルエミュレータでいくつかの管理タスクを実行しているときに、X Windowセッションがフリーズしてしまったので、テキストシェルを開く必要があります。

   - `tty` シェルを開くにはどうしますか？

- その時に読み込まれる起動ファイルは何ですか？

5. Lindaはあ るLinuxサーバのユーザです。彼女は、ログインしたときに日付と時刻を表示するように、システム管理者に依頼しました。システム管理者は、`.bash_login` ファイルを作成して、彼女の要望に応えました。多くのユーザーが同じ対応を求めたので、システム管理者は、すべての新しいユーザに `~/.bash_login` を提供することにしました。どうすればよいですか？

まとめ

このレッスンでは、次のことを学びました。

- シェルは、Linuxシステムにおける ユーザーの環境 を設定します。
- Bash は、GNU/Linuxディストリビューションにおいて最もよく使われているシェルです。
- シェルが最初に実行するジョブは、いくつかの起動ファイルを読み取って実行することです。
- 対話型 シェルと ログインシェル の概念。
- `bash`、`su`、`sudo`コマンドで、さまざまなタイプのシェルを起動する方法。`Ctrl`+`Alt`+`F1` ～ `F6`を使用して、`tty` コンソールからログインする方法。
- `echo $0` でシェルのタイプを確認する方法。
- 個人用の起動ファイル: `~/.bash_profile` 、`~/.profile`、 `~/.bash_login` 、`~/.bashrc` 、`~/.bash_logout`。
- システム全体の（グローバルな）起動ファイル: `/etc/profile`、`/etc/profile.d/*` 、`/etc/bash.bashrc`。
- ユーザーレベルの起動ファイルは、グローバルレベルの起動ファイルよりも優先されること。
- `>`（上書き）と `>>`（追加）を使用して、コマンドの出力をリダイレクトする方法。
- `skel` ディレクトリの役割。
- ファイルを読み込む方法（sourcing）。

このレッスンでは、以下のコマンドを説明しました。

`bash`
新しいシェルを起動します。

`su`
別のユーザーに切り替えて、新しいシェルを起動します。

`sudo`
別のユーザーに切り替えて、コマンドを実行します。

`usermod`
ユーザーアカウントの属性を変更します。

`echo`
テキストを表示します。

`ps`
現在のプロセスの状態をレポートします。

`less`
ページングしながらファイルを表示します。

`ssh`
（リモートマシンに対して）SSH接続を開始します。

`chmod`
ファイルのモードビットを変更します。たとえば、ファイルを実行可能にするなど。

`grep`
パターンに一致する行を出力する。

`ls`
ディレクトリの内容を一覧表示します。

`cd`
作業ディレクトリを変更します。

`mkdir`
ディレクトリを作成します。

`deluser`
ユーザーを削除します。

`adduser`
新しいユーザーを追加します。

`.`
ファイルを読み込みます。

`source`
ファイルを読み込みます。

`tail`
ファイルの末尾部分を表示します。

演習の解答

1. “シェルの起動方法” 列で指定した方法で起動したシェルのタイプを調べて、表を完成させてください。

シェルの起動方法	対話型？	ログインシェル？	`echo $0` の結果
`ssh user2@machine2`	Yes	Yes	`-bash`
`Ctrl`+`Alt`+`F2`	Yes	Yes	`-bash`
`su - user2`	Yes	Yes	`-su`
gnome-terminal	Yes	No	`bash`
konsole から sakura（端末エミュレータの一種）を起動	Yes	No	`/bin/bash`
`echo $0` を含む `test.sh` というスクリプト	No	No	`./test.sh`

2. 指定されたシェルを起動するには、どのような `su` および `sudo` コマンドを実行しますか？

user2 として、対話型のログインシェル

su

`su - user2` または `su -l user2`, `su --login user2`

sudo

`sudo su - user2` または `sudo su -l user2`、`sudo su --login user2`

root として、対話型のログインシェル

su

`su - root` または `su -`

sudo

`sudo su - root` または `sudo su -`、`sudo -i`

root として、対話型の非ログインシェル

su

`su root` または `su`

sudo

`sudo su root` または `sudo su`、`sudo -s`、 `sudo -u root -s`

user2 として、対話型の非ログインシェル

su

`su user2`

sudo

`sudo su user2` または `sudo -u user2 -s`

3. “タイプ” 列のシェルを起動した時に読み込まれる起動ファイルはどれですか？ 読み込まれるものをマークして表を完成させてください。

タイプ	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
`user2` の対話型ログインシェル	Yes	Yes	Yes	Yes
`root` の対話型ログインシェル	Yes	Yes	Yes	Yes
`root` の対話型・非ログインシェル	No	Yes	No	Yes
`user2` の対話型・非ログインシェル	No	Yes	No	Yes

発展演習の解答

1. Bashでは、ファイルに次のコードを含めることで、単純な `Hello world!` 関数を記述できます。

```
function hello() {
     echo "Hello world!"
}
```

   - 現在のシェルでこの関数を使用できるようにするにはどうしますか？

     現在のシェルで関数を有効とするには、関数を含むファイルを読み込みます（ソーシング）。

   - 使用できるようになったら、関数をどうやって呼び出しますか？

     ターミナルに関数名を入力します。

   - `user2` がXウィンドウセッションでターミナルを開いたときに、この関数を自動的に定義してそれを呼び出せるようにするには、コードをどの起動ファイルに置きますか？ また、その時のシェルのタイプは何ですか？

     `/home/user2/.bashrc` ファイルが最適です。起動されたシェルは、対話型の非ログインシェルになります。

   - ログインシェルであるかどうかに関係なく、`root` が新しい対話型シェルを起動したときに、関数を定義して実行するには、どのファイルにコードを置きますか？

     `/root/.bashrc` ファイルが最適です。このファイルは、ログインシェルであるかどうかに関係なく、すべての対話型シェルで実行されるからです。

2. 以下の単純な `Hello_world!` スクリプトがあります:

```
#!/bin/bash
#hello_world: a simple bash script to discuss interaction in scripts.
echo "Hello world!"
```

   - スクリプトを実行可能にして実行します。スクリプトを実行するシェルは対話型でしょうか？ またそれはどうしてですか？

     いいえ、対話型ではありません。スクリプトを実行するシェルは、プロンプトを表示してユーザーにコマンドを求めることがありませんから、非対話型になります。

   - どのようなときに非対話型のシェルが起動されますか？

     シェルスクリプトを実行するシェル（インタープリタとしてのシェル）は、一般的に非対話型になります。厳密な定義は、bashのmanページの「起動（INVOCATION）」セクションを参照してください。

3. `~/.bashrc` のいくつかの変数の値を変更してそれらを有効にしたいのですが、再起動や再ログインはしたくありません。その方法を2つ示して下さい。ホームディレクトリに居るとします。

```
$ source .bashrc
```

または、

```
$ . .bashrc
```

4. Johnは、LinuxサーバーでX Windowセッションを開始しました。ターミナルエミュレータでいくつかの管理タスクを実行しているときに、X Windowセッションがフリーズしてしまったので、テキストシェルを開く必要があります。
   - `tty` シェルを開くにはどうしますか？

     Ctrl+Alt+F1～Ctrl+Alt+F6を押して、6つのttyシェルの1つに入ります。（訳注仮想コンソールを使用できるのは、x86-64などの PCアーキテクチャ のマシンだけです。また、ディストリビューションによってファンクションキーの割り当てが異なります。）

   - 読み込まれる起動ファイルは何ですか？

     `/etc/profile`
     `/home/john/.profile`

5. LindaはあるLinuxサーバのユーザです。彼女は、ログインしたときに日付と時刻を表示するように、システム管理者に依頼しました。システム管理者は、`.bash_login` ファイルを作成して、彼女の要望に応えました。多くのユーザーが同じ対応を求めたので、システム管理者は、すべての新しいユーザに `~/.bash_login` を提供することにしました。どうすればよいですか？

   `/etc/skel` ディレクトリに、`.bash_login` を置きます。

105.1 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	105 シェルとシェルスクリプト
Objective:	105.1 シェル環境をカスタマイズして使用する
Lesson:	2 of 3

はじめに

情報を一時的に記録するための仮想的な箱を変数と呼びます。Bashでは、シェルに局所的　（変数を作成したシェルの中でのみ利用できるもの）と　環境に大域的　（子シェルやプロセスに継承されるもの）の2種類の変数があります。前のレッスンでは、シェルとその初期化スクリプトについて説明しました。それらの起動ファイルで、変数やエイリアス、関数を定義することで、好みのシェル環境を作成できることを学習していきましょう。

変数：割り当てと参照

変数は、値（内容）に名前を付けることで定義します。

名前に値を与えることを 変数への割り当て （ないしは 代入）と呼び、それによって変数が作成ないし設定されます。また、変数の名前からその内容にアクセスすることを 変数の参照 と呼びます。

変数に値を割り当てる構文は次のとおりです。

```
<変数名>=<値>
```

例えば次のようになります。

```
$ distro=zorinos
```

変数 `distro` は `zorinos` と同じです。つまり、値 `zorinos` を保持するメモリ領域があり、`distro`

はそこへのポインタです。

なお、変数を割り当てるときは、等号の両側にスペースを入れることはできないことに注意してください。

```
$ distro =zorinos
-bash: distro: command not found
$ distro= zorinos
-bash: zorinos: command not found
```

等号の前後に空白を入れると、Bashは `distro` ないし `zorinos` をコマンドと見なします。

変数を参照するには（つまり、その値を確認するには）、変数名の前に `$` を付けた `echo` コマンドを使用します。

```
$ echo $distro
zorinos
```

変数名

変数の名前を決めるときには、いくつかのルールがあります。

変数の名前には、文字（`a-z`、`A-Z`）、数字（`0-9`）、アンダースコア（`_`）のみが使えます。

```
$ distro=zorinos
$ echo $distro
zorinos
$ DISTRO=zorinos
$ echo $DISTRO
zorinos
$ distro_1=zorinos
$ echo $distro_1
zorinos
$ _distro=zorinos
$ echo $_distro
zorinos
```

先頭が数字ではいけません。Bashが混乱します。

```
$ 1distro=zorinos
-bash: 1distro=zorinos: command not found
```

空白を含めることはできません（引用符も使えません）。慣例的に、空白の代わりにアンダースコアを使用します。

```
$ "my distro"=zorinos
```

```
-bash: my: command not found
$ my_distro=zorinos
$ echo $my_distro
zorinos
```

変数の値

変数の値を参照するときにも、いくつかのルールをがあります。

変数には、任意の英数字（`a-z`、`A-Z`、`0-9`）と、ほとんどの記号（`?`、`!`、`*`、`.`、`/`、など）を含めることができます。（訳注：言語設定によりますが、日本語を含めることもできます。）

```
$ distro=zorin12.4?
$ echo $distro
zorin12.4?
```

変数値にひとつ以上の空白が含まれている場合は、引用符で囲む必要があります。

```
$ distro=zorin 12.4
-bash: 12.4: command not found
$ distro="zorin 12.4"
$ echo $distro
zorin 12.4
$ distro='zorin 12.4'
$ echo $distro
zorin 12.4
```

変数の値に、リダイレクトに使用される文字（`<`、`>`）やパイプ記号（`|`）などの記号が含まれている場合は、値全体を引用符で囲む必要があります。次のコマンドを実行すると、`zorin　`という名前の空のファイルが作成されてしまいます。

```
$ distro=>zorin
$ echo $distro

$ ls zorin
zorin
```

引用符を使用するとこうなります。

```
$ distro=">zorin"
$ echo $distro
>zorin
```

なお、一重引用符と二重引用符の働きは異なります。変数操作（代入なのか参照なのか）に応じて、どちらの引用符を使用するかによって結果が異なります。変数への代入の場合、一重引用符は値のすべての文字を 文字通り に扱いますが、二重引用符の場合は 変数置換 が行われます。

```
$ lizard=uromastyx
$ animal='My $lizard'
$ echo $animal
My $lizard
$ animal="My $lizard"
$ echo $animal
My uromastyx
```

一方、値に空白を含む変数を参照する場合には、（コマンドラインでの）フィールド分割　や　パス名の拡張 を回避するために、`echo` コマンドの引数を二重引用符で囲む必要があります。

```
$ lizard="   genus   |   uromastyx"
$ echo $lizard
genus | uromastyx
$ echo "$lizard"
   genus   |   uromastyx
```

変数に感嘆符を含む文字列を代入する場合、感嘆符は文字列の最後の文字なくてはいけません。（感嘆符の後に文字があると、Bashは　`history`　イベントを参照していると見なします）。（訳注：感嘆符を含む場合も一重引用符で囲めば特別な意味が無くなります。）

```
$ distro=zorin.?/!os
-bash: !os: event not found
$ distro=zorin.?/!
$ echo $distro
zorin.?/!
```

バックスラッシュは、別のバックスラッシュでエスケープする必要があります。また、バックスラッシュが文字列の最後の文字でありそれがエスケープされていない場合、Bashは改行が必要であると解釈して新しい行の入力を待ちます。

```
$ distro=zorinos\
>
$ distro=zorinos\\
$ echo $distro
zorinos\
```

次の2つのセクションでは、 ローカル 変数と 環境 変数の違いを説明します。

ローカル変数またはシェル変数

ローカル変数またはシェル変数は、それらを作成したシェルでのみ有効です。ローカル変数名は小文字で表記するのが慣例となっています。

挙動を確認するために、ローカル変数を作成しましょう。上で説明したように適当な変数名を選択し、適当な値を代入します。例えば：

```
$ reptile=tortoise
```

(訳注：reptileは「は虫類」、tortoiseは「カメ」の意)

`echo` コマンドで変数を参照して確認します。

```
$ echo $reptile
tortoise
```

スクリプトの作成など、変数を変更不可とすることが適当な場合があります。変数を変更不可（読み出し専用）にしたい場合は、`readonly` で変数を作成します。

```
$ readonly reptile=tortoise
```

あるいは、作成した後に変更不可とすることもできます。

```
$ reptile=tortoise
$ readonly reptile
```

ここで、`reptile` の値を変更しようとすると、エラーになります。

```
$ reptile=lizard
-bash: distro: readonly variable
```

NOTE 現在のセッションのすべての読み出し専用変数を一覧表示するには、ターミナルに `readonly` または `readonly -p` と入力します。

ローカル変数を扱うときに役立つコマンドは `set` です。

`set` は、現在割り当てられているすべてのシェル変数と関数を一覧表示します。行数が多いことがある（自分で試してみてください！）ので、`less` などのページャーと組み合わせて使用することがお勧めです。

```
$ set | less
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_fignore:histappend:inte
ractive_comments:login_shell:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_COMPAT_DIR=/etc/bash_completion.d
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="4" [2]="12" [3]="1" [4]="release" [5]="x86_64-pc-linux-gnu")
```

```
BASH_VERSION='4.4.12(1)-release'
(...)
```

作成した `reptile` 変数があるでしょうか？

```
$ set | grep reptile
reptile=tortoise
```

はい、ありました！

ただし、ローカル変数である `reptile` は、現在のシェルから生成された子プロセスには継承されません。

```
$ bash
$ set | grep reptile
$
```

そしてもちろん、その値をエコーで出力することもできません。

```
$ echo $reptile
$
```

NOTE ターミナルに `bash` コマンドを入力すると、新しい（子）シェルが起動します。

変数（ローカルまたはグローバルどちらも）を削除するには、`unset` コマンドを使用します。

```
$ echo $reptile
tortoise
$ unset reptile
$ echo $reptile
$
```

NOTE `unset` の後には、変数の名前だけを指定します（`$` 記号は必要ありません）。

グローバル変数または環境変数

グローバル変数（環境変数とも呼ぶ）は、現在のシェルと、シェルから生成されたすべてのプロセスに引き継がれます。環境変数名は大文字とするのが慣例です。

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

これらの変数の値を、他の変数に代入することができます。参照した変数の値と、代入された変数の値は同じになります。

```
$ my_shell=$SHELL
$ echo $my_shell
/bin/bash
$ your_shell=$my_shell
$ echo $your_shell
/bin/bash
$ our_shell=$your_shell
$ echo $our_shell
/bin/bash
```

シェルのローカル変数を環境変数にするには、`export` コマンドを使用します。

```
$ export reptile
```

`export reptile` によってローカル変数が環境変数に変換されたので、子シェルがその変数を使用できるようになりました。

```
$ bash
$ echo $reptile
tortoise
```

`export` を使用して、変数への代入とエクスポートを同時に実行できます。

```
$ export amphibian=frog
```

（訳注：amphibianは「両生類」、frogは「カエル」を意味します。）

これで、新しいBashのインスタンスからも、新しい変数を利用できます。

```
$ bash
$ echo $amphibian
frog
```

NOTE `export -n <環境変数名>` によって、環境変数をローカル変数に戻すことができます。

`export` コマンドを単体（ないしは `-p` オプション付きで）実行すると、すべての環境変数を一覧表示します。

```
$ export
declare -x HOME="/home/user2"
declare -x LANG="en_GB.UTF-8"
declare -x LOGNAME="user2"
(...)
declare -x PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
declare -x PWD="/home/user2"
```

```
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_CLIENT="192.168.1.10 49330 22"
declare -x SSH_CONNECTION="192.168.1.10 49330 192.168.1.7 22"
declare -x SSH_TTY="/dev/pts/0"
declare -x TERM="xterm-256color"
declare -x USER="user2"
declare -x XDG_RUNTIME_DIR="/run/user/1001"
declare -x XDG_SESSION_ID="8"
declare -x reptile="tortoise"
```

NOTE コマンド `declare -x` はコマンド `export` と同じです。

すべての環境変数の一覧表示するために、`env` コマンドと `printenv` コマンドの2つも使えます。

```
$ env
SSH_CONNECTION=192.168.1.10 48678 192.168.1.7 22
LANG=en_GB.UTF-8
XDG_SESSION_ID=3
USER=user2
PWD=/home/user2
HOME=/home/user2
SSH_CLIENT=192.168.1.10 48678 22
SSH_TTY=/dev/pts/0
MAIL=/var/mail/user2
TERM=xterm-256color
SHELL=/bin/bash
SHLVL=1
LOGNAME=user2
XDG_RUNTIME_DIR=/run/user/1001
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
_=/usr/bin/env
```

`printenv` コマンドは `env` と同様に一覧を表示するだけではなく、`echo` コマンドと同様に変数の値を確認するためにも使用できます。

```
$ echo $PWD
/home/user2
$ printenv PWD
/home/user2
```

ただし `printenv` では、変数名の前に `$` を付けないことに注意してください。

NOTE 変数 `PWD` には、現在の作業ディレクトリのパスが格納されています。これらの一般的な環境変数については後に学びます。

環境を変更してプログラムを実行する

`env` コマンドを使って、シェルの環境変数を、プログラムの実行時に変更することができます。

最小の環境で新しいBashセッションを開始するには、ほとんどの変数（および関数とエイリアス）をクリアする `-i` オプションを付けた `env` コマンドを使用します。

```
$ env -i bash
```

これで、ほとんどの環境変数がなくなりました。

```
$ echo $USER
$
```

ほんの少ししか残っていないことを確認してみます。

```
$ env
LS_COLORS=
PWD=/home/user2
SHLVL=1
_=/usr/bin/printenv
```

`env` を使用して、実行するプログラムに渡す環境を指定することもできます。

前のレッスンの、非対話的な非ログインシェル　　に関する説明で、スクリプトは標準の起動ファイルを読み込まず、代わりに　`BASH_ENV`　変数が存在する場合にはその値を起動ファイル名として使用することを説明しました。

その働きを確認してみましょう。

1. `.startup_script` という独自の起動スクリプトを作成し、以下の内容とします。

   ```
   CROCODILIAN=caiman
   ```

   （訳注：crocodilianは「ワニ類」、caimanは「カイマン」の意です。）

2. `test_env.sh` という名前のBashスクリプトを作成して、以下の内容とします。

   ```
   #!/bin/bash

   echo $CROCODILIAN
   ```

3. `test_env.sh` スクリプトに実行可能ビットを設定します。

   ```
   $ chmod +x test_env.sh
   ```

4. `env` を使用して `BASH_ENV` を `.startup_script` に設定して、`test_env.sh` を実行します。

```
$ env BASH_ENV=/home/user2/.startup_script ./test_env.sh
caiman
```

`env` コマンドは省略することもできます。（訳注：Bashには `env` コマンドの変数代入に相当する機能が内蔵されています。`env` コマンドのオプション（`-i` など）を指定する必要がない場合には、`env` コマンドを指定する必要はありません。）

```
$ BASH_ENV=/home/user2/.startup_script ./test_env.sh
caiman
```

NOTE

`test_env.sh` の先頭にある `#!/bin/bash` や `chmod +x` コマンドが分からなくても慌てる必要はありません！後のレッスンでシェルスクリプトに必要なすべてを学びます。今のところ、カレントディレクトリのスクリプトを実行するには、`./some_script` を使用することを覚えておいてください。

一般的な環境変数

ここからは、Bashの起動ファイルで設定されることが多い、一般的な環境変数のいくつかを紹介しましょう。デフォルトで設定されている環境変数は、ディストリビューションによって異なります。

`DISPLAY`

Xサーバーに関するもので、この変数の値には以下の3つの要素が含まれます。

- Xサーバーを実行しているホスト名（存在しない場合は `localhost` を意味する）。
- 区切り文字のコロン。
- 数字（通常は `0` で、そのコンピューターに接続されているディスプレイを指す）。

```
$ printenv DISPLAY
:0
```

この変数の値が空の場合は、Xウィンドウシステムを使用していないことを意味します。`my.xserver:0:1` のように、追加の番号がある場合は、複数存在する場合の画面番号を意味します。

`HISTCONTROL`

この変数は `HISTFILE`（後述）に保存するコマンド行の種類を制御します。以下の3つのいずれかの値を指定できます。

`ignorespace`
空白で始まるコマンド行は保存しない。

`ignoredups`
直前のコマンド行と同じコマンド行は保存しない。

`ignoreboth`
上記2種類のいずれかに該当するコマンド行は保存しない。


```
$ echo $HISTCONTROL
ignoreboth
```

HISTSIZE
シェルセッションの最中にメモリに保存されるコマンド行の最大数を指定します。

```
$ echo $HISTSIZE
1000
```

HISTFILESIZE
HISTFILE に保存されるコマンド行の数（開始時に読み込まれるコマンド数、終了時に書き込むコマンド数）を指定します。

```
$ echo $HISTFILESIZE
2000
```

HISTFILE
入力されたコマンドを格納するファイルの名前。デフォルトでは、~/.bash_history です。

```
$ echo $HISTFILE
/home/user2/.bash_history
```

NOTE HISTFILE の内容を表示するには、単純に history と入力します。あるいは、history 3 のように、引数としてコマンド数を指定することで、history が表示するコマンドの数を制限できます。

HOME
ユーザーがログインしたときに、この変数にはホームディレクトリの絶対パスが格納されます。

この ~/.profile の一節から一目瞭然でしょう。（このコードは "$HOME/.bashrc" が存在すれば、それを取り込んでいます。）

```
# include .bashrc if it exists
if [ -f "$HOME/.bashrc" ]; then
. "$HOME/.bashrc"
fi
```

NOTE if ステートメントを完全に理解していなくても心配することはありません。後のレッスンで説明します。

~ は $HOME と同じであることを忘れないで下さい。

```
$ echo ~; echo $HOME
/home/carol
```

```
/home/carol
```

NOTE 複数のコマンドをセミコロン（;）で区切り、続けて実行することができます。

`if` ステートメントの例を見てみましょう。

```
$ if [ ~ == "$HOME" ]; then echo "true"; else echo "false"; fi
true
```

NOTE 等号 `=` は変数への代入に、`==` は等しいことを調べるために、それぞれ使用されることを覚えておきましょう。

HOSTNAME
この変数には、ホストコンピューターの名前が格納されます。

```
$ echo $HOSTNAME
debian
```

HOSTTYPE
この変数には、ホストコンピュータのプロセッサ アーキテクチャが格納されます。

```
$ echo $HOSTTYPE
x86_64
```

LANG
この変数には、システムのロケール（言語指定）が格納されます。

```
$ echo $LANG
en_UK.UTF-8
```

LD_LIBRARY_PATH
この変数には、プログラムが使用する共有ライブラリの格納位置を、コロンで区切ったディレクトリのリストとして格納します。

```
$ echo $LD_LIBRARY_PATH
/usr/local/lib
```

MAIL
この変数には、Bashが電子メールの到着をチェックするファイル名を格納します。

```
$ echo $MAIL
/var/mail/carol
```

この変数に格納される値としては、`/var/spool/mail/$USER` などが一般的です。

MAILCHECK

この変数には、Bashが新着メールをチェックする頻度を秒単位の数値で格納します。

```
$ echo $MAILCHECK
60
```

PATH

この環境変数には、プログラムを実行するように指示されたときに、Bashが実行可能ファイルを探すディレクトリのリストが格納されています。サンプルに使用したマシンでは、システム全体の `/etc/profile` ファイルでこの変数が設定されています。

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH
```

`if` ステートメントによってユーザーIDを確認し、□—□その結果（`root` であるか一般ユーザーか）に応じて□—□いずれかの `PATH` が設定されます。最後に、選択した `PATH` を `export` でグローバルな環境変数にします。

`PATH` の値に関して、2つのことに注意してください。

- ディレクトリ名は絶対パスで記述します。
- コロンは区切り文字です。

一般ユーザーの `PATH` に `/usr/local/sbin` ディレクトリを含めたい場合は、行を次のように変更します。

```
(...)
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/sbin"
fi
export PATH
```

この変更によって、一般ユーザーとしてログインしたときに、変数の値がどのように変化するかを見てみましょう。

```
# su - carol
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/sbin
```

NOTE 現在の `PATH` に `/usr/local/sbin` を追加することもできます。コマンドライン

で `PATH=/usr/local/sbin:$PATH` または `PATH=$PATH:/usr/local/sbin` と入力します。前者は実行可能ファイルを検索する際に `/usr/local/sbin` を最初に、後者は同じディレクトリを最後に探します。

PS1

この変数には、Bashプロンプトの文字列が格納されます。次の部分コード（これも `/etc/profile` から）では、 `if` ステートメントでユーザーIDを確認し、それに応じたプロンプトを表示します（`root` の場合は `#`、一般ユーザーの場合は `$`）。

```
if [ "`id -u`" -eq 0 ]; then
  PS1='# '
else
  PS1='$ '
fi
```

NOTE `root` の `id` は `0` です。 `root` になって、`id -u` で確認してみましょう。

その他にも、以下のようなプロンプト変数があります。

PS2

コマンドが複数行に及ぶ場合に、継続行のプロンプトとして使用されます。通常は `>` です。

PS3

`select` コマンドのプロンプトとして使用されます。

PS4

デバッグ用に出力された行を示します。通常は `+` です。

SHELL

この変数には、現在のシェルの絶対パスが格納されます。

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

USER

この変数には、現在のユーザー名が格納されます。

```
$ echo $USER
carol
```


演習

1. “コマンド” 欄の変数割り当てで得られる変数が “ローカル”（シェル変数） か、 “グローバル”（環境変数） か示してください。

コマンド	ローカル	グローバル
`debian=mother`		
`ubuntu=deb-based`		
`mint=ubuntu-based; export mint`		
`export suse=rpm-based`		
`zorin=ubuntu-based`		

2. “コマンド” で得られる “出力” を調べて、その意味を説明してください。

コマンド	出力	意味
`echo $HISTCONTROL`	`ignoreboth`	
`echo ~`	`/home/carol`	
`echo $DISPLAY`	`reptilium:0:2`	
`echo $MAILCHECK`	`60`	
`echo $HISTFILE`	`/home/carol/.bash_history`	

3. “誤ったコマンド” 欄には誤った変数への割り当てが行われています。“正しいコマンド” 欄に正しいコマンド、“参照方法” 欄にその変数を参照するコマンド、“出力結果” 欄にその実行結果を記入して下さい。

誤ったコマンド	正しいコマンド	参照方法	出力結果
`lizard =chameleon`			
`cool lizard=chameleon`			
`lizard=cha|me|leon`			
`lizard=/** chameleon **/`			
`win_path=C:\path\to\dir\`			

4. “やりたいこと” 欄に応じて、“コマンド” 欄を記入してください。

やりたいこと	コマンド
シェルの言語指定をUTF-8のスペイン語（`es_ES.UTF-8`）にセットする	
作業ディレクトリの名前を出力する	

やりたいこと	コマンド
ssh 接続に関する情報を格納する環境変数を表示する	
実行可能ファイルを検索する最後のディレクトリとして /home/carol/scripts を含めるように、PATH を設定する	
my_path の値を PATH にセットする	
my_path の値を PATH の値にセットする	

5. mammal という名前のローカル変数を作成し、それに値 gnu を割り当てて下さい。

6. 別のローカル変数 ver_sub を作成します。mammal の変数置換を使用して、echo $var_sub でその値を参照した結果が The value of mammal is gnu となるようにして下さい。

7. mammal を環境変数に変換して下さい。

8. set と grep を使って、変数一覧から mammal を検索してください。

9. 今度は、env と grep を使って mammal を検索してください。

10. 値が penguin である BIRD という名前の環境変数を作ります。2つのコマンドを1行で連続して使ってください。

11. 今度は1つのコマンドで、値が yellow-eyed penguin である NEW_BIRD という名前の環境変数を作って下さい。

12. 自分が user2 であるとして、ホームディレクトリに bin という名前のフォルダを作成して下さい。3つの方法があります。

13. ~/bin フォルダーが bash がプログラムを検索する最初のディレクトリになるように PATH をセットするコマンドを実行して下さい。

14. 再起動後も PATH の値が変更されないように、~/.profile に if ステートメントを追加してください。

発展演習

1. `let`: 算術式の評価以上のもの。
   - helpコマンドないしWebで調べて、`let` コマンドが変数をセットする動作を調べてみましょう。`your_val` という新しいローカル変数を作成して、5に5を足した結果の値（すなわち `10`）を割り当てて下さい。
   - 次に、`your_val` と言う名前の変数を作成して、`my_val` の値を2で割った結果の値（すなわち `5`）を割り当てて下さい。
2. コマンドの実行結果を変数にすることも可能で、コマンド置換 と呼ばれます。それを調べてから、次の `music_info` という関数を見てください。

```
music_info(){
latest_music=`ls -t ~/Music | head -n 5`
echo -e "Your latest 5 music files:\n$latest_music"
}
```

コマンド `ls -t ~/Music | head -n 5` の結果が、`latest_music` 変数の値になります。`echo` コマンドで `latest_music` 変数を参照すると、`Music` フォルダーに保存されている音楽ファイルを新しい順に5つ出力します。

次のコマンドと同じ意味になるのはどれですか？

```
latest_music=`ls -t ~/Music | head -n 5`
```

選択肢 A:

```
latest_music=$(ls -t ~/Music| head -n 5)
```

選択肢 B:

```
latest_music="(ls -t ~/Music| head -n 5)"
```

選択肢 C:

```
latest_music=((ls -t ~/Music| head -n 5))
```

まとめ

このレッスンでは、次のことを学びました。

- シェル環境における変数は、シェル自身や他のプログラムによって利用されるので非常に重要であること。
- 変数を割り当てて、参照する方法。
- ローカル 変数と グローバル（環境）変数の違い。
- 変数を読み出し専用とする方法。
- `export` コマンドで、ローカル変数を環境変数に変換する方法。
- すべての環境変数を一覧表示する方法。
- 変更した環境でプログラムを実行する方法。
- 起動スクリプトによって変数を永続的とする方法。
- 一般的な環境変数: `DISPLAY`、`HISTCONTROL`、`HISTSIZE`、`HISTFILESIZE`、`HISTFILE`、`HOME`、`HOSTNAME`、`HOSTTYPE`、`LANG`、`LD_LIBRARY_PATH`、`MAIL`、`MAILCHECK`、`PATH`、`PS1` とその他のプロンプト変数、`SHELL`、`USER`。
- チルダ（~）の意味.
- `if` ステートメントの基本の基本。

このレッスンでは、以下のコマンドを説明しました。

`echo`
変数を参照する。

`ls`
ディレクトリの内容を一覧表示します。

`readonly`
変数を変更不可とする。 シェルセッションのすべての読み取り専用変数を一覧表示する。

`set`
シェルセッションのすべての変数と関数を一覧表示する。

`grep`
パターンに一致する行を出力する。

`bash`
新しいシェルを起動する。

`unset`
変数を削除する。

`export`
ローカル変数を環境変数に変換する。 環境変数を一覧表示する。

`env`
環境変数を一覧表示する。 環境を変更してプログラムを実行する。

`printenv`
環境変数を一覧表示する。 変数を参照する。

`chmod`
ファイルのモードビットを変更します。たとえば、ファイルを実行可能にするなど。

`history`
以前実行したコマンドを一覧表示する。

`su`
ユーザーIDを変更する、ないし、スーパーユーザーになる。

`id`
ユーザーIDを出力する。

演習の解答

1. “コマンド” 欄の変数割り当てで得られる変数が “ローカル”（シェル変数） か、 “グローバル”（環境変数） か示してください。

コマンド	ローカル	グローバル
`debian=mother`	Yes	No
`ubuntu=deb-based`	Yes	No
`mint=ubuntu-based; export mint`	No	Yes
`export suse=rpm-based`	No	Yes
`zorin=ubuntu-based`	Yes	No

2. “コマンド” で得られる “出力” を調べて、その意味を説明してください。

コマンド	出力	意味
`echo $HISTCONTROL`	`ignoreboth`	重複するコマンドとスペースで始まるコマンドは、どちらも `history` に保存されない
`echo ~`	`/home/carol`	`carol` の `HOME` は `/home/carol` である
`echo $DISPLAY`	`reptilium:0:2`	マシン `reptilium` でXサーバーが実行されており、最初のディスプレイの2番目の画面を使用している
`echo $MAILCHECK`	`60`	メールチェックは60分ごとに行われる
`echo $HISTFILE`	`/home/carol/.bash_history`	`history` は `/home/carol/.bash_history` に保存される

3. “誤ったコマンド” 欄には誤った変数への割り当てが行われています。“正しいコマンド” 欄に正しいコマンド、“参照方法” 欄にその変数を参照するコマンド、“出力結果” 欄にその実行結果を記入して下さい。

誤ったコマンド	正しいコマンド	参照方法	出力結果
`lizard =chameleon`	`lizard=chameleon`	`echo $lizard`	`chameleon`
`cool lizard=chameleon`	`cool_lizard=chameleon`（例）	`echo $cool_lizard`	`chameleon`
`lizard=cha|me|leon`	`lizard="cha|me|leon"` または `lizard='cha|me|leon'`	`echo $lizard`	`cha|me|leon`

誤ったコマンド	正しいコマンド	参照方法	出力結果
`lizard=/** chameleon **/`	`lizard="/** chameleon **/"` または `lizard='/** chameleon **/'`	`echo "$lizard"`	`/** chamelon **/`
`win_path=C:\path\to\dir\`	`win_path=C:\\path\\to\\dir\\`	`echo $win_path`	`C:\path\to\dir\`

4. “やりたいこと” 欄に応じて、“コマンド” 欄を記入してください。

やりたいこと	コマンド
シェルの言語指定をUTF-8のスペイン語（`es_ES.UTF-8`）にセットする	`LANG=es_ES.UTF-8`
作業ディレクトリの名前を出力する	`echo $PWD` ないし `pwd`
`ssh` 接続に関する情報を格納する環境変数を表示する	`echo $SSH_CONNECTION`
実行可能ファイルを検索する最後のディレクトリとして `/home/carol/scripts` を含めるように、`PATH` を設定する	`PATH=$PATH:/home/carol/scripts`
`my_path` の値を `PATH` にセットする	`my_path=PATH`
`my_path` の値を `PATH` の値にセットする	`my_path=$PATH`

5. `mammal` という名前のローカル変数を作成し、それに値 `gnu` を割り当てて下さい。

```
mammal=gnu
```

6. 別のローカル変数 `var_sub` を作成します。`mammal` の変数置換を使用して、`echo $var_sub` でその値を参照した結果が `The value of mammal is gnu` となるようにして下さい。

```
var_sub="The value of mammal is $mammal"
```

7. `mammal` を環境変数に変換して下さい。

```
export mammal
```

8. `set` と `grep` を使って、変数一覧から `mammal` を検索してください。

```
set | grep mammal
```

9. 今度は、`env` と `grep` を使って `mammal` を検索してください。

```
env | grep mammal
```

10. 値が `penguin` である `BIRD` という名前の環境変数を作ります。2つのコマンドを1行で連続して使ってください。

```
*BIRD=penguin; export BIRD*
```

11. 今度は1つのコマンドで、値が `yellow-eyed penguin` である `NEW_BIRD` という名前の環境変数を作って下さい。

```
export NEW_BIRD="yellow-eyed penguin"
```

または

```
export NEW_BIRD='yellow-eyed penguin'
```

12. 自分が `user2` であるとして、ホームディレクトリに `bin` という名前のフォルダを作成して下さい。3つの方法があります。

```
mkdir ~/bin
```

または

```
mkdir /home/user2/bin
```

または

```
mkdir $HOME/bin
```

13. `~/bin` フォルダーが `bash` がプログラムを検索する最初のディレクトリになるように `PATH` をセットするコマンドを実行して下さい。

```
PATH="$HOME/bin:$PATH"
```

`PATH=~/bin:$PATH` ないし `PATH=/home/user2/bin:$PATH` も正解です。

14. 再起動後も `PATH` の値が変更されないように、`~/.profile` に `if` ステートメントを追加してください。

```
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
```

```
fi
```

発展演習の解答

1. `let`: 計算だけじゃない
   - helpコマンドないしWebで調べて、`let` コマンドが変数をセットする動作を調べてみましょう。`your_val` という新しいローカル変数を作成して、5に5を足した結果の値（すなわち `10`）を割り当てて下さい。

     ```
     let "my_val = 5 + 5"
     ```

     または

     ```
     let 'my_val = 5 + 5'
     ```

   - 次に、`your_val` と言う名前の変数を作成して、`my_val` の値を2で割った結果の値（すなわち `5`）を割り当てて下さい。

     ```
     let "your_val = $my_val / 2"
     ```

     または

     ```
     let 'your_val = $my_val / 2'
     ```

2. コマンドの実行結果を変数に格納することもでき、コマンド置換 と呼ばれます。それを調べてから、次の `music_info` という関数を見てください。

   ```
   music_info(){
   latest_music=`ls -t ~/Music | head -n 5`
   echo -e "Your latest 5 music files:\n$latest_music"
   }
   ```

   コマンド `ls -t ~/Music | head -n 6` の結果が、`latest_music` 変数の値になります。`echo` コマンドで `latest_music` 変数を参照すると、`Music` フォルダーに保存されている音楽ファイルを新しい順に5つ出力します。

   次のコマンドと同じ意味になるのはどれですか？

   ```
   latest_music=`ls -t ~/Music | head -n 5`
   ```

   選択肢 Aが正解です。

   ```
   latest_music=$(ls -t ~/Music| head -n 5)
   ```

105.1 レッスン 3

Certificate:	LPIC-1
Version:	5.0
Topic:	105 シェルとシェルスクリプト
Objective:	105.1 シェル環境をカスタマイズする
Lesson:	3 of 3

はじめに

前のレッスンでは、シェルと起動スクリプト、変数について学習しました。このレッスンでは、2つの非常に興味深いシェルの機能である　エイリアス　と　関数　を学び、シェルのカスタマイズについてまとめます。実際の所、一連の変数とエイリアス、関数が互いに影響し合って「シェル環境」を構成しています。

エイリアス　と　関数　という2つのシェル機能は、いずれも柔軟で効率的であり、カプセル化という概念を実現します。つまり、これらは、一連のコマンドやその繰り返しを、1つのコマンドにまとめます。

エイリアスの作成

エイリアスは、コマンドに別の名前をつけるものです。エイリアスの定義に従って、入力したコマンドとは異なるコマンドを実行します。

エイリアスを宣言する構文は直感的です。キーワード　`alias`　に続けて、割り当てを宣言します。すなわち、エイリアス名、等号、割り当てる1つ以上の コマンド です。

```
alias alias_name=command(s)
```

例えば次のようになります。

```
$ alias oldshell=sh
```

この役立たずなエイリアスは、ユーザーが端末に `oldshell` と入力すると、古い `sh` シェルのインスタンスを実行します。

```
$ oldshell
$
```

エイリアスの威力は、長いコマンドに短い名前を付けることができることです。

```
$ alias ls='ls --color=auto'
```

NOTE `ls` が表示する色については、`dir_colors(5)` と `dircolors(1)` のmanページを見てください。

同様に、複数のコマンドを連続して実行するエイリアスも作成できます。コマンドラインと同様に、セミコロン（;）が区切り文字です。たとえば、`git` コマンドのパスとそのバージョンを表示するエイリアスは、次のようになります。

```
$ alias git_info='which git;git --version'
```

エイリアスを呼び出すには、その名前をコマンドラインに入力します。

```
$ git_info
/usr/bin/git
git version 2.7.4
```

`alias` コマンドは、起動中のシェルで使用可能なすべてのエイリアスを一覧表示します。

```
$ alias
alias git-info='which git;git --version'
alias ls='ls --color=auto'
alias oldshell='sh'
```

`unalias` コマンドはエイリアスを削除します。たとえば、`unalias git_info` を実行して、リストから消えることを確認します。

```
$ unalias git-info
$ alias
alias ls='ls --color=auto'
alias oldshell='sh'
```

前のレッスンの `alias hi='echo We salute you.'` で見たように、（引数があるために）コマンドに空白が含まれている場合は、コマンド全体を引用符（シングルまたはダブル）で囲みます。

```
$ alias greet='echo Hello world!'
```

```
$ greet
Hello world!
```

オプションを含むコマンドも空白を含みますから、引用符で囲みます。

```
$ alias ll='ls -al'
```

`ll` は、隠しファイルを含むすべてのファイル（`-a`）を長い形式（`-l`）で一覧表示します。

エイリアス内で変数を参照できます。一重引用符（'）で囲むことに注意してください（後述）。

```
$ reptile=uromastyx
$ alias greet='echo Hello $reptile!'
$ greet
Hello uromastyx!
```

エイリアス内で、変数を割り当てることもできます。

```
$ alias greet='reptile=tortoise; echo Hello $reptile!'
$ greet
Hello tortoise!
```

`\` でエイリアスをエスケープできます。

```
$ alias where?='echo $PWD'
$ where?
/home/user2
$ \where?
-bash: where?: command not found
```

エイリアスと同名のコマンドがある場合に、エスケープが役に立ちます。同名のコマンドがある場合はエイリアスの方が優先されますが、エスケープすることでコマンドにアクセスできます。

エイリアスを別のエイリアス内に入れ子にできます。

```
$ where?
/home/user2
$ alias my_home=where?
$ my_home
/home/user2
```

また、後で示すように、エイリアス内に関数を置くこともできます。

エイリアス定義における引用符の働き

一重引用符の中で環境変数を使用すると、その展開はエイリアスの実行時に行われます（動的）。

```
$ alias where?='echo $PWD'
$ where?
/home/user2
$ cd Music
$ where?
/home/user2/Music
```

二重引用符の中で環境変数を使用すると、その展開はエイリアスの定義時に行われます（静的）。

```
$ alias where?="echo $PWD"
$ where?
/home/user2
$ cd Music
$ where?
/home/user2
```

エイリアスの永続化：起動スクリプト

変数の場合と同様に、エイリアスを永続化するには、起動時に読み込まれる初期化スクリプトにエイリアスの定義を置きます。すでに見てきたように、ユーザーの個人用エイリアスを入れるのに適したファイルは `~/.bashrc` です。多くの場合、既にいくつかのエイリアスがあります（ほとんどはコメントアウトされており、先頭の `#` を削除すると使用できるようになることが多いです）。

```
$ grep alias .bashrc
# enable color support of ls and also add handy aliases
    alias ls='ls --color=auto'
    #alias dir='dir --color=
    #alias vdir='vdir --color=
    #alias grep='grep --color=
    #alias fgrep='fgrep --color'
    #alias egrep='egrep --color=
# some more ls aliases
#ll='ls -al'
#alias la='ls -A'
#alias l='ls -CF'
# ~/.bash_aliases, instead of adding them here directly.
if [ -f ~/.bash_aliases ]; then
   . ~/.bash_aliases
fi
```

最後の3行では、独自のエイリアスファイル（`~/.bash_aliases`）が存在すれば、ログインするたびに `.bashrc` からそれが取り込まれます。つまり、エイリアスファイルを作成して、そこに記入しておけばよいわけです。

```
###########
# .bash_aliases:
# a file to be populated by the user's personal aliases (and sourced by ~/.bashrc).
###########
alias git_info='which git;git --version'
alias greet='echo Hello world!'
alias ll='ls -al'
alias where?='echo $PWD'
```

関数の作成

関数を使うと、Bash に備わっている特別な変数や位置パラメータを活用することができるので、エイリアスよりもプログラム的な柔軟性を得ることができます。また、条件判断や繰り返しループなどのフロー制御構造を利用することもできます。関数は、一連のコマンドやロジックを備えた1つのコマンドであると考えればよいでしょう。

関数を作成する2つの方法

関数を定義する、2つの構文（書き方）があります。

キーワード `function` を使用する

最初の方法では、キーワード `function` の後に関数の名前を置き、さらに中括弧で囲んで一連のコマンドを置きます。

```
function function_name {
command #1
command #2
command #3
.
.
.
command #n
}
```

括弧 () を使用する

もう一つの方法では、キーワード `function` を省略し、関数名の後に波括弧（{}）で囲んで一連のコマンドを置きます。

```
function_name() {
command #1
command #2
command #3
.
.
.
command #n
}
```

関数は（スクリプト）ファイルに置くのが普通ですが、1行ずつシェルプロンプトに直接入力することもできます。次の行を待っていることを示す PS2（>）に注意してください。

```
$ greet() {
> greeting="Hello world!"
> echo $greeting
> }
```

どの場合でも、コマンドをセミコロンで区切ることで、関数を1行で記述することができます。（最後のコマンドの後のセミコロンにも注意してください）。

```
$ greet() { greeting="Hello world!"; echo $greeting; }
```

Enterキーを押しても bash は文句を言わなかったので、関数を呼び出す準備ができました。関数を呼び出すには、その名前を入力します。

```
$ greet
Hello world!
```

変数やエイリアスの場合と同様に、システムの再起動後も関数を永続化する場合は、それらを /etc/bash.bashrc（グローバル）や ~/.bashrc（ローカル）などのシェル初期化スクリプトに置いておく必要があります。

WARNING エイリアスや関数をいずれかの初期化スクリプトファイルに追加した後に、再ログインしたりシステムを再起動したりしたくない場合は、変更を有効にするために . または source を使用して変更したファイルを取り込みます。

Bashの組み込み変数

Bourne Again Shell （bash)には、関数やスクリプトで役立つ特別な変数群があります。普通の変数とは異なり、これらの変数は参照のみが可能で変更できません。最も重要なものを以下に示します。

$?

この変数を参照すると、最後に実行されたコマンドの結果（exit値）が返されます。0 が成功を意味します。

```
$ ps aux | grep bash
user2      420  0.0  0.4  21156  5012 pts/0    Ss   17:10   0:00 -bash
user2      640  0.0  0.0  12784   936 pts/0    S+   18:04   0:00 grep bash
$ echo $?
0
```

0 以外の値はエラーを意味します。

```
user1@debian:~$ ps aux |rep bash
-bash: rep: command not found
```

```
user1@debian:~$ echo $?
127
```

$$

シェルのPID（プロセスID）を返します。

```
$ ps aux | grep bash
user2      420  0.0  0.4  21156  5012 pts/0    Ss   17:10   0:00 -bash
user2      640  0.0  0.0  12784   936 pts/0    S+   18:04   0:00 grep bash
$ echo $$
420
```

$!

直前に実行したバックグラウンドジョブのPIDを返します。

```
$ ps aux | grep bash &
[1] 663
$ user2      420  0.0  0.4  21156  5012 pts/0    Ss+  17:10   0:00 -bash
user2      663  0.0  0.0  12784   972 pts/0    S    18:08   0:00 grep bash
^C
[1]+  Done                    ps aux | grep bash
$ echo $!
663
```

NOTE アンパサンド（&）は、バックグラウンドでプロセスを開始します。

位置パラメータ $0 ～ $9

これらは、関数（ないしスクリプト）に渡されたパラメータ（引数）を返します ～ `$0` はスクリプトやシェル自体の名前になります。

位置パラメータを表示する関数を作成しましょう ～ 続く行の入力待ちを示す `PS2`（`>`）に注意してください。

```
$ special_vars() {
> echo $0
> echo $1
> echo $2
> echo $3
}
```

続いて、3つのパラメータ（`debian`、`ubuntu`、`zorin`）を渡して、関数（`special_vars`）を呼び出します。

```
$ special_vars debian ubuntu zorin
-bash
debian
ubuntu
```

```
zorin
```

期待どおりに機能しました。

WARNING

エイリアスに位置パラメータを埋め込んでもエラーにはなりませんが、エイリアスを展開した結果の後にパラメータが置かれるので、位置パラメータ変数への参照はまったく機能しません。

```
$ alias great_editor='echo $1 is a great text editor'
$ great_editor emacs
is a great text editor emacs
```

Bashには、他にも以下のような特別な組み込み変数があります。

`$#`
コマンドに渡された引数の数を示します。

`$@`、`$*`
これらは、コマンドに渡された引数全体を示します。

`$_`
最後のパラメータ、ないしはスクリプトの名前を示します（詳細は、`man bash` を調べてください！）

関数内の変数

もちろん、関数内でも変数を使用できます。

例を示すために、`funed` という名前の新しい空のファイルを作成して、次の関数を置きます。

```
editors() {
editor=emacs
echo "My editor is: $editor. $editor is a fun text editor."
}
```

前述の通り、関数を呼び出せるようにするために、ファイルを取り込む必要があります。

```
$ . funed
```

ではやってみましょう！

```
$ editors
My editor is emacs. emacs is a fun text editor.
```

`editors` 関数を正しく機能させるために、最初に `editor` 変数を設定しています。その変数のスコー

プ（有効範囲）は現在のシェルに対してローカルであり、セッションが続いている間はそれを参照できます。

```
$ echo $editor
emacs
```

ローカル変数と同様に、関数内で環境変数を使うこともできます。

```
editors() {
editor=emacs
echo "The text editor of $USER is: $editor."
}

editors
```

今度は、ファイルの最後で関数を呼び出している（最後の行の `editors`）ことに注意してください。これにより、ファイルを取り込むとすぐに関数が呼び出されます。

```
$ . funed
The text editor of user2 is: emacs.
```

関数における位置パラメータ

（関数にパラメータを渡せば）位置パラメータで参照することができます。

ファイルまたはスクリプトから、関数にパラメータを渡してみましょう（最後の行 `editors tortoise` に着目してください）：

```
editors() {
editor=emacs
echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}

editors tortoise
```

ファイルを取り込んで、動作を確認します。

```
$ . funed
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

コマンドラインから関数に位置パラメータを渡すこともできます。確認するために、ファイルの最後の行を削除します。

```
editors() {
editor=emacs
echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}
```

ファイルを取り込みます。

```
$ . funed
```

コマンドラインから、パラメータに `tortoise` を指定して関数を呼び出します。パラメータは `$1` に格納されます。

```
$ editors tortoise
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

スクリプト内の関数

関数は主にBashスクリプトに置かれます。

`funed` ファイルをスクリプト（`funed.sh` と名付けます）に変換することは実に簡単です。

```
#!/bin/bash

editors() {
editor=emacs
echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}

editors tortoise
```

以上！ 2行だけ追加しました：

- 最初の行は `shebang` （`シバン` または `シェバン`）と呼び、スクリプトを解釈するプログラムを定義します。奇異に思うかもしれませんが、ここでは `bash` そのもの（`#!/bin/bash`）です。
- 最後の行は単なる関数呼び出しです。

あとひとつ。スクリプトを実行可能にします。

```
$ chmod +x funed.sh
```

さぁ、準備ができました！

```
$ ./funed.sh
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

NOTE 次からのレッスンで、 シェルスクリプト について学びます。

エイリアス内の関数

前述のように、エイリアス内にも関数を置くことができます。

```
$ alias great_editor='gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }; gr8_ed'
```

この長いエイリアス値について説明しましょう。

- 最初に関数定義があります：`gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }`
- 関数の最後のコマンド〜 `unset -f gr8_ed` 〜は、関数呼び出しの最後に、`bash` セッションから関数 `gr8_ed` を削除します。
- 最後に、1回だけ関数 `gr8_ed` を呼び出します。

エイリアスを呼び出して、うまく動くことを確認します。

```
$ great_editor emacs
emacs is a great text editor
```

`unset -f gr8_ed` が示すように、`unset` コマンドは、変数を削除するだけでなく、関数を削除することもできます。実は、対象を明示するオプションがあります。

`unset -v`
: 変数を削除する

`unset -f`
: 関数を削除する

オプションなしの場合、`unset` は最初に変数の削除を試みて、失敗した場合に関数の削除を試みます。

関数内の関数

次に、`user2` がシステムにログインするたびに、2つのことを `user2` に伝えたいとしましょう。

- helloと言って、テキストエディタを推薦/賞賛する。
- user2は `$HOME/Video` フォルダにたくさんの `Matroska`（訳注: マルチメディアデータ格納ファイル形式のひとつ）ビデオファイルを入れ始めているので、警告を与える。

そのために、次の2つの関数を `/home/user2/.bashrc` に追加します。

最初の関数 `check_vids` は、`.mkv` ファイルのチェックと警告を行います。

```
check_vids() {
    ls -1 ~/Video/*.mkv > /dev/null 2>&1
    if [ "$?" = "0" ];then
        echo -e "Remember, you must not keep more than 5 video files in your Video folder.\nThanks."
    else
        echo -e "You do not have any videos in the Video folder. You can keep up to 5.\nThanks."
    fi
}
```

`check_vids` は3つのことを行います：

- `~/Video` 内の `mkv` ファイルをリストして、出力（とエラー）をいわゆる ビットバケツ（`/dev/null`）に送信する。（訳注：ビットバケツはデータの捨て場所の意）
- 前のコマンドが成功したかどうかを確認する。
- テストの結果に応じて、2つのメッセージのいずれかを出力する。

2つ目の関数は、`editors` 関数の変更版です。

```
editors() {
editor=emacs
echo "Hi, $USER!"
echo "$editor is more than a text editor!"

check_vids
}

editors
```

着目点が2つあります。

- `editors` の最後で `check_vids` を呼び出すので、両方の機能が連続して実行されます。挨拶と賞賛、チェックと警告がこの順に実行されます。
- 最後の `editors` は関数を呼び出すためのものです。

では、`user2` としてログインして、動作を確認しましょう。

```
# su - user2
Hi, user2!
emacs is more than a text editor!
Remember, you must not keep more than 5 video files in your Video folder.
Thanks.
```


演習

1. エイリアスと関数の機能を考えて、「はい」または「いいえ」で表を完成させてください。

機能	エイリアス？	関数？
ローカル変数を使用できる		
環境変数を使用できる		
`\` でエスケープできる		
再帰的になることがある		
位置パラメータとともに使用すると効果的である		

2. シェルセッション内のすべてのエイリアスを一覧表示するコマンドは何ですか？

3. `~/Music` 内のすべての `ogg` ファイルを1行に1つずつリストする `logg` という名前のエイリアスを書いて下さい。

4. `logg` エイリアスを呼び出して、その動作を確認して下さい。

5. 次に、エイリアスを変更して、リストの前にセッションのユーザーとコロンをエコー出力してください。

6. もう一度呼び出して、この新しいバージョンの動作を確認して下さい。

7. すべてのエイリアスを一覧表示して、一覧に `logg` エイリアスが表示されることを確認して下さい。

8. そのエイリアスを削除して下さい。

9. “エイリアス名” 列と “エイリアスコマンド” 列に応じて、それぞれのエイリアスを定義してください。

エイリアス名	エイリアスコマンド	エイリアスの割り当て
`b`	`bash`	
`bash_info`	`which bash` + `echo "$BASH_VERSION"`	

エイリアス名	エイリアスコマンド	エイリアスの割り当て
`kernel_info`	`uname -r`	
`greet`	`echo Hi, $USER!`	
`computer`	`pc=slimbook + echo My computer is a $pc`	

10. `root` として、`/etc/bash.bashrc` に `my_fun` という関数を定義して下さい。関数はユーザーに挨拶し、PATHが何であるかをユーザーに伝えるものとします。ユーザーがログインするたびに2つのメッセージを受け取るように呼び出して下さい。

11. `user2` としてログインして、動作を確認して下さい。

12. 同じ関数を1行で記述して下さい。

13. 関数を呼び出して下さい。

14. 関数を削除して下さい。

15. 以下に `special_vars` 関数の修正バージョンを示します：

```
$ special_vars2() {
> echo $#
> echo $_
> echo $1
> echo $4
> echo $6
> echo $7
> echo $_
> echo $@
> echo $?
> }
```

次のコマンドで関数を呼び出します：

```
$ special_vars2 crying cockles and mussels alive alive oh
```

何が出力されますか？

コマンド	値
`echo $#`	
`echo $_`	
`echo $1`	
`echo $4`	
`echo $6`	
`echo $7`	
`echo $_`	
`echo $@`	
`echo $?`	

16. 「**関数内の関数**」セクションのサンプル関数（`check_vids`）を改造して、bash の起動スクリプトに含めるための関数 `check_music` を作って下さい。挙動を簡単に変更できるように、次の項目を（次の順序で）パラメータとして指定するものとします。
    - ファイルを保存するディレクトリ： `~/Music`
    - チェックするファイル種別： `ogg`
    - 保存できるファイルの数： `7`
    - ファイルの内容を示す名前： `music`

発展演習

1. 読み取り専用関数とは、内容を変更できない関数です。読み取り専用関数　　について調べて、次の表の空欄を埋めてください。

関数名	読み取り専用にするコマンド	すべての読み取り専用関数を一覧表示するコマンド
`my_fun`		

2. `PS1`　　を変更する方法についてWebを調べて、ユーザーに次の情報を提供して、`PS1`　　を指定のものに変更する、`fyi` という（起動スクリプトに置く）関数を作成して下さい。
   - ユーザーの名前
   - ホームディレクトリ
   - ホストの名前
   - オペレーティングシステムの種別
   - 実行可能ファイルの検索パス
   - メールディレクトリ
   - メールをチェックする頻度
   - 現在のセッションのシェルの深さ（ネスト数）
   - プロンプトを `<user>@<host-date>` に変更する

まとめ

このレッスンでは、以下の事柄を学びました:

- エイリアスと関数はどちらもシェルの重要な機能であり、繰り返し使われるコードをカプセル化できること。
- エイリアスは、長いコマンドや複雑なコマンドを短い入力で使用する場合に便利なこと。
- 関数はロジックを実装したもので、主にスクリプトでタスクを自動化するために使われること。
- エイリアスと関数を記述するための構文。
- セミコロン（`;`）を使用して複数のコマンドを連結する方法。
- エイリアスで適切な引用符を使用する方法。
- エイリアスや関数を永続化する方法。
- Bashの特別な組み込み変数：`$?`、`$$`、`$!`、位置パラメータ（`$0` ～ `$9`）、`$#`、`$@`、`$*`、`$_`。
- 関数で変数や位置パラメータを使用する方法。
- スクリプトで関数を使用する方法。
- エイリアスから関数を呼び出す方法。
- 関数から別の関数を呼び出す方法。
- `bash` スクリプトを作成する基礎。

このレッスンで取り上げたコマンドとキーワード：

`alias`
エイリアスを作成します。

`unalias`
エイリアスを削除します。

`cd`
作業ディレクトリを変更します。

`grep`
パターンに一致する行を出力する。

`function`
関数を作成するためのキーワード。

`.`
ファイルを読み込みます。

`source`
ファイルを読み込みます。

`ps`
現在のプロセスの状態をレポートします。

`echo`
テキストを表示します。

`chmod`
ファイルのモードビットを変更します。たとえば、ファイルを実行可能にするなど。

`unset`
変数や関数の定義を削除します。

`su`
ユーザーIDを変更する、ないし、スーパーユーザーになる。

演習の解答

1. エイリアスと関数の機能を考えて、「はい」または「いいえ」で表を完成させてください。

機能	エイリアス？	関数？
ローカル変数を使用できる	Yes	Yes
環境変数を使用できる	Yes	Yes
`\` でエスケープできる	Yes	No
再帰的になることがある	Yes	Yes
位置パラメータとともに使用すると効果的である	No	Yes

2. シェルセッション内のすべてのエイリアスを一覧表示するコマンドは何ですか？

```
alias
```

3. `~/Music` 内のすべての `ogg` ファイルを1行に1つずつリストする `logg` という名前のエイリアスを書いて下さい。

```
alias logg='ls -1 ~/Music/*ogg'
```

4. エイリアスを呼び出して、その機能を確認して下さい。

```
logg
```

5. 次に、エイリアスを変更して、リストの前にセッションのユーザーとコロンをエコー出力してください。

```
alias logg='echo $USER:; ls -1 ~/Music/*ogg'
```

6. もう一度呼び出して、この新しいバージョンの動作を確認して下さい。

```
logg
```

7. すべてのエイリアスを再び一覧表示して、logg エイリアスが一覧に表示されることを確認して下さい。

```
alias
```

8. そのエイリアスを削除して下さい。

```
unalias logg
```

9. “エイリアス名” 列と “エイリアスコマンド” 列に応じて、それぞれのエイリアスを定義してください。

エイリアス名	エイリアスコマンド	エイリアスの割り当て
`b`	`bash`	`alias b=bash`
`bash_info`	`which bash` + `echo "$BASH_VERSION"`	`alias bash_info='which bash; echo "$BASH_VERSION"'`
`kernel_info`	`uname -r`	`alias kernel_info='uname -r'`
`greet`	`echo Hi, $USER!`	`alias greet='echo Hi, $USER'`
`computer`	`pc=slimbook` + `echo My computer is a $pc`	`alias computer='pc=slimbook; echo My computer is a $pc'`

10. `root` として、`/etc/bash.bashrc` に `my_fun` という関数を定義して下さい。関数はユーザーに挨拶し、PATHが何であるかをユーザーに伝えるものとします。ユーザーがログインするたびに2つのメッセージを受け取るように呼び出して下さい。

    A案:

```
my_fun() {
echo Hello, $USER!
echo Your path is: $PATH
}
my_fun
```

    B案:

```
function my_fun {
echo Hello, $USER!
echo Your path is: $PATH
}
my_fun
```

11. `user2` としてログインして、動作を確認して下さい。

```
su - user2
```

12. 同じ関数を1行で記述して下さい。

    A案:

```
my_fun() { echo "Hello, $USER!"; echo "Your path is: $PATH"; }
```

B案:

```
function my_fun { echo "Hello, $USER!"; echo "Your path is: $PATH"; }
```

13. 関数を呼び出して下さい。

```
my_fun
```

14. 関数を削除して下さい。

```
unset -f my_fun
```

15. 以下に `special_vars` 関数の修正バージョンを示します：

```
$ special_vars2() {
> echo $#
> echo $_
> echo $1
> echo $4
> echo $6
> echo $7
> echo $_
> echo $@
> echo $?
> }
```

次のコマンドで関数を呼び出します：

```
$ special_vars2 crying cockles and mussels alive alive oh
```

何が出力されますか？

コマンド	値
`echo $#`	`7`
`echo $_`	`7`
`echo $1`	`crying`
`echo $4`	`mussels`
`echo $6`	`alive`

コマンド	値
`echo $7`	`oh`
`echo $_`	`oh`
`echo $@`	`crying cockles and mussels alive alive oh`
`echo $?`	`0`

16. 「関数内の関数」セクションのサンプル関数（`check_vids`）を改造して、bash の起動スクリプトに含めるための関数 `check_music` を作って下さい。挙動を簡単に変更できるように、次の項目を（次の順序で）パラメータとして指定するものとします。
    - ファイルを保存するディレクトリ： `~/Music`
    - チェックするファイル種別： `ogg`
    - 保存できるファイルの数： `7`
    - ファイルの内容を示す名前： `music`

```
check_music() {
    ls -1 ~/$1/*.$2 > ~/.mkv.log 2>&1
    if [ "$?" = "0" ];then
        echo -e "Remember, you must not keep more than $3 $4 files in your $1 folder.\nThanks."
    else
        echo -e "You do not have any $4 files in the $1 folder. You can keep up to $3.\nThanks."
    fi
}

check_music Music ogg 7 music
```

発展演習の解答

1. 読み取り専用関数とは、内容を変更できない関数です。読み取り専用関数　について調べて、次の表の空欄を埋めてください。

関数名	読み取り専用にするコマンド	すべての読み取り専用関数を一覧表示するコマンド
`my_fun`	`readonly -f my_fun`	`readonly -f`

2. `PS1`　を変更する方法についてWebを調べ、ユーザーに次の情報を提供して　`PS1`　を指定のものに変更する、`fyi` という（起動スクリプトに置く）関数を作成して下さい。
   - ユーザーの名前
   - ホームディレクトリ
   - ホストの名前
   - オペレーティングシステムの種類
   - 実行可能ファイルの検索パス
   - メールディレクトリ
   - メールがチェックされる頻度
   - 現在のセッションのシェルの深さ（ネスト数）
   - プロンプトを `<user>@<host-date>` に変更する

```
fyi() {
    echo -e "For your Information:\n
    Username: $USER
    Home directory: $HOME
    Host: $HOSTNAME
    Operating System: $OSTYPE
    Path for executable files: $PATH
    Your mail directory is $MAIL and is searched every $MAILCHECK seconds.
    The current level of your shell is: $SHLVL"
    PS1="\u@\h-\d "
}

fyi
```

105.2 簡単なスクリプトをカスタマイズする

LPI目標への参照

LPIC-1 5.0, Exam 102, Objective 105.2

総重量

4

主な知識分野

- 標準のsh構文を使用する(ループ、テスト)。
- コマンド置換を使用する。
- 成功または失敗の戻り値、またはコマンドによって提供されるその他の情報をテストする。
- 連続したコマンドを実行する。
- スーパーユーザーに条件付きメーリングを実行する。
- シバン(#!)行を使ってスクリプトインタープリタを正しく選択する。
- スクリプトの場所、所有権、実行、suid-rightsを管理する。

用語とユーティリティ

- `for`
- `while`
- `test`
- `if`
- `read`
- `seq`
- `exec`
- `||`
- `&&`

105.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	105 シェルとシェルスクリプト
Objective:	105.2 簡単なスクリプトの作成とカスタマイズ
Lesson:	1 of 2

はじめに

Linuxシェル環境では、シェル組み込みコマンドと、さまざまなプログラムが提供する　スクリプト　と呼ばれるコマンドを含むファイルを組み合わせて、ユーザーやシステム固有のタスクを自動化できます。実際に、オペレーティングシステムの保守作業の多くは、一連のコマンド、条件判断、条件ループなどから成るスクリプトで実行されます。オペレーティングシステム自体にかかわるタスクを行うスクリプトが多いですが、スクリプトを用いてファイルの名前変更を一括して行ったり、データの収集や解析を行ったり、あるいはコマンドを連続して実行するといった、ユーザー自身のタスクを行うことにも役立ちます。

スクリプトはテキストファイルにすぎませんが、プログラムのように動作します。本体のプログラム（インタプリタ）は、スクリプトファイルに書かれている命令を読み取って実行します。あるいは、Linuxシェルセッションと同様に（スクリプトを含む）コマンドを入力するたびに実行する対話型セッションで、インタプリタを利用することもできます。　エイリアスやシェル関数として実装できないような複雑な処理を行いたい場合に、命令とコマンドをグループ化するためにスクリプトファイルを使用します。さらに、スクリプトファイルは単なるテキストファイルであるため、テキストエディタのみで作成・変更できますから、容易にメンテナンスすることができます。

スクリプトの構造と実行方法

スクリプトファイルとは、対応するインタープリタによって実行されるコマンドの並びを内容とするファイルです。インタープリタがスクリプトを読み取る方法はさまざまですが、シェルセッションとはまったく異なります。スクリプトファイルの先頭には文字　`#!`　（シェバン　と読みます）が置かれていて、その後に続く文字列がインタープリタです。Bashスクリプトの場合は、最初の行が　`#!/bin/bash`　になり、続くすべてのコマンドは　`/bin/bash`　によって解釈されます。ハッシュ文字　`#`　で始まる行は無視されるので、コメントや覚え書きを置くために使用します。空白行も無視されます。最も簡単なシ

ェルスクリプトは次のようになります:

```
#!/bin/bash

# A very simple script
echo "Cheers from the script file! Current time is: "
date +%H:%M
```

このスクリプトには、`/bin/bash` によって解釈される2つの命令のみがあります。組み込みコマンド `echo` と、通常コマンド `date` です。スクリプトファイルを実行する最も基本的な方法は、スクリプトファイルのパスを引数としてインタプリタを実行することです。つまり、この例が現在のディレクトリの `script.sh` という名前のファイルに保管されている場合は、次のコマンドで実行します:

```
$ bash script.sh
Cheers from the script file! Current time is:
10:57
```

`echo` コマンドは文字列を表示した後に改行を出力しますが、オプション `-n` でこの動作を抑止できます。つまり、スクリプトで `echo -n` を使用すると、両方のコマンドの出力が同じ行に表示されます。

```
$ bash script.sh
Cheers from the script file! Current time is: 10:57
```

必須ではありませんが、ファイル名にサフィックス `.sh` を付けておくと、一覧や検索の際にシェルスクリプトを識別しやすくなります。

TIP

シェルは、`#!` の後に示されているコマンドをスクリプトファイルのインタプリタとして呼び出します。たとえば、シェバンを使用して、Python（`#!/usr/bin/python`）、Perl（`#!/usr/bin/perl`）、awk（`#!/usr/bin/awk`）など、他のスクリプト言語を呼び出すことができます。

他のユーザーにスクリプトを実行させることがある場合は、適切なパーミッションが設定されていることが重要です。コマンド `chmod o+r script.sh` は、システム内のすべてのユーザーに読み取り権限を与えて、コマンド `bash` の引数にスクリプトファイルのパス名を指定することで `script.sh` を実行できるようにします。あるいは、スクリプトファイルに実行パーミッションをセットして、通常のコマンドのようにファイルを実行できるようにします。スクリプトに実行パーミッションをセットするには、コマンド `chmod` を使用します。

```
$ chmod +x script.sh
```

実行パーミッションを有効にすると、現在のディレクトリにある `script.sh` という名前のスクリプトファイルを、コマンド `./script.sh` で直接実行できます。`PATH` 環境変数にリストされているディレクトリにスクリプトファイルを置けば、パスがなくても実行できます。

WARNING

スクリプトファイルに SetUID や SetGID のビットをセットしても、期待するように実行権限を変更する事はできません。起動されるプログラムはインタプリタ

> であり、スクリプトファイルではないからです。

シェルスクリプトのファイルでは、コマンドのインデントや書き方はそれほど厳密ではありません。スクリプトのすべての行は、先頭行から順に、普通のシェルコマンドとして実行されます。また、コマンドラインと同じルールが行毎に適用されます。例えば、2つ以上のコマンドをセミコロンで区切って同じ行に続けることができます。

```
echo "Cheers from the script file! Current time is:" ; date +%H:%M
```

この形式が便利な場合もありますが、1行に1つずつコマンドを置けばセミコロンで区切ったのと同じように続けて実行されるので、ほとんど使われません。つまり、Bashスクリプトファイルではセミコロンを改行文字に置き換えることができるのです。

スクリプトが実行されると、そこに含まれるコマンドは現在のセッションで直接実行されるのではなく、サブシェル と呼ばれる新しいBashプロセスによって実行されます。それにより、スクリプトが現在のセッションの環境変数を上書きしたり、現在のセッションに意図しない変更を残したりするのを防ぎます。現在のシェルセッションでスクリプトの内容を実行したい場合は、`source script.sh` ないし `. script.sh`（ドットとスクリプト名の間にスペースがあることに注意してください）を使って実行します。

通常のコマンド実行と同様に、スクリプトが実行を終えるまで、シェルプロンプトには戻りません。そして、その終了ステータスコードを `$?` 変数で参照することができるようになります。この動作を変更して、現在のシェルでスクリプトを実行したい場合は、スクリプト（ないしその他のコマンド）の前に `exec` コマンドを置きます。こうすると、スクリプトの実行が終了すると、現在のシェルが終了しますから、ログインシェルから実行した場合はログアウトしてしまうことになります。

変数

シェルスクリプトの変数は、対話セッションと同じように機能します。たとえば、`SOLUTION=42`（等号の前後にスペースを入れない）と書くと、`SOLUTION` という名前の変数に値 `42` を割り当てます。変数名には大文字を使用することが慣例ですが、必須ではありません。なお、変数名をアルファベット以外の文字で始めることはできません。（訳注: 日本語は使用できません。）

Bashスクリプトでは、ユーザーが定義する通常の変数だけではなく、特殊変数 と呼ばれる特別な変数も使えます。通常の変数とは異なり、特殊変数の名前は記号1文字ないし数字です。スクリプトに渡される引数やその他の有用な情報が、`0`、`*`、`?` などの特殊変数に格納されます。ドルマークに続けてこれら変数の値を参照すると、以下の情報が返されます:

`$*`
: スクリプトに渡されたすべての引数（すべての引数を連結した1つの文字列）

`$@`
: スクリプトに渡されたすべての引数。`"$@"` のように二重引用符内で使用すると、それぞれの引数が二重引用符で囲まれます

`$#`
: 引数の数

`$0`
: スクリプトファイルの名前

$!
最後に実行されたプログラムのPID。

$$
現在のシェルのPID

$?
最後に終了したコマンドの終了ステータスコードを示す数値。POSIX標準プロセスの場合、数値 `0` は、最後のコマンドが正常に実行されたことを意味します。これは、シェルスクリプトでも同様です。

位置変数 は、`0` 以外の1桁以上で示されるパラメータです。たとえば、変数 `$1` はスクリプトに指定された最初の引数に対応し、`$2` は2番目の引数に対応します。パラメータの位置が9より大きい場合は、`${10}`、`${11}` などのように、波括弧（中括弧）で囲みます。

一方、通常の変数には、スクリプト内でセットした値や、他のコマンドの出力を格納します。たとえば、コマンド `read` をスクリプト内で使用すると、スクリプトの実行中にユーザーに入力を求めることができます。

```
echo "Do you want to continue (y/n)?"
read ANSWER
```

入力された値が変数 `ANSWER` に格納されます。変数名が指定されていない場合は、デフォルトで変数 `REPLY` が使用されます。コマンド `read` を使用して、複数の変数を同時に読み取ることもできます。

```
echo "Type your first name and last name:"
read NAME SURNAME
```

この例では、変数 `NAME` と `SURNAME` に、空白で区切られた単語のそれぞれが割り当てられます。入力された単語の数が変数の数よりも多い場合、残りの単語は最後の変数に（空白で区切られて）格納されます。`read` のオプション `-p` を使用してユーザーに表示するメッセージを指定できるので、この例の `echo` コマンドは冗長です。

```
read -p "Type your first name and last name:" NAME SURNAME
```

システムタスクを実行するスクリプトは、多くの場合、他のプログラムによって提供される情報を利用します。バッククオート を使うと、コマンドの出力を変数に格納することができます。

```
$ OS=`uname -o`
```

この例では、コマンド `uname -o` の出力が、変数 `OS` に格納されます。`$()` でも同じ結果が得られます。

```
$ OS=$(uname -o)
```

変数名の前にハッシュ `#` を付加すると、変数の長さ、つまり変数に含まれる文字数が返されます。ただし、この機能を使うときは、変数を明示するために波括弧を使用します。

```
$ OS=$(uname -o)
$ echo $OS
GNU/Linux
$ echo ${#OS}
9
```

Bashでは1次元配列を格納する変数も使えるので、関連する複数の要素を1つの変数に格納できます。配列の各要素は数値のインデックスを使って読み書きします。通常の変数とは異なり、配列はBashの組み込みコマンド `declare` で宣言する必要があります。たとえば、`SIZES` という名前の変数を配列として宣言するにはこうします:

```
$ declare -a SIZES
```

括弧を使った表記法で項目のリストを定義すると、配列を暗黙的に宣言することもできます。

```
$ SIZES=( 1048576 1073741824 )
```

この例では、2つの大きな整数値が `SIZES` 配列に格納されます。配列要素を参照するには、波括弧と角括弧を使用します。配列のインデックスは0から始まるため、最初の要素の内容は `${SIZES[0]}` に、2番目の要素は `${SIZES[1]}` にあります。

```
$ echo ${SIZES[0]}
1048576
$ echo ${SIZES[1]}
1073741824
```

配列要素への代入は、読み取りとは異なり、波括弧なしで指定します（たとえば、`SIZES[0]=1048576`）。通常の変数と同様に、配列内の要素の長さをハッシュ文字で参照できます（たとえば、`SIZES` 配列の最初の要素の長さは `${#SIZES[0]}` となり、この例では7です）。インデックスに `@` または `*` を指定すると、配列内の要素数が返されます。

```
$ echo ${#SIZES[@]}
2
$ echo ${#SIZES[*]}
2
```

また、コマンド置換を利用することで、コマンドの出力を初期要素とする配列を宣言することもできます。次の例は、現在のシステムでサポートされているファイルシステムを要素とする配列を作成する方法を示しています:

```
$ FS=( $(cut -f 2 < /proc/filesystems) )
```

コマンド `cut -f 2 < /proc/filesystems` は、（`/proc/filesystems` ファイルの2列目の列にリストされている）実行中のカーネルで現在サポートされているすべてのファイルシステムを出力します。そのため、`FS` 配列には、それぞれの要素に1つのファイルシステムが含まれます。デフォルトでは、空白文字（空白、タブ、改行）で区切られた単語が配列要素になりますので、さまざまな形式のテキストを使用して配列を初期化できます。

TIP Bashでは、変数 `$IFS`（Input Field Separator）の文字それぞれを区切り文字として扱います。たとえば、フィールド区切り文字を改行文字のみに変更するには、コマンド `IFS=$'\n'` でIFS変数を変更します。

算術式

Bashでは、組み込みコマンド `expr` を使用して、実用的な整数算術演算を行えます。たとえば、`$VAL1` と `$VAL2` の2つの数値変数を足すには、次のコマンドを使います。

```
$ SUM=`expr $VAL1 + $VAL2`
```

例では、結果の値が `$SUM` 変数に格納されます。コマンド `expr` は `$(())` に置き換えることができるので、前の例は `SUM=$(( $VAL1 + $VAL2 ))` と書き直すことができます。二重アスタリスク演算子は累乗を表すので、前の配列宣言 `SIZES=( 1048576 1073741824)` は `SIZES=( $((1024**2)) $((1024**3)) )` と書くことができます。

算術式でもコマンド置換が使えます。ファイル `/proc/meminfo` には、RAMの空きバイト数などシステムメモリに関する詳細情報が含まれていますので例に取り上げましょう:

```
$ FREE=$(( 1000 * `sed -nre '2s/[^[:digit:]]//gp' < /proc/meminfo` ))
```

この例では、算術式の内部で `sed` コマンドを使い、`/proc/meminfo` の内容を解析しています。`/proc/meminfo` ファイルの2列目には、キロバイト単位の空きメモリ量がありますから、算術式でその数値に1000を掛けて、RAMの空きバイト数を取得します。（訳注: sedで2行目（MemFree）だけを取り出し、数値以外の文字を取り除いていることに注意してください。）

条件付き実行

スクリプトの中には、スクリプトファイル内のすべてのコマンドを実行するのではなく、あらかじめ定義した条件に一致するコマンドのみを実行するものがあります。たとえば、保守用のスクリプトでは、コマンドの実行に失敗した場合にのみ、管理者にメールで警告メッセージを送信します。Bashにはコマンド実行の成否を評価することに限定した方法と、一般のプログラミング言語に見られるような汎用的な条件実行の方法が用意されています。

コマンドを `&&` で区切ると、左側のコマンドでエラーが発生しなかった場合、つまり終了ステータスが `0` の場合にのみ、右側のコマンドが実行されます。（訳注: `&&` は AND を示す条件演算子です。1つでも失敗するコマンドが現れた時点で結果が False に決まりますから、後続のコマンドを実行する必要が無くなる、というワケです。）

```
COMMAND A && COMMAND B && COMMAND C
```


コマンドを `||` で区切ると、動作が逆転します。すなわち、左側のコマンドでエラーが発生した場合、つまり終了ステータスが0ではない場合にのみ、右側のコマンドが実行されます。（訳注: `||` は OR を示す条件演算子です。1つでも成功するコマンドが現れた時点で結果が TRUE に決まりますから、後続のコマンドを実行する必要が無くなる、と言うワケです。）

すべてのプログラミング言語において、あらかじめ定義した条件に応じてコマンドを実行することは、最も重要な機能の一つです。条件に応じてコマンドを実行する最も簡単な方法は、組み込みコマンド `if` を使うことです。これは、引数に与えられたコマンドがステータスコードに0（成功）を返す場合にのみ、ブロック内の1つないしは複数のコマンドを実行します。さまざまな条件を評価することができる `test` コマンドが用意されていて、ほとんどの場合は `if` と `test` を組み合わせて使用します。次の例では、ファイル `/bin/bash` が存在して実行できる場合に `Confirmed: /bin/bash is executable.` というメッセージを表示します。

```
if test -x /bin/bash ; then
  echo "Confirmed: /bin/bash is executable."
fi
```

`test` コマンドに `-x` オプションを指定すると、与えられたパスが実行可能なファイルである場合にのみ、ステータスコード 0 を返します。次の例では、`test` の代わりに角括弧を使い、まったく同じ結果を得る別の方法を示しています。（訳注: セミコロンの位置に注意してください。] までが、[コマンドの引数になります。[コマンドは、if ブロックを見やすく書くために用意された、`test` コマンドの特別な呼び出し方法です。）

```
if [ -x /bin/bash ] ; then
  echo "Confirmed: /bin/bash is executable."
fi
```

`else` コマンドは `if` ブロックのオプションであり、存在する場合は、条件式が真でない場合に実行する一連のコマンドを定義できます。

```
if [ -x /bin/bash ] ; then
  echo "Confirmed: /bin/bash is executable."
else
  echo "No, /bin/bash is not executable."
fi
```

Bashインタープリタが条件ブロックの終了を判断できるように、`if` ブロックは常に `fi` で終わる必要があります。

スクリプトからの出力

ファイル操作のみを行う含むスクリプトであっても、進捗に関するメッセージを標準出力に表示することは重要です。ユーザーに問題を知らせられますし、そのメッセージをログとして残すこともできます。

Bashの組み込みコマンド `echo` は、単純な文字列を表示するためによく使われますが、いくつかの拡張機能も備えています。`-e` オプションを指定すると、エスケープシーケンス (バックスラッシュと1文

字) を使って特殊文字を表示できます。例えば、以下のようになります:

```
#!/bin/bash

# Get the operating system's generic name
OS=$(uname -o)

# Get the amount of free memory in bytes
FREE=$(( 1000 * `sed -nre '2s/[^[:digit:]]//gp' < /proc/meminfo` ))

echo -e "Operating system:\t$OS"
echo -e "Unallocated RAM:\t$(( $FREE / 1024**2 )) MB"
```

オプションなしの `echo` では引用符を使わなくても構いませんが、オプション `-e` を使用する場合は引用符で囲む必要があります。前のスクリプトでは、両方の `echo` コマンドでテキストの位置を合わせるためにタブ文字 `\t` を使用して、次のような出力にしています:

```
Operating system:       GNU/Linux
Unallocated RAM:        1491 MB
```

改行文字 `\n` を使用すると出力行を区切ることができるので、2つの `echo` コマンドを1つに結合してもまったく同じ出力が得られます。

```
echo -e "Operating system:\t$OS\nUnallocated RAM:\t$(( $FREE / 1024**2 )) MB"
```

ほとんどのテキストメッセージの表示は `echo` コマンドで充分ですが、厳密な書式が必要となる場合があります。Bashの組み込みコマンド `printf` を使用すると、変数の表示方法をより細かく制御できます。`printf` コマンドでは、最初の引数に出力書式を指定します。コマンドラインに指定された順序で、プレースホルダが引数に置き換えられます。たとえば、前の例のメッセージを、次の `printf` コマンドで生成できます。

```
printf "Operating system:\t%s\nUnallocated RAM:\t%d MB\n" $OS $(( $FREE / 1024**2 ))
```

テキスト用のプレースホルダ `%s` が `$OS` 変数の内容で置き換えられ、整数用の `%d` プレースホルダが RAM内の空き容量（メガバイト単位）に置き換えられます。`printf` はテキストの最後に改行文字を追加しないので、必要に応じて改行文字 `\n` を出力書式の最後に置く必要があります。パターン全体を単一の引数として解釈する必要があるので、引用符で囲みます。

TIP `printf` に使えるプレースホルダは、`printf` コマンドのマニュアルに掲載されています。`man printf` で参照してください。

`printf` では、書式を示す文字列と、出力に埋め込む変数が独立しています。すなわち、書式を示す文字列を別の変数に格納することもできます。

```
MSG='Operating system:\t%s\nUnallocated RAM:\t%d MB\n'
```

```
printf "$MSG" $OS $(( $FREE / 1024**2 ))
```

この方法は、ユーザーの要件に応じて出力形式を変更する場合に特に便利です。たとえば、何種類かのCSV（カンマ区切り）リストを出力する場合に、種類ごとに書式文字列を切り替えて表示するスクリプトを簡単に作成できます。

演習

1. `read` コマンドの `-s` オプションは、入力された内容を画面に表示しないので、パスワードを入力するときに便利です。`read` コマンドで、入力された内容を見せずに、その値を `PASSWORD` 変数に格納するにはどうしますか？

2. `whoami` は、コマンドを呼び出したユーザーのユーザー名を表示するコマンドで、スクリプト内でコマンドを実行しているユーザーを識別するためよく使われます。Bashスクリプト内で、`whoami` コマンドの出力を `WHO` という名前の変数に格納するにはどうしますか？

3. `apt-get dist-upgrade` が成功したときだけ `systemctl reboot` を実行したい場合に、`apt-get dist-upgrade` と `systemctl reboot` コマンドの間に置くオペレータは何ですか？

発展演習

1. 新しく作成したBashスクリプトを実行しようとしたら、次のエラーメッセージが表示されました:

   ```
   bash: ./script.sh: Permission denied
   ```

   ファイル `./script.sh` が自作のものである場合、このエラーの原因として考えられることは何ですか？

2. `do.sh` という名前のスクリプトファイルが実行可能であり、`undo.sh` という名前のシンボリックリンクがそれを指しているとします。スクリプト内から、呼び出したファイル名が `do.sh` であるか `undo.sh` であるかを識別するにはどうしますか？

3. メールサービスが適切に構成されているシステムでは、次のコマンドで通知メッセージをroot宛に送信できます:

   `mail -s "Maintenance Error" root <<<"Scheduled task error"`

   cronジョブ などの自動化タスクでは、このようなコマンドでエラーをシステム管理者に通知することがよくあります。（それが何であれ）コマンドの終了ステータスが失敗であった場合に、この `mail` コマンドを実行する if ブロックを記述してください。

まとめ

このレッスンでは、Bashのシェルスクリプトを理解し、記述するための基本的な概念について説明しました。シェルスクリプトは、あらゆるLinuxディストリビューションの中核であり、システム管理やユーザーのタスクを自動化するための非常に柔軟な方法を提供しています。レッスンでは、以下のテーマを取り上げました:

- シェルスクリプトの構造と、スクリプトファイルのパーミッション
- スクリプトにおける引数
- 変数を使用したユーザー入力の読み取りと、コマンド出力の利用
- Bashの配列
- 簡単なテストと条件実行
- 出力の書式指定

以下のコマンドと手順を紹介しました:

- コマンド置換、配列の展開、算術式
- `||` および `&&` 演算子を使用した条件付きコマンド実行
- `echo`
- `chmod`
- `exec`
- `read`
- `declare`
- `test`
- `if`
- `printf`

演習の解答

1. `read` コマンドの `-s` オプションは、入力された内容を画面に表示しないので、パスワードを入力するときに便利です。`read` コマンドで、入力された内容を見せずに、その値を `PASSWORD` 変数に格納するにはどうしますか？

   `read -s PASSWORD`

2. `whoami` は、コマンドを呼び出したユーザーのユーザー名を表示するコマンドで、スクリプト内でコマンドを実行しているユーザーを識別するためよく使われます。Bashスクリプト内で、`whoami` コマンドの出力を `WHO` という名前の変数に格納するにはどうしますか？

   `WHO=\`whoami\`` または `WHO=$(whoami)`

3. `apt-get dist-upgrade` が成功したときだけ `systemctl reboot` を実行したい場合に、`apt-get dist-upgrade` と `systemctl reboot` コマンドの間に置くオペレータは何ですか？

   `&&` オペレータ。`apt-get dist-upgrade && systemctl reboot`

発展演習の解答

1. 新しく作成したBashスクリプトを実行しようとしたら、次のエラーメッセージが表示されました:

   ```
   bash: ./script.sh: Permission denied
   ```

   ファイル `./script.sh` が自作のものである場合、このエラーの原因として考えられることは何ですか？

   `./script.sh` ファイルに実行権限が付いていません。

2. `do.sh` という名前のスクリプトファイルが実行可能であり、`undo.sh` という名前のシンボリックリンクがそれを指しているとします。スクリプト内から、呼び出したファイル名が `do.sh` であるか `undo.sh` であるかを識別するにはどうしますか？

   位置変数 `$0` に、スクリプトの呼び出しに使用されたファイル名が格納されています。

3. メールサービスが適切に構成されているシステムでは、次のコマンドで通知メッセージをroot宛に送信できます:

   `mail -s "Maintenance Error" root <<<"Scheduled task error"`

   cronジョブ などの自動化タスクでは、このようなコマンドでエラーをシステム管理者に通知することがよくあります。（それが何であれ）コマンドの終了ステータスが失敗であった場合に、この `mail` コマンドを実行する if ブロックを記述してください。

   `if [ "$?" -ne 0 ]; then mail -s "Maintenance Error" root <<<"Scheduled task error"; fi`

105.2 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	105 シェルとシェルスクリプト
Objective:	105.2 簡単なスクリプトをカスタマイズまたは作成する
Lesson:	2 of 2

はじめに

シェルスクリプトは通常、ファイルやディレクトリに関連する操作を自動化することを目的としています。これは、コマンドラインで手動で実行できる操作と同じです。とはいえ、シェルスクリプトでは、Linuxオペレーティングシステムの多くの機能や構成情報を利用することが多く、ユーザーの文書ファイルだけを操作するとは限りません。

シェルスクリプトを作成するために、多くBash組み込みコマンドだけではなく、Linuxシステムに備わっている多くのコマンドラインユーティリティを組み合わせて使用する事もできます。

test コマンド

スクリプト言語としてのBashはファイルを処理することが多いので、Bashの組み込みコマンド `test` には、ファイルシステムオブジェクト（基本的にはファイルとディレクトリ）のプロパティを評価するための多くのオプションがあります。ファイルとディレクトリに焦点を当てたテストでは、たとえば、あるタスクを実行するために必要なファイルとディレクトリが存在していて、読み取り可能であることを確認します。結果を `if` 文で評価して、テストが成功した場合には、適切な一群のアクションが実行されます。

`test` コマンドでは、2つの異なる構文で式を評価できます。つまり、`test` コマンドの引数として条件式を指定する方法と、角括弧内に条件式を置く方法があります。この場合 `test` というコマンド名は使いません。例えば、`/etc` が有効なディレクトリであるかどうかを判定する条件式は、`test -d /etc` ないしは `[ -d /etc ]` と書くことができます。（訳注: `[` の後と `]` の前に空白が必要です。）

```
$ test -d /etc
```

```
$ echo $?
0
$ [ -d /etc ]
$ echo $?
0
```

特殊変数 `$?` には、直前に実行したコマンドの終了ステータスコードが格納されて、値0はテストが成功したことを意味しますから、どちらの構文でも `/etc` が有効なディレクトリであると評価しました。ファイルまたはディレクトリへのパスが変数 `$VAR` に格納されていると仮定すると、次の式を `test` への引数または角括弧内で使用できます。

`-a "$VAR"`
`VAR` のパスがファイルシステムに存在する場合に成功します（`-e` と同じ）

`-b "$VAR"`
`VAR` のパスがブロックデバイスである場合に成功します。

`-c "$VAR"`
`VAR` のパスがキャラクターデバイスである場合に成功します。

`-d "$VAR"`
`VAR` のパスがディレクトリである場合に成功します。

`-e "$VAR"`
`VAR` のパスがファイルシステムに存在する場合に成功します（`-a` と同じ）。

`-f "$VAR"`
`VAR` のパスが存在し、それが通常ファイルである場合に成功します。

`-g "$VAR"`
`VAR` のパスにSGIDパーミッションがある場合に成功します。

`-h "$VAR"`
`VAR` のパスがシンボリックリンクである場合に成功します（`-L` と同じ）。

`-L "$VAR"`
`VAR` のパスがシンボリックリンクである場合に成功します（`-h` と同じ）。

`-k "$VAR"`
`VAR` のパスに スティッキー ビットパーミッションがある場合に成功します。

`-p "$VAR"`
`VAR` のパスが pipe である場合に成功します。

`-r "$VAR"`
`VAR` のパスが現在のユーザーに読み取れる場合に成功します。

`-s "$VAR"`
`VAR` のパスが存在し、空でない場合に成功します。

`-S "$VAR"`
VAR のパスがソケットである場合に成功します。

`-t "$VAR"`
VAR のパスが端末で開かれている場合に成功します。

`-u "$VAR"`
VAR のパスにSUIDパーミッションがある場合に成功します。

`-w "$VAR"`
VAR のパスが現在のユーザーによって書き込み可能である場合に成功します。

`-x "$VAR"`
VAR のパスが現在のユーザーによって実行可能である場合に成功します。

`-O "$VAR"`
VAR のパスが現在のユーザーによって所有されている場合に成功します。

`-G "$VAR"`
VAR のパスが現在のユーザーが所属するグループに属している場合に成功します。

`-N "$VAR"`
VAR のパスが最後にアクセスされた後に変更されている場合に成功します。

`"$VAR1" -nt "$VAR2"`
VAR1 のパスが VAR2 のパスよりも後に変更された場合に成功します。

`"$VAR1" -ot "$VAR2"`
VAR1 のパスが VAR2 のパスよりも前に変更された場合に成功します。

`"$VAR1" -ef "$VAR2"`
VAR1 のパスと VAR2 のパスがハードリンクされている場合に成功します。

次に示すように、文字列の比較を行う事もできます。この場合、テストする変数を二重引用符で囲むことがお勧めです。条件式にはパラメーターが必要ですから、変数が空の場合には必要なパラメーターが無いために構文エラーが発生が発生することがあります。二重引用符で囲んでおくと、変数が空の場合でもそこに「空文字列」があることが分かります。

`-z "$TXT"`
変数 TXT が空（サイズがゼロ）である場合に成功します。

`-n "$TXT"` or `test "$TXT"`
変数 TXT が空でない場合に成功します。

`"$TXT1" = "$TXT2"` ないし `"$TXT1" == "$TXT2"`
TXT1 と TXT2 が等しい場合に成功します。

`"$TXT1" != "$TXT2"`
TXT1 と TXT2 が等しくない場合に成功します。

"$TXT1" < "$TXT2"
TXT1 が TXT2 よりもアルファベット順で前にある場合に成功します。

"$TXT1" > "$TXT2"
TXT1 が TXT2 よりもアルファベット順で後にある場合に成功します。

言語が異なると、アルファベットの順序が異なることがあります。常に同じ結果を得たい場合は、環境変数 LANG に C をセットして（LANG=C）、言語設定を上書きします。ログインシェルの設定を上書きしないように、シェルスクリプトの中で設定するようにしましょう。

数値比較の為の条件式もあります。

$NUM1 -lt $NUM2
NUM1 が NUM2 よりも小さい場合に成功します。（lt は less than の略）

$NUM1 -gt $NUM2
NUM1 が NUM2 よりも大きい場合に成功します。（gt は grater than の略）

$NUM1 -le $NUM2
NUM1 が NUM2 以下である場合に成功します。（le は less than or equal の略）

$NUM1 -ge $NUM2
NUM1 が NUM2 以上である場合に成功します。（ge は grater than or equal の略）

$NUM1 -eq $NUM2
NUM1 が NUM2 と等しい場合に成功します。

$NUM1 -ne $NUM2
NUM1 が NUM2 と等しくない場合に成功します。

論理演算を使って、条件式を反転したり、複数の条件式を結合することができます。

! EXPR
条件式 EXPR の結果を反転します。

EXPR1 -a EXPR2
EXPR1 と EXPR2 の両方が真である場合に真になります。

EXPR1 -o EXPR2
2つの式の少なくとも1つが真である場合に真になります。

if 文のバリエーションに、case 文があります。case 文は、そこで指定した文字列（WORD）の値に応じて、処理内容を選択する場合に便利です。case ブロックの中では、case 文で指定した文字列が、パターン□—□）で終わる縦棒 | で区切られた文字列のリスト□—□に続くブロックが実行されます。サンプルとして、引数に指定したディストリビューションが使用しているパッケージ形式を表示するスクリプトを見てみましょう。

```
#!/bin/bash
```

```
DISTRO=$1

echo -n "Distribution $DISTRO uses "
case "$DISTRO" in
  debian | ubuntu | mint)
    echo -n "the DEB"
    ;;
  centos | fedora | opensuse )
    echo -n "the RPM"
    ;;
  *)
    echo -n "an unknown"
    ;;
esac
echo " package format."
```

パターンに一致した場合に実行するコマンドのリストは、次のいずれかで終了します:

`;;`
: 他のパターンとの一致を試みず、caseブロックの次に進みます。

`:&`
: 次のバターンと一致するかどうかにかかわらず、そのコマンドリストの実行に進みます。

`::&`
: 次のパターンとの一致を調べて、一致すればそのコマンドリストの実行に進みます。

サンプルの最後のパターンである `*` は、いずれのパターンとも一致しなかった場合に実行されるコマンドリストを定義するためのものです。`case` ブロックは、`esac`（`case` の逆順）で終了します。

このサンプルスクリプトの名前が `script.sh` であり、最初の引数に `opensuse` を指定して実行すると、次のように出力されます。

```
$ ./script.sh opensuse
Distribution opensuse uses the RPM package format.
```

TIP

Bashには動作を調整するための特別な変数（フラグ）がいくつも用意されていて、その中の一つに `case` や `test` での文字列比較の際にアルファベットの大文字小文字を区別しないことを指定する `nocasematch` 変数があります。これらの変数を切り替えるには、組み込みコマンド `shopt` を使用し、`shopt -s` で有効化し、`shopt -u` で無効化します。`shopt -s nocasematch` を実行するとフラグが有効化されて、`case` では大文字・小文字を区別しないパターンマッチが行われるようになります。これらの変数は現在のシェルにのみ影響するため、サブシェルには継承されません。すなわち、必要があればシェルスクリプトの中で設定します。

検索文字列とパターンを評価する際には、チルダ展開、パラメータ展開、コマンド置換、算術展開が行われます。アイテムが引用符で囲まれている場合には、マッチングの前に引用符は削除されます。

ループ

スクリプトは同じタスクを何度も繰り返すためのツールとして利用されることがよくありますので、終了条件を満たすまで同じコマンドセットを繰り返し実行する　ループ　を作成するコマンドが備わっています。Bashでは、異なるループ構造を実現する　`for`、`until`、`while`　の3種類のループコマンドがあります。

`for`　ループでは、指定したリストから1つずつアイテムを取り出して、それぞれの単語に対してコマンドリストを実行します。ループの前にリストから取り出したアイテムが変数に割り当てられますので、その変数をコマンドリストの中で使用して処理を進めます。すべてのアイテムを処理すると、ループが終了します。`for` ループの構文は次の通りです。

```
for VARNAME in LIST
do
    COMMANDS
done
```

`VARNAME`　は取り出したアイテムを格納する変数で、`LIST`　は通常、空白で区切った単語の並びです。`LIST`　の区切り文字は、環境変数　`IFS`　で指定することができ、デフォルトでは　スペース、タブ、改行 です。コマンドのリストは、`do` と `done` で囲みます。

次の例では、リストから数値を順に取り出して1つずつ変数　`NUM`　に格納して、コマンドリストを実行します。

```
#!/bin/bash

for NUM in 1 1 2 3 5 8 13
do
    echo -n "$NUM is "
    if [ $(( $NUM % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
    fi
done
```

この例では、`if`　文の算術式で変数　`NUM`　が偶数か奇数かを判定しています。スクリプトファイルが現在のディレクトリにある `script.sh` だとすると、次のように実行します。

```
$ ./script.sh
1 is odd.
1 is odd.
2 is even.
3 is odd.
5 is odd.
8 is even.
13 is odd.
```

Bashでは、二重括弧を用いて、C言語の `for` 文と同等のインデックスを用いるループも使えます。

```
#!/bin/bash

SEQ=( 1 1 2 3 5 8 13 )

for (( IDX = 0; IDX < ${#SEQ[*]}; IDX++ ))
do
    echo -n "${SEQ[$IDX]} is "
    if [ $(( ${SEQ[$IDX]} % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
    fi
done
```

このサンプルも、先の例と同じ結果を出力します。ここでは、リストから取り出したアイテムを格納する `NUM` 変数ではなく、配列の要素番号（インデックス）を格納する `IDX` 変数を使用しています。`IDX` の値は、`IDX = 0` によって初期化され、ループを1回実行する度に `IDX++` によって1ずつ増加します。配列の要素数と比較する `IDX < ${#SEQ[*]}` が成立する（真となる）間は、ループが繰り返されます。

`until` ループは、条件が満たされるまで（たとえば `test` コマンドが0（成功）を返すまで）コマンドリストを実行し続けるものです。前の例と同じループ条件を `until` ループで書くと、次のようになります。

```
#!/bin/bash

SEQ=( 1 1 2 3 5 8 13 )

IDX=0

until [ $IDX -eq ${#SEQ[*]} ]
do
    echo -n "${SEQ[$IDX]} is "
    if [ $(( ${SEQ[$IDX]} % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
    fi
    IDX=$(( $IDX + 1 ))
done
```

`until` ループでは、インデックスの初期化や更新のコードが別に必要となるため、`for` ループよりも複雑に感じるかもしれませんが、ループの停止条件が単純な数値だけで決められない場合などには、より分かりやすく表現することができることがあります。ループが無限に実行されることが無いように、ループを回る度にインデックスを更新し、条件がいつかは満たされることが重要です。

`while` ループは `until` ループとは逆に、条件が満たされている間（たとえば `test` コマンドが0（成

功）を返している間）はコマンドリストを実行し続けるものです。前の例の `until [ $IDX -eq ${#SEQ[*]} ]` を `while` で書き直すと `while [ $IDX -lt ${#SEQ[*]} ]` になります。IDX の値が、SEQ の要素数よりも小さい間は、ループを繰り返すことになります。

より複雑な例

最後に少し実用的なサンプルを見てみましょう。ユーザーが指定するファイルやディレクトリを、別のストレージデバイスに定期的にバックアップするスクリプトを考えます。自動化して定期的に実行するアプリケーションのひとつです。

スクリプトは、第1引数にバックアップ対象のディレクトリやファイルを探す起点ディレクトリを指定し、第2引数にバックアップ先の（別デバイス上の）ディレクトリを指定するものとします。バックアップするファイルやディレクトリは、あらかじめ `~/.sync.list` ファイルにリストしておきます。ここでは次のように、1行に1つのディレクトリないしファイルを記入したファイルを作成しました。

```
$ cat ~/.sync.list
Documents
To do
Work
Family Album
.config
.ssh
.bash_profile
.vimrc
```

このファイルにはファイルとディレクトリが混在しており、名前に空白が含まれているものもあります。このようなファイルをシェルスクリプで読み込む場合には、Bashの組み込みコマンド `mapfile` を使うのが適当です。このコマンドは、任意のテキストを解析して、要素を配列変数に格納します。スクリプトファイルの名前は `sync.sh` として、以下の内容を記入します。

```
#!/bin/bash

set -ef

# List of items to sync
FILE=~/.sync.list

# Origin directory
FROM=$1

# Destination directory
TO=$2

# Check if both directories are valid
if [ ! -d "$FROM" -o ! -d "$TO" ]
then
    echo Usage:
    echo "$0 <SOURCEDIR> <DESTDIR>"
    exit 1
fi
```

```
# Create array from file
mapfile -t LIST < $FILE

# Sync items
for (( IDX = 0; IDX < ${#LIST[*]}; IDX++ ))
do
    echo -e "$FROM/${LIST[$IDX]} \u2192 $TO/${LIST[$IDX]}";
    rsync -qa --delete "$FROM/${LIST[$IDX]}" "$TO";
done
```

順に説明していきましょう。まず、`set` コマンドでシェルの動作オプションを調整します。`-e` オプションは、コマンドが失敗した場合（終了ステータスが0ではない場合）に、スクリプトの実行を直ぐに終了します。`-f` オプションは、ファイル名のグロブを無効にします。コマンドのオプションと同様に、2つのオプションをまとめて `-ef` と指定することができます。このステップは必須ではありませんが、想定外の状況でスクリプトが予期しない動作をすることの予防になります。

スクリプトの本質的な処理内容は、大きく3つの部分に分けることができます。

1. パラメーターを確認する

   `FILE` 変数には、コピーするファイルやディレクトリのリストを含むファイルのパス名⌧—⌧~/.sync.list⌧—⌧が格納されます。`FROM` 変数には起点となるディレクトリの、`TO` 変数にはコピー先ディレクトリの、パス名がそれぞれ格納されます。これらはユーザーが引数で指定するので、ディレクトリの存在を `[ ! -d "$FROM" -o ! -d "$TO" ]` で確認し、いずれかが無い場合には簡単なヘルプメッセージを表示して、ステータス1で終了します。

2. バックアップ対象のファイルとディレクトリのリストをロードする

   バックアップするファイルやディレクトリのリストを、`mapfile -t LIST < $FILE` で配列に読み取ります。`-t` オプションは、それぞれのアイテムの末尾にある改行を削除することを指定しています（つまり1行を1要素として読み取ります）。`$FILE` からリダイレクトで読み込んでいることに着目して下さい。

3. コピーしてユーザーに通知する

   二重括弧形式の `for` ループでは、`IDX` 変数をインデックスとして使用して、`LIST` からアイテムを一つずつ取り出して処理します。`echo` コマンドは、処理内容をユーザーに通知するもので、Unicode文字 `\u2192`（右向き矢印）を使用するので `-e` オプションを指定しています。

   `rsync` コマンドは、コピー元とコピー先を比較して、変更されたファイルのみをコピーする、バックアップ目的にはとても便利なコマンドです。`-q` オプションはエラーメッセージの出力を抑止し、`-a` オプションはファイルやディレクトリの属性をコピー先でも維持する アーカイブモード を指示しています。`--delete` オプションは、コピー元に無いファイルがコピー先に存在する場合にそれを削除するものです。

ユーザー `carol` のホームディレクトリ（ここに `.rsync.list` にリストされたアイテムがすべて存在するものとします）をコピー元とし、マウントされた外部ストレージデバイス `/media/carol/backup` をコピー先ディレクトリとして指定すると、実行結果は次のようになります。

```
$ sync.sh /home/carol /media/carol/backup
```

```
/home/carol/Documents → /media/carol/backup/Documents
/home/carol/"To do" → /media/carol/backup/"To do"
/home/carol/Work → /media/carol/backup/Work
/home/carol/"Family Album" → /media/carol/backup/"Family Album"
/home/carol/.config → /media/carol/backup/.config
/home/carol/.ssh → /media/carol/backup/.ssh
/home/carol/.bash_profile → /media/carol/backup/.bash_profile
/home/carol/.vimrc → /media/carol/backup/.vimrc
```

この例では、バックアップ対象ファイルのリストを収めたファイル（`.sync.list`）をチルダを使って指定しているので、`carol` 以外のユーザーでも利用することができます。この `script.sh` を、`PATH` が通ったディレクトリ（たとえば `/usr/local/bin`）に置くとよいでしょう。

演習

1. `test` コマンドで、変数 `FROM` に格納されているファイルパスが、変数 `TO` に格納されているファイルよりも新しいかどうかを確認するにはどうしますか？

2. 0から9までの数列を出力するスクリプトを作りたいのですが、次のスクリプトは0を無限に出力してしまいます。どう修正すればよいですか？

```
#!/bin/bash

COUNTER=0

while [ $COUNTER -lt 10 ]
do
  echo $COUNTER
done
```

3. ソートされたユーザー名のリストを出力するスクリプトを作成しました。あるシステムでは次のように表示されます。

```
carol
Dave
emma
Frank
Grace
henry
```

同じスクリプトで、別のシステムでは次のようにソートされます。

```
Dave
Frank
Grace
carol
emma
henry
```

出力が異なる理由を説明してください。

発展演習

1. スクリプトに与えられたすべての引数で、Bash配列を初期化するにはどうしますか？

1. 直感に反して、コマンド `test 1 > 2` がtrueと評価されるのはなぜですか？

2. フィールド区切り文字を一時的に改行文字のみに変更し、それを元に戻すにはどうしますか？

まとめ

このレッスンでは、条件判定を行う `test` コマンドと、条件によって実行するコマンドを制御する `if` 文ならびに `case` 文、さらにループ構造について説明しました。実用的なシェルスクリプトの例として、簡単なファイル同期スクリプトも示しました。このレッスンで説明した事柄は次の通りです。

- `if` と `case` による条件判断
- ループの作り方: `for`、`until`、`while`
- 配列やパラメータを列挙する方法

以下のコマンドと手順を紹介しました:

`test`
指定された条件を満たしているかどうかを判断します。

`if`
条件判断の結果に基づいてコマンドの実行を分岐します。

`case`
変数の値に応じてコマンドの実行を分岐します。

`for`
条件に基づいてコマンドの実行を繰り返します。

`until`
条件が満たされるまで、コマンドの実行を繰り返します。

`while`
条件が満たされている間、コマンドの実行を繰り返します。

演習の解答

1. `test` コマンドで、変数 `FROM` に格納されているファイルが、変数 `TO` に格納されているファイルよりも新しいかどうかを確認するにはどうしますか？

   コマンド `test "$FROM" -nt "$TO"` は、`FROM` 変数のファイルが `TO` 変数のファイルよりも新しい場合に、終了コード0（成功）を返します。

2. 0から9までの数列を出力するスクリプトを作りたいのですが、次のスクリプトは0を無限に出力してしまいます。どう修正すればよいですか？

   ```
   #!/bin/bash

   COUNTER=0

   while [ $COUNTER -lt 10 ]
   do
       echo $COUNTER
   done
   ```

   ループの最後で、変数 `COUNTER` の値を1増加します。つまり、`COUNTER=$(( $COUNTER + 1 ))` を `done` の前に追加します。

3. ソートされたユーザー名のリストを出力するスクリプトを作成しました。あるシステムでは次のように表示されます。

   ```
   carol
   Dave
   emma
   Frank
   Grace
   henry
   ```

   同じスクリプトで、別のシステムでは次のようにソートされます。

   ```
   Dave
   Frank
   Grace
   carol
   emma
   henry
   ```

   出力が異なる理由を説明してください。

   言語設定（ロケール）によって文字列の順序が異なります。言語による相違を回避するには、`LANG` 環境変数に `C` をセットしてから、ソートを行います。

発展演習の解答

1. スクリプトに与えられたすべての引数で、Bash配列を初期化するにはどうしますか？

   `PARAMS=( $* )` ないし `PARAMS=( "$@" )` で、すべての引数を含む `PARAMS` という配列を作成します。

2. 直感に反して、コマンド `test 1 > 2` がtrueと評価されるのはなぜですか？

   エスケープされていないので、`>` がリダイレクト指示と解釈されます。何も判断していないので `test` コマンドは成功します。また、`>` は文字列の比較を行うもので、数値の比較には使用できません。

3. フィールド区切り文字を一時的に改行文字のみに変更し、それを元に戻すにはどうしますか？

   `OLDIFS=$IFS` で、別の変数 `OLDIFS` に `IFS` の値を格納しておきます。次に `IFS=$'\n'` でフィールド区切り文字を変更し、コマンドを実行します。最後に、`IFS=$OLDIFS` で `IFS` の値を元に戻します。

Linux
Professional
Institute

課題 106: ユーザーインターフェースとデスクトップ

106.1 X11のインストールと設定

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 106.1

総重量

2

主な知識分野

- X11の基本的な構成の理解。
- X Window設定ファイルの基本的な理解と知識。
- キーボードレイアウトなどの、Xorgの設定に対して追加の設定を記述する。
- ディスプレイマネージャやウインドウマネージャなどの、デスクトップ環境のコンポーネントの理解。
- リモートXサーバの、Xサーバとディスプレイアプリケーションへのアクセスの管理。

用語とユーティリティ

- `/etc/X11/xorg.conf`
- `/etc/X11/xorg.conf.d/`
- `~/.xsession-errors`
- `xhost`
- `xauth`
- `DISPLAY`
- `X`

106.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	106 ユーザーインターフェイスとデスクトップ
Objective:	106.1 X11のインストールと構成
Lesson:	1 of 1

はじめに

X Window Systemは、画面にテキストとグラフィックスを表示するためのソフトウェアスタックです。Xクライアントの全体的な外観とデザインはX Window System自体によって決定されるのではなく、個々のXクライアント、ウィンドウマネージャー（Window Maker、Tab Window Managerなど）、ないしは、KDE、GNOME、Xfceなどの デスクトップ環境 によって処理されます。デスクトップ環境については、後のレッスンで説明します。このレッスンでは、X Window Systemの基本アーキテクチャと、管理者がXを構成するための一般的なツールを取り上げます。

X Window Systemはクロスプラットフォームであり、Linux、BSD、Solaris、その他のUnix系システムなど、さまざまなオペレーティングシステムで実行されます。AppleのmacOSやMicrosoft Windowsで利用可能なものもあります。

最新のLinuxディストリビューションで使用されているXプロトコルは X.org バージョン11で、通常はX11 と表記されます。Xプロトコルは、XクライアントとXサーバー間の通信機構です。 XクライアントとXサーバーの違いを以下で説明します。

NOTE

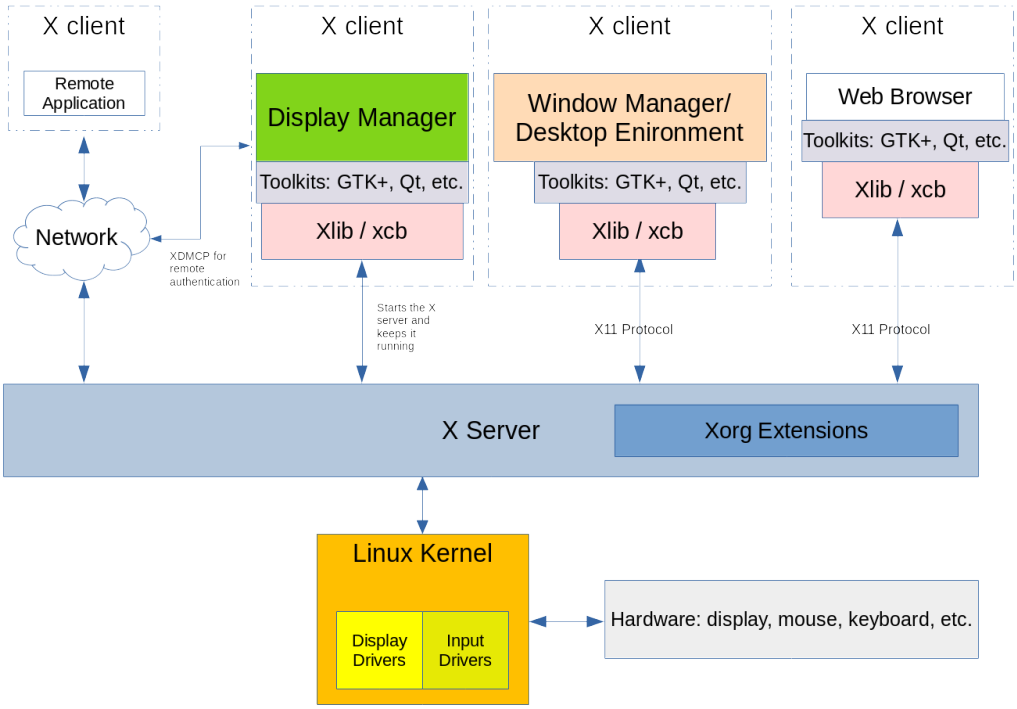
X Window Systemの前身は、IBM、DEC、およびMITの共同開発物である W と呼ばれるウィンドウシステムでした。このソフトウェアは1984年に Athenaプロジェクトから生まれました。新しいディスプレイサーバーの開発を始めたときに、開発者はアルファベットの文字“X”を選択しました。現在、X Window Systemは X.orgファウンデーション によって管理されています。

X Window Systemのアーキテクチャ

X Window Systemは、基本的な2次元図形（拡張機能によって3次元図形）をディスプレイに描画するための仕組みを提供します。クライアントとサーバーに分割されており、グラフィカルなデスクトップ

を備えるシステムでは、両方のコンポーネントが同じコンピューター上にあります。クライアントコンポーネントは、ターミナルエミュレータ、ゲーム、Webブラウザなど、アプリケーションの形を取ります。それぞれのクライアントアプリケーションは、コンピュータ画面上に表示するウィンドウの位置とサイズをXサーバーに通知します。クライアントがそのウィンドウに表示する内容を生成し、Xサーバーは要求された図形を画面に表示します。X Window Systemは、マウス、キーボード、トラックパッドなどのデバイスからの入力も処理します。

X Window Systemの基本構造



X Window Systemはネットワーク対応であり、ネットワーク上の異なるコンピューターから、複数のXクライアントが1つのリモートXサーバーに描画要求を行うことができます。つまり、ユーザーは、ローカルシステムには存在していない、リモートシステム上のグラフィカルアプリケーションにアクセスできます。

X Window Systemの重要な特徴は、モジュール化されていることです。X Window Systemが発展する過程で、新しい機能が開発されて、そのフレームワークに追加されました。それらの新しいコンポーネントは、Xサーバーの拡張機能として追加され、コアX11プロトコルはそのまま残されています。これらの拡張機能は、Xorg　　のライブラリファイルに含まれています。Xorgライブラリには、`libXrandr`、`libXcursor`、`libX11`、`libxkbfile`　　などが含まれていて、それぞれがXサーバーに拡張機能を提供します。

ディスプレイマネージャー　　は、ローカルないしネットワーク上のコンピュータへのグラフィカルログインを提供します。コンピューターがブートするとディスプレイマネージャーが起動して、ユーザー認証を行ってからそのユーザー用のXサーバーセッションを開始します。ディスプレイマネージャには、Xサーバーを稼働させ続ける役割もあります。ディスプレイマネージャーには、GDM、SDDM、LightDMなどがあります。

実行中のXサーバーの各インスタンスは、識別のために　ディスプレイ名　を持っています。ディスプレ

イ名の書式は次の通りです:

```
hostname:displaynumber.screennumber
```

グラフィカルアプリケーションはディスプレイ名を利用して、どのホスト（リモートX接続の場合）のどの画面に表示するかを決定します。

`hostname` は、アプリケーションのウィンドウを表示するシステムの名前を示します。ディスプレイ名にhostnameが含まれていない場合は、ローカルホストが仮定されます。

`displaynumber` は、ラップトップの画面1枚でも、ワークステーションの複数画面でも、使用する「画面群」を示します。実行中の各Xサーバーセッションには、`0` から始まる `displaynumber` が与えられます。

デフォルトの `screennumber` は `0` です。物理的なスクリーンが1つしかない場合や、複数の物理的なスクリーンが1つのスクリーンとして動作するように設定されている場合に当ります。マルチモニター環境において、すべてのスクリーンが1つの論理的なスクリーンにまとめられている場合は、アプリケーションのウィンドウをスクリーン間で自由に移動できます。それぞれのスクリーンが互いに独立して動作するように設定されている場合は、それぞれのスクリーンにその中で開かれたアプリケーションウィンドウが収容され、ウィンドウをスクリーン間で移動することはできません。独立したスクリーンには、それぞれ番号が割り当てられます。 使用中のスクリーンが1つだけの場合、ピリオドとscreennumberは省略されます。

実行中のXセッションのディスプレイ名は、`DISPLAY` 環境変数に格納されています。

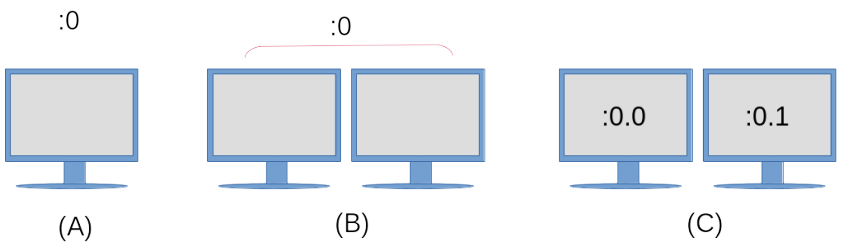
```
$ echo $DISPLAY
:0
```

出力されたディスプレイ名の意味は次のとおりです:

1. 使用中のXサーバーはローカルシステム上にあるため、コロンの左側には何もありません（hostnameが省略されています）。
2. コロンの直後の `0`（displaynumber） は、現在のXサーバーセッションが最初のセッションであることを示します。
3. 論理画面は1つだけなので、screennumberは省略されています。

次の図で、この概念を説明します。上の例は(B)のケースに当たります:

ディスプレイ設定の例



(A)
1つのモニター、1つのディスプレイ、および1つのスクリーン。

(B)
2つの物理モニターを、1つのディスプレイとして構成。アプリケーションウィンドウを、2つのモニター間で自由に移動できます。

(C)
（`:0` が示すように）1つのディスプレイですが、各モニターは独立したスクリーンです。両方の画面は、キーボードやマウスなどの入力デバイスを共有しますが、画面 `:0.0` で開いたアプリケーションを画面 `:0.1` に移動したり、その逆を行うことはできません。

アプリケーションを起動する画面を指定するには、アプリケーションを起動する前に、`DISPLAY` 環境変数にディスプレイ名を割り当てます。

```
$ DISPLAY=:0.1 firefox &
```

このコマンドでは、上図（C）の右側の画面でFirefox Webブラウザーを起動します。一部のツールキットには、アプリケーションに実行する画面を指示するためのオプションも用意されています。`gtk-options(7)` のマニュアルページの `--screen` と `--display` に例があります。

Xサーバーの設定

伝統的に、Xサーバーのメインの設定ファイルは `/etc/X11/xorg.conf` です。最新の Linux ディストリビューションでは、Xサーバーの開始時に自動的に設定ファイルが生成されるため、`xorg.conf` ファイルが存在しないことがあります。

`xorg.conf` ファイルは、セクション と呼ばれるブロックに分かれています。各セクションは `Section` という語で始まり、その後に設定するコンポーネントを示す セクション名 が続きます。それぞれの `Section` ブロックは、対応する `EndSection` で終了します。典型的な `xorg.conf` ファイルには、以下のセクションが含まれています:

`InputDevice`
キーボードやマウスのモデルやタイプを指定します。

`InputClass`
最近のLinuxディストリビューションでは、このセクションは通常 `/etc/X11/xorg.conf.d/` の下にある別の設定ファイルにあります。`InputClass` は特定のハードウェアではなく、キーボードやマウスなどのハードウェアデバイスの クラス を設定するために使用されます。以下は、`/etc/X11/xorg.conf.d/00-keyboard.conf` ファイルの例です。

```
Section "InputClass"
        Identifier "system-keyboard"
        MatchIsKeyboard "on"
        Option "XkbLayout" "us"
        Option "XkbModel" "pc105"
EndSection
```

オプション `XkbLayout` は、Dvorak、QWERTY、右利き/左利き、言語など、キーボードのキーレイアウトを指定します。 オプション `XkbModel` は、使用中のキーボードの種別を指定します。モデルやレイアウトとその説明は、`xkeyboard-config(7)` にあります。キーボードレイアウトに関連するファイルは、`/usr/share/X11/xkb` にあります。Chromebookコンピュータでのギリシャ語のPolytonicキーボードレイアウトの例は次のようになります。

```
Section "InputClass"
        Identifier "system-keyboard"
        MatchIsKeyboard "on"
        Option "XkbLayout" "gr(polytonic)"
        Option "XkbModel" "chromebook"
EndSection
```

また、実行中のXセッションで `setxkbmap` コマンドを使用して、キーボードのレイアウトを変更することもできます。 Chromebookコンピュータでギリシャ語のPolytonicレイアウトを設定するコマンドの例を次に示します。

```
$ setxkbmap -model chromebook -layout "gr(polytonic)"
```

この設定は、Xセッションの間だけ有効です。変更を永続化するには、`/etc/X11/xorg.conf.d/00-keyboard.conf` ファイルを変更して必要な設定を追加します。

NOTE `setxkbmap` コマンドは、 Xキーボード拡張機能 （XKB）を使用します。これは、X Window System の拡張機能による付加機能の一例です。

最新のLinuxディストリビューションでは、（`systemd` の一部として）`localectl` コマンドが提供されています。このコマンドはキーボードレイアウトを変更することもできて、設定ファイル `/etc/X11/xorg.conf.d/00-keyboard.conf` を自動的に生成します。今度は `localectl` コマンドを使用して、ギリシャ語のPolytonicキーボードレイアウトを Chromebook に設定してみましょう。

```
$ localectl --no-convert set-x11-keymap "gr(polytonic)" chromebook
```

ここでは、`localectl` がホストのコンソールキーマップを変更しないように、`--no-convert` オプションを指定しています。

Monitor

`Monitor` セクションには、使用する物理モニターとその接続先を記述します。2番目のディスプレイポートに接続されたモニターをプライマリモニターとして使用する設定例を次に示します。

```
Section "Monitor"
        Identifier  "DP2"
        Option      "Primary" "true"
EndSection
```

Device

`Device` セクションには、使用する物理的なビデオカードを記述します。このセクションには、ビデ

オカードのドライバとして使用するカーネルモジュールと、マザーボード上の物理的な位置（訳注: PCIのバス番号）も含まれています。

```
Section "Device"
        Identifier  "Device0"
        Driver      "i915"
        BusID       "PCI:0:2:0"
EndSection
```

Screen

`Screen` セクションで `Monitor` セクションと `Device` セクションを結び付けます。`Screen` セクションの例を次に示します:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Device0"
        Monitor    "DP2"
EndSection
```

ServerLayout

`ServerLayout` セクションは、マウス、キーボード、画面などのすべてのセクションを、1つのX Window Systemインターフェイスにグループ化します。

```
Section "ServerLayout"
    Identifier   "Layout-1"
    Screen       "Screen0" 0 0
    InputDevice  "mouse1"  "CorePointer"
    InputDevice  "system-keyboard"  "CoreKeyboard"
EndSection
```

NOTE 設定ファイルにすべてのセクションが存在とは限りません。セクションが存在しない場合は、実行中のXサーバーインスタンスのデフォルト値が使われます。

ユーザー指定の設定ファイルも `/etc/X11/xorg.conf.d/` に置かれます。ディストリビューションが提供する設定ファイルは、`/usr/share/X11/xorg.conf.d/` にあります。`/etc/X11/xorg.conf.d/` に設定ファイルがある場合、`/etc/X11/xorg.conf` ファイルより先に読み込まれます。

コンピュータ上で実行中のXサーバーインスタンスに関する情報を表示するには、`xdpyinfo` コマンドを使用します。実行例を以下に示します:

```
$ xdpyinfo
name of display:    :0
version number:    11.0
vendor string:    The X.Org Foundation
vendor release number:    12004000
X.Org version: 1.20.4
maximum request size:  16777212 bytes
```

```
motion buffer size:  256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    7
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 4, bits_per_pixel 8, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
    depth 15, bits_per_pixel 16, scanline_pad 32
    depth 16, bits_per_pixel 16, scanline_pad 32
    depth 24, bits_per_pixel 32, scanline_pad 32
    depth 32, bits_per_pixel 32, scanline_pad 32
keycode range:    minimum 8, maximum 255
focus:  None
number of extensions:    25
    BIG-REQUESTS
    Composite
    DAMAGE
    DOUBLE-BUFFER
    DRI3
    GLX
    Generic Event Extension
    MIT-SCREEN-SAVER
    MIT-SHM
    Present
    RANDR
    RECORD
    RENDER
    SECURITY
    SHAPE
    SYNC
    X-Resource
    XC-MISC
    XFIXES
    XFree86-VidModeExtension
    XINERAMA
    XInputExtension
    XKEYBOARD
    XTEST
    XVideo
default screen number:    0
number of screens:    1

screen #0:
  dimensions:    3840x1080 pixels (1016x286 millimeters)
  resolution:    96x96 dots per inch
  depths (7):    24, 1, 4, 8, 15, 16, 32
  root window id:    0x39e
  depth of root window:    24 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x25
  default number of colormap cells:    256
  preallocated pixels:    black 0, white 16777215
```



```
  options:    backing-store WHEN MAPPED, save-unders NO
  largest cursor:    3840x1080
  current input event mask:    0xda0033
    KeyPressMask             KeyReleaseMask           EnterWindowMask
    LeaveWindowMask          StructureNotifyMask      SubstructureNotifyMask
    SubstructureRedirectMask PropertyChangeMask       ColormapChangeMask
  number of visuals:    270
...
```

この例では、ディスプレイの名前（環境変数 `DISPLAY` と同じ)、Xサーバーのバージョン、Xorg拡張の数と一覧、画面自体に関する詳細な情報など、重要な情報を太字で記載しています。

基本的なXorg設定ファイルの作成

最近のLinuxでは、システムの起動時にXが自動設定されますが、今でも `xorg.conf` ファイルを使うことができます。永続的な `/etc/X11/xorg.conf` ファイルを生成するには、次のコマンドを実行します。

```
$ sudo Xorg -configure
```

NOTE

すでにXセッションが起動している場合は、コマンドに別の `DISPLAY` を指定します。以下に例を示します:

```
$ sudo Xorg :1 -configure
```

一部のLinuxディストリビューションでは、`X` が `Xorg` へのシンボリックリンクであるため、`Xorg` の代わりに `X` コマンドを使うこともできます。

`xorg.conf.new` ファイルがカレントディレクトリに作成されます。このファイルの内容は、Xサーバーがローカルシステムのハードウェアとドライバーで利用可能であることを確認したものです。このファイルを使用するには、ファイルを `/etc/X11/` ディレクトリに移動し、名前を `xorg.conf` に変更します。

```
$ sudo mv xorg.conf.new /etc/X11/xorg.conf
```

NOTE

X Window Systemのコンポーネントに関する詳細情報は、以下のmanページに記載されています: `xorg.conf(5)`、`Xserver(1)`、`X(1)`、`Xorg(1)`。

Wayland

Waylandは、X Window Systemに代わるものとして設計された新しい表示プロトコルです。最近のLinuxディストリビューションの多くは、これをデフォルトのディスプレイサーバとして使用しています。Xよりもシステムリソースの消費が少なく、インストール時のフットプリントも小さくなるように設計されています。このプロジェクトは2010年に始まり、現在も現役および元X.org開発者による作業を取り込みながら活発に開発が行われています。

X Window Systemとは異なり、クライアントとカーネルの間で動作するサーバーインスタンスはありません。代わりに、アプリケーションは直接ないしツールキット（Gtk+やQtなど）経由で、クライアント

ウィンドウに表示します。カーネルはWaylandプロトコルで、要求をWayland コンポーザ に転送します。 コンポーザは、デバイス入力を処理し、ウィンドウの構成と管理を行い、出力された要素を結合して画面に出力するシステムの一部分です。

Gtk+ 3 や Qt 5 などの最新のツールキットのほとんどは、X Window System と Wayland のいずれにも対応しています。しかしながら現時点では、すべてのアプリケーションが、Waylandでの表示をサポートしているわけではありません。X Window System用に書かれたアプリケーションやフレームワークは、XWayland に内包されたX Window System環境で実行されます。XWaylandシステムには、Waylandクライアントとして実装されたスタンドアロンのXサーバーインスタンスが備わっていて、それがクライアントウィンドウのコンテンツを（Wayland経由で）表示します。

X Window Systemが使用中の画面を識別するために `DISPLAY` 環境変数を使用するように、Wayland プロトコルは `WAYLAND_DISPLAY` 環境変数を使用します。Waylandを実行しているシステムのサンプル出力を以下に示します:

```
$ echo $WAYLAND_DISPLAY
wayland-0
```

この環境変数は、Xを実行しているシステムでは使用できません。

演習

1. システムで使用できるXorg拡張機能を確認するコマンドは何ですか?

2. コンピュータ用の新しい10ボタンマウスを入手しましたが、すべてのボタンを正しく機能させるには追加の設定が必要です。Xサーバー設定の他の部分に影響をあたえずに、このマウス用の新しい設定ファイルをどのディレクトリに作成し、どのセクションに記入しますか？

3. Linuxシステムにおいて、Xサーバーの実行を維持するコンポーネントは何ですか？

4. 新しい `xorg.conf` ファイルを作成する、`X` コマンドのオプションは何ですか？

発展演習

1. 3台のモニタを1つのディスプレイとして使用するように構成された `lab01` というシステムを使用しています。3つ目のスクリーンに開いたリモートマシンのターミナルエミュレータで、`DISPLAY` 環境変数の値を表示するとどうなりますか？

2. X Window Systemで使用するキーボード設定ファイルを作成するコマンドは何ですか？

3. 一般的なx86_64アーキテクチャのマシンにインストールしたLinuxでは、キーボードから `Ctrl`+`Alt`+`F1` ～ `F6` を押すことで仮想端末に切り替えることができます。グラフィカルインターフェイスを備えたマルチメディアシステムをセットアップするように求められたので、不正利用を防ぐためにこの機能を無効にする必要があります。`/etc/X11/xorg.conf.d/10-kiosk.conf` に設定ファイルを作成することにします。`ServerFlags` セクション（Xorgのグローバルオプションを置きます）に指定するオプションは何ですか？ `xorg(1)` のマニュアルページを参照して、適切なオプションを見つけてください。

まとめ

このレッスンでは、Linuxで使用されるX Window Systemについて説明しました。X Window Systemは、さまざまな設定ファイルの定義に応じて、画面に画像やテキストを描画します。マウスやキーボードなどの入力デバイスの設定は、X Window Systemによって行われます。このレッスンでは、以下の項目を説明しました。

- X Window System のアーキテクチャ概要
- X Window System 設定ファイルと、それらのファイルシステム上の位置
- `DISPLAY` 環境変数の使い方
- Wayland 簡単な紹介

以下のコマンドと設定ファイルについて説明しました:

- Xorgを使用するマシンにおいて、`setxkbmap` や `localectl` を使用してキーボードレイアウトを変更すること。
- 新しい `/etc/X11/xorg.conf` 設定ファイルを作成する、`Xorg` コマンドのオプション。
- 重要なXorg設定ファイルの内容： `/etc/X11/xorg.conf`、`/etc/X11/xorg.conf.d/`、`/usr/share/X11/xorg.conf.d/`。
- Xサーバーセッションに関する情報を表示する `xdpyinfo` コマンド。

演習の解答

1. システムで使用できるXorg拡張機能を確認するコマンドは何ですか?

   ```
   $ xdpyinfo
   ```

2. コンピュータ用の新しい10ボタンマウスを入手しましたが、すべてのボタンを正しく機能させるには、追加の設定が必要です。Xサーバー設定の他の部分に影響をあたえずに、このマウス用の新しい設定ファイルをどのディレクトリに作成し、どのセクションに記入しますか？

   ユーザー定義の設定は `/etc/X11/xorg.conf.d/` にあり、マウス設定を行うセクションは `InputDevice` です。

3. Linuxシステムにおいて、Xサーバーの実行を維持するコンポーネントは何ですか？

   ディスプレイマネージャーです。

4. 新しい `xorg.conf` ファイルを作成する、`X` コマンドのオプションは何ですか？

   ```
   -configure
   ```

   `X` コマンドは `Xorg` コマンドへのシンボリックリンクです。

発展演習の解答

1. 3台のモニタを1つのディスプレイとして使用するように構成された `lab01` というシステムを使用しています。3つ目のスクリーンに開いたリモートマシンのターミナルエミュレータで、`DISPLAY` 環境変数の値を表示するとどうなりますか？

```
$ echo $DISPLAY
lab01:0.2
```

2. X Window Systemで使用するキーボード設定ファイルを作成するコマンドは何ですか？

```
$ localectl
```

3. 一般的なx86_64アーキテクチャのマシンにインストールしたLinuxでは、キーボードから `Ctrl`+`Alt`+`F1` ～ `F6` を押すことで仮想端末に切り替えることができます。グラフィカルインターフェイスを備えたマルチメディアシステムをセットアップするように求められたので、不正利用を防ぐためにこの機能を無効にする必要があります。`/etc/X11/xorg.conf.d/10-kiosk.conf` に設定ファイルを作成することにします。`ServerFlags` セクション（Xorgのグローバルオプションを置きます）に指定するオプションは何ですか？ `xorg(1)` のマニュアルページを参照して、適切なオプションを見つけてください。

```
Section "ServerFlags"
    Option  "DontVTSwitch"  "True"
EndSection
```


106.2 グラフィカルデスクトップ

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 106.2

総重量

1

主な知識分野

- 主要なデスクトップ環境の知識
- リモートデスクトプセッションへアクセスするためのプロトコルの知識

用語とユーティリティ

- KDE
- Gnome
- Xfce
- X11
- XDMCP
- VNC
- Spice
- RDP

106.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	106 ユーザーインターフェイスとデスクトップ
Objective:	106.2 グラフィカルデスクトップ
Lesson:	1 of 1

はじめに

LinuxベースのOSは、高度なコマンドラインインターフェイスで知られていますが、技術者でないユーザーには敷居が高いものです。コンピューターをより直感的に使うために、高解像度ディスプレイとポインティングデバイスを組み合わせて利用する、グラフィカルユーザーインターフェースが生まれました。コマンドラインインターフェースではコマンド名とそのオプションに関する予備知識が必要ですが、グラフィカルユーザーインターフェース（GUI）では身近な図形をポイントしてプログラムの機能を選択できるので、学習曲線は緩やかなものとなります。さらにグラフィカルインターフェースは、マルチメディアなどの視覚的な作業に最適です。

実際のところ、グラフィカルユーザーインターフェイスはコンピューターインターフェイスとほぼ同義であり、ほとんどのLinuxディストリビューションにはデフォルトでグラフィカルインターフェイスがインストールされています。ただし、Linuxシステムでは、グラフィカルデスクトップの全機能を担当する、巨大な1つのプログラムは使いません。それぞれのグラフィカルデスクトップは、依存関係をもったさまざまなプログラムのコレクションであり、ディストリビューターやユーザーの個人的な好みによって選択されます。

X Window System

Linuxと他のUnix系オペレーティングシステムで採用されている、X Window System（X11 もしくは単に X とも呼ばれます）は、グラフィカルインターフェイスの表示や、ユーザー操作に対する低レベルのリソースを提供します。

- マウス操作やキー入力などの入力イベントの処理。
- アプリケーション間で、テキストのカット、コピー、ペースト。

- プログラムがグラフィック要素を描画するためのプログラミングインターフェイス。

X Window Systemはグラフィックディスプレイの制御を担当（ビデオドライバー自体もXの一部）しますが、それ自体で複雑な視覚要素を単独で描画することを意図したものではありません。形状、色、陰影などの視覚効果は、X上で実行されるアプリケーションが生成します。このアプローチによって、アプリケーションがさまざまにカスタマイズされたインターフェイスを実現できますが、アプリケーションの本質とは異なる開発オーバーヘッドや、他のプログラムのインターフェイスと異なる見た目や動作の不一致を引き起こす可能性もあります。

デスクトップ環境を導入すると、開発者の立場からはアプリケーション開発におけるGUIプログラミングが容易になり、ユーザーの立場からはさまざまなアプリケーション間での一貫した使用感が得られます。デスクトップ環境は、プログラミングインタフェース、ライブラリ、サポートプログラムなどを統合し、伝統的なデザインコンセプトを継承しながら進化しています。

デスクトップ環境

伝統的なデスクトップコンピュータのGUIは、実行中のプロセスに関連付けられた　ウィンドウ（アプリケーションに与えられた画面領域）で構成されます。X Window Systemは基本的な対話機能しか提供しないので、デスクトップ環境に内蔵されたコンポーネントがユーザー体験のすべてを提供します。

デスクトップ環境におけるおそらく最も重要な構成要素である　ウィンドウマネージャー　は、ウィンドウの配置と装飾を制御します。ウィンドウマネージャーは、ウィンドウにタイトルバーとコントロールボタン（最小化、最大化、閉じるなどのアクション）を追加し、ウィンドウの切り替えを管理します。

NOTE

デスクトップコンピュータにおけるグラフィックインターフェースの基本コンセプトは、現実のオフィスからヒントを得ています。　例えば、コンピュータの画面は机であり、そこには文書やフォルダなどのオブジェクトが置かれます。ドキュメントの内容が表示されたアプリケーションウィンドウは、フォームに記入したり図を描くといった行為を模しています。コンピュータデスクトップにはメモ帳、時計、カレンダーなどのソフトウェアアクセサリがありますが、それらのほとんどは実際の机に基づいています。

すべてのデスクトップ環境は、ウィジェット　ツールキットのルック＆フィールに合わせたウィンドウマネージャを備えています。ウィジェットとは、アプリケーションウィンドウ内に配置されるボタンやテキスト入力フィールドなど、情報提供や対話用の視覚的要素です。アプリケーションランチャー、タスクバーなどの標準的なデスクトップコンポーネントや、ウィンドウマネージャそのものは、ウィジェットツールキットを使用してインターフェースを構築しています。

GTK+　や　Qt　などのソフトウェアライブラリが、精巧なグラフィカルインターフェイスを構築するために使用するウィジェットをアプリケーションに提供します。以前はGTK+で開発されたアプリケーションとQtで作成されたアプリケーションはかなり異なって見えましたが、現在のデスクトップ環境ではテーマのサポートが強化されたので、違いが目立たなくなっています。

GTK+とQtはいずれも、ほぼ同じ機能のウィジェットを提供します。単純な操作パーツは見分けがつきませんが、複合的なウィジェット～例えばファイル選択のためのダイアログウィンドウなどはかなり違って見えます。とはいえ、異なるツールキットで作ったアプリケーションを混在して同時に実行できます。

デスクトップ環境は、それぞれが独立したプログラムである基本的なデスクトップコンポーネントと、同じ設計ガイドラインに沿って開発された一連の小さなアクセサリアプリケーションを提供することで、デスクトップのメタファーを実現します。すべての主要なデスクトップ環境で、次のようなアプリケーションが提供されています。

システム関連のアプリケーション
ターミナルエミュレーター、ファイルマネージャー、パッケージインストールマネージャー、システム構成ツール。

コミュニケーションとインターネット
アドレス帳、電子メールクライアント、Webブラウザー。

オフィスアプリケーション
カレンダー、電卓、テキストエディタ。

デスクトップ環境が、ログイン画面のグリーター、セッションマネージャー、プロセス間通信、キーリングのエージェントなど、デスクトップ以外のサービスやアプリケーションを含んでいることがあります。サウンド用の PulseAudio や、印刷用の CUPS など、サードパーティのシステムサービスが提供する機能も組み込まれています。これらの機能はグラフィカル環境がなくても動作しますが、デスクトップ環境がそれらリソースの操作や設定を行うためのグラフィカルなフロントエンドを提供します。

一般的なデスクトップ環境

ほとんどのプロプライエタリなオペレーティングシステムは、リリース毎に固有の公式デスクトップ環境を1つだけサポートしています（変更はできません）。それに対して、Linuxベースのオペレーティングシステムは、Xと組み合わせて使用できるさまざまなデスクトップ環境をオプションでサポートしています。それぞれのデスクトップ環境には独自の機能がありますが、一般的なコンセプトは共通しています。

- 利用できるアプリケーションを一覧表示するアプリケーションランチャー。
- ファイル種別やプロトコルに関連付ける、デフォルトのアプリケーションを定義するルール。
- デスクトップ環境の外観や動作をカスタマイズするためのツール。

Gnome は最も人気のあるデスクトップ環境の1つであり、Fedora、Debian、Ubuntu、SUSE Linux Enterprise、Red Hat Enterprise Linux、CentOSなどのディストリビューションのデフォルトです。バージョン3では、デスクトップのメタファーを捨てて Gnome Shell を導入するという、外観と構造の大きな変更を行いました。

Figure 1. Gnome ShellのActivities

従来のアプリケーションランチャーとタスクバーは、Gnomeの汎用的な全画面ランチャーであるActivities で置き換えられました。ただし、ログイン画面で Gnome Classic オプションを選択することで、古い外観のGnome 3を使用することもできます。

KDE は、アプリケーションと開発プラットフォームの大規模なエコシステムです。最新のデスクトップ環境である KDE Plasma は、openSUSE、Mageia、Kubuntuなどのディストリビューションのデフォルトです。Qtライブラリの採用がKDEの大きな特徴で、独特の外観と多くのオリジナルアプリケーションを備えています。KDEには、見た目をGTK+アプリケーションに見せかけるためのツールも用意されています。

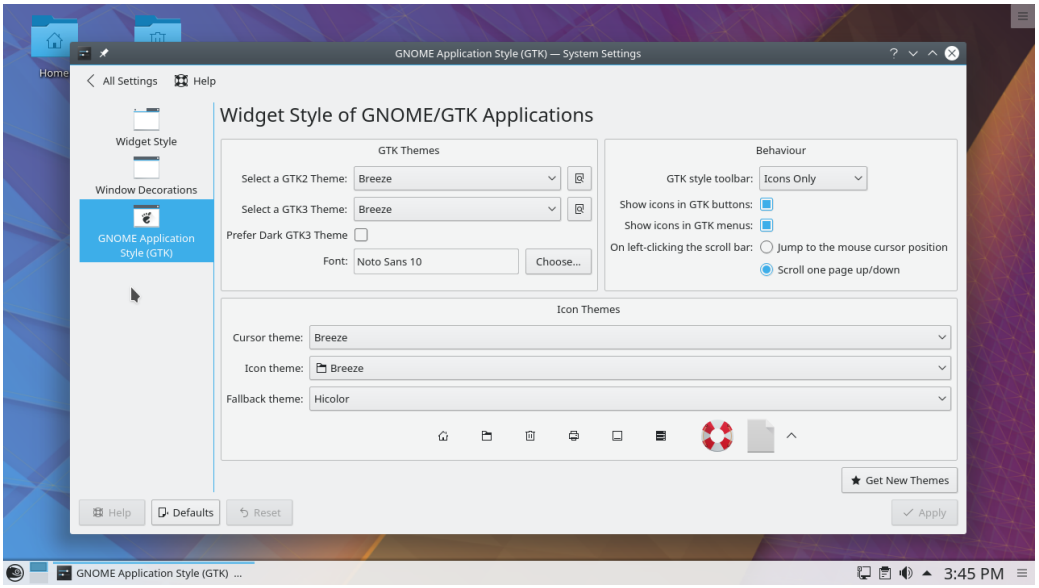


Figure 2. KDEにおけるGTKの設定画面

Xfce は、必要とするマシンリソースが少ないことと、美しい見た目を目標としたデスクトップ環境です。その構造は高度にモジュール化されており、ユーザーはニーズや好みに応じてコンポーネントを有効化ないし無効化できます。

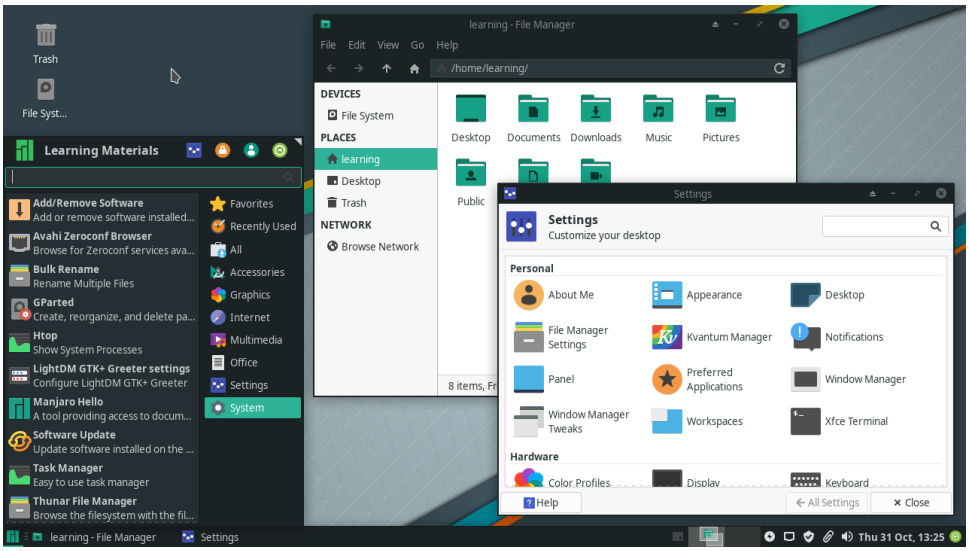


Figure 3. Xfceのデスクトップ

Linuxには他にも多くのデスクトップ環境があり、通常は代替ディストリビューションとして提供されます。たとえば、Linux Mintディストリビューションでは、Cinnamon（Gnome 3のフォーク）とMATE（Gnome 2のフォーク）の2つのオリジナルなデスクトップ環境が提供されます。LXDE は、リソース消費が少ないデスクトップ環境であるため、古い機器やシングルボードコンピューターへのインストールに適しています。重厚な（フルセットの）デスクトップ環境と同じ機能を提供するわけではありませんが、LXDEは現代的なグラフィカルユーザーインターフェイスに相応しいすべての基本機能をを備えています。

TIP

デスクトップ環境との対話では、キーボードショートカットが重要な役割を果たします。ウィンドウを切り替える`Alt`+`Tab`や、テキストをコピーする`Ctrl`+`c`などのキーボードショートカットはどのデスクトップ環境でも共通ですが、デスクトップ環境ごとに独自のキーボードショートカットがあります。デスクトップ環境に備わっているキーボード設定ツールを使って、ショートカットを追加、変更できます。

デスクトップの相互運用性仕様

Linux系のオペレーティングシステムのデスクトップ環境は多様であるため、デスクトップ環境それぞれに固有のサポートを実装することなく、異なるデスクトップ環境用のグラフィカルアプリケーションやシステムサービスを、異なるデスクトップ環境でも正しく動作させることが課題となっています。あるデスクトップ環境用に設計されたグラフィカルアプリケーションを、異なるデスクトップ環境でも動作させるためには、デスクトップ環境の仕様や呼び出し方法を共通化する必要があります。また、ユーザーが別のデスクトップ環境に切り替えた場合に、デスクトップの一般的な設定を維持することも重要な課題です。

freedesktop.org という組織が、デスクトップの互換性に関わる多くの仕様を取りまとめています（訳注: XDG Interoperability Specification と呼ばれ、freedesktop.org のWikiページから参照できます)。すべての仕様を満たす必要はありませんが、多くの仕様が広く採用されています。代表的な内容を示します。

ベースディレクトリ
: 各ユーザーの設定やユーザー固有のファイルを置くディレクトリ。

Desktop Entry
: デスクトップ環境でも、ターミナルエミュレーター経由でCLIアプリケーションを実行することができます。しかし、アプリケーションランチャーにそれらすべてを表示すると、ユーザーは混乱してしまうでしょう。デスクトップ環境が、利用できるデスクトップアプリケーションに関する情報を見つけられるように、サフィックスが `.desktop` であるテキストファイルの規格が定められています。

自動起動
: ユーザーがログインした後に自動的に起動する必要があるアプリケーションを示すDesktop Entry。

ドラッグアンドドロップ
: アプリケーションがドラッグ＆ドロップイベントを処理する方法。

ごみ箱
: ファイルマネージャによって削除されたファイルの一般的な保存場所と、そこからファイルを保存・削除する方法。

アイコンテーマ
: 互換性のあるアイコンライブラリの共通フォーマット。

シェルなどのテキストインターフェイスに比べると、デスクトップ環境ではリモートアクセス機能が欠けていることが欠点です。`ssh` などのツールでリモートマシンのシェル環境に容易にアクセスできるのに対して、グラフィック環境へのリモートアクセスには異なる方法が必要ですし、（データ量が多いため）低速な回線では満足なパフォーマンスを得ることができません。

リモートアクセス

X Window Systemsは、それぞれの ディスプレイ が独立して動作するようにデザインされていて、1つの ディスプレイマネージャ が、同時に1つ以上のグラフィカルなデスクトップセッションを制御できます。つまり、テキストターミナルと同様に、それぞれのディスプレイでオペレーティングシステムに対するセッションを確立し、マシンやアプリケーションに対する窓口とすることができます。通常はローカルマシンに対する1つのグラフィカルセッションを起動する設定が一般的ですが、以下のような設定を行う事もできます:

- 1つのマシンで、アクティブなグラフィカルデスクトップセッションを切り替える。
- 1つのマシンに複数のディスプレイデバイスセット（画面、キーボード、マウスなど）を接続し、それぞれがグラフィカルデスクトップセッションを制御する。
- リモートグラフィカルデスクトップセッション。すなわち、グラフィカルインターフェイスをネットワーク経由でリモートディスプレイに送信する。

Xはリモートデスクトップセッションを標準サポートしていて、XDMCP（X Display Manager Control Protol）を使用して、リモートのディスプレイとやり取りします。XDMCPは帯域幅を使用するので、インターネットや低速LANで使用されることはめったにありません。セキュリティの問題もXDMCPの懸念事項です。すなわち、ローカルディスプレイは特権を持つリモートXディスプレイマネージャーと通信してリモートプロシージャを実行するので、脆弱性があるとリモートマシンで特権コマンドを実行される可能性があります。

さらに、XDMCPでは接続の両側でXインスタンスを実行する必要があるため、X Window Systemを使用できないマシンでは利用できません。実際のところ、（XDMCPではなく）他のより効率的で安全な方法を使って、リモートとのグラフィカルデスクトップセッションを確立するのが一般的です。

VNC（Virtual Network Computing）は、RFB（Remote Frame Buffer）プロトコルを使用して、リモートのデスクトップ環境を表示および制御する、プラットフォームに依存しないツールです。ローカルのキーボードやマウスが生成したイベントをリモートデスクトップに送信し、画面の更新をローカルで表示するために返送します。1台のマシンで多数のVNCサーバーを実行することができますが、それぞれのVNCサーバーは、着信要求を受け入れるために専用のTCPポートを必要とします。慣例では、最初のVNCサーバーがTCPポート 5900を使用し、2番目のVNCサーバーが5901を使用します。

VNCサーバーを特権を持ったプロセスとしてで実行する必要はありません。一般ユーザーとしてリモートアカウントにログインして、そこで自分のVNCサーバーを起動できます。次に、ローカルマシンで好みのVNCクライアントアプリケーションを起動し、リモートデスクトップにアクセスします（対応するネットワークポートに到達できることが必要です）。`~/.vnc/xstartup` ファイルは、VNCサーバーが起動時に実行するシェルスクリプトで、VNCサーバーがVNCクライアントに提供するデスクトップ環境を定義します。VNCは最新の暗号化と認証方法を備えていないので、それらの機能を提供する別のアプリケーションと組み合わせて使用する必要があります。VCN接続を保護するためには、VPNやSSHトンネルがよく使われます。

RDP（Remote Desktop Protocol）は、主にTCP 3389番ポートを使って、Microsoft Windows のデスクトップにリモートアクセスするために使用されます。Microsoft社独自のRDPプロトコルを使用しますが、Linuxシステムで使用されるクライアント実装は GNU General Public License（GPL）でライセンスされるオープンソースプログラムで、使用に関する法的な制限は一切ありません。

Spice（Simple Protocol for Independent Computing Environments）は、ローカルないしリモートで 仮想化されている システムのデスクトップ環境にアクセスするためのツール群です。Spiceプロトコルは、リモートマシンからローカルデバイス（スピーカーやUSBデバイスなど）にアクセスする機能や、リモート/ローカル間でのファイル共有など、ローカルシステムとリモートシステムを統合する機能も備えています。

これらのリモートデスクトッププロトコルのそれぞれに接続する専用のクライアントコマンドがありますが、Remmina というリモートデスクトップクライアントを使用すると、統合されたグラフィカルインターフェイスを通して（それぞれのプロトコルに）接続し、その設定を保存しておくことができます。Remminaでは、プロトコルごとにプラグインがあり、XDMCP、VNC、RDP、Spice用のプラグインを提供しています。オペレーティングシステムやネットワーク接続の品質、リモートデスクトップ環境で利用する機能などに応じて、適切なツールを選択します。（訳注: Remmina は多くのLinuxディストリビューションでパッケージが提供されています。）

演習

1. デスクトップ環境でシェルセッションのウィンドウを提供するアプリケーションは何ですか？

2. Linuxデスクトップ環境はさまざまであるため、ウィジェットツールキットに応じて、1つのアプリケーションに複数のバージョンが存在することがあります。例えば、BitTorrentのクライアントであるTransmission には、transmission-gtk と transmission-qt の2つのバージョンがあります。KDEを最大限に活用するには、2つのうちどちらをインストールしますか？

3. 処理能力が低い安価なシングルボードコンピューターに推奨されるLinuxデスクトップ環境は何ですか？

発展演習

1. X Window System でテキストをコピーして貼り付けるには、Ctrl+C および Ctrl+V キー（ないしウィンドウメニュー）を使用する方法と、マウスの中央ボタンをクリックして選択中のテキストを張り付ける方法の2つがあります。ターミナルエミュレータで、テキストをコピーして貼り付けるには、どちらの方法が適切ですか？

2. ほとんどのデスクトップ環境では、ショートカット Alt+F2 が Run program ウィンドウに割り当てられています。このウィンドウでは、プログラムをコマンドライン形式で実行できます。KDE でデフォルトのターミナルエミュレータを実行するコマンドは何ですか？

3. Linux デスクトップ環境からリモートの Windows デスクトップにアクセスするのに最適なプロトコルはどれですか？

まとめ

このレッスンでは、Linuxシステムで使用できるグラフィカルデスクトップの概要を説明しました。X Window Systemだけでは、単純なインターフェイス機能しか提供されませんが、デスクトップ環境ではグラフィカルウィンドウインターフェイスのユーザー体験が拡張されます。レッスンでは、以下のトピックについて説明しました。

- グラフィックインターフェイスとXウィンドウシステムの概念。
- Linuxで利用可能なデスクトップ環境。
- デスクトップ環境の類似点と相違点。
- リモートのデスクトップ環境にアクセスする方法。

取り上げた概念とプログラムは以下のとおりです:

- X Window System。
- 一般的なデスクトップ環境: KDE、Gnome、Xfce。
- リモートアクセスプロトコル: XDMCP、VNC、RDP、Spice。

演習の解答

1. デスクトップ環境でシェルセッションのウィンドウを提供するアプリケーションは何ですか？

   Konsole、Gnome terminal、xtermなどのターミナルエミュレーターが、ローカルの対話型シェルセッションへのアクセスを提供します。

2. Linuxデスクトップ環境はさまざまであるため、ウィジェットツールキットに応じて、1つのアプリケーションに複数のバージョンが存在することがあります。例えば、BitTorrentのクライアントであるTransmission には、transmission-gtk と transmission-qt の2つのバージョンがあります。KDEを最大限に活用するには、2つのうちどちらをインストールしますか？

   KDEはQtライブラリの上に構築されているので、Qtバージョン 〜transmission-qt 〜をインストールします。

3. 処理能力が低い安価なシングルボードコンピューターに推奨されるLinuxデスクトップ環境は何ですか？

   XfceやLXDEなど、視覚効果をあまり使用しない簡素なデスクトップ環境。

発展演習の解答

1. X Window System でテキストをコピーして貼り付けるには、Ctrl+C および Ctrl+V キー（ないしウィンドウメニュー）を使用する方法と、マウスの中央ボタンをクリックして選択中のテキストを張り付ける方法の2つがあります。ターミナルエミュレータで、テキストをコピーして貼り付けるには、どちらの方法が適切ですか？

   対話的なシェルセッションでは、Ctrl+C でプログラムの実行が停止されるため、中央ボタンの方法がお勧めです。

2. ほとんどのデスクトップ環境では、ショートカット Alt+F2 が Run program ウィンドウに割り当てられています。このウィンドウでは、プログラムをコマンドライン形式で実行できます。KDE でデフォルトのターミナルエミュレータを実行するコマンドは何ですか？

   KDEのターミナルエミュレータを実行するには、`konsole` コマンドを実行しますが、terminal など一般的な用語でも大丈夫です。

3. Linuxデスクトップ環境からリモートのWindowsデスクトップにアクセスするのに最適なプロトコルはどれですか？

   RDP（Remote Desktop Protocol）です。WindowsとLinuxの両方でサポートされています。

106.3 アクセシビリティ

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 106.3

総重量

1

主な知識分野

- ビジュアル設定とテーマの基礎知識。
- 補助技術の基礎知識。

用語とユーティリティ

- 高コントラクト、ラージプリントデスクトップテーマ
- Screen Reader.
- Braille Display.
- Screen Magnifier.
- On-Screen Keyboard.
- Sticky/Repeat keys.
- Slow/Bounce/Toggle keys.
- ジェスチャー
- 音声認識

106.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	106 ユーザーインターフェイスとデスクトップ
Objective:	106.3 アクセシビリティ
Lesson:	1 of 1

はじめに

Linuxデスクトップ環境には、障がいを持つ人々が使いやすいユーザーインターフェイスを実現する多くの設定とツールがあります。通常のヒューマンインターフェイスデバイス（画面、キーボード、マウス/タッチパッド）を、視覚や運動の障がいに合わせて設定することが可能です。

たとえば、デスクトップの配色を調整して、色覚異常の人に使いやすいサービスを提供することができます。また、反復運動損傷（訳注:　何度も同じ動きを繰り返すことで、手指に痛みや痺れ・震えなどが起きる症状。スマホ指、ゲーマー指などとも言う。）の人に優しい、タイピングやポインティング方法に変更することもできます。

これらのアクセシビリティ機能は、GnomeやKDEなどのデスクトップ環境自体が提供しているものもありますし、追加のプログラムが提供しているものもあります。後者の場合、デスクトップ環境と統合されたツールを選択することが重要で、より質の高い支援が行えます。

アクセシビリティ設定

すべての主要なLinuxディストリビューションは、ほぼ同じアクセシビリティ機能を提供しており、デスクトップ環境に付属する設定マネージャでカスタマイズできます。アクセシビリティ設定モジュールは、Gnomeデスクトップでは　Universal　Access、KDEでは　System　Settings→Personalization→Accessibility　にあります（訳注:　メニュー項目とその訳語はディスリビューションによって異なりますので、あえて訳出していません）。Xfce　などの他のデスクトップ環境でも、グラフィック設定マネージャに　Accessibility　という項目がありますが、GnomeやKDEに比べると機能が少なくなっています。

例えばGnomeでは、画面右上にユニバーサルアクセスメニューを常時表示し、アクセシビリティ機能を素早く切り替えられるように設定できます。例えば、音による警告を視覚的なものに置き換えて、聴覚

障害のあるユーザーがシステムからの警告をより簡単に知覚できるようにします。KDEには同様のクイックアクセスメニューはありませんが、視覚的な警告に切り替えることができて、visual bell という名前で提供されています。

キーボードとマウスの補助

ある種の動作障がいを回避するように、キーボードとマウスのデフォルト動作を変更することができます。手に動作障がいがあるユーザーには、複数キーの同時押し下げや、キーの自動リピート、（震えによる）意図しないキー入力などが問題となることがあります。それらの対策として、キーボード関連のアクセシビリティ機能、Sticky keys、Bounce keys、Slow keys が用意されています。

Gnomeでは、Universal Accessの Typing Assist セクションにある sticky keys を使用すると、複数のキーを1つずつ押すことでキーボードショートカットを入力できます。（訳注: stickyは「べとべとする」「ねばつく」と言った意味の英単語です）。有効にすると、`Ctrl`+`C` のような組み合わせキーを同時に押す必要がなくなります。まず `Ctrl` キーを押して放し、次に `C` キーを押せばよいのです。KDEでは、アクセシビリティ設定の Modifier Keys タブにあります。KDEには Locking Keys オプションも用意されていて、これを有効にすると、Caps Lockキーの動作と同様に、`Alt`、`Ctrl`、`Shift` を2回押すと、“押し下げた” ままの状態を保持します。同じキーをもう一度押して解放する必要があることも、Caps Lockと同じです。

Bounce keys は、次のキーの押し下げ検出を遅延させることで、（手の震えによる）意図しない入力を抑制する機能です。つまり、最後にキーを押してから指定した時間が経過しないと、新しいキーの押し下げを受け付けません。1文字入力するつもりで、何度もキーを押してしまうことを避けるために、Bounce keys機能が役立ちます。Gnomeの場合は同じキーを繰り返し押した場合のみに機能しますが、KDE では異なるキーを押し下げた場合にも機能させることができて、Keyboard Filter タブで指定します。

Slow keys 機能は、偶発的なキー入力を回避します。Slow keysを有効にすると、キーを指定した時間押し続けないと入力が受け付けられません。ユーザーの状態によっては、キーを押し続けた場合のリピート入力時間を調整することも有用です。

Sticky keysとSlow Keys機能は、キーボードの ジェスチャー でオンとオフを切り替えることができます。Gnomeでは Typing Assist 設定画面の Enable by Keyboard をチェックし、KDEでは Use gestures for activating sticky keys and slow keys をチェックすると、ジェスチャーが有効になります。ジェスチャーを有効にすると、シフトキーを連続5回押すとSticky keysが有効になり、シフトキーを8秒間連続して押し続けるとSlow Keysが有効になります。

マウスやタッチパッドよりもキーボードを使用する方が快適なユーザーは、キーボードショートカットを使用してデスクトップ環境を操作できます。また、Mouse Keys と呼ばれる機能を使って、フルサイズのデスクトップキーボードや大型のラップトップに搭載されているテンキーパッドを使用して、マウスポインターを制御できます。

テンキーは正方形に配置されているので、数字がそれぞれの方向に対応します。`2`はカーソルを下に、`4`はカーソルを左に、`7`はカーソルを左上に、などにそれぞれ移動します。デフォルトでは、`5`キーがマウスの左クリックに対応します。

Gnomeでは Universal Access setting ウィンドウにマウスキーオプションを有効にするスイッチがあるだけですが、KDEでは System settings→Mouse→Keyboard Navigation にあり、速度や加速度などのオプションをカスタマイズできます。

TIP Slow keys、Sticky keys、Bounce kyes、Mouse Keysといったアクセシビリティ機能は、X Window Systemのキーボード拡張におけるAccessXリソースによって設定しま

> す。AccessXの設定は、コマンドラインから `xkbset` コマンドを使用して変更することもできます。

キーボードが使えない場合や面倒な場合には、マウスやタッチパッドを使ってキーボード入力を生成できます。Gnomeのユニバーサルアクセス設定で Screen keyboard を有効にすると、テキストフィールドにカーソルがある時に仮想キーボードが現れて、マウスやタッチパネルでキーをクリックすることで文字を入力できます。スマートフォンの仮想キーボードと似た機能です。

KDEなどのデスクトップ環境では、デフォルトではScreen keyboardが提供されていない場合がありますが、onboard パッケージを手動でインストールすると、任意のデスクトップ環境で使用できるシンプルなScreen keyboardを利用できます。インストールすれば、アプリケーションランチャーから通常のアプリケーションとして利用できます。

マウスのクリックやドラッグが困難な場合などには、ポインターの動作を変更することもできます。例えば、マウスボタンを素早くダブルクリックできない場合には、システム設定ウィンドウの マウス設定で、ダブルクリックするためにマウスボタンを2回押す時間を長く設定することができます。

マウスのボタンを押すことができない場合は、マウスクリックをシミュレートする方法があります。Gnome Universal Access の Click Assist セクションの Simulate a right mouse click（マウスの右クリックをシミュレート） オプションは、マウスの左ボタンを押し続けると右クリックを生成します。Simulate clicking by hovering（ホバーによるクリックのシミュレート） オプションを有効にすると、マウスを（ボタンなどの上で）静止させると、クリックイベントを生成します。KDEでは、KMouseTool アプリケーションが、マウスアクションを支援する同様の機能を提供します。

視覚障害

視力が低下している場合でも、モニター画面を使ってコンピュータを操作することができます。ニーズに応じて、多くの視覚調整を行い、標準的なグラフィカルデスクトップでは詳細が見えにくい部分を改善することが可能です。

Gnomeでは、Universal Access 設定の Seeing （外観）セクションに、視力が低下している人々を補助するオプションが提供されています。

ハイコントラスト
ウィンドウとボタンをより鮮明な色で描画して見やすくします。

大きなテキスト
デフォルトの画面フォントサイズを大きくします。

カーソルサイズ
マウスカーソルを大きくして、画面上で見つけやすくします。

アクセシビリティと直接関係ない機能のため、多くのデスクトップ環境では、設定ユーティリティの appearance（見た目）セクションで調整します。視覚的な要素の識別が困難なユーザーは、コントラストの高いテーマを選択して、ボタンや重なったウィンドウなどを区別しやすくできます。

外観の調整だけでは不十分な場合は、画面用虫めがね（拡大鏡）を使って画面の一部を拡大できます。Gnomeの Universal Access 設定では Zoom と呼ばれていて、拡大率や表示位置、色調整などのオプションをカスタマイズできます。

KDEでは、ランチャーから起動する通常のアプリケーションである KMagnifier プログラムが同じ機能を提供します。他のデスクトップ環境でも、独自の虫めがね機能が提供されています。たとえばXfce

では、`Alt` キーを押しながらマウスのスクロールホイールを回転させると、画面をズームイン/ズームアウトします。

最後に、グラフィカルインターフェイスを使用できないユーザーは、スクリーンリーダー　　を使用してコンピューターとやり取りすることができます。どのデスクトップ環境を使っているかに関係なく、Linuxシステムで最も一般的なスクリーンリーダーは　Orca　で、ほとんどのディストリビューションにデフォルトでインストールされています。Orcaは画面上のイベントを合成音声でレポートし、マウスカーソルの下にあるテキストを読み上げます。Orcaは、点字端末（指先で触ることができる小さなピンを持ち上げて点字を表示する特殊なデバイス）とも連携します。すべてのデスクトップアプリケーションがスクリーンリーダーに対応しているわけではありませんし、ユーザーもさまざまですから、多くの画面「表現」方法を提供することが重要です。

演習

1. `Alt`キーと`Tab`キーを同時に押すことができない場合に、開いているウィンドウを切り替えるのに役立つアクセシビリティ機能は何ですか？

2. 手の震えがタイピングを妨げるユーザーに対して、Bounce keys アクセシビリティ機能はどのように役立ちますか？

3. Sticky keys アクセシビリティ機能を有効化する、一般的なジェスチャーは何ですか？

発展演習

1. アクセシビリティ機能は、ひとつのアプリケーションだけで提供されるとは限らず、デスクトップ環境ごとに異なります。KDEにおいて、マウスカーソルが停止するたびにマウスをクリックすることで、反復運動損傷のある人を支援するアプリケーションは何ですか？

2. 画面上のテキストを読みやすくするために、グラフィカル環境の外観をどのように変更すればよいでしょう？

3. 視覚障がい者がデスクトップ環境を操作する場合に、Orca　　アプリケーションはどのように役立ちますか？

まとめ

このレッスンでは、Linuxシステムで利用できる一般的なアクセシビリティ機能について説明しました。主なデスクトップ環境、特にGnomeとKDEは、視覚障害や運動障害のある人をサポートするために、多くの組み込みアプリケーションとサードパーティのアプリケーションを提供しています。レッスンでは、次のトピックを説明しました。

- アクセシビリティの設定を変更する方法
- キーボードとマウスを使用する別の方法
- 視覚障害者向けのデスクトップ拡張機能

以下のコマンドと手順を紹介しました:

- キーボードのアクセシビリティ設定: Sticky keys、Slow keys、Bounce keys
- マウスイベントの自動生成
- 仮想（スクリーン）キーボード
- 読みやすさを向上させるためのビジュアル設定
- 高コントラストや、大きなフォントのデスクトップテーマ
- 虫めがね（拡大鏡）
- Orcaスクリーンリーダー

演習の解答

1. `Alt`キーと`Tab`キーを同時に押すことができない場合に、開いているウィンドウを切り替えるのに役立つアクセシビリティ機能は何ですか？

   Sticky kyes。キーを1つずつ押し下げることで、キーボードショートカットを入力します。

2. 手の震えがタイピングを妨げるユーザーに対して、Bounce keys アクセシビリティ機能はどのように役立ちますか？

   Bounce keysを有効にすると、最後にキーを押してから所定の時間が経過した後に、次のキーの押し下げを受け付けます。

3. Sticky keys アクセシビリティ機能を有効化する、一般的なジェスチャーは何ですか？

   ジェスチャーが有効になっている場合に、`Shift`キーを連続して5回押すと、Sticky keys機能がアクティブになります。

発展演習の解答

1. アクセシビリティ機能は、ひとつのアプリケーションだけで提供されるとは限らず、デスクトップ環境ごとに異なります。KDEにおいて、マウスカーソルが停止するたびにマウスをクリックすることで、反復運動損傷のある人を支援するアプリケーションは何ですか？

   KMouseTool アプリケーション。

2. 画面上のテキストを読みやすくするために、グラフィカル環境の外観をどのように変更すればよいでしょう？

   デスクトップ構成でフォントサイズを大きくすると、すべてのテキストが読みやすくなります。

3. 視覚障がい者がデスクトップ環境を操作する場合に、Orca　　アプリケーションはどのように役立ちますか？

   Orcaはスクリーンリーダーで、合成音声で画面のイベントを知らせたり、マウスカーソルの下のテキストを読み上げたりします。また、点字ディスプレイと呼ばれる装置と連動し、触覚パターンでテキストを識別することができます。

Linux
Professional
Institute

課題 107: 管理タスク

107.1 ユーザーおよびグループアカウントと関連するシステムファイルを管理する

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 107.1

総重量

5

主な知識分野

- ユーザーとグループの追加、変更、削除。
- パスワード/グループデータベースのユーザー/グループ情報を管理する。
- 専用アカウントと限定アカウントの作成と管理。

用語とユーティリティ

- `/etc/passwd`
- `/etc/shadow`
- `/etc/group`
- `/etc/skel/`
- `chage`
- `getent`
- `groupadd`
- `groupdel`
- `groupmod`
- `passwd`
- `useradd`
- `userdel`
- `usermod`

107.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	107 管理タスク
Objective:	107.1 ユーザーとグループのアカウント管理と関連ファイル
Lesson:	1 of 2

はじめに

ユーザーとグループの管理は、システム管理者の非常に重要な業務です。最新のLinuxディストリビューションは、この重要な業務に関わるすべてのアクティビティを簡単かつ効率的に行うためのグラフィカルインターフェイスを備えています。それらのツールは、グラフィカルな表現は異なっていますが機能的にはほぼ同じで、ローカルユーザーとグループを表示、編集、追加、および削除できます。ただし、高度な管理を行うには、コマンドラインを使用する必要があります。

ユーザーアカウントの追加

Linuxでは、`useradd` コマンドを使用して新しいユーザーアカウントを追加します。たとえば、`michael` という名前の新しいユーザーアカウントをデフォルト設定で作成するには、root権限で次のコマンドを実行します。

```
# useradd michael
```

`useradd` コマンドを実行すると、新しく作成されたユーザーアカウントの情報が、パスワードデータベースとグループデータベースに追加されます。オプションを指示すれば新しいユーザーのホームディレクトリも作成されます。多くのディストリビューションでは、新しいユーザーアカウントと同名のグループも作成されます。

新しいユーザーを作成したら、`passwd` コマンドを使用してそのパスワードを設定します。`id` と `groups` コマンドを使用すれば、作成したユーザーアカウントのユーザーID（UID）、グループID（GID）、所属するグループを確認できます。

```
# passwd michael
Changing password for user michael.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
# id michael
uid=1000(michael) gid=100(michael) groups=100(michael)
# groups michael
michael : michael
```

NOTE

一般ユーザーを含めて、引数を与えない `id` および `groups` コマンドで自分のUIDとGID、所属グループを確認できますし、`passwd` コマンドで自らのパスワードを変更できます。root権限を持つユーザーのみが、すべての ユーザーのパスワードを変更できます。

`useradd` コマンドの主要なオプションを以下に示します:

`-c`
作成するユーザーアカウントに、独自のコメント（たとえばユーザーのフルネーム）をセットします。

`-d`
作成するユーザーアカウントの、ホームディレクトリを指定します。

`-e`
作成するユーザーアカウントを無効化する日付を（YYYY-MM-DD形式で）指定します。

`-f`
作成するユーザーアカウントに対して、パスワードの有効期限が切れてから、アカウントが無効化されるまでの日数（その間にユーザーはパスワードを更新できる）を指定します。

`-g`
作成するユーザーアカウントのプライマリグループID（GID）を指定します。

`-G`
作成するユーザーアカウントが所属する、複数のセカンダリグループを（GIDないしグループ名で）指定します。

`-k`
作成するユーザーアカウントに対して、コピーするスケルトンファイルが存在するディレクトリを指定します。（このオプションは、`-m` または `--create-home` オプションが指定されている場合にのみ有効です）。

`-m`
ユーザーアカウントのホームディレクトリを作成します（存在しない場合）。

`-M`
作成するユーザーアカウントのホームディレクトリを作成しないことを指定します。

-s
作成するユーザーアカウントのログインシェルを指定します。

-u
作成するユーザーアカウントのプライマリUIDを指定します。

すべてのオプションのリストは、useradd コマンドのマニュアルを参照してください。

NOTE
すべてのユーザーアカウントは、少なくとも1つのグループに所属する必要あり、それをプライマリグループと呼びます。その他のグループはセカンダリグループです。プライマリグループとセカンダリグループに機能的な違いはありませんが、保管されるデータベースが異なっています。

ユーザーアカウントの変更

ログイン名、ログインシェル、パスワードの有効期限など、既存のユーザーアカウントの属性を変更するには、usermod コマンドを使用します。

```
# usermod -s /bin/tcsh michael
# usermod -c "Michael User Account" michael
```

useradd コマンドと同様に、usermod コマンドの実行にはroot権限が必要です。

1つ目の例では、michael のログインシェルを変更していて、2つの目の例ではユーザーに簡単なメモ（通常はフルネームや所属部署など）を追加しています。1つのコマンドに複数のオプションを指定すれば、複数の属性を一度に変更できます。

usermod コマンドの主要なオプションを以下に示します:

-c
指定したユーザーアカウントに、簡単なメモ（本名や所属部署など）を追加します。

-d
指定したユーザーアカウントのホームディレクトリを変更します。-m オプションを一緒に指定すると、現在のホームディレクトリの内容が新しいホームディレクトリに移動されます。新しいホームディレクトリがまだ存在しない場合には、作成されます。

-e
指定したユーザーアカウントが無効となる日付を、YYYY-MM-DD 形式で指定します。

-f
指定したユーザーアカウントに対して、パスワードの有効期限が切れてから、アカウントが無効化されるまでの日数（その間にユーザーはパスワードを更新できる）を指定します。

-g
指定したユーザーアカウントのプライマリグループを変更します（変更後のグループが存在している必要があります）。

-G
指定したユーザーアカウントに、セカンダリグループを追加します。追加するグループは存在している必要があり、複数のグループを追加する場合には空白を挟まずにコンマで次のグループと区切ります。このオプションのみを指定するとそれまでにユーザーが属していたセカンダリグループから外されますが、`-a` オプションを併用すると既存のグループに加えて新しいセカンダリグループに参加します。

-l
指定したユーザーアカウントのログイン名を変更します。ホームディレクトリは変更されません。

-L
指定したユーザーアカウントをロックします。これにより、`/etc/shadow` ファイル内の暗号化されたパスワードの前に感嘆符が付けられて、そのユーザーのパスワードによるアクセスが無効になります。詳しくは次のレッスンで解説します。

-s
指定したユーザーアカウントのログインシェルを変更します。

-u
指定したユーザーアカウントのUIDを変更します。ユーザーのホームディレクトリ以下にあるユーザー所有のディレクトリやファイルの所有者UIDも変更されます。

-U
指定したユーザーアカウントのロックを解除します。これにより、`/etc/shadow` ファイルで暗号化されたパスワードの前にある感嘆符が削除されます。詳しくは次のレッスンで解説します。

すべてのオプションのリストは、`usermod` コマンドのマニュアルを参照してください。

TIP
ユーザーアカウントのログイン名を変更した場合（`-l` オプション）、そのユーザーのホームディレクトリ名や、メールスプールファイル名などの、ユーザーに関連するアイテムの名前を変更する必要があります。また、ユーザーアカウントのUIDを変更した場合、ホームディレクトリ以外のファイルとディレクトリの所有者を修正する必要があります（ユーザーのメールボックスと、ホームディレクトリにあるそのユーザーが所有するすべてのファイルについては、自動的にユーザーIDが変更されます）。

ユーザーアカウントの削除

ユーザーアカウントを削除したい場合は、`userdel` コマンドを使用します。このコマンドは、アカウントデータベース（次のレッスンで説明します）を更新し、指定したユーザーに関わるすべてのエントリを削除します。`-r` オプションを指定すると、ユーザーのホームディレクトリと、そこに含まれるすべての内容と、ユーザーのメールスプールも削除します。ホームディレクトリ以外の場所にあるファイルは、手動で検索して削除する必要があります。

```
# userdel -r michael
```

`useradd` や `usermod` と同様に、ユーザーアカウントを削除するにはroot権限が必要です。

グループの追加、変更、削除

グループを追加、変更、および削除するには、ユーザーの場合と同様にroot権限で `groupadd`、`groupmod`、`groupdel` コマンドを使用します。`developer` という名前の新しいグループを作成するには、次のコマンドを使用します:

```
# groupadd -g 1090 developer
```

このコマンドの `-g` オプションには、作成するグループのGID指定します。

WARNING 現在の主要なディストリビューションのほとんどは、User Private Group（UPG）と言って、ユーザーの作成時にユーザー名と同じ名前のグループ（所属メンバーは同名のユーザーのみ）を作成し、それをユーザーのプライマリグループとするポリシーを採用しています。このポリシーが有効となっている場合（後述）は、プライマリグループが自動的に作成されます。UPGが無効の場合は、ユーザーの作成に先立って、そのユーザーが所属するプライマリグループをあらかじめ作成しておく必要があります。

グループ名を `developer` から `web-developer` に変更すると共に、そのGIDを（1050に）変更するには、次のコマンドを使用します:

```
# groupmod -n web-developer -g 1050 developer
```

TIP `-g` オプションを使用してGIDを変更した場合は、そのグループに属するすべてのファイルやディレクトリを探し出して、その所有グループを変更する必要があります。

`web-developer` グループを削除するには、次のコマンドを使用します:

```
# groupdel web-developer
```

いずれかのユーザーのプライマリグループに指定されているグループは、削除することができません。つまり、グループを削除する前にユーザーを削除するか、そのユーザーのプライマリグループを変更する必要があります。ユーザーを削除した場合と同様に、グループを削除しても、そのグループに属するファイルはそのままファイルシステムに残り、削除されたり別のグループに割り当てられたりすることはありません。

スケルトンディレクトリ

新しいユーザーを追加する時にそのホームディレクトリを作成することを指示する（`-m` オプション）と、新しく作成されたホームディレクトリに、スケルトンディレクトリ（デフォルトでは `/etc/skel`）からディレクトリとファイルがコピーされます。この背後にある考え方は単純で、システム管理者は、新しいユーザーのホームディレクトリに、所定のディレクトリとファイルを置きたいのです。したがって、新しいユーザーアカウントのホームディレクトリに自動的にコピーされるディレクトリやファイルをカスタマイズしたい場合は、スケルトンディレクトリの内容をカスタマイズします。

TIP スケルトンディレクトリ内のすべてのファイルやディレクトリを一覧したい時は、`ls -al` コマンドを使用することに注意してください。（訳注: 多くの隠しファイルがあるのが普通です。）

/etc/login.defs ファイル

Linuxでは、ユーザやグループの作成を制御するパラメータが、/etc/login.defs ファイルに格納されています。ここまでに説明してきたコマンドは，このファイルを参照してデフォルトの動作を決定しています.

このファイルに指定できる重要なパラメーターを以下に示します:

UID_MIN と UID_MAX
新しい一般ユーザーに割り当てるユーザーIDの範囲。

GID_MIN と GID_MAX
新しいグループに割り当てるグループIDの範囲。

CREATE_HOME
ユーザー作成時にそのホームディレクトリをデフォルトで作成することを指定します。

USERGROUPS_ENAB
User Private Group（UPG）と呼ばれる、ユーザー/グループ管理ポリシーを有効にします。すなわち、ユーザー作成時に同じ名前のグループを作成し、ユーザーを削除したときにそのプライマリグループに他のメンバーが居なければそのグループも削除します。

MAIL_DIR
メールスプールディレクトリ。

PASS_MAX_DAYS
あるパスワードを使用できる最大日数。（訳注: 以下、次のレッスンで解説しますが、既存のユーザーには影響しません。）

PASS_MIN_DAYS
次にパスワードを変更できるようになるまでの最小日数（訳注: パスワードを変更してすぐに前のパスワードに戻すことへの対策です）。

PASS_MIN_LEN
パスワードの長さの最小値。

PASS_WARN_AGE
パスワードの有効期限が切れる前に、変更を促す警告を表示する日数。

TIP
ユーザー/グループの管理ポリシーはディストリビューションによって異なります。複数のディストリビューションを混在して使用している場合などには、このファイルをチェックして管理ポリシーを確認しておくとよいでしょう。組織のポリシーに応じて、必要があれば変更して下さい。

passwd コマンド

このコマンドは、主にユーザーのパスワードを変更するために使用します。前述のように、すべてのユーザーが自分のパスワードを変更できますが、rootだけが すべてのユーザー のパスワードを変更できます。passwd コマンドにはSUIDビットが設定されている（所有者の実行可能フラグが s）ので、ファイルの所有者（つまりroot）の権限で実行されます。そのため、すべてのユーザーが自分のパスワード

を変更できるのです。

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
```

`passwd` のオプションには、パスワードの有効期間など（パスワードエージングと呼びます）を制御するものがあります:

`-d`
ユーザーアカウントのパスワードを削除します（すなわちユーザーを無効にします）。

`-e`
ユーザーにパスワードの変更を強制します（ユーザー次にログインしたときに、パスワードの変更が求められます）。

`-i`
パスワードの有効期限が切れてから、アカウントが無効化されるまでの日数（その間はパスワードを更新できます）を指定します。

`-l`
ユーザーアカウントをロックします（`/etc/shadow` ファイルの暗号化されたパスワードの前に感嘆符を追加します）。

`-n`
次にパスワードを変更できるようになるまでの最小日数を指定します。

`-S`
指定したユーザーアカウントのパスワードの状態（ロックの有無、パスワードの有無など）を出力します

`-u`
ユーザーアカウントのロックを解除します（`/etc/shadow` ファイルのパスワードから感嘆符が削除されます）。

`-x`
パスワードの最大有効日数を指定します。

`-w`
パスワードの有効期限が切れる前に、パスワードの変更を促す警告を表示する日数を指定します。

NOTE

グループにもパスワードを設定することができ、`gpasswd` コマンドを使用します。メンバーではないがグループのパスワードを知っているユーザーは、`newgrp` コマンドを使用して一時的にグループに参加できます。また `gpasswd` コマンドでは、グループへのメンバーの追加や削除、グループ管理者リストの管理なども行えます。

`chage` コマンド

このコマンドは “change age” の略で、ユーザーのパスワードエージング情報を変更します。`chage`

コマンドの実行にはroot権限が必要ですが、一般ユーザーは `-l` オプションで自分のアカウントのパスワードエージング情報を表示することができます。

`chage` コマンドの主要なオプションは次のとおりです:

`-d`
ユーザーが最後にパスワードを変更した日付を変更します。

`-E`
ユーザーアカウントが無効化される日付を指定します。（日付は `YYYY-MM-DD` 形式ないし1970年1月1日からの日数で指定します）

`-I`
パスワードの有効期限が切れた後、ユーザーがパスワードを更新できる期間を日数で指定します（期間内に更新しないと、アカウントが無効になります）。

`-m`
ユーザーアカウントのパスワードの最小有効期間（パスワードを変更してから次に変更できるまでの日数）を指定します。

`-M`
ユーザーアカウントのパスワードの最大有効期間（日数）を指定します。

`-W`
パスワードが失効する前に、パスワードを変更を促す警告を表示する日数を指定します。

NOTE
訳注:　パスワードの有効期間は、かつてパスワードの定期的な変更が推奨されていた頃の名残です。セキュリティ上、現在では定期的なパスワードの変更を強制することは非推奨となっています。

演習

1. 以下のコマンドそれぞれの役割は何ですか？

`usermod -L`	
`passwd -u`	
`chage -E`	
`groupdel`	
`useradd -s`	
`groupadd -g`	
`userdel -r`	
`usermod -l`	
`groupmod -n`	
`useradd -m`	

2. 以下に示す `passwd` コマンドに対応する、`chage` コマンドは何ですか？

`passwd -n`	
`passwd -x`	
`passwd -w`	
`passwd -i`	
`passwd -S`	

3. 前問におけるそれぞれのコマンドの働きを説明して下さい。

4. ユーザーアカウントをロックするコマンドは何ですか？　また、そのロックを解除するコマンドは何ですか？

発展演習

1. `groupadd` コマンドを使用して、`administrators` グループと `developers` グループを作成してください。rootとして作業しているものとします。

2. 前問でグループを作成したので、次のコマンドを実行します: `useradd -G administrators,developers kevin`

   `/etc/login.defs` の `CREATE_HOME` と `USERGROUPS_ENAB` が、`yes` であるとします。このコマンドは何を行いますすか？

3. `designers` という名前の新しいグループを作成し、名前を `web-designers` に変更して、この新しいグループを ユーザー `kevin` のセカンダリグループに追加して下さい。さらに、`kevin` が属するすべてのグループとそのIDを表示して下さい。

4. `kevin` のセカンダリグループから `developers` グループのみを削除して下さい。

5. ユーザー `kevin` のアカウントにパスワードを設定して下さい。

6. `chage` コマンドを使用して、まず `kevin` アカウントの有効期限を確認してから、2022年12月31日に変更して下さい。ユーザーアカウントの有効期限を変更するためのコマンドには、どのようなものがありますか？

7. ユーザー `emma` という新しいユーザーをUID 1050で作成します。プライマリグループを `administrators` とし、セカンダリグループを `developers` と `web-designers` として下さい。PUGは無効であるとします。

8. `emma` のログインシェルを `/bin/sh` に変更して下さい。

9. `emma` と `kevin` のユーザーアカウント、ならびに `administrators`、`developers`、`web-designers` グループを削除して下さい。

まとめ

このレッスンでは、以下の事柄を学びました:

- Linuxにおける、ユーザーとグループ管理の基礎知識
- ユーザーを追加、変更、削除する方法
- グループを追加、変更、削除する方法
- スケルトンディレクトリの管理
- ユーザーとグループの作成を制御するファイル（login.defs）の役割
- ユーザーのパスワードを変更する方法
- ユーザーアカウントのパスワードエージング情報を変更する方法

このレッスンでは、以下に示すファイルとコマンドを説明しました:

`useradd`
新しいユーザーアカウントを作成します。

`usermod`
ユーザーアカウントの属性を変更します。

`userdel`
ユーザーアカウントを削除します。

`groupadd`
新しいグループを作成します。

`groupmod`
グループの属性（グループ名、GID）を変更します。

`groupdel`
グループを削除します。

`passwd`
ユーザーのパスワードを変更したり、パスワードエージングを制御したりします。

`chage`
ユーザーのパスワードエージング情報を変更します。

`/etc/skel`
デフォルトのスケルトンディレクトリ。

`/etc/login.defs`
ユーザーとグループの作成を制御し、ユーザーアカウント属性のデフォルト値を保持するファイル。

演習の解答

1. 以下のコマンドそれぞれの役割は何ですか？

`usermod -L`	ユーザーアカウントをロックします
`passwd -u`	ユーザーアカウントのロックを解除します
`chage -E`	ユーザーアカウントの有効期限を設定します
`groupdel`	グループを削除します
`useradd -s`	ログインシェルを指定して、ユーザーアカウントを作成します
`groupadd -g`	GIDを指定して、新しいグループを作成します
`userdel -r`	ユーザーアカウントと、そのホームディレクトリ、そこに含まれるファイル、メールスプールを削除します
`usermod -l`	ユーザーアカウントのログイン名を変更します
`groupmod -n`	グループの名前を変更します
`useradd -m`	ユーザーアカウントとそのホームディレクトリを作成します

2. 以下に示す `passwd` コマンドに対応する、`chage` コマンドは何ですか？

`passwd -n`	`chage -m`
`passwd -x`	`chage -M`
`passwd -w`	`chage -W`
`passwd -i`	`chage -I`
`passwd -S`	`chage -l`

3. 前問におけるそれぞれのコマンドの働きを説明して下さい。

   `passwd -n` コマンド（ないし `chage -m`）は、パスワードの最小有効期間を設定します。 `passwd -x` コマンド（ないし `chage -M`）は、パスワードの最大有効期間を設定します。 `passwd -w` コマンド（ないし `chage -W`）は、パスワードの変更を促す警告を表示する日数を設定します。 `passwd -i` コマンド（ないし `chage -I`）は、有効期間が切れたパスワードを更新できる日数を設定します。 `passwd -S` コマンド（ないし `chage -l`）は、ユーザーアカウントのパスワードの状態を表示します。

4. ユーザーアカウントをロックするコマンドは何ですか？　また、そのロックを解除するコマンドは何ですか？

   ユーザーアカウントをロックするには、`usermod -L`、`usermod --lock`、`passwd -l` のいずれかのコマンドを使用します。逆にロックを解除するには、`usermod -U`、`usermod --unlock`、`passwd -u` のいずれかを使用します。

発展演習の解答

1. `groupadd` コマンドを使用して、`administrators` グループと `developers` グループを作成してください。rootとして作業しているものします。

```
# groupadd administrators
# groupadd developers
```

2. 前問でグループを作成したので、次のコマンドを実行します: `useradd -G administrators,developers kevin`

   `/etc/login.defs` の `CREATE_HOME` と `USERGROUPS_ENAB` が、`yes` であるとします。このコマンドは何を行いますか？

   このコマンドは、システムに `kevin` という名前の新しいユーザーを追加し、そのホームディレクトリ（`CREATE_HOME` がyesに設定されているので `-m`オプションを省略できます）と、新しいグループ `kevin` をこのユーザーアカウントのプライマリグループとして（`USERGROUPS_ENAB` がyesに設定されているため）作成します。最後に、スケルトンディレクトリに含まれるディレクトリとファイルを、`kevin` のホームディレクトリにコピーします。

3. `designers` という名前の新しいグループを作成し、名前を `web-designers` に変更して、この新しいグループを `kevin` のセカンダリグループに追加して下さい。さらに、`kevin` が属するすべてのグループとそのIDを表示して下さい。

```
# groupadd designers
# groupmod -n web-designers designers
# usermod -a -G web-designers kevin
# id kevin
uid=1010(kevin) gid=1030(kevin) groups=1030(kevin),1028(administrators),1029(developers),1031(web-
designers)
```

4. `kevin` のセカンダリグループから `developers` グループのみを削除してください。

```
# usermod -G administrators,web-designers kevin
# id kevin
uid=1010(kevin) gid=1030(kevin) groups=1030(kevin),1028(administrators),1031(web-designers)
```

   `usermod` コマンドには、1つのグループのみを削除するオプションがありません。そのため、ユーザーが属するすべてのセカンダリグループを再指定します。

5. ユーザー `kevin` のアカウントにパスワードを設定して下さい。

```
# passwd kevin
Changing password for user kevin.
New UNIX password:
Retype new UNIX password:
```

```
passwd: all authentication tokens updated successfully.
```

6. `chage` コマンドを使用して、まず kevin アカウントの有効期限を確認してから、2022年12月31日に変更して下さい。ユーザーアカウントの有効期限を変更するためのコマンドには、どのようなものがありますか？

```
# chage -l kevin | grep "Account expires"
Account expires     : never
# chage -E 2022-12-31 kevin
# chage -l kevin | grep "Account expires"
Account expires     : dec 31, 2022
```

`usermod` コマンドの `-e` オプションは、`chage -E` と同じです。

7. `emma` という名前の新しいユーザーをUID 1050で作成します。プライマリグループを `administrators` とし、セカンダリグループを `developers` と `web-designers` として下さい。PUGは無効であるとします。

```
# useradd -u 1050 -g administrators -G developers,web-designers emma
# id emma
uid=1050(emma) gid=1028(administrators) groups=1028(administrators),1029(developers),1031(web-designers)
```

8. `emma` のログインシェルを `/bin/sh` に変更して下さい。

```
# usermod -s /bin/sh emma
```

あるいは、`chsh` コマンド（CHange SHellの意）を使います。`chsh` は一般ユーザーが自分のシェルを変更するためにも使えます。

9. `emma` と `kevin` のユーザーアカウント、ならびに `administrators`、`developers`、`web-designers` グループを削除して下さい。

```
# userdel -r emma
# userdel -r kevin
# groupdel administrators
# groupdel developers
# groupdel web-designers
```

Linux Professional Institute

107.1 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	107 管理タスク
Objective:	107.1 ユーザーとグループのアカウント管理と関連ファイル
Lesson:	2 of 2

はじめに

前のレッスンで説明したコマンドラインツールや、各ディストリビューションが提供するグラフィカルなアカウント管理のアプリケーションは、ユーザーとグループに関する情報を保持するファイル群を更新する働きをします。

それらは `/etc/` ディレクトリにある以下のファイルです。

`/etc/passwd`
ユーザーに関する基本的な情報を含む、コロンで区切られた7つのフィールドからなるファイル。

`/etc/group`
グループに関する基本的な情報を含む、コロンで区切られた4つのフィールドからなるファイル。

`/etc/shadow`
暗号化されたユーザーパスワードなどを含む、コロンで区切られた9つのフィールドからなるファイル。

`/etc/gshadow`
暗号化されたグループパスワードなどを含む、コロンで区切られた4つのフィールドからなるファイル。

これら4つのファイルはプレーンテキストですが、直接編集するのではなく、必ずディストリビューションが提供するツールを使って編集してください。たとえば、ユーザー情報を変更するときは、前のレッスンで学んだように、`usermod` コマンドを使用します。

/etc/passwd

これは、各行がひとりのユーザーを示す、誰でも読み取り可能なファイルです。それぞれの行には、コロンで区切られた7つのフィールドが含まれています。

ユーザー名
ユーザーがシステムにログインするときに使用する名前。

パスワード
暗号化されたパスワード（/etc/shadowが使用されている場合は、文字 `x`）。

ユーザーID（UID）
システム上でユーザーに割り当てられたID番号。

グループID（GID）
システム上でユーザーに割り当てられたプライマリグループ番号。

GECOS
オプションのコメントフィールドで、ユーザーに関する追加情報（フルネームなど）を記載します。カンマで区切って、複数のエントリを含む事ができます。（訳注:　「ジーコス」と発音することが多く、1960年代のメインフレーム用OSの名前に由来します。）

ホームディレクトリ
ユーザーのホームディレクトリの絶対パス。

シェル
ユーザーがシステムにログインしたときに起動されるプログラムの絶対パス（通常は　`/bin/bash`　などの対話型シェル）。

/etc/group

これは、各行が1つのグループを示す、誰でも読み取り可能なファイルです。それぞれの行には、コロンで区切られた4つのフィールドが含まれています。

グループ名
グループの名前。

グループパスワード
暗号化されたグループパスワード（/etc/gshadowが使用されている場合は、文字 `x`）。

グループID（GID）
システム内のグループに割り当てられたID番号。

メンバーリスト
グループに属するユーザーをコンマで区切ったリスト。（このグループをプライマリグループとするユーザーは、記載する必要が無い。）

/etc/shadow

これは、各行がひとりのユーザーを示し、暗号化されたパスワードなどを含む、rootとその権限を持つユーザーのみが読み取ることができるファイルです。それぞれの行には、コロンで区切られた9つのフィールドが含まれています。

ユーザー名
ユーザーがシステムにログインするときに使用する名前。（/etc/passwdと同じである必要があります）。

暗号化されたパスワード
ユーザーの暗号化されたパスワード（`!` で始まる場合は、アカウントがロックされています）。

パスワードの変更日
1970年1月1日からの日数で、最後にパスワードを変更した日付を示します（値0は、次にログインしたときにパスワードを変更する必要があることを意味します）。

パスワードの最小寿命
パスワードを変更してから、再度パスワードを変更できるようになるまでの日数。（訳注: 0ないし空欄は、最小寿命の制限無し。）

パスワードの最長寿命
パスワードの変更が必要になるまでの最大日数。（訳注: 空欄は寿命の制限無し。）

パスワード警告期間
パスワードの最長寿命が切れる前に、パスワードの変更をユーザーに促す日数。

パスワード無効期間
パスワードの最長寿命が切れた後、パスワードを更新できる日数。この期間の内にパスワードを変更しない場合、アカウントは無効化される。

アカウント有効期限
1970年1月1日からの日数で、ユーザーアカウントが無効化される日付を示します。（空欄の場合は、無効化されません。）

予約フィールド
将来の使用のために予約されているフィールド。

/etc/gshadow

これは、各行がひとつのグループを示し、暗号化されたグループパスワードなどを含む、rootとその権限を持つユーザーのみが読み取ることができるファイルです。それぞれの行には、コロンで区切られた4つのフィールドが含まれています。

グループ名
グループの名前。（/etc/groupと同じである必要があります）。

暗号化されたパスワード
暗号化されたグループパスワード（グループのメンバーではないユーザーが `newgrp` コマンドを使用して一時的にグループに参加する場合に使用します。`!` で始まる場合は、誰も `newgrp` でグルー

プにアクセスできません）。

グループ管理者
グループの管理者（グループのパスワードを変更したり、`gpasswd` コマンドを使用してグループメンバーを追加/削除したりできるメンバーを意味します）を、コンマで区切って並べたリスト。

グループメンバー
グループのメンバーを、コンマで区切ったリスト。（訳注: /etc/group と同じです。）

パスワードとグループのデータベースから抽出する

これら4つのファイルに保存されているユーザーとグループに関する情報を確認したり、特定のレコードを抽出したりすることは少なくありません。そのような場合には、`grep` コマンドを使用するか、`cat` と `grep` を連結します。

```
# grep emma /etc/passwd
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
# cat /etc/group | grep db-admin
db-admin:x:1050:grace,frank
```

これらのデータベースにアクセスする別の方法は、`getent` コマンドを使用することです。このコマンドには、NSS（Name Service Switch）がサポートするデータベースの名前と、そこから探したいキーの値を引数として指定します。引数にキーが指定されていない場合は、指定したデータベースのすべてのエントリを表示します（データの列挙をサポートしていないデータベースを除く）。引数に1つ以上のキー引数が指定されていれば、それに応じてフィルタリングして、データベースエントリを表示します。（訳注: NSSは情報の参照先（この場合はデータベース ファイル）を切り替える仕組みです。DNSのトピックで説明します）。

```
# getent passwd emma
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
# getent group db-admin
db-admin:x:1050:grace,frank
```

`getent` コマンドの実行にはroot権限を必要としませんが、対象のデータベースを読み取れる必要があります。

NOTE `getent` は `/etc/nsswitch.conf` ファイルに設定されているデータベースにのみアクセスできることに注意してください。

演習

1. 以下の出力を見て、質問に答えてください。

```
# cat /etc/passwd | grep '\(root\|mail\|catherine\|kevin\)'
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/spool/mail:/sbin/nologin
catherine:x:1030:1025:User Chaterine:/home/catherine:/bin/bash
kevin:x:1040:1015:User Kevin:/home/kevin:/bin/bash
# cat /etc/group | grep '\(root\|mail\|db-admin\|app-developer\)'
root:x:0:
mail:x:8:
db-admin:x:1015:emma,grace
app-developer:x:1016:catherine,dave,christian
# cat /etc/shadow | grep '\(root\|mail\|catherine\|kevin\)'
root:$6$1u36Ipok$ljt8ooPMLewAhkQPf.lYgGopAB.jClTO6ljsdczxvkLPkpi/amgp.zyfAN680zrLLp2avvpdKA0llpssdfcPppOp:1
8015:0:99999:7:::
mail:*:18015:0:99999:7:::
catherine:$6$ABCD25jlld14hpPthEFGnnssEWw1234yioMpliABCdef1f3478kAfhhAfgbAMjY1/BAeeAsl/FeEdddKd12345g6kPACci
k:18015:20:90:5:::
kevin:$6$DEFGabc123WrLp223fsvp0ddx3dbA7pPPc4LMaa123u6Lp02Lpvm123456pyphhh5ps012vbArL245.PR1345kkA3Gas12P:18
015:0:60:7:2::
# cat /etc/gshadow | grep '\(root\|mail\|db-admin\|app-developer\)'
root:*::
mail:*::
db-admin:!:emma:emma,grace
app-developer:!::catherine,dave,christian
```

   - `root` と `catherine` のユーザーID（UID）とグループID（GID）は何ですか？
   - `kevin` のプライマリグループ名は何ですか？ そのグループに他のメンバーはいますか？
   - `mail` に設定されているシェルは何ですか？ その働きは何ですか？
   - `app-developer` グループのメンバーは誰ですか？ そのうち、グループ管理者と通常メンバーは誰ですか？
   - `catherine` のパスワードの最小寿命は何日ですか？ そのパスワードの最長寿命は何日ですか？
   - `kevin` のパスワード無効期間は何日ですか？

2. 慣例的に、システムアカウントに割り当てられるIDの範囲と、通常のユーザーに割り当てられるIDの

範囲は、それぞれどうなりますか？

3. 以前はシステムにアクセスできたユーザーアカウントが、ロックされているかどうかを調べるにはどうしますか？ システムではシャドウパスワードを使用しています。

発展演習

1. `useradd -m` コマンドで `christian` という名前のユーザーアカウントを作成し、そのユーザーID（UID）とグループID（GID）、シェルを確認してください。

2. `christian` のプライマリグループ名は何ですか？ そこから何がわかりますか？

3. `getent` コマンドで、`christian` のパスワードエージング情報を調べてください。

4. `christian` のセカンダリグループに、`editor` グループを追加してください。このグループには、`emma`、`dave`、`frank` が一般メンバーとして含まれているものとします。このグループには管理者がいないことを確認するにはどうしますか？

5. `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` コマンドを実行し、ファイルパーミッションの観点から出力結果を分析して下さい。これら4つのファイルのうち、セキュリティ上の理由で秘匿されているものはどれですか？ システムではシャドウパスワードを使用しています。

まとめ

このレッスンでは、以下の事柄を学びました:

- ユーザーとグループに関する情報を保存するファイルの位置と名前。
- パスワードとグループのデータベースに保存されている、ユーザーとグループの管理情報。
- パスワードとグループのデータベースから情報を取得する方法。

このレッスンでは、以下に示すファイルとコマンドを説明しました:

`/etc/passwd`
ユーザーに関する基本情報を保持するファイル。

`/etc/group`
グループに関する基本情報を保持するファイル。

`/etc/shadow`
暗号化されたユーザーパスワードなどを保持するファイル。

`/etc/gshadow`
暗号化されたグループパスワードなどを保持するファイル。

`getent`
パスワードとグループのデータベースからエントリを抽出するコマンド。

演習の解答

1. 以下の出力を見て、質問に答えてください。

```
# cat /etc/passwd | grep '\(root\|mail\|catherine\|kevin\)'
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/spool/mail:/sbin/nologin
catherine:x:1030:1025:User Chaterine:/home/catherine:/bin/bash
kevin:x:1040:1015:User Kevin:/home/kevin:/bin/bash
# cat /etc/group | grep '\(root\|mail\|db-admin\|app-developer\)'
root:x:0:
mail:x:8:
db-admin:x:1015:emma,grace
app-developer:x:1016:catherine,dave,christian
# cat /etc/shadow | grep '\(root\|mail\|catherine\|kevin\)'
root:$6$1u36Ipok$ljt8ooPMLewAhkQPf.lYgGopAB.jClTO6ljsdczxvkLPkpi/amgp.zyfAN680zrLLp2avvpdKA0llpssdfcPppOp:1
8015:0:99999:7:::
mail:*:18015:0:99999:7:::
catherine:$6$ABCD25jlld14hpPthEFGnnssEWw1234yioMpliABCdef1f3478kAfhhAfgbAMjY1/BAeeAsl/FeEdddKd12345g6kPACci
k:18015:20:90:5:::
kevin:$6$DEFGabc123WrLp223fsvp0ddx3dbA7pPPc4LMaa123u6Lp02Lpvm123456pyphhh5ps012vbArL245.PR1345kkA3Gas12P:18
015:0:60:7:2::
# cat /etc/gshadow | grep '\(root\|mail\|db-admin\|app-developer\)'
root:*::
mail:*::
db-admin:!:emma:emma,grace
app-developer:!::catherine,dave,christian
```

- `root` と `catherine` のユーザーID（UID）とグループID（GID）は何ですか？

  `root` のUIDとGIDは0と0です、`catherine` のUIDとGIDは1030と1025です。

- `kevin` のプライマリグループ名は何ですか？ そのグループに他のメンバーはいますか？

  プライマリグループ名は `db-admin` です。`emma` と `grace` が、このグループのメンバーです。

- `mail` に設定されているシェルは何ですか？ その働きは何ですか？

  `mail` はシステムユーザーアカウントであり、そのシェルは `/sbin/nologin` です。`mail`、`ftp`、`news`、`daemon` などのシステムユーザーアカウントは管理タスクの実行に使用されるので、通常のログインを禁止します。そのために、シェルとして `/sbin/nologin` ないし `/bin/false` を指定します。

- `app-developer` グループのメンバーは誰ですか？ そのうち、グループ管理者と通常メンバーは誰ですか？

  メンバーは `catherine`、`dave`、`christian` で、全員が一般メンバーです。

- `catherine` のパスワードの最小寿命は何日ですか？ そのパスワードの最長寿命は何日ですか？

  パスワードの最小寿命は20日で、パスワードの最長寿命は90日です。

- `kevin` のパスワード無効期間は何日ですか？

  パスワードの無効期間は2日です。この期間中に、`kevin` はパスワードを更新する必要があり、そうしないとアカウントが無効化されます。

2. 慣例的に、システムアカウントに割り当てられるIDの範囲と、通常のユーザーに割り当てられるIDの範囲は、それぞれどうなりますか？

   システムアカウントには、100未満、ないしは500～1000のUIDを割り当てます。一般ユーザーのUIDは1000から始まりますが、一部の古いシステムでは500から始まることがあります。`root` ユーザーのUIDは0です。`/etc/login.defs` の `UID_MIN` と `UID_MAX` が、一般ユーザーに割り当てるUIDの範囲を定義します。LPI Linux EssentialsとLPIC-1では、システムアカウントのUIDは1000未満であり、一般ユーザーのUIDは1000以上としています。

3. 以前はシステムにアクセスできたユーザーアカウントが、ロックされているかどうかを調べるにはどうしますか？ システムではシャドウパスワードを使用しています。

   シャドウパスワードを使用する場合、暗号化されたユーザーパスワードが `/etc/shadow` に保存されるため、`/etc/passwd` の2番目のフィールドには 文字 `x` が置かれます。ユーザーアカウントの暗号化されたパスワードは、`/etc/shadow` ファイルの2番目のフィールドに保存され、それが感嘆符で始まる場合はアカウントがロックされています。

発展演習の解答

1. useradd -m コマンドで christian という名前のユーザーアカウントを作成し、そのユーザーID（UID）とグループID（GID）、シェルを確認してください。

```
# useradd -m christian
# cat /etc/passwd | grep christian
christian:x:1050:1060::/home/christian:/bin/bash
```

christian のUIDとGIDはそれぞれ、1050と1060です（/etc/ passwd の3番目と4番目のフィールド）。このユーザーのログインシェルは /bin/bash です（/etc/passwd の7番目のフィールド）。

2. christian のプライマリグループ名は何ですか？ そこから何がわかりますか？

```
# cat /etc/group | grep 1060
christian:x:1060:
```

christian のプライマリグループ名は christian（/etc/group の最初のフィールド）です。/etc/login.defs の USERGROUPS_ENAB がyesに設定されていて、useradd がデフォルトでユーザーアカウントと同じ名前のグループを作成したことが分かります。

3. getent コマンドで、christian のパスワードエージング情報を調べて下さい。

```
# getent shadow christian
christian:!:18015:0:99999:7:::
```

christian ユーザーアカウントにパスワードが設定されておらずロックされています（/etc/shadow の2番目のフィールドには感嘆符が含まれています）。このユーザーアカウントのパスワードの最小有効期間と最大有効期間はありません（/etc/shadow の4番目と5番目のフィールドは0日と99999日に設定されています）が、パスワード警告期間は7日に設定されています（/etc/shadow の6番目のフィールド）。最後に、無効期間はなく（ /etc/shadow の7番目のフィールド）、アカウントは期限切れになりません（/etc/shadow の8番目のフィールド）。

4. christian のセカンダリグループに editor を追加して下さい。このグループには、emma、dave、frank が一般メンバーとして含まれているものとします。このグループには管理者がいないことを確認するにはどうしますか？

```
# cat /etc/group | grep editor
editor:x:1100:emma,dave,frank
# usermod -a -G editor christian
# cat /etc/group | grep editor
editor:x:1100:emma,dave,frank,christian
# cat /etc/gshadow | grep editor
editor:!::emma,dave,frank,christian
```

/etc/ghadow の3番目と4番目のフィールドには、グループの管理者と通常メンバーが含まれます。

ここでは `editor` の3番目のフィールドが空ですから、このグループに管理者は存在しません（`emma`、`dave`、`frank`、`christian` はすべて通常メンバーです）。

5. `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` コマンドを実行し、ファイルパーミッションの観点から出力結果を分析して下さい。これら4つのファイルのうち、セキュリティ上の理由で秘匿されているものはどれですか？ システムではシャドウパスワードを使用しています。

```
# ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow
-rw-r--r-- 1 root root    853 mag  1 08:00 /etc/group
-rw-r----- 1 root shadow 1203 mag  1 08:00 /etc/gshadow
-rw-r--r-- 1 root root   1354 mag  1 08:00 /etc/passwd
-rw-r----- 1 root shadow 1563 mag  1 08:00 /etc/shadow
```

`/etc/passwd` ファイルと `/etc/group` ファイルは誰でも読み取り可能で、セキュリティ上の理由でシャドウ化されています。パスワードがシャドウ化されている場合、これらのファイルの2番目のフィールドは `x` になります。ユーザーとグループの暗号化されたパスワードは、私の手元のマシンではrootと `shadow` グループのメンバーのみが読み出せる `/etc/shadow` と `/etc/gshadow` に保存されます。（グループ名とパーミッションはディストリビューションによって異なります）。

107.2 ジョブのスケジュール設定によるシステム管理タスクの自動化

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 107.2

総重量

4

主な知識分野

- cronとatのジョブを管理する。
- cronおよびサービスへのユーザーアクセスを構成する。
- systemdタイマーユニットの理解。

用語とユーティリティ

- `/etc/cron.{d,daily,hourly,monthly,weekly}/`
- `/etc/at.deny`
- `/etc/at.allow`
- `/etc/crontab`
- `/etc/cron.allow`
- `/etc/cron.deny`
- `/var/spool/cron/`
- `crontab`
- `at`
- `atq`
- `atrm`
- `systemctl`
- `systemd-run`

107.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	107 管理タスク
Objective:	107.2 ジョブをスケジュールしてシステム管理タスクを自動化する
Lesson:	1 of 2

はじめに

優れたシステム管理者の最も重要な仕事の一つは、定期的に実行しなければならないジョブをスケジュールすることです。例えばシステム管理者は、バックアップ、システムのアップグレードなど、多くの繰り返し作業を自動的に実行するジョブを作成することができます。これには、`cron` 機能を用いて、定期的なジョブをスケジュールします。

cronでジョブをスケジュールする

`cron`（クローン）は常時稼働し続けるデーモンで、1分ごとに起動して、一連のテーブルから実行すべきタスクを探します。これらのテーブルは crontabs と呼ばれていて、いわゆる クローンジョブ を含んでいます。cronは、スケジュールした時刻にシステムが稼働している場合にのみクローンジョブを実行するので、常に電源が入っているサーバーシステムに適しています。システムのcrontabを管理するrootユーザーだけでなく、一般ユーザーも独自の `crontab` を作って `cron` を利用することができます。

NOTE

Linuxには `anacron` という機能もあり、デスクトップやラップトップなどの電源が切られることがあるシステムに適しています。`anacron` では、ジョブを実行する時刻にマシンがオフになっていると、次にマシンがONになった時にジョブが実行されますが、利用できるのはrootのみです。`anacron` はLPIC-1認定試験の範囲外です。

ユーザーのcrontab

ユーザーがcronジョブのスケジュールを定義するテキストファイルを、`ユーザーcrontabs` と呼びます。作成したユーザーアカウントと同名のファイルで、通常は `/var/spool/cron` のサブディレクトリ

に置かれますが、ファイルの位置はディストリビューションによって異なります。

ユーザーcrontabの各行には、空白で区切った6つのフィールドが含まれています。

- 時刻の分（0-59）
- 日の時刻（0〜23）
- 月の日付（1-31）
- 年の月（1-12）
- 曜日（0-7、日曜日が0ないし7）
- 実行するコマンド

月と曜日は、対応する数値の代わりに名前の最初の3文字でも指定できます。

最初からの5つのフィールドでいつ実行するかを、6番目のフィールドで実行するコマンドをそれぞれ指示します。実行タイミングを指示する5つのフィールドには、範囲や複数の値を含めることができます。よく使われるものを示します:

`*` (アスタリスク)
任意の値を示します。毎分、毎時、毎月曜などの、定期的な繰り替えを指示するときに便利です。たとえば `30 12 * * 0` で、毎日曜日の12時30分を指定します。

`,` (コンマ)
複数の値を区切ります。たとえば `0,30 * * * *` で、毎時0分と30分を指定します。

`-` (ダッシュ)
値の範囲を指定します。たとえば `0 9-17 * * 1-6` で、平日（月〜金曜）9時〜17時の、毎0分を指定します。

`/` (スラッシュ)
値の増分（間隔）を指定します。たとえば `*/5 * * * *` で、5分毎を指定します。

システム全体用のcrontabファイルである `/etc/crontabs` の書式はユーザーcrontabファイルとは異なるのですが、実行タイミングを指示する5つのフィールドについては同じです。分かりやすいコメントが書かれているので見てみましょう。以下は、Debianの `/etc/crontab` ファイルの先頭部分です。

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

システムのcrontab

システムレベルのcronジョブのスケジュールを定義するテキストファイルを、システムcrontabs と呼び、rootユーザーのみが編集できます。`/etc/crontab` と `/etc/cron.d` ディレクトリ内のすべてのファイルが、システムcrontabsです。

ほとんどのディストリビューションには、`/etc/cron.hourly`、 `/etc/cron.daily`、 `/etc/cron.weekly`、`/etc/cron.monthly` ディレクトリがあり、それぞれに所定の頻度で実行されるスクリプトが含まれています。たとえば、毎日実行するスクリプトは、`/etc/cron.daily` に置かれています。

WARNING

一部のディストリビューションでは、`/etc/cron.d/hourly`、 `/etc/cron.d/daily`、`/etc/cron.d/weekly`、 `/etc/cron.d/monthly` が使われます。cronが実行するスクリプトを配置するディレクトリを確認しておきましょう。

システムcrontabsの書式はユーザーcrontabsに似ていますが、cronジョブを実行するユーザーを指定するフィールドが追加されています。つまり、システムcrontabsの各行には、空白で区切った7つのフィールドが含まれています。

- 時刻の分（0-59）
- 日の時刻（0〜23）
- 月の日付（1-31）
- 年の月（1-12）
- 曜日（0-7、日曜日が0ないし7）
- **コマンドを実行するユーザーアカウント名**
- 実行するコマンド

ユーザーcrontabと同様に、`*`、`,`、`-`、`/` を使用して、実行タイミングフィールドに複数の値を指定できます。また、対応する数字の代わりに、名前の最初の3文字で月と曜日を指定することもできます。

特別な時間の指定方法

crontabsファイルでは、実行タイミング指定の5フィールドに、よく使われる特別な短縮表記を使用できます。

`@reboot`
再起動後に、指定したタスクを1回実行します。

`@hourly`
指定されたタスクを毎時0分に実行します。

`@daily`（または `@midnight`）
指定されたタスクを毎日0時0分に実行します。

`@weekly`
指定されたタスクを週に1回、日曜日の0時0分に実行します。

`@monthly`
指定されたタスクを毎月、1日の0時0分に実行します。

`@yearly`（または `@annually`）
指定されたタスクを毎年、1月1日0時0分に実行します。

Crontabの変数

crontabファイルには、スケジュールするタスクを宣言する前に、環境変数を割り当てることができます。あらかじめ設定されていることが多い環境変数を以下に示します:

`HOME`
`cron` がコマンドを呼び出す際のワーキングディレクトリ（デフォルトではユーザーのホームディレクトリ）。

`MAILTO`
標準出力とエラー出力をメールで送信するユーザー名ないしメールアドレス（デフォルトではcrontabの所有者）。コンマで区切って複数の値を指定することもでき、空の場合はメールを送信しません。

`PATH`
コマンドを探すパス。

`SHELL`
使用するシェル（デフォルトでは `/bin/sh`）。

ユーザーcronジョブの作成

個人用のcrontabファイルを設定するには、`crontab` コマンドを使います。`crontab -e` コマンドを実行すると、独自のcrontabファイルを編集したり、まだ存在しない場合は作成したりできます。

```
$ crontab -e
no crontab for frank - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        < □□□□ easiest
  3. /usr/bin/emacs24
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

`crontab` コマンドは、`VISUAL` ないし `EDITOR` 環境変数で指定されたエディタをデフォルトで開きます。したがって、好みのエディタでcrontabファイルの編集を行えます。上記の例のように、`crontab` を初めて実行するときに、リストからエディタを選択できるディストリビューションもあります。

ホームディレクトリにある `foo.sh` スクリプトを毎日午前10時に実行する場合は、crontabファイルに次の行を追加します:

```
0 10 * * * /home/frank/foo.sh
```

以下に示すcrontabエントリを見てみましょう:

```
0,15,30,45 08 * * 2 /home/frank/bar.sh
30 20 1-15 1,6 1-5 /home/frank/foobar.sh
```

1行目は、毎週火曜日の午前8時、午前8時15分、午前8時30分、午前8時45分に、`bar.sh` スクリプトを実行することを指示しています。2行目は、1月と6月の1日から15日までの間の月曜日から金曜日の午後8時30分に、`foobar.sh` スクリプトを実行することを指示しています。

WARNING

crontabファイルを手作業で編集することもできますが、常に `crontab` コマンドを使用することがお勧めです。crontabファイルのパーミッションは、通常 `crontab` コマンド経由でのみ編集できるように設定されています。

上記の `-e` オプション意外にも、`crontab` コマンドには便利なオプションが備わっています:

`-l`
現在のcrontabを標準出力に表示します。

`-r`
現在のcrontabを削除します。

`-u`
このオプションで指定したユーザーのcrontabファイルを操作します。このオプションにはroot権限が必要であり、rootユーザーが一般ユーザーのcrontabファイルを編集するために使用します。

システムcronジョブの作成

一般ユーザーのcrontabsとは異なり、システムのcrontabsはエディタを使って直接編集します。つまり、`/etc/crontab` や `/etc/cron.d` 内のファイルを編集するために、`crontab` コマンドを使うことはありません。システムcrontabsを編集するときは、cronジョブを実行するユーザー（通常はroot）を指定する必要があることに注意してください。

たとえば、`/root` ディレクトリにある `barfoo.sh` スクリプトを毎日午前1時30分に実行する場合は、好みのエディタで `/etc/crontab` を開き、次の行を追加します:

```
30 01 * * * root /root/barfoo.sh >>/root/output.log 2>>/root/error.log
```

この例では、ジョブの標準出力を `/root/output.log` に追加し、エラー出力を `/root/error.log` に追加します。

WARNING

上記の例のように出力をファイルにリダイレクトしない限り（そして `MAILTO` 変数が空白以外の値に設定されていれば）、cronジョブからのすべての出力が電子メール経由でユーザーに送信されます。一般的には、標準出力を `/dev/null`（ないし後に確認できるログファイル）にリダイレクトし、エラー出力はリダイレクトしません。こうすれば、エラー出力があった場合にのみ、それがユーザーに電子メールで通知されます。

ジョブスケジューリングへのアクセスを制限する

Linuxでは、`/etc/cron.allow` ファイルと `/etc/cron.deny` ファイルを使用して、ユーザー毎に `crontab` の利用を制限することができます。`/etc/cron.allow` が存在する場合、その中にリストされているユーザーとrootのみが、`crontab` コマンドを使用してジョブをスケジュールできます。`/etc/cron.allow` は存在しないが `/etc/cron.deny` が存在する場合、このファイルにリストされている一般ユーザーは、`crontab` コマンドを使用してジョブをスケジュールできません（空の `/etc/cron.deny` は、すべての一般ユーザーが `crontab` を使用してジョブをスケジュールできることを意味します）。どちらのファイルも存在しない場合の挙動は、ディストリビューションによって異なります。

NOTE `/etc/cron.allow` ファイルと `/etc/cron.deny` ファイルには、1行に1人のユーザー名をリストします。

cronの代替

サービスマネージャとしてsystemdを採用しているマシンでは、`cron` の代わりに タイマー をセットしてタスクをスケジュールできます。タイマーは、サフィックスが `.timer` であるsystemdユニットファイルで、タイマーが発動したときに実行するタスクを記述したユニットファイルとペアで存在する必要があります。`timer` はデフォルトで、サフィックス以外は同じ名前のサービスユニットを起動します。

タイマーには、ジョブをいつ実行するかを指定する `[Timer]` セクションが含まれます。具体的には、`OnCalendar=` オプションを使用して、cronジョブと同様に動作する リアルタイムタイマー を（カレンダーイベント表記を用いて）定義できます。`OnCalendar=` オプションは、次の書式で指定します。

```
DayOfWeek Year-Month-Day Hour:Minute:Second
```

`DayOfWeek`（曜日）は省略可能です。`*`、`/`、`,` オペレータは、cronジョブのものと同じ意味を持ちますが、連続する範囲を示すには `..` を使用します。曜日は、英単語ないしその最初の3文字で指定します。

NOTE ある開始時点（たとえば、マシンが起動したときや、タイマー自体がアクティブになったとき）から一定時間経過した後に実行される モノトニックタイマー （monotonic timer）を定義することもできます。

たとえば、毎月第1月曜日の05:30に `/etc/systemd/system/foobar.service` という名前のサービスを実行する場合は、対応する `/etc/systemd/system/foobar.timer` に次の行を記入します。

```
[Unit]
Description=Run the foobar service

[Timer]
OnCalendar=Mon *-*-1..7 05:30:00
Persistent=true

[Install]
WantedBy=timers.target
```

新しいタイマーを作成したら、rootとして次のコマンドを実行してタイマーを有効にして開始します。

```
# systemctl enable foobar.timer
# systemctl start foobar.timer
```

スケジュールされたジョブの頻度を変更するには、`OnCalendar` の値を変更してから、`systemctl daemon-reload` コマンドを実行します。

最後に、アクティブなタイマーのリストを、実行される順にソートして表示するには、`systemctl list-timers` コマンドを使用します。アクティブではないタイマーユニットも表示するには、`--all` オプションを追加します。

> **NOTE** タイマーはsystemdジャーナルに記録されるので、さまざまなユニットのログを `journalctl` コマンドを使用して確認できます。また、一般ユーザーとして実行する場合は、`systemctl` と `journalctl` コマンドに `--user` オプションを使用します。

`OnCalender=` には、上記の正規化された長い形式に代えて、ジョブの実行頻度を指定する簡単な短縮表記も使えます。

`hourly`
指定されたタスクを毎時0分に実行します。

`daily`
指定されたタスクを毎日0時0分に実行します。

`weekly`
指定されたタスクを週に1回、月曜日の0時0分に実行します。（訳注: cronの@weeklyは日曜日であることに注意してください）。

`monthly`
指定されたタスクを毎月、1日の0時0分に実行します。

`yearly`
指定されたタスクを毎年、1月1日の0時0分に実行します。

日時の指定ではタイムゾーンを指定することもできます。詳細は `systemd.timer(7)` のマニュアルページを参照してください。

演習

1. 以下に示す `crontab` の短縮表記について、同じ意味の時間指定（ユーザー `crontab` ファイルの先頭5列）を示してください。

`@hourly`	
`@daily`	
`@weekly`	
`@monthly`	
`@annually`	

2. 以下に示す `OnCalendar` の短縮表記について、同じ意味の時間指定（正規化された長い形式）を示してください。

`hourly`	
`daily`	
`weekly`	
`monthly`	
`yearly`	

3. `crontab` ファイルにある次の時間指定の意味を説明してください。

`30 13 * * 1-5`	
`00 09-18 * * *`	
`30 08 1 1 *`	
`0,20,40 11 * * Sun`	
`00 09 10-20 1-3 *`	
`*/20 * * * *`	

4. タイマーユニットファイルの `OnCalendar` に指定する、次の時間指定の意味を説明してください。

`*-*-* 08:30:00`	
`Sat,Sun *-*-* 05:00:00`	
`*-*-01 13:15,30,45:00`	
`Fri *-09..12-* 16:20:00`	
`Mon,Tue *-*-1,15 08:30:00`	
`*-*-* *:00/05:00`	

発展演習

1. `cron` を使用してジョブをスケジュールすることを許可された一般ユーザーが、自分のcrontabファイルを作成するコマンドは何ですか？

2. 毎週金曜日の午後1時に `date` コマンドを実行する単純なジョブをスケジュールして下さい。このジョブの出力を確認するにはどうしますか？

3. `foobar.sh` スクリプトを1分ごとに実行し、その標準出力をホームディレクトリの `output.log` ファイルに書き込み、標準エラー出力をメールで送信するジョブを追加して下さい。

4. 前問で追加した `crontab` エントリを参照して下さい。標準出力を保存するファイルを絶対パスで指定する必要がないのはなぜですか？ また、`./foobar.sh` コマンドでスクリプトを実行できるのはなぜですか？

5. 追加した `crontab` エントリから出力リダイレクトを削除すると共に、1番目のcronジョブを無効にして下さい。

6. ジョブの標準出力とエラー出力を、電子メールでユーザー `emma` に送信するにはどうしますか？ また、それらのいずれも送信しないようにするにはどうしますか？

7. コマンド `ls -l /usr/bin/crontab` を実行して下さい。設定されている特別なパーミッションと、その意味は何ですか？

まとめ

このレッスンでは、以下の事柄を学びました。

- 定期的にジョブを実行するために `cron` を使用すること。
- cronジョブの管理方法。
- cronジョブをスケジュールできるユーザーの制限方法。
- `cron` の代わりとなる systemdタイマーユニットの働き。

このレッスンでは、以下に示すコマンドとファイルについて説明しました。

`crontab`
一般ユーザーの `crontab` ファイルを保守します。

`/etc/cron.allow` と `/etc/cron.deny`
`crontab` コマンドの実行を制限するために使用するファイル。

`/etc/crontab`
システムcrontabファイル。

`/etc/cron.d`
システムcrontabsファイルを置くディレクトリ。

`systemctl`
systemdとサービスマネージャーを制御します。タイマーユニットを有効にして起動する場合などに使用します。

演習の解答

1. 以下に示す `crontab` の短縮表記について、同じ意味の時間指定（ユーザー crontab ファイルの最初の5列）を示してください。

`@hourly`	`0 * * * *`
`@daily`	`0 0 * * *`
`@weekly`	`0 0 * * 0`
`@monthly`	`0 0 1 * *`
`@annually`	`0 0 1 1 *`

2. 以下に示す `OnCalendar` の短縮表記について、同じ意味の時間指定（正規化された長い形式）を示してください。

`hourly`	`*-*-* *:00:00`
`daily`	`*-*-* 00:00:00`
`weekly`	`Mon *-*-* 00:00:00`
`monthly`	`*-*-01 00:00:00`
`yearly`	`*-01-01 00:00:00`

3. `crontab` ファイルにある次の時間指定の意味を説明してください。

`30 13 * * 1-5`	月曜日から金曜日の午後1時30分
`00 09-18 * * *`	毎日午前9時から午後6時までの0分
`30 08 1 1 *`	1月1日の午前8時30分
`0,20,40 11 * * Sun`	毎週日曜日の午前11:00、午前11:20、午前11:40
`00 09 10-20 1-3 *`	1月、2月、3月の、10日から20日までの午前9時0分
`*/20 * * * *`	20分ごと

4. タイマーユニットファイルの `OnCalendar` に指定する、次の時間指定の意味を説明してください。

`*-*-* 08:30:00`	毎日午前8時30分
`Sat,Sun *-*-* 05:00:00`	土曜日と日曜日の午前5時
`*-*-01 13:15,30,45:00`	毎月1日の午後1時15分、午後1時30分、午後1時45分
`Fri *-09..12-* 16:20:00`	9月、10月、11月、12月の毎週金曜日の午後4時20分
`Mon,Tue *-*-1,15 08:30:00`	毎月1日、15日が、月曜日ないし火曜日である日の、午前8時30分

`*-*-* *:00/05:00`	5分ごと

発展演習の解答

1. `cron` を使用してジョブをスケジュールすることを許可された一般ユーザーが、自分のcrontabファイルを作成するコマンドは何ですか？

```
dave@hostname ~ $ crontab -e
no crontab for dave - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        < ---- easiest
  3. /usr/bin/emacs24
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

2. 毎週金曜日の午後1時に `date` コマンドを実行する単純なジョブをスケジュールして下さい。このジョブの出力を確認するにはどうしますか？

```
00 13 * * 5 date
```

   出力はユーザーにメールで送信されます。表示するには、`mail` コマンドを使います。

3. `foobar.sh` スクリプトを1分ごとに実行し、その標準出力をホームディレクトリの `output.log` ファイルに書き込み、標準エラー出力をメールで送信するジョブを追加して下さい。

```
*/1 * * * * ./foobar.sh >> output.log
```

4. 前問で追加した crontab エントリを参照して下さい。標準出力を保存するファイルを絶対パスで指定する必要がないのはなぜですか？ また、./foobar.sh コマンドでスクリプトを実行できるのはなぜですか？

   `crontab` ファイル内の `HOME` 環境変数で別の場所が指定されていなければ、`cron` はユーザーのホームディレクトリをカレントディレクトリとしてコマンドを呼び出します。したがって、相対パスで出力ファイルを指定し、`./foobar.sh` でスクリプトを指定できます。

5. 追加した `crontab` エントリから出力リダイレクトを削除すると共に、1番目のcronジョブを無効にして下さい。

```
#00 13 * * 5 date
*/1 * * * * ./foobar.sh
```

   cronジョブを無効にするには、`crontab` ファイル内の対応する行をコメントにします。

6. ジョブの標準出力とエラー出力を、電子メールでユーザー emma に送信するにはどうしますか？また、それらのいずれも送信しないようにするにはどうしますか？

標準出力とエラー出力を `emma` に送信するには、`crontab` ファイルで `MAILTO` 環境変数を次のように設定します。

```
MAILTO="emma"
```

`cron` にメールを送信させないためには、`MAILTO` 環境変数に空の値を割り当てます。

```
MAILTO=""
```

7. コマンド ls -l /usr/bin/crontab を実行して下さい。設定されている特別なパーミッションと、その意味は何ですか？

```
$ ls -l /usr/bin/crontab
-rwxr-sr-x 1 root crontab 25104 feb 10  2015 /usr/bin/crontab
```

`crontab` コマンドにはSGIDビットが設定されています（グループの実行可能ビットが `s`）。これは、コマンドが所有グループの権限（ここでは `crontab`）で実行されることを意味します。そのため、一般ユーザーが `crontab` コマンドを実行した時でも、`crontab` ファイルを編集できます。多くのディストリビューションでは、`crontab` ファイルを編集できるのは `crontab` コマンドのみとなるように、ファイルのパーミッションが設定されています。

107.2 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	107 管理タスク
Objective:	107.2 ジョブをスケジュールしてシステム管理タスクを自動化する
Lesson:	2 of 2

はじめに

前のレッスンで学習したように、cronやsystemdタイマーを使用して（定期的な）ジョブをスケジュールできますが、1回だけジョブを実行する必要があるかもしれしれません。これを行うには、別の強力なユーティリティである `at` コマンドを使用します。

at でジョブをスケジュールする

`at` コマンドは、ジョブを1回だけ実行する時間を指定して使用します。コマンドラインで `at` コマンドと実行するタイミングを入力すると `at` プロンプトが表示されますから、実行するコマンドを入力します。 `Ctrl`+`D` キーを入力するとプロンプトが終了します。

```
$ at now +5 minutes
warning: commands will be executed using /bin/sh
at> date
at> Ctrl+D
job 12 at Sat Sep 14 09:15:00 2019
```

上の例の `at` ジョブは、5分後に `date` コマンドを実行するだけです。`cron` と同様に、標準出力とエラー出力は電子メールで送信されます。`at` ジョブをスケジュールするには、`atd` デーモンが実行されている必要があります。

NOTE Linuxには、`at` に似た `batch` コマンドがあります。`batch` ジョブは、システムの負荷が十分に低い場合にのみ実行されます。

`at` コマンドの重要なオプションを以下に示します:

`-c`
指定したジョブIDのコマンドを標準出力に表示します。

`-d`
指定したIDを持つジョブを削除します。`atrm` のエイリアスです。

`-f`
標準入力ではなく、指定したファイルからジョブを読み込みます。

`-l`
自分の実行待ちのジョブを一覧表示します。rootの場合は、すべてのユーザーのすべてのジョブを一覧表示します。これは `atq` のエイリアスです。

`-m`
ジョブに出力がなかった場合にも、終了時にユーザーにメールを送信します。

`-q`
ジョブを投入するキューを、`a` ～ `z` と `A` ～ `Z` の1文字で指定します（デフォルトでは、`at` の場合は `a`、`batch` の場合は `b`）。高位のキュー内のジョブは、より大きなナイス値を持って実行されます。大文字のキューに送られたジョブは、`batch` ジョブとして扱われます。

`-v`
ジョブの前に、ジョブが実行されるタイミング（時刻）を表示します。

`atq` でスケジュールされたジョブを一覧表示する

では、2つの `at` ジョブをスケジュールしてみましょう: 1つ目は午前9時30分に `foo.sh` スクリプトを実行し、2つ目は1時間後に `bar.sh` スクリプトを実行します。

```
$ at 09:30 AM
warning: commands will be executed using /bin/sh
at> ./foo.sh
at> Ctrl+D
job 13 at Sat Sep 14 09:30:00 2019
$ at now +2 hours
warning: commands will be executed using /bin/sh
at> ./bar.sh
at> Ctrl+D
job 14 at Sat Sep 14 11:10:00 2019
```

実行待ちのジョブを一覧表示するには、`atq` コマンドを使用します。ジョブごとに次の情報が表示されます: ジョブID、実行される日付と時刻、キュー、ユーザー名。

```
$ atq
14      Sat Sep  14 11:10:00 2019 a frank
13      Sat Sep  14 09:30:00 2019 a frank
```



```
12      Sat Sep  14 09:15:00 2019 a frank
```

`at -l` コマンドは `atq` のエイリアスです。

NOTE rootとして `atq` を実行すると、すべてのユーザーのキューにあるジョブが表示されます。

`atrm` でジョブを削除する

`at` ジョブを削除する場合は、`atrm` コマンドに続けてジョブIDを指定します。たとえば、ID 14のジョブを削除するには、次のコマンドを実行します:

```
$ atrm 14
```

`atrm` に、空白で区切って複数のIDを指定すると、複数のジョブを削除できます。`at -d` コマンドは `atrm` のエイリアスです。

NOTE rootとして `atrm` を実行すると、すべてのユーザーのジョブを削除できます。

ジョブスケジューリングへのアクセスを制限する

`/etc/at.allow` ファイルと `/etc/at.deny` ファイルを使用して、ユーザーごとに `at` ジョブの利用を制限することができます。`/etc/at.allow` が存在する場合、その中にリストされているユーザーとrootのみが `at` ジョブをスケジュールできます。`/etc/at.allow` は存在しないが、`/etc/at.deny` が存在する場合、その中にリストされている一般ユーザーは、`at` ジョブをスケジュールできません（空の `/etc/at.deny` ファイルは、すべての一般ユーザーが `at` ジョブをスケジュールできることを意味します）。どちらのファイルも存在しない場合の挙動は、ディストリビューションによって異なります。

実行時刻の指定方法

`at` ジョブの実行時刻は、`HH:MM` の形式で指定します。12時間表記の場合は、AMまたはPMを続けます。指定の時刻を過ぎている場合は、翌日と見なされます。ジョブが実行される日付を指定したい場合は、時刻の後に次のいずれかの形式で日付を指定します: `月 日付`、`月 日付 年`、`MMDDYY`、`MM/DD/YY`、`DD.MM.YY`、`YYYY-MM-DD`（訳注: 月は英単語かその3文字省略形で指定します）。

あるいは次のようなキーワードも使用できます: `midnight`（0時）、`noon`（12時）、`teatime`（4 pm）、`now`。これらのキーワードにプラス記号（`+`）と数字、時間単位（minutes、hours、days、weeks）を続けられます。最後に、`today` ないし `tomorrow` という単語を置いて、今日と明日のどちらに実行するかを指示できます。たとえば、`at 07:15 AM Jan 01` は1月1日の07:15 amにジョブを実行し、`at now +5 minutes` は5分後にジョブを実行します。時刻表記方法の正確な定義は、`/usr/share/doc/at` にある `timespec` ファイルにあります。

`at` の代替

サービスマネージャとしてsystemdを採用しているマシンでは、`systemd-run` コマンドを使用して1回限りのタスクをスケジュールすることができます。このコマンドは通常、サービスファイルを作成することなく、指定した時刻にコマンドを実行する一時的なタイマーユニットを作成するために使用しま

す。例えば、rootとして次のコマンドを実行すれば、2019年10月6日午前11時30分に `date` コマンドを実行させられます。

```
# systemd-run --on-calendar='2019-10-06 11:30' date
```

2分後に現在の作業ディレクトリにある `foo.sh` スクリプトを実行するには、以下のようにします。

```
# systemd-run --on-active="2m" ./foo.sh
```

`systemd-run` の機能については、`systemd-run(1)` のマニュアルページを参照してください。

NOTE

タイマーはsystemdジャーナルに記録されるので、さまざまなユニットのログを `journalctl` コマンドを使用して確認できます。また、一般ユーザーとして実行する場合は、`systemd-run` と `journalctl` コマンドに `--user` オプションを使用します。

演習

1. `at` に対する時間指定として、有効なものはどれですか？

`at 08:30 AM next week`	
`at midday`	
`at 01-01-2020 07:30 PM`	
`at 21:50 01.01.20`	
`at now +4 days`	
`at 10:15 PM 31/03/2021`	
`at tomorrow 08:30 AM`	

2. `at` でスケジュールしたジョブの、コマンドを確認するにはどうしますか？

3. 実行待ちの `at` ジョブを確認するコマンドは何ですか？ また、それらを削除するコマンドは何ですか？

4. systemdにおいて、`at` の代わりに使用するコマンドは何ですか？

発展演習

1. 一般ユーザーとして、10月31日の午前10時30分に、ホームディレクトリにある `foo.sh` スクリプトを実行する `at` ジョブを作成してください。

2. 別の一般ユーザーとしてシステムにログインし、明日の午前10時に `bar.sh` スクリプトを実行する `at` ジョブを作成してください。スクリプトはそのユーザーのホームディレクトリにあるものとします。

3. さらに別の一般ユーザーとしてシステムにログインし、30分後に `foobar.sh` スクリプトを実行する `at` ジョブを作成してください。スクリプトはそのユーザーのホームディレクトリにあるものとします。

4. 次に、rootになって `atq` コマンドを実行して、ユーザーがスケジュールしたすべての `at` ジョブを確認してください。一般ユーザーとしてこのコマンドを実行するとどうなりますか？

5. rootとして、1つのコマンドで、実行待ちの `at` ジョブすべてを削除して下さい。

6. ls -l /usr/bin/at コマンドを実行し、そのパーミッションを調べてください。

まとめ

このレッスンでは、以下の事柄を学びました。

- `at` を使用して、指定の時刻に1回限りのジョブを実行すること。
- `at` ジョブの管理。
- `at` ジョブスケジューリングへのユーザーアクセスを制限すること。
- `at` の代わりに `systemd-run` を使用すること。

このレッスンでは、以下に示すコマンドとファイルについて説明しました。

`at`
指定した時刻にコマンドを実行する。

`atq`
一般ユーザーの場合、そのユーザーの `at` ジョブを一覧表示する。rootの場合はすべてのユーザーの `at` ジョブを表示する。

`atrm`
ジョブ番号を指定して `at` ジョブを削除する。

`/etc/at.allow` および `/etc/at.deny`
`at` の利用を制限する設定ファイル。

`systemd-run`
1回限り実行される `timer` ユニットを作成して開始する。

演習の解答

1. `at` に対する時間指定として、有効なものはどれですか？

`at 08:30 AM next week`	有効
`at midday`	無効
`at 01-01-2020 07:30 PM`	無効
`at 21:50 01.01.20`	有効
`at now +4 days`	有効
`at 10:15 PM 31/03/2021`	無効
`at tomorrow 08:30 AM monotonic`	無効

2. `at` でスケジュールしたジョブの、コマンドを確認するにはどうしますか？

   `at -c` コマンドの後に、コマンドを確認するジョブIDを指定します。出力には、ジョブがスケジュールされたときに有効だったほとんどの環境変数も含まれます。rootはすべてのユーザーのジョブを確認できることに注意してください。

3. 実行待ちの `at` ジョブを確認するコマンドは何ですか？ また、それらを削除するコマンドは何ですか？

   `at -l` コマンドで実行待ちのジョブを確認し、`at -d` コマンドでジョブを削除します。`at -l` は `atq` のエイリアスで、`at -d` は `atrm` のエイリアスです。rootは、すべてのユーザーのジョブを表示および削除できます。

4. systemdにおいて、at の代わりに使用するコマンドは何ですか？

   `at` の代わりに `systemd-run` コマンドを使用して、1回限りのジョブをスケジュールできます。たとえば、指定の時刻にコマンドを実行したり、任意の基準時点からの カレンダータイマー や モノトニックタイマー を定義したりできます。

発展演習の解答

1. 一般ユーザーとして、10月31日の午前10時30分に、ホームディレクトリにある `foo.sh` スクリプトを実行する `at` ジョブを作成してください。

```
$ at 10:30 AM October 31
warning: commands will be executed using /bin/sh
at> ./foo.sh
at> Ctrl+D
job 50 at Thu Oct 31 10:30:00 2019
```

2. 別の一般ユーザーとしてシステムにログインし、明日の午前10時に `bar.sh` スクリプトを実行する `at` ジョブを作成してください。スクリプトはそのユーザーのホームディレクトリにあるものとします。

```
$ at 10:00 AM tomorrow
warning: commands will be executed using /bin/sh
at> ./bar.sh
at> Ctrl+D
job 51 at Sun Oct 6 10:00:00 2019
```

3. さらに別の一般ユーザーとしてシステムにログインし、30分後に `foobar.sh` スクリプトを実行する `at` ジョブを作成してください。スクリプトはそのユーザーのホームディレクトリにあるものとします。

```
$ at now +30 minutes
warning: commands will be executed using /bin/sh
at> ./foobar.sh
at> Ctrl+D
job 52 at Sat Oct 5 10:19:00 2019
```

4. 次に、rootになって `atq` コマンドを実行して、ユーザーがスケジュールしたすべての `at` ジョブを確認してください。一般ユーザーとしてこのコマンドを実行するとどうなりますか？

```
# atq
52      Sat Oct  5 10:19:00 2019 a dave
50      Thu Oct 31 10:30:00 2019 a frank
51      Sun Oct  6 10:00:00 2019 a emma
```

`atq` コマンドをrootとして実行すると、すべてのユーザーの `at` ジョブが一覧表示されます。一般ユーザーとして実行すると、そのユーザーの `at` ジョブのみが一覧表示されます。

5. rootとして、1つのコマンドで、実行待ちの `at` ジョブすべてを削除して下さい。

```
# atrm 50 51 52
```

6. ls -l /usr/bin/at コマンドを実行し、そのパーミッションを調べてください。

```
# ls -l /usr/bin/at
-rwsr-sr-x 1 daemon daemon 43762 Dec  1  2015 /usr/bin/at
```

このディストリビューションでは、`at` コマンドにSUID（所有者の実行フラグが文字 `s`）とSGID（グループの実行フラグが文字 `s`）の両方が設定されているので、ファイルの所有者と所有グループの権限（両方とも `daemon`）で実行されます。このため、一般ユーザーでも `at` でジョブをスケジュールできます。

107.3 ローカリゼーションと国際化

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 107.3

総重量

3

主な知識分野

- ロケール設定と環境変数を設定する。
- タイムゾーン設定と環境変数を設定する。

用語とユーティリティ

- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `LC_*`
- `LC_ALL`
- `LANG`
- `TZ`
- `/usr/bin/locale`
- `tzselect`
- `timedatectl`
- `date`
- `iconv`
- `UTF-8`
- `ISO-8859`
- ASCII
- ユニコード

107.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	107 管理タスク
Objective:	107.3 ローカリゼーションと国際化
Lesson:	1 of 1

はじめに

すべての主要なLinuxディストリビューションは、国際化の機能設定をカスタマイズできます。例えば、タイムゾーンや文字エンコーディングなど、地域や言語に関する設定を、オペレーティングシステムのインストール中や、インストール後に変更できます。

それぞれのアプリケーションは、システムの設定ファイルやコマンドと環境変数を利用して、使用する言語や時刻を決定します。そのため、ほとんどのLinuxディストリビューションにおいて、時間や言語などのローカリゼーション設定を調整する方法が標準化されています。これらを調整することは、ユーザーエクスペリエンスを向上させるだけでなく、システムイベント（たとえばセキュリティ問題の報告）のタイミングを正しく決定するためにも重要です。

現代的なオペレーティングシステムは（母国語だけではない）さまざまな文字を取り扱うための　文字エンコーディング規格　　を必要としており、Linuxも例外ではありません。コンピュータは数値しか扱えないので、文字というのは図形記号に対応付けられた数値（コード）にすぎません。コンピュータープラットフォームが異なると文字に割り当てる数値が異なることがありますから、互換性を持たせるためには同じ文字エンコード規格を使う必要があります。あるシステムで作成したテキストドキュメントを別のシステムで読み取るには、文字セット（訳注:　文字とそのコード番号の対応付けを定める規格。符号化文字集合）とエンコード形式（訳注:　文字コードの並びをどのように表現するかを定める規格。文字符号化方式）の両方を揃えるか、少なくとも相互の変換方法を知っている必要があります。（訳注: 文字セットとエンコード形式は本来独立した概念ですが、併せて1つの規格としてまとめられることが少なくありません。日本では単に「文字コード」とか「エンコード」と呼ばれることが多いですが、本章では両者を併せて「文字エンコード規格」と表記しています）。

Linuxベースのシステムのローカリゼーション設定は統一されておらず、ディストリビューション間に微妙な違いがあります。とはいえ、すべてのディストリビューションは、システムを国際化するための同じ概念と基本的な設定ツールを使っています。

タイムゾーン

タイムゾーンは、1時間に相当するくさび（訳注:　リンゴのくし切りを思い出してください）で地球の全表面を区切ったものです。つまり、同じ時刻にあるとみなす地域です。1日の始まりは経度によって異なるので、経度0である　子午線　を基準にしており、そこでの時刻を　UTC（Coordinated　Universal Time/協定世界時）　と呼びます。タイムゾーンの自然な区切りは（15度おきの）経度線になりますが、実用上の理由から、国や州・県などの境界にあわせて人為的に調整されています。

行政区画との関連性が高いため、タイムゾーンの名前は通常、そのゾーン内の大国や大都市など、地域内の主要な地理的目標物にちなんで付けられます。また、あるタイムゾーンを示すために、UTCを基準にした時差を使うこともあります。たとえば、タイムゾーン　GMT-5　は、UTC時刻より5時間遅れている時刻の地域を示します。同様に、タイムゾーン　GMT+3　はUTC時間より3時間進んでいる時刻の地域を示します。GMT（グリニッジ標準時）という用語（Greenwich　Mean　Time）は、時差に基づくゾーン名表記においてUTCの同義語として使用されます。（訳注:　日本時間（JST）は　GMT+9　になり、GMTよりも9時間進んでいる（早い）タイムゾーンにあたります。）

ネットワークに接続したマシンには世界のさまざまな場所からアクセスできるため、ハードウェアクロックをUTC（GMT+0タイムゾーン）に設定し、使用場所に応じてタイムゾーンを選択するのがお勧めです。たとえば、クライアントや他のサーバーとの間で時刻が異なることを避けるために、クラウドサービスではUTCを使用するのが一般的です。対して、サーバーに対してリモートセッションを開くユーザーは、自分のローカルタイムゾーンを使用するのがお勧めです。状況に応じてそれぞれにふさわしいタイムゾーンを扱うことことは、オペレーティングシステムの働きです。

コマンド `date` は、日付と時刻に加えて、タイムゾーンも出力します。

```
$ date
Mon Oct 21 10:45:21 -03 2019
```

値　`-03`　がUTCからの時差を示していて、表示された時刻がUTCより3時間遅れていることを意味します。つまり、ローカルのタイムゾーンは　GMT-3　になります（訳注:　南米の一部やグリーンランドが該当します）。systemdを使用するディストリビューションで　`timedatectl`　を使うと、システムの時刻と日付に関する詳細を表示します。

```
$ timedatectl
                      Local time: Sat 2019-10-19 17:53:18 -03
                  Universal time: Sat 2019-10-19 20:53:18 UTC
                        RTC time: Sat 2019-10-19 20:53:18
                       Time zone: America/Sao_Paulo (-03, -0300)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
```

`Time　zone`　エントリには、地域名に基づくタイムゾーン名（`America/Sao_Paulo`　など）が示されることもあります。システムのデフォルトタイムゾーンは、`/etc/timezone`　ファイルに地域に基づくゾーン名、ないしは時差オフセットのいずれかを記入することで決定されます。UTCからの時差オフセットでタイムゾーン名を指定する場合には、名前の前に　`Etc`　を付加します。つまり、デフォルトのタイムゾーンをGMT+3に設定するには、タイムゾーンの名前を `Etc/GMT+3` と指定します。

```
$ cat /etc/timezone
```

```
Etc/GMT+3
```

地域に基づくタイムゾーン名を使えば時差を知っている必要はありませんが、正しいタイムゾーン名を選択するのはそれほど簡単ではありません。1つのタイムゾーンに複数の名前があることが多いため、時差を覚えにくいのです。`tzselect` コマンドを使うと対話的に正しいタイムゾーンを定義できるので、迷うことが少なくなります。`tzselect` は、GNU Cライブラリに関連する重要なコマンドと同じパッケージに含まれるので、ほぼすべてのLinuxディストリビューションで使用できるはずです。（訳注: タイムゾーンは

`tzselect` コマンドは、たとえば“Brazil”の“São Paulo City”のタイムゾーンを指定する場合など便利です。`tzselect` は、対象の大まかな地域を尋ねることから始めます。

```
$ tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
 3) Antarctica
 4) Asia
 5) Atlantic Ocean
 6) Australia
 7) Europe
 8) Indian Ocean
 9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the time zone using the Posix TZ format.
#? 2
```

オプション `2` は南北アメリカを示し、複数のタイムゾーンを含みます。緯度経度（オプション `10`）や時差表記（Posix TZ形式 とも呼ばれます/オプション `12`）を使用してタイムゾーンを指定することもできます。（地域を指定した場合）次のステップは国を選択することです:

```
Please select a country whose clocks agree with yours.
 1) Anguilla              19) Dominican Republic    37) Peru
 2) Antigua & Barbuda     20) Ecuador               38) Puerto Rico
 3) Argentina             21) El Salvador           39) St Barthelemy
 4) Aruba                 22) French Guiana         40) St Kitts & Nevis
 5) Bahamas               23) Greenland             41) St Lucia
 6) Barbados              24) Grenada               42) St Maarten (Dutch)
 7) Belize                25) Guadeloupe            43) St Martin (French)
 8) Bolivia               26) Guatemala             44) St Pierre & Miquelon
 9) Brazil                27) Guyana                45) St Vincent
10) Canada                28) Haiti                 46) Suriname
11) Caribbean NL          29) Honduras              47) Trinidad & Tobago
12) Cayman Islands        30) Jamaica               48) Turks & Caicos Is
13) Chile                 31) Martinique            49) United States
14) Colombia              32) Mexico                50) Uruguay
15) Costa Rica            33) Montserrat            51) Venezuela
16) Cuba                  34) Nicaragua             52) Virgin Islands (UK)
17) Curaçao               35) Panama                53) Virgin Islands (US)
```

```
18) Dominica              36) Paraguay
#? 9
```

ブラジル国内には4つのタイムゾーンがあるので、国だけではタイムゾーンが決まりません。次のステップで、`tzselect` は国内の地域を要求します:

```
Please select one of the following time zone regions.
 1) Atlantic islands
 2) Pará (east); Amapá
 3) Brazil (northeast: MA, PI, CE, RN, PB)
 4) Pernambuco
 5) Tocantins
 6) Alagoas, Sergipe
 7) Bahia
 8) Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS)
 9) Mato Grosso do Sul
10) Mato Grosso
11) Pará (west)
12) Rondônia
13) Roraima
14) Amazonas (east)
15) Amazonas (west)
16) Acre
#? 8
```

すべての地域名が利用できるわけではありませんが、最も近い地域を選択すれば十分です。`tzselect` は、指定された情報から対応するタイムゾーンを表示します。

```
The following information has been given:

        Brazil
        Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS)

Therefore TZ='America/Sao_Paulo' will be used.
Selected time is now:   sex out 18 18:47:07 -03 2019.
Universal Time is now:  sex out 18 21:47:07 UTC 2019.
Is the above information OK?
1) Yes
2) No
#? 1

You can make this change permanent for yourself by appending the line
        TZ='America/Sao_Paulo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
America/Sao_Paulo
```

結果のタイムゾーン名 `America/Sao_Paulo` を `/etc/timezone` に書き込めば、システムのデフォルトタイムゾーンを設定できます。 （訳注: 日本では `Asia - Japan` を指定すると、`Asia/Tokyo` が表示されます。）

```
$ cat /etc/timezone
America/Sao_Paulo
```

`tzselect` の出力にも示されますが、システムのデフォルトのタイムゾーンにかかわらず、環境変数 `TZ` がシェルセッションのタイムゾーンを定義します。行 `TZ='America/Sao_Paulo'; export TZ` をファイル `~/.profile` に記入して、`TZ` 環境変数をエクスポートすると、 `America/Sao_Paulo` が以後のセッションのタイムゾーンになります。異なるタイムゾーンの時刻を表示するために、現在のセッション中に `TZ` 環境変数を一時的に変更することもできます。

```
$ env TZ='Africa/Cairo' date
Mon Oct 21 15:45:21 EET 2019
```

この `env` コマンドの例では、引数 `TZ='Africa/Cairo'` によって `TZ` 環境変数が変更され、その他は現在のセッションと同じ環境変数を使用して、新しいサブシェルセッションでコマンド（`date`）を実行します。

サマータイム

多くの地域ではサマータイムを採用しており、夏期の時刻を1時間早めています。そのため、システム設定が誤っていると、誤った時刻が表示されてしまうことがあります。 `/etc/localtime` ファイルには、オペレーティングシステムが時計を調整するためのデータが含まれています。標準的なLinuxシステムでは、`/usr/share/zoneinfo/` ディレクトリにすべてのタイムゾーンのファイルが置かれ、`/etc/localtime` はそのディレクトリ内のデータファイルへのシンボリックリンクになっています。`/usr/share/zoneinfo/` 内のファイルは、対応するタイムゾーンの名前で整理されていて、例えば `America/Sao_Paulo` タイムゾーンのデータファイルは `/usr/share/zoneinfo/America/Sao_Paulo` になります。

サマータイムの定義は（地域の都合で）変更されることがあるため、`/usr/share/zoneinfo/` 内のファイルを最新の状態に保つことが重要です。タイムゾーンファイルが新しいバージョンに更新されるたびに、ディストリビューションのパッケージ管理ツールでupgradeコマンドを使ってそれらを更新する必要があります。

言語と文字エンコード

Linuxシステムでは、`ロケール（locale）` と呼ばれる、様々な言語や非西洋の文字エンコーディングを使用することができます。環境変数 `LANG` を定義することで、最も基本的なロケールの設定を行い、ほとんどのプログラムはこの変数から使用する言語を決定します。

`LANG` 変数の値は `ab_CD` という形式であり、ここで `ab` は言語コード、`CD` は地域コードです。言語コードはISO-639（言語名コード規格）に、地域コードはISO-3166（国名コード規格）に、それぞれ準拠します。たとえば、ブラジルのポルトガル語を使用する場合は、`LANG` 変数に `pt_BR.UTF-8` を定義します。（訳注: 日本語の場合は `ja_JP.UTF-8` など。）

```
$ echo $LANG
pt_BR.UTF-8
```

サンプルに示すように、LANG　変数には（言語と地域だけでなく）文字エンコードも含まれます。電子通信で広く使用された最初の文字エンコード規格は、ASCII（アスキー/American Standard Code for Information Interchange）です。ASCIIは、英語のアルファベットに基づいているため、使用可能な数値の範囲が限られ、アルファベット以外の文字や、言語に固有の記号類は使えません。UTF-8エンコーディングは、西洋文字だけでなく漢字などを含む全世界の文字を一元化した Unicode規格 の一部です。規格の管理者である Unicode Consortium では、コンピュータープラットフォーム間の互換性を確保するために、デフォルトでUTF-8を採用することを推奨しています。

[Unicode Consortium による Unicodeとは？ を引用]

> Unicode規格は、言語やプラットフォーム、デバイス、アプリケーションなどに関係なく、すべての文字に一意の番号を割り当てるものです。現在では、ソフトウェアプロバイダのほとんどすべてが採用しており、さまざまなプラットフォーム、デバイス、アプリケーション間で、情報を交換できるようになりました。Unicodeをサポートすることで、主要なオペレーティングシステム、検索エンジン、ブラウザ、ラップトップ、スマートフォン、さらにインターネットとWorld Wide Web（URL、HTML、XML、CSS、JSONなど）で、さまざまな言語や文字を表現するための基盤となっています。（..中略..）Unicode規格とそれをサポートするツールを利用することは、グローバルなソフトウェアテクノロジにおける最も重要なトレンドの1つです。

システムによっては、非ASCII文字のエンコードにISOに基づく規格（ISO-8859-1規格など）を使用している場合があります。現在では、このような（Unicode以前の）文字エンコードは非推奨として、Unicodeエンコードを採るべきです。すべての主要なオペレーティングシステムは、デフォルトでUnicodeを採用しています。

システム全体のロケールは、/etc/locale.conf　ファイルで設定します。このファイルで、シェル変数と同様に LANG などのロケール関連の環境変数を割り当てます。次に例を示します:

```
$ cat /etc/locale.conf
LANG=pt_BR.UTF-8
```

LANG　環境変数を再定義することで、ユーザーごとに独自のロケールを設定できます。~/.bash_profile や　~/.profile　などのBashプロファイルに定義を追加すれば、自らのセッションのロケールが変更されます。ただし、ユーザーセッションの外ではデフォルトのシステムロケールが有効となるので、ディスプレイマネージャのログイン画面などは影響を受けません。

TIP　システムマネージャーに　systemd　を採用しているシステムに備わっている　localectl　コマンドで、システムロケールの参照や変更を行うこともできます。例： localectl set-locale LANG=en_US.UTF-8 。

LANG　環境変数以外にも、通貨記号や数値の区切り方など、ある種のロケールを変更する環境変数があります:

LC_COLLATE
文字列の並び替え順を設定します。ファイルやディレクトリをリストする場合などの表示順序が変更されます。

LC_CTYPE
文字の分類方法を設定します。たとえば、どの文字を 大文字 または 小文字 と見なすかなどを定義します。（訳注: 日本語ではひらがな・カタカナ、約物（句読点やカンマ、ピリオドなど）などの判定方法が影響を受けます。）

LC_MESSAGES
プログラムが表示する文字列の言語を設定します。

LC_MONETARY
通貨記号と表現形式を設定します。（訳注: 日本語では円マークと3桁ごとのカンマなどが影響を受けます。）

LC_NUMERIC
金額以外の数値の表現形式を設定します。主に3桁ごとの区切り記号や小数点を表す文字など。

LC_TIME
日付や時刻の表現形式を設定します。

LC_PAPER
標準の用紙サイズを設定します。

LC_ALL
LANG を含む、ロケールに関わるすべての環境変数をオーバーライドします。

`locale` コマンドは、現在のロケール構成で定義されているすべての環境変数を表示します。

```
$ locale
LANG=pt_BR.UTF-8
LC_CTYPE="pt_BR.UTF-8"
LC_NUMERIC=pt_BR.UTF-8
LC_TIME=pt_BR.UTF-8
LC_COLLATE="pt_BR.UTF-8"
LC_MONETARY=pt_BR.UTF-8
LC_MESSAGES="pt_BR.UTF-8"
LC_PAPER=pt_BR.UTF-8
LC_NAME=pt_BR.UTF-8
LC_ADDRESS=pt_BR.UTF-8
LC_TELEPHONE=pt_BR.UTF-8
LC_MEASUREMENT=pt_BR.UTF-8
LC_IDENTIFICATION=pt_BR.UTF-8
LC_ALL=
```

この例では、`LC_ALL` のみが未定義で、これを使用してすべてのロケール設定を一時的にオーバーライドできます。次の例は、システムロケールが `pt_BR.UTF-8` であるマシンで、`LC_ALL` 変数をオーバーライドして `date` コマンドを実行します:

```
$ date
seg out 21 10:45:21 -03 2019
$ env LC_ALL=en_US.UTF-8 date
```

```
Mon Oct 21 10:45:21 -03 2019
```

`LC_ALL` 変数を変更したので、月名と曜日がアメリカ英語（`en_US`）で表示されています。なお、すべての変数に同じロケール値を設定する必要はありません。たとえば、言語を `pt_BR` として、数値形式（`LC_NUMERIC`）を米国式とすることができます。

ローカライゼーションの設定によって、プログラムが文字列の並び順や数値の書式を扱う方法が変わります。ほとんどのプログラムは一般的なロケールを正しく処理できますが、一部の（古い）スクリプトなどは、例えば文字列の並び順を変更すると予期しない動作をすることがあります。このような場合には、`LANG=C` として環境変数 `LANG` に `C` ロケールを設定すると、うまく動くことがあります。Cロケールは、（ロケールの概念が登場する前の）伝統的なUNIX環境を指定するもので、ASCIIエンコードを使用して単純なバイト列として文字列の比較を行います。国際化の処理を行わないので単純で高速ですが、米国英語以外の言語はほぼ使えないと考えて下さい。

エンコーディング変換

異なる文字エンコーディングのシステムで作成したテキストを表示すると、意味不明な文字が表示されることがあります。`iconv` コマンドを使うと、ファイルの文字エンコーディングを別のものに変換することができます。たとえば、ISO-8859-1（訳注: ラテンアルファベット191文字を、1文字1バイトで表現する規格）でエンコードされている `original.txt` というファイルを、UTF-8 に変換して `converted.txt` というファイルに書き込むには、以下のコマンドを使用します:

```
$ iconv -f ISO-8859-1 -t UTF-8 original.txt > converted.txt
```

オプション `-f ISO-8859-1`（`--from-code=ISO-8859-1`）は元のファイルのエンコーディングを、オプション `-t UTF-8`（`--to-code=UTF-8`）は、変換後のファイルのエンコーディングをそれぞれ指定します。`iconv` コマンドがサポートしているすべてのエンコーディングは、`iconv -l` または `iconv --list` で一覧表示されます。例では出力のリダイレクトを使用していますが、`-o converted.txt`（`--output converted.txt`）オプションで出力ファイルを指定することもできます。

訳注: Unicode以前に、日本語かな漢字では大別して3種類のエンコーディングが使用されていました: 1)JIS漢字コード（ISO-2022-JPなど）、2)日本語EUC（EUC-JPなど）、3)シフトJIS漢字（SJISなど）。現在、これらの文字エンコードを使用する事はほとんどありませんが、古い文書ファイルなどを読み出す時に変換が必要となることがあります。日本語に特化した `nkf`（Network Kanji filter）という変換コマンドが、ほとんどのLinuxディストリビューションで利用できます（インストールが必要です）から、必要な場合には調べてみて下さい。`iconv` は規格に則った融通の利かないツールですが、`nkf` は実情に合わせた柔軟な処理を行い、現在もメンテナンスされています。

演習

1. 以下の `date` コマンドの出力から推測して、システムのタイムゾーンをGMT表記で書いて下さい。

```
$ date
Mon Oct 21 18:45:21 +05 2019
```

2. システムデフォルトのローカルタイムを `Europe/Brussels` にする場合、`/etc/localtime` シンボリックリンクが指すファイルは何ですか？

3. システムの文字エンコーディングと、テキストファイルの文字エンコーディングが異なる場合、正しく表示されないことがあります。`iconv` を使用して、WINDOWS-1252でエンコードされたファイル `old.txt` を、UTF-8 でエンコーディングされた `new.txt` ファイルに変換するにはどうしますか？

発展演習

1. 現在のシェルセッションで、タイムゾーンを `Pacific/Auckland` とするにはどうしますか？

2. `uptime` コマンドは、システムの ロードアベレージ を小数で表示します。小数点をドットで表記するかコンマで表記するかは、ロケール設定によって異なります。たとえば、ロケールが `de_DE.UTF-8`（ドイツの標準ロケール）の場合はコンマですし、`en_US.UTF-8`（アメリカ英語）ではドットです。現在のセッションで `uptime` コマンドが少数を表示する際に、ドットを使うようにするコマンドはどうなりますか？

3. `iconv` コマンドは、変換先の文字セットには存在しない文字をすべて疑問符に置き換えます。変換先のエンコーディングの末尾に `//TRANSLIT` を追加すると、変換先では存在しない文字を、1つ以上の類似した文字に置き換えます（字訳）。この機能を使用して、 `readme.txt` という名前のUTF-8テキストファイルを `ascii.txt` という名前のプレーンASCIIファイルに変換するにはどうすればよいですか？（注: 日本語では使用できない機能です。）

まとめ

このレッスンでは、Linuxシステムの言語とタイムゾーンを設定する方法について説明しました。テキストを正しく表示するために非常に重要であるため、文字エンコードの概念と設定についても説明しました。このレッスンでは、以下ののトピックについて説明しました。

- Linuxシステムにおいて、シェルセッションのメッセージを表示する言語を選択する方法。
- 現地時刻とタイムゾーンの概念。
- タイムゾーンを選択して、システム設定を変更する方法。
- 文字エンコードの概念と、それらを変換する方法。

以下のコマンドと手順を紹介しました:

- ロケールと時刻に関連する環境変数: `LC_ALL`、`LANG`、`TZ` など。
- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `locale`
- `tzselect`
- `timedatectl`
- `date`
- `iconv`

演習の解答

1. 以下の `date` コマンドの出力から推測して、システムのタイムゾーンをGMT表記で書いて下さい。

   ```
   $ date
   Mon Oct 21 18:45:21 +05 2019
   ```

   `Etc/GMT+5` タイムゾーンです。

2. システムデフォルトのローカルタイムを `Europe/Brussels` にする場合、`/etc/localtime` シンボリックリンクが指すファイルは何ですか？

   リンク `/etc/localtime` は、 `/usr/share/zoneinfo/Europe/Brussels` を指している必要があります。

3. システムの文字エンコーディングと、テキストファイルの文字エンコーディングが異なる場合、正しく表示されないことがあります。`iconv` を使用して、WINDOWS-1252でエンコードされたファイル `old.txt` を、UTF-8 でエンコーディングされた `new.txt` ファイルに変換するにはどうしますか？

   `iconv -f WINDOWS-1252 -t UTF-8 -o new.txt old.txt` で行えます。

発展演習の解答

1. 現在のシェルセッションで、タイムゾーンを `Pacific/Auckland` とするにはどうしますか？

   `export TZ=Pacific/Auckland`

2. `uptime` コマンドは、システムの ロードアベレージ を小数で表示します。小数点をドットで表記するかコンマで表記するかは、ロケール設定によって異なります。たとえば、ロケールが `de_DE.UTF-8`（ドイツの標準ロケール）の場合はコンマですし、`en_US.UTF-8`（アメリカ英語）ではドットです。現在のセッションで `uptime` コマンドが少数を表示する際に、ドットを使うようにするコマンドはどうなりますか？

   `export LC_NUMERIC=en_US.UTF-8` または `export LC_ALL=en_US.UTF-8`

3. `iconv` コマンドは、変換先の文字セットには存在しない文字をすべて疑問符に置き換えます。変換先のエンコーディングの末尾に `//TRANSLIT` を追加すると、変換先では存在しない文字を、1つ以上の類似した文字に置き換えます（字訳）。この機能を使用して、 `readme.txt` という名前のUTF-8テキストファイルを `ascii.txt` という名前のプレーンASCIIファイルに変換するにはどうすればよいですか？（注: 日本語では使用できない機能です。）

   `iconv -f UTF-8 -t ASCII//TRANSLIT -o ascii.txt readme.txt` で行えます。

Linux
Professional
Institute

課題 108: 必須システムサービス

108.1 システム時刻を管理する

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 108.1

総重量

3

主な知識分野

- システムの日付と時刻を設定する。
- ハードウェアクロックをUTCで正しい時刻に設定する。
- 正しいタイムゾーンを設定する。
- ntpdとchronyを利用した、基本的なNTP設定。
- pool.ntp.orgサービスの使用に関する知識。
- ntpqコマンドの知識。

用語とユーティリティ

- `/usr/share/zoneinfo/`
- `/etc/timezone`
- `/etc/localtime`
- `/etc/ntp.conf`
- `/etc/chrony.conf`
- `date`
- `hwclock`
- `timedatectl`
- `ntpd`
- `ntpdate`
- `chronyc`
- `pool.ntp.org`

108.1 レッスン 1

Certificate:	LPIC-1 (102)
Version:	5.0
Topic:	108 基本的なシステムサービス
Objective:	108.1 システムの時刻を保守する
Lesson:	1 of 2

はじめに

現代のコンピューティングでは、正確な時刻管理が必須です。しかし、時刻管理の実装は驚くほど複雑です。エンドユーザーにとっては、正確な時刻の維持は簡単なことのように思えるかもしれませんが、システムは多様なケースや例外をインテリジェントに処理する必要があります。タイムゾーンは、管理の都合や政治上の決定によって変更されることがあるので、静的ではありません。国がサマータイムを止める選択をする可能性もあります。すべてのプログラムは、これらの変更を論理的に処理する必要があります。システム管理者にとって幸いなことに、Linuxオペレーティングシステムにおける時刻管理は成熟していて堅牢であり、通常は保守する必要があまりありません。

Linuxコンピュータが起動するとすぐに、時間のカウントが始まります。オペレーティングシステムがカウントするものを　システムクロック　と呼びます。また、最近のコンピューターには、ハードウェアまたは　リアルタイムクロック　も備えています。ほとんどのマザーボードはハードウェアクロックを備えていて、コンピュータが起動されているかどうかに関わりなく時刻を更新しています。起動の際にはハードウェアクロックからシステムクロックが設定されますが、その他のほとんどの場合は、これら2つのクロックは互いに独立して機能しています。このレッスンでは、システムクロックとハードウェアクロックの両方を操作する方法について説明します。

最近のほとんどのLinuxシステムでは、システムクロックとハードウェアクロックの両方が、Network Time Protocol（NTP）によって　ネットワーク時刻　に同期されます。ほとんどの場合、ユーザーはタイムゾーンを設定するだけで、残りはNTPが処理します。ここでは手動で時刻を調整する方法を説明し、次のレッスンでネットワーク時刻の調整について説明します。

ローカルタイムとユニバーサルタイム

システムクロックは、英国グリニッジの現地時刻でもある協定世界時（UTC）に設定されています。ユーザーは自分の　ローカルタイム　を知りたいことが多いでしょう。UTC時刻を取得し、タイムゾーンと

サマータイムに基づく オフセット を適用することで、ローカルタイム（現地時刻）を得ることができます。この方法で、多くの複雑さを回避できます。

システムクロックは、UTC時刻ないし現地時刻のいずれにも設定できますが、UTC時刻に設定することがお勧めです。

dateコマンド

`date` コマンドは、デフォルトでは現地時刻を現地言語で出力します。（訳注：例は米国東部標準時EST（UTC-0500）です。）

```
$ date
Sun Nov 17 12:55:06 EST 2019
```

`date` コマンドのオプションを使って、出力を変更できます。

たとえば、`date -u` は、現在のUTC時刻を表示します。

```
$ date -u
Sun Nov 17 18:02:51 UTC 2019
```

オプションを指定することで、さまざまな形式で現在時刻を表示することができます。

`-I`
: ISO 8601形式で日付と時刻を返します。`-I` のみ、ないし `-Idate` を指定すると、日付のみを表示します。`date` の他に `hours`、`minutes`、`seconds`、`ns`（ナノ秒）を指定でき、指定した精度までの時刻を表示します。（訳注:ISO 8601は、日付と時刻を表記方法を規定する国際規格です。たとえば日本時間2020年7月15日14時36分15.23秒は、`20200715T143615.23+0900`（基本形式）ないし `2020-07-15T14:36:15.23+0900`（拡張形式）と表記します。詳しくはWikipediaなどを参照して下さい。）

`-R`
: RFC 5322形式で日付と時刻を返します。（訳注:RFC 5322は、電子メールにおけるテキストメッセージのフォーマットを規定する規格です。たとえば日本時間2020年7月15日14時36分15.23秒は、`Wed, 15 Jul 2020 14:36:15 +0900` と表示されます。この形式がLinuxのデフォルトです。）

`--rfc-3339`
: RFC 3339形式で日付と時刻を返します。（訳注:RFC 3339は、インターネット上で日時の表現方法を規定する規格です。ISO 8601とほぼ同じ書式ですが、よりコンピューターで処理しやすい形で定義されています。あまり使用されていません。）

`date` の表示形式は、manページに記載されたシーケンス（メタ文字）を使用してカスタマイズできます。たとえば次のようにすると、現在時刻をUnix時刻（後述）で表示します。

```
$ date +%s
1574014515
```

`date` のマニュアルページを参照すれば、`%s` がUnix時刻を表すことがわかります。

Unix時刻とは、ほとんどのUnix系システムが内部的に使用している時刻表現です。UTCの1970年1月1日0時0分0秒を Epoch （訳注: 記念日、起源などの意）と呼び、そこからの秒数で時刻を表します。

NOTE

現時点でUnix時刻を保持するために必要なビット数は32ビットです。将来、2038年1月19日には、32ビットではUnix時刻を保持できなくなるために、32ビットLinuxシステムで深刻な問題が発生することが指摘されています。（訳注:64ビットLinuxシステムでは既に解決されています。）

これらのシーケンスを使用すれば、さまざまなアプリケーションが期待する形式で日付と時刻をフォーマットできます。もちろん、規格に基づく標準的なフォーマットを使用するに超したことはありません。

また、`date --date` を使用すると、現在時刻ではなく指定した時刻を、標準的な形式で表示できます。たとえば、Unix時刻で日時を指定するとこうなります。

```
$ date --date='@1564013011'
Wed Jul 24 20:03:31 EDT 2019
```

`--debug` オプションを使用すると、日時の解析プロセスを確認することができます。有効な日付をコマンドに渡してみましょう。

```
$ date --debug --date="Fri, 03 Jan 2020 14:00:17 -0500"
date: parsed day part: Fri (day ordinal=0 number=5)
date: parsed date part: (Y-M-D) 2020-01-03
date: parsed time part: 14:00:17 UTC-05
date: input timezone: parsed date/time string (-05)
date: using specified time as starting value: '14:00:17'
date: warning: day (Fri) ignored when explicit dates are given
date: starting date/time: '(Y-M-D) 2020-01-03 14:00:17 TZ=-05'
date: '(Y-M-D) 2020-01-03 14:00:17 TZ=-05' = 1578078017 epoch-seconds
date: timezone: system default
date: final: 1578078017.000000000 (epoch-seconds)
date: final: (Y-M-D) 2020-01-03 19:00:17 (UTC)
date: final: (Y-M-D) 2020-01-03 14:00:17 (UTC-05)
```

このオプションは、日時を生成するアプリケーションのトラブルシューティングを行うときに便利です。

ハードウェアクロック

`hwclock` コマンドは、リアルタイムクロックの時刻を表示します。このコマンドの実行にはroot権限が必要なので、`sudo` を使用してコマンドを呼び出します。

```
$ sudo hwclock
2019-11-20 11:31:29.217627-05:00
```

`--verbose` オプションを使用すると、トラブルシューティングに役立つであろうより多くの出力が返されます。

```
$ sudo hwclock --verbose
hwclock from util-linux 2.34
System Time: 1578079387.976029
Trying to open: /dev/rtc0
Using the rtc interface to the clock.
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
...got clock tick
Time read from Hardware Clock: 2020/01/03 19:23:08
Hw clock time : 2020/01/03 19:23:08 = 1578079388 seconds since 1969
Time since last adjustment is 1578079388 seconds
Calculated Hardware Clock drift is 0.000000 seconds
2020-01-03 14:23:07.948436-05:00
```

`Calculated Hardware Clock drift` 行に着目しましょう。この行は、システムクロックとハードウェアクロックのズレを示しています。

timedatectlコマンド

`timedatectl` コマンドは、日付と時刻の状態を確認するためのコマンドで、ネットワーク時刻との同期状態も表示します。（ネットワークタイムプロトコルについては次のレッスンで取り上げます）。

デフォルトの `timedatectl` は、`date` と同様の情報を返しますが、RTC（ハードウェア）時刻とNTPサービスの状態が追加されています。

```
$ timedatectl
               Local time: Thu 2019-12-05 11:08:05 EST
           Universal time: Thu 2019-12-05 16:08:05 UTC
                 RTC time: Thu 2019-12-05 16:08:05
                Time zone: America/Toronto (EST, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

timedatectlを使用した時間の設定

`date` や `hwclock` で時刻を設定することもできますが、`systemd` を採用しているシステムでNTPが使用できない場合は、`timedatectl` コマンドで時刻を設定するのがお勧めです。

```
# timedatectl set-time '2011-11-25 14:00:00'
```

HH:MM:SS の形式を使用して、時刻のみを設定することもできます。（後述しますが、`date` コマンドでも同様です。）

timedatectlを使用したタイムゾーンの設定

`systemd` ベースのLinuxシステムにおいて、GUIツールが使えない場合には、`timedatectl` を使って

ローカルタイムゾーンを設定します。`timedatectl` で、まず指定できるタイムゾーンの一覧を表示して、そのいずれかを引数として使用してタイムゾーンを設定します。

まず、指定できるタイムゾーンをリストします。

```
$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Algiers
Africa/Bissau
Africa/Cairo
...
```

タイムゾーンのリストは長いので、`grep` コマンドを使うのがお勧めです。（訳注：リストはページャ経由で表示されるので、その検索機能を使うのもよいでしょう。）

次に、返されたリストから1つを選び、タイムゾーンを設定します:

```
$ timedatectl set-timezone Africa/Cairo
$ timedatectl
               Local time: Thu 2019-12-05 18:18:10 EET
           Universal time: Thu 2019-12-05 16:18:10 UTC
                 RTC time: Thu 2019-12-05 16:18:10
                Time zone: Africa/Cairo (EET, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

タイムゾーンの名前を正確に指定することに注意してください。たとえば、タイムゾーンを `Africa/Cairo` と指定できますが、`cairo` や `africa/cairo` と指定することはできません。タイムゾーン名は、ディストリビューションによって異なります。

timedatectlを使用したNTPの無効化

NTPを無効にしたいことがあります。`systemctl` を使うこともできますが、ここでは `timedatectl` コマンドを使います:

```
# timedatectl set-ntp no
$ timedatectl
              Local time: Thu 2019-12-05 18:19:04 EET Universal time: Thu 2019-12-05 16:19:04 UTC
                RTC time: Thu 2019-12-05 16:19:04
               Time zone: Africa/Cairo (EET, +0200)
             NTP enabled: no
        NTP synchronized: no
         RTC in local TZ: no
              DST active: n/a
```

timedatectlを使わずにタイムゾーンを設定する

タイムゾーンを設定することは、Linuxを新規インストールする場合の標準的な手順のひとつです。グラフィカル画面によるインストールでは、ほとんどの場合、何かを入力する必要もありません。

`/usr/share/zoneinfo` ディレクトリには、さまざまなタイムゾーン情報が含まれています。`zoneinfo` ディレクトリには、大陸名などのサブディレクトリないしはシンボリックリンクがあります。大陸名から始めて、自分の地域の `zoneinfo` を見つけるとよいでしょう。

`zoneinfo` ファイルには、UTCに対する現地時間のオフセットを計算するために必要な、サマータイムを含むルールが含まれています。Linuxがローカルタイムゾーンを決定する場合は、`/etc/localtime` を読み取ります。GUIを使用せずにタイムゾーンを設定するには、`/etc/localtime` を `/usr/share/zoneinfo` 内の自分の地域のzoneinfoを指すシンボリックリンクとして作成します。例を示します。

```
$ ln -s /usr/share/zoneinfo/Canada/Eastern /etc/localtime
```

正しいタイムゾーンを設定したら、以下のコマンドを実行します:

```
# hwclock --systohc
```

これにより、システムクロック に ハードウェアクロック をあわせます（つまり、リアルタイムクロックが、`date` が表示するのと同じ時刻にセットされます）。このコマンドを実行するには、root権限が必要なことに注意してください。

`/etc/timezone` は、`/etc/localtime` と似た役割を持っています。ローカルタイムゾーンの名前を保持しているので、`cat` で読み取ることができます。

```
$ cat /etc/timezone
America/Toronto
```

Linuxディストリビューションによっては、このファイルが使われてないこともあります。

timedatectlを使わずに日時を設定する

NOTE

最近のほとんどのLinuxシステムは、構成とサービスに `systemd` を使用しているので、時刻の設定に `date` や `hwclock` を使用することは避けましょう。`systemd` は、`timedatectl` を使って両方の時刻を変更します。とはいえ、古いシステムを管理する場合には、これらのレガシーコマンドを知っておくことが重要です。

dateコマンドを使う

`date` コマンドには、システム時刻をセットするオプションがあります。`--set` ないし `-s` を使用して、日付と時刻をセットします。`--debug` オプションを使って、コマンドの解析内容を確認することもできます。

```
# date --set="11 Nov 2011 11:11:11"
```

日時をセットするには、root権限が必要であることに注意してください。時間と日付を別々に変更することもできます。

```
# date +%Y%m%d -s "20111125"
```

この場合、文字列を適切に解釈するように、形式を示すシーケンスを指定する必要があります。たとえば、`%Y` は年を表すので、最初の4桁 `2011` は2011年として解釈されます。時刻のシーケンスは `%T` ですから、以下に時間をセットする例を示します:

```
# date +%T -s "13:11:00"
```

システムクロックを変更した後には、システムクロックとハードウェアクロックを同期するために、ハードウェアクロックも変更するとよいでしょう。

```
# hwclock --systohc
```

`systohc` は、“システムクロックからハードウェアクロックへ” を意味します。（SYStem to Hardware Clockの略です。）

hwclockを使う

システムクロックをセットしてハードウェアクロックをあわせるのではなく、逆にハードウェアクロックをセットしてシステムクロックをあわせても構いません。

```
# hwclock --set --date "4/12/2019 11:15:19"
# hwclock
Fri 12 Apr 2019 6:15:19 AM EST -0.562862 seconds
```

`hwclock` にはUTC時刻をセットするのが普通ですが、返されるのは現地時刻であることに注意してください。

ハードウェアクロックをセットしたら、それにシステムクロックをあわせます。`hctosys` は “ハードウェアクロックからシステムクロック” を意味しています。

```
# hwclock --hctosys
```

演習

1. 表のコマンドが　システムクロック　と　ハードウェアクロック　のどちらを表示ないし変更するかを示してください。

コマンド	システム	ハードウェア	両方
`date -u`			
`hwclock --set --date "12:00:00"`			
`timedatectl`			
`timedatectl | grep RTC`			
`hwclock --hctosys`			
`date +%T -s "08:00:00"`			
`timedatectl set-time 1980-01-10`			

2. 以下の出力を見て、正しく解釈されるように引数の書式を修正してください。

```
$ date --debug --date "20/20/12 0:10 -3"

date: warning: value 20 has less than 4 digits. Assuming MM/DD/YY[YY]
date: parsed date part: (Y-M-D) 0002-20-20
date: parsed time part: 00:10:00 UTC-03
date: input timezone: parsed date/time string (-03)
date: using specified time as starting value: '00:10:00'
date: error: invalid date/time value:
date:     user provided time: '(Y-M-D) 0002-20-20 00:10:00 TZ=-03'
date:        normalized time: '(Y-M-D) 0003-08-20 00:10:00 TZ=-03'
date:                                 ---- --
date:      possible reasons:
date:        numeric values overflow;
date:        incorrect timezone
date: invalid date  ‘20/20/2 0:10 -3’
```

3. `date`　　コマンドのシーケンスを使用して、システムクロックの月を2月にセットしてください。年と日、時刻は変更してはいけません。

4. 前問のコマンドが成功したと仮定して、`hwclock`　　を使ってシステムクロックにハードウェアクロックをあわせてください。

5. `Eucla` と呼ばれる地名があります。それはどの大陸にありますか？ `grep` コマンドを使って調べて下さい。

6. タイムゾーンを `Eucla` のタイムゾーンに設定してください。

発展演習

1. 時計を合わせる最適な方法は何ですか？ また、その方法が使えない場面はどのようなものでしょう？

2. システム時刻をセットする方法がたくさんあるのはなぜだと思いますか？

3. 2038年1月19日以降、システム時刻を保存するには64ビットの数値が必要になります。あるいは、“Epochを変更する” ことでも問題を解決できます。たとえば、2038年1月1日の深夜0時を、新しいEpochの0とすればよいのです。この解決方法を採用しない理由を考察して下さい。

まとめ

このレッスンでは、以下の事柄を学びました:

- コマンドラインから、日時をさまざまな形式で表示する方法。
- Linuxのシステムクロックとハードウェアクロックの違い。
- システムクロックを手動で設定する方法。
- ハードウェアクロックを手動で設定する方法。
- システムのタイムゾーンを変更する方法。

このレッスンでは、以下のコマンドを説明しました。

`date`
システムクロックを表示または変更します。主なオプション：

`-u`
UTC時刻を表示します。

`+%s`
Unix時刻を指定するシーケンス。

`--date=`
現在時刻ではなく、指定した時刻を表示します。

`--debug`
指定した日時の文字列を解析するときのデバッグメッセージを表示します。

`-s`
システムクロックを手動設定します。

`hwclock`
ハードウェアクロックを表示または変更します。

`--systohc`
システムクロックにハードウェアクロックをあわせます。

`--hctosys`
ハードウェアクロックにシステムクロックをあわせます。

`--set --date`
ハードウェアクロックを手動で設定します。

`timedatectl`
systemdベースのLinuxシステムにおいて、システムとハードウェアクロックの時刻と、NTPの状態を表示します。

`set-time`
時間を手動で設定します。

`list-timezones`
指定できるタイムゾーンを一覧表示します。

`set-timezone`
timzoneを手動で設定します。

`set-ntp`
NTPの有効/無効を切り替えます。

演習の解答

1. 表のコマンドが　システムクロック　と　ハードウェアクロック　のどちらを表示ないし変更するかを示してください。

コマンド	システム	ハードウェア	両方
`date -u`	√		
`hwclock --set --date "12:00:00"`		√	
`timedatectl`			√
`timedatectl | grep RTC`		√	
`hwclock --hctosys`	√		
`date +%T -s "08:00:00"`	√		
`timedatectl set-time 1980-01-10`			√

2. 以下の出力を見て、正しく解釈されるように引数の書式を修正してください。

```
$ date --debug --date "20/20/12 0:10 -3"

date: warning: value 20 has less than 4 digits. Assuming MM/DD/YY[YY]
date: parsed date part: (Y-M-D) 0002-20-20
date: parsed time part: 00:10:00 UTC-03
date: input timezone: parsed date/time string (-03)
date: using specified time as starting value: '00:10:00'
date: error: invalid date/time value:
date:     user provided time: '(Y-M-D) 0002-20-20 00:10:00 TZ=-03'
date:        normalized time: '(Y-M-D) 0003-08-20 00:10:00 TZ=-03'
date:                                 ---- --
date:      possible reasons:
date:        numeric values overflow;
date:        incorrect timezone
date: invalid date  ‘20/20/2 0:10 -3’
```

   `date --debug --set "12/20/20 0:10 -3"`

3. `date`　　コマンドのシーケンスを使用して、システムクロックの月を2月にセットしてください。年と日、時刻は変更してはいけません

   `date +%m -s "2"`

4. 前問のコマンドが成功したと仮定して、`hwclock`　　を使ってシステムクロックにハードウェアクロックをあわせてください。 `hwclock -systohc`
5. `Eucla`　と呼ばれる地名があります。それはどの大陸にありますか？　`grep`　コマンドを使って調べて

下さい。

```
timedatectl list-timezones | grep -i eucla
```

または

```
grep -ri eucla /usr/share/zoneinfo
```

6. タイムゾーンを `Eucla` のタイムゾーンに設定してください。

   ```
   timedatectl set-timezone 'Australia/Eucla'
   ```

   または、

   ```
   ln -s /usr/share/zoneinfo/Australia/Eucla /etc/localtime
   ```


発展演習の解答

1. 時計を合わせる最適な方法は何ですか？ また、その方法が使えない場面はどのようなものでしょう？

   ほとんどのLinuxディストリビューションでは、NTPがデフォルトで有効になっているので、干渉しないようにそのままにしておきます。ただし、インターネットに接続されていないLinuxシステムは、NTPにアクセスできません。たとえば、産業用機器で実行されている組み込みLinuxシステムでは、ネットワーク接続がないことがあります。

2. システム時刻をセットする方法がたくさんあるのはなぜだと思いますか？

   何十年も前から、すべてのUnix系システムは正確な時刻を必要としていました。そのため、時刻を設定するための多くのレガシーメソッドが残されています。

3. 2038年1月19日以降、システム時刻を保存するには64ビットの数値が必要になります。あるいは、“Epochを変更する”　ことでも問題を解決できます。たとえば、2038年1月1日の深夜0時を、新しいEpochの0とすればよいのです。この解決方法を採用しない理由を考察して下さい。

   2038年までには、大半のコンピュータが64ビットCPUを搭載して、64ビットの数値を使用してもパフォーマンスが大きく低下することはないと予測されます。対して、このような “リセット” する方法のリスクを見積もることは不可能です。影響を受ける可能性があるレガシーソフトウェアがたくさんあります。例えば、銀行や大企業は古いプログラムを大量に抱え、それに依存していることが少なくありません。つまり、この解決方法は、他の多くの解決方法と同様に、トレードオフの問題です。2038年にまだ稼働している32ビットシステムはEpoch時刻のオーバーフローの影響を受けますし、レガシーソフトウェアは変更されたEpochの値の影響を受けることになります。

108.1 レッスン 2

Certificate:	LPIC-1 (102)
Version:	5.0
Topic:	108 基本的なシステムサービス
Objective:	108.1 システムの時刻を保守する
Lesson:	2 of 2

はじめに

パーソナルコンピューターはそれ自体でかなり正確な時刻を維持することができますが、ネットワーク環境で稼働するコンピューターシステムは、より正確な時刻を維持する必要があります。最も正確な時刻を刻む原子時計が　基準時計　として使われています。現在では、インターネットに接続されたすべてのコンピュータをこの基準時計に同期させる、Network Time Protocol（NTP）と呼ばれるシステムが考案されて使われています。NTPを実行するコンピュータは、そのシステムクロックを基準時計から提供される時刻に同期させることができます。システムクロックとNTPサーバーから提供された時刻が異なる場合、コンピューターは、システムクロックがネットワーク時刻に一致するまで、内部のシステムクロックを段階的に調整します。

NTPは階層構造で時刻を配信します。階層の一番上にあるサーバーには、基準時計が接続されます。最上位のサーバーは　Stratum　1　（訳注:　地層、階層の意）と呼ばれ、一般には公開されません。Stratum 1のマシンは Stratum 2 マシンと同期し、Stratam 2のマシンは Stratum 3 のマシンと同期します。Stratum　2以下の階層のサーバーは、広く一般に公開されます。大きなネットワーク用にNTPを設定する場合、少数のコンピュータがStratum 2以下のサーバーに接続し、そのマシンが他のマシンすべてにNTPを提供するのが一般的です。これにより、Stratum 2のマシンに対するリクエストを最低限に抑えることができます。

NTPを理解するに当たって、重要な用語がいくつかあります。これらの用語のいくつかは、それぞれのマシンにおけるNTPのステータスを追跡するためのコマンドで使用されます。

Offset（オフセット）
: システムクロックとNTPクロックの絶対差を指します。たとえば、システムクロックが12:00:02で、NTPクロックが11:59:58の場合、2つのクロック間のオフセットは4秒です。

Step（ステップ）
NTPプロバイダー（正確な時刻を提供する側）とコンシューマー（時刻を受け取って利用する側）の間のオフセットが128msより大きい場合、NTPはシステム時刻を徐々に調整するのではなく、1回でシステム時刻を大きく変更します。これを stepping（ステッピング） と呼びます。

Slew（スルー）
システム時刻とNTPの間のオフセットが128ms未満の場合に、システム時刻を徐々に変更することをSlewing（スルーイング） と言います。

Insane Time（不適切な時刻）
システム時刻とNTPの間のオフセットが17分より大きい場合、システム時刻は　不適切　であると見なされ、NTPはシステム時刻を変更しません。システム時刻を適切な時刻から17分以内にするには、特別な手順（手動設定など）を実行する必要があります。

Drift（ドリフト）
2つのクロックが時間の経過とともに同期しなくなる現象をドリフトと呼びます。同期していた2つのクロックが、時間の経過とともに同期しなくなる場合に、クロックドリフトが発生していると言います。

Jitter（ジッター）
最後にクロックが同期されてからのドリフト量をジッターと呼びます。例えば、17分前に最後のNTP同期が行われて、NTPプロバイダーとコンシューマーの間のオフセットが3ミリ秒である場合、ジッターは3ミリ秒です。

ここから、LinuxにおけるNTPの実装をいくつか説明します。

timedatectlコマンド

`timedatectl`　を使用しているLinuxディストリビューションでは、デフォルトではフルセットのNTPではなく SNTP（Simple Network Time Protocol）クライアントを実行します。これはネットワーク時刻の軽便な実装であり、ネットワーク上の他のマシンにNTPを提供することはできません。

SNTP機能を実行するには、`timesyncd`　サービスを実行する必要があります。他のsystemdサービスと同様に、次のコマンドで実行状態を確認できます。

```
$ systemctl status systemd-timesyncd
● systemd-timesyncd.service - Network Time Synchronization
   Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/systemd-timesyncd.service.d
           └─disable-with-time-daemon.conf
   Active: active (running) since Thu 2020-01-09 21:01:50 EST; 2 weeks 1 days ago
     Docs: man:systemd-timesyncd.service(8)
 Main PID: 1032 (systemd-timesyn)
   Status: "Synchronized to time server for the first time 91.189.89.198:123 (ntp.ubuntu.com)."
    Tasks: 2 (limit: 4915)
   Memory: 3.0M
   CGroup: /system.slice/systemd-timesyncd.service
           └─1032 /lib/systemd/systemd-timesyncd

Jan 11 13:06:18 NeoMex systemd-timesyncd[1032]: Synchronized to time server for the first time
91.189.91.157:123 (ntp.ubuntu.com).
```

```
...
```

`timedatectl` によるSNTPの同期状態は、`show-timesync` を使用して確認できます。

```
$ timedatectl show-timesync --all
LinkNTPServers=
SystemNTPServers=
FallbackNTPServers=ntp.ubuntu.com
ServerName=ntp.ubuntu.com
ServerAddress=91.189.89.198
RootDistanceMaxUSec=5s
PollIntervalMinUSec=32s
PollIntervalMaxUSec=34min 8s
PollIntervalUSec=34min 8s
NTPMessage={ Leap=0, Version=4, Mode=4, Stratum=2, Precision=-23, RootDelay=8.270ms, RootDispersion=18.432ms,
Reference=91EECB0E, OriginateTimestamp=Sat 2020-01-25 18:35:49 EST, ReceiveTimestamp=Sat 2020-01-25 18:35:49
EST, TransmitTimestamp=Sat 2020-01-25 18:35:49 EST, DestinationTimestamp=Sat 2020-01-25 18:35:49 EST,
Ignored=no PacketCount=263, Jitter=2.751ms }
Frequency=-211336
```

システムの時計を正確に保つにはこれだけで十分ですが、前述のようにネットワーク上の複数のクライアントを同期させたい場合には不十分です。そのような場合には、完全なNTPパッケージをインストールします。

NTPデーモン

システムクロックを定期的にネットワーク時刻と比較するには、バックグラウンドで　デーモン　を実行する必要があります。多くのLinuxシステムでは、`ntpd`　デーモンが使われています。`ntpd`　を使用すると、時間を受け取る　（つまり、ネットワーク時刻に自分のシステムクロックを同期できる）だけでなく、他のマシンに時間を 提供する こともできます。

systemdベースのシステムにおいて、デーモンを制御するために　`systemctl`　を使用しているとしましょう。パッケージマネージャーで　`ntp`　パッケージをインストールしたら、ステータスをチェックして `ntpd` が実行されていることを確認します。

```
$ systemctl status ntpd

● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-12-06 03:27:21 EST; 7h ago
  Process: 856 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 867 (ntpd)
   CGroup: /system.slice/ntpd.service
           `-867 /usr/sbin/ntpd -u ntp:ntp -g
```

場合によっては、`ntpd`　の有効化と起動の両方が必要になることがあります。ほとんどのLinuxマシンでは、次のように実行します。

```
# systemctl enable ntpd && systemctl start ntpd
```

NTPのクエリは、TCP/UDPのポート123を使用します（訳注:主にはUDPが使用されますが、必要に応じてTCPが使われることもあります）。NTPが失敗した場合は、これらのポートが開放されていることを確認してください。

NTPの設定

NTPはいくつかの情報源をポーリングして、システムクロックの設定に使用する候補サーバーを選択します。ネットワーク接続が失われた場合、NTPは履歴から必要な調整量を推定し、その後も調整を続けます。

ディストリビューションによって、NTPサーバーのリストが保存される位置が異なります。`ntp` がインストールされているものとします。

ネットワーク時間と同期するための情報は、ファイル `/etc/ntp.conf` に置かれています。このファイルは、`vi` や `nano` で編集できます。

デフォルトでは、以下のセクションに使用するNTPサーバーが指定されています。

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

NTPサーバーを追加するには、次の構文を使います:

```
server IPアドレス
server サーバーのホスト名
```

DNSが適切に設定されていれば、サーバーのアドレスはIPアドレスかホスト名で指定できます（訳注:サーバーの変更などに備えて、ホスト名で指定することが推奨されています）。指定したサーバーには、常にNTPクエリが送られます。

NTPサーバーには、複数のNTPプロバイダーをグループにまとめた pool を指定することもできます。poolには複数のNTPプロバイダーがあり、すべてが同じ時刻に同期されているNTPデーモンを実行していると想定できます。クライアントがプールにクエリを送るたびに、1つのプロバイダーがランダムに選択されます。これにより、プール内の1台のマシンがすべてのNTPクエリを処理するのではなく、ネットワーク負荷を多くのマシンに分散することができます。

一般的に、`/etc/ntp.conf` には `pool.ntp.org` などのサーバープールを指定します。Linuxディストリビューターが、デフォルトのサーバープールを運用していることもあります。

pool.ntp.org

デフォルトで使用される pool.ntp.org は、オープンソースプロジェクトです。詳細は、 ntppool.org を参照してください。

> ntppoolの利用がニーズに合っているかを検討してください。ビジネスや組織、生活が、正しい時間に依存している場合、あるいは時刻が間違っていることで損害を受ける可能性がある場合は、“インターネットから取得する” べきではありません。ntppoolは概して非常に高品質ですが、ボランティアが空き時間を使って運用しているサービスです。信頼性の高いサービスをセットアップしたい場合は、機器やサービスのベンダーに相談してください。利用規約もご覧ください。Meinbergのタイムサーバーをお勧めしますが、End Run、Spectracomなど、他にも多くのタイムサーバーがあります。
>
> — ntppool.org

（訳注:日本国内から利用する場合は jp.pool.ntp.org を指定するか、ISPが提供しているNTPサーバー（ないしはプール）を使用すると良いでしょう。 国立研究開発法人情報通信機構や、 インターネットマルチフィード株式会社なども、公共のNTPサービスを運用しています）。

ntpdateコマンド

初期セットアップ中など、システムクロックとNTPが大幅に異なることがあります。システムクロックとNTクロックの オフセット が17分を超える場合、NTPデーモンはシステムクロックを変更しません。この場合、手作業による介入が必要です。

まず、`ntpd` が実行中であれば、サービスを 停止 します。`systemctl stop ntpd` で行えます。

次に、`ntpdate pool.ntp.org` コマンドを実行して、最初の1回限りの同期を行います。ここで `pool.ntp.org` は、NTPサーバーのIPアドレスまたはホスト名です。複数回の同期が必要な場合もあります。

ntpqコマンド

`ntpq` は、NTPの状態を確認するユーティリティです。NTPデーモンを設定し起動した後に、`ntpq` を使用してその状態を確認できます。

```
$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+37.44.185.42    91.189.94.4      3 u   86  128  377  126.509  -20.398   6.838
+ntp2.0x00.lv    193.204.114.233  2 u   82  128  377  143.885   -8.105   8.478
*inspektor-vlan1 121.131.112.137  2 u   17  128  377  112.878  -23.619   7.959
 b1-66er.matrix. 18.26.4.105      2 u  484  128   10   34.907   -0.811  16.123
```

ここでの `-p` オプションは peer（通信相手） を意味していて、同期相手の要約を出力します。`-n` を使用して、ホスト名ではなくIPアドレスを返すこともできます。

`remote`
NTPプロバイダーのホスト名

refid
NTPプロバイダーの参照ID

st
プロバイダーの階層（Stratum）

when
最後のクエリからの秒数

poll
クエリ間の秒数（ポーリング間隔）

reach
サーバーに到達したかどうかを示すステータスID。直近8回のクエリが成功したか失敗したかを示すビット列の8進数表示。1,3,7,10,11…377と増えていきます。

delay
クエリから応答までの時間（ミリ秒単位）

offset
システム時刻とNTP時刻の差（ミリ秒単位）

jitter
直近のクエリにおけるシステム時刻とNTP時刻のオフセット（ミリ秒単位）

オプションや引数なしで `ntpq` を実行すると、対話モードに入ります。`?` を入力すると、`ntpq` が解釈できるコマンドのリストを表示します。

chrony

`chrony` は、もうひとつのNTP実装です。一部のLinuxシステムにはデフォルトでインストールされていて、主要なすべてのディストリビューションでダウンロード可能です。`chronyd` はchronyデーモンであり、`chronyc` はコマンドラインインターフェイスです。`chronyc` を操作する前に、`chronyd` を有効にして起動する必要があります。

chronyが設定されていれば、`chronyc tracking` コマンドで、NTPとシステムクロックの情報が表示されます。

```
$ chronyc tracking
Reference ID    : 3265FB3D (bras-vprn-toroon2638w-lp130-11-50-101-251-61.dsl.)
Stratum         : 3
Ref time (UTC)  : Thu Jan 09 19:18:35 2020
System time     : 0.000134029 seconds fast of NTP time
Last offset     : +0.000166506 seconds
RMS offset      : 0.000470712 seconds
Frequency       : 919.818 ppm slow
Residual freq   : +0.078 ppm
Skew            : 0.555 ppm
Root delay      : 0.006151616 seconds
Root dispersion : 0.010947504 seconds
```

```
Update interval : 129.8 seconds
Leap status     : Normal
```

他の実装に比べると、この出力には多くの情報が含まれています。

`Reference ID`
コンピューターが同期しているホスト名と参照ID

`Stratum`
NTPプロバイダーの階層

`Ref time`
NTPプロバイダを参照した時のUTC時刻

`System time`
NTP時刻とシステムクロックの差違

`Last offset`
最後のクロック更新における推定オフセット

`RMS offset`
オフセット値の長期的な平均値

`Frequency`
chronydがシステムクロックを修正しない場合に予想されるシステムクロックの誤差レート。ppm（100万分の1秒）単位。

`Residual freq`
基準ソースからの周波数とシステムクロックの周波数との差を示す余剰周波数

`Skew`
周波数の推定誤差限界

`Root delay`
コンピューターが同期している最上位のコンピューターへの、ネットワークによる遅延の合計

`Leap status`
時刻調整のステータス。正常（normal）、秒を挿入（insert second）、秒を削除（delete second）、非同期（not synchronized）のいずれか

最後のNTP更新に関する詳細情報も確認できます。

```
# chrony ntpdata
Remote address  : 172.105.97.111 (AC69616F)
Remote port     : 123
Local address   : 192.168.122.81 (C0A87A51)
Leap status     : Normal
Version         : 4
Mode            : Server
```

```
Stratum         : 2
Poll interval   : 6 (64 seconds)
Precision       : -25 (0.000000030 seconds)
Root delay      : 0.000381 seconds
Root dispersion : 0.000092 seconds
Reference ID    : 61B7CE58 ()
Reference time  : Mon Jan 13 21:50:03 2020
Offset          : +0.000491960 seconds
Peer delay      : 0.004312567 seconds
Peer dispersion : 0.000000068 seconds
Response time   : 0.000037078 seconds
Jitter asymmetry: +0.00
NTP tests       : 111 111 1111
Interleaved     : No
Authenticated   : No
TX timestamping : Daemon
RX timestamping : Kernel
Total TX        : 15
Total RX        : 15
Total valid RX  : 15
```

最後に、`chronyc sources` は時刻の同期に使用するNTPサーバーの情報を返します。

```
$ chronyc sources
210 Number of sources = 0
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
210 Number of sources = 0
```

この例では、このマシンには提供元が設定されていません。chronyの設定ファイルを開いて、`pool.ntp.org` などのソースを追加できます。設定ファイルは通常、`/etc/chrony.conf` です。このファイルを開くと、いくつかのサーバーがデフォルトでリストされていることでしょう。

```
==============================================================================
# Most computers using chrony will send measurement requests to one or
# more 'NTP servers'.  You will probably find that your Internet Service
# Provider or company have one or more NTP servers that you can specify.
# Failing that, there are a lot of public NTP servers.  There is a list
# you can access at http://support.ntp.org/bin/view/Servers/WebHome or
# you can use servers from the 3.arch.pool.ntp.org project.

! server 0.arch.pool.ntp.org iburst iburst
! server 1.arch.pool.ntp.org iburst iburst
! server 2.arch.pool.ntp.org iburst iburst

! pool 3.arch.pool.ntp.org iburst
```

これらのサーバーは、独自のサーバーを指定する際の構文ガイドとしても有用です。この場合は、各行の先頭にある `!` を削除してコメントを削除して、デフォルトの `pool.ntp.org` プロジェクトのサーバ

ーを使用します。

また、このファイルで、スキュー、ドリフト、ドリフトファイルとキーファイルの位置などの、デフォルト設定を変更することもできます。

インストール直後など、最初にクロックを大幅に修正する可能性がある時は、次のコメントを解除して大幅な時刻変更を可能としておきます。

```
! makestep 1.0 3
```

設定ファイルを変更したら、`chronyd` サービスを再起動して、`chronyc makestep` でシステムクロックを更新します。

```
# chronyc makestep
200 OK
```

さらに前述のように `chronyc tracking` を使用して、変更が行われたことを確認するとよいでしょう。

演習

1. 定義と用語の表を完成して下さい。

定義	用語
ネットワーク時間を提供するコンピューター	
基準時計からの距離（ホップまたはステップ単位）	
システム時刻とネットワーク時刻の差	
前回のNTPポーリング以降の、システム時刻とネットワーク時刻の差	
負荷分散された、ネットワーク時刻を提供するサーバー群	

2. 以下の値を出力できるコマンドをチェックしてください。

値	`chronyc tracking`	`timedatectl show-timesync --all`	`ntpq -pn`	`chrony ntpdata`	`chronyc sources`
Jitter（ジッター）					
Drift（ドリフト）					
Interval of Poll（ポーリング間隔）					
Offset（オフセット）					
Stratum（層）					
IP Address of Provider（プロバイダーのIPアドレス）					
Root Delay（ルートからの遅延）					

3. 1台のLinuxサーバーと複数のLinuxデスクトップで構成される、企業ネットワークを設定しています。サーバーの固定IPアドレスは192.168.0.101です。サーバーが `pool.ntp.org` に接続し、デスクトップにNTP時刻を提供します。サーバーとデスクトップの設定はどうなりますか？。

4. Linuxマシンの時計があっていません。NTPのトラブルシューティングをどう行いますか？

発展演習

1. SNTPとNTPを比較して下さい。

SNTP	NTP

2. システム管理者が `pool.ntp.org` を使用 しない のはどんな時ですか？

3. システム管理者が、`pool.ntp.org` プロジェクトに参加ないし貢献するにはどうしますか？

まとめ

このレッスンでは、以下の事柄を学びました:

- NTPとは何か、また、なぜそれが重要なのか
- `pool.ntp.org` プロジェクトを利用するNTPデーモンの構成
- `ntpq` を使用してNTPの設定を確認する方法
- 別のNTPサービスとして `chrony` を使用する方法

このレッスンでは、以下のコマンドを説明しました。

`timedatectl show-timesync --all`
`timedatectl` を使用している場合に、SNTP情報を表示します。

`ntpdate <address>`
1回限りのNTPステップを実行します。

`ntpq -p`
最近のNTPポーリング状況を表示します。`-n` オプションはホスト名ではなくIPアドレスを表示します。

`chronyc tracking`
chronyを使用している場合に、NTPの状態を表示します。

`chronyc ntpdata`
直近のポーリングに関するNTP情報を表示します。

`chronyc sources`
NTPプロバイダーに関する情報を表示します。

`chronyc makestep`
chronyを使用している場合に、1回限りのNTPステップを実行します。

演習の解答

1. 定義と用語の表を完成して下さい。

定義	用語
ネットワーク時間を提供するコンピューター	Provider （プロバイダー）
基準時計からの距離（ホップまたはステップ単位）	Stratum（ストラタム）
システム時刻とネットワーク時刻の差	Offset（オフセット）
前回のNTPポーリング以降の、システム時刻とネットワーク時刻の差	Jitter（ジッター）
負荷分散された、ネットワーク時刻を提供するサーバー群	Pool（プール）

2. 以下の値を出力できるコマンドをチェックしてください。

値	`chronyc tracking`	`timedatectl show-timesync --all`	`ntpq -pn`	`chrony ntpdata`	`chronyc sources`
Jitter（ジッター）		√	√		
Drift（ドリフト）					
Interval of Poll（ポーリング間隔）	√	√	√（`when` 列）	√	√
Offset（オフセット）	√		√	√	
Stratum（階層）	√	√	√	√	√
IP Address of Provider（プロバイダーのIPアドレス）		√	√	√	√
Root Delay（ルート遅延）	√			√	

3. 1台のLinuxサーバーと複数のLinuxデスクトップで構成される、企業ネットワークを設定しています。サーバーの固定IPアドレスは192.168.0.101です。サーバーが `pool.ntp.org` に接続し、デスクトップにNTP時刻を提供します。サーバーとデスクトップの設定はどうなりますか？。

   サーバーでSNTPではなくntpdサービスが実行されていることを確認します。`/etc/ntp.conf` ないし `/etc/chrony.conf` ファイルに、`pool.ntp.org` プールを指定します。クライアントでは、それぞれの `/etc/ntp.conf` ないし `/etc/chrony.conf` ファイルに `192.168.0.101` を指定します。

4. Linuxマシンの時計があっていません。NTPのトラブルシューティングはどう行いますか？

   まず、マシンがインターネットに接続されていることを確認します。これには `ping` を使います。`systemctl status ntpd` ないし `systemctl status systemd-timesyncd` を使って、ntpdないしSNTPサービスが起動していることを確認します。エラーメッセージが有用な情報を提供することがあります。最後に、`ntpq -p` や `chrony tracking` などのコマンドを使って、リクエストが行われたことを確認します。システム時刻がネットワーク時刻と大幅に異なる場合は、システム時刻が“不適切”と見なして、手作業による介入が必要です。これには、`ntpdate pool.ntp.org` コマンドで1回限りのntp同期を実行するか、前のレッスンのコマンドを使います。

発展演習の解答

1. SNTPとNTPを比較してください。

SNTP	NTP
精度が低い	精度が高い
必要なリソースが少ない	多くのリソースが必要
時刻を提供することはできない	時刻を提供することができる
単発変更のみ	単発変更と徐々に同期の両方
1つのソースに時刻を要求	複数のNTPサーバーを調べて最適なプロバイダーを利用

2. システム管理者が pool.ntp.org を使用 しない のはどんな時ですか？

   ntppool.orgには次のように書かれています:　正しい時刻を維持することが絶対的に重要な場合は、別の方法を検討する必要があります。例えば、インターネットプロバイダーがNTPサービスを提供している場合は、それを使用した方がよいでしょう。（訳注:ネットワーク的に「近い」サーバーを選択すれば、応答速度のばらつき（ジッター）が少ないために同期の精度の高まると言われています。）

3. システム管理者が、`pool.ntp.org` プロジェクトに参加ないし貢献するにはどうしますか？

   ntppool.orgには次のように書かれています:　サーバーには固定IPアドレスと継続的なインターネット接続が必要です。固定IPアドレスは変更されないことが望ましく、少なくとも1年以上は変更されないことが必要です。それに対して、帯域幅の要件は控えめで、384〜512Kビットの帯域幅が求められます。Stratum 3ないし4のサーバーの参加を歓迎します。

108.2 システムロギング

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 108.2

総重量

4

主な知識分野

- rsyslogの基本的な設定。
- 標準的なファシリティ、プロパティ、アクションの理解。
- systemdジャーナルへの問い合わせ。
- 日付・サービス・優先度などによるsystemdジャーナルのデータをフィルタする。
- 永続systemdジャーナルストレージとジャーナルサイズの設定する。
- systemdの古いジャーナルデータを削除する。
- レスキューシステムやファイルシステムのコピーから、systemdジャーナルデータを探す。
- systemd-journaldのrsyslogのやり取りを理解している。
- logrotateの設定。
- syslogとsyslog-ngの知識。

用語とユーティリティ

- `/etc/rsyslog.conf`
- `/var/log/`
- `logger`
- `logrotate`
- `/etc/logrotate.conf`
- `/etc/logrotate.d/`
- `journalctl`
- `systemd-cat`
- `/etc/systemd/journald.conf`

- `/var/log/journal/`

108.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	108 基本的なシステムサービス
Objective:	108.2 システムロギング
Lesson:	1 of 2

はじめに

ログは、システム管理者の強い味方です。ログとは、起動した瞬間からシステムやネットワークに関するすべてのイベントを時系列順に記録したファイル（通常はテキストファイル）です。したがって、ログに含まれる情報には、認証の失敗、プログラムやサービスのエラー、ファイアウォールでブロックされたホストなど、システムに関するほぼすべてのイベントが含まれます。トラブルシューティング、リソースのチェック、プログラムの異常な動作の検出など、ログはシステム管理者の日々の業務に非常に重要であることが容易に想像できるでしょう。

このレッスンでは、現在のGNU/Linuxディストリビューションで最も一般的なログ記録機構の一つである `rsyslog` について説明します。さまざまなログの種類、どこに保存されていてどのような情報を含んでいるか、それらの情報を取得してフィルタリングする方法を学びます。また、ログをIPネットワーク上のあるサーバに保存する方法、ログローテーション、カーネル リング バッファについても説明します。

システムロギング

カーネルやシステム内のさまざまなプロセスが起動して互いに通信し始めると、メッセージという形で多くの情報が生成され、多くの場合はログに記録されます。

ログがなければシステム管理者がサーバーで発生したイベントを見つけることは困難ですから、あらゆるシステムイベントを記録する標準的な集中管理方法を備えることが重要です。ログは、トラブルシューティングとセキュリティに関して極めて有用であり、システム統計を把握して傾向を予測するための信頼できるデータソースになります。

とりあえず `systemd-journald` は、次のレッスンで説明するので置いておきます。ロギングは従

来、syslog、syslog-ng（syslog新世代）、ならびに rsyslog（r は ”rocketのように高速” の意）という、3種の専用サービスによって処理されてきました。rsyslog には（RELPサポートなどの）重要な改良が追加されたので、現在、最も人気のある選択肢になっています。これらのサービスは、他のサービスやプログラムからメッセージを集めて、通常は /var/log の下のログファイルに保存します。ただし、中には独自にログを処理するサービス（たとえば、Apache HTTPD Webサーバーや、CUPS印刷システム）もあります。また、Linuxカーネルは、ログメッセージを格納するためにメモリ内のリングバッファを使用します。

NOTE RELP は Reliable Event Logging Protocol の略で、syslogプロトコルを拡張して、リモートマシンにログを転送する場合の信頼性を担保する新しいプロトコルです。旧来のsyslogプロトコルは単純なUDPパケットを使用しているため、ログメッセージが届かないことがあり得ます。

ほぼすべての主要なディストリビューションで rsyslog が事実上標準のロギング機構として使用されているので、このレッスンではそれに焦点を当てます。rsyslog はクライアントサーバーモデルを使用します。クライアントとサーバーは、同じホストにあっても、異なるホストにあっても構いません。メッセージは規定の形式で送受信され、ネットワーク全体で集中管理された rsyslog サーバーに保持することができます。rsyslogデーモン（rsyslogd）は、klogd（カーネルメッセージを管理する）と連携して動作します。次のセクションでは、rsyslog とその仕組みについて説明します。

NOTE デーモンとは、バックグラウンドで実行されるサービスです。デーモン名末尾の d に着目してください: klogd や rsyslogd。

ログの種類

ログは 変化していく データであるため、通常は /var/log に置かれます。ログを大まかに分類すると、システムログ と サービスないしアプリケーションプログラムのログ に分類できます。

いくつかのシステムログと、それらが保持する情報を見てみましょう。

NOTE ログファイルの名前や書き込まれる情報は、ディストリビューションによって異なっています。ここで取り上げている例は、主にDebian系のものです。ディストリビューションによらない標準的なものは明記しています。

/var/log/auth.log
: 認証処理に関する活動: ユーザーのログイン、sudo の実行、cronジョブ、失敗したログイン試行など。

/var/log/syslog
: rsyslogd に集約されたほぼすべてのログを納めるファイル。非常に多くの種類のログが集まるので、/etc/rsyslog.conf による設定で、他のファイルに分散することもあります。（主にDebian系ディストリビューション）

/var/log/debug
: プログラムからのデバッグ情報。（訳注: ほとんど使われません。）

/var/log/kern.log
: カーネルからのメッセージ。

/var/log/messages
: さまざまなサービス（カーネルを除く）からのメッセージ。（主にRedHat系ディストリビューショ

ン）

/var/log/daemon.log
バックグラウンドで実行されているデーモン（サービス）に関する情報。

/var/log/mail.log
メールサービス（postfixなど）に関する情報。

/var/log/Xorg.0.log
グラフィックカードに関する情報。

/var/run/utmp と /var/log/wtmp
成功したログインに関する情報。（どのディストリビューションにも共通）

/var/log/btmp
失敗したログインに関する情報。例えば、sshへのブルートフォース攻撃など。（どのディストリビューションにも共通）

/var/log/faillog
失敗した認証に関する情報。

/var/log/lastlog
ユーザーが最後にログインした日時。（どのディストリビューションにも共通）

次に、サービスに関するログの例をいくつか見てみましょう。

/var/log/cups/
印刷システム（Common Unix Printing System）のログを格納するディレクトリ。一般的なデフォルトでは、次のログファイルが置かれます: `error_log`、`page_log`、`access_log`。

/var/log/apache2/ ないし /var/log/httpd
Apache Webサーバー のログを格納するディレクトリ。一般的なデフォルトでは、次のログファイルが置かれます: `access.log`、`error_log`、`other_vhosts_access.log`。

/var/log/mysql
MySQLリレーショナルデータベース管理システム のログを格納するディレクトリ。一般的なデフォルトでは、以下のログファイルが置かれます: `error_log`、`mysql.log`、`mysql-slow.log` 。

/var/log/samba/
Samba（Windows Serverと互換性のあるファイル、プリント、ディレクトリなどのサービスを提供するサーバー）のログを格納するディレクトリ。一般的なデフォルトでは、以下のログファイルが置かれます: `smbd.log`、`nmbd.log` など。

ログの読み出し

ログファイルを読み出すには、まず、rootユーザーであるか、ファイルに対する読み取り権限を持っていることを確認してください。その上で、次のようなさまざまなユーティリティが使用できます。

`less` ないし `more`

ページごとに表示やスクロールを行うページャー。

```
root@debian:~# less /var/log/auth.log
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
Sep 12 18:49:46 debian sshd[905]: Accepted password for carol from 192.168.1.65 port 44296 ssh2
Sep 12 18:49:46 debian sshd[905]: pam_unix(sshd:session): session opened for user carol by (uid=0)
Sep 12 18:49:46 debian systemd-logind[331]: New session 2 of user carol.
Sep 12 18:49:46 debian systemd: pam_unix(systemd-user:session): session opened for user carol by (uid=0)
(...)
```

`zless` ないし `zmore`

`less` や `more` と同じですが、`gzip`（`logrotate` の標準機能）で圧縮されたログを扱えます。（訳注: ディストリビューションによっては `less` の拡張機能が有効になっていて、`less` コマンドで圧縮ファイルを読み出せます。）

```
root@debian:~# zless /var/log/auth.log.3.gz
Aug 19 20:05:57 debian sudo:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ; COMMAND=/sbin/shutdown -h
now
Aug 19 20:05:57 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)
Aug 19 20:05:57 debian lightdm: pam_unix(lightdm-greeter:session): session closed for user lightdm
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event2 (Power Button)
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event3 (Sleep Button)
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event4 (Video Bus)
Aug 19 23:50:49 debian systemd-logind[333]: New seat seat0.
Aug 19 23:50:49 debian sshd[409]: Server listening on 0.0.0.0 port 22.
(...)
```

`tail`

ファイルの最後の行を表示します（デフォルトは10行です）。tailの威力は `-f` スイッチにあり、新しい行が追加されるとすぐに表示します。

```
root@suse-server:~# tail -f /var/log/messages
2019-09-14T13:57:28.962780+02:00 suse-server sudo: pam_unix(sudo:session): session closed for user root
2019-09-14T13:57:38.038298+02:00 suse-server sudo:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/usr/bin/tail -f /var/log/messages
2019-09-14T13:57:38.039927+02:00 suse-server sudo: pam_unix(sudo:session): session opened for user root by
carol(uid=0)
2019-09-14T14:07:22+02:00 debian carol: appending new message from client to remote server...
```

`head`

ファイルの最初の行を表示します（デフォルトは10行です）。

```
root@suse-server:~# head -5 /var/log/mail
2019-06-29T11:47:59.219806+02:00 suse-server postfix/postfix-script[1732]: the Postfix mail system is not
running
2019-06-29T11:48:01.355361+02:00 suse-server postfix/postfix-script[1925]: starting the Postfix mail system
2019-06-29T11:48:01.391128+02:00 suse-server postfix/master[1930]: daemon started -- version 3.3.1,
configuration /etc/postfix
2019-06-29T11:55:39.247462+02:00 suse-server postfix/postfix-script[3364]: stopping the Postfix mail system
2019-06-29T11:55:39.249375+02:00 suse-server postfix/master[1930]: terminating on signal 15
```

`grep`
指定した文字列を抽出する、フィルタリングユーティリティです。

```
root@debian:~# grep "dhclient" /var/log/syslog
Sep 13 11:58:48 debian dhclient[448]: DHCPREQUEST of 192.168.1.4 on enp0s3 to 192.168.1.1 port 67
Sep 13 11:58:49 debian dhclient[448]: DHCPACK of 192.168.1.4 from 192.168.1.1
Sep 13 11:58:49 debian dhclient[448]: bound to 192.168.1.4 -- renewal in 1368 seconds.
(...)
```

実行例で気づいたかもしれませんが、ログファイルの出力には次の項目が含まれます（訳注:　タイムスタンプの書式が異なる例が混ざっていますが、rsyslogではエントリごとに書式を設定する事ができます（LPIC-1範囲外です）。ほとんどの場合は、英語表記の現地時間です。）:

- タイムスタンプ（日時）
- メッセージを送信したホスト名
- メッセージを生成したプログラム/サービスの名前
- メッセージを生成したプログラムのPID
- 実行されたアクションの説明（プログラム/サービスが出力したメッセージ）

テキストではなくバイナリファイルのログがいくつかあり、解析するには専用のコマンドが必要となります（訳注: 重要なツールについては、トピック110.1で説明します。）:

`/var/log/wtmp`
`who` ないし `w` を使用します。

```
root@debian:~# who
root    pts/0        2020-09-14 13:05 (192.168.1.75)
root    pts/1        2020-09-14 13:43 (192.168.1.75)
```

`/var/log/btmp`
`utmpdump` ないし `last -f` を使用します。

```
root@debian:~# utmpdump /var/log/btmp
Utmp dump of /var/log/btmp
[6] [01287] [    ] [dave     ] [ssh:notty   ] [192.168.1.75        ] [192.168.1.75   ] [2019-09-
07T19:33:32,000000+0000]
```

/var/log/faillog
`faillog` を使用します。

```
root@debian:~# faillog -a | less
Login       Failures Maximum Latest                  On

root            0        0   01/01/70 01:00:00 +0100
daemon          0        0   01/01/70 01:00:00 +0100
bin             0        0   01/01/70 01:00:00 +0100
sys             0        0   01/01/70 01:00:00 +0100
sync            0        0   01/01/70 01:00:00 +0100
games           0        0   01/01/70 01:00:00 +0100
man             0        0   01/01/70 01:00:00 +0100
lp              0        0   01/01/70 01:00:00 +0100
mail            0        0   01/01/70 01:00:00 +0100
(...)
```

/var/log/lastlog
`lastlog` を使用します。

```
root@debian:~# lastlog | less
Username         Port     From             Latest
root                                       Never logged in
daemon                                     Never logged in
bin                                        Never logged in
sys                                        Never logged in
(...)
sync                                       Never logged in
avahi                                      Never logged in
colord                                     Never logged in
saned                                      Never logged in
hplip                                      Never logged in
carol            pts/1    192.168.1.75     Sat Sep 14 13:43:06 +0200 2019
dave             pts/3    192.168.1.75     Mon Sep  2 14:22:08 +0200 2019
```

NOTE ログファイルを読み出すためのグラフィカルツールもあります。例: `gnome-logs` や KSystemLog

メッセージをログに記録する仕組み

ログファイルへのメッセージの書き出しは、次のような流れで行われます:

1. アプリケーションやサービスは `/dev/log`（ソケット）に、カーネルは `/dev/kmsg`（メモリバッファ）などの特殊なファイルにメッセージを書き込みます。
2. `rsyslogd` が、ソケットやメモリバッファから情報を取得します。
3. `/etc/rsyslog.conf` や `/etc/ryslog.d/` 内のファイルに記載されたルールに応じて、`rsyslogd` が情報（メッセージ）を指定されたログファイル（通常は `/var/log` に置かれる）に書き込みます。

NOTE ソケットは、プロセス間で情報をやりとりするために使用する特別なファイルです。システム上のすべてのソケットを一覧表示するには、`systemctl list-sockets --all` を使用します。

ファシリティ（分類）、プライオリティ（優先度）、アクション（動作）

`rsyslog` の設定ファイルは `/etc/rsylog.conf` です (ディストリビューションによっては `/etc/rsyslog.d/` にもあります)。通常は `MODULES`、`GLOBAL DIRECTIVES`、`RULES` という3つのセクションに分かれています。Debian GNU/Linux 10（buster）の `rsyslog.conf` ファイルを見てみましょう―`less /etc/rsyslog.conf` を実行します。

`rsyslog` はモジュール構造で、組み込む機能を選択することができます。`MODULES` セクションでは、`rsyslog` に組み込むモジュールを定義します。通常は、ローカルマシンでのロギングのみが有効となっていて、リモートマシンからのログを扱うには設定が必要です。

```
##################
#### MODULES ####
##################

module(load="imuxsock") # provides support for local system logging
module(load="imklog")   # provides kernel logging support
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")
```

`GLOBAL DIRECTIVES` セクションでは、ログやログディレクトリのパーミッションなど、さまざまな設定を行います。

```
###########################
#### GLOBAL DIRECTIVES ####
###########################

#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

#
# Set the default permissions for all log files.
#
$FileOwner root
$FileGroup adm
$FileCreateMode 0640
```

```
$DirCreateMode 0755
$Umask 0022

#
# Where to place spool and state files
#
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf
```

RULES　セクションでは、送られてきたログメッセージに応じて、書き込むログファイルを選択するルールを定義します。ログメッセージに含まれる　ファシリティ（分類）　と　プライオリティ（優先度）に応じて、アクション（動作）　を指定します。アクションには、主にログファイルの絶対パスを指定します。このセクションに記載されたルールを理解するには、ファシリティとプライオリティの概念を理解する必要があります。ファシリティ　には以下のものが定義されていて、ログメッセージを生成するデーモンが適当なものを選択します。たとえば、メールサーバーである `postfix` は、`mail` ファシリティのログメッセージを生成します。（訳注: syslogはさまざまなシステムで使われていて、システムによって使用できるファシリティが微妙に異なります。）

番号	キーワード	説明
0	kern	Linuxカーネルメッセージ
1	user	ユーザーレベルのメッセージ
2	mail	メールシステム
3	daemon	システムデーモン
4	auth、authpriv	セキュリティ上の認証/承認に関するメッセージ
5	syslog	syslogdメッセージ
6	lpr	ラインプリンタサブシステム
7	news	ネットワークニュースサブシステム
8	uucp	UUCP (Unix-to-Unix Copy Protocol) サブシステム
9	cron	時計デーモン（システムによって異なる）
10	auth、authpriv	セキュリティ上の認証/承認に関するメッセージ
11 - 15		（rsyslogでは使えません）
16 - 23	local0 から local7	ローカル使用0-7

また、各メッセージには プライオリティ（優先度）レベルが割り当てられます。

コード	重大度	キーワード	説明
0	Emergency（緊急事態）	emerg、 panic	システムが使用できない
1	Alert（警報）	alert	すぐに行動を起こす必要がある
2	Critical（重大）	crit	危険な状態
3	Error（エラー）	err、 error	エラー状態
4	Warning（警告）	warn、 warning	警告状態
5	Notice（通知）	notice	正常だが注意が必要
6	Informational（情報）	info	情報メッセージ
7	Debug（デバッグ）	debug	デバッグレベルのメッセージ

以下は、Debian GNU/Linux 10（Baster）の `rsyslog.conf` の一部から、いくつかのルールをサンプルとして取り出したものです:

```
###############
#### RULES ####
###############

# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err

#
# Some "catch-all" log files.
#
*.=debug;\
        auth,authpriv.none;\
    news.none;mail.none     -/var/log/debug
*.=info;*.=notice;*.=warn;\
    auth,authpriv.none;\
    cron,daemon.none;\
```

```
    mail,news.none          -/var/log/messages
```

ルールの書式はこうなります: `<facility>.<priority> <action>`

`<facility>.<priority>` をセレクターと呼び、一致するメッセージを絞り込みます。`<facility>` には、カンマで区切って複数のキーワードを指定することができます。`<priority>` には、指定したレベルよりも重大なもの（コードが小さいもの）が一致します。例を見ながら掘り下げていきましょう。

```
auth,authpriv.*                 /var/log/auth.log
```

ファシリティが `auth` ないし `authpriv` のメッセージを、プライオリティに関係なく（`*`）、`/var/log/auth.log` に書き込みます。

```
*.*;auth,authpriv.none          -/var/log/syslog
```

すべてのメッセージ（すべてのプライオリティ（`*`）と、すべてのファシリティ（`*`））を選択しますが、ファシリティが `auth` または `authpriv` のものを除きます。プライオリティに `none` と書くと、直前のファシリティを「除外する」という意味になります。メッセージは `/var/log/syslog` に書き込まれますが、パスの前のマイナス記号（`-`）はディスクへの書き込み回数を抑えることを指示します。セミコロン（`;`）は複数のセレクターを区切るものであり、コンマ（`,`）は1つのセレクターに複数のファシリティ（`auth,authpriv`）を列挙するための区切りです。

```
mail.err                        /var/log/mail.err
```

この例では、ファシリティが `mail` で、プライオリティが `error` 以上（`critical`、`alert`、`emergency`）のメッセージを、`/var/log/mail.err` に書き込みます。

```
*.=debug;\
        auth,authpriv.none;\
    news.none;mail.none     -/var/log/debug
```

この例では、すべてのファシティからプライオリティが `debug`（`=` により指定したレベルのみに限定されます）のメッセージがまず選択されますが、ファシリティ `auth`、`authpriv`、`news`、`mail` であるメッセージが除外されて（プライオリティが `none`）、`/var/log/debug` ファイルに書き込まれます。

`logger` コマンド: ログへの手動送信

シェルスクリプトの中や（ログ設定の）テスト中は、`logger` コマンドが便利です。`logger` コマンドは、ファシリティ `user`、プライオリティ `notice` で、指定されたメッセージをsyslogに送ります（訳注: デフォルトでは、Debian系は `/var/log/syslog`、Redhat系は `/var/log/messages` に書き込まれることになります。ファシリティやプライオリティを指定する時には、`-p` オプションを使います）。

```
carol@debian:~$ logger this comment goes into "/var/log/syslog"
```

`/var/log/syslog` ファイルの最後の行を出力するには、tailコマンドに `-1` オプションを指定します。

```
root@debian:~# tail -1 /var/log/syslog
Sep 17 17:55:33 debian carol: this comment goes into /var/log/syslog
```

`rsyslog` で集中ログサーバーを作る

説明用に、以下の諸元の新しいホストを作成し、これを集中ログサーバーとしてみましょう。つまり、クライアントのsyslogメッセージを、すべてこのサーバーに送信して、集中的に管理できるようにします。

役割	ホスト名	OS	IPアドレス
集中ログサーバー	`suse-server`	openSUSE Leap 15.1	192.168.1.6
クライアント	`debian`	Debian GNU/Linux 10 (buster)	192.168.1.4

サーバーを構成することから始めましょう。まず、`rsyslog` が稼働していることを確認します:

```
root@suse-server:~# systemctl status rsyslog
 rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-09-17 18:45:58 CEST; 7min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 832 (rsyslogd)
    Tasks: 5 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─832 /usr/sbin/rsyslogd -n -iNONE
```

openSUSEには、リモートロギング用の設定ファイル `/etc/rsyslog.d/remote.conf` が付属しています。TCP経由でクライアント（リモートホスト）からのメッセージ受信を有効にしましょう。モジュールをロードし、TCPポート514でサーバーを起動する行のコメントを外します。（訳注: `$ModLoad imtcp.so` 行と `$InputTCPServerRun` 行。ここではTCPでログメッセージを送信する例を挙げている。）。

```
# ######### Receiving Messages from Remote Hosts ##########
# TCP Syslog Server:
# provides TCP syslog reception and GSS-API (if compiled to support it)
$ModLoad imtcp.so  # load module
##$UDPServerAddress 10.10.0.1  # force to listen on this IP only
$InputTCPServerRun 514  # Starts a TCP server on selected port

# UDP Syslog Server:
#$ModLoad imudp.so  # provides UDP syslog reception
##$UDPServerAddress 10.10.0.1  # force to listen on this IP only
#$UDPServerRun 514  # start a UDP syslog server at standard port 514
```

設定を変更したら、rsyslogサービスを再起動して、サーバーがポート514でリッスンしていることを確認します。（訳注: netstatがインストールされていない場合は、`ss -tln` を使用して下さい。トピック109.3で説明しています。）

```
root@suse-server:~# systemctl restart rsyslog
root@suse-server:~# netstat -nltp | grep 514
[sudo] password for root:
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN      2263/rsyslogd
tcp6       0      0 :::514                  :::*                    LISTEN      2263/rsyslogd
```

次に、ファイアウォールのポートを開いて設定を再ロードします。（訳注: LPIC-1範囲外です）。

```
root@suse-server:~# firewall-cmd --permanent --add-port 514/tcp
success
root@suse-server:~# firewall-cmd --reload
success
```

NOTE openSUSE Leap 15.0の登場で、古典的な `SuSEFirewall2` は `firewalld` に置き換えられました。

テンプレートとフィルター条件

ここまでの設定で、リモートから届いたログメッセージが、ローカルマシンからログメッセージと同様にファシリティとプライオリティに応じたファイルに書き込まれるようになりました。それぞれのログメッセージには、そのログを送ったホスト名が含まれていますから、通常はこれだけの設定で十分です。

ここでは、リモートマシンごとにログを保存するディレクトリを分けるように設定してみましょう。`rsyslog` の テンプレート 機能と、フィルター条件 を使います（訳注: いずれもLPIC-1範囲外）。以下の行を、`/etc/rsyslog.conf`（または `/etc/rsyslog.d/remote.conf` など）に追加します。

```
$template RemoteLogs,"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"
if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs
& stop
```

Template

最初に指定する template行では、ログファイル名を動的に生成しますします。この行は以下の項目から成っています。

- templateディレクティブ（`$template`）
- テンプレート名（`RemoteLogs`）
- テンプレート本体（`"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"`）
- オプション（省略）

このテンプレートの名前は `RemoteLogs` であり、その本体は `/var/log/remotehosts` から始まるパス名です。`%` で挟まれた文字列は プロパティ置換子 と呼び、以下のように展開されます。ルールの中でテンプレートを呼び出すと、置換によって生成された文字列が<action>として扱われます。

- `%HOSTNAME%`: 送信元のホスト名
- `%NOW%: 現在時刻（YYYY-MM-DD形式）
- `%syslogseverity-text%`: プライオリティ名

フィルタ条件

続く2行は、フィルタ条件と、一致した時のアクションを定義しています:

- 式ベースのフィルタ（`if $FROMHOST-IP=='192.168.1.4'`）
- アクション（`then ?RemoteLogs`、`& stop`）

1行目は、ログメッセージの送信元リモートホストのIPアドレスが指定のIPアドレスと一致した場合に`RemoteLogs`テンプレートを呼び出します。2行目（`& stop`）は、メッセージを`/var/log/remotehosts/<リモートホスト名>`ディレクトリ内のファイルにのみ書き込み、他のルールを実行しないこと（つまり`/var/log/messages`などには書き込まないこと）を示しています。

NOTE テンプレート、プロパティ、ルールの詳細については、`rsyslog.conf`のマニュアルを参照して下さい。

設定を変更したら`rsyslog`を再起動し、`/var/log`に`<リモートホスト名>`ディレクトリがまだできていないことを確認します。

```
root@suse-server:~# systemctl restart rsyslog
root@suse-server:~# ls /var/log/
acpid             chrony     localmessages   pbl.log          Xorg.0.log
alternatives.log  cups       mail            pk_backend_zypp  Xorg.0.log.old
apparmor          firebird   mail.err        samba            YaST2
audit             firewall   mail.info       snapper.log      zypp
boot.log          firewalld  mail.warn       tallylog         zypper.log
boot.msg          krb5       messages        tuned
boot.omsg         lastlog    mysql           warn
btmp              lightdm    NetworkManager  wtmp
```

以上でサーバーの設定を終えました。次にクライアントを設定します。

ここでも、`rsyslog`がインストールされて稼働していることを確認します。

```
root@debian:~# systemctl status rsyslog
 rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset:
   Active: active (running) since Thu 2019-09-17 18:47:54 CEST; 7min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 351 (rsyslogd)
    Tasks: 4 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─351 /usr/sbin/rsyslogd -n
```

このサンプル環境では、`/etc/hosts`に行`192.168.1.6 suse-server`を追加して、クライアントがサーバーの名前を解決できるようにしています。つまり、クライアントは、名前（`suse-server`）ないしIP

アドレス（192.168.1.6）のいずれかでサーバーを参照できます。

クライアントのdebianには /etc/rsyslog.d/ に remote.conf ファイルが無いので、/etc/rsyslog.conf に設定を追加します。ファイルの末尾に以下の行を書き込みます。

```
*.* @@suse-server:514
```

@@ はTCPでリモートホスト（suse-server）宛てにログメッセージを送信することを示しています。（訳注: UDPを使用する場合には、@ を指定します）。

最後に、クライアントの rsyslog を再起動します。

```
root@debian:~# systemctl restart rsyslog
```

さて、サーバー suse-server に戻って、/var/log ディレクトリに remotehosts ディレクトリができていることを確認しましょう。

```
root@suse-server:~# ls /var/log/remotehosts/debian/
2019-09-17.info.log  2019-09-17.notice.log
```

テンプレートで指定した /var/log/remotehosts ディレクトリには、すでに2つのログが書き込まれています。この実習の最後に、クライアントdebianから 手動で ログを送信し、メッセージがログファイルに追加されることを確認しましょう。suse-server で tail -f 2019-09-17.notice.log を実行します。

```
root@suse-server:~# tail -f /var/log/remotehosts/debian/2019-09-17.notice.log
2019-09-17T20:57:42+02:00 debian dbus[323]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'
2019-09-17T21:01:41+02:00 debian anacron[1766]: Anacron 2.3 started on 2019-09-17
2019-09-17T21:01:41+02:00 debian anacron[1766]: Normal exit (0 jobs run)
```

クライアント debian で logger コマンドを実行します。-t オプションでメッセージの本文を指定します。

```
carol@debian:~$ logger -t DEBIAN-CLIENT Hi from 192.168.1.4
```

サーバー suse-server のログ末尾に、クライアント debian から送信したメッセージが表示されます。

```
root@suse-server:~# tail -f /var/log/remotehosts/debian/2019-09-17.notice.log
2019-09-17T20:57:42+02:00 debian dbus[323]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'
2019-09-17T21:01:41+02:00 debian anacron[1766]: Anacron 2.3 started on 2019-09-17
2019-09-17T21:01:41+02:00 debian anacron[1766]: Normal exit (0 jobs run)
```

```
2019-09-17T21:04:21+02:00 debian DEBIAN-CLIENT: Hi from 192.168.1.4
```

ログローテーションの仕組み

以下の2つの理由から、ログを定期的に切り替える（ローテーションする）ことが必要です。

- 古いログファイルが必要以上にディスク容量を使用することを防ぐ。
- 操作しやすいように、ログを扱いやすい長さに保つ。

ログローテーションを行うユーティリティは `logrotate` といい、ログファイルの名前を変更したり、圧縮したり、システム管理者に電子メールで送信したり、古くなったものを削除するといった処理を行います。ローテーションの際の、古いログファイルに対する命名規則にはさまざまなものが考えられます（たとえば、ファイル名の末尾に日付を追加する）が、整数のサフィックス（接尾辞）を追加するのが最も一般的です。

```
root@debian:~# ls /var/log/messages*
/var/log/messages  /var/log/messages.1  /var/log/messages.2.gz  /var/log/messages.3.gz  /var/log/messages.4.gz
```

この場合に、次回のログローテーションでは次の処理が行われます:

1. `messages.4.gz` が削除されます。
2. `messages.3.gz` は `messages.4.gz` に移動（名前変更）されます。
3. `messages.2.gz` は `messages.3.gz` に移動（名前変更）されます。
4. `messages.1` を圧縮して `messages.2.gz` とします。
5. `messages` は `messages.1` に移動されます。空の `messages` が作成され、新しいログエントリを登録する準備が整います。

後で `logrotate` のディレクティブ（指示子）を説明しますが、3世代目よりも古いログファイルが圧縮されるのに対し、2つの新しいログファイルは圧縮されないことに注意しましょう。またここでは、毎週ログを切り替えるものとして、前の週のログを読むには `messages.1` を参照し、過去4〜5週間のログを保持するものとします。

`logrotate` は、自動化プロセス、すなわちcronによって毎日実行される `/etc/cron.daily/logrotate` スクリプトから実行されます。設定ファイルは `/etc/logrotate.conf` であり、このファイルには目的や簡単な説明を示すコメントと共にいくつものグローバルオプション（ディレクティブ）が含まれています。

```
carol@debian:~$ sudo less /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create
```

```
# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

(...)
```

`/etc/lorotate.d` ディレクトリには、（ログを利用する）パッケージ用の設定ファイルが置かれています。これらのファイルには、それぞれのパッケージに固有の設定として、ローテーションするログファイル名と、その処理方法が指定されています。これらのファイルは `/etc/logrotate.conf` の後に読み込まれるので、ここで指定するオプションはグローバルオプションよりも優先されます。`/etc/logrotate.d/rsyslog` ファイルの一部を見てみましょう:

```
/var/log/messages
{
        rotate 4
        weekly
        missingok
        notifempty
        compress
        delaycompress
        sharedscripts
        postrotate
                invoke-rc.d rsyslog rotate > /dev/null
        endscript
}
```

ディレクティブとその値は、空白ないしはイコール（`=` 省略可能）で区切ります。`postrotate` と `endscript` の間の行は、そのままコマンドとしてローテーション処理の最後に実行されます。行毎に説明しましょう:

`rotate 4`
4週間分のログを保持します。

`weekly`
毎週ログファイルをローテートする。

`missingok`
ログファイルが存在していない場合でもエラーとせず、処理を続行する。

`notifempty`
ログが空の場合は、ローテーションを行わない。

`compress`
`gzip`（デフォルト）でログファイルを圧縮する。

`delaycompress`
直近のログファイルの圧縮を、次のローテーションサイクルに延期する（compressを指定した場合にのみ有効）。プログラムに使用中のログファイルを閉じることを指示できず、現在のログファイルへの書き込みが続く可能性がある場合に指定します。

`sharedscripts`
postrotate を指定した場合に、そのスクリプトの実行方法を指定するもので、指定しているログファイルの数にかかわらず（例えば `/var/log/*` など）、スクリプトを1回だけ実行します。なお、ローテーションを必要とするファイルが1つも無い場合には、スクリプトは実行されません。また、スクリプトがエラーで終了した時は、後続のコマンドは実行されません。

`postrotate`
postrotate スクリプトの開始を示します。

`invoke-rc.d rsyslog rotate > /dev/null`
ログローテーションが完了した後に `/bin/sh` で実行されるコマンド、すなわち postrotate スクリプトの本体です。（訳注: `invode-rc.d` コマンドは、SysV Init形式の起動スクリプト（トピック101.2）を実行するコマンドです。）

`endscript`
postrotate スクリプトの終わりを示します。

NOTE ディレクティブの一覧とその説明は、`logrotate.conf` のマニュアルページを参照してください。

カーネル リング バッファ

カーネルは起動時にいくつものメッセージを生成しますが、その時点ではまだ `rsyslogd` が起動していませんから、カーネルのメッセージを記録するための仕組みが必要です。そのために カーネルリングバッファ（kernel ring buffer) という固定サイズのデータ構造が用意されています。固定サイズなので、いっぱいになると古いメッセージが新しいメッセージで上書きされて消えていきます。

`dmesg` コマンドは、カーネルリングバッファの内容を出力します。バッファのサイズはかなり大きく、出力が長くなることが多いので、`grep` でフィルタリングして使用するのが一般的です。たとえば、USBデバイスに関連するメッセージを検索するには、次のようにします:

```
root@debian:~# dmesg | grep "usb"
[    1.241182] usbcore: registered new interface driver usbfs
[    1.241188] usbcore: registered new interface driver hub
[    1.250968] usbcore: registered new device driver usb
[    1.339754] usb usb1: New USB device found, idVendor=1d6b, idProduct=0001, bcdDevice= 4.19
[    1.339756] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
(...)
```

演習

1. 以下のシナリオで使うコマンドは何ですか？

やりたいこと	コマンド
`/var/log/syslog.7.gz` を読む	
`/var/log/syslog` を読む	
`/var/log/syslog` から、単語 `renewal` をフィルタする	
`/var/log/faillog` を読む	
`/var/log/syslog` を実時間で読む	

2. 適切なログメッセージになるように、以下の要素を並び替えて下さい:
   - `debian-server`
   - `sshd`
   - `[515]:`
   - `Sep 13 21:47:56`
   - `Server listening on 0.0.0.0 port 22`

   正しい順番:

3. それぞれの処理を実現するために、`/etc/rsyslog.conf` に指定するルールはどうなりますか？
   - ファシリティが `mail` で、プライオリティが `crit` 以上のメッセージを、`/var/log/mail.crit` に送る。
   - ファシリティが `mail` で、プライオリティが `alert` と `emergency` メッセージを、`/var/log/mail.urgent` に送る。
   - ファシリティが `cron` および `ntp` のものを除き、プライオリティに関係なく、すべてのメッセージを `/var/log/allmessages` に送る。
   - 必要なすべての設定が行われているものとして、TCPのデフォルトポートを使用して、ファシリティが `mail` であるすべてのメッセージを、IPアドレス `192.168.1.88` のリモートホストに送信する。
   - ファシリティに関係なく、プライオリティが `warning` であるメッセージを、`/var/log/warnings` に、ディスクへの書き込み回数を抑えながら書き込む。

4. 以下に示す `/etc/logrotate.d/samba` の一節をみて、それぞれのオプションを説明してください。

```
carol@debian:~$ sudo head -n 11 /etc/logrotate.d/samba
/var/log/samba/log.smbd {
        weekly
        missingok
        rotate 7
        postrotate
                [ ! -f /var/run/samba/smbd.pid ] || /etc/init.d/smbd reload > /dev/null
        endscript
        compress
        delaycompress
        notifempty
}
```

オプション	意味
`weekly`	
`missingok`	
`rotate 7`	
`postrotate`	
`endscript`	
`compress`	
`delaycompress`	
`notifyempty`	

発展演習

1. “テンプレートとフィルター条件” 節では、フィルター条件として 式ベースのフィルター を説明しました。`rsyslogd` では プロパティベースのフィルター を使うこともできます。表中の 式ベースのフィルター を、プロパティベースのフィルター に変換してください。

式ベースのフィルター	プロパティベースのフィルター
`if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs`	

2. `omusrmsg` は `rsyslog` の 組み込み モジュールで、ユーザーがログイン中の端末にログメッセージを送信します。すべてのファシリティの emergency メッセージを、`root` と一般ユーザー `carol` に送信するルールはどうなりますか？

まとめ

このレッスンでは、以下の事柄を学びました:

- ロギングはシステム管理にとって非常に重要です。
- `rsyslogd` は、ログを整理整頓するためのユーティリティです。
- サービスによっては、（`rsyslog` を使わずに）独自のログ管理を行うものがあります。（Apache HTTPD Web Serverなど）
- 大まかに言えば、ログは、システムログとサービス（アプリケーション）ログに分類できます。
- ログの読み取りに便利なユーティリティがあります: `less`、`more`、`zless`、`zmore`、`grep`、`head`、`tail`。
- ログファイルのほとんどはプレーンテキストファイルですが、少数ながらバイナリログも存在します。
- `rsyslogd` は、特別なファイル（ソケットやメモリバッファ）から、ログを取り出して処理します。
- `rsyslogd` は、`/etc/rsyslog.conf` または `/etc/rsyslog.d/*` のルールを使ってログを分類します。
- ユーザーは、`logger` ユーティリティでログに独自のメッセージを書き込めます。
- `rsyslog` を使用すると、ローカルネットワーク全体のすべてのログを集中ログサーバーに保持できます。
- ログファイル名を動的に指定するには、テンプレート機能を使用します。
- ログローテーションの目的は2つ: 古いログファイルが必要以上にディスク容量を使用することを防ぐことと、操作しやすいようにログを扱いやすい長さに保つことです。

演習の解答

1. 以下のシナリオで使うコマンドは何ですか？

やりたいこと	コマンド
`/var/log/syslog.7.gz` を読む	`zmore` または `zless`
`/var/log/syslog` を読む	`more` または `less`
`/var/log/syslog` から、単語 `renewal` をフィルタする	`grep`
`/var/log/faillog` を読む	`faillog -a`
`/var/log/syslog` を実時間で読む	`tail -f`

2. 適切なログメッセージになるように、以下の要素を並び替えてください。
   - `debian-server`
   - `sshd`
   - `[515]:`
   - `Sep 13 21:47:56`
   - `Server listening on 0.0.0.0 port 22`

   正しい順序は次のとおりです。

```
Sep 13 21:47:56 debian-server sshd[515]: Server listening on 0.0.0.0 port 22
```

3. それぞれの処理を実現するために、`/etc/rsyslog.conf` に指定するルールはどうなりますか？
   - ファシリティが `mail` で、プライオリティが `crit` 以上のメッセージを、`/var/log/mail.crit` に送る。

```
mail.crit               /var/log/mail.crit
```

   - ファシリティが `mail` で、プライオリティが `alert` と `emergency` のメッセージを、`/var/log/mail.urgent` に送る。

```
mail.alert                      /var/log/mail.urgent
```

   - ファシリティが `cron` および `ntp` のものを除き、プライオリティに関係なく、すべてのメッセージを `/var/log/allmessages` に送る。

```
*.*;cron.none;ntp.none                /var/log/allmessages
```

   - 必要なすべての設定が行われているものとして、TCPのデフォルトポートを使用して、ファシリテ

ィが `mail` であるすべてのメッセージを、IPアドレス `192.168.1.88` のリモートホストに送信する。

```
mail.* @@192.168.1.88:514
```

- ファシリティに関係なく、プライオリティが `warning` であるメッセージを、`/var/log/warnings` に、ディスクへの書き込み回数を抑えながら書き込む。

```
*.=warning                      -/var/log/warnings
```

4. 以下に示す `/etc/logrotate.d/samba` の一節をみて、それぞれのオプションを説明してください。

```
carol@debian:~$ sudo head -n 11 /etc/logrotate.d/samba
/var/log/samba/log.smbd {
        weekly
        missingok
        rotate 7
        postrotate
                [ ! -f /var/run/samba/smbd.pid ] || /etc/init.d/smbd reload > /dev/null
        endscript
        compress
        delaycompress
        notifempty
}
```

オプション	意味
`weekly`	ログファイルを毎週ローテーションする。
`missingok`	ログファイルが存在していない場合でもエラーとせず、処理を続行する。
`rotate 7`	7週間分の古いログを保持する。
`postrotate`	ログをローテーションした後に、次の行からのスクリプトを実行する。
`endscript`	postrotate スクリプトの終わりを示す。
`compress`	`gzip` でログを圧縮する。
`delaycompress`	`compress` と組み合わせて、圧縮を次のローテーションサイクルに延期する。
`notifyempty`	ログが空の場合は、ローテーションを行わない。

発展演習の解答

1. “テンプレートとフィルター条件” 節では、フィルター条件として 式ベースのフィルター を説明しました。`rsyslogd` では プロパティベースのフィルター を使うこともできます。表中の 式ベースのフィルター を、プロパティベースのフィルター に変換してください。

式ベースのフィルター	プロパティベースのフィルター
`if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs`	`:fromhost-ip, isequal, "192.168.1.4" ?RemoteLogs`

2. `omusrmsg` は、ユーザー端末にログメッセージを送信する、`rsyslog` の 組み込み モジュールです。すべてのファシリティの emergency メッセージを、`root` と一般ユーザー `carol` に送信するルールはどうなりますか？

```
*.emerg                         :omusrmsg:root,carol
```

108.2 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	108 基本的なシステムサービス
Objective:	108.2 システムロギング
Lesson:	2 of 2

はじめに

すべての主要なディストリビューションが `systemd` を採用したことで、ジャーナルデーモン（`systemd-journald`）が標準のロギングサービスになりました。このレッスンでは、その動作と、さまざまな操作方法を説明します。例えば、ジャーナルへの問い合わせ、さまざまな条件によるフィルタリング、ストレージとサイズの設定、古いデータの削除、レスキューシステムやファイルシステムのコピーからのデータの取得などです。そして最後に、重要な `rsyslogd` との相互作用を学習します。

`systemd` の基本

Fedoraで初めて導入された `systemd` は、ほとんどの主要なLinuxディストリビューションにおいて、SysV Initに代わる 事実上の システムとサービスのマネージャとなっています。長所を以下に示します:

- 容易な設定: SysV Initのスクリプトとはまったく異なるユニットファイルを使います。
- 広範囲な管理: デーモンやプロセスだけではなく、デバイス、ソケット、マウントポイントなども管理します。
- SysV InitおよびUpstartとの下位互換性を持っています。
- 起動時の並列ロード: Sysv Initがサービスを1つずつ実行するのに対して、systemdは複数のサービスを並列に実行します。
- journal と呼ばれるロギングサービスを備えており、以下の長所を備えています。
  - すべてのログを1か所で集中管理します。
  - ログローテーションは必要ありません。

- ログを無効にしたり、RAMに置いたり、あるいはディスクに永続化することができます。

ユニットとターゲット

`systemd` は ユニット を単位として動作します。ユニットとは、`systemd` が管理できるあらゆるリソースです（サービス、ネットワーク、Bluetoothなど）。それぞれのユニットは、ユニットファイルで定義します。ユニットファイルは `/lib/systemd/system` にあるプレーンテキストファイルで、管理するリソース用の設定情報を（セクション と ディレクティブ の形式で）含んでいます。ユニットには多くの種類があります: `service`、`mount`、`automount`、`swap`、`timer`、`device`、`socket`、`path`、`timer`、`snapshot`、`slice`、`scope`、`target`。ユニットファイルの名前は、`<リソース名>.<ユニット種別>` となります（例えば `reboot.service`）。

target は、従来のSysV Initにおけるランレベルに似た、特殊なユニット種類です。つまり、target unit は、複数のリソース（ユニット）をまとめて、所定のシステム状態を表します（たとえば `graphical.target` は、SysV Initにおける `ランレベル 5` に相当します）。システムの現在のターゲットを確認するには、`systemctl get-default` コマンドを使用します。

```
carol@debian:~$ systemctl get-default
graphical.target
```

ターゲットは親子（依存）関係を持てますから、あるターゲットから別のターゲットを起動できますが、SysV Initに親子関係の概念はありません。

NOTE `systemd` ユニットがどのように動作するかの説明は、このレッスンの範囲外です。（訳注: 101試験のトピック101.2で取り上げています）

システムジャーナル: `systemd-journald`

`systemd-journald` は、さまざまなソースからログ情報を受け取るシステムサービスです: カーネルメッセージ、（syslog形式の）シンプルなシステムメッセージ、サービスの標準出力と標準エラー出力、カーネル監査サブシステムからの監査レコードなど。（詳細は、`systemd-journald` のマニュアルページを参照してください）。構造化され索引付けされたジャーナルを作成および保守しています。

その設定ファイルは `/etc/systemd/journald.conf` で、他のサービスと同様に `systemctl` コマンドを使用して start、restart、stop したり、status で稼働状態を確認できます。

```
root@debian:~# systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset: enabled)
   Active: active (running) since Sat 2019-10-12 13:43:06 CEST; 5min ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 178 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/systemd-journald.service
           └─178 /lib/systemd/systemd-journald
(...)
```

`journal.conf.d/*.conf` にパッケージに固有の設定ファイルを置くこともできます（詳細は、`journald.conf` のマニュアルページを参照してください）。

`systemd-journald` を有効にすると、ジャーナルがディスク（永続的）ないしRAMベースのファイルシステム（揮発性）に保存されます。ジャーナルはプレーンテキストファイルではなく、バイナリファイルです。したがって、`less` や `more` などのテキスト表示ツールでその内容を読み取ることはできません。代わりに `journalctl` コマンドを使用します。

ジャーナル内容の調査

`journalctl` は、`systemd` ジャーナルを照会するためのユーティリティです。rootであるか、`sudo` を使用して呼び出す必要があります（訳注: 多くのディストリビューションでは、一般ユーザー権限でジャーナルを参照することができます）。オプションなしでクエリを実行すると、ジャーナル全体が時系列で表示されます（最も古いエントリが先頭に表示されます）。

```
root@debian:~# journalctl
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:19:46 CEST. --
Oct 12 13:43:06 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (...)
Oct 12 13:43:06 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=b6be6117-5226-
4a8a-bade-2db35ccf4cf4 ro qu
(...)
```

オプションを指定して、より具体的な問い合わせを行います。

`-r`
逆順で（新しい順に）ジャーナルメッセージを表示します。

```
root@debian:~# journalctl -r
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:30:30 CEST. --
Oct 12 14:30:30 debian sudo[1356]: pam_unix(sudo:session): session opened for user root by carol(uid=0)
Oct 12 14:30:30 debian sudo[1356]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -r
Oct 12 14:19:53 debian sudo[1348]: pam_unix(sudo:session): session closed for user root
(...)
```

`-f`
直近のジャーナルメッセージを出力し、内容が追加されるとそのエントリを出力し続けます—`tail -f` と同様です。

```
root@debian:~# journalctl -f
-- Logs begin at Sat 2019-10-12 13:43:06 CEST. --
(...)
Oct 12 14:44:42 debian sudo[1356]: pam_unix(sudo:session): session closed for user root
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root by carol(uid=0)

(...)
```

q`-e`
ジャーナルの末尾部分を、画面いっぱいに（ページャーで）表示します。

```
root@debian:~# journalctl -e
(...)q
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root by carol(uid=0)
Oct 12 14:45:57 debian sudo[1375]: pam_unix(sudo:session): session closed for user root
Oct 12 14:48:39 debian sudo[1378]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -e
Oct 12 14:48:39 debian sudo[1378]: pam_unix(sudo:session): session opened for user root by carol(uid=0)
```

`-n <value>`、`--lines=<value>`
ジャーナル末尾の value 行を出力します（`<value>` を指定しない場合のデフォルトは10）。

```
root@debian:~# journalctl -n 5
(...)
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root by carol(uid=0)
Oct 12 14:45:57 debian sudo[1375]: pam_unix(sudo:session): session closed for user root
Oct 12 14:48:39 debian sudo[1378]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -e
Oct 12 14:48:39 debian sudo[1378]: pam_unix(sudo:session): session opened for user root by carol(uid=0)
```

`-k`、`--dmesg`
`dmesg` コマンドと同様。

```
root@debian:~# journalctl -k
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:53:20 CEST. --
Oct 12 13:43:06 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (gcc version
6.3.0 20170516 (Debian 6.3.0-18
Oct 12 13:43:06 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=b6be6117-
5226-4a8a-bade-2db35ccf4cf4 ro qu
Oct 12 13:43:06 debian kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
Oct 12 13:43:06 debian kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
(...)
```

ジャーナルのナビゲーションと検索

次の方法でジャーナルの出力位置を移動できます。（訳注:　主要なコマンドは　`less`　と同じです。ただし、行の折り返しは行われません）。

- PageUp、PageDown、矢印キーを使用して、上下左右に移動します。（横スクロールもできます）。
- `>`で、出力の最後に移動します。

- `<`で、出力の先頭に移動します。

現在の位置から文字列を順方向・逆方向で検索できます。

- 前方検索: `/`を押して検索する文字列を入力し、Enterキーを押します。
- 後方検索: `?`を押して検索する文字列を入力し、Enterキーを押します。

前後の一致文字列に移動するには、`N`で次の一致へ、`Shift`☒+☒`N`で前の一致へ、それぞれ移動します。

ジャーナルデータのフィルタリング

ジャーナルでは、さまざまな条件でログデータをフィルタリングできます。

ブート番号

`--list-boots`

保存されているブート履歴を一覧表示します。出力は3つの列で構成され、最初のものはブート番号を示します（`0` が現在の、`-1` が直前の、`-2` がその前のブートを示します）。2番列目はブートIDで、3列目はタイムスタンプです。

```
root@debian:~# journalctl --list-boots
 0 83df3e8653474ea5aed19b41cdb45b78 Sat 2019-10-12 18:55:41 CEST—Sat 2019-10-12 19:02:24 CEST
```

`-b`、`--boot`

現在のブートにおけるすべてのジャーナルを表示します。以前のブートのログメッセージを表示するには、前項で説明したブート番号を指定します。たとえば、前回のブートメッセージを出力するには、`journalctl -b -1` と入力します。ただし、以前のログから情報を得るには、ジャーナルの永続化を有効にする必要があります（次のセクションで学習します）。

```
root@debian:~# journalctl -b -1
Specifying boot ID has no effect, no persistent journal was found
```

プライオリティ（優先度）

`-p`

`-p` オプションで、プライオリティ（優先度）でフィルタリングすることもできます。

```
root@debian:~# journalctl -b -0 -p err
-- No entries --
```

現在のブートでは、プライオリティが `error`（以上）のメッセージがないことが示されました。なお、現在のブートを参照する場合には `-b -0` を省略できます。

> **NOTE** `syslog` のプライオリティについては、前のレッスンを参照してください。

期間指定

`journalctl` に、指定の期間（日時）内にログに記録されたジャーナルのみを出力させるには、`--since` と `--until` オプションを使用します。日付の指定には、`YYYY-MM-DD HH:MM:SS` 形式を

使います。時刻を省略すると0時0分0秒と見なされ、日付を省略すると今日と見なされます。たとえば、午後7時から午後7時1分までにログに記録されたメッセージを表示するには、次のように指定します。

```
root@debian:~# journalctl --since "19:00:00" --until "19:01:00"
-- Logs begin at Sat 2019-10-12 18:55:41 CEST, end at Sat 2019-10-12 20:10:50 CEST. --
Oct 12 19:00:14 debian systemd[1]: Started Run anacron jobs.
Oct 12 19:00:14 debian anacron[1057]: Anacron 2.3 started on 2019-10-12
Oct 12 19:00:14 debian anacron[1057]: Normal exit (0 jobs run)
Oct 12 19:00:14 debian systemd[1]: anacron.timer: Adding 2min 47.988096s random time.
```

ちょっと前を意味する時刻指定も使用できます: たとえば、2分前からログに記録されたメッセージを表示するには、`journalctl --since "2 minutes ago"`、ないしは `+` と `-` を使用して、`--since "-2 minutes"` と指定することもできます。

分ではなく、キーワードで時刻を指定することもできます。

`yesterday`
前日の0時0分0秒

`today`
当日の0時0分0秒

`tomorrow`
翌日の0時0分0秒

`now`
現在時刻

今日の0時0分0秒から21:00までのすべてのメッセージを見てみましょう。

```
root@debian:~# journalctl --since "today" --until "21:00:00"
-- Logs begin at Sat 2019-10-12 20:45:29 CEST, end at Sat 2019-10-12 21:06:15 CEST. --
Oct 12 20:45:29 debian sudo[1416]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/systemctl r
Oct 12 20:45:29 debian sudo[1416]: pam_unix(sudo:session): session opened for user root by carol(uid=0)
Oct 12 20:45:29 debian systemd[1]: Stopped Flush Journal to Persistent Storage.
(...)
```

NOTE 時刻指定の詳細な構文については、`systemd.time` のマニュアルページを参照してください。

プログラム

ある実行可能ファイルに関連するジャーナルのみを表示するには、次の構文を使用します: `journalctl /path/to/executable`

```
root@debian:~# journalctl /usr/sbin/sshd
-- Logs begin at Sat 2019-10-12 20:45:29 CEST, end at Sat 2019-10-12 21:54:49 CEST. --
```

```
Oct 12 21:16:28 debian sshd[1569]: Accepted password for carol from 192.168.1.65 port 34050 ssh2
Oct 12 21:16:28 debian sshd[1569]: pam_unix(sshd:session): session opened for user carol by (uid=0)
Oct 12 21:16:54 debian sshd[1590]: Accepted password for carol from 192.168.1.65 port 34052 ssh2
Oct 12 21:16:54 debian sshd[1590]: pam_unix(sshd:session): session opened for user carol by (uid=0)
```

ユニット

ユニットとは、`systemd` が処理する1つのリソースでした。ユニットでジャーナルをフィルタリングすることもできます。

`-u`

指定したユニットに関するメッセージのみを表示します。

```
root@debian:~# journalctl -u ssh.service
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 12:22:59 CEST. --
Oct 13 10:51:00 debian systemd[1]: Starting OpenBSD Secure Shell server...
Oct 13 10:51:00 debian sshd[409]: Server listening on 0.0.0.0 port 22.
Oct 13 10:51:00 debian sshd[409]: Server listening on :: port 22.
(...)
```

NOTE

ロードされて有効なユニットをすべて出力するには、`systemctl list-units` を使用します。インストールされているすべてのユニットファイルを表示するには、`systemctl list-unit-files` を使用します。

フィールド

指定した フィールド の値でジャーナルをフィルタリングするには、次の構文のいずれかを使用します:

- `<フィールド名>=<値>`
- `_<フィールド名>=<値>_`
- `__<フィールド名>=<値>`

（訳注: フィールド名の前にある `_`（アンダースコア）の数によって、アプリケーションが出力した値（アンダースコア無し）、ジャーナルに自動的に付加された値（アンダースコア1個）、ジャーナルエントリの位置を示す値（アンダースコア2個）に分類することが出来ます。）

`PRIORITY=`

`syslog` プライオリティを、プライオリティ番号で指定します:

```
root@debian:~# journalctl PRIORITY=3
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:30:50 CEST. --
Oct 13 10:51:00 debian avahi-daemon[314]: chroot.c: open() failed: No such file or directory
```

`journalctl -perr` コマンドと同じ結果になることに注意してください。

`SYSLOG_FACILITY=`

`syslog` ファシリティを、ファシリティ番号で指定します。たとえば、次の例はファシリティ番号が1（rsyslogではuser）であるメッセージを表示します。

```
root@debian:~# journalctl SYSLOG_FACILITY=1
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:42:52 CEST. --
Oct 13 10:50:59 debian mtp-probe[227]: checking bus 1, device 2:
"/sys/devices/pci0000:00/0000:00:06.0/usb1/1-1"
Oct 13 10:50:59 debian mtp-probe[227]: bus: 1, device: 2 was not an MTP device
Oct 13 10:50:59 debian mtp-probe[238]: checking bus 1, device 2:
"/sys/devices/pci0000:00/0000:00:06.0/usb1/1-1"
Oct 13 10:50:59 debian mtp-probe[238]: bus: 1, device: 2 was not an MTP device
```

_PID=
指定したPIDを持つプロセスによって生成されたメッセージを表示します。次の例は、`systemd`（訳注: PIDは必ず1です）によって生成されたすべてのメッセージを表示します。

```
root@debian:~# journalctl _PID=1
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:50:15 CEST. --
Oct 13 10:50:59 debian systemd[1]: Mounted Debug File System.
Oct 13 10:50:59 debian systemd[1]: Mounted POSIX Message Queue File System.
Oct 13 10:50:59 debian systemd[1]: Mounted Huge Pages File System.
Oct 13 10:50:59 debian systemd[1]: Started Remount Root and Kernel File Systems.
Oct 13 10:50:59 debian systemd[1]: Starting Flush Journal to Persistent Storage...
(...)
```

_BOOT_ID
ブートIDを指定し、そのブート時のメッセージを選択します。例: `journalctl _BOOT_ID=83df3e8653474ea5aed19b41cdb45b78`

_TRANSPORT
指定したトランスポートから受信したメッセージを表示します。次の値を指定できます: `audit`（カーネル監査サブシステム）、`driver`（カーネル内部）、`syslog`（syslogソケット）、`journal`（ネイティブジャーナルプロトコル）、`stdout`（サービスの標準出力ないし標準エラー出力）、`kernel`（カーネルリングバッファ—`dmesg`、`journalctl -k` ないし `journalctl --dmesg` と同じ）

```
root@debian:~# journalctl _TRANSPORT=journal
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:46:36 CEST. --
Oct 13 20:19:58 debian systemd[1]: Started Create list of required static device nodes for the current
kernel.
Oct 13 20:19:58 debian systemd[1]: Starting Create Static Device Nodes in /dev...
Oct 13 20:19:58 debian systemd[1]: Started Create Static Device Nodes in /dev.
Oct 13 20:19:58 debian systemd[1]: Starting udev Kernel Device Manager...
(...)
```

フィールドの組み合わせ

フィールドによるフィルタリングは、組み合わせて使用することができます。デフォルトではすべての条件に一致するメッセージのみが表示されます（論理 AND）:

```
root@debian:~# journalctl PRIORITY=3 SYSLOG_FACILITY=0
```

```
-- No entries --
root@debian:~# journalctl PRIORITY=4 SYSLOG_FACILITY=0
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:21:55 CEST. --
Oct 13 20:19:58 debian kernel: acpi PNP0A03:00: fail to add MMCONFIG information, can't access extended PCI
configuration (...)
```

論理 OR で2つの式を組み合わせるには、`+` を使います:

```
root@debian:~# journalctl PRIORITY=3 + SYSLOG_FACILITY=0
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:24:02 CEST. --
Oct 13 20:19:58 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (...9
Oct 13 20:19:58 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID= (...)
(...)
```

また、同じフィールドに複数の値を指定すると、いずれかの値に一致するすべてのエントリが表示されます:

```
root@debian:~# journalctl PRIORITY=1
-- Logs begin at Sun 2019-10-13 17:16:24 CEST, end at Sun 2019-10-13 17:30:14 CEST. --
-- No entries --
root@debian:~# journalctl PRIORITY=1 PRIORITY=3
-- Logs begin at Sun 2019-10-13 17:16:24 CEST, end at Sun 2019-10-13 17:32:12 CEST. --
Oct 13 17:16:27 debian connmand[459]: __connman_inet_get_pnp_nameservers: Cannot read /pro
Oct 13 17:16:27 debian connmand[459]: The name net.connman.vpn was not provided by any .se
```

NOTE ジャーナルファイルには、記録される情報（フィールド）の違いによって5種類があります:。詳細は `systemd.journal-fields(7)` の `man` ページを参照してください。

システムジャーナルへの手動入力: `systemd-cat`

前のレッスンでは、`logger` コマンドでコマンドラインからシステムログにメッセージを送信することができることを学びました（訳注: `logger` コマンドはジャーナルにもメッセージを送信します）。`systemd-cat` コマンドも同様にジャーナルにメッセージを送信しますが、より柔軟です。シェルの機能とあわせて、標準入力（stdin）、標準出力（stdout）、標準エラー出力（stderr）をジャーナルに送信できます。

パラメータを付けずに呼び出した場合は、stdin から読み取ったメッセージをジャーナルに送信します。入力を終わるには、`Ctrl`+`D`（EOF）を入力します。

```
carol@debian:~$ systemd-cat
This line goes into the journal.
^D
```

コマンドの出力をパイプで繋げば、コマンドの標準出力がジャーナルに送信されます:

```
carol@debian:~$ echo "And so does this line." | systemd-cat
```

systemd-catコマンドの後に別のコマンドを続けると、そのコマンドの出力が（stderr　込みで）ジャーナルに送信されます:

```
carol@debian:~$ systemd-cat echo "And so does this line too."
```

`-p` オプションでプライオリティを指定することもできます:

```
carol@debian:~$ systemd-cat -p emerg echo "This is not a real emergency."
```

その他のオプションについては、`systemd-cat` のマニュアルページを参照してください。

ジャーナルの最後の4行を表示してみましょう：

```
carol@debian:~$ journalctl -n 4
(...)
-- Logs begin at Sun 2019-10-20 13:43:54 CEST. --
Nov 13 23:14:39 debian cat[1997]: This line goes into the journal.
Nov 13 23:19:16 debian cat[2027]: And so does this line.
Nov 13 23:23:21 debian echo[2030]: And so does this line too.
Nov 13 23:26:48 debian echo[2034]: This is not a real emergency.
```

NOTE　プライオリティが　emergency　のジャーナルエントリは、ほとんどのシステムで太字の赤で表示されます。

永続的なジャーナルストレージ

ジャーナルの保管方法には、次の3つのオプションがあります。

- ジャーナリングを行わない（ただし、コンソールへの送信など、他の機能へのリダイレクトは行われる）。
- ジャーナルをメモリに保持し、システムを再起動するたびにログを削除する。この場合、`/run/log/journal`　ディレクトリが作成されて、ジャーナルの保管に使われます。（訳注:　`/run` 以下のディレクトリは、tmpfs（メモリファイルシステム）であることが一般的です。）
- ログをディスクに書き込んで永続化する。この場合、ジャーナルは　`/var/log/journal`　ディレクトリのファイルに記録されます。

デフォルトの動作は次のとおりです:　`/var/log/journal/`　が存在しない場合、ジャーナルは再起動時に失われるディレクトリ　`/run/log/journal/`　内の、マシンID（`/etc/machine-id`　に保存された小文字の16進数32桁）を名前とするディレクトリに置かれます。

```
carol@debian:~$ ls /run/log/journal/8821e1fdf176445697223244d1dfbd73/
system.journal
```

lessコマンドで読み取ろうとすると警告が表示されるので、`journalctl` コマンドで表示します:

```
root@debian:~# less /run/log/journal/9a32ba45ce44423a97d6397918de1fa5/system.journal
"/run/log/journal/9a32ba45ce44423a97d6397918de1fa5/system.journal" may be a binary file.  See it anyway?
root@debian:~# journalctl
-- Logs begin at Sat 2019-10-05 21:26:38 CEST, end at Sat 2019-10-05 21:31:27 CEST. --
(...)
Oct 05 21:26:44 debian systemd-journald[1712]: Runtime journal
(/run/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 4.9M, max 39.5M, 34.6M free.
Oct 05 21:26:44 debian systemd[1]: Started Journal Service.
(...)
```

/var/log/journal/ が存在する場合は、その中の永続的なファイルにジャーナルが保存されます。このディレクトリを削除すると、systemd-journald はそのディレクトリ再作成するのではなく、/run/log/journal にジャーナルを書き込みます。/var/log/journal/ を作成しなおして systed を再起動すると、すぐに永続的なファイルへの保存が再開します。

```
root@debian:~# mkdir /var/log/journal/
root@debian:~# systemctl restart systemd-journald
root@debian:~# journalctl
(...)
Oct 05 21:33:49 debian systemd-journald[1712]: Received SIGTERM from PID 1 (systemd).
Oct 05 21:33:49 debian systemd[1]: Stopped Journal Service.
Oct 05 21:33:49 debian systemd[1]: Starting Journal Service...
Oct 05 21:33:49 debian systemd-journald[1768]: Journal started
Oct 05 21:33:49 debian systemd-journald[1768]: System journal
(/var/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 8.0M, max 1.1G, 1.1G free.
Oct 05 21:33:49 debian systemd[1]: Started Journal Service.
Oct 05 21:33:49 debian systemd[1]: Starting Flush Journal to Persistent Storage...
(...)
```

NOTE ログインしているユーザー用のジャーナルファイルも、デフォルトでは /var/log/journal/ に置かれるので、そこには system.journal だけでなく user-1000.journal などのファイルもあります。

ジャーナルデーモンがどこにジャーナルを置くかを指定するには、設定ファイル /etc/systemd/journald.conf を調整します。オプション Storage= に次のいずれかの値を指定します。

Storage=volatile
ジャーナルは /run/log/journal/ に保存されます。ディレクトリが存在しない場合は作成されます。

Storage=persistent
ジャーナルはディスク（/var/log/journal/）に保存されます。起動の初期段階など、ディスクに書き込めない場合はメモリ（/run/log/journal/）にフォールバックします。必要に応じて、両方のディレクトリが作成されます。

Storage=auto
auto は persistent と同じですが、ディレクトリ /var/log/journal が自動的に作成されることがありません。これがデフォルトです。

Storage=none
: すべてのジャーナルは破棄されます。ただし、コンソールやカーネルログバッファ、syslogソケットなど、他のターゲットへの転送は行われます。

たとえば、`systemd-journald` が自動的に `/var/log/journal/` を作成し、ジャーナルをディスクに保存したい場合は、`/etc/systemd/journald.conf` を編集して `Storage=persistent` を指定します。その後、`sudo systemctl restart systemd-journald` でデーモンを再起動します。正常に再起動したことを確認するために、デーモンのステータスを確認しましょう。

```
root@debian:~# systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset: enabled)
   Active: active (running) since Wed 2019-10-09 10:03:40 CEST; 2s ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 1872 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1 (limit: 3558)
   Memory: 1.1M
   CGroup: /system.slice/systemd-journald.service
           └─1872 /lib/systemd/systemd-journald

Oct 09 10:03:40 debian10 systemd-journald[1872]: Journal started
Oct 09 10:03:40 debian10 systemd-journald[1872]: System journal
(/var/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 8.0M, max 1.2G, 1.2G free.
```

NOTE

`/var/log/journal/<machine-id>/` ないし `/run/log/journal/<machine-id>/` に置かれるジャーナルファイル名には、サフィックスとして `.journal` が付けられます（例えば `system.journal`）。それが破損している場合、あるいはデーモンが異常終了した場合は、ファイル名の末尾に `~` がさらに追加されて（例えば `system.journal~`）、デーモンは新しいクリーンなファイルへの書き込みを開始します。

古いジャーナルデータの削除：ジャーナルサイズ

ジャーナルデータは、`/run/log/journal` または `/var/log/journal` に置かれる ジャーナルファイルに保存されます。ファイル名のサフィックスは `.journal` または `.journal~` です。ジャーナルファイル全体が占有しているディスク容量を確認するには、`--disk-usage` オプションを指定します。

```
root@debian:~# journalctl --disk-usage
Archived and active journals take up 24.0M in the filesystem.
```

`systemd` のデフォルトでは、ジャーナル全体が使用する容量が、それが保存されているファイルシステムのサイズの最大10%に制限されます。たとえば、1GBのファイルシステムでは、100MBを超えません。上限に達すると古いログが消されて、この値の近くに留まります。

保存するジャーナルファイルのサイズ制限は、`/etc/systemd/journald.conf` の設定オプションで調整できます。オプションは2種類あり、永続的なファイルシステム（`/var/log/journal`）に対するものは先頭に `System` が、メモリファイルシステム（`/run/log/journal`）に対するものは先頭に `Runtime` という単語が付きます。順に見ていきましょう。

SystemMaxUse=、RuntimeMaxUse=
ジャーナルが占めるディスク容量を制御します。デフォルトではファイルシステム容量の10%ですが、4GiBまでの範囲で変更できます（たとえば SystemMaxUse=500M）。

SystemKeepFree=、RuntimeKeepFree=
ユーザー用に空けておくディスク容量を制御します。デフォルトではファイルシステム容量の15%ですが、4GiBまでの範囲で変更できます（たとえば SystemKeepFree=500M）。

*MaxUse と *KeepFree の両方を指定した場合は、両方を満たすようにジャーナルサイズが制限されます。なお、アクティブなジャーナルファイルは削除されないことに注意してください。

SystemMaxFileSize=、RuntimeMaxFileSize=
個々のジャーナルファイルの最大サイズを制御します。デフォルトは *MaxUse の1/8です。制限値を超えた時にジャーナルのローテーションが同期的に行われます。値はバイト単位、もしくは、K（Kibibytes）、M（Mebibytes）、G（Gibibytes）、T（Tebibytes）、P（Pebibytes）、E（Exbibytes）単位で指定できます。

SystemMaxFiles=、RuntimeMaxFiles=
保存するアーカイブの最大数を指定します（アクティブなジャーナルファイルは含みません）。デフォルトは100です。

サイズに基づくログメッセージの削除およびローテーションとは別に、MaxRetentionSec= と MaxFileSec= の2つのオプションを使用して、経過時間に基づくローテーションを指定することもできます。詳細は、journald.conf のマニュアルページを参照してください。

NOTE systemd-journald のデフォルト動作を変更するために /etc/systemd/journald.conf を変更した場合は、変更を有効にするためにデーモンを再起動する必要があります。

ジャーナルのお掃除

アクティブではない（アーカイブ）ジャーナルファイルを手動でクリーンアップするには、次の3つのオプションのいずれかを使用します。

--vacuum-time=
この時間に基づくオプションは、指定した時間よりも古いジャーナルメッセージを削除します。時間は、以下のいずれかのサフィックスを付けた数値で指定します: s、m、h、days（d）、months、weeks（w）、years（y）。例として、ジャーナルのアーカイブから1ヶ月以上前のメッセージをすべて削除してみましょう:

```
root@debian:~# journalctl --vacuum-time=1months
Deleted archived journal
/var/log/journal/7203088f20394d9c8b252b64a0171e08/system@27dd08376f71405a91794e632ede97ed-0000000000000001-
00059475764d46d6.journal (16.0M).
Deleted archived journal /var/log/journal/7203088f20394d9c8b252b64a0171e08/user-
1000@e7020d80d3af42f0bc31592b39647e9c-000000000000008e-00059479df9677c8.journal (8.0M).
```

--vacuum-size=
この容量に基づくオプションは、ジャーナルが指定したサイズを下回るまで、ジャーナルアーカイブを削除します。容量は、以下のいずれかのサフィックスを付けた数値で指定します: K、M、G、T。例

として、アーカイブされたジャーナルファイルが 100メガバイト未満になるまで削除してみましょう:

```
root@debian:~# journalctl --vacuum-size=100M
Vacuuming done, freed 0B of archived journals from /run/log/journal/9a32ba45ce44423a97d6397918de1fa5.
```

`--vacuum-files=`
このオプションは、アーカイブジャーナルファイルの数が指定した数を下回るように古いアーカイブファイルを削除します。例として、10世代よりも古いアーカイブを削除してみましょう:

```
root@debian:~# journalctl --vacuum-files=10
Vacuuming done, freed 0B of archived journals from /run/log/journal/9a32ba45ce44423a97d6397918de1fa5.
```

削除されるのは、アーカイブされたジャーナルファイルのみです。アクティブなジャーナルを含む、すべてのエントリを削除したい場合は、`--rotate` オプションを使用します。このオプションは、古いジャーナルの削除に先立って、デーモンにジャーナルファイルの即時ローテーションを指示するシグナル（SIGUSR2）を送信します。他にもデーモンに指示を与えるシグナルを送信するオプションがあります。

`--flush（SIGUSR1）`
ジャーナルをディスクに保存して永続化するために、`/run/` から `/var/` へのジャーナルファイルのフラッシュを指示します。永続ロギングが有効で、`/var/` がマウントされている必要があります。

`--sync（SIGRTMIN+1）`
書き込まれていないすべてのログデータの、ディスクへの書き込み指示します。

NOTE ジャーナルファイルの内部整合性を確認するには、`journalctl` の `--verify` オプションを使用します。チェック中には進行状況バーが表示され、問題があれば表示されます。

レスキューシステムからジャーナルデータを取得する

システム管理者は、レスキューシステム（ハードディスクを使用せずCDやUSBディスクから起動するLinuxシステム）を用いて、障害のあるマシンのハードディスク上のジャーナルファイルにアクセスしなければいけないことがあります。

`journalctl` は、ジャーナルファイルを `/var/log/journal/<machine-id>/` ディレクトリから探します。レスキューシステムと障害が発生しているシステムのマシンIDは異なるため、次のオプションを使用する必要があります。

`-D </path/to/dir>、--directory=</path/to/dir>`
このオプションで、`journalctl` がジャーナルファイルを検索するディレクトリパスを指定します。

つまり、障害のあるシステムの ルートFS（`/dev/sda1`）を、レスキューシステムのファイルシステム（`/media/carol/faulty.system/`）にマウントしたとすると、次のようにターゲットシステムのジャーナルファイルにアクセスします。

```
root@debian:~# journalctl -D /media/carol/faulty.system/var/log/journal/
-- Logs begin at Sun 2019-10-20 12:30:45 CEST, end at Sun 2019-10-20 12:32:57 CEST. --
```

```
oct 20 12:30:45 suse-server kernel: Linux version 4.12.14-lp151.28.16-default (geeko@buildhost) (...)
oct 20 12:30:45 suse-server kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.12.14-lp151.28.16-default
root=UUID=7570f67f-4a08-448e-aa09-168769cb9289 splash=>
oct 20 12:30:45 suse-server kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
oct 20 12:30:45 suse-server kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
(...)
```

このケースでは、次のようなオプションも役立ちます。

`-m`、`--merge`
`/var/log/journal` にある、リモートマシンから転送されたものを含むすべてのジャーナルをマージします。

`--file`
指定したファイルのみからエントリを表示します。たとえば `journalctl --file /var/log/journal/64319965bda04dfa81d3bc4e7919814a/user-1000.journal` など。

`--root`
ジャーナルを探すファイルシステムのルートディレクトリを指定します。たとえば `journalctl --root /faulty.system/` など。

詳細は、`journalctl` のマニュアルページを参照してください。

ログメッセージを `syslog` に転送する

以下のいずれかの方法で、ジャーナルからのログデータを、`syslog` デーモンで利用できます。

- journaldの設定ファイルに `ForwardToSyslog=yes` オプションを指定して、ログメッセージを `/run/systemd/journal/syslog` ソケット経由で `syslogd` に転送する。
- `rsyslog` デーモンにジャーナルを読み込むモジュールを組み込み、ジャーナルファイルから直接ログメッセージを読み取らせる。この場合、journaldの設定ファイルで、`Storage` が `none` 以外の値を指定する必要があります。

NOTE
他の宛先にログメッセージを転送することもでき、次のオプションを使用します: `ForwardToKMsg`（カーネルログバッファ/`kmsg`）、`ForwardToConsole`（システムコンソール）、`ForwardToWall`（`wall` 経由ですべてのログインユーザー）。詳細は、`journald.conf` のmanページを参照してください。

演習

1. 目的に応じた `journalctl` コマンドで表を完成させてください。

目的	コマンド
カーネルからのエントリを表示する	
ジャーナルの先頭から2番目のブートのメッセージを表示する	
ジャーナルの末尾から2番目のブートのメッセージを表示する	
直近のログメッセージを表示し、新しいメッセージを監視し続ける	
新着のメッセージを表示し続ける	
ブート以後のプライオリティが `warning` であるメッセージを逆順に表示する	

2. ジャーナルデーモンのストレージは、`/etc/systemd/journald.conf` の `Storage` オプションで制御されます。それぞれの動作に対応する `Storage` の値はどれですか？

動作	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
ログデータは破棄されるが、転送は行われる				
システムが起動するとログデータが `/var/log/journal` に保存される。ディレクトリが存在しない場合は作成される				
システムが起動するとログデータが `/var/log/journal` に保存される。ディレクトリが存在しない場合でも作成さない				
ログデータが `/var/run/journal` の下に保存されるが、再起動後によって無くなる				

3. ジャーナルを、時間、サイズ、ファイル数に基づいて手動で削除できることを学びました。次のタスクを実行するための、`journalctl` コマンドはどうなりますか？

- ジャーナルファイルが占めるディスク容量を確認する:

- アーカイブジャーナルのファイル容量を、200MiB以内におさめる:

- ディスク容量を再確認して、結果を分析する:

発展演習

1. メッセージを `/dev/tty5` に転送するために、`/etc/systemd/journald.conf` に指定するオプションと値は何ですか？

2. 目的の出力を得るための、`journalctl` フィルターは何ですか？

目的	**フィルター + 値**
あるユーザーによるメッセージを表示する	
`debian` という名前のホストからのメッセージを表示する	
あるグループに属するユーザーによるメッセージを表示する	
`root` によるメッセージを表示する	
コマンドのパスを用いて、`sudo` からのメッセージを表示する	
コマンド名を用いて、`sudo` からのメッセージを表示する	

3. プライオリティでフィルタリングする場合は、指定したプライオリティより高いプライオリティのログも含まれます。たとえば、`journalctl -p err` は、error、critical、alert および emergency のメッセージを出力します。しかし、`journalctl` が表示するプライオリティ範囲を指定することもできます。`journalctl` に、プライオリティが warning、error、critical であるメッセージのみを出力させるにはどうしますか？

4. プライオリティを数値で指定することもできます。数値表現を使用して、前の演習のコマンドを書き直してください。

まとめ

このレッスンでは、以下の事柄を学びました:

- システム/サービスマネージャーとして `systemd` を使用する利点。
- `systemd` ユニットとターゲットの基本。
- `systemd-journald` がどこからログデータ取得するか。
- `systemd-journald` を制御する `systemctl` のオプション: `start`、`status`、`restart`、`stop`。
- ジャーナルの設定ファイル（`/etc/systemd/journald.conf`）と、その主要なオプション。
- ジャーナルから条件に一致するデータのみを取り出す方法。
- ジャーナルをナビゲートおよび検索する方法。
- ジャーナルファイルを保存する方法: メモリ vs ディスク。
- ジャーナル記録を無効化する方法。
- ジャーナルが占有しているディスク容量を確認する方法、保存するジャーナルファイルのサイズを制限する方法、アーカイブされたジャーナルファイルを手動で削除する方法（バキューム）。
- レスキューシステムからジャーナルデータを取得する方法。
- ログデータを従来の `syslog` デーモンに転送する方法。

このレッスンでは、以下のコマンドを説明しました。

`systemctl`
`systemd` システム/サービスマネージャーを制御します。

`journalctl`
`systemd` ジャーナルを条件に従って表示します。

`ls`
ディレクトリの内容を一覧表示します。

`less`
ファイルの内容を（ページごとに）表示します。

`mkdir`
ディレクトリを作成します。

演習の解答

1. 目的に応じた `journalctl` コマンドで表を完成させてください。

目的	コマンド
カーネルからのエントリを表示する	`journalctl -k` ないし `journalctl --dmesg`
ジャーナルの先頭から2番目のブートのメッセージを表示する	`journalctl -b 2`
ジャーナルの末尾から2番目のブートのメッセージを表示する	`journalctl -b -2 -r` オプションの順序は問わない
直近のログメッセージを表示し、新しいメッセージを監視し続ける	`journalctl -f`
新着のメッセージを表示し続ける	`journalctl --since "now" -f`
ブート以後のプライオリティが `warning` であるメッセージを逆順に表示する	`journalctl -b -1 -p warning -r`

2. ジャーナルデーモンのストレージは、`/etc/systemd/journald.conf` の `Storage` オプションで制御されます。それぞれの動作に対応する `Storage` の値はどれですか？

動作	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
ログデータは破棄されるが、転送は行われる		○		
システムが起動するとログデータが `/var/log/journal` に保存される。ディレクトリが存在しない場合は作成される			○	
システムが起動するとログデータが `/var/log/journal` に保存される。ディレクトリが存在しない場合でも作成さない	○			
ログデータが `/var/run/journal` の下に保存されるが、再起動後によって無くなる				○

3. ジャーナルを、時間、サイズ、ファイル数に基づいて手動で削除できることを学びました。次のタスクを実行するための、`journalctl` コマンドはどうなりますか？

- ジャーナルファイルが占めるディスク容量を確認する:

```
journalctl --disk-usage
```

- アーカイブジャーナルのファイル容量を、200MiB以内に設定する:

```
journalctl --vacuum-size=200M
```

- ディスク容量を再確認して、結果を分析する:

```
journalctl --disk-usage
```

`--disk-usage` はアクティブなジャーナルとアーカイブされたジャーナルの合計容量を示しますが、`--vacuum-size` はアーカイブのみに適用されます。

発展演習の解答

1. メッセージを `/dev/tty5` に転送するために、`/etc/systemd/journald.conf` に指定するオプションと値は何ですか？

```
ForwardToConsole=yes
TTYPath=/dev/tty5
```

2. 目的の出力を得るための、`journalctl` フィルターは何ですか？

目的	フィルター + 値
あるユーザーによるメッセージを表示する	`_ID=<user-id>`
`debian` という名前のホストからのメッセージを表示する	`_HOSTNAME=debian`
あるグループに属するユーザーによるメッセージを表示する	`_GID=<group-id>`
`root` によるメッセージを表示する	`_UID=0`
コマンドのパスに基づいて、`sudo` からのメッセージを表示する	`_EXE=/usr/bin/sudo`
コマンド名に基づいて、`sudo` からのメッセージを表示する	`_COMM=sudo`

3. プライオリティでフィルタリングする場合は、指定したプライオリティより高いプライオリティのログも含まれます。たとえば、`journalctl -p err` は、error、critical、alert および emergency のメッセージを出力します。しかし、`journalctl` が表示するプライオリティ範囲を指定することもできます。`journalctl` に、プライオリティが warning、error、critical であるメッセージのみを出力させるにはどうしますか？+

```
journalctl -p warning..crit
```

4. プライオリティを数値で指定することもできます。数値表現を使用して、前の演習のコマンドを書き直してください。

```
journalctl -p 4..2
```


108.3 メール転送エージェント(MTA)の基本

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 108.3

総重量

3

主な知識分野

- 電子メールエイリアスを作成する。
- 電子メール転送を構成する。
- 一般的に利用可能なMTAプログラム(postfix, sendmail, exim)に関する知識(設定なし)。

用語とユーティリティ

- `~/.forward`
- sendmail emulation layer commands
- `newaliases`
- `mail`
- `mailq`
- `postfix`
- `sendmail`
- `exim`

108.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	108 基本的なシステムサービス
Objective:	108.3 メール転送エージェント（MTA）の基礎
Lesson:	1 of 1

はじめに

LinuxなどのUnix系オペレーティングシステムでは、すべてのユーザーが専用の 受信箱（メールボックス）を持っています。受信箱の正体は、他のユーザーがアクセスできないそのユーザー専用のファイルないしディレクトリです。メールメッセージの受信から受信箱への配送、ならびに、メールメッセージのポストから宛先アドレスのメールサーバーへの配送全体を制御するソフトウェアを、歴史的に メール転送エージェント（Mail Transfer Agent） と呼びます。その役割の1つは、ネットワークないしはローカルユーザーから届いたローカルユーザ宛のメールメッセージを、そのユーザーのメールボックスに配送することです。

MTAのもうひとつの重要な役割は、ローカルユーザーが投函したメッセージ（ローカルマシンにログインして投函するケースと、ネットワーク経由で投函するケースがあります）を、宛先アドレスのメールサーバーに宛てて SMTP（Simple Mail Transfer Protocol） というプロトコルを使ってネットワーク経由で送信することです。宛先アドレスに応じたメールサーバーは、アドレスのドメイン部（`@`記号以降の部分）から判断します。ネットワークのトラブルなどに対応するために、代行の宛先メールサーバーを使用したり、時間を空けて配送を再試行するといった仕組みも備わっています。

ローカルMTAとリモートMTA

ローカルユーザー同士でメッセージをやりとりするのが、電子メールの最もシンプルなケースです。この場合は、MTAのみでメッセージをメールメッセージを相手のメールボックスに届けることができます。ローカルマシンのメールボックスを読み出すためのユーティリティについては後で説明します。現在では、リモートのメールサーバーにアカウントを作成して、メールクライアントアプリケーション（Mail User Agent）からネットワーク経由でサーバーにアクセスするのが一般的です。この場合、各ユーザーのクライアントマシンに、MTAをインストールする必要はありません。

メールクライアントからメールサーバー上のメールボックスにアクセスするには、ユーザー認証を行っ

てからPOP3やIMAPと呼ばれるプロトコルを使います（LPIC-1の範囲外ですので本レッスンでは触れません）。また、メールクライアントからメールサーバーにメッセージを送信する際には、やはりユーザー認証を行ってから、電子メールサーバー間でのやりとりと同様のSMTPを使います。これをSubmissionサービスと呼ぶことがあります。

NOTE

現在では、ユーザーが使うローカルマシンにメールメッセージを配送することは稀で、すべての従業員アカウントをホストする会社の集中メールサーバーや、Gmail などのメールサービスにリモートアクセスするのが一般的です。ローカルマシンにメッセージを配信するのではなく、メールクライアントアプリケーションでリモートメールボックスに接続して、メッセージを読み出します。

MTAを実行しているマシンのユーザーは、同様にMTAを実行しているリモートマシンのユーザーに宛てたメールを送信することも出来ます。ローカルマシンでポストされたメッセージは、ローカルのMTAによって、リモートマシンのMTAに伝送されます。その際に使用されるSMTPは、TCPポート25番を使用します。（同じSMTPですが、Submissionサービスでは587（STARTTLS）ないし465（SSL）を使うのが普通です。）

MTAサービスを実行しているシステム間で、電子メールの交換を行う仕組みは以下の通りです:

- 送信するメッセージを、送信キューと呼ばれる所定のディレクトリに置きます。ローカルMTAはメッセージファイルを1つずつ取り出して、その宛先アドレスから送信先のシステムを決定します（後述）。
- ローカルMTAは、SMTPを使用してリモートMTAにメールメッセージを送信します。
- リモートMTAは、宛先のユーザーが存在することを確認します。ユーザーが存在しなければ、SMTPの中で「宛先ユーザーが居ない」エラーで受信を拒否します。
- 宛先ユーザーが存在すれば、そのメールボックスにメッセージを追加します。メールボックスは、mbox 形式と呼ばれるすべてのメールメッセージを順に並べた1つのテキストファイルか、Maildir 形式と呼ばれる個々のメールメッセージファイルを収めたディレクトリです。

メールアドレスのドメイン名（例えば `info@lpi.org` の `lpi.org`）から、配送先のホストが分かります。つまり、送信元のMTAは、宛先のドメイン名に対するMXレコードをDNSに問い合わせます。MXレコードには、そのドメイン宛のメールを受信して処理するメールサーバーのホスト名ないしIPアドレスが記載されています。ひとつのドメインに複数のMXレコードが定義されていることがあり、その場合は優先度にしたがって順に接続を試みます。ドメイン名に対応するMXレコードが無い場合は、`@` より右側の部分を ホスト名 と見なしてそのホストに対して接続を試みます。

MTAを実行するホストをインターネットに接続する場合は、セキュリティに十分に注意しなくてはなりません。たとえば、迷惑メール（SPAM）の送信を行いたい攻撃者は、送信者の身元を確認することなくメールを中継する オープンリレー と呼ばれるMTAを常に探し回っています（後述）。認証によって身元が確認された場合にのみメール送信を許可すると共に、自分とは無関係なドメインからのメッセージを中継しないように設定することが必須です。

Linuxには、それぞれに特徴を持ったいくつかのMTA実装がありますが、いずれも同じ原則に則って、同じような機能を実現しています。

LinuxのMTA

Linuxに限らず多くのUNIX系オペレーティングシステムで実行できるMTAでは、Sendmail が最も歴史があるものです。後に Postfix、qmail、Exim などが登場してきました。これらを使うと、Sendmail では複雑な設定作業が必要となる機能を、比較的簡単に実現することができます。また、多くのディストリビューションでは、一般的な構成用の設定を含む優先MTAを提供しています。ま

た、新しいMTAはいずれもSendmailと互換性がありますので、Sendmail用に書かれたアプリケーションはそのまま動作します。（訳注:2023年現在、Linuxで人気のMTAは `Exim`、`Postfix`、`Sendmail` の3つです）。

メールは重要なアプリケーションであり、その中核となるMTAの設定は運用環境によって異なります。実用的なメールサーバーを構築するためには、さまざまなツールを複合的に組み合わせることも必要です。そのため、MTAの設定はLPIC-1の範囲外となっています。しかしながら、ごく一般的かつ汎用的な構成 --- たとえば、インターネットに接続して1つのドメインのメールサーバーとする、中央のメールサーバー（スマートホスト）に全メッセージを転送する、ローカルメールだけを取り扱う、など --- を指定して、半自動でMTAの設定を行うツールが、ディストリビューションから提供されていることがあります。簡単なメール送受信の実験環境を構築してみる場合には利用してみましょう。ここでは、SMTPの概要を紹介します。

TIP

セキュリティの理由から、ほとんどのLinuxディストリビューションは、デフォルトではMTAをインストールしません。以下の例を試してみる場合には、すべてのマシンでMTAデーモンが実行されていて、TCP25番ポートで接続を受け入れていることが必要です。テストマシンがファイアウォールの内側にあり、インターネットから接続できないことを確認して下さい。

MTAデーモンがネットワークからの着信を受け付ける場合、メッセージの送受信には SMTP（Simple Mail Transfer Protocol） を使用します。ネットワーク上でデータをやり取りする汎用的なユーティリティである `nc` コマンドを使用して、MTAにSMTPコマンドを送信することができます。（`nc` コマンドは、ncat パッケージ、ないし nmap-ncat パッケージに含まれているのが一般的です）。MTAと直接SMTPをやり取りしてみると、プロトコルと電子メールの仕組みをより深く理解できますし、トラブルシュートの際にも役立ちます。（訳注: いずれもLPIC-2の出題範囲で、LPIC-1では取り上げられません）。

`lab1.campus` のユーザー `emma` が、`lab2.campus` のユーザー `dave` にメッセージを送信する例を見ていきましょう。`nc` コマンドを使用して `lab2.campus` のMTAに直接接続します。MTAデーモンはTCPポート25番で待ち受けています。（訳注: `lab1.compus` や `lab2.campus` は、ローカルネットワークにおける「ホスト名」であることに注意してください。組織内DNSやhostsファイルで名前解決ができることが必要です。）

```
$ nc lab2.campus 25
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov 2019 00:16:07 GMT
HELO lab1.campus
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to meet you
MAIL FROM: emma@lab1.campus
250 2.1.0 emma@lab1.campus... Sender ok
RCPT TO: dave@lab2.campus
250 2.1.5 dave@lab2.campus... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject: Recipient MTA Test

Hi Dave, this is a test for your MTA.
.
250 2.0.0 xAG0G7Y0000595 Message accepted for delivery
QUIT
221 2.0.0 lab2.campus closing connection
```

リモートMTAに接続すると、ホスト名やプロトコル名、バージョン番号などが提示されます。送信する最初の文字列 `HELO lab1.campus` は、自分のホスト名が `lab1.campus` であることを提示します。続く2つのコマンド、`MAIL FROM: emma@lab1.campus` と `RCPT TO: dave@lab2.campus` は、送信者メールアドレスと受信者メールアドレスを示します。`DATA` コマンドの後からピリオドだけの1行の直前までが、電子メールメッセージになります。メールヘッダとして `subject` などのフィールドを追加するには、例に示すように、`DATA` コマンドの後に続けて送信します。すべてのヘッダー行を送信したら、1つの空行を空けてからメッセージ本文を送信し、最後にピリオドのみの行を送信します。メッセージが受け付けられて、配信に回されたことがリモートMTAから通知されます。最後の `QUIT` コマンドは、`lab2.campus` ホストのMTAとの接続を終了します。

`lab2.campus` ホストでユーザー `dave` がシェルセッションを開始するとすぐに、`You have new mail in /var/spool/mail/dave` のようなメッセージを受け取ることでしょう。このファイル（メールボックス）には、`emma` によって送信されたメールメッセージ本文と、MTAによって追加されたヘッダーが含まれます。

```
$ cat /var/spool/mail/dave
From emma@lab1.campus  Sat Nov 16 00:19:13 2019
Return-Path: <emma@lab1.campus>
Received: from lab1.campus (lab1.campus [10.0.3.134])
    by lab2.campus (8.15.2/8.15.2) with SMTP id xAG0G7Y0000595
    for dave@lab2.campus; Sat, 16 Nov 2019 00:17:06 GMT
Date: Sat, 16 Nov 2019 00:16:07 GMT
From: emma@lab1.campus
Message-Id: <201911160017.xAG0G7Y0000595@lab2.campus>
Subject: Recipient MTA Test

Hi Dave, this is a test for your MTA.
```

`Received:` ヘッダーは、`lab1.campus` からのメッセージを `lab2.campus` が受信したことを示しています。通常、MTAはローカル受信者へのメッセージのみを受け入れます。次の例では、ユーザー `emma` が `lab3.campus` ホストのユーザー `henry` に電子メールを送信しようとしていますが、`lab3.campus` のMTAではなく、`lab2.campus` のMTAに接続しているためにエラーとなっています。

```
$ nc lab2.campus 25
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov 2019 00:31:44 GMT
HELO lab1.campus
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to meet you
MAIL FROM: emma@lab1.campus
250 2.1.0 emma@lab1.campus... Sender ok
RCPT TO: henry@lab3.campus
550 5.7.1 henry@lab3.campus... Relaying denied
```

受信側の `lab2.campus` が返すメッセージの先頭にある3桁の数字は、応答のステータスを示しています。100の位が5でである番号はエラーを示していて、この場合は `Relaying denied` がその理由を示しています。（訳注: `lab1.campus` から `lab3.campus` に宛てたメールですから、`lab2.campus` から見ると「中継」を依頼されたように見えるため「中継拒否」のエラーになります。このような「中継」を無条件に受け入れてしまうのが「オープンリレー」です）。100の位が2である番号は正常に受け付けられたことを示し、4である番号は一時的に受け付けられないので後で再送して欲しいことを示します。

この場合はエラーを返していますが、リレーが望ましい場面もあります。たとえば、大きな組織で部署ごとにメールドメインとメールサーバーを利用しているような場合には、全社で共通利用する親メールサーバーを用意して、外部と各部署メールサーバ間のリレーを許可します（スマートホストと言います）。そして、部署メールサーバーでは自分の部署ドメイン宛て以外のメールメッセージをすべて親メールサーバーに送信するように設定します。

こうすると、それぞれの部署サーバーではメール配送のために外部DNSを検索する必要が無くなりますし、すべてのメールメッセージがスマートホストを経由するので、スマートホストのセキュリティを強化することで全社のメールセキュリティを強化することができます。

ほとんどすべてのMTAには、伝統的なsendmailと互換性を維持するための `sendmail` コマンドが備わっています。通常は後述する MUA（Mail User Agent） を使用してメールを送信しますが、`sendmail` コマンド直接呼び出してメールを送信することもできます。たとえば、ユーザー `emma@lab1.campus` が `dave@lab2.campus` にメッセージを送信する例を見てみましょう。この場合も、メールヘッダーはユーザーが入力する必要があります。

```
$ sendmail -f emma@lab1.campus dave@lab2.campus
From: emma@lab1.campus
To: dave@lab2.campus
Subject: Sender MTA Test

Hi Dave, this is a test for my MTA.
.
```

`-f` オプションで送信者のアドレスを、引数で宛先のアドレスを指定します。`sendmail` コマンドは、標準入力からメールメッセージの入力を待ち受けますので、ヘッダー行に続けて1行の空行を置いてメールメッセージの本文を入力します。ピリオド（ドット）のみの行を入力して、メッセージの終了を示します。ローカルMTAが宛先のリモートMTAに接続できればメッセージはすぐに送信されますが、宛先MTAが応答しないといった場合には、送信キュー と呼ばれるディレクトリにメッセージファイルが保存されて、所定の時間が経過した後に再度送信を試みます。

送信キュー に保留されているメッセージの一覧を、`mailq` コマンドで確認することができます。`lab2.campus` が応答しなかった場合には、次のように未配信のメッセージとその理由を表示します。

```
# mailq
        /var/spool/mqueue (1 request)
-----Q-ID----- --Size-- -----Q-Time----- ------------Sender/Recipient-----------
xAIK3D9S000453       36 Mon Nov 18 20:03 <emma@lab1.campus>
                (Deferred: Connection refused by lab2.campus.)
                    <dave@lab2.campus>
        Total requests: 1
```

送信キューのデフォルトは `/var/spool/mqueue/` ですが、MTAによって異なります。たとえば、Postfixは、`/var/spool/postfix/` の下にディレクトリツリーを作成してキューを管理します。`mailq` は `sendmail` へのシンボリックリンクであり、`sendmail -bp` と同じ働きをします。

SMTPでは、DNSで宛先ドメインのMXレコードを検索して送信先メールサーバーを見つけることを述べました。複数のMXレコードがある場合には、優先度に従ってすべてのメールサーバーへの接続を順に試みます。いずれにも接続できなかった場合には、送信キューにメッセージを置いて、いったん処理を終

了します。MTAは送信キューを常時監視しており、所定の時間が経過するたびにキューに残っているメッセージの再配送を試みます。`sendmail -q` コマンドで、送信キューに残っているすべてのメールメッセージの再送をすぐに試みさせることができます。所定の期間（通常は数日から1週間）が過ぎても送信キューに残っているメッセージは、エラーメッセージを添えて送信者に返送されます。

ユーザー毎のメールボックス（受信箱）はMTAごとに異なり、たとえば `dave` のメールボックスは、sendmailでは `/var/spool/mail/dove`、postfixでは `/var/mail/dave` がデフォルトです。メールボックスを直接読み出して、メールメッセージの正体をもう一度見てみましょう。今度は、メールメッセージが中継されていく模様に着目します。前の例と同じく、`emma@lab1.campus` がsendmailコマンドを使用して、`dave@lab2.campus` にメッセージを送った場合を取り上げます。

```
$ cat /var/spool/mail/dave
From emma@lab1.campus  Mon Nov 18 20:07:39 2019
Return-Path: <emma@lab1.campus>
Received: from lab1.campus (lab1.campus [10.0.3.134])
    by lab2.campus (8.15.2/8.15.2) with ESMTPS id xAIK7clC000432
    (version=TLSv1.3 cipher=TLS_AES_256_GCM_SHA384 bits=256 verify=NOT)
    for <dave@lab2.campus>; Mon, 18 Nov 2019 20:07:38 GMT
Received: from lab1.campus (localhost [127.0.0.1])
    by lab1.campus (8.15.2/8.15.2) with ESMTPS id xAIK3D9S000453
    (version=TLSv1.3 cipher=TLS_AES_256_GCM_SHA384 bits=256 verify=NOT)
    for <dave@lab2.campus>; Mon, 18 Nov 2019 20:03:13 GMT
Received: (from emma@localhost)
    by lab1.campus (8.15.2/8.15.2/Submit) id xAIK0doL000449
    for dave@lab2.campus; Mon, 18 Nov 2019 20:00:39 GMT
Date: Mon, 18 Nov 2019 20:00:39 GMT
Message-Id: <201911182000.xAIK0doL000449@lab1.campus>
From: emma@lab1.campus
To: dave@lab2.campus
Subject: Sender MTA Test

Hi Dave, this is a test for my MTA.
```

メールヘッダの `Received:` 行は、下から順にメッセージがたどった経路を示しています。一番下の `Received:` ヘッダーは、`lab1.campus` で、ユーザー `emma` が、`sendmail` コマンドを直接使って、`dave@lab2.campus` 宛のメッセージを送信したことを示しています。 次の `Received:` 行は、`lab1.campus` のMTAデーモン（この場合はsendmail）が、そのメッセージを受け付けたことを示しています。3分ほど経過していることから、メッセージを送信してからMTAデーモンを起動したことが推察されます。 一番上の `Received:` ヘッダは、`lab1.campus` からESMTPS（暗号化されたSMTPの改良版）で送られたメッセージを、`lab2.campus` のMTAが受信したことを示しています。`lab1@campus` のIPアドレスが記録されていることにも着目しましょう。このように、メールヘッダの `Received:` 行をたどると、メッセージがたどってきたホストのIPアドレスや、利用されたプロトコルなどを調べることが出来ます。

ユーザーのメールボックスにメッセージを保存したら、MTAの仕事は完了です。ただし現在は、スパムブロッカーやユーザーによるフィルタリングなど、追加の処理を行うことが一般的です。それらの処理は、MTAと連携する別のアプリケーションで行います。たとえば、SpamAssassin（スパムアサシン）という迷惑メールフィルタリングアプリケーションを呼び出して、メッセージテキストを分析して迷惑メールをマークするなどの処理を行います。

メールボックスのファイルを直接読み出すこともできますが、通常はメールクライアントアプリケーシ

ョン（たとえば
ThunderbirdやEvolution、KMailなど）を使います。多くのメールクライアントは、メールボックスを分かりやすく表示するだけでなく、「返信」などの一般的なメール操作や、個人用フォルダなどの機能も備えています。

`mail` コマンドとMUA（Mail User Agent)

送信するメールメッセージを適切なフォーマットで作成するためにも、メールクライアントアプリケーションである MUA（Mail User Agent） を使用することがお勧めです。

MUAには多くの種類があります。Mozilla ThunderbirdやGnomeのEvolutionのようなデスクトップアプリケーションは、ローカルとリモートの両方のメールアカウントをサポートしています。WebメールもMTAと連係して動作しますから、一種のMUAと考えられます。MUAにはCLIで動作するものもあり、特にシェルスクリプトでメール関連タスクを行う場合に広く使われています。

Unixの `mail` コマンドは、元々はローカルシステムユーザー間でメッセージをやり取りすることだけを目的としていました（最初の `mail` コマンドは、1971年にリリースされた最初のUnixにまでさかのぼります）。ネットワーク上でのメール交換が一般的になると、新しい配信システムに対処するためにさまざまなプログラムが作成されて、古い `mail` プログラムを置き換えていきました。

現在も広く使われている `mail` コマンドは、最新のメール機能に対応している `mailx` パッケージによって提供されています（訳注: `mailx` パッケージが見つからない場合は、s-nail パッケージを探してみてください。ディストリビューションによっては、`mail` コマンドで `s-nail` が呼び出されます）。GNU Mailutils パッケージなど、基本的に `mailx` と同じ機能を備えた別の実装もありますが、コマンドラインオプションなどにわずかな違いがあります。このレッスンでは、最も基本的な使用方法のみを説明します。

どの実装でも、`mail` コマンドは 受信モード と 送信モード を持っています。コマンド引数にメールアドレスを指定すると送信モードで、その他の場合はノーマル（受信）モードです。ノーマルモードでは、メールボックスにある受信したメッセージの一覧が番号付きで表示されるので、プロンプトに対してコマンドを入力します。たとえば、`print 1` コマンドは、1番のメッセージを表示します。コマンドには省略形があり、`print` は `p`、`delete` は `d`、`reply` は `r` などが使えます。メッセージ番号を省略した時には、直近に到着したメッセージか、直近に参照したメッセージを仮定します。`quit` ないし `q` コマンドで、プログラムは終了します。

メール送信を自動化する場合には、送信モード を使います。たとえば、定時のメンテナンススクリプトの実行に失敗した場合に、管理者宛にメールを送信するといった場合です。送信モードでは、標準入力の内容を、メッセージ本文として使用します。

```
$ mail -s "Maintenance fail" henry@lab3.campus <<<"The maintenance script failed at `date`"
```

この例では、メッセージに件名（Subject）フィールドを追加するために、`-s` オプションを指定しています。標準入力がメッセージ本文になりますから、直接タイプするか、ヒアドキュメントやパイプで本文を入力します。直接タイプする場合、入力の終了にはEOF（Ctrl+D）を使います。オリジナルの `mail` コマンドはメッセージを送信キューに置いて終了しますが、現在は `sendmail` コマンドを呼び出してメールを送信するものが多いようです。オプションで送信方法を指定できるものもあるので、インストールしたパッケージのマニュアルを参照してください。

配信のカスタマイズ

デフォルトでは、システムにおけるアカウント名がそのままメールアドレスになります。たとえば、Carolの `lab2.campus` におけるアカウントが `carol` である場合、そのメールアドレスは `carol@lab2.campus` になります。アカウント名とメールアドレスは1対1に対応付けられますが、ほとんどのLinuxディストリビューションが備えている、`/etc/aliases` ファイルによるメールアドレスの別名機能によって、対応付けを変更したり、拡張することができます。

メールの別名（エイリアス）は 仮想的な メール受信者であり、受信したメッセージは、既存アカウントのメールボックスや、その他のメールストアに保存されるか、別のアドレスに宛てて転送されます。たとえば、`postmaster@lab2.campus` 宛に送信されたメッセージを、`lab2.campus` のユーザー carol のローカルメールボックスに配送することができます。そのためには、`lab2.campus` の `/etc/aliases` ファイルに、`postmaster: carol` という行を追加します。`/etc/alias` ファイルを変更した後には、`newaliases` コマンドを実行してMTAが参照する別名データベースを更新して有効化します。`sendmail -bi` ないし `sendmail -I` を使用して、別名データベースを更新することもできます。

エイリアスは1行に1つずつ、`<alias（別名）>: <destination（宛先）>` という形式で定義します。カンマ `,` で区切って、複数の `<destination>` を指定できます。ユーザーアカウントに対応する通常のローカルメールボックスだけでなく、次のような宛先も指定できます。

- ファイルの（/ で始まる）フルパス名。エイリアス宛のメッセージは、ファイルに追記されます。
- メッセージを処理するコマンド。`<destination>` の先頭にパイプ文字（|）を置き、コマンドを続けます。コマンドに特殊文字（空白など）が含まれる場合は、二重引用符で囲みます。たとえば、`lab2.campus` のエイリアス `subscribe: |subscribe.sh` は、`subscribe@lab2.campus` 宛に送られたメッセージを、コマンド `subscribe.sh` の標準入力に送ります。現在のsendmailでは、制限付きモードのシェル（rmrsh）を用いてコマンドを呼び出すので、所定のディレクトリ（ディストリビューションによって異なる）にあるコマンドのみを実行することができます。（訳注: sendmail以外のMTAでは特別な制約はありませんが、潜在的なセキュリティリスクに注意してください）。
- ファイルの取り込み。1つのエイリアスに対して複数の宛先を指定することができるため、エイリアス毎の外部ファイルを使って宛先アドレスを指定すると便利です。たとえば `:include:/var/local/destinations` のように、キーワード `:include:` に続けて、宛先アドレスを収めたファイルパスを指定します。
- 外部アドレス。ローカルユーザーだけでなく、インターネット形式の外部メールアドレスにメッセージを転送することもできます。
- 別のエイリアス名。

一般ユーザーは、自分専用のエイリアスを定義することができます。すなわち、自分のアドレスに届いたメールを、別のアドレスに転送することができます。ホームディレクトリに、`.forward` というファイルを作成し、1行に1つずつ<destination>を記入します。たとえば、ユーザー `dave` がすべての受信メールを `emma@laba1.campus` に転送するには、次の `~/.forward` ファイルを作成します。なお、`.forward` ファイルのパーミッションは、所有者のみが書き込み可能でなくてはなりません。

```
$ cat ~/.forward
emma@lab1.campus
```

転送を行いながら、ローカルマシンのメールボックスにもメッセージを残したい場合は、<destination>の一つとして自分のユーザー名を置きます。ただし、エイリアスのループを避けるために、ユーザー名の前に `\` を付ける必要があります（例の場合は `\dave` になります）。`.forward` に

は、`/etc/aliases` ファイルと同様に、パイプなどのルールを置くこともできます。なお、`.forward` は隠しファイルですから、ユーザーがエイリアスに気付かない可能性があります。メール配信の問題を診断するときは、`.forward` ファイルの存在を確認することが重要です。

演習問題

1. `mail henry@lab3.campus` コマンドを実行すると入力待ち状態になるので、`henry@lab3.campus` 宛のメッセージを入力します。メッセージが終了した後に、入力モードを終えてメールを送信するにはどのキーを入力しますか？

2. ローカルシステムに留まっている未配信のメッセージを、一覧表示するコマンドは何ですか？

3. 標準的なMTAを利用しているシステムで、一般ユーザーが自分宛のすべてのメールを、`dave@lab2.campus` に自動的に転送するにはどうしますか？

発展演習

1. `mail` コマンドで、`emma@lab1.campus` 宛に、`uname -a` コマンドの出力を本文とし、`log.tar.gz` ファイルを添付したメールを送信するには、どのようなコマンドを使いますか？

2. ネットワーク経由のメール転送を監視したいが、自分やユーザーのメールボックスにテストメッセージを送りたくありません。`test` 宛に送られたすべてのメールを、ファイル `/dev/null` にリダイレクトするには、システム全体のエイリアスをどのように設定しますか？

3. `/etc/alias` に新しいエイリアスを追加した後に、そのデータベースを更新する `newaliases` 以外のコマンドにはどのようなものがありますか？

まとめ

このレッスンでは、Linuxシステムにおける MTA（Mail Transfer Agent） の役割と使用方法について説明しました。MTAは、標準的な電子メールの送受信方法を提供し、他のソフトウェアと組み合わせて機能を追加することもできます。このレッスンでは、次のトピックについて説明しました。

- メール関連技術、メールボックス、プロトコルの概念
- MTAがネットワーク経由でメッセージを交換する方法
- さまざまな MUA（Mail User Agents）。特にCLIで使用する `mail` コマンドの基本的な使い方
- メールのエイリアスと転送

以下の技術、コマンド、手順を説明しました:

- SMTP関連のプロトコル
- Linuxで利用可能なMTA: Sendmail、Postfix、qmail、Exim
- MTAコマンドとMUAコマンド: `sendmail`、`mail`
- 管理ファイルとコマンド: `mailq`、`/etc/aliases`、`newaliases`、`~/.forward`

演習の解答

1. `mail henry@lab3.campus` コマンドを実行すると入力待ち状態になるので、`henry@lab3.campus` 宛のメッセージを入力します。メッセージが終了した後に、入力モードを終えてメールを送信するためにはどのキーを入力しますか？

   EOF（Ctrl+D）を押すと、プログラムが閉じて電子メールが送信されます。

2. ローカルシステムに留まっている未配信のメッセージを、一覧表示するコマンドは何ですか？

   `mailq` ないし `sendmail -bp`

3. 標準的なMTAを利用しているシステムで、一般ユーザーが自分宛のすべてのメールを、`dave@lab2.campus` に自動的に転送するにはどうしますか？

   `~/.forward` に、`dave@lab2.campus` と書きます。

発展演習の解答

1. `mail` コマンドで、`emma@lab1.campus` 宛に、`uname -a` コマンドの出力を本文とし、`log.tar.gz` ファイルを添付したメールを送信するには、どのようなコマンドを使いますか？

   `uname -a | mail -a logs.tar.gz emma@lab1.campus`

2. ネットワーク経由のメール転送を監視したいが、自分やユーザーのメールボックスにテストメッセージを送りたくありません。`test` 宛に送られたすべてのメールを、ファイル `/dev/null` にリダイレクトするには、システム全体のエイリアスをどのように設定しますか？

   `/etc/aliases` に `test: /dev/null` という行を追加します。これにより、`test` 宛のすべてのメッセージが、ファイル `/dev/null` にリダイレクトされます。

3. `/etc/alias` に新しいエイリアスを追加した後に、そのデータベースを更新する `newaliases` 以外のコマンドにはどのようなものがありますか？

   `sendmail -bi` ないし `sendmail -I`

108.4 プリンタの管理と印刷

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 108.4

総重量

2

主な知識分野

- 基本的なCUPS設定(ローカルプリンタとリモートプリンタ用)。
- ユーザーの印刷キューを管理する。
- 一般的な印刷の問題のトラブルシューティング。
- 構成済みのプリンタキューからジョブを追加および削除する。

用語とユーティリティ

- CUPS configuration files, tools and utilities
- `/etc/cups/`
- lpd legacy interface (`lpr`, `lprm`, `lpq`)

108.4 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	108 基本的なシステムサービス
Objective:	108.4 プリンタと印刷の管理
Lesson:	1 of 1

はじめに

コンピュータの登場によってもたらされた “ペーパーレス社会” 宣言は、今のところ誤りであることが証明されています。多くの組織は、依然として印刷された、つまり “ハードコピー” の情報に依存しています。この事実は、コンピュータユーザーにとっては印刷方法を知ることが、システム管理者にとってはコンピュータとプリンタを連携させる方法を知ることが、重要であることを示しています。 Linuxでは、他の多くのOSと同様に、CUPS（Common Unix Printing System）ソフトウェアスタックが、コンピュータからの印刷とプリンタの管理を行います。ここでは、CUPSを使ってLinuxがファイルを印刷する方法の概要を示します:

1. 印刷するファイルをユーザーが送信する。
2. CUPSデーモン `cupsd` が、印刷ジョブを スプール する。印刷ジョブにはIDが割り当てられ、ジョブが投入された印刷キューと、ドキュメントの名前が関連付けられる。
3. CUPSは、インストールされている filters を使って、プリンタが理解できるフォーマットのファイルを生成する。
4. CUPSは、フォーマット済みのファイルをプリンタに送信する。

これらの手順について詳しく説明し、Linuxでプリンタをセットアップして管理する方法も説明します。

CUPSサービス

ほとんどのデスクトップ版Linuxには、CUPSパッケージがデフォルトでインストールされています。ディストリビューションによりますが、最小構成のLinuxではCUPSパッケージがインストールされていないこともあります。Debianシステムでは、以下のようにして基本的なCUPSをインストールできます。

```
$ sudo apt install cups
```

Fedoraシステムでも同様に、インストールは簡単です。FedoraなどのRed Hatベースのディストリビューションでは、インストール後に手動でCUPSサービスを起動する必要があります。(訳注:　常にサービスを起動したいときは、`systemctl enable` でサービス起動を有効化しておきましょう）。

```
$ sudo dnf install cups
...
$ sudo systemctl start cups.service
```

インストールしたら、`systemctl` コマンドで、CUPSサービスが実行されていることを確認します。

```
$ systemctl status cups.service
● cups.service - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-06-25 14:35:47 EDT; 41min ago
     Docs: man:cupsd(8)
 Main PID: 3136 (cupsd)
    Tasks: 2 (limit: 1119)
   Memory: 3.2M
   CGroup: /system.slice/cups.service
           ├─3136 /usr/sbin/cupsd -l
           └─3175 /usr/lib/cups/notifier/dbus dbus://
```

他の多くのデーモンと同様に、一連の設定ファイルでCUPSの動作を調整します。システム管理者にとって重要なものを以下に示します:

`/etc/cups/cupsd.conf`
: このファイルには、CUPSサービス本体の設定が含まれます。CUPSの設定ファイルはApache　Webサーバーの設定ファイルとよく似た構文を使用しています。`cupsd.conf`　　ファイルには、CUPSのWebインターフェースが有効かどうか、印刷キューへのアクセス権限、デーモンが使用するロギングのレベルなどの設定が含まれます。

`/etc/printcap`
: CUPSが登場する前に使われていたLPD（Line　Printer　Daemon）プロトコルが使用していたレガシーファイルで、互換性のために作成されます。多くの場合は　`/run/cups/printcap`　へのシンボリックリンクで、各行に1つのプリンタ情報が含まれます。

`/etc/cups/printers.conf`
: CUPSシステムで使用できるように構成したプリンタが含まれます。それぞれのプリンタとその印刷キューが、`<Printer></Printer>`　節で囲まれます。このファイルから、`/etc/printcap`　に置かれるプリンタの一覧が作られます。

> **WARNING**　CUPSサービスの実行中に、`/etc/cups/printers.conf`　　ファイルを直接変更してはいけません。

`/etc/cups/ppd/`
: このディレクトリには、プリンタのPPD（PostScript Printer Description）ファイルが置かれます。PPDファイル（サフィックスが `.ppd`）は、各プリンタの機能を示した、所定のフォーマットのプレーンテキストファイルです。

CUPSサービス、Apache 2サービスとほぼ同様のロギングを行います。`/var/log/cups/` 内に保存される `access_log`、`page_log`、および `error_log` がログファイルです。`access_log` には、CUPS Webインターフェースへのアクセスと、そこで実行されたプリンタ管理などの操作が記録されます。`page_log` は、CUPSが管理する印刷キューに送られた印刷ジョブの記録です。`error_log` には、印刷ジョブの失敗や、Webインターフェイスによるエラーが記録されます。

次に、CUPSサービスの管理に使用するツールとユーティリティについて説明します。

Webインターフェース

前述したのように、`/etc/cups/cupsd.conf` ファイルで、CUPSのWebインターフェースの有効・無効を設定します。次のようにオプションを指定します:

```
# Web interface setting...
WebInterface Yes
```

Webインターフェースを有効にすると、ブラウザからデフォルトURLの `http://localhost:631` にアクセスすることでCUPSを管理できます。デフォルトでは、ユーザーがプリンタと印刷キューを表示できますが、設定を変更する場合は、rootアクセス権を持つユーザーで認証する必要があります。管理機能へのアクセスを制限するためには、`/etc/cups/cupsd.conf` ファイルに次のスタンザを置きます。

```
# All administration operations require an administrator to authenticate...
<Limit CUPS-Add-Modify-Printer CUPS-Delete-Printer CUPS-Add-Modify-Class CUPS-Delete-Class CUPS-Set-Default>
  AuthType Default
  Require user @SYSTEM
  Order deny,allow
</Limit>
```

この設定を詳しく説明します:

`AuthType Default`
: 権限を必要とする場合に、Basic認証を使用します。

`Require user @SYSTEM`
: 管理者権限を持つユーザーのみが操作できることを示します。これを `@groupname` に変更すれば、グループ `groupname` のメンバーがCUPSサービスを管理できます。また、`Require user carol, tim` のようにユーザーのリストで管理者を指定することもできます。

`Order deny,allow`
: Apache 2の設定オプションと同様に、ユーザー（またはグループのメンバー）として認証されない限り、アクセスを拒否します。

CUPSのWebインターフェイスを無効にするには、まずCUPSサービスを停止して、`WebInterface` オ

プションを `Yes` から `No` に変更し、CUPSサービスを再起動します。

CUPSのWebインターフェースは、CUPSシステムの機能ごとにナビゲーションタブに分割された、シンプルなWebサイトです。 Webインターフェースのタブには、以下のものがあります:

Home（ホーム）
Homeタブには、インストールされているCUPSのバージョンが表示されます。また、次のようなセクションに整理したヘルプへのメニューを表示します。

CUPS for Users
CUPSの概要、コマンドラインからの印刷とオプションなど

CUPS for Administrator
プリンタの追加や管理インターフェイス、ネットワークプリンタの操作など

CUPS for Developer
CUPS自体の開発、プリンタ用PPDファイルの作成方法など

Administration（管理）
Administorationタブも、セクションに分かれています。

Printer
管理者は、新しい（ローカル）プリンタを追加したり、（ネットワーク）プリンタを見つけたり、インストール済みのプリンタを管理できます。

Class（クラス）
新しいクラスを作成および管理するためのセクションです。クラスとは、ポリシーグループとプリンタを関連付ける仕組みです。ある部署に属するユーザーだけが印刷できるクラスに、あるフロアにある一群のプリンタを追加するといった場合に使用します。あるいは、印刷できるページ数に制限を設けたクラスを設けることなどもできます。CUPSのインストール時にはクラスが定義されていないので、管理者が定義します。

Job（ジョブ）
管理者は、CUPSが管理するすべてのプリンタの、キューにあるすべての印刷ジョブを表示できます。

Server（サーバー）
管理者は、`/etc/cups/cupsd.conf` ファイルを変更することができます。また、プリンタをネットワーク上で共有する、高度な認証を利用する、リモートプリンタ管理を許可するといったオプションを選択することもできます。

Class（クラス）
プリンタクラスが定義されている場合には、このページに表示されます。プリンタクラス毎に、すべてのプリンタをまとめて管理するオプションと、クラスのプリンタキューにあるすべてのジョブを表示するオプションがあります。

Help（ヘルプ）
このタブには、そのシステムで利用可能なすべてのドキュメントへのリンクがあります。

Job（ジョブ）
このタブでは、印刷ジョブを検索したり、すべての印刷ジョブを一覧表示したりできます。

Printer（プリンタ）
このタブには、システムで利用可能なすべてのプリンタと、それらの概要ステータスが一覧表示されます。表示されているプリンタをクリックすると、それぞれのプリンタの管理ページに移動します。このタブのプリンタ情報は、`/etc/cups/printers.conf` ファイルから取得されます。

プリンタのインストール

CUPSのWebインターフェイスを使って、システムにプリンタキューを簡単に追加できます。

1. **Administration** タブをクリックし、**Add Printer** ボタンをクリックします。
2. 次のページでは、プリンタの接続方法に応じてさまざまなオプションが提示されます。ローカルプリンタの場合は、接続ポートや、インストール済みのプリンタソフトウェアなどが提示され、適当なオプションを選択します。また、CUPSはネットワークプリンタ検出して、ここに表示します。プリンタがサポートするネットワーク印刷プロトコルに応じて、直接接続するオプションを選択することもできます。適切なオプションを選択して **Continue** ボタンをクリックします。
3. 次のページでは、プリンタの名前、説明、設置場所（“バックオフィス” や “フロントデスク” など）を指定します。プリンタをネットワークで共有する場合は、このページでそのオプションを選択することもできます。入力したら **Continue** ボタンをクリックします。
4. 次のページでは、プリンタのメーカーとモデルを選択します。CUPSはデータベースを検索して、プリンタに適したドライバーとPPDファイルを探します。プリンタベンダーがPPDファイルを提供している場合は、その位置を指定します。指定が終わったら **Add Printer** ボタンをクリックします。
5. 最後のページでは、ページサイズや、印刷解像度などのデフォルトオプションを設定します。**Set Default Options** ボタンをクリックすると、プリンタがシステムにインストールされます。

NOTE

デスクトップ版Linuxの多くは、プリンタをセットアップするためのさまざまなツールを備えています。GNOMEとKDEデスクトップ環境には、プリンタのインストールと管理用の独自アプリケーションが組み込まれています。また、いくつかのディストリビューションでは、別のプリンタ管理アプリケーションが採用されています。しかしながら、印刷するユーザーが多いサーバーでは、CUPSのWebインタフェースが最適なツールとなることでしょう。

レガシーなLPD/LPRコマンドを使用して、プリンタのキューをセットアップすることもできます（訳注: 以下に取り上げるコマンドはCUPS以前の印刷システムに由来する　レガシー　コマンドです。コマンドラインでプリンタを設定・操作するために使われます。使用頻度は高くないでしょうが。コマンド名と役割くらいは覚えておきましょう）。`lpadmin` コマンドの例を示します：

```
$ sudo lpadmin -p ENVY-4510 -L "office" -v socket://192.168.150.25 -m everywhere
```

コマンドを分解して、それぞれのオプションを説明します。

- システムにプリンタを追加するには管理者権限が必要なので、`lpadmin`　コマンドの前に　`sudo`　を付けます。
- `-p`　オプションは印刷ジョブの宛先で、ユーザーに分かりやすい名前です。一般的には、プリンタ名を指定します。

- `-L` オプションでプリンタの位置を指定します。これはオプションですが、たくさんのプリンタが色々な場所にある場合に役立ちます。
- `-v` オプションには、プリンタデバイスのURIを指定します。CUPSは、レンダリングされたプリントジョブを、特定のプリンタに送信するためにデバイスURIを利用します。この例では、IPアドレスでネットワークでの送り先を指定しています。
- 最後のオプション `-m` には、“everywhere” を指定しています。このオプションにプリンタのモデルを指定することで、使用するPPDファイルが決定されます。現在のCUPSでは “everywhere” を指定して、CUPSにデバイスURI（前述の `-v` オプション）からプリンタ用の適切なPPDファイルを自動的に決定させるのが最適です。今では、CUPSが後で述べるIPPを利用するからです。

CUPSにプリンタに最適なPPDファイルを自動的に決定させるのがベストです。とはいえ、レガシーの `lpinfo` コマンドを使用してインストールされているPPDファイルを調べることもできます。`--make-and-model` オプションにセットアップしたいプリンタのモデル名と `-m` オプションを指定します:

```
$ lpinfo --make-and-model "HP Envy 4510" -m
hplip:0/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
hplip:1/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
hplip:2/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
drv:///hpcups.crv/hp-envy_4510_series.ppd HP Envy 4510 Series, hpcups 3.17.10
everywhere IPP Everywhere
```

ここでは、使用できるプリンタドライバを調べるために取り上げましたが、`lpinfo` コマンドの利用は推奨されていません。

WARNING

CUPSの将来のバージョンでは、プリンタドライバが廃止されて、代わりにIPP（Internet Printing Protocol）と、標準ファイルフォーマットに重点が置かれます。IPPは、プリントドライバと同じタスクを実現するもので、CUPSのWebインターフェースと同様に、TCPの631ポートを使用します。

印刷ジョブの大部分（またはすべて）を特定のプリンタに送る場合には、`lpoptions` コマンドでデフォルトのプリンタを指定することができます。`-d` オプションでデフォルトプリンタを指定します。

```
$ lpoptions -d ENVY-4510
```

プリンタの管理

プリンタをインストールすると、Webインターフェイスでプリンタのオプションを管理できますが、`lpadmin` コマンドを直接使用することもできます。

例えば、プリンタをネットワーク上で共有しましょう。`-p` オプションでプリンタを指定し、`printer-is-shared`（プリンタを共有する）オプションを有効化します。

```
$ sudo lpadmin -p FRONT-DESK -o printer-is-shared=true
```

印刷キューを、特定のユーザーからの印刷ジョブのみを受け入れるように構成することもできます。次のように、各ユーザーをコンマで区切って指定します。

```
$ sudo lpadmin -p FRONT-DESK -u allow:carol,frank,grace
```

逆に、特定のユーザーの印刷キューへのアクセス禁止することもできます。

```
$ sudo lpadmin -p FRONT-DESK -u deny:dave
```

ユーザーグループでプリンタのキューへのアクセスを許可/禁止するには、グループ名の前に “アットマーク”（`@`）を付けます。

```
$ sudo lpadmin -p FRONT-DESK -u deny:@sales,@marketing
```

ジョブの実行に問題が発生した場合の動作を、印刷キューのポリシーとして指定することができます。以下のポリシーを指定できます:

- エラー時にジョブを中断する（`abort-job`）
- 後でジョブの再実行を試みる（`retry-job`）
- エラー時にプリンタを停止する（`stop-printer`）
- 直ちにジョブを再実行する（`retry-current-job`）

以下に、`FRONT-DESK` プリンタでエラーが発生した場合に、印刷ジョブを中断するポリシーを指定する例を示します。

```
$ sudo lpadmin -p FRONT-DESK -o printer-error-policy=abort-job
```

このコマンドの詳細は、`lpadmin(8)` のmanページを参照してください。

印刷ジョブの送信

多くのデスクトップアプリケーションでは、メニューやキーボードショートカット `Ctrl`+`p` で印刷ジョブを送信できます。デスクトップ環境を利用しないLinuxシステムでは、レガシーなLPD/LPRコマンドでファイルをプリンタに送信します。

印刷ジョブをプリンタのキューに送信するには、`lpr`（“line printer remote”）コマンドを使用します。コマンドの最も基本的な使い方は、`lpr` コマンドにファイル名を指定するだけです。

```
$ lpr report.txt
```

上のコマンドは、ファイル `report.txt` を、システムのデフォルト印刷キュー（`/etc/cups/printers.conf` ファイルで指定します）に送信します。

CUPSに複数のプリンタがインストールされている場合は、`lpstat` コマンドに `p` オプションを指定すれば、利用可能なプリンタのリストが表示され、`d` オプションを指定すると、どのプリンタがデフォルトであるかが表示されます。

```
$ lpstat -p -d
printer FRONT-DESK is idle.  enabled since Mon 03 Aug 2020 10:33:07 AM EDT
printer PostScript_oc0303387803 disabled since Sat 07 Mar 2020 08:33:11 PM EST -
    reason unknown
printer ENVY-4510 is idle.  enabled since Fri 31 Jul 2020 10:08:31 AM EDT
system default destination: ENVY-4510
```

この例では、`report.txt` ファイルは、デフォルトとして設定されている `ENVY-4510` プリンタに送信されます。ファイルを別のプリンタで印刷する場合は、`-P` オプションでプリンタを指定します。

```
$ lpr -P FRONT-DESK report.txt
```

印刷ジョブがCUPSに送信されると、デーモンはそのタスクを処理するのに最も適しているバックエンドを判断します。CUPSは、さまざまなプリンタドライバ、フィルタ、ハードウェアポートモニタ、別のソフトウェアなどを使用して、文書を適切に印刷します。文書を どのように 印刷するかを調整したいことがあります。多くのグラフィックアプリケーションでは簡単なことですが、コマンドラインではオプションを指定します。`lpr` コマンドで印刷ジョブを送信する時に、`-o` オプションに所定の単語を指定して、文書のレイアウトを調整します。使用する単語を以下に示します:

`landscape`
用紙を横置き（横長）で印刷します。`orientation-requested=4` でも同じ結果が得られます。

`two-sided-long-edge`
縦置き（縦長）用紙に両面印刷します。（プリンタがこの機能をサポートしている場合）

`two-sided-short-edge`
横置き（横長）用紙に両面印刷します。（プリンタがこの機能をサポートしている場合）

`media`
用紙サイズを指定します。使用できる用紙サイズはプリンタによって異なりますが、一般的なサイズは次の通りです:

単語	用紙
`A4`	ISO A4
`Letter`	USレター
`Legal`	USリーガル
`DL`	ISO DL封筒
`COM10`	US #10封筒

`collate`
文書を複数部数印刷する場合に、部単位の印刷を指定します。`true` を指定すると部単位、`false` を指定するとページ単位になります。

`page-ranges`
印刷するページ範囲を指定します。1ページのみや、指定のページのみが印刷されます。例えば `-o`

`page-ranges=5-7,9,15` と指定すると、5、6、7ページが印刷され、続いて9ページと15ページが印刷されます。

`fit-to-page`
用紙サイズに合わせて文書を拡大/縮小して印刷します。文書ファイルにページサイズに関する情報が含まれていない場合は正しく拡大/縮小されずに、文書の一部が欠けたり、小さすぎることがあります。

`outputorder`
文書の印刷順序を `-o outputorder=normal` ないし `-o outputorder=reverse` で指定します。ページを下向きに排出するプリンタでは `normal` が、ページを上向きに排出するプリンタでは `reverse` がデフォルトです。

オプションの指定例を以下に示します:

```
$ lpr -P ACCOUNTING-LASERJET -o landscape -o media=A4 -o two-sided-short-edge finance-report.pdf
```

印刷部数を指定するには、#オプションを指定します。`-#N` と言う形式で、`N` には部数を置きます。デフォルトのプリンタで7部を印刷する例を示します。

```
$ lpr -#7 -o collate=true status-report.pdf
```

`lpr` コマンドとは別に、`lp` コマンドも使用できます。（訳注: `lpr` はBSD由来の、`lp` は System V由来のコマンドです）。`lpr` コマンドオプションの多くを `lp` コマンドでも使用できますが、いくつかの違いがあります。`lp(1)` のmanページを参照してください。以下に、`-d` オプションで宛先プリンタを指定し、`lp` コマンドで先の `lpr` コマンドと同様の出力を得る例を示します。

```
$ lp -d ACCOUNTING-LASERJET -n 7 -o collate=true status-report.pdf
```

印刷ジョブの管理

前述しましたが、印刷キューにジョブを送信すると、CUPSがジョブIDを表示します。`lpq` コマンドで、自分が送信した印刷ジョブを表示できます。`-a` オプションを指定すると、CUPSが管理しているすべてのプリンタのキューが表示されます。

```
$ lpq -a
Rank    Owner   Job     File(s)                 Total Size
1st     carol   20      finance-report.pdf      5072 bytes
```

前に紹介した `lpstat` コマンドにも、プリンタキューを表示するオプションがあります。`-o` オプションですべての印刷キューを表示しますが、印刷キューの名前を指定することもできます。

```
$ lpstat -o
ACCOUNTING-LASERJET-4                 carol       19456   Wed 05 Aug 2020 04:29:44 PM EDT
```

ジョブが送信されたキューの名前の末尾にジョブIDの数値が付加され、ジョブを送信したユーザー名、ファイルのサイズ、送信時刻が続きます。

プリンタで印刷ジョブが停止した場合、あるいは、印刷ジョブをキャンセルしたい場合は、`lpq` コマンドで調べたジョブIDを指定して、`lprm` コマンドを使用します。

```
$ lprm 20
```

印刷キュー内のすべてのジョブをキャンセルするには、`lprm` に `-`（ハイフン） のみを指定します。

```
$ lprm -
```

`cancel` コマンドを使えば、現在の印刷ジョブをキャンセルできます。

```
$ cancel
```

キャンセルするジョブを指定したいときは、プリンタ名を前に付けたジョブID（訳注: `lpstat` の出力形式）を指定します。

```
$ cancel ACCOUNTING-LASERJET-20
```

印刷ジョブを、ある印刷キューから別の印刷キューに移動することもできます。プリンタが応答しなくなったり、印刷する文書に別のプリンタの機能が必要な場合などに役立ちます。この処理には、プリンタ管理の特権が必要であることに注意してください。前の例と同じ方法で印刷ジョブを指定して、`FRONT-DESK` プリンタのキューに移動する例を示します。

```
$ sudo lpmove ACCOUNTING-LASERJET-20 FRONT-DESK
```

プリンタの削除

プリンタを削除する前に、CUPSサービスが管理しているすべてのプリンタをリストアップすると便利です。`lpstat` コマンドを使います。

```
$ lpstat -v
device for FRONT-DESK: socket://192.168.150.24
device for ENVY-4510: socket://192.168.150.25
device for PostScript_oc0303387803: ///dev/null
```

`-v` オプションを指定すると、プリンタだけではなく、接続されている位置（と方法）も表示されます。まず、ユーザーがプリンタに新しいジョブを送信することを禁止して、プリンタが新しいジョブを受け入れない理由が提示されるようにすると良いでしょう。次の方法で行います:

```
$ sudo cupsreject -r "Printer to be removed" FRONT-DESK
```

このタスクにはプリンタ管理の特権が必要ですから、`sudo` を使用します。

プリンタを削除するには、`lpadmin` コマンドに `-x` オプションを指定します。

```
$ sudo lpadmin -x FRONT-DESK
```

演習

1. `office-mgr` という名前のワークステーションに新しいプリンタをインストールしました。このワークステーションのデフォルトとして、そのプリンタを設定するコマンドはどうなりますか？

2. ワークステーションで使用できるプリンタを表示するコマンドはどうなりますか？

3. `office-mgr` という名前のプリンタのキューにある、IDが15の印刷ジョブを削除する `cancel` コマンドはどうなりますか？

4. `FRONT-DESK` プリンタにキューに送ったIDが2の印刷ジョブは、印刷を完了するには用紙が足りないことに気付きました。このジョブを、`ACCOUNTING-LASERJET` プリンタの印刷キューに移動するコマンドはどうなりますか？

発展演習

実際のプリンタをインストールせずに、CUPSプリンタ管理の実習を行います。Debian系のディストリビューションでは `printer-driver-cups-pdf` パッケージを、Red Hat系のディストリビューションでは `cups-pdf` パッケージをインストールします（訳注: いずれもいわゆる PDFプリンタ を提供するパッケージです)。また、`cups-client` パッケージをインストールして、System Vスタイルの印刷コマンドを使用します。

1. CUPSデーモンが実行されていること、PDFプリンタが有効で、デフォルトに設定されていることを確認します。

2. `/etc/services` ファイルを印刷して、ホームディレクトリの `PDF` という名前のディレクトリにPDFを置くコマンドを実行します。

3. プリンタを無効化するコマンドを実行します。次いで、すべてのステータス情報を表示するコマンドを実行して、PDFプリンタが無効になっていることを確認します。

4. 今度は、`/etc/fstab` ファイルをPDFプリンタに印刷します。何が起こりますか？

5. 印刷ジョブをキャンセルしてから、PDFプリンタを削除します。

まとめ

CUPSデーモンは、ローカルおよびリモートのプリンタに印刷するために広く使用されているプラットフォームです。従来のLPDプロトコルに取って代わったものですが、ツールには互換性があります。

このレッスンで説明したファイルとコマンドは次のとおりです:

`/etc/cups/cupsd.conf`
CUPSサービス本体の設定ファイル。CUPSのWebインターフェイスへのアクセスも制御します。

`/etc/printcap`
LPDによって使用されていたレガシーファイルで、システムに接続されているプリンタの情報を行ごとに保持しています。

`/etc/cups/printers.conf`
プリンタ情報を保持するCUPSの設定ファイル。

CUPSのWebインターフェースは、デフォルトで `http://localhost:631` にあります。ポート番号が631/TCPであることに注意してください。

以下に示す、レガシーなLPD/LPRコマンドについても説明しました。

`lpadmin`
プリンタとプリンタクラスのインストールと削除を行う。

`lpoptions`
プリンタオプションを表示、および設定する。

`lpstat`
接続されているプリンタのステータス情報を表示する。

`lpr`
印刷ジョブをプリンタのキューに送信する。

`lp`
印刷ジョブをプリンタのキューに送信する。

`lpq`
印刷キューにある印刷ジョブを一覧表示します。

`lprm`
ジョブIDを指定して印刷ジョブをキャンセルする。ジョブIDは、`lpq` コマンドで取得できる。

`cancel`
ジョブIDを指定して印刷ジョブをキャンセルする。`lprm` コマンドと同様。

CUPSのさまざまなツールとユーティリティについては、次のマニュアルページを確認してください: `lpadmin(8)`、`lpoptions(1)`、`lpr(1)`、`lpq(1)`、`lprm(1)`、`cancel(1)`、`lpstat(1)`、`cupsenable(8)`、`cupsaccept(8)`。`http://localhost:631/help` にあるオンラインヘルプのドキュメントも有用です。

演習の解答

1. `office-mgr` という名前のワークステーションに新しいプリンタをインストールしました。このワークステーションのデフォルトとして、そのプリンタを設定するコマンドはどうなりますか？

```
$ lpoptions -d office-mgr
```

2. ワークステーションで使用できるプリンタを表示するコマンドはどうなりますか？

```
$ lpstat -p
```

`-p` オプションは、使用可能なすべてのプリンタと、それらが印刷可能かどうかを一覧表示します。

3. `office-mgr` という名前のプリンタのキューにある、IDが15の印刷ジョブを削除する `cancel` コマンドはどうなりますか？

```
$ cancel office-mgr-15
```

4. `FRONT-DESK` プリンタにキューに送ったIDが2の印刷ジョブは、印刷を完了するには用紙が足りないことに気付きました。このジョブを、`ACCOUNTING-LASERJET` プリンタの印刷キューに移動するコマンドはどうなりますか？

```
$ sudo lpmove FRONT-DESK-2 ACCOUNTING-LASERJET
```

発展演習の解答

実際のプリンタをインストールせずに、CUPSプリンタ管理の実習を行います。Debian系のディストリビューションでは `printer-driver-cups-pdf` パッケージを、Red Hat系のディストリビューションでは `cups-pdf` パッケージをインストールします（訳注: いずれもいわゆる PDFプリンタ を提供するパッケージです)。また、`cups-client` パッケージをインストールして、System Vスタイルの印刷コマンドを使用します。

1. CUPSデーモンが実行されていること、PDFプリンタが有効で、デフォルトに設定されていることを確認します。

   PDFプリンタのステータスを確認する方法の1つに、次のコマンドがあります。

   ```
   $ lpstat -p -d
   printer PDF is idle.  enabled since Thu 25 Jun 2020 02:36:07 PM EDTi
   system default destination: PDF
   ```

2. `/etc/services` ファイルを印刷して、ホームディレクトリの `PDF` というディレクトリにPDFを置くコマンドを実行します。

   ```
   $ lp -d PDF /etc/services
   ```

   これによって、PDFディレクトリ内にPDFファイルが作成されます。

3. プリンタを無効化するコマンドを実行します。次いで、すべてのステータス情報を表示するコマンドを実行して、PDFプリンタが無効になっていることを確認します。

   ```
   $ sudo cupsdisable PDF
   ```

   PDFプリンタが無効になりました。

   プリンタのステータスをすべて表示する `lpstat -t` コマンドを実行します。次のようになります。

   ```
   $ lpstat -t
   scheduler is running
   system default destination: PDF
   device for PDF: cups-pdf:/
   PDF accepting requests since Wed 05 Aug 2020 04:19:15 PM EDT
   printer PDF disabled since Wed 05 Aug 2020 04:19:15 PM EDT -
   Paused
   ```

4. 今度は `/etc/fstab` ファイルをPDFプリンタに印刷します。何が起こりますか？

   コマンド `lp -d PDF /etc/fstab` を実行すると、ジョブIDが表示されます。しかし、ホームディレクトリのPDFフォルダを確認しても、新しいファイルはありません。`lpstat -o` コマンドで印刷キューを確認すると、ジョブが一覧表示されます。

5. 印刷ジョブをキャンセルしてから、PDFプリンタを削除します。`

   `lp` コマンドが出力したIDを使用して、`cancel` コマンドでジョブを削除します。例を示します。

   ```
   $ cancel PDF-4
   ```

   再度 `lpstat -o` コマンドを実行して、ジョブが削除されたことを確認します。

   `sudo lpadmin -x PDF` コマンドでPDFプリンタを削除します。`lpstat -a` で、プリンタが削除されたことを確認します。

課題 109: ネットワークの基礎

109.1 インターネットプロトコルの基礎

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 109.1

総重量

4

主な知識分野

- ネットワークマスクとCIDR表記の理解を示す。
- プライベートとパブリック "ドット付きクワッド" IPアドレスの違いについての知識。
- 一般的なTCPおよびUDPポートとサービスに関する知識(20, 21, 22, 23, 25, 53, 80, 110, 123, 139, 143, 161, 162, 389, 443, 465, 514, 636, 993, 995)。
- UDP、TCP、ICMPの違いと主要な特徴についての知識。
- IPv4とIPv6の主な違いの知識。
- IPv6の基本機能に関する知識。

用語とユーティリティ

- `/etc/services`
- IPv4, IPv6
- Subnetting
- TCP, UDP, ICMP

109.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.1 インターネットプロトコル（IP）の基礎
Lesson:	1 of 2

はじめに

TCP/IP（Transmission Control Protocol/Internet Protocol）は、インターネットを含むコンピュータネットワークで標準的に利用されているプロトコルのスタック、すなわち インターネット プロトコル スイート（Internet protocol suite）です。プロトコルスタックは、IP、TCP、UDP、ICMP、DNS、SMTP、ARPなど、複数のプロトコルで構成されています。

IP（インターネットプロトコル）

IPは、ホストの論理的なアドレスを指定するプロトコルであり、あるホストから別のホストにパケットを送信する役割を担います。具体的には、ネットワーク上の1つのデバイスには少なくと1つのIPアドレスが割り当てられますが、複数のIPアドレスが割り当てられる事もあります（訳注：多くの場合は、1つの NIC（Network Interface Controller） ごとに1つのIPアドレスを割り当てます）。

IPプロトコルバージョン4（IPv4）では、32ビットで1つのIPアドレスを表現します。32ビットの数値を、8ビットごとに4つに区切った “dotted quad”（ドットで区切った4個組み）と呼ばれる10進数で表記します。例えば:

2進数形式（8ビット4つ）

```
11000000.10101000.00001010.00010100
```

10進数形式

```
192.168.10.20
```

それぞれのオクテット（8ビット）の値は、0〜255（バイナリ形式の11111111）の範囲になります。

アドレスクラス

IPアドレスの先頭数ビットのパターンに応じてアドレス範囲を　クラス　に分類することがあります。現在ではほとんど使われないのですが、特別な役割を持つアドレス範囲を理解する助けになりますので、ここに示しておきます。

クラス	第1オクテット	範囲	例
A	1-126（先頭1ビットが0）	`1.0.0.0` – `126.255.255.255`	`10.25.13.10`
B	128-191（先頭2ビットが10）	`128.0.0.0` – `191.255.255.255`	`141.150.200.1`
C	192-223（先頭3ビットが110）	`192.0.0.0` – `223.255.255.255`	`200.178.12.242`

パブリックIPアドレスとプライベートIPアドレス

通信を行うためには、ネットワーク上の機器それぞれが少なくとも1つのユニークなIPアドレスを持つ必要があります。しかしながら、インターネットに接続する世界中の機器それぞれにユニークなIPアドレスを与えるには、IP（v4）のアドレス範囲では不十分です。そこでプライベートなIPアドレスが定義されました。

プライベートIPアドレスとは、企業や組織、家庭などの　閉じた内部（プライベート）　ネットワークで使用するために予約されているIPアドレスの範囲です。プライベートネットワーク内ではプライベートアドレスを使用し、プライベートネットワークとインターネットの境界で、公式なパブリックIPアドレスに変換します。

つまり、インターネット上では（公式な、すなわちルーティングされる）パブリックIPアドレスを使用してやり取りを行いますが、プライベートネットワーク内では（予約された範囲内の）プライベートIPアドレスを使用します。境界ルーターでは　NAT（Network　Address　Translation）　ないし　NAPT（Network Address and Port Translation）　と呼ばれる機能を有効化し、プライベートネットワークとパブリックネットワークの間で（IPアドレスを含む）トラフィックの変換を行います。（NAPT　のことを マスカレード（Masquerade）　と呼ぶこともあります。）

プライベートIPアドレスの範囲は、次の表に示すように、クラスごとに決められています:

クラス	第1オクテット	範囲	プライベートIP
A	1-126	`1.0.0.0` – `126.255.255.255`	`10.0.0.0` – `10.255.255.255`
B	128-191	`128.0.0.0` – `191.255.255.255`	`172.16.0.0` – `172.31.255.255`
C	192-223	`192.0.0.0` – `223.255.255.255`	`192.168.0.0` – `192.168.255.255`

10進数から2進数への変換

ここで、IPアドレスを2進数形式と10進数形式の間で変換する方法を解説しておきましょう。

10進数から2進数への変換は、2による除算を繰り返すことで行います。例として、10進数の105を2進数に変換してみましょう:

1. 105を2で割ると、次のようになります。

```
105/2
商 = 52
余り = 1
```

2. 商が1になるまで、商を2で除算し続けます。

```
52/2
商 = 26
余り = 0
```

```
26/2
商 = 13
余り = 0
```

```
13/2
商 = 6
余り = 1
```

```
6/2
商 = 3
余り = 0
```

```
3/2
商 = 1
余り = 1
```

3. 最後の商（1）の後に、すべての余りを並べていきます：

```
1101001
```

4. 8ビットが揃うまで、左側に0を置きます。

```
01101001
```

5. 以上で、10進数の105は、2進数では 01101001 になることがわかります。

2進数から10進数への変換

2進数 `10110000` を例に取りましょう。

1. 各ビットは、2を底とする累乗（冪:べき）に関連付けられています。累乗の指数（冪指数）は0から始まり、桁上がりの度に増加します。この例では、次のようになります。

1	0	1	1	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

2. ビットが1のときにはそれぞれの累乗の値を割り当て、ビットが0のときは0を割り当てます。

1	0	1	1	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	0	32	16	0	0	0	0

3. すべての値を合計します。

   128 + 32 + 16 = 176

4. つまり、2進数の10110000は、10進数の176になります。

ネットマスク

IPアドレスには2つの意味が含まれていて、上位ビットがネットワークを、下位ビットが（そのネットワークにおける）各ホストを表します。ネットマスク（netmask）は、ネットワークを表すビットを、IPアドレスと同じ形式（8ビットが4つで32ビット）で示すものです。例を示します。

10進数	2進数	CIDR
`255.0.0.0`	`11111111.00000000.00000000.00000000`	`8`
`255.255.0.0`	`11111111.11111111.00000000.00000000`	`16`
`255.255.255.0`	`11111111.11111111.11111111.00000000`	`24`

ネットマスク `255.255.0.0` を例にとると、前半の16ビット（2つの10進数）がネットワーク/サブネットを示し、後半の16ビットがそのネットワーク内のホストを示します。

表にあるCIDR（Classless Inter-Domain Routing/サイダーと読む）というのは、ネットワークを示すビット数がいくつかであるかを示す数値で、ネットマスクを簡単に示したものです。たとえば、`255.255.255.0` を、`/ 24` と示します。（訳注: 日本では プレフィックス長、CIDR表記 などと呼びます。）

歴史的には、アドレスクラスごとにネットマスクが決められていました。IPv4アドレスの利用効率を高めるためにCIDRの概念が登場して任意長のプリフィックス長が使えるようになってからは、アドレスクラスの概念はほぼ使われなくなっています。

クラス	最初のオクテット	範囲	デフォルトマスク
A	1-126	`1.0.0.0` – `126.255.255.255`	`255.0.0.0 / 8`
B	128-191	`128.0.0.0` – `191.255.255.255`	`255.255.0.0 / 16`
C	192-223	`192.0.0.0` – `223.255.255.255`	`255.255.255.0 / 24`

なおこれらのパターンは、あくまでも「デフォルトの」ネットマスクを示しているだけです。実際には次に示すように、どのIPアドレスでも任意のマスクを使用することができます。

IPアドレスとネットマスク、マスク長の使用例を示します:

```
192.168.8.12 / 255.255.255.128 / 24
```

範囲
`192.168.8.0` - `192.168.8.255`

ネットワークアドレス
`192.168.8.0`

ブロードキャストアドレス
`192.168.8.255`

ホスト
`192.168.8.1` - `192.168.8.254`

この場合、IPアドレスの上位24ビット（3つの10進数）でネットワークを定義し、下位8ビット（最後の10進数）がホストを指定するアドレスになります。つまり、このネットワークの範囲は `192.168.8.0` から `192.168.8.255` になります。

重要なポイントが2つあります: ネットワーク/サブネットごとに2つの予約アドレスがあります。範囲内の最初のアドレス（この場合は `192.168.8.0`）を ネットワークアドレス と呼び、ネットワーク/サブネットそのものを示します。範囲内の最後のアドレス（この場合は `192.168.8.255`）を ブロードキャストアドレス と呼び、ネットワーク/サブネット内の全てのホストに同じメッセージ（パケット）を同報するために使われます。

ネットワークアドレスとブロードキャストアドレスは、ネットワーク上のホストマシンに割り当てることができません。したがって、実際に使用できるIPアドレスは、`192.168.8.1` から `192.168.8.254` の範囲です。

同じIPアドレスですが、ネットマスクが異なる場合を見てみましょう。

```
192.168.8.12 / 255.255.252.0 / 22
```

範囲
`192.168.8.0` - `192.168.11.255`

ネットワークアドレス
`192.168.8.0`

ブロードキャストアドレス
`192.168.11.255`

ホスト
`192.168.8.1` – `192.168.11.254`

ネットマスクの変更による、ネットワーク/サブネット内のIPアドレス範囲の変化に着目してください。

2番目の例に示したように、ネットマスクによるネットワークの分割は、デフォルト値（8、16、24）に限られません。必要に応じて、ネットワーク部分のビットを追加・削除して、サブネットの大きさ（収納できるホストの台数）を調整することができます。

例えば次のようになります。

```
11111111.11111111.11111100.00000000 = 255.255.252.0 = 22
```

このネットワークを2つに分割する場合は、次のように、ネットマスクのネットワーク部のビットを追加します。

```
11111111.11111111.11111110.00000000 = 255.255.254.0 = 23
```

これにより、次の2つのサブネットができます。

```
192.168.8.0   - 192.168.9.255
192.168.10.0 - 192.168.11.255
```

さらにサブネットに分割したければ、次のようにします:

```
11111111.11111111.11111111.00000000 = 255.255.255.0 = 24
```

以下の4つのサブネットに分かれます:

```
192.168.8.0  - 192.168.8.255
192.168.9.0  - 192.168.9.255
192.168.10.0 - 192.168.10.255
192.168.11.0 - 192.168.11.255
```

それぞれのサブネットには、ネットワーク自体（範囲の最初）とブロードキャスト（範囲の最後）の予約アドレスが取られるので、ネットワークを細分化するほどに、ホストに使用できるIPアドレスが少な

くなります。

ネットワークアドレスとブロードキャストアドレスを見つける

IPアドレスとネットマスクから、ネットワークアドレスとブロードキャストアドレスを見つけて、ネットワーク/サブネットにおけるホストのIPアドレス範囲を知ることができます。

ネットワークアドレスは、IPアドレスとネットマスクで、ビット単位の “論理積（AND）” を求めることで取得できますます。IPアドレス `192.168.8.12` と、ネットマスク `255.255.255.192` の例を見てみましょう。

前述の方法で、10進数から2進数に変換すると、次のようになります。

```
11000000.10101000.00001000.00001100 (192.168.8.12)
11111111.11111111.11111111.11000000 (255.255.255.192)
```

“論理積（AND）” では、1&1=1、0&0=0、1&0=0なので、次のようになります。

```
11000000.10101000.00001000.00001100 (192.168.8.12)
11111111.11111111.11111111.11000000 (255.255.255.192)
11000000.10101000.00001000.00000000
```

したがって、このサブネットのネットワークアドレスは `192.168.8.0` です。

次にブロードキャストアドレスを求めてみましょう。IPアドレスのホスト部のビットをすべて1にします。

```
11000000.10101000.00001000.00000000 (192.168.8.0)
11111111.11111111.11111111.11000000 (255.255.255.192)
11000000.10101000.00001000.00111111
```

この場合のブロードキャストアドレスは `192.168.8.63` になります。

まとめると、次のようになります。

```
192.168.8.12 / 255.255.255.192（ネットマスク） / 26（プレフィックス長）
```

範囲
`192.168.8.0` - `192.168.8.63`

ネットワークアドレス
`192.168.8.0`

ブロードキャストアドレス
`192.168.8.63`

ホスト
`192.168.8.1` – `192.168.8.62`

デフォルトルート

IPパケットを直接やり取りできるのは、同じネットワーク/サブネット内にあるマシンに限られます。

次の例ではどうでしょう。

ネットワーク1
`192.168.10.0/24`

ネットワーク2
`192.168.200.0/24`

この場合、`192.168.10.20` のマシンは、異なる論理ネットワーク上にある `192.168.200.100` のマシンにパケットを直接送信することができません。

この2台のホストが通信するためには、ルーター（またはルーター群）が必要です。ルーターは2つのネットワーク間を橋渡しするので、ゲートウェイとも呼ばれます。ルータは複数のネットワークインターフェイスを持ち、それぞれのネットワークにアクセスできるようにIPアドレスが構成されます。例えば、`192.168.10.1` と `192.168.200.1` です。これで、ルーターがネットワークをまたぐ通信を中継できます。

ネットワーク上のそれぞれのホストでは、デフォルトルート を構成します。デフォルトルートとは、それぞれのホストが、「自分とは異なるネットワークに宛てたパケット」を送出するIPアドレスです。

上の例では、`192.168.10.0/24` ネットワーク上のマシンのデフォルトルートがルーター/ゲートウェイのアドレスである `192.168.10.1` に、`192.168.200.0/24` 上のマシンのデフォルトルートが `192.168.200.1` になります。

デフォルトルートは、プライベートネットワーク（LAN）上のマシンが、ルーターを介してインターネット（WAN）にアクセスするためにも使用されます。

演習

1. IPアドレスが `172.16.30.230`、ネットマスクが `255.255.255.224` の場合に、以下はどうなりますか？

ネットマスクのCIDR表記	
ネットワークアドレス	
ブロードキャストアドレス	
このサブネットでホストに使用できるIPの数	

2. 異なる論理ネットワークのホストとIPで通信する場合に、ホストで設定が必要となるのは何ですか？

発展演習

1. IPアドレスのクラス（クラスA～C）に、`127` から始まるアドレス範囲と、`224` 以降のアドレス範囲が含まれないのはなぜですか？

2. IPパケットに含まれる非常に重要なフィールドの1つにTTL（Time To Live）があります。このフィールドの役割と仕組みを述べてください。

3. NATの機能とそれを使用する場面について説明してください。

まとめ

このレッスンでは、ネットワークに接続されたホストの間の通信を司る、IPv4プロトコルの主な概念を説明しました。

IPアドレスをさまざまな形式で表現して、ネットワークとサブネットの論理構成を分析したり構築するために、エンジニアが知っておくべき知識も紹介しました。

このレッスンでは、以下の事柄を取り上げました。

- IPアドレスのクラス
- パブリックIPアドレスと、プライベートIPアドレス
- IPアドレスを10進数表記と2進数表記の間で相互に変換する方法
- ネットワークマスク（ネットマスク）
- IPアドレスとネットマスクから、ネットワークアドレスとブロードキャストアドレスを知る方法
- デフォルトルート

演習の解答

1. IPアドレスが `172.16.30.230`、ネットマスクが `255.255.255.224` の場合に、以下はどうなりますか？

ネットマスクのCIDR表記	`27`
ネットワークアドレス	`172.16.30.224`
ブロードキャストアドレス	`172.16.30.255`
このサブネットでホストに使用できるIPの数	`30`

2. 異なる論理ネットワークのホストとIPで通信する場合に、ホストで設定が必要となるのは何ですか？

   デフォルトルート

発展演習の解答

1. IPアドレスのクラス（クラスA～C）に、`127` から始まるアドレス範囲と、`224` 以降のアドレス範囲が含まれないのはなぜですか？

   `127` で始まる範囲（127.0.0.0/8）はループバックアドレス用に予約されていて、`127.0.0.1` などアドレスは、ホスト内でのテストやプロセス間通信に使用されます。また、`224` から始まる範囲（224.0.0.0/4）は、マルチキャストなどの用途に予約されていて、ホストアドレスとしては使用されません。

2. IPパケットに含まれる非常に重要なフィールドの1つにTTL（Time To Live）があります。このフィールドの役割と仕組みを述べてください。

   TTLは、パケットの寿命を定義します。送信元で定義された初期値が、パケットがゲートウェイ/ルーターを通過する度（“ホップ” と呼びます）にデクリメントされます。このカウンタが0になると、パケットは破棄されます。

3. NATの機能とそれを使用する場面について説明してください。

   NAT（Network Address Translation）機能は、プライベートアドレスを使用しているローカルネットワーク上の機器のアドレスを、ゲートウェイ（ルーター）でルーターに割り当てられているパブリックIPアドレスに変換することで、インターネットに直接アクセスしているかのようにするものです。

109.1 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.1 インターネットプロトコルの基礎
Lesson:	2 of 2

はじめに

前のレッスンでは、TCP/IPスタック（インターネット　プロトコル　スイート）の中からIPプロトコルを取り上げて、IPアドレス（IPv4）、ネットマスク、デフォルトルートなどについて学習しました。

IPはネットーワーク層のプロトコルであり、「ホスト間」のデータのやり取りを司ります。1つのホストではたくさんのサービスやアプリケーションが動作しますから、TCPやUDPなどのトランスポート層のプロトコルで、利用するサービスやアプリケーションを指定することが必要です。

TCPならびにUDPでは、ホストで稼働しているサービスやアプリケーションを識別するために、ネットワークポート　という概念を使用します。つまり、あるプログラムが別のホストで稼働しているサービスと通信する場合には、IPアドレスでホストを指定し、さらに　ポート番号　でサービスを指定します。もちろん、発信側のプログラムにもポート番号が割り当てられます。

ポート番号は16ビットの数値ですから、0を除く65,535個までのポートをそれぞれのホストが使用できます。ただし1023以下のポート番号は　特権ポート　と呼ばれ、root権限を持つプロセスのみが使用できます。発信側のプログラムでは、非特権ポート　ないしソケットポートと呼ばれる1024から65,535までのポート範囲を使用します。

受信側であるサービスが使用するポート番号は、サービスの種類ごとにIANA（Internet　Assigned Numbers　Authority）によって標準化されています。つまり（特別な事情がない限り）どのシステムでも、SSHサービスは22番ポートを、HTTPサービスは80番ポートを使用します。

次の表に、主なサービスとそれぞれのポートを示します。

ポート	サービス
20	FTP (データ転送用)
21	FTP (制御用)
22	SSH (Secure Socket Shell)
23	Telnet (暗号化なしのリモート接続)
25	SMTP (Simple Mail Transfer Protocol) メールの送信
53	DNS (Domain Name System)
80	HTTP (Hypertext Transfer Protocol)
110	POP3 (Post Office Protocol) メールの受信
123	NTP (Network Time Protocol)
139	Netbios
143	IMAP (Internet Message Access Protocol) メールへのアクセス
161	SNMP (Simple Network Management Protocol)
162	SNMPTRAP、SNMP通知
389	LDAP (Lightweight Directory Access Protocol)
443	HTTPS (Secure HTTP)
465	SMTPS (Secure SMTP)
514	RSH (Remote Shell)
636	LDAPS (Secure LDAP)
993	IMAPS (Secure IMAP)
995	POP3S (Secure POP3)

Linuxシステムでは `/etc/services` ファイルに、サービスごとのポート番号がリストされています。

接続の際に宛先のサービスを指定するには、IPアドレスの後に文字 `:`（コロン）で区切り、ポート番号を置きます。たとえば、IPホスト `200.216.10.15` で稼働しているHTTPSサービスにアクセスする場合は、宛先として `200.216.10.15:443` を指定します。

それぞれのサービスは、その働きに応じたトランスポートを使用します。ほとんどの場合は、TCPかUDPのいずれかです。

Transmission Control Protocol（伝送制御プロトコル、TCP）

TCPは、コネクション型のトランスポートプロトコルです。クライアントのソケットポートからサーバーの標準ポートに宛てて、接続が確立されます。TCPプロトコルでは、ネットワークエラーのために失われたパケットの再送信を行うなどの機能が組み込まれていて、パケットの整合性や到着順序が保証されています。

したがって、TCPプロトコルを使用するアプリケーションでは、データフロー制御をOSに任せればよく、アプリケーションで制御する必要はありません。

User Datagram Protocol (ユーザーデータグラム、UDP)

UDPはコネクションレス型のトランスポートプロトコルです。クライアントのソケットポートとサーバーの標準ポートの間でパケットの送受信が行われますが、接続の状態やデータの送受信は一切制御されません。つまり、パケットが紛失したり壊れていたとしても、OSは何もしてくれません。アプリケーション側で、必要な制御をすべて実装する必要があります。

その分、UDPのデータフローは高速です。ある種のサービスにとっては、とても重要な特徴です。

Internet Control Message Protocol （インターネット制御通知プロトコル、ICMP）

ICMPは、IPと同じネットワーク層プロトコルです。次に示すような、ネットワーク自体を分析ならびに制御するために使われます。

- トラフィック量の制御
- 到達不能な宛先の検出
- 経路のリダイレクト
- リモートホストのステータス確認

ICMPは `ping` コマンドで使用されるプロトコルであり、別のレッスンで取り上げます。

IPv6

ここまでに、IPプロトコルのバージョン4、つまりIPv4について学習してきました。これは、ほぼすべてのネットワークおよびインターネット環境で使用される標準バージョンです。しかし、使用可能なIPv4アドレスが足りないことや、さまざまな種類のデバイスをインターネットに接続しておきたいというニーズ（IoTなど）から、IPプロトコルのバージョン6を使用することがますます一般的になっています。通常、IPv6と記述されています。

IPv6では、さまざまな機能が変更ないし追加されただけではなく、IPアドレス自体の形式も変更されています。

IPv6アドレスは128ビット（IPv4の4倍長）の数値です。16ビットずつ8つのグループに分け、コロン `:` で区切って16進数で表現します。

例えば次のようになります。

```
2001:0db8:85a3:08d3:1319:8a2e:0370:7344
```

省略形

IPv6アドレスは長い（最長39文字）ので、状況に応じて一部を省略して短く表現する方法が定義されています。次のアドレス例を見てみましょう。

```
2001:0db8:85a3:0000:0000:0000:0000:7344
```

最初のルールは、16進数の上位の `0` を省略することです。`0000` は1つの `0` に短縮できます。

```
2001:db8:85a3:0:0:0:0:7344
```

次のルールは、連続する `0` ブロックを、1つの `::` にまとめることです。`0:0:0:0` は、次のように1つの `::` に短縮できます。

```
2001:db8:85a3::7344
```

ただし、この `::` による省略は、1箇所のみに限られます。最も長い部分を置き換えるのが普通です。

```
2001:db8:85a3:0000:0000:1319:0000:7344

2001:db8:85a3:0:0:1319:0:7344

2001:db8:85a3::1319:0:7344
```

IPv6のアドレス種別

IPv6には、次の3種類のアドレスがあります。

ユニキャスト
: 1つのネットワークインターフェイスを表します。デフォルトでは、上位64ビットはネットワーク（サブネットプレフィックスないしネットワークID）を表し、下位64ビットはネットワークインターフェイス（インターフェイスID）を表します。

マルチキャスト
: 複数のノードに割り当てられるアドレスです。マルチキャストアドレスに送信されたパケットは、そのグループに属する すべての インターフェイスに送信されます。なお、IPv4のブロードキャストに相当するアドレスはIPv6にはありません。

エニーキャスト
: 複数のノードに割り当てられるアドレスですが、マルチキャストアドレスがすべてのノードに送られるのに対して、エニーキャストアドレスは いずれかの ノード（通常はネットワーク的に最も近いもの）に送られます。

IPv4とIPv6の違い

アドレスの違いだけではなく、IPバージョン4と6の間には他にもいくつもの違いがあります。重要ないくつかを紹介します。

- ポート番号に違いはありません。ただし、宛先のIPアドレスとポート番号をセットで表現する場合に、IPv6ではIPアドレスを []（角かっこ）で囲みます。（他にもIPv6アドレスであることを明示するために、アドレス全体を [] で囲むことがあります）。

  IPv4
  `200.216.10.15:443`

IPv6

```
[2001:db8:85a3:8d3:1319:8a2e:370:7344]:443
```

- IPv6には、IPv4にあるようなブロードキャスト機能がありません。ただし、リンクローカルアドレスの一種である `ff02::1` を使用して、同じ結果を得ることができます。IPv4で宛先にマルチキャストアドレス `224.0.0.1` を使用するのと同様に、ローカルネットワーク上のすべてのホストにパケットが到着します。
- IPアドレスの自動設定機能。SLAAC（Stateless Address Autoconfiguration）と呼ばれる機能により、IPv6ホストはIPアドレスを自動構成できます。DHCPv6を利用することもできます。
- IPv4パケットのTTL（Time to Live）フィールドは、IPv6ヘッダーの“HopLimit”に置き換えられました。働きは同じです。
- すべてのIPv6インターフェースには、リンクローカルアドレスと呼ばれるローカルアドレスが自動的に割り当てられます。プレフィックスは `fe80::/10` です。
- IPv4におけるARPに代わり、IPv6では Neighbor Discovery Protocol（近隣探索プロトコル、NDP）を使用します。NDPはARPよりも広範囲なネットワーク制御を行います。

演習

1. SMTPプロトコルのデフォルトは何番ポートですか？

2. システムが使用できるポート数はいくつですか？

3. すべてのパケットが正しい順序で間違いなく配信されることが保証されているトランスポート層のプロトコルは何ですか？

4. IPv6で、グループに属するすべてのインターフェイスにパケットを送信するためのパケット種別は何ですか？

発展演習

1. デフォルトでTCPプロトコルを使用するサービスの例を4つ挙げてください。

2. IPv4におけるTTLと同じ働きをする、IPv6ヘッダーのフィールドは何ですか？

3. Neighbor Discovery Protocol（NDP）が扱う情報にはどのようなものがありますか？

まとめ

このレッスンでは、TCP/IPプロトコルスタックで使用されるトランスポートプロトコルとサービスについて説明しました。

もう1つの重要なトピックとして、アドレスの違いを含むIPv6とIPv4の主な違いを説明しました。

このレッスンでは、以下の事柄を取り上げました。

- ポート番号とサービスの関係
- TCP（Transmission Control Protocol）
- UDP（User Datagram Protocol）
- ICMP（Internet Control Message Protocol）
- IPv6アドレスとその省略表記方法
- IPv6アドレス種別
- IPv4とIPv6の主な違い

演習の解答

1. SMTPプロトコルのデフォルトは何番ポートですか？

   25

2. システムが使用できるポート数はいくつですか？

   65535

3. すべてのパケットが正しい順序で間違いなく配信されることが保証されているトランスポート層のプロトコルは何ですか？

   TCP

4. IPv6で、グループに属するすべてのインターフェイスにパケットを送信するためのパケット種別は何ですか？

   マルチキャスト

発展演習の解答

1. デフォルトでTCPプロトコルを使用するサービスの例を4つ挙げてください。

   FTP、SMTP、HTTP、POP3、IMAP、SSH

2. IPv4におけるTTLと同じ働きをする、IPv6ヘッダーのフィールドは何ですか？

   ホップリミット

3. Neighbor Discovery Protocol（NDP）が扱う情報にはどのようなものがありますか？

   NDPでは、他のノード、アドレスの重複、ルート、DNSサーバー、ゲートウェイなど、ネットワークからさまざまな情報を取得できます。

109.2 基本的なネットワーク構成

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 109.2

総重量

4

主な知識分野

- 基本的なホストのTCP/IP設定の理解。
- NetworkManagerを利用した、ethernetとwi-fiネットワークの設定。
- systemd-networkdの知識。

用語とユーティリティ

- `/etc/hostname`
- `/etc/hosts`
- `/etc/nsswitch.conf`
- `/etc/resolv.conf`
- `nmcli`
- `hostnamectl`
- `ifup`
- `ifdown`

109.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.2 永続的なネットワーク構成
Lesson:	1 of 2

はじめに

TCP/IPネットワークでは、すべてのノードがそのネットワーク仕様にあわせて構成される必要があります。システム管理者は、オペレーティングシステムが起動するたびに適当なネットワークインターフェイスを、適切にセットアップするようにシステムを構成する必要があります。

ネットワークの仕様はオペレーティングシステムに依存しませんが、その仕様を満たすようにシステムを設定する方法はオペレーティングによってまちまちです。Linuxシステムでは、`/etc` ディレクトリにあるプレーンテキストファイルに保存された情報を読み出して、ネットワークインターフェイスを設定します。ネットワークへの接続を確立するためには、これらの設定ファイルの使用方法を理解しておく必要があります。

ネットワークインターフェース

オペレーティングシステムは、システムに接続されたイーサネット（Ethernet）やWi-Fiなどのネットワークデバイスを通してネットワークに接続します。ネットワークとの接続点を ネットワークインターフェイス と呼び、それらのネットワークデバイスをNIC（Network Interface Controller）と呼びます。ネットワークインターフェイスの主な役割は、ローカルデータを送信し、リモートデータを受信する出入り口を提供することです。接続するネットワークにあわせてネットワークインターフェイスを適切に設定しない限り、オペレーティングシステムはネットワーク内の他のマシンと通信することができません。なお、ネットワークインターフェイスの中には、同じマシンで動作するプロセス同士が通信するために使用される *ループバックインターフェイス* と呼ばれるものもあります。

ほとんどの場合、オペレーティングシステムのインストール中に、正しいネットワークインターフェイスがデフォルト設定されるか、手作業での設定が求められます。インストール後にネットワークが適切に機能していなかったり、ネットワーク設定をカスタマイズする必要がある場合は、これらの設定を検査したり変更したりすることがあります。

システム上のネットワークインターフェイスを一覧表示するコマンドがいくつかありますが、最もポピュラーでほぼすべてのディストリビューションで利用できるのは `ip` コマンドです。これはLinuxで標準的に利用されている基本的なネットワークツールである `iproute` ないし `iproute2` パッケージの一部です。`ip link show` コマンドで、ネットワークインターフェイスをすべて表示します。

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp3s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen
1000
        link/ether 00:16:3e:8d:2b:5b brd ff:ff:ff:ff:ff:ff
```

`nmcli device` コマンドが使用できることがあります。（訳注：デスクトップ環境を備えたLinuxシステムの大部分で利用可能です）。

```
$ nmcli device
DEVICE      TYPE      STATE      CONNECTION
enp3s5      ethernet  connected  Gigabit Powerline Adapter
lo          loopback  unmanaged  --
```

例示したコマンドはシステム設定を変更しないので、一般ユーザーが実行できます。どちらのコマンドでも、`lo`（ループバックインターフェース）と `enp3s5`（イーサネットインターフェース）の、2つのネットワークインターフェースを表示します。

一般的なデスクトップないしラップトップマシンでは、2〜3個のネットワークインターフェイスが定義されます。1つはループバックインターフェイスであり、もう1つ（ないし2つ）はネットワークハードウェア（イーサネットとWi-Fiなど）です。サーバーには数十のネットワークインターフェイスが定義されることがあります。オペレーティングシステムがデバイスを抽象化するので、どのようなネットワークハードウェアであっても、同じ方法でネットワークインターフェイスを設定することができます。

インターフェイスの基礎となるハードウェアの詳細を知っておくことは、通信が期待通りに機能しないときに何が起こっているのかを理解するのに役立ちます。たくさんのネットワークインターフェイスが利用可能なシステムでは、例えば、どれがWi-Fiでどれがイーサネットなのかを明確にしたいことがあります。このため、Linuxでは、ネットワークインターフェイスがどのデバイスとポートに対応するのかを簡単に識別できるように、ネットワークインターフェイスの命名規則を定めています。

インターフェース名

古いLinuxディストリビューションでは、カーネルがデバイスを見つけた順番に、イーサネットネットワークインターフェイスに `eth0`、`eth1` などの名前を付け、同様にワイヤレスネットワークインターフェイスには、`wlan0`、`wlan1` などの名前を付けていました。この命名規則では、例えば、どのイーサネットポートがインターフェイス `eth0` になるのかが特定できません。ハードウェアの検出タイミングによって、再起動後に2つのネットワークインターフェイスの名前が入れ替わってしまうことさえありました。

このあいまいさを払拭するために、最近のLinuxシステムでは、ネットワークインターフェイスとハードウェアを確定的に対応付けるための命名規則を採用しています。

systemdによる命名規則を使用するLinuxディストリビューションでは、すべてのネットワークインター

フェースに、インターフェースのタイプに応じた2文字のプレフィックスで始まる名前を付けます。

en
Ethernet

ib
InfiniBand

sl
シリアルラインIP（スリップ）

wl
Wireless local area network (WLAN)

ww
Wireless wide area network (WWAN)

Linuxは、以下の優先度順位に基づいて、ネットワークインターフェイスの名前と番号を付けていきます。

1. BIOSないしデバイスのファームウェアが提供するインデックス値に基づく名前（例：`eno1`）。
2. ファームウェアが提供するPCIエクスプレススロット番号に基づく名前（例：`ens1`）。
3. バスアドレスに基づく名前（例：`enp3s5`）。
4. インターフェイスのMACアドレスに基づく名前（例：`enx78e7d1ea46da`）。
5. レガシー規則に基づく名前（例：`eth0`）。

たとえば、ネットワークインターフェイス `enp3s5` は、最初の2つの命名方法に適合しなかったので、3番目のルールによってPCIバスのアドレスに基づいて命名されています。`lspci` コマンドの出力から、PCIデバイスアドレス `03:05.0` のデバイスであることが分かります。

```
$ lspci | fgrep Ethernet
03:05.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8110SC/8169SC Gigabit Ethernet (rev 10)
```

Linuxカーネルがネットワークインターフェイスを作成し、通常は自動的に構成されます。ネットワークを手動で構成するためのさまざまなコマンドは、名前で指定されたネットワークインターフェイスを通してカーネルとやり取りして、ネットワーク構成を確認・変更します。

インターフェース管理

長年にわたり、Linuxのネットワーク機能を制御するために `ifconfig` コマンドが使用されてきました。今でも簡単な調査や設定のために使用する事ができますが、イーサネット以外のサポートが不十分であるため非推奨とされています。現在では、ルーティングやトンネルなどTCP/IPのさまざまな側面を管理できる `ip` コマンドを使用します。

`ifconfig` コマンドは多機能で分かりにくいため、シンプルにネットワークインターフェイスを有効化・無効化するための `ifup` と `ifdown` コマンドもあります（訳注：新しいディストリビューションではデフォルトではインストールされないことが多いようです）。これらのコマンド

は、`/etc/network/interfaces` ファイルに書かれた定義に基づいてネットワークインターフェースを構成します。

`ifup` と `ifdown` で管理したいネットワークインターフェースは、`/etc/network/interfaces` ファイルで定義します。ネットワークインターフェイス名の前に `auto` と書かれた行は、`ifup -a` コマンドを実行したときにそのインターフェイスが順に有効化されることを示します。システム起動時には `ifup -a` が自動的に実行されますから、`auto` と書かれたインターフェイスは起動時に有効化されます。

WARNING

`ifup` および `ifdown` が参照する設定ファイルは、ディストリビューションによってかなり異なります。（訳注：これらのコマンドを使用する古いマシンに出会わない限り（まだかなりたくさんの現役マシンがあるはずです）、あえて覚える必要はありません）。

`iface` という単語で始まる行で、ネットワークインターフェイスの構成方法やパラメータを定義します。`iface` に続けて、インターフェイス名、アドレスファミリ、構成方法 を指定します。次の例では、インターフェース `lo`（ループバック）と `enp3s5` の構成を示しています。

```
auto lo
iface lo inet loopback

auto enp3s5
iface enp3s5 inet dhcp
```

TCP/IPネットワークでは、アドレスファミリ は `inet` ないし `inet6` です。`lo` インターフェイスでは、構成方法 に `loopback` を指定します。`dhcp` を指定すれば、DHCPサーバーからIP設定情報を取得して使用します。この設定例では、`ifup enp3s5` コマンドを実行すると、ネットワークインターフェイス `enp3s5` が有効化されます。

```
# ifup enp3s5
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/enp3s5/00:16:3e:8d:2b:5b
Sending on   LPF/enp3s5/00:16:3e:8d:2b:5b
Sending on   Socket/fallback
DHCPDISCOVER on enp3s5 to 255.255.255.255 port 67 interval 4
DHCPOFFER of 10.90.170.158 from 10.90.170.1
DHCPREQUEST for 10.90.170.158 on enp3s5 to 255.255.255.255 port 67
DHCPACK of 10.90.170.158 from 10.90.170.1
bound to 10.90.170.158 -- renewal in 1616 seconds.
```

この例では、`enp3s5` インターフェースを `dhcp` で構成していますから、`ifup enp3s5` コマンドでDHCPクライアントプログラムが呼び出されて、DHCPサーバーからIP設定を取得します。同様に、`ifdown enp3s5` コマンドを実行すれば、`enp3s5` インターフェースが無効化されます。

DHCPサーバーのないネットワークでは 構成方法 に `static` を指定して、`/etc/network/interfaces` にIP設定を直接記入します。例を示します。

```
iface enp3s5 inet static
    address 192.168.1.2/24
    gateway 192.168.1.1
```

構成方法 が `static` であるインターフェイスは、ネットワークハードウェアが検出される毎に自動的に起動されますから、`auto` で起動を指定する必要はありません。

同じインターフェースに複数の `iface` エントリがある場合、そのインターフェースを起動するとすべてのエントリが有効になります。これは、同じインターフェイスでIPv4アドレスとIPv6アドレスを構成する場合や、1つのインターフェイスに複数のIPアドレスを構成する場合などに有用です。

ローカル名とリモート名

TCP/IPの設定（IPアドレス、ゲートウェイ）だけでは、インターネットを活用することができません。人間が理解しやすい名前で、ホストを指定できることが必要です。

システムが自分をどう名乗るか（ローカル名）は自由に決めることができます。マシンをネットワークに接続しない場合でも、愛着の湧く名前を付けてあげるとよいでしょう。ローカル名にはドット（`.`）を含まない1単語として、ドメイン形式のネットワーク名と揃えた名前にするのがお勧めです。オペレーティングシステムは起動時に `/etc/hostname` ファイル（通常は1行のみのファイル）を読み込んで、ローカル名を決定します。

`/etc/hostname` ファイルを直接編集しても構いませんが、マシンのホスト名は `hostnamectl` コマンドで定義することがお勧めです（使用可能文字などがチェックされます）。`hostnamectl` コマンドの `set-hostname` サブコマンドは、引数に指定された名前を、`/etc/hostname` ファイルに書き込みます。

```
# hostnamectl set-hostname storage
# cat /etc/hostname
storage
```

`/etc/hostname` に定義する名前は 静的 なホスト名、つまり、起動時にシステムのローカル名としてセットする名前です。静的ホスト名は、64文字以内のASCII小文字のみ（スペースやドットは使用しない）とすることがお勧めです。また、厳密な要件ではありませんが、DNSドメイン名ラベルに使用できる形式（使用できる記号はハイフンのみ、など）とするのが良いでしょう。

`hostnamectl` では、静的ホスト名に加えて、2種類のホスト名を設定できます。

プリティホスト名
: 静的ホスト名とは異なり、プリティホスト名にはあらゆる種類の記号を含めることができます。よりわかりやすい名前を設定するために使います。例：“LAN Shared Storage”

```
# hostnamectl --pretty set-hostname "LAN Shared Storage"
```

仮ホスト名
: 静的なホスト名が設定されていないか、デフォルトの `localhost` の場合に使用されます。通常は自動設定された名前ですが、`hostnamectl` で変更できます。

`--pretty` オプションも `--transient` オプションも指定しない場合、指定された名前が3種の名前すべてに設定されます。静的ホスト名のみを設定する場合は、`--static` オプションを指定します。いずれの場合も、静的ホスト名のみが `/etc/hostname` ファイルに保存されます。コマンド `hostnamectl` を使用して、実行中のシステムに関するさまざまな説明や識別IDを表示することもできます。

```
$ hostnamectl status
     Static hostname: storage
     Pretty hostname: LAN Shared Storage
  Transient hostname: generic-host
          Icon name: computer-server
            Chassis: server
         Machine ID: d91962a957f749bbaf16da3c9c86e093
            Boot ID: 8c11dcab9c3d4f5aa53f4f4e8fdc6318
    Operating System: Debian GNU/Linux 10 (buster)
              Kernel: Linux 4.19.0-8-amd64
        Architecture: x86-64
```

`hostnamectl` コマンドのサブコマンドを省略すると、`status` を指定したのと同じになります。

ネットワークノードの名前とIPアドレスを対応付けるための基本的な方法が2つあります。1つはローカルデータベース（ファイル）を参照する方法で、もう1つはDNSサーバーを参照する方法です。両方を併用することができ、優先順位を ネームサービススイッチ （設定ファイルは `/etc/nsswitch.conf`）で構成することができます。ネームサービススイッチは、ホスト名とIPアドレスの対応付けだけではなく、さまざまな名前サービスが利用する情報源を決定するためにも使用されます。

ホスト名とIPアドレスの対応付けは、hosts データベースににによって管理されます。`/etc/nsswitch.conf` 内の `hosts` 行で、その情報源を提供するサービスを指定します。

```
hosts: files dns
```

このエントリ例は、ホスト情報をどこから探すかを指定しています。`files` はローカルファイルを参照することを、`dns` はDNSサービスを参照することを示していて、この順に検索が行われます。

ホストデータベースのローカルファイルは `/etc/hosts` です。これは、IPアドレスとホスト名を関連付ける単純なテキストファイルで、IPアドレスごとに1行です。

```
127.0.0.1 localhost
```

IPアドレス 127.0.0.1 は、ループバックインターフェイスのデフォルトアドレスですから、ホスト名 localhost と対応付けられています。

ひとつのIPアドレスに複数のホスト名（エイリアス）を関連付けることもできます。エイリアスには、短いホスト名やスペル違いの名前などに使われます。

```
192.168.1.10 foo.mydomain.org foo
```

`/etc/hosts` ファイルの書式は次のとおりです。

- エントリ（行）は、任意個数の空白やタブ文字でフィールドに区切られます。
- 文字 # から行末までのテキストはコメントとして無視されます。
- ホスト名には、英数字とハイフンおよびピリオドのみを含めることができます。
- ホスト名は英字で始まり、英数字で終わる必要があります。

IPv6アドレスを `/etc/hosts` に記入することもできます。次のエントリは、IPv6のループバックアドレスを示しています。

```
::1 localhost ip6-localhost ip6-loopback
```

`files` サービスで求める情報が得られなかった場合には、次の `dns` サービスに進みます。文字通りDNSサービスを利用して名前解決を行うものです。DNSサービスは リゾルバ（resolver） によって提供され、その構成ファイルは `/etc/resov.conf` です。次の例は、GoogleのパブリックDNSサーバーを利用する場合の `/etc/resolv.conf` を示しています。

```
nameserver 8.8.4.4
nameserver 8.8.8.8
```

`resolv.conf` では、`nameserver` エントリで **DNSサーバーのIPアドレス** を指定します。必要なネームサーバーは1つだけですが、3つまでのネームサーバーを指定でき、フォールバックとして使用されます。`nameserver` エントリが存在しない場合は、ローカルマシンのネームサーバーに接続を試みます。

DNSサービスでホスト名を検索する際に、自動的にドメインを補うようにリゾルバを構成することができます。例を示します。

```
nameserver 8.8.4.4
nameserver 8.8.8.8
domain mydomain.org
search mydomain.net mydomain.com
```

`domain` エントリに、ローカルドメイン名 `mydomain.org` を設定しておくと、短い（1ワードの）ホスト名を検索する際に、自動的にローカルドメイン名が補われます。`search` エントリも同様で、指定されているドメイン名を順に補って検索を試行します。

演習

1. システムに存在するネットワークアダプタを一覧表示するコマンドは何ですか？

2. インターフェイス名 `wlo1` は、どの種類のネットワークアダプタですか？

3. 起動時における `/etc/network/interfaces` ファイルの役割は何ですか？

4. ネットワークインターフェイス `eno1` を DHCPで設定するには、`/etc/network/interfaces` にどう書きますか？

発展演習

1. `hostnamectl` コマンドを使用して、ローカルマシンの 静的な ホスト名のみを `firewall` に変更するにはどうすればよいですか？

2. `hostnamectl` コマンドで設定できるホスト名以外の情報には何がありますか？

3. `firewall` および `router` という名前で、IPアドレスが `10.8.0.1` のホストにアクセスできるように、`/etc/hosts` のエントリを書いてください。

4. すべてのDNS要求を `1.1.1.1` に送信するには、`/etc/resolv.conf` ファイルにどう書きますか？

まとめ

このレッスンでは、Linuxの標準的なコマンドとファイルを使用して、永続的なネットワーク構成を行う方法を説明しました。インストール時に指定したネットワーク構成を変更する場合もあります。このレッスンでは、以下のトピックを取り上げました。

- Linuxがネットワークインターフェイスを識別する方法。
- 起動時のネットワークインターフェースの有効化と、基本的なIPネットワークの構成方法。
- オペレーティングシステムがホストと名前を対応付ける方法。

以下の概念、コマンド、手順を取り上げました。

- インターフェイスの命名規則。
- `ip` および `nmcli` を使用したネットワークインターフェイスの一覧表示。
- `ifup` と `ifdown` によるインターフェースのアクティブ化。
- `hostnamectl` コマンドと `/etc/hostname` ファイル。
- 設定ファイル：`/etc/nsswitch.conf`、`/etc/hosts`、`/etc/resolv.conf`。

演習の解答

1. システムに存在するネットワークアダプタを一覧表示するコマンドは何ですか？

   `ip link show` および `nmcli device`。レガシーな `ifconfig` が使える事もあります。

2. インターフェイス名 `wlo1` は、どの種類のネットワークアダプタですか？

   名前が `wl` で始まるので、無線LANアダプタです。

3. 起動時における `/etc/network/interfaces` ファイルの役割は何ですか？

   ブート時に有効化するネットワークインターフェイスの構成方法を保存します。（訳注：`/etc/network/interface` を使用するディストリビューションは減りつつあるようです。ディストリビューションに固有の方法を調べてください。）

4. ネットワークインターフェイス `eno1` を DHCPで設定するには、`/etc/network/interfaces` にどう書きますか？

   ```
   iface eno1 inet dhcp
   ```


発展演習の解答

1. `hostnamectl` コマンドを使用して、ローカルマシンの 静的な ホスト名のみを `firewall` に変更するにはどうすればよいですか？

   `--static` オプションを使用します： `hostnamectl --static set-hostname firewall`。

2. `hostnamectl` コマンドで設定できるホスト名以外の情報には何がありますか？

   `hostnamectl` で、ローカルマシンのデフォルトアイコン、その筐体種別、設置場所、設置環境を設定できます。

3. `firewall` および `router` という名前で、IPアドレスが `10.8.0.1` のホストにアクセスできるように、`/etc/hosts` のエントリを書いてください。

   `10.8.0.1 firewall router`

4. すべてのDNS要求を `1.1.1.1` に送信するには、 `/etc/resolv.conf` ファイルにどう書きますか？

   `nameserver` エントリとして `nameserver 1.1.1.1` のみを書きます。

Linux Professional Institute

109.2 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.2 永続的なネットワーク構成
Lesson:	2 of 2

はじめに

Linuxは、デスクトップ、サーバー、コンテナ、仮想マシン、モバイルデバイスなどが接続に使用する、ほぼすべてのネットワークテクノロジーをサポートしています。さまざまなテクノロジーが混在する場面におけるネットワークノード間の接続は動的でもあるため、オペレーティングシステムはネットワークを適切に管理できる必要があります。

ちょっと前まで、動的なネットワークインフラストラクチャを管理する方法は、ディストリビューションごとに独自のものでした。今日では、NetworkManager や systemd のようなツールが、より包括的で統合された機能を提供していて、あらゆる要求に応えています。

NetworkManager

ほとんどのLinuxディストリビューションは、ネットワーク接続を設定・制御するために、NetworkManager サービスを採用しています。NetworkManagerの目的は、ネットワーク設定をできるだけシンプルかつ自動的に行うことです。例えばDHCPを使用する場合であれば、必要に応じてNetworkManagerがIPアドレスの取得、経路の変更、DNSサーバーの切り替えを手配します。有線接続と無線接続の両方が利用可能な場合、NetworkManagerはデフォルトで有線接続を優先します。NetworkManagerは、可能な限り、常に少なくとも1つの接続をアクティブに保ち続けます。

NOTE

ネットワークへのリンクが確立されるとすぐに、ネットワークアダプタからDHCP（Dynamic Host Configuration Protocol）リクエストが送信されます。ネットワーク上でアクティブなDHCPサーバーは、IP通信のために必要な設定情報（IPアドレス、ネットワークマスク、デフォルトルートなど）を、クライアント（要求を送ったホスト）に返送します。

NetworkManagerデーモンは、デフォルトでは `/etc/network/interfaces` ファイルに記載されていな

いネットワークインターフェースのみを制御します。これは、他のネットワーク構成ツールとの干渉を防ぐためで、制御されていないネットワークインターフェースのみを扱います。

NetworkManagerサービスはroot権限で実行され、システムをオンラインに保つために必要な処理を行います。root権限を持たない一般ユーザーであっても、NetworkManagerデーモンと通信するクライアントアプリケーションを使用して、ネットワーク接続を作成したり変更したりできます。

NetworkManagerのクライアントアプリケーションには、コマンドライン環境で使用するものと、グラフィカル環境で使用するものがあります。グラフィカル環境で使用するものには、デスクトップ環境に応じて nm-tray、network-manager-gnome、nm-applet、plasma-nm などがあります。これらはデスクトップ環境のアクセサリとして提供されていて、通常はデスクトップバーのインジケーターアイコン、ないし、システム構成ユーティリティからアクセスできます。

コマンドライン環境では、NetworkManager自体が `nmcli` と `nmtui` という2つのクライアントプログラムを提供しています。どちらのプログラムも基本的な機能は同じですが、`nmtui` はcursesベースのメニューインターフェイスがあり、`nmcli` はスクリプトでも使用できる包括的なコマンドです。`nmcli` コマンドでは、NetworkManagerが制御するネットワーク関連プロパティを オブジェクトと呼ぶカテゴリに分類しています。

`general`
ステータスと操作全般

`networking`
ネットワーク制御全般

`radio`
無線接続

`connection`
接続情報

`device`
NetworkManagerが管理するデバイス

`agent`
シークレットエージェントならびにpolkitエージェント

`monitor`
変更の監視

`nmcli` コマンドの1番目の引数は、オブジェクトです。たとえば、システムの接続ステータスを表示するには、引数に `general` を指定します。

```
$ nmcli general
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

`STATE` 欄は、システムがネットワークに接続されているかどうかを示します。設定ミスやアクセス制限のために外部への接続が制限されている場合、`CONNECTIVITY` 欄には `full` 以外のステータスが表示さ

れます。`CONNECTIVITY` 欄に `Portal` と表示されている場合は、接続するために（通常はウェブブラウザ経由で）特別な認証ステップが必要です。その他の欄は無線接続の状態を示します。2種類の無線接続 `WIFI` と `WWAN`（Wide Wireless Area Network、つまり携帯電話ネットワーク）がサポートされています。サフィックスが `HW` の欄はデバイスの状態を示していて、サフィックスが付かない欄はネットワークの接続状態を示しています。

`nmcli` コマンドには、オブジェクト名に続けてその引数を指定します。引数が省略された場合は `status` が指定されたものと見なされます。つまり `nmcli general` コマンドは、実際には `nmcli general status` と解釈されます。

有線ネットワークでアクセスポイント（ルーター）に接続されている場合、行うべきことはほとんどありません。無線ネットワークでは追加の操作が必要になりますが、`nmcli` は簡単な操作で接続することができ、その設定が保存されて次回の接続でも自動的に利用されますから、ラップトップやモバイルデバイスではとても便利です。

Wi-Fiに接続する前に、そこで利用可能なネットワークを一覧表示できると便利です。システムにWi-Fiアダプターがある場合、 `nmcli device wifi list` で利用可能なWi-Fiネットワークを一覧表示します。

```
$ nmcli device wifi list
IN-USE  BSSID              SSID        MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
        90:F6:52:C5:FA:12  Hypnotoad   Infra  11    130 Mbit/s  67      ▂▄▆_  WPA2
        10:72:23:C7:27:AC  Jumbao      Infra  1     130 Mbit/s  55      ▂▄__  WPA2
        00:1F:33:33:E9:BE  NETGEAR     Infra  1     54 Mbit/s   35      ▂▄__  WPA1 WPA2
        A4:33:D7:85:6D:B0  AP53        Infra  11    130 Mbit/s  32      ▂▄__  WPA1 WPA2
        98:1E:19:1D:CC:3A  Bruma       Infra  1     195 Mbit/s  22      ▂___  WPA1 WPA2
```

ほとんどの場合は `SSID` 欄の名前を使用してネットワークを選択します。SSIDが `Hypnotoad` であるネットワークに接続するには、次のようにします。

```
$ nmcli device wifi connect Hypnotoad
```

デスクトップ環境のターミナルエミュレータ内でコマンドを実行した場合には、Wi-Fiネットワークのパスフレーズを要求するダイアログボックスが表示されます。コンソールやネットワーク経由でログインしている場合は、次のようにパスワードを引数に指定します。

```
$ nmcli device wifi connect Hypnotoad password MyPassword
```

Wi-FiネットワークがSSID名を非表示にしている場合でも、次のように `hidden yes` を指定すれば接続できます。

```
$ nmcli device wifi connect Hypnotoad password MyPassword hidden yes
```

システムに複数のWi-Fiアダプターがある場合は、使用するアダプターを `ifname` で指定します。たとえば、`wlo1` という名前のアダプタを使用して接続するには、次のようにします。

```
$ nmcli device wifi connect Hypnotoad password MyPassword ifname wlo1
```

Wi-Fi接続に成功すると、NetworkManagerはSSIDに由来する名前を付けて、後の接続のために保存します。`nmcli connection show` コマンドで、接続名とそのUUIDを一覧表示します。

```
$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Ethernet            53440255-567e-300d-9922-b28f0786f56e  ethernet  enp3s5
tun0                cae685e1-b0c4-405a-8ece-6d424e1fb5f8  tun       tun0
Hypnotoad           6fdec048-bcc5-490a-832b-da83d8cb7915  wifi      wlo1
4G                  a2cf4460-0cb7-42e3-8df3-ccb927f2fd88  gsm       --
```

接続の種類（`ethernet`、`wifi`、`tun`、`gsm`、`bridge` など）と、接続が使用しているデバイスが表示されます。ある接続に対する操作を行うためには、その名前またはUUIDを指定します。`Hypnotoad` への接続を無効化する例を示します。

```
$ nmcli connection down Hypnotoad
Connection 'Hypnotoad' successfully deactivated
```

逆に `nmcli connection up Hypnotoad` コマンドで、接続を有効化できます。既に `Hypnoad` への接続はNetworkManagerに保存されていますから、デバイス名やパスワードを指定する必要はありません。インターフェイス名を使用して再接続することもできますが、この場合は `connection` オブジェクトではなく `device` オブジェクトを使用します。

```
$ nmcli device disconnect wlo2
Device 'wlo1' successfully disconnected.
```

デバイス名を使用して、接続を再確立してみましょう。

```
$ nmcli device connect wlo2
Device 'wlo1' successfully activated with '833692de-377e-4f91-a3dc-d9a2b1fcf6cb'.
```

接続が確立されるたびにUUIDは変更されるので、接続名（SSID）かデバイス名を使うのがお勧めです。

ワイヤレスデバイスを使用していない場合は、電源をオフにして電力を節約できます。`nmcli` に radio オブジェクトを渡します。

```
$ nmcli radio wifi off
```

もちろんワイヤレスデバイスは、`nmcli radio wifi on` コマンドで再びオンにできます。

NetworkManagerが利用可能な既知のネットワークを見つけると、それらに自動的に接続します。いったん接続を確立すれば、再び手動で操作する必要はありません。NetworkManagerには、機能を拡張す

るプラグインがあり、VPN接続をサポートするものなどがあります。

systemd-networkd

systemdを採用しているシステムには、組み込みデーモンを使用してネットワーク接続を管理するオプションがあります。`systemd-networkd` はネットワークインターフェイスを制御し、`systemd-resolved` はローカルの名前解決を管理します。これらのサービスは従来のネットワーク構成方法と互換性がありますが、特にネットワークインターフェイスの構成には、知っておく価値のある機能が備わっています。

systemd-networkdがネットワークインターフェイスを構成するために使用する設定ファイルは、次の3つのディレクトリのいずれかに置かれます。

`/lib/systemd/network`
: システムネットワークディレクトリ。

`/run/systemd/network`
: 揮発性のランタイムネットワークディレクトリ。

`/etc/systemd/network`
: ローカル管理ネットワークディレクトリ。

ファイルは辞書順に処理されるので、数字で始まるファイル名を使用すると、順序が分かりやすくて設定しやすくなります。

同名の設定ファイルが複数のディレクトリにある場合は、`/etc` にあるファイルの優先度が最もが高く、次に `/run` にあるファイル、最後に `/lib` にあるファイルの順になります。優先度の低いファイルは無視されます。このようにファイルを配置することで、元のファイルを変更することなくインターフェース設定を変更することができます。たとえば `/etc/systemd/network` にあるファイルを変更して、`/lib/systemd/network` にあるファイルを無視させることができます。

設定ファイルの役割に応じてサフィックスが異なります。systemd-networkdは、名前が `.netdev` で終わるファイルを参照して bridge や tun などの仮想ネットワークデバイスを作成します。また、名前が `.link` で終わるファイルを参照して、ネットワークインターフェースの低レベルな（リンク層の）構成を行います。systemd-networkd は、ネットワークデバイスの追加を自動的に検出して設定します—他の方法で設定されたデバイスは無視します—から、これらのファイルを追加する必要はほとんどありません。

最も重要なファイルはサフィックスが `.network` のもので、ネットワークアドレスとルーティングを設定します。ファイル内の `[Match]` セクションで、対象とするネットワークインターフェース名を指定します。

たとえば `/etc/systemd/network/30-lan.network` ファイルでイーサネットインターフェイス `enp3s5` を設定するのであれば、その `[Match]` セクションに `Name=enp3s5` と記入します。

```
[Match]
Name=enp3s5
```

複数のインターフェイス名を空白で区切ったり、`en*` などのグロブ文字を使うことで、複数のネットワークインターフェイスを一度に指定することができます。また、MACアドレスでネットワークデバイス

を指定するなど、さまざまな方法でネットワークインターフェイスを指定できます。

```
[Match]
MACAddress=00:16:3e:8d:2b:5b
```

デバイスに設定するパラメータは、`[Network]` セクションに置きます。単純な静的ネットワークでは、`Address` と `Gateway` のエントリのみが必要です。

```
[Match]
MACAddress=00:16:3e:8d:2b:5b

[Network]
Address=192.168.0.100/24
Gateway=192.168.0.1
```

静的なIPアドレスではなく、DHCPでアドレスを設定するには、`DHCP` エントリを使用します。

```
[Match]
MACAddress=00:16:3e:8d:2b:5b

[Network]
DHCP=yes
```

DHCPを指定した場合、systemd-networkdサービスは、ネットワークインターフェイスにIPv4アドレスとIPv6アドレスの両方を割り当てようとします。IPv4のみを使用するには、`DHCP=ipv4` を、IPv6のみを使用するには `DHCP=ipv6` を指定します。

systemd-networkdで、パスワードで保護されたワイヤレスネットワークに接続することもできますが、あらかじめネットワークアダプターにワイヤレスネットワークの認証情報を登録しておくことが必要です。そのためには WPA_supplicant ツールを使用します。

最初のステップは、`wpa_passphrase` コマンドで資格情報ファイルを作成することです。

```
# wpa_passphrase MyWifi > /etc/wpa_supplicant/wpa_supplicant-wlo1.conf
```

引数は Wi-FiネットワークのSSIDです。このコマンドは、標準入力から `MyWifi` ワイヤレスネットワークのパスワードを読取って、そのハッシュ値を `/etc/wpa_supplicant/wpa_supplicant-wlo1.conf` に保存します。ファイル名に、ワイヤレスインターフェイスの名前 `wlo1` が含まれていることに注意してください。

`WPA_supplicant` サービスが実行されていれば、`/etc/wpa_supplicant/` にあるWPAパスフレーズファイルを自動的に読み取って、インターフェイスを制御するサービスを作成します。この例であれば、`wpa_supplicant@wlo1.service` になります。`systemctl start wpa_supplicant@wlo1.service` で、ワイヤレスアダプタをアクセスポイントに接続します。システム起動時に接続したければ、`systemctl enable wpa_supplicant@wlo1.service` コマンドを実行します。

最後に、`wlo1` インターフェース用の `.network` ファイルを `/etc/systemd/network/` に作成します。

これにより、WPA_Supplicantが `wlo1` インターフェイスをアクセスポイントに接続するとすぐに、systemd-networkdが `wlo1` インターフェースを構成します。

演習

1. `nmcli general status` コマンドの出力において、`CONNECTIVITY` 欄に `Portal` と表示されています。この意味は何ですか？

2. コンソール端末からログインしている一般ユーザーが、`nmcli` コマンドでWi-Fiネットワークに接続するにはどうしますか？ Wi-FiネットワークのSSIDは `MyWifi`、パスワードは `MyPassword` とします。

3. 以前に使用していたワイヤレスアダプタがOffになっている場合、どのコマンドでワイヤレスアダプタをOnにできますか？

4. systemd-networkdがネットワークインターフェイスを管理している場合、独自の構成ファイルを配置するディレクトリはどこですか？

発展演習

1. `nmcli` コマンドで、未使用の `Hotel Internet` という名前の接続を削除するにはどうしますか？

2. NetworkManagerはWi-Fiネットワークを定期的にスキャンしていて、`nmcli device wifi list` コマンドは最後のスキャンで見つかったアクセスポイントのみを一覧表示します。利用可能なアクセスポイントをすぐに再スキャンさせる `nmcli` コマンドは何ですか？

3. systemd-networkdの構成ファイルにおいて、すべてのイーサネットインターフェイスを指定したい場合に、`[Match]` セクションの `name` エントリにどう記入しますか？

4. `wpa_passphrase` コマンドで、パスワードを標準入力からではなく、引数で指定するにはどうしますか？

まとめ

このレッスンでは、さまざまな動的ネットワーク接続を管理するための一般的なツールについて説明しました。NetworkManager や systemd-networkd などのツールを使用すると、ユーザーの介入を最小限に押さえることができます。レッスンでは、次のトピックについて説明しました。

- NetworkManagerとsystemd-networkdの役割と関係。
- NetworkManagerとsystemd-networkdを使い分ける方法。
- NetworkManagerならびにsystemd-networkdにおける、基本的なネットワークインターフェース構成方法。

以下の概念、コマンド、手順を取り上げました。

- NetworkManagerのクライアントコマンド：`nmcli` と `nmtui`
- `nmcli` を使用して、ワイヤレスネットワークをスキャンして接続する方法
- systemd-networkdとWPA_supplicantを使用したWi-Fiネットワーク接続

演習の解答

1. `nmcli general status` コマンドの出力において、`CONNECTIVITY` 欄に `Portal` と表示されています。この意味は何ですか？

   接続プロセスを完了するために、追加の認証手順（通常はWebブラウザー経由）が必要であることを示しています。

2. コンソール端末からログインしている一般ユーザーが、`nmcli` コマンドでWi-Fiネットワークに接続するにはどうしますか？ Wi-FiネットワークのSSIDは `MyWifi`、パスワードは `MyPassword` とします。

   コマンドラインの端末では、以下のコマンドを使います。

   ```
   $ nmcli device wifi connect MyWifi password MyPassword
   ```

3. 以前に使用していたワイヤレスアダプタがOffになっている場合、どのコマンドでワイヤレスアダプタをOnにできますか？

   ```
   $ nmcli radio wifi on
   ```

4. systemd-networkdがネットワークインターフェイスを管理している場合、独自の構成ファイルを配置するディレクトリはどこですか？

   ローカル用のネットワーク管理ディレクトリに置きます：`/etc/systemd/network`。

発展演習の解答

1. `nmcli` コマンドで、未使用の `Hotel Internet` という名前の接続を削除するにはどうしますか？

```
$ nmcli connection delete "Hotel Internet"
```

2. NetworkManagerはWi-Fiネットワークを定期的にスキャンしていて、`nmcli device wifi list` コマンドは最後のスキャンで見つかったアクセスポイントのみを一覧表示します。利用可能なアクセスポイントをすぐに再スキャンさせる `nmcli` コマンドは何ですか？

   rootユーザーは、`nmcli device wifi rescan` コマンドで、利用可能なアクセスポイントを再スキャンできます。

3. systemd-networkdの構成ファイルにおいて、すべてのイーサネットインターフェイスを指定したい場合に、[Match] セクションの name エントリにどう記入しますか？

   `name=en*` と記入します。Linuxではイーサネットのインターフェイスは `en` がプレフィックスになります。また、systemd-networkdではグロブ文字が使えます。

4. wpa_passphrase コマンドで、パスワードを標準入力からではなく、引数で指定するにはどうしますか？

   `wpa_passphrase MyWifi MyPassword` のように、SSIDの次にパスワードを指定します。

109.3 基本的なネットワークのトラブルシューティング

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 109.3

総重量

4

主な知識分野

- ネットワークインターフェイスの追加、開始、停止、再起動、削除、または再構成を含むように、iproute2を用いてネットワークインターフェイスの手動設定できる。
- ルーティングテーブルを手動で自動的に構成する　　ルーティングテーブルを変更、表示、または構成し、不適切に設定されたデフォルトルートを、iproute2を用いて手動で修正する。
- ネットワーク構成に関連するデバッグの問題。
- net-toolsコマンドの知識。

用語とユーティリティ

- `ip`
- `hostname`
- `ss`
- `ping`
- `ping6`
- `traceroute`
- `traceroute6`
- `tracepath`
- `tracepath6`
- `netcat`
- `ifconfig`
- `netstat`
- `route`

109.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.3 基本的なネットワークのトラブルシューティング
Lesson:	1 of 2

はじめに

Linuxのネットワーク機能は非常に柔軟で強力です。事実、ネットワークデバイスは、高価な商業用装置も含め、LinuxベースのOSを搭載していることが多いです。Linuxネットワークは、それだけで一つの資格試験を作れるくらい膨大な領域です。このレッスンでは、基本構成とトラブルシューティングツールのごく一部を紹介するに過ぎないことをご承知おきください。

このレッスンより前にある、インターネットプロトコルとネットワーク構成についてのレッスンを復習するようにしてください。このレッスンでは、IPv4とIPv6のネットワーク構成とトラブルシューティングに使うツールを紹介します。

公式の試験範囲には含まれませんが、`tcpdump` のような パケットスニファ は、トラブルシューティングにおいて有用なので、ここで少し触れることとします。パケットスニッファを使うと、ネットワークインターフェイスを行き来するパケットを確認したり記録したりできます。ヘックス（16進数）ビューア や プロトコルアナライザ のようなツールを併用すると、より詳細にパケットを分析するのに役立ちます。少なくともこのようなプログラムが存在するのだということを知っておいても損はないでしょう。

`ip` コマンド

`ip` コマンドは、ネットワーク構成に関するあらゆる確認と設定を行える、新しいユーティリティです。このレッスンでは、`ip` コマンドのうち使用頻度の高いものを取り上げますが、それは `ip` コマンドでできることのごく一部でしかありません。より詳しく知りたい場合には、`ip` コマンドのマニュアルを読んでみてください。

`ip` の操作対象ごとにマニュアルページがあります。`ip` のマニュアルページの `SEE ALSO` セクションに、そのリストが挙がっています。

```
$ man ip
...
SEE ALSO
       ip-address(8), ip-addrlabel(8), ip-l2tp(8), ip-link(8), ip-maddress(8),
       ip-monitor(8), ip-mroute(8), ip-neighbour(8), ip-netns(8), ip-
       ntable(8), ip-route(8), ip-rule(8), ip-tcp_metrics(8), ip-token(8), ip-
       tunnel(8), ip-xfrm(8)
       IP Command reference ip-cref.ps
...
```

`ip` コマンド自体のマニュアルページではなく、`ip` と操作対象を `-` でつなげた名前で、操作対象に応じたマニュアルページを読むことができます（例：`man ip-route`）。

ヘルプ機能からも情報を得られます。操作対象のあとに `help` と入力して実行すると、ヘルプを確認できます。

```
$ ip address help
Usage: ip address {add|change|replace} IFADDR dev IFNAME [ LIFETIME ]
                                                  [ CONFFLAG-LIST ]
       ip address del IFADDR dev IFNAME [mngtmpaddr]
       ip address {save|flush} [ dev IFNAME ] [ scope SCOPE-ID ]
                            [ to PREFIX ] [ FLAG-LIST ] [ label LABEL ] [up]
       ip address [ show [ dev IFNAME ] [ scope SCOPE-ID ] [ master DEVICE ]
                         [ type TYPE ] [ to PREFIX ] [ FLAG-LIST ]
                         [ label LABEL ] [up] [ vrf NAME ] ]
       ip address {showdump|restore}
IFADDR := PREFIX | ADDR peer PREFIX
...
```

ネットマスクとルーティングの復習

IPv4とIPv6は、ネットワーク層のプロトコルです。ネットワークの設計者が、トラフィックの流れを制御できるように設計されています。イーサネットはリンク層のプロトコルで機器同士を接続するために使われますが、それだけでネットワークトラフィックの流れを制御することはほとんどできません。ネットワーク層のプロトコル、すなわちIPプロトコルが、トラフィックを制御する主役です。

IPプロトコルでは、ネットワークをいくつかのセグメントに分割した *サブネット* を作ることができます。これにより、1つのネットワークに接続するデバイス数を適切な規模に保ちながら、トラフィックが経由するルートを冗長化するなどの、トラフィック管理を行うことができます。

IPv4とIPv6のアドレスは、ネットワーク部とホスト部の2つに分けられます。上位ビットがネットワークを、下位ビットがホストを表します。ネットワークを表すビット数がいくつかであるかを、ネットマスク（サブネットマスク とも呼ばれます）で指定します。これを `/20` のような *プレフィックス長* と呼ばれる形で表すこともあります。呼び方はどうであれ、アドレスのうち、ネットワーク部として扱うビット数のことを指しています。IPv4では、（IPv4アドレス自体と同様の）ドットで区切った4つの10進数で表記すること（例えば 255.255.255.0）も多いです。

以下はIPv4の例です。ネットマスクによりネットワーク部とホスト部がどのように分割されたとしても、オクテット（8ビット）内の2進数の位取りは変わらないことに注意してください。

```
192.168.130.5/20

            192      168      130      5
            11000000 10101000 10000010 00000101

20ビット    = 11111111 11111111 11110000 00000000

ネットワーク = 192.168.128.0
ホスト      =           2.5
```

IPアドレスのネットワーク部は、IPv4ないしIPv6のマシンがパケットをどのネットワークインターフェイスに送り出すかを決定するために使用されます。それぞれのマシンは、それぞれのネットワークインターフェイスと、そこ接続されているネットワークアドレスの対応表（ルーティングテーブル）を持っています。ルーティングが有効となっているマシンが、自分宛ではないパケットを受け取ると、宛先のネットワークアドレスをルーティングテーブルから探します。マッチするエントリが見つかれば、パケットを対応するネットワークインターフェイスに送付します。マッチするエントリが見つからず、デフォルトルートが設定されていれば、デフォルトルートとして指定されたネットワークインターフェイス（ないしはリモートマシン）に宛てて送付します。マッチするエントリが見つからず、デフォルトルートが設定されていなければ、パケットは破棄されます。これが ルーティング の基本的な働きです。何台ものルーターが同じ働きでパケットを転送していくことで、パケットはインターネットの端から端まで届けられるのです。

ネットワークインターフェイスの構成

ネットワークインターフェイスの構成に使用するツールを2つ紹介します。`ifconfig` と `ip` です。`ifconfig` はまだ広く使われていますが、古いツールと位置づけられており、新しいシステムでは使えないかもしれません。

TIP 新しいLinuxディストリビューションでは、`net-tools` パッケージないし `inetutils-tools` パッケージをインストールすると、`ifconfig` などの古いネットワークコマンドツールを使えるようになります。

構成する前に、使用できるネットワークインターフェイスを確認しましょう。確認方法はいくつかあります。`-a` オプションを指定して `ifconfig` を実行するのが一つの方法です。

```
$ ifconfig -a
```

`ip` を使う方法もあります。`ip addr`、`ip a`、`ip address` のいずれかを実行すると、IPアドレスとともにネットワークインターフェイスを一覧表示します。この3つのコマンドの意味はどれも同じです。正式な操作対象の指定方法は `ip address` です（訳注：先頭数文字を指定すれば有効です）。よって、マニュアルページを参照する際には、`man ip-addr` ではなく `man ip-address` を実行します。

`ip` の操作対象として `link` を指定すると、使用できるネットワークインターフェイスを一覧表示します。

```
$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default
qlen 1000
    link/ether 08:00:27:54:18:57 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default
qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
```

sys ファイルシステムがマウントされていれば、/sys/class/net ディレクトリを一覧表示することでも、使用できるネットワークインターフェイスを確認できます。

```
$ ls /sys/class/net
enp0s3  enp0s8  lo
```

ifconfig や ip でネットワークインターフェイスを構成する場合は、rootユーザーとしてログインするか、sudo などのユーティリティを使ってroot権限で実行する必要があります。以下に例を載せます。

```
# ifconfig enp1s0 192.168.50.50/24
```

Linuxの ifconfig は、以下に例を示すように、サブネットマスクをいろいろな形式で柔軟に指定できます。

```
# ifconfig eth2 192.168.50.50 netmask 255.255.255.0
# ifconfig eth2 192.168.50.50 netmask 0xffffff00
# ifconfig enp0s8 add 2001:db8::10/64
```

IPv6では add という語を使っていることに気をつけてください。IPv6アドレスの前に add がなければエラーになります。

ip でネットワークインターフェイスを構成するなら、次のようなコマンドを実行します。

```
# ip addr add 192.168.5.5/24 dev enp0s8
# ip addr add 2001:db8::10/64 dev enp0s8
```

ip は、同じ形のコマンドでIPv4とIPv6のどちらでも構成できます。

低水準の構成

ip link コマンドを使うと、VLAN、ARP、MTUといったネットワークインターフェイスやプロトコルの低水準な機能を構成したり、ネットワークインターフェイスを無効にすることができます。

ip link でのよくある作業は、ネットワークインターフェイスの無効化／有効化です。ifconfig で

も同じことができます。

```
# ip link set dev enp0s8 down
# ip link show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
# ifconfig enp0s8 up
# ip link show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default
qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
```

ネットワークインターフェイスのMTU（Mximum Transmission Unit、1回に送信できるデータサイズ）を調整したいことがあります。これも `ip link` ないし `ifconfig` で調整できます。（訳注：MTUはメディアやプロトコルによって異なり、イーサネットでは1500バイト、光ファイバでは4352バイトなどと決められています。）

```
# ip link set enp0s8 mtu 2000
# ip link show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2000 qdisc pfifo_fast state UP mode DEFAULT group default
qlen 1000
    link/ether 08:00:27:54:53:59 brd ff:ff:ff:ff:ff:ff
# ifconfig enp0s3 mtu 1500
# ip link show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default
qlen 1000
    link/ether 08:00:27:54:53:59 brd ff:ff:ff:ff:ff:ff
```

ルーティングテーブル

ルーティングテーブルを確認するには、`route`、`netstat -r`、`ip route` のいずれかを実行します。ルーティングテーブルを変更するには、`route` ないし `ip route` を使います。3通りの方法でルーティングテーブルを確認してみます。

```
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.2.2        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
192.168.150.0   0.0.0.0         255.255.255.0   U         0 0          0 enp0s8
$ ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.150.0/24 dev enp0s8 proto kernel scope link src 192.168.150.200
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
```

```
192.168.150.0   0.0.0.0         255.255.255.0   U     0      0        0 enp0s8
```

IPv6に関する出力はありませんでした。IPv6のルーティングテーブルを確認するには、`route -6`、`netstat -6r`、`ip -6 route` のいずれかを実行します。

```
$ route -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
2001:db8::/64                  [::]                       U    256 0      0 enp0s8
fe80::/64                      [::]                       U    100 0      0 enp0s3
2002:a00::/24                  [::]                       !n   1024 0      0 lo
[::]/0                         2001:db8::1                UG   1   0      0 enp0s8
localhost/128                  [::]                       Un   0   2     84 lo
2001:db8::10/128               [::]                       Un   0   1      0 lo
fe80::a00:27ff:fe54:5359/128   [::]                       Un   0   1      0 lo
ff00::/8                       [::]                       U    256 1      3 enp0s3
ff00::/8                       [::]                       U    256 1      6 enp0s8
```

`netstat -r6` の出力例は、`route -6` の出力とまったく同じであるため、割愛しています。上記の `route` ないし `route -6` コマンドを実行した出力のうち、`Destination` 列は宛先のネットワークアドレス、`Gateway` 列ならびに `Next Hop` 列はゲートウェイルーターのホストアドレス、`Genmask` 列はサブネットマスクを、それぞれ示しています。`Flag` 列はルートについての情報を示し、`U` は有効、`!` は使用されない、`n` はキャッシュされていない、`G` はゲートウェイという意味です。カーネルは、探索速度向上のため、既知のルートを個別にキャッシュします。`Metric` ないし `Met` 列は、カーネルが利用するものではなく、ルーティングプロトコル（訳注：ルーター同士がルート情報を自動的に交換し合うためのプロトコル）が動的にルートを決めるために使うAD（administrative distance）値です。`Ref` 列は参照数、すなわちルートが使われた回数のことです。`Metric` と同様に、Linuxカーネルが利用するものではありません。`Use` 列はルートの探索数を示します。`Iface` ないし `If` 列は、ネットワークインターフェイスを示しています。

`netstat -r` の出力の説明に移ります。`MSS` 列はそのルートでのTCPパケットの最大サイズを示します。`Window` 列はTCPのデフォルトウィンドウサイズを示します。`irtt` はパケットの往復時間を示します。

`ip route` と `ip -6 route` の出力は、次のように読み解きます。

1. 宛先ネットワーク
2. 使用するネットワークインターフェイス（アドレスがある場合にはネットワークインターフェイスの前に表示される）
3. 文字 `proto` とルートを追加したルーティングプロトコル
4. 文字 `scope` とそのルートのスコープ。この部分が省略されていたら、グローバルスコープ、すなわちゲートウェイです。
5. 文字 `Metric` と、そのルートのコストを示す値。動的ルーティングプロトコルで使用されます。
6. （IPv6の場合のみ）RFC4191によるルータ優先度

2つほど例を見てみましょう。

IPv4の例

```
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
```

1. 宛先はデフォルトルートです。
2. ネットワークインターフェイス `enp0s3` を使用します（ゲートウェイアドレス `10.0.2.2` に送付します）。
3. DHCPがこのルートを追加しました。
4. スコープは省略されているのでグローバルです。
5. コスト値は `100` です。
6. IPv6ルータ優先度はありません。

IPv6の例

```
fc0::/64 dev enp0s8 proto kernel metric 256 pref medium
```

1. 宛先は `fc0::/64` です。
2. ネットワークインターフェイス `enp0s8` を使用します。
3. カーネルが自動的にこのルートを追加しました。
4. スコープは省略されているのでグローバルです。
5. コスト値は `256` です。
6. IPv6ルータ優先度は `medium` です。

ルーティングテーブルの管理

ルーティングテーブルは、`route` ないし `ip route` を使って管理できます。`route` コマンドを使い、ルートを追加して、削除する例を以下に示します。IPv6については `-6` オプションを指定します。

```
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
# route -6 add 2001:db8:1::/64 gw 2001:db8::3
# ping6 -c 2 2001:db8:1::20
PING 2001:db8:1::20(2001:db8:1::20) 56 data bytes
64 bytes from 2001:db8:1::20: icmp_seq=1 ttl=64 time=0.451 ms
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.438 ms
# route -6 del 2001:db8:1::/64 gw 2001:db8::3
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
```

同じことを `ip route` コマンドで行う例を以下に示します。

```
# ping6 -c 2 2001:db8:1:20
connect: Network is unreachable
# ip route add 2001:db8:1::/64 via 2001:db8::3
# ping6 -c 2 2001:db8:1:20
```

```
PING 2001:db8:1::20(2001:db8:1::20) 56 data bytes
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.529 ms
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.438 ms
# ip route del 2001:db8:1::/64 via 2001:db8::3
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
```


演習

1. ネットワークインターフェイスを一覧表示するコマンドは何ですか？

2. ネットワークインターフェイス `wlan1` を一時的に無効にし、再び有効にしてください。

3. 以下の選択肢のうち、IPv4のサブネットマスクとして適切なものをすべて選んでください。

`0.0.0.255`	
`255.0.255.0`	
`255.252.0.0`	
`/24`	

4. デフォルトルートを確認するコマンドは何ですか？

5. ネットワークインターフェイス `enp0s9` に `172.16.15.16/16` という2つ目のIPアドレスを追加してください。

発展演習

1. `ip` コマンドで、ネットワークインターフェイス `enp0s9` に idが `20` のタグVLANを構成してください。

2. デフォルトルートとして、IPアドレス `192.168.1.1` を設定してください。

3. `ip neighbour` コマンドのマニュアルページを読み、そのコマンドを実行するとどうなるかを答えてください。

4. ルーティングテーブルのバックアップと復元を行うコマンドは、それぞれ何ですか？

5. `ip` コマンドで、ネットワークインターフェイス `enp0s9` に優先度が `50` のスパニングツリーを構成してください。

まとめ

ネットワークは、通常、システムの起動スクリプトか、NetworkManagerのようなヘルパーが構成します。多くのディストリビューションでは、起動スクリプトが使用する構成ファイルを編集するツールが付属しています。詳細についてはお使いのディストリビューションのドキュメントを参照してください。

手動でネットワークを構成できるようになれば、トラブルシューティングを効果的に行えます。また、バックアップからの復元や新しいハードウェアへの移行といった、起動スクリプトやヘルパーが存在しない最小限の環境での作業にも役立ちます。

このレッスンで取り上げたユーティリティには、ここで説明したよりも多くの機能があります。各ユーティリティのマニュアルページをめくり、オプションに習熟する価値があります。`ss` と `ip` は新しいコマンドで、その他のコマンドは、まだ広く用いられているものの、古いツールです。

このレッスンで紹介したツールに慣れるには練習あるのみです。それなりのメモリ（RAM）を搭載したコンピュータを使っているなら、仮想マシンで仮想ネットワークを構築して練習できます。仮想マシンが3台あれば、練習には充分です。

このレッスンでは、次のコマンドを取り上げました。

`ifconfig`
ネットワークインターフェイスの構成と確認を行う古いユーティリティです。

`ip`
ネットワークインターフェイスの構成や確認などを行う、新しい多機能なユーティリティです。

`netstat`
ネットワーク接続とルーティング情報を確認する古いコマンドです。

`route`
ルーティングテーブルの確認と変更を行う古いコマンドです。

演習の解答

1. ネットワークインターフェイスを一覧表示するコマンドは何ですか？

   `ip link`、`ifconfig -a`、`ls /sys/class/net` のいずれかです。

2. ネットワークインターフェイス `wlan1` を一時的に無効にし、再び有効にしてください。

   `ifconfig` ないし `ip link` を使います。

   `ifconfig` を使うなら次のコマンドを実行します。

   ```
   $ ifconfig wlan1 down
   $ ifconfig wlan1 up
   ```

   `ip link` を使うなら次のコマンドを実行します。

   ```
   $ ip link set wlan1 down
   $ ip link set wlan1 up
   ```

3. 以下の選択肢のうち、IPv4のサブネットマスクとして適切なものをすべて選んでください。
   - `255.252.0.0`
   - `/24`

     この2つ以外の選択肢は、アドレスをネットワーク部とホスト部の2つに分割していないので、適切ではありません。左側のネットワーク部のビットはすべて1で、右側のホスト部のビットはすべて0でなければなりません。

4. デフォルトルートを確認するコマンドは何ですか？

   `route`、`netstat -r`、`ip route` のいずれかです。

   ```
   $ route
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   default         server          0.0.0.0         UG    600    0        0 wlan1
   192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan1
   $ netstat -r
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
   default         server          0.0.0.0         UG        0 0          0 wlan1
   192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 wlan1
   $ ip route
   default via 192.168.1.20 dev wlan1 proto static metric 600
   192.168.1.0/24 dev wlan1 proto kernel scope link src 192.168.1.24 metric 600
   ```

5. ネットワークインターフェイス `enp0s9` に `172.16.15.16/16` という2つ目のIPアドレスを追加して

ください。

`ip address` ないし `ifconfig` を使います。`ifconfig` を使うのは古いやり方です。

```
$ ip addr add 172.16.15.16/16 dev enp0s9 label enp0s9:sub1
```

上記コマンド中の `label enp0s9:sub1` という部分は、`enp0s9` にエイリアスを追加しています。`ifconfig` を使わないのであれば、この部分は省略可能です。`ifconfig` を使うのであれば、この部分を省略すると、正常に動作はしますが、追加したアドレスが `ifconfig` で出力されません。

`ifconfig` を使ってIPアドレスを追加するなら、次のコマンドを実行します。

```
$ ifconfig enp0s9:sub1 172.16.15.16/16
```

発展演習の解答

1. `ip` コマンドで、ネットワークインターフェイス `enp0s9` に idが `20` のタグVLANを構成してください。

   `ip link` には `vlan` オプションがあり、これを使います。次のコマンドを実行します。（訳注：VLANはLPIC-1の範囲外です。）

   ```
   # ip link add link enp0s9 name enp0s9.20 type vlan id 20
   ```

2. デフォルトルートとして、IPアドレス `192.168.1.1` を設定してください。

   `route` ないし `ip route` を使います。次のコマンドのいずれかを実行します。

   ```
   # route add default gw 192.168.1.1
   # ip route add default via 192.168.1.1
   ```

3. `ip neighbour` コマンドのマニュアルページを読み、そのコマンドを実行するとどうなるかを答えてください。

   次のコマンドを実行してマニュアルページを読みます。

   ```
   $ man ip-neigbour
   ```

   `ip neighbour` コマンドは、ARPキャッシュを表示します。（訳注：ARPはLPIC-1の範囲外です。）

   ```
   $ ip neighbour
   10.0.2.2 dev enp0s3 lladdr 52:54:00:12:35:02 REACHABLE
   ```

4. ルーティングテーブルのバックアップと復元を行うコマンドは、それぞれ何ですか？

   次のコマンドを実行すると、ルーティングテーブルのバックアップと復元を行います。

   ```
   # ip route save > /root/routes/route_backup
   # ip route restore < /root/routes/route_backup
   ```

5. `ip` コマンドで、ネットワークインターフェイス `enp0s9` に優先度が `50` のスパニングツリーを構成してください。

   VLANを構成するのと似たようなやり方で `ip link` コマンドを実行し、`bridge` タイプを使うことで、スパニングツリーを設定できます。次のコマンドを実行します。（訳注：スパニングツリーはLPIC-1の範囲外です。）

```
# ip link add link enp0s9 name enp0s9.50 type bridge priority 50
```

109.3 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.3 基本的なネットワークのトラブルシューティング
Lesson:	2 of 2

はじめに

LinuxベースのOSには、ネットワークのトラブルシューティングツールが多数付属しています。このレッスンでは、その中でも一般的なツールを取り上げます。OSI参照モデルやTCP/IPモデルなどのネットワーク階層モデル、IPアドレス（IPv4、IPv6）、ルーティングとスイッチングの基礎を理解していることを前提として話を進めます。

ネットワーク接続が正常であるかを確認するには、（ウェブブラウザ等の）アプリケーションを使ってみるのが手っ取り早いです。正常に動作しなければ、これから紹介するいろいろなツールを使って問題を突き止めます。

`ping` で接続確認

`ping` と `ping6` コマンドを使えば、それぞれIPv4アドレスとIPv6アドレスに、ICMPエコー要求のパケットを送信します。ICMPエコー要求では、宛先アドレスに少量のデータを送信します。宛先アドレスに到達すると、同じデータのICMPエコー応答が送り返されます。

```
$ ping -c 3 192.168.50.2
PING 192.168.50.2 (192.168.50.2) 56(84) bytes of data.
64 bytes from 192.168.50.2: icmp_seq=1 ttl=64 time=0.525 ms
64 bytes from 192.168.50.2: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 192.168.50.2: icmp_seq=3 ttl=64 time=0.449 ms

--- 192.168.50.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
```

```
rtt min/avg/max/mdev = 0.419/0.464/0.525/0.047 ms
```

```
$ ping6 -c 3 2001:db8::10
PING 2001:db8::10(2001:db8::10) 56 data bytes
64 bytes from 2001:db8::10: icmp_seq=1 ttl=64 time=0.425 ms
64 bytes from 2001:db8::10: icmp_seq=2 ttl=64 time=0.480 ms
64 bytes from 2001:db8::10: icmp_seq=3 ttl=64 time=0.725 ms

--- 2001:db8::10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.425/0.543/0.725/0.131 ms
```

`-c` オプションではパケットを送信する回数を指定します。このオプションを省略すると、`ping` と `ping6` は、Ctrl+Cなどで停止されるまでパケットを送信し続けます。

pingで応答が返ってこないことが直ちに接続不能を意味するわけではありません。ファイアウォールやルーターのアクセスコントロールが必要最小限の接続しか許可していないことは多々あります。ICMPエコー要求と応答がブロックされていることも多いです。ICMPエコー要求のパケットには任意のデータを含めることができるので、攻撃者がこれを利用してデータを盗み取ろうとする可能性があるからです。（訳注：ICMP（Internet Control Message Protocol）は、その名前の通りネットワーク制御のための重要な機能を実装しています。自分が何をしているのか理解していない限り、（特にICMPv6では）ICMP全体をブロックしてはいけません。）

`traceroute` で経路の追跡

`traceroute` と `traceroute6` コマンドを使えば、それぞれIPv4アドレスとIPv6アドレスについて、パケットが宛先に到着するまでの経路（ルート）がわかります。IPヘッダーのTTL（Time To Live）フィールドの値を1ずつ増やしながら、複数のパケットを宛先に送信するという仕組みになっています。TTLはルーターを通過する度に1ずつ減り、TTLが0になるとルーターはパケットを破棄してメッセージを送り返すので、経由しているルートがわかるというわけです。（訳注：ICMPエコー要求よりもブロックされていることが多いので、インターネットではほとんど使い物になりません）。

```
$ traceroute 192.168.1.20
traceroute to 192.168.1.20 (192.168.1.20), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.396 ms  0.171 ms  0.132 ms
 2  192.168.1.20 (192.168.1.20)  2.665 ms  2.573 ms  2.573 ms
$ traceroute 192.168.50.2
traceroute to 192.168.50.2 (192.168.50.2), 30 hops max, 60 byte packets
 1  192.168.50.2 (192.168.50.2)  0.433 ms  0.273 ms  0.171 ms
$ traceroute6 2001:db8::11
traceroute to 2001:db8::11 (2001:db8::11), 30 hops max, 80 byte packets
 1  2001:db8::11 (2001:db8::11)  0.716 ms  0.550 ms  0.641 ms
$ traceroute 2001:db8::11
traceroute to 2001:db8::11 (2001:db8::11), 30 hops max, 80 byte packets
 1  2001:db8::10 (2001:db8::11)  0.617 ms  0.461 ms  0.387 ms
$ traceroute net2.example.net
traceroute to net2.example.net (192.168.50.2), 30 hops max, 60 byte packets
 1  net2.example.net (192.168.50.2)  0.533 ms  0.529 ms  0.504 ms
$ traceroute6 net2.example.net
traceroute to net2.example.net (2001:db8::11), 30 hops max, 80 byte packets
```

```
 1  net2.example.net (2001:db8::11)  0.738 ms  0.607 ms  0.304 ms
```

traceroute をオプションを指定せずに実行すると、TTLを増やしながら、意味を持たないデータで3回ずつのUDPパケットを、33434番ポートに送信します。このコマンドの出力行は、パケットが通過したルーターインターフェイスです。各行に表示される時間は、パケットが往復するのに要した時間です。当該ルーターインターフェイスのIPアドレスとともに示されます。名前解決ができれば、そのルーターインターフェイスのDNS名を表示します。時間の代わりに * が表示されることがあります。これは、traceroute がそのパケットのTTL超過メッセージを受け取っていないことを意味します。* が表示されるということは、その直前の応答がそのルートでの最後のホップだったのかもしれません。（訳注：ほとんどの場合はブロックされたことを示します。）

root 権限で実行できるなら、-I オプションを指定することで、traceroute がUDPパケットではなくICMPエコー要求を送信するようになります。宛先のホストは、UDPパケットよりもICMPエコー要求に応答する可能性が高いので、このほうが多少は実効的です。

```
# traceroute -I learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 60 byte packets
 1  047-132-144-001.res.spectrum.com (47.132.144.1)  9.764 ms  9.702 ms  9.693 ms
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  8.389 ms  8.481 ms  8.480 ms
 3  dtr01hlrgnc-gbe-4-15.hlrg.nc.charter.com (96.34.64.172)  8.763 ms  8.775 ms  8.770 ms
 4  acr01mgtnnc-vln-492.mgtn.nc.charter.com (96.34.67.202)  27.080 ms  27.154 ms  27.151 ms
 5  bbr01gnvlsc-bue-3.gnvl.sc.charter.com (96.34.2.112)  31.339 ms  31.398 ms  31.395 ms
 6  bbr01aldlmi-tge-0-0-0-13.aldl.mi.charter.com (96.34.0.161)  39.092 ms  38.794 ms  38.821 ms
 7  prr01ashbva-bue-3.ashb.va.charter.com (96.34.3.51)  34.208 ms  36.474 ms  36.544 ms
 8  bx2-ashburn.bell.ca (206.126.236.203)  53.973 ms  35.975 ms  38.250 ms
 9  tcore4-ashburnbk_0-12-0-0.net.bell.ca (64.230.125.190)  66.315 ms  65.319 ms  65.345 ms
10  tcore4-toronto47_2-8-0-3.net.bell.ca (64.230.51.22)  67.427 ms  67.502 ms  67.498 ms
11  agg1-toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.114)  61.270 ms  61.299 ms  61.291 ms
12  dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  61.101 ms  61.177 ms  61.168 ms
13  207.35.12.142 (207.35.12.142)  70.009 ms  70.069 ms  59.893 ms
14  unassigned-117.001.centrilogic.com (66.135.117.1)  61.778 ms  61.950 ms  63.041 ms
15  unassigned-116.122.akn.ca (66.135.116.122)  62.702 ms  62.759 ms  62.755 ms
16  208.94.166.201 (208.94.166.201)  62.936 ms  62.932 ms  62.921 ms
```

ICMPエコー要求と応答がブロックされていることもあります。そういう場合にはTCPが使えます。開いていることがわかっているTCPのポートを用いれば、宛先ホストが応答すること請け合いです。TCPを使うには、-T オプションと -p オプションにポート番号を指定します。ICMPエコー要求と同じように、TCPを使うためには root 権限で実行しなければなりません。

```
# traceroute -m 60 -T -p 80  learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 60 hops max, 60 byte packets
 1  * * *
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  12.178 ms  12.229 ms  12.175 ms
 3  dtr01hlrgnc-gbe-4-15.hlrg.nc.charter.com (96.34.64.172)  12.134 ms  12.093 ms  12.062 ms
 4  acr01mgtnnc-vln-492.mgtn.nc.charter.com (96.34.67.202)  31.146 ms  31.192 ms  31.828 ms
 5  bbr01gnvlsc-bue-3.gnvl.sc.charter.com (96.34.2.112)  39.057 ms  46.706 ms  39.745 ms
 6  bbr01aldlmi-tge-0-0-0-13.aldl.mi.charter.com (96.34.0.161)  50.590 ms  58.852 ms  58.841 ms
 7  prr01ashbva-bue-3.ashb.va.charter.com (96.34.3.51)  34.556 ms  37.892 ms  38.274 ms
 8  bx2-ashburn.bell.ca (206.126.236.203)  38.249 ms  36.991 ms  36.270 ms
 9  tcore4-ashburnbk_0-12-0-0.net.bell.ca (64.230.125.190)  66.779 ms  63.218 ms tcore3-ashburnbk_100ge0-12-0-
```

```
0.net.bell.ca (64.230.125.188)  60.441 ms
10  tcore4-toronto47_2-8-0-3.net.bell.ca (64.230.51.22)  63.932 ms  63.733 ms  68.847 ms
11  agg2-toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.118)  60.144 ms  60.443 ms agg1-toronto47_xe-7-0-
0_core.net.bell.ca (64.230.161.114)  60.851 ms
12  dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  67.246 ms dis4-clarkson16_7-0.net.bell.ca
(64.230.131.102)  68.404 ms dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  67.403 ms
13  207.35.12.142 (207.35.12.142)  66.138 ms  60.608 ms  64.656 ms
14  unassigned-117.001.centrilogic.com (66.135.117.1)  70.690 ms  62.190 ms  61.787 ms
15  unassigned-116.122.akn.ca (66.135.116.122)  62.692 ms  69.470 ms  68.815 ms
16  208.94.166.201 (208.94.166.201)  61.433 ms  65.421 ms  65.247 ms
17  208.94.166.201 (208.94.166.201)  64.023 ms  62.181 ms  61.899 ms
```

ping と同様、traceroute には限界があります。ファイアウォールやルーターが、traceroute により送受信されるパケットをブロックしている可能性があります。root 権限があれば、ICMPやTCPを使うことで、多少はマシな結果が得られることでしょう。（訳注：ホスト名を表示させるにはDNSの逆引きを行う必要があり、長い時間がかかるが普通です。-n オプションを付けて、ホスト名ではなくIPアドレスを表示させるとよいでしょう。）

tracepath でMTUを調査

tracepath コマンドは traceroute コマンドと似ています。経路とともに MTU（Maximum Transmission Unit、1回に送信できるデータサイズ）を追跡する点が異なります。MTUはネットワークインターフェイスの設定、あるいはハードウェアがそのプロトコルで送受信できる最大データ単位の上限によって決まります。tracepath の仕組みは traceroute と同じで、TTLを増やしながら複数のパケットを送ります。ただし、非常に大きなUDPデータグラムを送るという点が異なります。経路上でMTUの最も小さいデバイスが送信できるデータサイズを上回ることは必定です。そうなると、そのデバイスは、パケットを宛先に届けられなかったという応答を返します。そうして送り返されるICMP宛先到達不能パケットには、パケットを送信し切れなかったリンクのMTUを示すフィールドがあります。tracepath は、以後、そのMTUのサイズのパケットを送信します。

```
$ tracepath 192.168.1.20
 1?: [LOCALHOST]                                         pmtu 1500
 1:  10.0.2.2                                              0.321ms
 1:  10.0.2.2                                              0.110ms
 2:  192.168.1.20                                          2.714ms reached
     Resume: pmtu 1500 hops 2 back 64
```

IPv4とIPv6の両方に使える traceroute とは異なり、IPv6には tracepath6 を使います。

```
$ tracepath 2001:db8::11
tracepath: 2001:db8::11: Address family for hostname not supported
$ tracepath6 2001:db8::11
 1?: [LOCALHOST]                      0.027ms pmtu 1500
 1:  net2.example.net                                      0.917ms reached
 1:  net2.example.net                                      0.527ms reached
     Resume: pmtu 1500 hops 1 back 1
```

出力は traceroute に似ています。tracepath を使う利点は、最終行にリンク全体で最小のMTUが出力されることです。これは、フラグメントを処理できない接続をトラブルシューティングする際に役立

ちます。

`ping` や `traceroute` と同様、ファイアウォールやルーターが、`tracepath` により送受信されるパケットをブロックしている可能性があります。

任意の接続を作成

`nc` プログラム（netcatと呼ばれることもあります）を使うと、TCP接続またはUDP接続で、任意のデータを送受信できます。以下の実例を通じて機能を説明します。

`1234` 番ポートで接続を待ち受けることにします。（訳注：このコマンドを実行するホストが `net2.example.net` であると想定して以下の内容を読み進めてください。）

```
$ nc -l 1234
LPI Example
```

この `nc -l 1234` コマンドは、TCPの1234番ポートで接続を待ち受けます。別のマシンで次のコマンド（`nc net2.example.net 1234`）を実行して、`LPI Example` と打ち込むと、待ち受けていた側のターミナルに打ち込んだ文字が表示されます。

```
$ nc net2.example.net 1234
LPI Example
```

受信側でも送信側でも、接続を中止するには、`Ctrl`+`C`を押してください。

`nc`（netcat）はIPv4とIPv6のどちらでも動作しますし、TCPでもUDPでも動作します。素のリモートシェルを立ち上げることさえできます。

WARNING

（以下に示す素のリモートシェルを立ち上げる例では `-e` オプションを使っていますが）インストールされている `nc` が `-e` オプションをサポートしているとは限りません。インストールされている `nc` のマニュアルページを読み、`-e` オプションのセキュリティについての情報と、`-e` オプションがサポートされていない場合にリモートシステムでコマンドを実行する代替手段を確認するようにしてください。

```
$ hostname
net2
$ nc -u -e /bin/bash -l 1234
```

`-u` オプションを指定するとUDP接続になります。`-e` オプションを指定して接続を待ち受けると、受信したものをそのまますべて `-e` オプションの次に指定した実行可能ファイル（上記の例では `/bin/bash`）の標準入力に送ります。

```
$ hostname
net1
$ nc -u net2.example.net 1234
hostname
```

```
net2
pwd
/home/emma
```

`nc -u net2.example.net 1234` 実行後の `hostname` コマンドと `pwd` コマンドの出力結果は、接続を待ち受けていたホストが出力しているものであることに注目してください。

現在の接続と待ち受けの確認

`netstat` ないし `ss` を使うと、現在の接続と待ち受けの状態を確認できます。`netstat` は `ifconfig` と同じように古いツールです。`netstat` と `ss` は出力とオプションが似ています。どちらのプログラムでも使えるオプションを紹介します。

`-a`
すべてのソケットを表示します。

`-l`
待ち受けているソケットを表示します。

`-p`
接続と関連するプロセスを表示します。

`-n`
ポートとアドレスについて、名前解決を行いません。

`-t`
TCP接続を表示します。

`-u`
UDP接続を表示します。

両方のプログラムについて、よく使うオプションの指定例とその出力結果を、以下に示します。

```
# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      892/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1141/master
tcp6       0      0 :::22                   :::*                    LISTEN      892/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1141/master
udp        0      0 0.0.0.0:68              0.0.0.0:*                           692/dhclient
# ss -tulnp
Netid  State      Recv-Q Send-Q     Local Address:Port                    Peer Address:Port      Process
udp    UNCONN     0      0                     :68                                *:
users:(("dhclient",pid=693,fd=6))
tcp    LISTEN     0      128                   :22                                *:
users:(("sshd",pid=892,fd=3))
tcp    LISTEN     0      100            127.0.0.1:25                                :
users:(("master",pid=1099,fd=13))
tcp    LISTEN     0      128                [::]:22                             [::]:*
```

```
users:(("sshd",pid=892,fd=4))
tcp    LISTEN     0      100                [::1]:25                              [::]:*
users:(("master",pid=1099,fd=14))
```

`Recv-Q` 列は、ソケットが受信したけれどもユーザープログラムに渡されなかったパケットの総バイト数です。`Send-Q` 列は、ソケットが送信したけれども受け入れられなかったパケットの総バイト数です。残りの列は、プロトコル（`Proto` ないし `Netid`）、ローカル側ソケットのアドレスとポート番号（`Local Address` ないし `Local Address:Port`）、リモート側ソケットのアドレスとポート番号（`Foreign Address` ないし `Peer Address:Port`）、状態（`State`）、ソケットが属しているプログラム（`PID/Program name` ないし `Process`）です。

演習

1. ICMPエコーを `learning.lpi.org` に送信してください。

2. `8.8.8.8` への経路を明らかにしてください。

3. TCPの80番ポートで待ち受けているプロセスが存在するかどうかを確かめてください。

4. 22番ポートで接続を待ち受けているプロセス（PIDだけでなくプログラム名も）を特定してください。

5. `somehost.example.com` に至る経路全体での最小のMTUを明らかにしてください。

発展演習

1. netcat（nc）でウェブサーバーにHTTPリクエストを送信することができます。`learning.lpi.org` の80番ポートに `/index.html` の取得を要求するHTTPリクエストを送信してください。

2. pingが失敗する理由を思いつく限り挙げてください。

3. ネットワークインターフェイスを行き来するパケットを確認したり記録したりできるツールの名前を挙げてください。

4. `traceroute` で使用するインターフェイスを指定するオプションは何ですか？

5. `traceroute` でMTUを出力することはできますか？

まとめ

ネットワークは、通常、システムの起動スクリプトか、NetworkManagerのようなヘルパーが構成します。多くのディストリビューションでは、起動スクリプトが使用する構成ファイルを編集するツールが付属しています。詳細についてはお使いのディストリビューションのドキュメントを参照してください。

手動でネットワークを構成できるようになれば、トラブルシューティングを効果的に行えます。また、バックアップからの復元や新しいハードウェアへの移行といった、起動スクリプトやヘルパーが存在しない最小限の環境での作業にも役立ちます。

このレッスンで取り上げたユーティリティには、ここで説明したよりも多くの機能があります。各ユーティリティのマニュアルページをめくり、オプションに習熟する価値があります。`ss` と `ip` は新しいコマンドで、その他のコマンドは、まだ広く用いられているものの、古いツールです。

このレッスンで紹介したツールに慣れるには練習あるのみです。それなりのメモリ（RAM）を搭載したコンピュータを使っているなら、仮想マシンで仮想ネットワークを構築して練習できます。仮想マシンが3台あれば、練習には充分です。

このレッスンでは、次のコマンドを取り上げました。

`ping`、`ping6`
: ICMPパケットを伝送します。ネットワークの疎通確認に使えます。

`traceroute`、`traceroute6`
: ネットワークの経路を追跡し、ネットワークの接続具合を明らかにします。

`tracepath`、`tracepath6`
: ネットワークの経路に加えてMTUを明らかにします。

`nc`
: 任意の接続を作成し、ネットワーク接続を確かめられます。利用できるサービスやデバイスの探索（ポートスキャン）にも使えます。

`netstat`
: 開いているネットワーク接続と統計情報を表示する古いツールです。

`ss`
: 開いているネットワーク接続と統計情報を表示する新しいツールです。

演習の解答

1. ICMPエコーを `learning.lpi.org` に送信してください。

   `ping` ないし `ping6` を使います。

   ```
   $ ping learning.lpi.org
   ```

   または

   ```
   $ ping6 learning.lpi.org
   ```

2. `8.8.8.8` への経路を明らかにしてください。

   `tracepath` ないし `traceroute` コマンドを使います。

   ```
   $ tracepath 8.8.8.8
   ```

   または

   ```
   $ traceroute 8.8.8.8
   ```

3. TCPの80番ポートで待ち受けているプロセスが存在するかどうかを確かめてください。

   `ss` を使うなら次のコマンドを実行します。

   ```
   $ ss -ln | grep ":80"
   ```

   `netstat` を使うなら次のコマンドを実行します。

   ```
   $ netstat -ln | grep ":80"
   ```

   試験範囲のこの単元では取り上げられてていませんが、`lsof` を使うなら次のコマンドを実行します。

   ```
   # lsof -Pi:80
   ```

4. 22番ポートで接続を待ち受けているプロセス（PIDだけでなくプログラム名も）を特定してください。

   複数の方法があります。前問の解答で示したのと同じやり方でポート番号を22にして `lsof` を使うのが一つの方法です。`-p` オプションを指定して `netstat` ないし `ss` を実行する方法もありま

す。ただし、`netstat` は古いツールです。

```
# netstat -lnp | grep ":22"
```

`netstat` と `ss` のオプションは同じです。

```
# ss -lnp | grep ":22"
```

5. `somehost.example.com` に至る経路全体での最小のMTUを明らかにしてください。

   `tracepath` コマンドを実行します。

```
$ tracepath somehost.example.com
```

発展演習の解答

1. netcat（nc）でウェブサーバーにHTTPリクエストを送信することができます。`learning.lpi.org`の80番ポートに`/index.html`の取得を要求するHTTPリクエストを送信してください。

   `nc` で `learning.lpi.org` の80番ポートに接続してから、HTTPリクエストラインとヘッダーと空行をターミナルに入力します。（訳注：以下の出力例と異なる出力になることがありますが、HTTPレスポンスが返ってきていれば、正常に実行できています。）

   ```
   $ nc learning.lpi.org 80
   GET /index.html HTTP/1.1
   HOST: learning.lpi.org

   HTTP/1.1 302 Found
   Location: https://learning.lpi.org:443/index.html
   Date: Wed, 27 May 2020 22:54:46 GMT
   Content-Length: 5
   Content-Type: text/plain; charset=utf-8

   Found
   ```

2. pingが失敗する理由を思いつく限り挙げてください。

   いろいろな理由が考えられます。以下に列挙します。

   - リモートホストがダウンしている。
   - ルーターのアクセスコントロールがpingをブロックしている。
   - リモートホストのファイアウォールがpingをブロックしている。
   - ホスト名またはIPアドレスが間違っている。
   - 名前解決が間違っている。
   - ローカルマシンのネットワーク構成に問題がある。
   - ローカルマシンのファイアウォールがpingをブロックしている。
   - リモートホストのネットワーク構成に問題がある。
   - ローカルマシンのネットワークインターフェイスが切れている。
   - リモートマシンのネットワークインターフェイスが切れている。
   - ローカルマシンとリモートマシンとの間にあるネットワークコンポーネント（スイッチ、ケーブル、ルーターなど）に問題がある。

3. ネットワークインターフェイスを行き来するパケットを確認したり記録したりできるツールの名前を挙げてください。

   `tcpdump` や `wireshark` などです。

4. `traceroute` で使用するインターフェイスを指定するオプションは何ですか？

   `-i` オプションです。たとえば次のように実行します。

```
$ traceroute -i eth2 learning.lpi.org
traceroute -i eth2 learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 60 byte packets
...
```

5. `traceroute` でMTUを出力することはできますか？

   `--mtu` オプションを指定するとMTUを出力します。

```
# traceroute -I --mtu  learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 65000 byte packets
 1  047-132-144-001.res.spectrum.com (47.132.144.1)  9.974 ms F=1500  10.476 ms  4.743 ms
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  8.697 ms  9.963 ms  10.321 ms
...
```

109.4 クライアント側のDNSを設定する

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 109.4

総重量

2

主な知識分野

- リモートDNSサーバーを問い合わせる。
- ローカル名解決を構成し、リモートDNSサーバーを使用する。
- 名前解決が行われる順序を変更する。
- 名前解決に関するエラーをデバッグする。
- systemd-resolvedの知識。

用語とユーティリティ

- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- `host`
- `dig`
- `getent`

109.4 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	109 ネットワークの基礎
Objective:	109.4 クライアント側DNSの構成
Lesson:	1 of 1

はじめに

このレッスンでは、名前解決のクライアント側での構成と、CLI（コマンドラインインターフェイス）で名前解決を行うツールを取り上げます。

IPアドレス、UID、GIDなどの数値を覚えなければならないというのでは不便です。名前解決のサービスは、覚えやすい名前を数値に、あるいは逆に数値を名前に、変換します。このレッスンではホスト名に焦点を当てますが、ユーザー名、グループ名、ポート番号などでも似たような仕組みで名前解決を行います。

名前解決の仕組み

名前解決プログラムは、たいてい、標準Cライブラリ（LinuxシステムではGNUプロジェクトのglibc）に含まれている関数を利用しています。その関数は、まず、`/etc/nsswitch.conf` ファイルを読み取り、名前解決をどのような方法で行うかを決めます。このレッスンではホストの名前解決に焦点を当てますが、ほかの名前解決でも同様で、`/etc/nsswitch.conf` を読み取って、そこに指定されている方法で名前解決を行います。`/etc/nsswitch.conf` はプラグインをサポートしていますから、次にどういう動作をするかはさまざまです。関数が名前解決を終えると、その結果（名前または数値）を返します。

DNSクラス

DNSクラスには、IN、HS、CHの3種類あります。このレッスンでは、INのDNSクエリだけを扱います。INはTCP/IPスタックを使うインターネットアドレスです。CHはChaosNetで、今はもう使われていないネットワークテクノロジーです。HSはHesiodで、/etc/passwdや/etc/groupの情報をDNSで提供します（訳注：IN以外は古く、見かけることもまずありません）。

/etc/nsswitch.conf ファイル

このファイルについて理解するには、マニュアルページを読むのが早道です。たいていのシステムでは man nsswitch.conf コマンドを実行すれば読めますし、そのコマンドで読めなければ https://man7.org/linux/man-pages/man5/nsswitch.conf.5.html でも読めます。

以下はマニュアルページから抜粋した、/etc/nsswitch.conf の例です。

```
        passwd:         compat
        group:          compat
        shadow:         compat

        hosts:          dns [!UNAVAIL=return] files
        networks:       nis [NOTFOUND=return] files
        ethers:         nis [NOTFOUND=return] files
        protocols:      nis [NOTFOUND=return] files
        rpc:            nis [NOTFOUND=return] files
        services:       nis [NOTFOUND=return] files
        # This is a comment. It is ignored by the resolution functions.
```

このファイルは列形式になっています。左端の列は名前データベースの種類です。残りの列は名前解決の方法を示します。名前解決の方法は左から順番に読んでいきます。[] 内は、その直前の方法（[] の左隣に記載されている方法）に適用される条件ロジックです。

learning.lpi.org というホストの名前解決を例に取って説明します。しかるべきCライブラリ（おそらくは gethostbyname）を呼び出すと、まず /etc/nsswitch.conf を読み取ります。ホストの名前解決ですから、hosts で始まる行を探します。その行には dns と書かれていますので、名前解決にDNSを使用します。次の列には [!UNAVAIL=return] と書かれているため、DNSが利用不可（unavailable）でなければ、そこで終了し、次のソース（方法）を試みることはしません。言い換えると、DNSが利用可能であれば、名前解決ができなかったとしても、そこで打ち止めだということです。DNSが利用不可であれば、次のソース、上記の記載例では files に移ります。

[result=action] という形式の列は、その直前の方法（[] の左隣に記載されている方法）が result であれば、action を実行するという意味になります。result の前に ! がついていると、result でなければ action を実行するという、反対の意味になります。result と action に何を記載できるかはマニュアルページに書かれています。

次はポート番号の名前解決を行う例を考えてみましょう。services の行を読みます。最初のソースはNIS（Network Information Service）です。これはイエローページ（yellow pages）とも呼ばれる、ユーザーなどを中央コンピュータが一元管理する古いサービスです。セキュリティの脆弱性があるため現在では使われることがめったにありません。その次の列は [NOTFOUND=return] ですから、名前解決に成功したけれどもそのポート番号に対応するサービス名が見つからなかった場合はそこで名前解決を終えることになります。[] 内の条件が満たされない場合（NISによる名前解決ができなかった場合など）は、ローカルファイル（files）による名前解決に移ります。

の右側に書かれた部分はコメントで、名前解決には使用されません。

/etc/resolv.conf ファイル

/etc/resolv.conf ファイルでは、DNSによるホストの名前解決方法を構成します。ディストリビュー

ションによっては、起動スクリプトやデーモン、その他のツールが、このファイルを自動的に生成することがあります。手動でこのファイルを編集する際には、そのことに注意して、必要に応じてお使いのディストリビューションとネットワーク構成ツールのドキュメントを確認してください。NetworkManagerなどのツールで自動的に生成された `/etc/resolv.conf` ファイルには、手動による変更は上書きされる旨のコメントが記載されています。

`/etc/nsswitch.conf` と同様に、`/etc/resolv.conf` ファイルにもマニュアルページが存在します。`man resolv.conf` を実行するか、https://man7.org/linux/man-pages/man5/resolv.conf.5.html を参照してください。

このファイルも列形式になっていて、左端の列がオプション名、残りの列がそのオプションの値です。

最も重要なオプションは `nameserver` です。DNSサーバーのIPアドレス（IPv4、IPv6）を指定します。現時点では、`nameserver` を最大3つまで指定できます。`/etc/resolv.conf` に `nameserver` オプションが存在しなければ、デフォルトでローカルマシンのネームサーバーを使用します。

以下は一般的な構成例です。

```
search lpi.org
nameserver 10.0.0.53
nameserver fd00:ffff::2:53
```

`search` オプションを指定すると、デフォルトのドメイン名を省略して名前解決を行えます。上の例では、`lpi.org` というドメインが指定されていますから、ドメイン部のないホスト名の名前解決を行う時に、`.lpi.org` が付加されます。例えば、`learning` というホスト名の名前解決を行うと、レゾルバが `learning.lpi.org` を検索します。`search` オプションでは、最大6つまでドメインを指定できます。

`search` とほぼ同じ働きの `domain` という古いオプションもあり、今でもたまに見かけることがあります。使用は推奨されていませんので、自分で設定する場合には `search` を使用しましょう。

レゾルバ（クライアント側で名前解決を行うためのライブラリ）の動作を変更するオプションがいくつかあります。`options` キーワードに続けて項目名を書き、項目によってはさらに `:` に続けて値を書きます。以下の例では、タイムアウト（timeout）という、ネームサーバーの応答を待つ秒数を設定する項目に、3という値を設定しています。

```
options timeout:3
```

`resolv.conf` にはまだほかにもオプションがありますが、ここで紹介したものを知っておけば足りるでしょう。

/etc/hosts ファイル

`/etc/hosts` ファイルは、ホスト名からIPアドレス、あるいは逆にIPアドレスからホスト名への対応を調べるために使います。IPv4とIPv6の両方を設定できます。左端の列がIPアドレスで、残りの列はそのIPアドレスに対応する名前（ホスト名）です。一般的には、`localhost` とループバックアドレスなど、DNSによる名前解決ができない、あるいはふさわしくないホスト名とIPアドレスを対応づけるために `/etc/hosts` を利用します。以下の実例では、そのようなIPアドレスから重要なものをいくつか定義しています。

```
127.0.0.1       localhost
127.0.1.1       proxy
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

10.0.0.1        gateway.lpi.org gateway gw
fd00:ffff::1    gateway.lpi.org gateway gw

10.0.1.53       dns1.lpi.org
fd00:ffff::1:53 dns1.lpi.org
10.0.2.53       dns2.lpi.org
fd00:ffff::2:53 dns2.lpi.org
```

systemd-resolved

systemdには `systemd-resolved` というサービスがあります。通常のDNSクライアントの機能に加えて、mDNSやLLMNRなどの補助的なプロトコルもサポートしています。`systemd-resolved` が起動すると、`127.0.0.53` でDNSリクエストを待ち受けます。そこで完結するのではなく、`/etc/systemd/resolv.conf` や `/etc/resolv.conf` で構成されたサーバーに問い合わせます。`/etc/nsswitch.conf` の `hosts` 行に `resolve` と書けば、`systemd-resolved` を使用できます。ディストリビューションによっては、デフォルトで `systemd-resolved` パッケージが入っていないことがあります。

名前解決ツール

Linuxユーザーが利用できる名前解決ツールはたくさんあります。このレッスンでは、3つのツールを取り上げます。`getent` では、実際のリクエストがどのように名前解決を行うかを確認できます。`host` はシンプルなDNSクエリを実行するのに便利です。`dig` を使えば複雑なDNSの操作を行えるので、DNSサーバーのトラブルシューティングに役立ちます。

getent コマンド

`getent` は、名前データベースから取得したエントリを表示するユーティリティです。`/etc/nsswitch.conf` で構成できるすべての名前解決に対応しています。

`getent` に続けて名前解決を行うデータベースの種類と、任意の個数の調べるエントリを指定して実行します。調べるエントリを指定せずに実行すると、その種類のエントリを表示できるだけすべて表示します。

```
$ getent hosts
127.0.0.1       localhost
127.0.1.1       proxy
10.0.1.53       dns1.lpi.org
10.0.2.53       dns2.lpi.org
127.0.0.1       localhost ip6-localhost ip6-loopback
$ getent hosts dns1.lpi.org
fd00:ffff::1:53 dns1.lpi.org
```


glibcのバージョン2.2.5（2006年リリース）から、`-s` オプションでデータソースを指定できるようになりました。以下に例を示します。

```
$ getent -s files hosts learning.lpi.org
::1             learning.lpi.org
$ getent -s dns hosts learning.lpi.org
208.94.166.198  learning.lpi.org
```

`host` コマンド

`host` はDNSエントリを調べるシンプルなプログラムです。オプションなしで実行すると、名前が与えられた場合はAレコードとAAAAレコードとMXレコードを、IPアドレス（IPv4、IPv6）が与えられた場合はPTRレコードを表示します。

```
$ host wikipedia.org
wikipedia.org has address 208.80.154.224
wikipedia.org has IPv6 address 2620:0:861:ed1a::1
wikipedia.org mail is handled by 10 mx1001.wikimedia.org.
wikipedia.org mail is handled by 50 mx2001.wikimedia.org.
$ host 208.80.154.224
224.154.80.208.in-addr.arpa domain name pointer text-lb.eqiad.wikimedia.org.
```

`-t` オプションに続けてDNSのリソースレコードの種類を指定できます。

```
$ host -t NS lpi.org
lpi.org name server dns1.easydns.com.
lpi.org name server dns3.easydns.ca.
lpi.org name server dns2.easydns.net.
$ host -t SOA lpi.org
lpi.org has SOA record dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300
```

`/etc/resolv.conf` に指定されていないネームサーバーに宛てて問い合わせることもできます。使用するネームサーバーのIPアドレスないし名前を、`host` コマンドの最後の引数に指定します。

```
$ host -t MX lpi.org dns1.easydns.com
Using domain server:
Name: dns1.easydns.com
Address: 64.68.192.10#53
Aliases:

lpi.org mail is handled by 10 aspmx4.googlemail.com.
lpi.org mail is handled by 10 aspmx2.googlemail.com.
lpi.org mail is handled by 5 alt1.aspmx.l.google.com.
lpi.org mail is handled by 0 aspmx.l.google.com.
lpi.org mail is handled by 10 aspmx5.googlemail.com.
lpi.org mail is handled by 10 aspmx3.googlemail.com.
lpi.org mail is handled by 5 alt2.aspmx.l.google.com.
```

dig コマンド

dig もDNSサーバーに問い合わせるツールですが、host よりもはるかに冗長な結果を出力します。dig はデフォルトでAレコードを問い合わせます。IPアドレスないしホスト名を調べるだけでしたら、dig の出力はあまりにも冗長でしょう。DNSサーバー構成のトラブルシューティングにより適しています。

```
$ dig learning.lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63004
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ca7a415be1cec45592b082665ef87f3483b81ddd61063c30 (good)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.   600 IN  A   208.94.166.198

;; AUTHORITY SECTION:
lpi.org.        86400   IN  NS  dns2.easydns.net.
lpi.org.        86400   IN  NS  dns1.easydns.com.
lpi.org.        86400   IN  NS  dns3.easydns.ca.

;; ADDITIONAL SECTION:
dns1.easydns.com.   172682  IN  A   64.68.192.10
dns2.easydns.net.   170226  IN  A   198.41.222.254
dns1.easydns.com.   172682  IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   170226  IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 135 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Sun Jun 28 07:29:56 EDT 2020
;; MSG SIZE  rcvd: 266
```

ご覧のとおり、dig は大量の情報を表示します。出力はセクションに分かれています。最初のセクションでは、インストールされている dig のバージョン、送信されたクエリ、コマンド実行時に使われたオプション、クエリと応答の情報が示されます。

次のセクションでは、用いられたDNS拡張メカニズム（EDNS）とクエリの情報が示されます。上の例では、DNSクッキーが用いられたこと、learning.lpi.org のAレコードを問い合わせたことが示されています。

その次のセクションは問い合わせ結果です。2列目の数字は、そのリソースレコードのTTL（単位は秒）です。

出力の残りの部分は、そのドメインのネームサーバーについての情報です。NSレコードと、NSレコード内のサーバーのAレコードおよびAAAAレコードです。

`host` と同様に、`-t` オプションに続けてリソースレコードの種類を指定できます。

```
$ dig -t SOA lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> -t SOA lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16695
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 185c67140a63baf46c4493215ef8906f7bfbe15bdca3b01a (good)
;; QUESTION SECTION:
;lpi.org.           IN  SOA

;; ANSWER SECTION:
lpi.org.        600 IN  SOA dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300

;; AUTHORITY SECTION:
lpi.org.        81989   IN  NS  dns1.easydns.com.
lpi.org.        81989   IN  NS  dns2.easydns.net.
lpi.org.        81989   IN  NS  dns3.easydns.ca.

;; ADDITIONAL SECTION:
dns1.easydns.com.   168271  IN  A   64.68.192.10
dns2.easydns.net.   165815  IN  A   198.41.222.254
dns3.easydns.ca.    107 IN  A   64.68.196.10
dns1.easydns.com.   168271  IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   165815  IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 94 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Sun Jun 28 08:43:27 EDT 2020
;; MSG SIZE  rcvd: 298
```

digには、サーバーへのクエリと応答の出力を調整するたくさんのオプションがあります。`+` 記号で始まるオプションです。`short` オプションを指定すると、結果だけを出力します。

```
$ dig +short lpi.org
65.39.134.165
$ dig +short -t SOA lpi.org
dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300
```

`nocookie` オプションを指定すると、DNSクッキーを使いません。

```
$ dig +nocookie -t MX lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> +nocookie -t MX lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47774
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;lpi.org.           IN  MX

;; ANSWER SECTION:
lpi.org.        468 IN  MX  0 aspmx.l.google.com.
lpi.org.        468 IN  MX  10 aspmx4.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx5.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx2.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx3.googlemail.com.
lpi.org.        468 IN  MX  5 alt2.aspmx.l.google.com.
lpi.org.        468 IN  MX  5 alt1.aspmx.l.google.com.

;; AUTHORITY SECTION:
lpi.org.        77130   IN  NS  dns2.easydns.net.
lpi.org.        77130   IN  NS  dns3.easydns.ca.
lpi.org.        77130   IN  NS  dns1.easydns.com.

;; ADDITIONAL SECTION:
dns1.easydns.com.   76140   IN  A   64.68.192.10
dns2.easydns.net.   73684   IN  A   198.41.222.254
dns1.easydns.com.   76140   IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   73684   IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 2 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Mon Jun 29 10:18:58 EDT 2020
;; MSG SIZE  rcvd: 389
```

演習

1. 次のコマンドが何を行うかを説明してください。

```
$ getent group openldap
```

2. `getent`、`host`、`dig` の間の、一番大きな違いは何ですか？

3. `dig` ないし `host` で、取得するリソースレコードの種類を指定するオプションは何ですか？

4. `/etc/hosts` の記載として正しいのは、次のうちどちらでしょう？

`::1 localhost`	
`localhost 127.0.0.1`	

5. `getent` で調べるデータソースを指定するオプションは何ですか？

発展演習

1. 以下の `/etc/resolv.conf` ファイルを手動で編集したとしたら、どうなるでしょうか？

```
# Generated by NetworkManager
nameserver 192.168.1.20
```

手動で加えた変更がNetworkManagerによって上書きされます。	
手動で加えた変更に応じてNetworkManagerが構成を更新します。	
手動で変更を加えてもシステムの動作には影響しません。	
NetworkManagerが無効になります。	

2. `/etc/nsswitch.conf` ファイル内の次の行はどういう意味ですか？

```
hosts: files [SUCCESS=continue] dns
```

3. 次の `/etc/resolv.conf` ファイルでは、DNSによる名前解決ができませんでした。なぜでしょうか？

```
search lpi.org
#nameserver fd00:ffff::1:53
#nameserver 10.0.1.53
```

4. `dig +noall +answer +question lpi.org` コマンドを実行すると、どうなりますか？

5. `dig` のデフォルトの動作を変えて、実行時にコマンドラインでオプションを指定せずにすむようにするには、どうすればよいですか？

まとめ

`getent` は、レゾルバの要求結果を確認するツールです。`host` は、シンプルなDNSクエリを行い、使いやすく出力がわかりやすいです。`dig` は、DNSクエリの詳細な情報や調整が必要な場合に最適です。

Linuxでは、共有ライブラリプラグインを追加してレゾルバの動作を構成できるので、さまざまな種類の名前解決を行えます。`getent` はレゾルバライブラリを使用して名前解決を行うのに対し、`host` と `dig` はDNSサーバーに問い合わせます。

`/etc/nsswitch.conf` ファイルでレゾルバの動作を構成します。データソースを変更でき、複数のソースについてちょっとした条件ロジックを記述できます。

`/etc/resolv.conf` ファイルでDNSを構成します。何らかのツールがこのファイルを管理しているディストリビューションが多いですから、手動での変更が上書きされてしまう場合にはシステムのドキュメントを読んでください。

`/etc/hosts` ファイルではホスト名とIPアドレスを対応づけます。`localhost` など、DNSでは名前解決できないホスト名を定義するのが典型的な利用法です。

このレッスンで取り上げた構成ファイルでは、`#` の右側に書かれた部分はコメントとして無視されます。

演習の解答

1. 次のコマンドが何を行うかを説明してください。

```
$ getent group openldap
```

   `/etc/nsswitch.conf` ファイルを読み取り、`group` で始まる行に記載されたソースから `openldap` グループを調べ、見つかればその結果を表示します。

2. `getent`、`host`、`dig` の間の、一番大きな違いは何ですか？

   `getent` はレゾルバライブラリを用いて名前解決を行うのに対し、`host` と `dig` はDNSクエリにより名前解決を行うという点が、一番大きな違いです。`getent` を使うと、`/etc/nsswitch.conf` ファイルに記載された名前解決ライブラリの構成をトラブルシューティングできます。`host` と `dig` は、DNSレコードを調べます（`dig` のほうが `host` よりも冗長な結果を出力します）。

3. `dig` ないし `host` で、取得するリソースレコードの種類を指定するオプションは何ですか？

   `-t` オプションです。

4. `/etc/hosts` の記載として正しいのは、次のうちどちらでしょう？

`::1 localhost`	◯
`localhost 127.0.0.1`	

   `::1 localhost` が正しいです。左端の列は常にIPアドレス（IPv4、IPv6）です。

5. `getent` で調べるデータソースを指定するオプションは何ですか？

   `-s` オプションです。例えば次のように実行します。

```
$ getent -s files hosts learning.lpi.org
192.168.10.25   learning.lpi.org
$ getent -s dns hosts learning.lpi.org
208.94.166.198  learning.lpi.org
```

発展演習の解答

1. 以下の `/etc/resolv.conf` ファイルを手動で編集したとしたら、どうなるでしょうか？

```
# Generated by NetworkManager
nameserver 192.168.1.20
```

手動で加えた変更がNetworkManagerによって上書きされます。	○
手動で加えた変更に応じてNetworkManagerが構成を更新します。	
手動で変更を加えてもシステムの動作には影響しません。	
NetworkManagerが無効になります。	

2. `/etc/nsswitch.conf` ファイル内の次の行はどういう意味ですか？

```
hosts: files [SUCCESS=continue] dns
```

ホストの名前解決では `/etc/hosts` ファイルを最初に調べ、次にDNSを調べます。`/etc/hosts` ファイルとDNSの両方で名前解決ができる場合には、DNSでの名前解決が採用されます。

3. 次の `/etc/resolv.conf` ファイルでは、DNSによる名前解決ができませんでした。なぜでしょうか？

```
search lpi.org
#nameserver fd00:ffff::1:53
#nameserver 10.0.1.53
```

DNSサーバーを記述した行が2行ともコメントアウトされていて、ローカルホストでネームサーバーが起動していないからです。

4. `dig +noall +answer +question lpi.org` コマンドを実行すると、どうなりますか？

`lpi.org` のAレコードを調べ、クエリと応答のみを表示します。

5. `dig` のデフォルトの動作を変えて、実行時にコマンドラインでオプションを指定せずにすむようにするには、どうすればよいですか？

ホームディレクトリに `.digrc` ファイルを作成します。

Linux
Professional
Institute

課題 110: セキュリティ

 | |

110.1 セキュリティ管理タスクを実行する

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 110.1

総重量

3

主な知識分野

- suid / sgidビットがセットされたファイルを見つけるシステムを監査する。
- ユーザーパスワードとパスワードエージング情報の設定または変更。
- nmapとnetstatを使ってシステム上のオープンポートを発見できること。
- ユーザーのログイン、プロセス、およびメモリの使用に関する制限を設定する。
- どのユーザーがシステムにログインしたことがあるか、または現在ログインしているユーザーを判別する。
- 基本的なsudoの設定と使い方。

用語とユーティリティ

- `find`
- `passwd`
- `fuser`
- `lsof`
- `nmap`
- `chage`
- `netstat`
- `sudo`
- `/etc/sudoers`
- `su`
- `usermod`
- `ulimit`

- `who`, `w`, `last`

110.1 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	110 セキュリティ
Objective:	110.1 セキュリティ管理タスクを実行する
Lesson:	1 of 1

はじめに

システム管理ではセキュリティへの配慮が欠かせません。ファイルの特別なパーミッション、ユーザーのパスワード、開いているポートとソケット、システムリソースの制限、ログインしているユーザーの把握、`su` と `sudo` による権限昇格など、気をつけなければならないことがたくさんあります。このレッスンでは、これらのセキュリティに関する留意事項を概観します。

SUIDとSGIDが設定されたファイルを確認する

Linuxシステムのファイルには、読み書き実行 という通常のパーミッションに加えて、SUID と SGID という特別なパーミッションがあります。

SUIDが設定されたファイルは、所有者の権限で実行されます。数値表記では `4000`、記号表記では所有者の 実行 パーミッションの有無に応じて `s` ないし `S` で表されます。SUIDパーミッションが設定されているファイルの代表例は `passwd` です。（訳注：ディレクトリにSUIDを設定しても何も起きません。）

```
carol@debian:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 63736 jul 27  2018 /usr/bin/passwd
```

所有者に対するパーミッション `rws` の小文字の `s` が、所有者の 実行 パーミッションとSUIDの両方が設定されていることを示しています。大文字の `S` なら所有者の 実行 パーミッションがなく、SUIDのみが設定されていることを意味します。

NOTE 次節で詳しく説明しますが、`passwd` はユーザーのパスワードを設定／変更するために

> 主としてrootユーザーが使います。SUIDが設定されていますから、一般ユーザーも自分自身のパスワードを変更することができるのです。

SGIDはファイルとディレクトリのどちらにでも設定できます。ファイルに設定すると、そのファイルの所有グループの権限で実行されるようになります。これはSUIDと似ています。ディレクトリに設定すると、そのディレクトリ内に作成したファイルの所有グループが、そのディレクトリの所有グループと同じになります。SUIDと同様に、記号表記では所有グループの　実行　パーミッションの有無に応じて　`s`　ないし `S` で表されます。数値表記では `2000` で表されます。

`chmod`　で既存のディレクトリにSGIDを設定できます。数値表記なら、通常のパーミッションを表す3桁の数字の先頭に　`2`　を付け足します。以下の例では、パーミッションが　`755`　のディレクトリにSGIDを設定しています。

```
carol@debian:~$ ls -ld shared_directory
drwxr-xr-x 2 carol carol 4096 may 30 23:55 shared_directory
carol@debian:~$ sudo chmod 2755 shared_directory/
carol@debian:~$ ls -ld shared_directory
drwxr-sr-x 2 carol carol 4096 may 30 23:55 shared_directory
```

SUIDとSGIDの一方あるいは両方が設定されているファイルを検索するには、`find`　コマンドの　`-perm`　オプションを使います。数値表記でも記号表記でも検索できます。そのまま、`-`（ダッシュ）付き、`/`（スラッシュ）付きの3通りの検索方法があります。それぞれの検索方法の意味は次のとおりです。

`-perm 数値表記` または、`-perm 記号表記`
　指定したパーミッションに一致するファイルを検索します。

`-perm -数値表記` または、`-perm -記号表記`
　少なくとも指定したパーミッションが全部あるファイルを検索します。

`-perm /数値表記` または、`-perm /記号表記`
　指定したパーミッションのうち1つでも設定されているファイルを検索します。

例えば、現在の作業ディレクトリからSUID　だけ　が設定されているファイルを検索するなら、次のコマンド（`find . -perm 4000`）を実行します。

```
carol@debian:~$ find . -perm 4000
carol@debian:~$ touch file
carol@debian:~$ chmod 4000 file
carol@debian:~$ find . -perm 4000
./file
```

最初にfindコマンドを実行したときにはSUIDだけが設定されているファイルは存在しなかったので、検索結果が表示されるようにSUIDだけが設定されているファイルを作ってみました。記号表記でも同じように検索できます。

```
carol@debian:~$ find . -perm u+s
./file
```

`/usr/bin/` ディレクトリからSUIDが設定されているファイル（他のパーミッションはどうなっていてもよい）を検索するなら、次のコマンドを実行します。数値表記でも記号表記でも動作は同じです。

```
carol@debian:~$ sudo find /usr/bin -perm -4000
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
carol@debian:~$ sudo find /usr/bin -perm -u+s
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
```

SGIDが設定されているファイル（他のパーミッションはどうなっていてもよい）を検索するなら、`find /usr/bin/ -perm -2000` か `find /usr/bin/ -perm -g+s` を実行します。

SUIDとSGIDのいずれか一方でも設定されているファイルを検索するなら、`/` を使います（`/6000` の `6` はSUIDの `4` とSGIDの `2` を足した `6` です）。

```
carol@debian:~$ sudo find /usr/bin -perm /6000
/usr/bin/dotlock.mailutils
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/wall
/usr/bin/ssh-agent
/usr/bin/chage
/usr/bin/dotlockfile
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/expiry
/usr/bin/sudo
/usr/bin/bsd-write
/usr/bin/crontab
/usr/bin/su
```

パスワードを管理する

先に述べたように、`passwd` で一般ユーザーも自分自身のパスワードを変更できます。`-S` ないし `--status` オプションを指定すると、自分のユーザーアカウントについての情報を得られます。

```
carol@debian:~$ passwd -S
carol P 12/07/2019 0 99999 7 -1
```

出力される7つのフィールドを説明します。

`carol`
ユーザー名です。

`P`
`P` はそのユーザーが有効なパスワードを設定していることを示します。`L` はパスワードがロックされていることを、`NP` はパスワードを設定していないことを示します。

`12/07/2019`
最後にパスワードを変更した日付です。

`0`
パスワード変更不可能期間（パスワードを変更してから次に変更するまでの最小期間）です。単位は日です。`0` はいつでもパスワードを変更できることを意味します。（訳注：ユーザーがパスワードを変更してすぐに前のパスワードに戻すことを防ぐために設定することがあります。）

`99999`
パスワード有効期間です。単位は日です。`99999` はパスワードが失効せず無期限で有効であることを意味します。

`7`
パスワード失効前警告期間（パスワードの有効期間が終了して失効するどれくらい前に警告を受けるか）です。単位は日です。

`-1`
パスワード無効化期間（パスワードの有効期間が終了して失効してからユーザーアカウントがロックされるまでの期間）です。単位は日です。`-1` はユーザーアカウントをロックしないことを意味します。

rootユーザーで `passwd` コマンドを実行すると、ユーザーアカウントのパスワードを管理できます。`-l` オプションを指定してパスワードをロックする、`-u` オプションを指定してパスワードのロックを解除する（アンロック）、`-e` オプションを指定してパスワードを失効させ次回ログイン時にパスワードの変更を強制する、`-d` オプションを指定してパスワードを削除する、といった具合です。

これらのオプションを試してみるために、ここで `su`（Substitute User）コマンドを紹介します。`su` コマンドを実行すると、ユーザーを変更した新しいシェルを起動します。rootユーザーとして `passwd` コマンドを実行して `carol` のパスワードをロックし、ユーザーを `carol` に切り替えてパスワードがロックされていて（`L` と表示されていて）変更できないことを確認し、rootユーザーに戻って `carol` のパスワードのロックを解除するという例を示します。

```
root@debian:~# passwd -l carol
passwd: password expiry information changed.
root@debian:~# su - carol
carol@debian:~$ passwd -S
carol L 05/31/2020 0 99999 7 -1
carol@debian:~$ passwd
Changing password for carol.
Current password:
passwd: Authentication token manipulation error
passwd: password unchanged
carol@debian:~$ exit
logout
root@debian:~# passwd -u carol
passwd: password expiry information changed.
```

パスワードのロックとロックの解除（アンロック）は、`usermod` コマンドでもできます。

`carol` のパスワードをロックするコマンド

`usermod -L carol` または `usermod --lock carol`

`carol` のパスワードのロックを解除するコマンド

`usermod -U carol` または `usermod --unlock carol`

NOTE

`usermod` コマンドに `-f` ないし `--inactive` オプションを指定すれば、パスワードが失効してからユーザーアカウントがロックされて無効になるまでの日数を設定できます（例：`usermod -f 3 carol`）。

パスワードの有効期間を `passwd` や `usermod` より直接扱うコマンドは `chage`（CHange AGE）です。rootユーザーとして、`-l`（`--list`）オプションとユーザー名を指定して `chage` コマンドを実行すると、そのユーザーのパスワードに関する情報を出力します。一般ユーザーは自分自身の情報しか出力できません。

```
carol@debian:~$ chage -l carol
Last password change                    : Aug 06, 2019
Password expires                        : never
Password inactive                       : never
Account expires                         : never
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

オプションを付けずにユーザー名だけ指定して `chage` を実行すると、対話モードになります。

```
root@debian:~# chage carol
Changing the aging information for carol
Enter the new value, or press ENTER for the default

    Minimum Password Age [0]:
    Maximum Password Age [99999]:
```

```
    Last Password Change (YYYY-MM-DD) [2020-06-01]:
    Password Expiration Warning [7]:
    Password Inactive [-1]:
    Account Expiration Date (YYYY-MM-DD) [-1]:
```

パスワードに関する設定を変更する `chage` のオプションは次のとおりです。

`-m` 日数 ユーザー名 または `--mindays` 日数 ユーザー名
パスワードを変更してから次にパスワードを変更できるまでの最小日数を設定します（例：`chage -m 5 carol`）。`0` に設定するといつでもパスワードを変更できます。

`-M` 日数 ユーザー名 または `--maxdays` 日数 ユーザー名
1つのパスワード使用し続けることができる最長日数を設定します（例：`chage -M 30 carol`）。パスワードを失効させず無期限で有効にするには、`99999` を設定します。

`-d` 日数 ユーザー名 または `--lastday` 日数 ユーザー名
最後にパスワードを変更してからの経過日数を設定します（例：`chage -d 10 carol`）。`0` に設定すると次回ログイン時にパスワードの変更を強制します。

`-W` 日数 ユーザー名 または `--warndays` 日数 ユーザー名
パスワード失効の何日前から警告を行うかを設定します。

`-I` 日数 ユーザー名 または `--inactive` 日数 ユーザー名
パスワードの有効期間が切れてからアカウントがロックされるまでの日数を設定します（例：`chage -I 10 carol`）。`usermod -f` ないし `usermod --inactive` と同じです。設定した日数が経過する前にユーザーがログインしてパスワードを変更しないと、アカウントはロックされます。`0` に設定すると、ユーザーアカウントをロックしません。

`-E` 日付 ユーザー名 または `--expiredate` 日付 ユーザー名
アカウントが無効となる日付（または UNIXエポック からの日数）を設定します。日付は `YYYY-MM-DD` というフォーマットで設定します（例：`chage-E 2050-12-13 carol`）。

NOTE `passwd`、`usermod`、`chage` の詳細やオプションについては、マニュアルページを参照してください。

開いているポートを検出する

開いているポートを監視するには、`lsof`、`fuser`、`netstat`、`nmap` の4つのユーティリティを使います。本節では、これらのユーティリティを1つずつ紹介します。

`lsof`（LiSt Open Files）は、開いているファイルの一覧を表示します。Linuxではあらゆるものがファイルとして扱われますから、これはとても重要なことです。ターミナルで `lsof` とだけ入力して実行すると、通常ファイル、デバイスファイル、ソケットなど、たくさんの結果が出力されるはずです。このレッスンではネットワークポートに焦点を絞ります。`lsof` に `-i` オプションを指定して実行すると、使用中のネットワークポートの一覧を表示します。（訳注：ネットワークポートはファイルではありませんが、ファイルの一種と見なすことができる ソケット と結びつけられて初めて利用可能となります。）

```
root@debian:~# lsof -i
```

```
COMMAND  PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient 357     root    7u  IPv4  13493      0t0  UDP *:bootpc
sshd     389     root    3u  IPv4  13689      0t0  TCP *:ssh (LISTEN)
sshd     389     root    4u  IPv6  13700      0t0  TCP *:ssh (LISTEN)
apache2  399     root    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
apache2  401 www-data    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
apache2  402 www-data    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
sshd     557     root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510 (ESTABLISHED)
sshd     569    carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510 (ESTABLISHED)
```

DHCPで使用される `bootpc` を別にすれば、2種類のサービスが接続を待ち受けていることが、この出力のLISTENという部分からわかります。`ssh` とApacheウェブサーバー（`httpd`）の2種類です。そして、2つのSSHセッションが確立していることが、ESTABLISHEDという部分からわかります。

```
root@debian:~# lsof -i@192.168.1.7
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    557  root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510 (ESTABLISHED)
sshd    569 carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510 (ESTABLISHED)
```

NOTE `-i4` または `-i6` オプションを指定すると、IPv4またはIPv6に絞って表示できます。

`-i` オプション（ないし `-i@ip-address` オプション）に `:port` 引数を渡すと、そのポートに絞って表示できます。

```
root@debian:~# lsof -i :22
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    389  root    3u  IPv4  13689      0t0  TCP *:ssh (LISTEN)
sshd    389  root    4u  IPv6  13700      0t0  TCP *:ssh (LISTEN)
sshd    557  root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510 (ESTABLISHED)
sshd    569 carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510 (ESTABLISHED)
```

カンマ区切り（またはハイフンでの範囲）で複数のポートを対象にすることもできます。

```
root@debian:~# lsof -i@192.168.1.7:22,80
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    705  root    3u  IPv4  13960      0t0  TCP 192.168.1.7:ssh->192.168.1.4:44766 (ESTABLISHED)
sshd    718 carol    3u  IPv4  13960      0t0  TCP 192.168.1.7:ssh->192.168.1.4:44766 (ESTABLISHED)
```

NOTE `lsof` には非常に多くのオプションがあるので、詳しくはマニュアルページを参照してください。

次に `fuser`（File's USER）を説明します。これはファイルのユーザー（どのプロセスがファイルを開いているか）を調べるために使います。ファイルをどのように利用しているかもわかります。例えば、現在の作業ディレクトリを調べるなら、`fuser .` を実行します。`-v`（`--verbose`）オプションを指定して実行すると冗長な情報を得られます。

```
root@debian:~# fuser .
```

```
/root:                580c
root@debian:~# fuser -v .
                     USER        PID ACCESS COMMAND
/root:               root        580 ..c.. bash
```

出力について説明します。（）内に該当する部分を示します。

ファイル
調査対象のファイル（`/root`）

USER 列
そのファイルの所有者（`root`）

PID 列
プロセスID（`580`）

ACCESS 列
ファイルをどのように利用しているかを示すフラグ（`..c..`）

c
カレントディレクトリ

e
実行中のファイル

f
開いているファイル（デフォルトでは表示されません）

F
書き込みのために開いているファイル（デフォルトでは表示されません）

r
ルートディレクトリ

m
共有ライブラリなど、メモリにマップされたファイル

.
プレースホルダー（デフォルトでは表示されません）

COMMAND 列
ファイルにアクセスしているコマンド（`bash`）

`-n`（`--namespace`）オプションを指定すると、ネットワークポート／ソケットについて調べられます。ネットワークプロトコルとポート番号を引数として渡します。Apacheウェブサーバーについて調べるなら、次のコマンドを実行します。

```
root@debian:~# fuser -vn tcp 80
                     USER        PID ACCESS COMMAND
```

```
80/tcp:               root       402 F.... apache2
                      www-data   404 F.... apache2
                      www-data   405 F.... apache2
```

NOTE

`-k`（`--kill`）オプションを指定して `fuser` コマンドを実行すると（例：`fuser -k 80/tcp`）、ファイルにアクセスしているプロセスを強制終了できます。詳しくはマニュアルページを参照してください。

次は `netstat` を説明します。`netstat`（NETwork STATistics）は、ネットワークの状態を幅広く調べられるツールです。

オプションを付けずに `netstat` を実行すると、アクティブなインターネット接続とUnixソケットの情報を表示します。大量に出力されるでしょうから、パイプで `less` につなげたほうがよいです。

```
carol@debian:~$ netstat |less
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.7:ssh         192.168.1.4:55444       ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    10509    /run/systemd/journal/syslog
unix  3      [ ]         DGRAM                    10123    /run/systemd/notify
(...)
```

`-l`（`--listening`）オプションを指定すると、接続を待ち受けているポートとソケットだけを表示します。`-t`（`--tcp`）および `-u`（`--udp`）オプションを指定すると、それぞれTCPだけあるいはUDPだけを表示します（これらのオプションを同時に指定してTCPとUDPだけを表示することもできます）。`-e`（`--extend`）オプションを指定すると、表示される情報が増えます。

```
carol@debian:~$ netstat -lu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
carol@debian:~$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 localhost:smtp          [::]:*                  LISTEN
carol@debian:~$ netstat -lute
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User       Inode
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      root       13729
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN      root       14372
tcp6       0      0 [::]:http               [::]:*                  LISTEN      root       14159
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      root       13740
tcp6       0      0 localhost:smtp          [::]:*                  LISTEN      root       14374
```

```
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*                           root       13604
```

`-l` オプションを指定しなければ、確立している接続だけを表示します。

```
carol@debian:~$ netstat -ute
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User       Inode
tcp        0      0 192.168.1.7:ssh         192.168.1.4:39144       ESTABLISHED root       15103
```

`-n`（`--numeric`）オプションを指定すると、ポートとホストを名前ではなく番号とIPアドレスで表示します。`-n` オプションを加えた下記の実行結果では、先ほどの実行結果で `ssh` と表示されていた部分が `22` になっています。（訳注：ホスト名を表示するとDNS検索を行うので長い時間がかかりがちです。それを避けるためにとてもよく使うオプションです。）

```
carol@debian:~$ netstat -uten
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User       Inode
tcp        0      0 192.168.1.7:22          192.168.1.4:39144       ESTABLISHED 0          15103
```

ここまでもそうしてきたように、オプションを組み合わせて `netstat` を使うと便利です。必要に応じてマニュアルページを参照してください。（訳注：新しいディストリビューションでは、`netstat` ではなく `ss` コマンドを使用することが多いです。）

最後に `nmap`（Network MAPper）を説明します。IPアドレスまたはホスト名を指定して実行すると、開いているポートをスキャンできます。

```
root@debian:~# nmap localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:29 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.58 seconds
```

`nmap` では、複数のホストのポートをスキャンできます。

複数のホスト
スペースで区切ります（例：`nmap localhost 192.168.1.7`）

ホスト範囲
ハイフンで範囲を指定します（例：`nmap 192.168.1.3-20`）

サブネット
ワイルドカードまたはCIDR表記を使います（例：`nmap 192.168.1.*` または `nmap 192.168.1.0/24`）。除外指定もできます（例：`nmap 192.168.1.0/24 --exclude 192.168.1.7`）。

`-p` オプションの後にポート番号ないしサービス名を指定すると、そのポートだけをスキャンします（`nmap -p 22` と `nmap -p ssh` は同じ結果になります）。

```
root@debian:~# nmap -p 22 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:54 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000024s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

カンマで区切るかハイフンで範囲を指定すれば、複数のポートをスキャンできます。

```
root@debian:~# nmap -p ssh,80 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:58 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000051s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

```
root@debian:~# nmap -p 22-80 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:58 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000011s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 57 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.47 seconds
```

その他、`nmap` には次のオプションがあります。

`-F`
よく使われる100のポートを迅速にスキャンします。

-v
冗長な出力を行います（-vv を指定するとさらに冗長な出力になります）。

NOTE nmap はより複雑なスキャンを行えますが、このレッスンでは基本的なスキャンのみを取り上げました。

ユーザーのプロセスやメモリ使用量などを制限する

Linuxシステムのリソースは有限ですから、システム管理者は、OSが適切に動き続けるように、ユーザーが使用できるリソースを制限しなければならないことがあります。そのためには ulimit コマンドを使います。

ulimit に -S オプションを指定すると ソフトリミット を、-H オプションを指定すると ハードリミット を扱います。ハードリミット とはシステム管理者が設定するリミット値で、ソフトリミットとはユーザーごとに設定するリミット値です。ソフトリミットによるリミット値は、ハードリミットによるリミット値を超えることができません。（訳注:LPIC-1レベルではハードリミットを操作することはありません。）

オプションなしで ulimit を実行すると、現在のユーザーのファイルブロック（シェルとその子プロセスが生成できるファイルの最大サイズ）のソフトリミットを表示します。

```
carol@debian:~$ ulimit
unlimited
```

-a オプションを指定して ulimit を実行すると（-Sa オプションを指定しても同じです）、現在のユーザーのすべてのソフトリミットを表示します。-Ha オプションを指定するとすべてのハードリミット表示します。

```
carol@debian:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
(...)
carol@debian:~$ ulimit -Ha
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
(...)
```

リソースを指定するには次のオプションを使います。

-b
ソケットバッファの最大サイズ

-f
シェルとその子プロセスが生成できるファイルの最大サイズ

-l
メモリにロックできる最大サイズ

-m
常駐セットサイズ（プロセスが確保している物理メモリの量、RSS）の最大値

-v
仮想メモリの最大量

-u
1人のユーザが使用できるプロセスの最大数

`ulimit` の後に `-S` ないし `-H` と上記のリソースオプションを指定すると、そのリソースのソフトリミットないしハードリミットを表示します。`-S` も `-H` も指定しなければ、ソフトリミットを表示します。

```
carol@debian:~$ ulimit -u
10000
carol@debian:~$ ulimit -Su
10000
carol@debian:~$ ulimit -Hu
15672
```

`ulimit` の後に `-S` ないし `-H` と上記のリソースオプションを指定し、その後に値を指定すると、新しいリミット値を設定します。値は数値で指定するか、`soft`（現在のソフトリミットの値）、`hard`（現在のハードリミットの値）、`unlimited`（無制限）のいずれかで指定します。`-S` も `-H` も指定しなければ、ソフトリミットとハードリミットの両方を設定します。例として、シェルとその子プロセスが生成できるファイルの最大サイズ（`-f`）を取り上げます。まず、現在のソフトリミットとハードリミットを確認します。

```
root@debian:~# ulimit -Sf
unlimited
root@debian:~# ulimit -Hf
unlimited
```

次に、`unlimited` から `500` ブロックへとリミット値を変更します。`-S` も `-H` も指定しなければ、ソフトリミットとハードリミットの両方を設定します。

```
root@debian:~# ulimit -f 500
root@debian:~# ulimit -Sf
500
root@debian:~# ulimit -Hf
500
```

ソフトリミットを 200 ブロックに変更します。

```
root@debian:~# ulimit -Sf 200
```

```
root@debian:~# ulimit -Sf
200
root@debian:~# ulimit -Hf
500
```

ハードリミットを増やせるのはrootユーザーだけです。ハードリミットを減らしたり、ソフトリミットをハードリミットの値まで増やしたりすることは、一般ユーザーでもできます。新しく設定したリミットを再起動後も有効にするには `/etc/security/limits.conf` ファイルに記載します。このファイルは、システム管理者がユーザーを指定してリミットを設定するためにも使えます。（訳注：limists.conはPAMの設定ファイルであり、LPIC-1の範囲外です。）

NOTE `ulimit` はbashの組み込みコマンドですから、manページはありません。`help ulimit` で簡単なヘルプが表示されます。また、リミット値の設定は、現在のシェルとその子孫にのみ適用されます。

ユーザーのログインを管理する

システム管理者の業務としてユーザーのログイン状況を調べたいことがあります。`last`、`who`、`w` の3つのユーティリティがこの仕事に役立ちます。

`last` はログインしたユーザーの一覧をログイン日時の降順で表示します。

```
root@debian:~# last
carol    pts/0        192.168.1.4      Sat Jun  6 14:25   still logged in
reboot   system boot  4.19.0-9-amd64   Sat Jun  6 14:24   still running
mimi     pts/0        192.168.1.4      Sat Jun  6 12:07 - 14:24  (02:16)
reboot   system boot  4.19.0-9-amd64   Sat Jun  6 12:07 - 14:24  (02:17)
(...)
wtmp begins Sun May 31 14:14:58 2020
```

上の表示例では、直近にログインした2人の状況がわかります。

1. ユーザー `carol` は、ターミナル `pts/0` で、ホスト `192.168.1.4` から、`Sat Jun 6 14:25`（6月6日土曜日14時25分）にセッションを開始し、`still logged in`（現在もログイン中）です。その1分前（`Sat Jun 6 14:24`（6月6日土曜日14時24分）に、システムがカーネル `4.19.0-9-amd64` を使用して起動し、`still running`（現在も稼働中）です。
2. ユーザー `mimi` は、ターミナル `pts/0` で、ホスト `192.168.1.4` から、`Sat Jun 6 12:07`（6月6日土曜日12時07分）にセッションを開始し、`14:24`（14時24分）にログアウトしました（セッションは `02:16`（2時間16分）続きました）。システムは同じ時刻（`Sat Jun 6 12:07`（6月6日土曜日12時07分））にカーネル `4.19.0-9-amd64` を使用して起動し、`14:24`（14時24分）に終了しました（`02:17`（2時間17分）実行されていました）。つまり、ユーザー `mini` はシステムを起動してすぐにログインし、2時間余り作業を行い、すぐにシステムをシャットダウンとしたことが分かります。

NOTE `last` は、専用のログファイル `/var/log/wtmp` から情報を取得しています。そのファイルには `Sun May 31 14:14:58 2020`（2020年5月31日14時14分58秒）からの記録が残されていることが、上の表示例からわかります。

`last` にユーザー名を渡すとそのユーザーの行だけを表示します。

```
root@debian:~# last carol
carol    pts/0        192.168.1.4      Sat Jun  6 14:25   still logged in
carol    pts/0        192.168.1.4      Sat Jun  6 12:07 - 14:24  (02:16)
carol    pts/0        192.168.1.4      Fri Jun  5 00:48 - 01:28  (00:39)
(...)
```

ターミナルを表す2列目について少し説明します。`pts` は Pseudo Terminal Slave（疑似端末スレーブ） で、`tty` は TeleTYpewriter（テレタイプ端末） です。`0` は最初のターミナルであることを示します。一般的に `tty` はシステム本体のディスプレイとキーボード（コンソール）からのログインを示し、`pts` はネットワークからのログインを示します。

NOTE `last` ではなく `lastb` を実行すると、失敗したログインを表示します。

`who` と `w` は、どちらも現在ログインしているユーザーを調べるコマンドです。`who` は現在ログインしているユーザーを表示し、`w` はそのユーザーの活動状況も表示します。

オプションを付けずに `who` を実行すると、ログインしているユーザー、ターミナル、ログイン日時、ホスト名の4列を表示します。

```
root@debian:~# who
carol    pts/0        2020-06-06 17:16 (192.168.1.4)
mimi     pts/1        2020-06-06 17:28 (192.168.1.4)
```

よく使うオプションを紹介します。

`-b`、`--boot`
システムが起動した日時を表示します。

`-r`、`--runlevel`
現在のランレベルを表示します。

`-H`、`--heading`
列見出しを表示します。

`w` は `who` よりも多くの情報を表示します。

```
root@debian:~# w
 17:56:12 up 40 min,  2 users,  load average: 0.04, 0.12, 0.09
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
carol    pts/0    192.168.1.4    17:16    1.00s  0.15s  0.05s sshd: carol [priv]
mimi     pts/1    192.168.1.4    17:28   15:08   0.05s  0.05s -bash
```

1行目に表示されているのは、現在の時刻（`17:56:12`）、システムが起動してからの経過時間（`up 40 min`）、現在ログイン中のユーザー数（`2 users`）、ロードアベレージ（`load average: 0.04, 0.12, 0.09`）です。ロードアベレージは、前から順に、1分、5分、15分の値です。

2行目からは8列表示です。

USER
ログイン中のユーザー名

TTY
そのユーザーが使用しているターミナル

FROM
そのユーザーが使用しているホスト（ホスト名ないしIPアドレス）

LOGIN@
そのユーザーがログインした時間

IDLE
何もしていない（アイドルである）時間

JCPU
そのターミナルから実行されたプロセス（バックグラウンドジョブを含む）が使用したCPU時間の合計

PCPU
現在のプロセス（`WHAT` 列に示されているもの）が使用しているCPU時間

WHAT
現在のプロセスのコマンドライン

`who` と同様に、`w` にユーザー名を渡すとそのユーザーの行だけを表示します。

```
root@debian:~# w mimi
 18:23:15 up  1:07,  2 users,  load average: 0.00, 0.02, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
mimi     pts/1    192.168.1.4      17:28    9:23   0.06s  0.06s -bash
```

基本的な `sudo` の設定と使用法

このレッスンの冒頭で学んだように、`su` を実行して変更後のユーザーのパスワードを入力すると、そのユーザーに切り替えることができます。しかし、rootユーザーのパスワードを他のユーザーと共有することは避けねばなりません。つまり、セキュリティのベストプラクティスとしては、`su` ではなく後述する `sudo` を用いるべきです。`sudo` を学習する前に、ここでは `su` でrootユーザーへの切り替えを試しておきます。

`su` は `su - 変更後のユーザー名` のように実行します。rootユーザーに切り替えるときは変更後のユーザー名を省略できます。（訳注：多くのディストリビューションではrootユーザーが有効化されておらず、rootのパスワードが **決まっていない** ことが少なくありません。その場合は `sudo passwd root` コマンドで、まずrootユーザーのパスワードをセットすれば、以下のコマンドを実行してみることができます。後で元に戻すことを忘れずに！）

```
carol@debian:~$ su - root
Password:
```



```
root@debian:~# exit
logout
carol@debian:~$ su -
Password:
root@debian:~#
```

`-`（ハイフン）を付けると変更後のユーザー環境をロードします。すなわち、`~/.profile` など **ログイン時に実行される** 初期化スクリプトを実行します。`-` を付けなければ変更前のユーザー環境を引き継ぎます。すなわち、変更後のユーザーの初期化スクリプトは実行しません。

```
carol@debian:~$ su
Password:
root@debian:/home/carol#
```

セキュリティの観点からは、`su` でrootユーザーに切り替えるのではなく、`sudo` コマンドを使用すべきです。このコマンドを使用すると、rootユーザーとして（正確にはrootユーザーに限らず他のユーザーとして）任意のコマンドを実行できます。`sudo` には `su` と比べて、2つの利点があります。

1. root権限でコマンドを実行するためにrootユーザーのパスワードは必要ありません。`sudoers` というセキュリティポリシーを設定しておけば、自分のパスワードを入力することでroot権限でコマンドを実行できます。デフォルトのセキュリティポリシーは、`/etc/sudoers` ファイルと `/etc/sudoers.d/` ディレクトリに置かれたファイルです。
2. rootユーザーとして新しいサブシェルを立ち上げる `su` とは異なり、`sudo` はシェルを経由せずにコマンドをroot権限で直接実行します。

`sudo` は `sudo -u 実行ユーザー名 コマンド` のように使います。rootユーザーとしてコマンドを実行する場合は `-u 実行ユーザー名` を省略できます。

```
carol@debian:~$ sudo -u mimi whoami
mimi
carol@debian:~$ sudo whoami
root
```

NOTE

`sudoers` はユーザーごと（ターミナルごと）に認証情報をキャッシュするので、一度自分のパスワードを入力して認証に成功すると、デフォルトでは15分間、パスワードを入力せずに `sudo` を実行できます。このデフォルト値は `/etc/sudoers` の `Defaults` に `timestamp_timeout` オプションを追加すると変更できます（例えば `Defaults timestamp_timeout=1` と記載すると認証情報のキャッシュを1分間にします）。

`/etc/sudoers` ファイル

`sudo` の設定ファイルは `/etc/sudoers`（と `/etc/sudoers.d` ディレクトリ）です。このファイルでユーザーの `sudo` 権限を設定します。以下のように、誰が、どのホストから、どのユーザーとして、どのコマンドを実行できるかを設定します。

```
carol@debian:~$ sudo less /etc/sudoers
(...)
```

```
# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
(...)
```

rootユーザーの権限指定は `ALL=(ALL:ALL) ALL` です。rootユーザーは（`root`）、すべてのホストから（`ALL`）、すべてのユーザー及びグループとして（`(ALL:ALL)`）、すべてのコマンドを実行できます（`ALL`）。`%`（パーセント）に続く名前はグループ名とみなされるので、`sudo` グループにも同様の権限が指定されています。

ユーザー `carol` が、すべてのホストから、すべてのユーザー及びグループとして、`apache2` サービスを再起動できるようにするには、`sudoers` ファイルに以下の行を追加します。

```
carol   ALL=(ALL:ALL) /usr/bin/systemctl restart apache2
```

`carol` が `systemctl restart apache2` を実行する際に、パスワードの入力を求めないようにするなら、以下のように記載します。

```
carol   ALL=(ALL:ALL) NOPASSWD: /usr/bin/systemctl restart apache2
```

carolがホスト `192.168.1.7` からログインする場合に限って、ユーザー `mimi` として `/usr/local/bin/weekly_report` コマンドを実行するできるようにするなら、以下のように記載します。

```
carol   192.168.1.7=(mimi) /usr/local/bin/weekly_report
```

`carol` がシステム管理者になることが決まり、彼女にroot権限を与える場合には、`usermod` コマンドの `-G` オプションを使って、`carol` を `sudo` グループのメンバーとするのが簡単です。現在属しているセカンダリグループへの所属をそのままにするなら、`-a` オプションも指定して実行します（訳注：`usermod` の古いバージョンには `-a` オプションがないものがあります）。

```
root@debian:~# sudo usermod -aG sudo carol
```

NOTE Red Hat系のディストリビューションでは `wheel` グループが、Debian系のシステムの `sudo` グループに相当します。

`/etc/sudoers` ファイルを編集する場合には、必ず `visudo` コマンドを使用するべきです。`visudo` は、規定のテキストエディタで sudoers ファイルを開き、編集を終えてファイルを閉じると、書式をチェックしてから書き込みます。つまり、編集ミスによりsudoersファイルを壊してしまい、sudoコマンドが使えなくなる最悪の事態を避けることができます。デフォルトのテキストエディタを変更するには、`/etc/sudoers` の `Defaults` に `editor` オプションを追加します。例えば、デフォルトのテキストエディタを `nano` にするなら、次の行を追加します。

```
Defaults    editor=/usr/bin/nano
```

NOTE

`visudo` や `vipw` などの **規定のエディタ** を実行するコマンドでは、環境変数 `EDITOR` ないし `VISUAL` に使いたいテキストエディタをセットすることで、テキストエディタを指定できます（例：`EDITOR=/usr/bin/nano`）。

`/etc/sudoers` ではエイリアス（別名）が使えます。ユーザー、ターゲット（RunAs）、ホスト、コマンド のそれぞれに対して、複数の項目のリストを1つのエイリアスで参照する機能です。

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias SERVICES = /usr/bin/systemctl *

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=SERVICES

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

このサンプルの `sudoers` ファイルに沿って、それぞれのエイリアスを説明します。

ホストエイリアス

カンマ区切りで、ホスト名、IPアドレス、ネットワーク（ネットマスク）、ネットグループ（`+` を付けます）を並べます。上記のサンプルでは、1つのIPアドレスと2つのホスト名に対して、`SERVERS` というホストエイリアスを定義しています。

```
Host_Alias SERVERS = 192.168.1.7, server1, server2
```

ユーザーエイリアス

カンマ区切りで、ユーザー名、ユーザーID（`#` を付けます）、グループ名（`%` を付けます）、グループID（`%#` を付けます）、ネットグループ（`+` を付けます）を並べます。否定を意味する `!` を付けて、除外するユーザーを指定することもできます。上記のサンプルでは、ユーザー `carol`、グループ `sudo` に属するメンバー、`PRIVILEGE_USERS` というユーザーエイリアスで示されるユーザー（`mimi`）に対して、`ADMINS` というユーザーエイリアスを定義しています。ただし、`REGULAR_USERS` というエイリアスで示されるユーザー（`john`、`mary`、`alex`）は除きます。

```
User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS
```

コマンドエイリアス

カンマ区切りで、コマンドないしディレクトリを並べます。ディレクトリを指定すると、そのディレクトリ内のすべてのファイルが含まれることになりますが、サブディレクトリは含まれません。上記のサンプルでは、`/usr/bin/systemctl` という1つのコマンド（`*` ですべてのサブコマンドを含むことを示す）に対して、`SERVICES` というコマンドエイリアスを定義しています。

```
Cmnd_Alias SERVICES = /usr/bin/systemctl *
```

これらのエイリアスを読み解くと、`User privilege specification` セクションの `ADMINS SERVERS=SERVICES` という行は、ユーザーエイリアス `ADMINS` で示されるすべてのユーザーが、ホストエイリアス `SERVERS` で示されるすべてのホストから、コマンドエイリアス `SERVICES` で示されるすべてのコマンドを、`sudo` を使って実行できると読めます。ターゲット（どのユーザーやグループに切り替えることができるか）は指定されていないので、`(ALL:ALL)` と同じです。

NOTE

ターゲット（`Runas`）エイリアスは、`Runas_Alias` で定義することと、ポリシー定義での参照位置が違うことだけがユーザーエイリアスと異なります。ポリシー定義では `(Runas_Alias:Runas_Alias)` と指定します。なお、ターゲット（`Runas`）エイリアスを用いるか否かに関わらず、`:` 以下のグループ定義を省略してユーザー定義だけを指定することができますし、ユーザー定義を省略して `:` 以下のグループ定義だけを指定することもできます。

演習

1. 特別なパーミッションに関する次の表を埋めてください。

特別なパーミッション	**数値表記**	**記号表記**	**そのパーミッション** だけ **が設定されているファイルを検索するコマンド**
SUID			
SGID			

2. `SUID` だけ が設定されているファイルや `SGID` だけ が設定されているファイルを検索して表示しても、あまり実用的ではありません。もっと実用的な以下の検索を実行してください。
   - `/usr/bin` 以下で、他のパーミッションがどうであれ、`SUID` が設定されているファイルを検索してください。
   - `/usr/bin` 以下で、他のパーミッションがどうであれ、`SGID` が設定されているファイルを検索してください。
   - `/usr/bin` 以下で `SUID` か `SGID` のいずれか一方でも設定されているファイルを検索してください。

3. `chage` コマンドを使うとユーザーのパスワードの有効期間を変更できます。ユーザー `mary` に対して、以下の表の左列に書いてある内容を実現するコマンドを、右列の空欄に書いてください。rootユーザーとして実行することを前提にしてください。

内容	`chage` **コマンド**
パスワードの有効期間を365日にする。	
次回ログイン時にパスワードの変更を強制する。	
パスワード変更不可能期間を1日に設定する。	
パスワードを失効しないようにする。	
ユーザーがいつでもパスワードを変更できるようにする。	
パスワード失効前警告期間を7日にして、アカウントが無効になる日を2050年8月20日に設定する。	
ユーザーの現在のパスワード有効期間情報を表示する。	

4. 以下の表の左列に書いてある内容を実行するネットワークユーティリティコマンドを、右列の空欄に書いてください。

内容	コマンド
ホスト `192.168.1.55` で、ポート `22` を使用しているファイルを `lsof` で表示する。	
Apacheウェブサーバーのデフォルトポートにアクセスしているプロセスを `fuser` で表示する。	
接続を待ち受けているudpソケットを `netstat` ですべて表示する。	
ホスト `192.168.1.55` のポート `80` から `443` を `nmap` でスキャンする。	

5. 一般ユーザーとして、常駐セットサイズ（RSS）に関する以下の作業を `ulimit` コマンドで実行してください。
   - RSSの最大値 の ソフト リミットを表示します。
   - RSSの最大値 の ハード リミットを表示します。
   - RSSの最大値 の ソフト リミットを5,000キロバイトに設定します。
   - RSSの最大値 の ハード リミットを10,000キロバイトに設定します。
   - RSSの最大値 の ハード リミットを15,000キロバイトに増やします。できない場合は、その理由を答えてください。

6. 次の `last` コマンドの出力行を見て、根拠とともに以下の問いに答えてください。

```
carol    pts/0        192.168.1.4      Sun May 31 14:16 - 14:22  (00:06)
```

   - `carol` はリモートホストから接続したでしょうか？
   - `carol` のセッションはどれくらいの時間続きましたか？

7. 以下に示す `/etc/sudoers` の例を見て、根拠とともに以下の問いに答えてください。

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2
```

```
# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias WEB_SERVER_RESTART = /usr/bin/systemctl restart apache2

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=WEB_SERVER_RESTART

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

`alex` はどのホストからApacheウェブサーバーを再起動できますか？

発展演習

1. SUID と SGID とは別に、スティッキービット という特別なパーミッションがあります。/tmp ディレクトリなどに設定されており、一般ユーザーは所有者が自分であるファイル以外を削除したり移動したりできなくなります。以下の指示を実行してください。
   - パーミッションが755の ~/temporal に スティッキービット を設定してください。
   - ホームディレクトリ以下から、他のパーミッションがどうであれ、スティッキービット が設定されているファイルを検索してください。
   - ~/temporal の スティッキービット を削除してください。
2. passwd -l ユーザー名 または usermod -L ユーザー名 を実行すると、そのユーザーのパスワードはロックされます。パスワードがロックされていることは /etc/shadow ファイルのどこを見ればわかりますか？
3. chage -E 日付 ユーザー名 ないし chage --expiredate 日付 ユーザー名 に相当する、usermod コマンドを実行してください。
4. localhostの65535個のポートすべてをスキャンする nmap のコマンドを2つ挙げてください。

まとめ

このレッスンでは、様々なセキュリティ管理タスクを説明しました。次のトピックを取り上げました。

- 特別なパーミッションである `SUID` と `SGID` が設定されているファイルの検索
- ユーザーのパスワードの設定／変更、パスワードの有効期間の管理
- いろいろなネットワークユーティリティを使った、ホスト／ネットワーク上で開いているポートの検出
- ユーザーが使用できるシステムリソースの制限
- ユーザーのシステムへのログイン履歴と現在ログインしているユーザーの確認
- `sudo` の使用方法と設定方法（`/etc/sudoers` ファイルの書き方）

以下のコマンドを取り上げました。

`find`
ディレクトリ階層の中でファイルを検索します。

`passwd`
ユーザーのパスワードを変更します。

`chmod`
ファイルのモードビット（パーミッション）を変更します。

`chage`
ユーザーのパスワードの有効期間を変更します。

`lsof`
開いているファイルを一覧表示します。

`fuser`
ファイルないしソケットを使用しているプロセスを特定します。

`netstat`
ネットワークの接続状態を表示します。

`nmap`
ネットワークの調査及びポートスキャンを行います。

`ulimit`
ユーザーが使用できるリソースのリミットを確認／設定します。

`last`
ログインしたユーザーを一覧表示します。

`lastb`
ログインに失敗したユーザーを一覧表示します。

/var/log/wtmp
ユーザーのログインを記録したデータベースです。

who
現在ログインしているユーザーを表示します。

w
現在ログインしているユーザーとその活動状況を表示します。

su
ユーザーを切り替えます。

sudo
別のユーザーとしてコマンドを実行します。rootユーザーとしてコマンドを実行するためによく使います。

/etc/sudoers
sudo のセキュリティポリシーの設定ファイルです。

演習の解答

1. 特別なパーミッションに関する次の表を埋めてください。

特別なパーミッション	**数値表記**	**記号表記**	**そのパーミッション**だけ**が設定されているファイルを検索するコマンド**
SUID	4000	s、S	find -perm 4000、find -perm u+s
SGID	2000	s、S	find -perm 2000、find -perm g+s

2. SUID だけ が設定されているファイルや SGID だけ が設定されているファイルを検索して表示しても、あまり実用的ではありません。もっと実用的な以下の検索を実行してください。
   - /usr/bin 以下で、他のパーミッションがどうであれ、SUID が設定されているファイルを検索してください。

     find /usr/bin -perm -4000 または find /usr/bin -perm -u+s
   - /usr/bin 以下で、他のパーミッションがどうであれ、SGID が設定されているファイルを検索してください。

     find /usr/bin -perm -2000 または find /usr/bin -perm -g+s
   - /usr/bin 以下で SUID か SGID のいずれか一方でも設定されているファイルを検索してください。

     find /usr/bin -perm /6000

3. chage コマンドを使うとユーザーのパスワードの有効期間を変更できます。ユーザー mary に対して、以下の表の左列に書いてある内容を実現するコマンドを、右列の空欄に書いてください。rootユーザーとして実行することを前提にしてください。

内容	chage **コマンド**
パスワードの有効期間を365日にする。	chage -M 365 mary、chage --maxdays 365 mary
次回ログイン時にパスワードの変更を強制する。	chage -d 0 mary、chage --lastday 0 mary
パスワード変更不可能期間を1日に設定する。	chage -m 1 mary、chage --mindays 1 mary
パスワードを失効しないようにする。	chage -M 99999 mary、chage --maxdays 99999 mary
ユーザーがいつでもパスワードを変更できるようにする。	chage -m 0 mary、chage --mindays 0 mary
パスワード失効前警告期間を7日にして、アカウントが無効になる日を2050年8月20日に設定する。	chage -W 7 -E 2050-08-20 mary、chage --warndays 7 --expiredate 2050-08-20 mary

内容	`chage` **コマンド**
ユーザーの現在のパスワード有効期間情報を表示する。	`chage -l mary`、`chage --list mary`

4. 以下の表の左列に書いてある内容を実行するネットワークユーティリティコマンドを、右列の空欄に書いてください。

内容	**コマンド**
ホスト `192.168.1.55` で、ポート `22` を使用しているファイルを `lsof` で表示する。	`lsof -i@192.168.1.55:22`
Apacheウェブサーバーのデフォルトポートにアクセスしているプロセスを `fuser` で表示する。	`fuser -vn tcp 80`、`fuser --verbose --namespace tcp 80`
接続を待ち受けているudpソケットを `netstat` ですべて表示する。	`netstat -lu`、`netstat --listening --udp`
ホスト `192.168.1.55` のポート `80` から `443` を `nmap` でスキャンする。	`nmap -p 80-443 192.168.1.55`

5. 一般ユーザーとして、常駐セットサイズ（RSS）に関する以下の作業を `ulimit` コマンドで実行してください。
   - RSSの最大値 の ソフト リミットを表示します。

     `ulimit -m`、`ulimit -Sm`
   - RSSの最大値 の ハード リミットを表示します。

     `ulimit -Hm`
   - RSSの最大値 の ソフト リミットを5,000キロバイトに設定します。

     `ulimit -Sm 5000`
   - RSSの最大値 の ハード リミットを10,000キロバイトに設定します。

     `ulimit -Hm 10000`
   - RSSの最大値 の ハード リミットを15,000キロバイトに増やします。できない場合は、その理由を答えてください。

     一般ユーザーはハードリミットを増やすことはできません。

6. 次の `last` コマンドの出力行を見て、根拠とともに以下の問いに答えてください。

```
carol    pts/0        192.168.1.4      Sun May 31 14:16 - 14:22  (00:06)
```

   - `carol` はリモートホストから接続したでしょうか？

     リモートホストから接続しました。3列目にリモートホストのIPアドレスが表示されています。

- `carol` のセッションはどれくらいの時間続きましたか？

  6分間です（最後の列に示されています）。

7. 以下に示す `/etc/sudoers` の例を見て、根拠とともに以下の問いに答えてください。

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias WEB_SERVER_RESTART = /usr/bin/systemctl restart apache2

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=WEB_SERVER_RESTART

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

`alex` はどのホストからApacheウェブサーバーを再起動できますか？

どのホストからも再起動できません。`alex` はユーザーエイリアス `REGULAR_USERS` に含まれていて、`ADMINS` からは除外されているからです。`SERVERS` から `systemctl restart apache2` を実行できるのは、`ADMINS` に含まれるユーザーだけです。

発展演習の解答

1. SUID と SGID とは別に、スティッキービット という特別なパーミッションがあります。/tmp ディレクトリなどに設定されており、一般ユーザーは所有者が自分であるファイル以外を削除したり移動したりできなくなります。以下の指示を実行してください。
   - パーミッションが755の ~/temporal に スティッキービット を設定してください。

     chmod +t temporal、chmod 1755 temporal

   - ホームディレクトリ以下から、他のパーミッションがどうであれ、スティッキービット が設定されているファイルを検索してください。

     find ~ -perm -1000、find ~ -perm /1000

   - ~/temporal の スティッキービット を削除してください。

     chmod -t temporal、chmod 0755 temporal

2. passwd -l ユーザー名 または usermod -L ユーザー名 を実行すると、そのユーザーのパスワードはロックされます。パスワードがロックされていることは /etc/shadow ファイルのどこを見ればわかりますか？

   ユーザー名の次の列である2列目が感嘆符で始まる場合、そのユーザーのパスワードはロックされています。（例：mary:!6gOg9xJgv…）。

3. chage -E 日付 ユーザー名 ないし chage --expiredate 日付 ユーザー名 に相当する、usermod コマンドを実行してください。

   usermod -e 日付 ユーザー名、usermod --expiredate 日付 ユーザー名

4. localhostの65535個のポートすべてをスキャンする nmap のコマンドを2つ挙げてください。

   nmap -p 1-65535 localhost、nmap -p- localhost

110.2 ホストのセキュリティを設定する

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 110.2

総重量

3

主な知識分野

- シャドウパスワードの知識とその動作。
- 使用していないネットワークサービスを無効にする。
- TCP wrapperの役割を理解する。

用語とユーティリティ

- `/etc/nologin`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/xinetd.d/`
- `/etc/xinetd.conf`
- `systemd.socket`
- `/etc/inittab`
- `/etc/init.d/`
- `/etc/hosts.allow`
- `/etc/hosts.deny`

110.2 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	110 セキュリティ
Objective:	110.2 ホストセキュリティの設定
Lesson:	1 of 1

はじめに

この章では、ホストのセキュリティを高める4つの基本的な手段を説明します。

1. シャドウパスワードを使って認証のセキュリティを高める
2. 受信ネットワーク接続を待ち受けるスーパーデーモンを利用する
3. 使用していないネットワークサービスを無効にする
4. ファイアウォールとしてTCPラッパーを用いる

シャドウパスワードを使って認証のセキュリティを高める

ユーザーアカウントの基本的な情報は、`/etc/passwd` ファイルに保存されます。このファイルには、ログイン名、パスワード代わりの文字、ユーザーID、グループID、コメント（GECOSとも呼ばれます）、ホームディレクトリのパス、デフォルトシェルの、7つのフィールドが含まれています。この順番を覚えるには、ログインの時に何が起きるかを考えるのが簡単です。まず、ログイン名とパスワード（passwd ファイルの中では `x` になっているのが普通です）を入力します。システムはそれらをユーザーID（UID）とグループID（GID）に対応付けます。コメントないしGECOSがあって、それからユーザーのホームディレクトリに移動して、デフォルトのシェルを起動します。

最近のシステムでは、`/etc/passwd` ファイルにパスワードが保存されず、パスワードフィールドには小文字の `x` だけが記載されています。`/etc/passwd` ファイルはすべてのユーザーが読み取り可能ですから、そこにパスワードを保存すべきではないのです。`/etc/passwd` ファイルのパスワードフィールドに記載された小文字の `x` は、暗号化された（ハッシュ化された）パスワードが `/etc/shadow` ファイルに保存されていることを示します。`/etc/shadow` ファイルは、すべてのユーザーが読み取り可能にはなっていません。

`passwd` コマンドと `chage` コマンドでパスワードに関する属性を設定できます。これらのコマンドは、`/etc/shadow` ファイル内のエントリを変更します。ユーザー `emma` のパスワードを設定するには、root権限で次のコマンドを実行します。

```
$ sudo passwd emma
New password:
Retype new password:
passwd: password updated successfully
```

確認のため新しいパスワードを2回入力することを求められます。

ユーザー `emma` のパスワードの有効期間その他の設定を表示するには、次のコマンドを実行します。

```
$ sudo chage -l emma
Last password change                    : Apr 27, 2020
Password expires                    : never
Password inactive                   : never
Account expires                     : never
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

ユーザーアカウントを無効化して `emma` がシステムにログインできないようにするには、root権限で、アカウントが無効になる日付に現在よりも昔の日付を設定します。例えば、今日が2020年3月27日だとしたら、次のように、アカウントが無効になる日付に2020年3月26日を設定します。

```
$ sudo chage -E 2020-03-26 emma
```

ユーザー `emma` のパスワードをロックするには、root権限で次のコマンドを実行します。

```
$ sudo passwd -l emma
```

`-l` オプションを指定して `passwd` コマンドを実行すると、パスワードをロックします。

（`sudo chage -E 2020-03-26 emma` のようなコマンドを実行して）ユーザーアカウントを無効化したらどうなるかを、`emma` としてログインしてみて確認します。

```
$ sudo login emma
Password:
Your account has expired; please contact your system administrator

Authentication failure
```

rootユーザー以外のすべてのユーザーがシステムにログインできないようにするには、root権限で `/etc/nologin` という名前のファイルを作成します。このファイルには、例えば「システムメンテナン

ス中」のように、ログインできない理由を知らせるメッセージを含められます。詳細は `man 5 nologin` を参照してください。

ユーザーのデフォルトのシェルに `nologin` コマンドを設定することで、そのユーザーがログインできないようにするという方法もあります。`emma` にそうするなら、次のコマンドを実行します。

```
$ sudo usermod -s /sbin/nologin emma
```

詳細については `man 8 nologin` を参照してください。

受信ネットワーク接続を待ち受けるスーパーデーモンを利用する

ウェブサーバー、メールサーバー、プリントサーバーなどのネットワークサービスは、専用のポートで接続を待ち受けるスタンドアロンのサービスであることが一般的です。こうしたスタンドアロンのネットワークサービスは、独立して並列的に動いています。サービスを管理するには、昔ながらのSysVinitに基づいたシステムでは `service` コマンドを、最近のsystemdに基づいたシステムでは `systemctl` コマンドを使います。

かつてはコンピューターリソースが非常に貧弱だったので、たくさんのサービスを並行してスタンドアローンモードで実行することには難がありました。そこで、スーパーデーモンが受信ネットワーク接続の待ち受けを一手に引き受け、要求に応じてそれぞれのネットワークサービスを開始していました。この方法ではネットワーク接続に少し余計な時間がかかります。よく知られているスーパーデーモンは `inetd` と `xinetd` です。systemdに基づく最近のシステムでは、`systemd.socket` ユニットをこの目的で使えます。この節では、`xinetd` が接続を待ち受けて要求に応じて `sshd` デーモンを開始するという例を示してから、それと同等のsystemdソケットユニットの例を示します。

`xinetd` の設定に入る前に、いくつかの準備をします。Debian系でもRed Hat系でも違いはありません。ここでの説明はDebian/GNU Linux 9.9で試したものですが、systemdに基づく最近のシステムであれば同じように実行できるはずです。

まず、`openssh-server` と `xinetd` がインストールされていることを確認します。（訳注：デフォルトで `xinetd` がインストールされているディストリビューションはかなり減っています。インストールされていない場合は、`xinetd` パッケージをインストルーしてください。）次に、以下のコマンドを実行して、SSHサービスが稼働していることを確かめます。

```
$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-04-27 09:33:48 EDT; 3h 11min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 430 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 460 (sshd)
    Tasks: 1 (limit: 1119)
   Memory: 5.3M
   CGroup: /system.slice/ssh.service
           └─460 /usr/sbin/sshd -D
```

以下のコマンドを実行して、SSHサービスがポート22番で接続を待ち受けていることを確認します。

```
$ sudo lsof -i :22
COMMAND  PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
sshd    1194 root    3u  IPv4 16053268      0t0  TCP *:ssh (LISTEN)
sshd    1194 root    4u  IPv6 16053270      0t0  TCP *:ssh (LISTEN)
```

`sshd` ではなく `xinetd` が接続を待ち受けるようにするために、以下のコマンドを実行してSSHサービスを止めます。

```
$ sudo systemctl stop sshd.service
```

再起動した後もSSHサービスを止めておくなら、`systemctl disable sshd.service` を実行します。

ここからxinedの設定に入ります。以下の内容を記載した `/etc/xinetd.d/ssh` ファイルを作成します。

```
service ssh
{
    disable     = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    server      = /usr/sbin/sshd
    server_args = -i
    flags       = IPv4
    interface   = 192.168.178.1
}
```

以下のコマンドを実行してxinetdを再起動します。

```
$ sudo systemctl restart xinetd.service
```

SSH接続の受信をどのサービスが待ち受けているかを確認します。

```
$ sudo lsof -i :22
COMMAND   PID USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
xinetd  24098 root    5u  IPv4 7345141      0t0  TCP 192.168.178.1:ssh (LISTEN)
```

xinetdサービスがポート22番で接続を待ち受けていることがわかります。

`xinetd` の設定の詳細を説明します。設定ファイルは `/etc/xinetd.conf` です。

```
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/
```

```
defaults
{

# Please note that you need a log_type line to be able to use log_on_success
# and log_on_failure. The default is the following :
# log_type = SYSLOG daemon info

}

includedir /etc/xinetd.d
```

デフォルトを設定する部分（`defaults { }`）を除くと、ディレクトリを読み込むディレクティブ（`includedir /etc/xinetd.d`）だけがあります。このディレクトリ `/etc/xinetd.d` 内に、`xinetd` が扱うサービスごとに設定ファイルを作成していきます。先ほどの例では、SSHサービスの設定ファイルを `/etc/xinetd.d/ssh` という名前で作成しました。`.`（ドット）が含まれておらず `~`（チルダ）で終わらない限りファイル名は何でもよいのですが、設定するサービス名をファイル名にするのが一般的です。

ディレクトリ `/etc/xinetd.d` 内に、最初から作られている設定ファイルがあります。

```
$ ls -l /etc/xinetd.d
total 52
-rw-r--r-- 1 root root 640 Feb  5  2018 chargen
-rw-r--r-- 1 root root 313 Feb  5  2018 chargen-udp
-rw-r--r-- 1 root root 502 Apr 11 10:18 daytime
-rw-r--r-- 1 root root 313 Feb  5  2018 daytime-udp
-rw-r--r-- 1 root root 391 Feb  5  2018 discard
-rw-r--r-- 1 root root 312 Feb  5  2018 discard-udp
-rw-r--r-- 1 root root 422 Feb  5  2018 echo
-rw-r--r-- 1 root root 304 Feb  5  2018 echo-udp
-rw-r--r-- 1 root root 312 Feb  5  2018 servers
-rw-r--r-- 1 root root 314 Feb  5  2018 services
-rw-r--r-- 1 root root 569 Feb  5  2018 time
-rw-r--r-- 1 root root 313 Feb  5  2018 time-udp
```

あまりないとは思いますが、`daytime`（時刻を同期するために大昔に使っていたサーバーです）などの古いサービスを使用する場合には、これらのファイルをテンプレートとして使えます。これらのテンプレートファイルには、`disable = yes` というディレクティブが含まれていますから、`yes` を `no` に変更すると有効になります。

先のsshの例で `/etc/xinetd.d/ssh` に記載したディレクティブについて、少し詳しく見てみましょう。

```
service ssh
{
    disable     = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    server      = /usr/sbin/sshd
```

```
    server_args     = -i
    flags       = IPv4
    interface   = 192.168.178.1
}
```

service
xinetdが制御するサービスを記載します。22のようにポート番号を記載してもよいですし、ssh のように /etc/services でポート番号と対応づけられているサービス名を記載してもよいです。

{
中括弧内に詳細設定を記載します。

disable
no にすると設定が有効になります。設定を無効にしたいときは yes にします。

socket_type
TCPソケットでは stream に、UDPソケットでは dgram にします。

protocol
TCPまたはUDPのいずれかを記載します。

wait
TCP接続の場合は通常 no に設定します。

user
開始するサービスの所有者を記載します。

server
xinetd が開始するサービスのフルパスを指定します。

server_args
サービス（コマンド）のオプションを追加します。スーパーデーモンからの起動を指示するオプションを指定します。SSHなら -i オプションです。

flags
IPv4やIPv6などのオプション。通常は指定する必要がありません。

interface
xinetd が待ち受けるネットワークインターフェイス名、ないしはIPアドレスを記載します。bind ディレクティブも同じ意味です。

}
中括弧を閉じるのを忘れないようにしてください。

systemdを使用するシステムでは、xinetd ではなく、systemdソケットユニットを使用するのが普通です。SSHサービスがインストールされていれば、そのsystemdソケットユニットが既に存在しますから、設定ファイルの作成は必要ありません。通常は、systemdソケットユニットではなくssh（ないしsshd）サービスユニットが起動しているはずですから、まずそれを停止します。xinetd も停止していることを確かめてください。（訳注：sudo lsof -i :22 を実行して何も表示されなければ、sshd

も `xinetd` も起動していないことがわかります。）

以下のコマンドを実行してSSHソケットユニットを開始します。

```
$ sudo systemctl start ssh.socket
```

もう一度 `lsof` を実行して、今はどのサービスがポート22番を待ち受けているかを確認してみましょう。ここでは `-P` オプションを指定して、サービス名ではなくポート番号を表示します。

```
$ sudo lsof -i :22 -P
COMMAND PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
systemd   1 root   57u  IPv6 14730112      0t0  TCP *:22 (LISTEN)
```

この節の締めくくりとして、好みのSSHクライアントでログインしてみてください。sshdプロセスが起動していることをpsコマンドで確認してみましょう。

TIP `systemctl start ssh.socket` が動作しなければ、`systemctl start sshd.socket` を試してみてください。

使用していないネットワークサービスを無効にする

システムリソースとセキュリティの観点から、どのサービスが実行中かを把握しておくことは重要です。使われていない不要なサービスは無効にすべきです。例えば、ウェブページを提供する必要がないのであれば、Apacheやnginxといったウェブサーバーを実行する必要はありません。

SysVinitに基づいたシステムでは、次のコマンドを実行するとサービスの状況を確認できます。

```
$ sudo service --status-all
```

このコマンドを実行して出力されるサービスの一覧を確認して、不要なサービスを無効にします。Debian系のシステムでは次のコマンドを実行します。

```
$ sudo update-rc.d サービス名 remove
```

Red Hat系のシステムでは次のコマンドを実行します。

```
$ sudo chkconfig サービス名 off
```

最近主流のsystemdに基づいたシステムでは、次のコマンドを実行すると、すべての実行中のサービスを表示します。

```
$ systemctl list-units --state active --type service
```

次のコマンドを実行して不要なサービスユニットを無効にします。

```
$ sudo systemctl disable ユニット --now
```

このコマンドは、指定したユニットのサービスを停止して、有効なサービスのリストから削除します。そのため、次回以降の起動時には自動実行されなくなります。

さらに、接続を待ち受けているネットワークサービスの調査も行います。古いシステムでは、`netstat` コマンド（`net-tools` パッケージをインストールすると使えるようになります）を実行します。

```
$ netstat -ltu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
```

新しいシステムでは、`ss`（Socket Services）コマンドを実行します。

```
$ ss -ltu
Netid      State         Recv-Q        Send-Q      Local Address:Port           Peer Address:Port
udp        UNCONN        0             0                 0.0.0.0:bootpc             0.0.0.0:*
tcp        LISTEN        0             128               0.0.0.0:ssh                0.0.0.0:*
tcp        LISTEN        0             80              127.0.0.1:mysql              0.0.0.0:*
tcp        LISTEN        0             128                    *:http                     *:*
tcp        LISTEN        0             128                [::]:ssh                   [::]:*
```

ファイアウォールとしてTCPラッパーを用いる

Linuxでファイアウォールが使えなかった時代には、TCPラッパーをホストのネットワーク接続のセキュリティを高めるために使っていました。現在では、TCPラッパーに対応していないプログラムがほとんどです。Fedora 29などのRed Had系のディストリビューションではTCPラッパーがサポートされなくなりました。それでも、古いシステムではTCPラッパーが用いられていることがありますから、基本的な事柄を知っておくことは有用です。（訳注：インストールされていない場合は、`tcpd` パッケージをインストールします。）

ここでもSSHサービスを例に取ります。ローカルネットワークからしかSSH接続できないようにします。まず、SSHデーモンが、TCPラッパーをサポートする `libwrap` ライブラリを使っているかを確認します。

```
$ ldd /usr/sbin/sshd | grep "libwrap"
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f91dbec0000)
```

次に、`/etc/hosts.deny` ファイルに以下の行を追記します。

```
sshd: ALL
```

最後に、`/etc/hosts.allow` ファイルに、ローカルネットワークからのSSH接続を例外的に許容するために、以下の行を追記します。

```
sshd: LOCAL
```

この変更は即座に反映されます。サービスを再起動する必要はありません。ローカルネットワーク以外から `ssh` クライアントで接続して確かめてみてください。

演習

1. ロックされている `emma` のパスワードを解除（アンロック）してください。

2. ユーザーアカウント `emma` の有効期間を無期限（いつまでもアカウントが無効にならない）に設定してください。

3. 印刷ジョブを処理するCUPSプリントサービスを無効にしてください。そして、CUPSプリントサービスで使われるポートで接続を待ち受けていないことを確かめてください。（CUPSプリントサービスを利用しているシステムで実施する場合には注意してください。）

4. インストールしたnginxウェブサーバーがTCPラッパーをサポートしているかどうかを調べてください。

発展演習

1. `/etc/nologin` ファイルが存在すると、ユーザー `root` もログインできなくなるでしょうか？

2. `/etc/nologin` ファイルが存在すると、パスワードを使用しないSSH鍵によるログインもできなくなるでしょうか？

3. `login currently is not possible` とだけ記載された `/etc/nologin` ファイルが存在すると、ログイン時にどうなりますか？

4. 一般ユーザーである `emma` は、`grep root /etc/passwd` コマンドを実行して `/etc/passwd` の中のユーザー `root` の情報を見ることができますか？

5. 一般ユーザーである `emma` は、`grep -i emma /etc/shadow` コマンドを実行して `/etc/shadow` の中の自分のハッシュ化されたパスワードを見ることができますか？

6. 時刻を同期するために大昔に使っていたdaytimeサービスを、xinetdで利用するためにはどうすればよいでしょうか？（これはあくまでも説明のための演習です。本番環境では実行しないでください。）

まとめ

このレッスンでは、以下の事柄を学びました:

1. パスワードとその有効期間などの設定が保存されているファイル
2. `xinetd` スーパーデーモンの目的と `xinetd` が `sshd` サービスを開始する仕組み
3. 実行中のネットワークサービスの確認と不要なサービスの無効化
4. シンプルなファイアウォールとしてのTCPラッパー

以下のコマンドを取り上げました。

`chage`
パスワードの有効期間などを変更します。

`chkconfig`
Red Hat系のシステムで、サービスを自動起動するかどうかを設定します。（古いコマンド）

`netstat`
ネットワークポートにアクセスするサービスの状態を表示する古いユーティリティです（`net-tools` パッケージに含まれています）。

`nologin`
ユーザーのシェルとして設定すると、そのユーザーがログインできなくなります。

`passwd`
ユーザーのパスワードをセットないし変更します。

`service`
サービスの停止や開始など、デーモンの状態を制御する古いコマンドです。

`ss`
ソケットの詳細状態を表示する、現在主流のコマンドです。（古いコマンドは `netstat`）。

`systemctl`
systemdを採用しているシステムでサービスやソケットなどを制御します。

`update-rc.d`
Debian系のシステムで、サービスを自動起動するかどうかを設定します。（古いコマンド）

`xinetd`
要求に応じてネットワークサービスを起動して、そこに接続させるスーパーデーモンです。それぞれのサービスは `xinetd` から呼び出されて起動します。

演習の解答

1. ロックされている `emma` のパスワードを解除（アンロック）してください。

   root権限で `passwd -u emma` を実行します。

2. ユーザーアカウント `emma` の有効期間を無期限（いつまでもアカウントが無効にならない）に設定してください。

   root権限で `chage -E -1 emma` を実行します（`-E` オプションに数値 `-1` を指定）。`chage -l emma` を実行すると設定を確認できます。

3. 印刷ジョブを処理するCUPSプリントサービスを無効にしてください。そして、CUPSプリントサービスで使われるポートで接続を待ち受けていないことを確かめてください。（CUPSプリントサービスを利用しているシステムで実施する場合には注意してください。）

   root権限で次のコマンドを実行します。

   ```
   systemctl disable cups.service --now
   ```

   次のいずれかのコマンドで接続を待ち受けていないことを確かめます。

   ```
   netstat -l | grep ":ipp "
   ```

   または

   ```
   ss -l | grep ":ipp "
   ```

4. インストールしたnginxウェブサーバーがTCPラッパーをサポートしているかどうかを調べてください。

   次のコマンドを実行します。

   ```
   ldd /usr/sbin/nginx | grep "libwrap"
   ```

   nginxがTCPラッパーをサポートしている場合、エントリが表示されます。

発展演習の解答

1. `/etc/nologin` ファイルが存在すると、ユーザー `root` もログインできなくなるでしょうか？

   ユーザー `root` はログインできます。

2. `/etc/nologin` ファイルが存在すると、パスワードを使用しないSSH鍵によるログインもできなくなるでしょうか？

   パスワードを使用しないログインもできなくなります。

3. `login currently is not possible` とだけ記載された `/etc/nologin` ファイルが存在すると、ログイン時にどうなりますか？

   `login currently is not possible` というメッセージが表示され、ログインはできません。

4. 一般ユーザーである `emma` は、`grep root /etc/passwd` コマンドを実行して `/etc/passwd` の中のユーザー `root` の情報を見ることができますか？

   `/etc/passwd` ファイルは誰でも読み取り可能ですからから、ユーザー `root` の情報を見ることができます。

5. 一般ユーザーである `emma` は、`grep -i emma /etc/shadow` コマンドを実行して `/etc/shadow` の中の自分のハッシュ化されたパスワードを見ることができますか？

   一般ユーザーには `/etc/shadow` ファイルの読み取りパーミッションがありませんから、自分のハッシュ化されたパスワードを見ることができません。

6. 時刻を同期するために大昔に使っていたdaytimeサービスを、xinetdで利用するためにはどうすればよいでしょうか？（これはあくまでも説明のための演習です。本番環境では実行しないでください。）

   まず、`/etc/xinetd.d/daytime` ファイルの `disable` ディレクティブを `disable = no` に変更します。次に、`systemctl restart xinetd.service`（SysVinitに基づいたシステムでは `service xinetd restart`）コマンドを実行してxinetdサービスを再起動します。`nc localhost daytime`（`netcat localhost daytime`）コマンドを実行すると、daytimeサービスが動いているかどうかを確かめられます。

110.3 暗号化によるデータの保護

LPI目標への参照

LPIC-1 version 5.0, Exam 102, Objective 110.3

総重量

4

主な知識分野

- 基本的なOpenSSH 2クライアントの設定と使用法を実行する。
- OpenSSH 2サーバのホスト鍵の役割を理解する。
- 基本的なGnuPGの設定・利用・廃棄の実行。
- 基本的なGnuPGの設定、使用法、取消しを実行する。
- SSHポートトンネル(X11トンネルを含む)について理解する。

用語とユーティリティ

- `ssh`
- `ssh-keygen`
- `ssh-agent`
- `ssh-add`
- `~/.ssh/id_rsa` and `id_rsa.pub`
- `~/.ssh/id_dsa` and `id_dsa.pub`
- `~/.ssh/id_ecdsa` and `id_ecdsa.pub`
- `~/.ssh/id_ed25519` and `id_ed25519.pub`
- `/etc/ssh/ssh_host_rsa_key` and `ssh_host_rsa_key.pub`
- `/etc/ssh/ssh_host_dsa_key` and `ssh_host_dsa_key.pub`
- `/etc/ssh/ssh_host_ecdsa_key` and `ssh_host_ecdsa_key.pub`
- `/etc/ssh/ssh_host_ed25519_key` and `ssh_host_ed25519_key.pub`
- `~/.ssh/authorized_keys`
- `ssh_known_hosts`

- `gpg`
- `gpg-agent`
- `~/.gnupg/`

110.3 レッスン 1

Certificate:	LPIC-1
Version:	5.0
Topic:	110 セキュリティ
Objective:	110.3 暗号化によるデータの保護
Lesson:	1 of 2

はじめに

今日のシステム管理では、暗号化してデータを保護することが非常に重要です。リモート接続する場合は特にそうです。telnet、rlogin、FTP といった方法は安全ではありませんが、SSH（Secure SHell、安全なシェル）プロトコルは安全に利用できるよう設計されています。SSHでは、公開鍵暗号を使ってホストとクライアント（ユーザー）の両者をそれぞれ認証し、すべてのやり取りを暗号化します。さらに、SSHトンネル で接続すると、暗号化されたSSH接続を通して暗号化されていないプロトコルでのデータの送受信ができます。SSHプロトコルの現在推奨されているバージョンは2.0です。SSHプロトコルを使用して接続できる OpenSSH というフリーでオープンなソフトウェアがあります。

このレッスンでは、OpenSSH クライアントの基本設定と、OpenSSH サーバーのホスト鍵について説明し、SSHトンネルにも触れます。以下の2台のマシンを想定して話を進めます。

マシンの役割	OS	IPアドレス	ホスト名	ユーザー
クライアント	Debian GNU/Linux 10 (buster)	`192.168.1.55`	`debian`	`carol`
サーバー	openSUSE Leap 15.1	`192.168.1.77`	`halof`	`ina`

OpenSSHクライアントの基本設定と使い方

OpenSSHのサーバーとクライアントは別々のパッケージではありますが、両方を一気にインストールするのが普通です（訳注：`ssh` という名前のメタパッケージをインストールすると両方を一気にインストールできますが、それが見つからなければ `openssh-server` と `openssh-clients`（`openssh-client`）を

それぞれインストールしてください）。`ssh` コマンドに続けて、リモートマシンでのユーザー名と@でつなげたIPアドレスないしホスト名を指定して実行すると、SSHサーバーとのリモートセッションを確立します。初回接続時には次のようなメッセージが表示されます。

```
carol@debian:~$ ssh ina@192.168.1.77
The authenticity of host '192.168.1.77 (192.168.1.77)' can't be established.
ECDSA key fingerprint is SHA256:5JF7anupYipByCQm2BPvDHRVFJJixeslmppi2NwATYI.
Are you sure you want to continue connecting (yes/no)?
```

`yes` と入力してエンターキーを押すと、リモートマシンのユーザーのパスワードの入力を求められます。正しいパスワードを入力してエンターキーを押すと、警告メッセージが表示されてリモートホストにログインできます。

```
Warning: Permanently added '192.168.1.77' (ECDSA) to the list of known hosts.
Password:
Last login: Sat Jun 20 10:52:45 2020 from 192.168.1.4
Have a lot of fun...
ina@halof:~>
```

この一連のメッセージについて説明します。`192.168.1.77` のリモートサーバーへの初回接続なので、ローカルマシンはそのサーバーの真正性をデータベースに照らし合わせて確認することができません。そこで、リモートサーバーの公開鍵に `SHA256` ハッシュ関数を適用して生成した `ECDSA key fingerprint（ECDSA鍵指紋）` を提示します。`yes` とエンターキーを入力してそのサーバーへの接続を許可すると、そのサーバーの公開鍵が known hosts（既知のホスト） データベースに追加されます。次回からは、このデータベースと照らし合わせてサーバーの真正性を確認できます。known hosts（既知のホスト） の公開鍵のリストは `~/.ssh` ディレクトリ内にある `known_hosts` ファイルに保存されています。

```
ina@halof:~> exit
logout
Connection to 192.168.1.77 closed.
carol@debian:~$ ls .ssh/
known_hosts
```

`.ssh` ディレクトリと `known_hosts` ファイルは、初めてリモート接続を確立した時に作られます。`~/.ssh` が、ユーザー固有の設定と認証情報を格納するデフォルトのディレクトリです。

NOTE 1行の `ssh` コマンドで、リモートホストで一つのコマンドを実行してからローカルのターミナルに戻ることができます（例：`ssh ina@halof ls`）。

ローカルホストとリモートホストで同じユーザー名を使うなら、SSH接続をする際にユーザー名を指定せずにすみます。たとえば、`debian` にユーザー `carol` としてログインしていて、`halof` にユーザー `carol` として接続するなら、`ssh 192.168.1.77` あるいは（名前解決ができるなら）`ssh halof` と入力して実行するだけでよいです。

```
carol@debian:~$ ssh halof
Password:
```

```
Last login: Wed Jul  1 23:45:02 2020 from 192.168.1.55
Have a lot of fun...
carol@halof:~>
```

`halof` と同じIPアドレスである別のリモートホストに接続しようとしたら（LANでDHCPを使っているとあり得ることです）、中間者 攻撃の可能性があるという警告が表示されます。

```
carol@debian:~$ ssh john@192.168.1.77
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    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 ECDSA key sent by the remote host is
SHA256:KH4q3vP6C7e0SEjyG8Wlz9fVlf+jmWJ5139RBxBh3TY.
Please contact your system administrator.
Add correct host key in /home/carol/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/carol/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.1.77"
ECDSA host key for 192.168.1.77 has changed and you have requested strict checking.
Host key verification failed.
```

この場合は 中間者 攻撃ではありませんので、新しいホストの公開鍵指紋を `.ssh/known_hosts` に追加しても安全です。警告文に示されているように、`ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.1.77"` を実行して 問題となっている 鍵を削除します（`-f` ファイル名 を指定せずに実行すると `~/.ssh/known_hosts` から鍵を削除するので、`-f` オプションを省略して `ssh-keygen -R 192.168.1.77` でも構いません）。そうすると、同じIPアドレスである別のリモートホストに接続できます。

SSH公開鍵認証

ログイン時にパスワードではなく公開鍵を使うようにSSHクライアントを設定できます。そのほうがはるかに安全なので、SSHでリモートサーバーに接続する方法としては、公開鍵認証が望ましいです。そのためにはまずクライアント側のマシンで鍵ペアを作成します。`ssh-keygen` コマンドで `-t` オプションに続けて暗号方式を指定します。ここではecdsa（Elliptic Curve Digital Signature Algorithm、楕円曲線デジタル署名アルゴリズム）にします。鍵ペアを保存するパス（デフォルトの `~/.ssh/` にしておくと便利です）とパスフレーズを尋ねられます。パスフレーズはオプションで、なくても鍵ペアを作成できますが、パスフレーズを設定することが強く推奨されます。

```
carol@debian:~/.ssh$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/carol/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/carol/.ssh/id_ecdsa.
Your public key has been saved in /home/carol/.ssh/id_ecdsa.pub.
The key fingerprint is:
```

```
SHA256:tlamD0SaTquPZYdNepwj8XN4xvqmHCbe8g5FKKUfMo8 carol@debian
The key's randomart image is:
+---[ECDSA 256]---+
|      .          |
|     o .         |
|    = o o        |
|     B *         |
|    E B S o      |
|     o & O       |
|      @ ^ =      |
|     *.@ @.      |
|    o.o+B+o      |
+----[SHA256]-----+
```

NOTE 鍵ペアを作成するときに、`ssh-keygen` コマンドで `-b` オプションに続けてビット単位で鍵のサイズを指定できます（例：`ssh-keygen -t ecdsa -b 521`）。（訳注：指定できる値は暗号方式によって異なります。）

先に示したコマンドを実行すると、`~/.ssh` ディレクトリに2つのファイルが作成されます。

```
carol@debian:~/.ssh$ ls
id_ecdsa  id_ecdsa.pub  known_hosts
```

`id_ecdsa`
秘密鍵です。

`id_ecdsa.pub`
公開鍵です。

NOTE 非対称暗号（公開鍵暗号）では、公開鍵と秘密鍵には数学的な関連があり、一方の鍵で暗号化されたものは他方の鍵でしか復号できません。

次に、接続したいリモートホストのユーザーの `~/.ssh/authorized_keys` ファイルに、先ほど作成した公開鍵を追加します（`~/.ssh` ディレクトリが存在しなければ、公開鍵を追加する前にそのディレクトリを作成しなければなりません）。公開鍵を追加するには、USBメモリを使う、SSHでファイルを転送する `scp` コマンドを使うなど、いろいろな方法がありますが、ここでは公開鍵の中身を `cat` で出力して、`ssh` にパイプでつなげ、`cat` で `~/.ssh/authorized_keys` ファイルに追記します。

```
carol@debian:~/.ssh$ cat id_ecdsa.pub |ssh ina@192.168.1.77 'cat >> .ssh/authorized_keys'
Password:
```

公開鍵をリモートホストの `authorized_keys` ファイルに追加してから新たに接続するときに、パスフレーズの有無に応じて次の2通りのシナリオが考えられます。

- 鍵ペアの作成時にパスフレーズを設定しなかったなら、`ssh` コマンドを実行すると自動的にログインできます。これは便利ですが、状況によっては安全ではありません。

```
carol@debian:~$ ssh ina@192.168.1.77
```

```
Last login: Thu Jun 25 20:31:03 2020 from 192.168.1.55
Have a lot of fun...
ina@halof:~>
```

- 鍵ペアの作成時にパスフレーズを設定したなら、`ssh` の接続時に毎回パスフレーズを入力しなければなりません。SSHのパスワード認証と似ているように見えますが、公開鍵とパスフレーズの2層でセキュリティを担保しているので、より安全です。しかし、利便性に関しては、接続時に毎回パスワードを入力しなければならないSSHパスワード認証と同じ煩わしさがあります。これは不便に感じますが、パスフレーズを設定していない秘密鍵が流出したら、その秘密鍵を使って、ペアになる公開鍵が追加されているどのサーバーにでも接続できてしまうので、パスフレーズを設定したほうがよいのです。

```
carol@debian:~/.ssh$ ssh ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
Last login: Thu Jun 25 20:39:30 2020 from 192.168.1.55
Have a lot of fun...
ina@halof:~>
```

セキュリティと利便性を両立させる方法があります。SSH認証エージェント（`ssh-agent`）を使うのです。認証エージェントがシェルを立ち上げて、そのセッションの最中はパスフレーズで解読された秘密鍵をメモリに保持することで、公開鍵認証を行うという仕組みです。詳しく見てみましょう。

1. `ssh-agent` で新しいBashシェルを立ち上げます。

```
carol@debian:~/.ssh$ ssh-agent /bin/bash
carol@debian:~/.ssh$
```

2. `ssh-add` で秘密鍵をメモリの安全な領域に追加します。鍵ペアの作成時にパスフレーズを設定していたら（セキュリティのためにはパスフレーズを設定することが推奨されます）、ここで入力を求められます。

```
carol@debian:~/.ssh$ ssh-add
Enter passphrase for /home/carol/.ssh/id_ecdsa:
Identity added: /home/carol/.ssh/id_ecdsa (carol@debian)
```

   以後は公開鍵が追加されているどのリモートサーバーにも、パスフレーズを入力せずにログインできます。コンピュータの起動時に上記のコマンドが実行されるようにしておくと、コンピュータをシャットダウンをするまで（あるいは手動で鍵をアンロードするまで）秘密鍵がメモリに保持されます（訳注：ログイン時に自動的に `ssh-agent` が実行されるディストリビューションもあります）。

`ssh-keygen` で指定できる4種類の公開鍵暗号方式のアルゴリズムを紹介して、この節を締めくくります。

RSA

Ron Rivest、Adi Shamir、Leonard Adlemanという3名の考案者の名前から命名された方式で、1977年に公表されました。現在でも安全だとされており、広く用いられています。鍵の最小サイズは1024ビットです（デフォルトは3072ビットです）。

DSA
Digital Signature Algorith（デジタル署名アルゴリズム） です。安全ではないと判明したので、OpenSSH 7.0から非推奨になっています。DSAの鍵は1024ビットごとのサイズになります。

ecdsa
DSAを改良したものでより安全だと考えられている、Elliptic Curve Digital Signature Algorithm（楕円曲線デジタル署名アルゴリズム） です。楕円曲線暗号を使用します。ECDSAの鍵のサイズは、256ビット、384ビット、521ビットという楕円曲線のサイズのいずれかです。

ed25519
EdDSA（Edwards-curve Digital Signature Algorithm、エドワーズ曲線デジタル署名アルゴリズム）を実装したものです。強力なCurve25519を用いています。4種類の暗号方式の中で最も安全だとされています。Ed25519の鍵のサイズは256ビットに固定です。

NOTE `-t` オプションを付けずに `ssh-keygen` コマンドを実行すると、`RSA` の鍵ペアがデフォルトで作成されます。

OpenSSHサーバーのホスト鍵の役割

OpenSSHのグローバル設定ディレクトリは、`/etc/ssh` です。

```
halof:~ # tree /etc/ssh
/etc/ssh
├── moduli
├── ssh_config
├── ssh_host_dsa_key
├── ssh_host_dsa_key.pub
├── ssh_host_ecdsa_key
├── ssh_host_ecdsa_key.pub
├── ssh_host_ed25519_key
├── ssh_host_ed25519_key.pub
├── ssh_host_rsa_key
├── ssh_host_rsa_key.pub
└── sshd_config

0 directories, 11 files
```

`moduli` とクライアントの設定ファイル（`ssh_config`）とサーバーの設定ファイル（`sshd_config`）を除けば、OpenSSHサーバーのインストール時に作成された、4種類の暗号方式アルゴリズムに対応する鍵ペアがあります。先述したように、サーバーはこれらの ホスト鍵 をクライアントの要求に応じて提示します。ファイル名は次のパターンになっています。

秘密鍵
`ssh_host_` + アルゴリズム + `_key`（例：`ssh_host_rsa_key`）

公開鍵（または公開鍵指紋）
`ssh_host_` + アルゴリズム + `_key.pub`（例：`ssh_host_rsa_key.pub`）

NOTE 公開鍵に暗号学的ハッシュ関数を適用して鍵指紋を作成します。鍵指紋は鍵自体よりも短いので、鍵を管理する際に扱いやすいです。

秘密鍵ファイルのパーミッションは `0600`、すなわち `rw-------` で、所有者（root）だけが読み書き可能です。公開鍵ファイルは所有グループとその他も読み取り可能です（`0644`、すなわち `rw-r--r--`）。

```
halof:~ # ls -l /etc/ssh/ssh_host_*
-rw------- 1 root root 1381 Dec 21 20:35 /etc/ssh/ssh_host_dsa_key
-rw-r--r-- 1 root root  605 Dec 21 20:35 /etc/ssh/ssh_host_dsa_key.pub
-rw------- 1 root root  505 Dec 21 20:35 /etc/ssh/ssh_host_ecdsa_key
-rw-r--r-- 1 root root  177 Dec 21 20:35 /etc/ssh/ssh_host_ecdsa_key.pub
-rw------- 1 root root  411 Dec 21 20:35 /etc/ssh/ssh_host_ed25519_key
-rw-r--r-- 1 root root   97 Dec 21 20:35 /etc/ssh/ssh_host_ed25519_key.pub
-rw------- 1 root root 1823 Dec 21 20:35 /etc/ssh/ssh_host_rsa_key
-rw-r--r-- 1 root root  397 Dec 21 20:35 /etc/ssh/ssh_host_rsa_key.pub
```

`-l` オプションと `-f` オプションに続けて鍵ファイルのパスを指定して `ssh-keygen` を実行すると、鍵指紋を確認できます。

```
halof:~ # ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
halof:~ # ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
```

`-v` オプションを加えると、ランダムアートも見ることができます。

```
halof:~ # ssh-keygen -lv -f /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
+--[ED25519 256]--+
|              +oo|
|             .+o.|
|        .    ..E.|
|         + .  +.o|
|        S +  + *o|
|          ooo Oo=|
|     . . . =o+.==|
|      = o =oo o=o|
|     o.o +o+..o.+|
+----[SHA256]-----+
```

SSHトンネル

OpenSSHは非常に強力な転送機能を備えています。転送元のポートをSSHプロセスを通じてトンネル化・暗号化して、転送先のポートにリダイレクトします。この仕組みは、ポートトンネリング や ポート転送（ポートフォワーディング） と呼ばれます。次のような大きな利点があります。

- ファイアウォールをう回してリモートホストのポートに接続できます。
- 外部からプライベートネットワーク内のホストに接続できます。
- データの受け渡しを暗号化できます。

ポート転送は、ローカルポート転送とリモートポート転送の2つに大別されます。

ローカルポート転送

ローカルホストのポート番号を指定して、そこに送られたトラフィックを接続先のホストに転送します。接続先のホストでは、そのトラフィックを最終的な宛先として指定されたホストの指定されたポートに宛てて送信します。例えば、ローカルホストの `8585` ポートから、接続先サイト `halof` を経由して、`www.gnu.org` の `80` 番ポート（http）に転送するには、このようにします。

```
carol@debian:~$ ssh -L 8585:www.gnu.org:80 ina@halof
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':

Last login: Thu Jun 25 20:39:30 2020 from 192.168.1.55
Have a lot of fun...
ina@halof:~>
```

順に説明します。`-L` オプションに続けて、接続を待ち受けるローカルポート `8585` を指定します。続く `www.gnu.org:80` は、最終的な宛先ホストとポート番号を示します。ローカルポート `8585` への接続が開かれると、そのトラフィックはローカルホストと接続先のリモートホストhalofのssh接続を通して転送され、halofから `www.gnu.org:80` への接続が開かれます。ウェブブラウザで `http://localhost:8585` に接続すると、halofを経由して `www.gnu.org` に転送されます。ローカルマシンで `lynx`（昔からあるテキストベースのウェブブラウザ）を使って試してみましょう。

```
carol@debian:~$ lynx http://localhost:8585
(...)
  * Back to Savannah Homepage
     * Not Logged in
     * Login
     * New User
     * This Page
     * Language
     * Clean Reload
     * Printer Version
     * Search
     * _
(...)
```

ポート転送だけを行いたければ、次のようにします。

```
carol@debian:~$ ssh -L 8585:www.gnu.org:80 -Nf ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol@debian:~$
carol@debian:~$ lynx http://localhost:8585
(...)
  * Back to Savannah Homepage
     * Not Logged in
     * Login
     * New User
     * This Page
```

```
     * Language
     * Clean Reload
     * Printer Version
     * Search
     * _
(...)
```

次の3点に注意してください。

- `-N` オプションを指定したため、`halof` にログインせずポート転送を行います。
- `-f` オプションを指定し、SSHをバックグラウンドで実行しています。
- `ina@192.168.1.77` で、ホスト `halof` 上の ユーザー `ina` を指定しています（`ina@halof` でも構いません）。

リモートポート転送

リモートポート転送（リバースポート転送）では、リモートサーバーの指定したポートに到達したトラフィックが、ローカルホストで実行しているSSHプロセスへと転送され、最終的な宛先として指定されたホストの指定されたポート（外部のホストではなくローカルホストのポートを指定しても構いません）へと送られます。たとえば、プライベートネットワーク外から、`halof`（`192.168.1.77`）で実行しているSSHサーバーのポート `8585` を経由して、ローカルホストで実行しているApacheウェブサーバー（ポート `80`）にアクセスできるようにするなら、次のコマンドを実行します。

```
carol@debian:~$ ssh -R 8585:localhost:80 -Nf ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol@debian:~$
```

`halof` のポート `8585` に接続すると、`Debian` のApache2のデフォルトホームページを目にすることになります。

```
carol@debian:~$ lynx 192.168.1.77:8585
(...)
                                                                Apache2 Debian Default Page: It works (p1 of
3)
   Debian Logo Apache2 Debian Default Page
   It works!

   This is the default welcome page used to test the correct operation of the Apache2 server after
   installation on Debian systems. If you can read this page, it means that the Apache HTTP server
   installed at this site is working properly. You should replace this file (located at
   /var/www/html/index.html) before continuing to operate your HTTP server.
(...)
```

NOTE ダイナミックポート転送という、3つ目のより複雑なポート転送がありますが、このレッスンでは扱いません。ダイナミックポート転送では、SOCKSプロキシーとしてポート番号を限定しないTCP通信を転送します。

X11転送

ポート転送の説明を終えたので、X11転送（X11トンネル）の話をして締めくくります。X11転送では、リモートホストの X Window System をローカルマシンに転送します。`ssh` コマンドで `-X` オプションを指定します。

```
carol@debian:~$ ssh -X ina@halof
...
```

これで `firefox` ウェブブラウザのようなグラフィカルアプリケーションを、リモートマシンで立ち上げられるようになります。リモートマシンではローカルホストのX Windowサーバーに接続するためのリバーストンネルが作成され、シェルにはそのトンネルを使用するための `DISPLAY` 環境変数がセットされます。これにより、リモートサーバーで実行されたアプリケーションは、ローカルマシンのディスプレイを使って入出力を行います。

`-x` オプションを指定して新しいSSHセッションを開始すると、X11転送 は無効化されます。`firefox` を起動しようとすると、以下のようなエラーになります。

```
carol@debian:~$ ssh -x ina@halof
carol@192.168.0.106's password:
(...)
ina@halof:~$ firefox

(firefox-esr:1779): Gtk-WARNING **: 18:45:45.603: Locale not supported by C library.
    Using the fallback 'C' locale.
Error: no DISPLAY environment variable specified
```

NOTE ローカルポート転送、リモートポート転送、X11転送に関係する設定のディレクティブは、それぞれ、`AllowTcpForwarding`、`GatewayPorts`、`X11Forwarding` です。詳細は `man ssh_config` と `man sshd_config` を参照してください。

演習

1. クライアントマシンにユーザー `sonya` としてログインし、リモートサーバー `halof` で、SSHに関連する以下の課題を実行してください。
   - リモートホストでユーザー `serena` として `~/.ssh` の中身を一覧表示するコマンドを実行し、ローカルマシンのターミナルに戻ってきてください（1行のコマンドで実行してください）。
   - リモートホストにユーザー `serena` としてログインしてください。
   - リモートホストにユーザー `sonya` としてログインしてください。
   - ローカルマシンの `~/.ssh/known_hosts` ファイルから `halof` に属している鍵をすべて削除してください。
   - クライアントマシンで、256ビットの `ecdsa` 鍵ペアを作成してください。
   - クライアントマシンで、256ビットの `ed25519` 鍵ペアを作成してください。
2. 以下のステップを、SSH認証エージェント を使ってSSH接続をする適切な順番に並べ替えてください。
   - クライアント側で `ssh-agent /bin/bash` を実行して 認証エージェント の新しいBashシェルを起動します。
   - クライアント側で `ssh-keygen` を実行して鍵ペアを作成します。
   - クライアント側で `ssh-add` を実行して秘密鍵をメモリの安全な領域に追加します。
   - リモートホストでログインしたいユーザーの `~/.ssh/authorized_keys` ファイルにクライアントの公開鍵を追加します。
   - リモートホストでログインしたいユーザーの `~/.ssh` を作成します（`~/.ssh` が存在していない場合）。
   - リモートサーバーに接続します。

   正しい順序は次のとおりです。

Step 1:	
Step 2:	
Step 3:	
Step 4:	
Step 5:	

Step 6:	

3. ポート転送 の種類ごとに、関係するオプションとディレクティブを書き込んでください。

種類	**オプション**	**ディレクティブ**
ローカル		
リモート（リバース）		
X		

4. `ssh -L 8888:localhost:80 -Nf ina@halof` コマンドをクライアントマシンのターミナルに打ち込んで実行しました。クライアントマシンのブラウザで `http://localhost:8888` に接続しました。何が表示されますか？

発展演習

1. SSHのディレクティブに関して、次の問いに答えてください。
   - `root` ログインを許可する `/etc/ssh/sshd_config` のディレクティブは何ですか？
   - SSH接続できるアカウント（ユーザー）を指定する `/etc/ssh/sshd_config` のディレクティブは何ですか？
2. クライアントとサーバーで同じユーザーを使っている場合に、SSH公開鍵認証ができるようにクライアントの公開鍵をサーバーに簡単に追加できるコマンドは何ですか？
3. リモートサーバー `halof` で実行しているSSHプロセスを通じて、非特権ポート（番号が1024以上のポート）である、8080で `www.gnu.org`、8585で `www.melpa.org` のウェブサイトに接続する、ローカルポート転送を1行のコマンドで実行してください。リモートサーバーではユーザー `ina` を使ってください。`-Nf` オプションを指定します。

まとめ

このレッスンでは、SSH（Secure　Shell）　プロトコルを利用してサーバーとクライアントの間の通信を暗号化する OpenSSH の説明をしました。以下の事柄を学びました。

- リモートサーバーへのログイン
- リモートコマンド実行
- 鍵ペアの作成
- 公開鍵認証
- セキュリティと利便性を両立させるための 認証エージェント の利用
- OpenSSH がサポートしている公開鍵暗号方式のアルゴリズム（`RSA`、`DSA`、`ecdsa`、`ed25519`）
- OpenSSH ホスト鍵の役割
- ローカルポート転送、リモートポート転送、X11転送

このレッスンでは、次のコマンドについて説明しました。

`ssh`
　リモートマシンにログインしてコマンドを実行します。

`ssh-keygen`
　認証鍵を生成、管理、変換します。

`ssh-agent`
　OpenSSHの認証エージェントです。

`ssh-add`
　認証エージェントに秘密鍵を追加します。

演習の解答

1. クライアントマシンにユーザー `sonya` としてログインし、リモートサーバー `halof` で、SSHに関連する以下の課題を実行してください。
   - リモートホストでユーザー `serena` として `~/.ssh` の中身を一覧表示するコマンドを実行し、ローカルマシンのターミナルに戻ってきてください（1行のコマンドで実行してください）。

     ```
     ssh serena@halof ls .ssh
     ```

   - リモートホストにユーザー `serena` としてログインしてください。

     ```
     ssh serena@halof
     ```

   - リモートホストにユーザー `sonya` としてログインしてください。

     ```
     ssh halof
     ```

   - ローカルマシンの `~/.ssh/known_hosts` ファイルから `halof` に属している鍵をすべて削除してください。

     ```
     ssh-keygen -R halof
     ```

   - クライアントマシンで、256ビットの `ecdsa` 鍵ペアを作成してください。

     ```
     ssh-keygen -t ecdsa -b 256
     ```

   - クライアントマシンで、256ビットの `ed25519` 鍵ペアを作成してください。

     ```
     ssh-keygen -t ed25519
     ```

2. 以下のステップを、SSH認証エージェント を使ってSSH接続をする適切な順番に並べ替えてください。
   - クライアント側で `ssh-agent /bin/bash` を実行して 認証エージェント の新しいBashシェルを起動します。
   - クライアント側で `ssh-keygen` を実行して鍵ペアを作成します。
   - クライアント側で `ssh-add` を実行して秘密鍵をメモリの安全な領域に追加します。
   - リモートホストでログインしたいユーザーの `~/.ssh/authorized_keys` ファイルにクライアントの公開鍵を追加します。
   - リモートホストでログインしたいユーザーの `~/.ssh` を作成します（`~/.ssh` が存在していない場合）。

- リモートサーバーに接続します。

  正しい順序は次のとおりです。

Step 1:	クライアント側で `ssh-keygen` を実行して鍵ペアを作成します。
Step 2:	リモートホストでログインしたいユーザーの `~/.ssh` を作成します（`~/.ssh` が存在していない場合）。
Step 3:	リモートホストでログインしたいユーザーの `~/.ssh/authorized_keys` ファイルにクライアントの公開鍵を追加します。
Step 4:	クライアント側で `ssh-agent /bin/bash` を実行して 認証エージェント の新しいBashシェルを起動します。
Step 5:	クライアント側で `ssh-add` を実行して秘密鍵をメモリの安全な領域に追加します。
Step 6:	リモートサーバーに接続します。

3. ポート転送 の種類ごとに、関係するオプションとディレクティブを書き込んでください。

種類	**オプション**	**ディレクティブ**
ローカル	`-L`	`AllowTcpForwarding`
リモート（リバース）	`-R`	`GatewayPorts`
X	`-X`	`X11Forwarding`

4. `ssh -L 8888:localhost:80 -Nf ina@halof` コマンドをクライアントマシンのターミナルに打ち込んで実行しました。クライアントマシンのブラウザで `http://localhost:8888` に接続しました。何が表示されますか？

   `halof` のウェブサーバーのホームページが表示されます。`localhost` というのは、リモートサーバーから見た `localhost` になります。

発展演習の解答

1. SSHのディレクティブに関して、次の問いに答えてください。
   - `root` ログインを許可する `/etc/ssh/sshd_config` のディレクティブは何ですか？

     `PermitRootLogin`

   - SSH接続できるアカウント（ユーザー）を指定する `/etc/ssh/sshd_config` のディレクティブは何ですか？

     `AllowUsers`

2. クライアントとサーバーで同じユーザーを使っている場合に、SSH公開鍵認証ができるようにクライアントの公開鍵をサーバーに簡単に追加できるコマンドは何ですか？

   `ssh-copy-id`

3. リモートサーバー `halof` で実行しているSSHプロセスを通じて、非特権ポート（番号が1024以上のポート）である、8080で `www.gnu.org`、8585で `www.melpa.org` のウェブサイトに接続する、ローカルポート転送を1行のコマンドで実行してください。リモートサーバーではユーザー `ina` を使ってください。`-Nf` オプションを指定します。

   ```
   ssh -L 8080:www.gnu.org:80 -L 8585:www.melpa.org:80 -Nf ina@halof
   ```

110.3 レッスン 2

Certificate:	LPIC-1
Version:	5.0
Topic:	110 セキュリティ
Objective:	110.3 暗号化によるデータの保護
Lesson:	2 of 2

はじめに

前のレッスンでは、OpenSSH を使ってリモートホストにログインすることと、そこでやり取りする情報を暗号化する方法を説明しました。リモートホストにログインせず、ファイルやメールを暗号化して、覗き見されることなく安全に受け手まで届けたいという場面もあるでしょう。改ざんされないようにファイルやメールにデジタル署名をする必要もあるかもしれません。

こうした場面では、GNU Privacy Guard（GnuPGやGPGと略されます） を使えます。これは、プロプライエタリなソフトウェアである Pretty Good Privacy（PGP） を、フリーでオープンなソフトウェアとして実装したものです。GPGは、Internet Engineering Task Force（IETF）の OpenPGPワーキンググループ がRFC 4880において定義した OpenPGP 規格に準拠しています。このレッスンでは、GNU Privacy Guardの基本を概観します。

GnuPGの基本設定・利用・失効

SSHと同様に、GPGは 非対称暗号（公開鍵暗号） の仕組みを利用しています。ユーザーは 秘密鍵と 公開鍵 で構成される鍵ペアを生成します。公開鍵と秘密鍵には数学的な関連があり、一方の鍵で暗号化されたものは他方の鍵でしか復号できません。GPGを使ってやり取りをするには、ユーザーの公開鍵を通信相手に配布する必要があります。

GnuPGの設定と利用

GPGを使って作業するコマンドは `gpg` です。このコマンドには様々なオプションがあります。ユーザー `carol` が鍵ペアを作成するところから始めましょう。そのためには、`gpg --gen-key` コマンドを実行します。

```
carol@debian:~$ gpg --gen-key
gpg (GnuPG) 2.2.12; Copyright (C) 2018 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/carol/.gnupg' created
gpg: keybox '/home/carol/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name:

(...)
```

設定ディレクトリ `~/.gnupg` と公開鍵の鍵束 `~/.gnugpg/pubring.kbx` が作成されたことなどが通知されたあとに、名前とメールアドレスの入力を求められます。

```
(...)
Real name: carol
Email address: carol@debian
You selected this USER-ID:
    "carol <carol@debian>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit?
```

入力した名前とメールアドレスを組み合わせた `USER-ID` が表示されているものでよければ、`O`を押してください。すると、次にパスフレーズの入力を求められます（複雑なパスフレーズにすることが推奨されます）。

```
 ┌──────────────────────────────────────────────────────┐
┐
 | Please enter the passphrase to                       |
 | protect your new key                                 |
 |                                                      |
 | Passphrase:   |

(...)
```

鍵本体とその他のファイルが作成されたことなどを伝えるメッセージが表示され、鍵の生成が完了します。

```
(...)
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/carol/.gnupg/trustdb.gpg: trustdb created
```

```
gpg: key 19BBEFD16813034E marked as ultimately trusted
gpg: directory '/home/carol/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/carol/.gnupg/openpgp-
revocs.d/D18FA0021F644CDAF57FD0F919BBEFD16813034E.rev'
public and secret key created and signed.

pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid                      carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

`~/.gnupg` ディレクトリ（GPGの設定ディレクトリ）の内容を見てみましょう。

```
carol@debian:~/.gnupg$ ls -l
total 16
drwx------ 2 carol carol 4096 Jul  3 23:34 openpgp-revocs.d
drwx------ 2 carol carol 4096 Jul  3 23:34 private-keys-v1.d
-rw-r--r-- 1 carol carol 1962 Jul  3 23:34 pubring.kbx
-rw------- 1 carol carol 1240 Jul  3 23:34 trustdb.gpg
```

ディレクトリとファイルについて説明します。

`opengp-revocs.d`
: 鍵ペアと同時に作られた失効証明書が保管されます。失効証明書にアクセスできると鍵を失効させることができてしまうので、このディレクトリのパーミッションは本人限定になっています（失効については次のサブセクションで詳説します）。

`private-keys-v1.d`
: 秘密鍵を保管するディレクトリであり、パーミッションは本人限定になっています。

`pubring.kbx`
: 公開鍵の鍵束（key ring）です。作成した自分の公開鍵と、インポートした他者の公開鍵を保管します。

`trustdb.gpg`
: 信用データベースです。信用の輪（Web of Trust）という概念と関係しています（このレッスンの範囲外です）。

NOTE
GnuPG 2.1 では大きな変更がありました。`secring.gpg` ファイルの代わりに `private-keys-v1.d` が、`pubring.gpg` ファイルの代わりに `pubring.kbx` が導入されました。

鍵ペアの作成後に `gpg --list-keys` を実行すると、公開鍵の鍵束の内容が表示され、公開鍵を確認できます。

```
carol@debian:~/.gnupg$ gpg --list-keys
/home/carol/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
```

```
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid          [ultimate] carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

16進数の文字列 `D18FA0021F644CDAF57FD0F919BBEFD16813034E` が 公開鍵指紋 です。

NOTE

`USER-ID`（上記の例では `carol`）とは別に、`KEY-ID` があります。`KEY-ID` は16進数表記の公開鍵指紋の最後の8桁（上記の例では `6813 034E`）です。`gpg --fingerprint USER-ID` コマンドを実行すると、自分の鍵指紋を確認できます。

鍵の配布と失効

自分の公開鍵を作成したので、将来の通信相手に配布できるように、その公開鍵をファイルに エクスポート して保存します。通信相手は、その公開鍵を使って、ファイルやメッセージを暗号化できます（その公開鍵とペアになる秘密鍵を持っているのは自分だけですから、そのファイルやメッセージを復号できるのは自分だけだということになります）。同様に、通信相手は、その公開鍵を使って、自分が秘密鍵で暗号化・署名したファイルやメッセージを復号・検証できます。公開鍵をファイルにエクスポートして保存するには、`gpg --export` のあとに `USER-ID` を指定して実行し、適当なファイル名を指定してその出力をリダイレクトします。

```
carol@debian:~/.gnupg$ gpg --export carol > carol.pub.key
carol@debian:~/.gnupg$ ls
carol.pub.key  openpgp-revocs.d  private-keys-v1.d  pubring.kbx  trustdb.gpg
```

NOTE

`gpg --export` で `-a` ないし `--armor` オプションを指定すると（例：`gpg --export --armor carol > carol.pub.key`）、デフォルトのOpenPGPバイナリ形式ではなくASCII armoredテキスト形式で出力するので、メールで安全に送れます。

先述したように、公開鍵ファイル（`carol.pub.key`）を、情報をやり取りしたい通信相手に送らなければなりません。一例として、`scp`（Secure CoPy）を使い、公開鍵ファイルをリモートサーバー `halof` の `ina` に送ります。

```
carol@debian:~/.gnupg$ scp carol.pub.key ina@halof:/home/ina/
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol.pub.key                                                                 100%
1740   775.8KB/s   00:00
carol@debian:~/.gnupg$
```

`ina` は `carol.pub.key` を取得できたので、これを使ってファイルを暗号化して `carol` に送れます（次の節で実演します）。

NOTE

鍵サーバー を利用して公開鍵を配布するという方法も広く使われています。`gpg --keyserver keyserver-name --send-keys KEY-ID` コマンドで公開鍵を鍵サーバーにアップロードすると、通信相手は `gpg --keyserver keyserver-name --recv-keys KEY-ID` コマンドでその公開鍵を入手（インポート）できます。（訳注：`keys.openpgp.org` など、公開されているPGP鍵サーバーがいくつもあります。PGP鍵サーバーに公開鍵を登録しておけば、通信相手がメールアドレスから検索して、あなたの公開鍵を見つけてくれます。）

この節の最後に、鍵の失効について説明します。秘密鍵が漏洩した場合や秘密鍵の利用を終了する場合には、鍵を失効させるべきです。鍵を失効させるには、まず、`gpg` コマンドで `--gen-revoke` オプションと `USER-ID` を指定して、失効証明書を作成します。`--gen-revoke` オプションの前に `--output` オプションとファイル名を指定すると、失効証明書をターミナルに出力する代わりにファイルに保存します。このコマンドを実行すると表示されるメッセージについて特に説明する必要はないでしょう（訳注：鍵を失効させる理由の選択と備考の記載が求められ、失効証明書を厳重に保管するように警告する内容が表示されます）。

```
sonya@debian:~/.gnupg$ gpg --output revocation_file.asc --gen-revoke sonya

sec  rsa3072/0989EB7E7F9F2066 2020-07-03 sonya <sonya@debian>

Create a revocation certificate for this key? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified  （理由を指定しない）
  1 = Key has been compromised  （鍵の漏洩）
  2 = Key is superseded  （鍵の切り替え）
  3 = Key is no longer used  （鍵の破棄）
  Q = Cancel
(Probably you want to select 1 here)
Your decision? 1
Enter an optional description; end it with an empty line:
> My laptop was stolen.
>
Reason for revocation: Key has been compromised
My laptop was stolen.
Is this okay? (y/N) y
ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
your media become unreadable.  But have some caution:  The print system of
your machine might store the data and make it available to others!
（要約：安全なメディアにファイルを保管します。誰かに漏洩すると悪者はあなたの鍵を無効化できます。メディアが読めなくなることに備えて印刷しておくとよいでしょう。）
```

失効証明書が `revocation_file.asc` ファイルに保存されます（ASCII形式だということを示すために `.asc` というサフィックスにしました）。

```
sonya@debian:~/.gnupg$ ls
openpgp-revocs.d  private-keys-v1.d  pubring.kbx  revocation_file.asc  trustdb.gpg
sonya@debian:~/.gnupg$ cat revocation_file.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: This is a revocation certificate

iQHDBCABCgAtFiEEiIVjfDnnpieFi0wvnlcN6yLCeHEFAl8ASx4PHQJzdG9sZW4g
bGFwdG9wAAoJEJ5XDesiwnhxT9YMAKkjQiMpo9Uyiy9hyvukPPSrLcmtAGLk4pKS
pLZfzA5kxa+HPQwBglAEvfNRR6VMxqXUgUGYC/IAyQQM62oNAcY2PCPrxyJNgVF7
8l4mMZKvW++5ikjZwyg6WWV0+w6oroeo9qruJFjcu752p4T+9gsHVa2r+KRqcPQe
```

```
aZ65sAvsBJlcsUDZqfWUXg2kQp9mNPCdQuqvDaKRgNCHA1zbzNFzXWVd2X5RgFo5
nY+tUP8ZQA9DTQPBLPcggICmfLopMPZYB2bft5geb2mMi2oNpf9CNPdQkdccimNV
aRjqdUP9C89PwTafBQkQiONlsR/dWTFcqprG5KOWQPA7xjeMV8wretdEgsyTxqHp
v1iRzwjshiJCKBXXvz7wSmQrJ4OfiMDHeS4ipR0AYdO8QCzmOzmcFQKikGSHGMy1
z/YRlttd6NZIKjf1TD0nTrFnRvPdsZOlKYSArbfqNrHRBQkgirOD4JPI1tYKTffq
iOeZFx25K+fj2+0AJjvrbe4HDo5m+Q==
=umI8
-----END PGP PUBLIC KEY BLOCK-----
```

鍵を失効させるには、証明書を鍵にマージする必要があります。これは、失効証明書ファイルを鍵束にインポートすることで行います。

```
sonya@debian:~/.gnupg$ gpg --import revocation_file.asc
gpg: key 9E570DEB22C27871: "sonya <sonya@debian>" revocation certificate imported
gpg: Total number processed: 1
gpg:    new key revocations: 1
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2022-07-04
```

鍵の一覧を表示すると、revokedと記載されていることから、鍵が失効していることがわかります。

```
sonya@debian:~/.gnupg$ gpg --list-keys
/home/sonya/.gnupg/pubring.kbx
pub   rsa3072 2020-07-04 [SC] [revoked: 2020-07-04]
      8885637C39E7A627858B4C2F9E570DEB22C27871
uid           [ revoked] sonya <sonya@debian>
```

最後に、失効前の公開鍵を持っている人（鍵サーバーにストアした公開鍵を含む）に、失効後の公開鍵を配布するのを忘れないようにしてください。

GPGを使ってファイルを暗号化・復号・署名・検証する

前節では、`carol` が `ina` に公開鍵を配布しました。GPGを使ってファイルを暗号化・復号・署名・検証する方法を説明します。

ファイルの暗号化と復号

`ina` が `carol` の公開鍵（`carol.pub.key`）を自分の鍵束にインポートするところから始めます。

```
ina@halof:~> gpg --import carol.pub.key
gpg: /home/ina/.gnupg/trustdb.gpg: trustdb created
gpg: key 19BBEFD16813034E: public key "carol <carol@debian>" imported
gpg: Total number processed: 1
gpg:               imported: 1
ina@halof:~> gpg --list-keys
/home/ina/.gnupg/pubring.kbx
----------------------------
```

```
pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid           [ unknown] carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

次に、メッセージを書いたファイルを作成し、gpg コマンドでそのファイルを暗号化します。carol の公開鍵には署名がないので、その鍵が本当に carol のものであると信用するかどうか尋ねられます。（訳注：公開鍵には別の人の署名を付けることができます。--lsign-key を使用して自分で署名すれば、その鍵を信用したと見なされ、以後このような確認をされなくなります。また、公開鍵に共通の知人の署名が付けられていれば、社会的な意味で鍵を信用できると見なすことができるでしょう。これが「信用の輪」Web of Trust の考え方です。）

```
ina@halof:~> echo "This is the message ..." > unencrypted-message
ina@halof:~> gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-message
gpg: 0227347CC92A5CB1: There is no assurance this key belongs to the named user
sub  rsa3072/0227347CC92A5CB1 2020-07-03 carol <carol@debian>
 Primary key fingerprint: D18F A002 1F64 4CDA F57F  D0F9 19BB EFD1 6813 034E
      Subkey fingerprint: 9D89 1BF9 39A4 C130 E44B  1135 0227 347C C92A 5CB1

It is NOT certain that the key belongs to the person named
in the user ID.  If you really know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

実行した gpg コマンドについて、一つずつ見ていきましょう。

--output encrypted-message
: 元ファイルを暗号化したファイルの名前を指定します（ここでは encrypted-message というファイル名を指定しています）。

--recipient carol
: 受け手の USER-ID を指定します（ここでは carol を指定しています）。この指定をしなければ、--default-recipient を指定しない限り、コマンド実行時に USER-ID の入力を求められます。

--armor
: ASCII armoredテキスト形式を出力します。テキスト形式ですから、メールの本文に貼り付けられます。

--encrypt unencrypted-message
: 暗号化する元ファイルのパスを指定します。

scp コマンドで debian の carol に encrypted-message を送ります。

```
ina@halof:~> scp encrypted-message carol@debian:/home/carol/
carol@debian's password:
encrypted-message                                             100%  736     1.8MB/s   00:00
```

`carol` がこの `encrypted-message` ファイルをそのまま読もうとすると、暗号化されていて判読できません。

```
carol@debian:~$ cat encrypted-message
-----BEGIN PGP MESSAGE-----

hQGMAwInNHzJKlyxAQv/brJ8Ubs/xya35sbv6kdRKm1C7ONLxL3OueWA4mCs0Y/P
GBna6ZEUCrMEgl/rCyByj3Yq74kuiTmzxAIRUDdvHfj0TtrOWjVAqIn/fPSfMkjk
dTxKo1i55tLJ+sj17dGMZDcNBinBTP4U1atuN71A5w7vH+XpcesRcFQLKiSOmYTt
F7SN3/5x5J6io4ISn+b0KbJgiJNNx+Ne/ub4Uzk4NlK7tmBklyC1VRualtxcG7R9
1klBPYSld6fTdDwT1Y4MofpyILAiGMZvUR1RXauEKf7OIzwC5gWU+UQPSgeCdKQu
X7QL0ZIBS0Ug2XKrO1k93lmDjf8PWsRIml6n/hNelaOBA3HMP0b6Ozv1gFeEsFvC
IxhUYPb+rfuNFTMEB7xIO94AAmWB9N4qknMxdDqNE8WhA728Plw6y8L2ngsplY15
MR4lIFDpljA/CcVh4BXVe9j0TdFWDUkrFMfaIfcPQwKLXEYJp19XYIaaEazkOs5D
W4pENN0YOcX0KWyAYX6r0l8BF0rq/HMenQwqAVXMG3s8ATuUOeqjBbR1x1qCvRQP
CR/3V73aQwc2j5ioQmhWYpqxiro0yKX2Ar/E6rZyJtJYrq+CUk8O3JoBaudknNFj
pwuRwF1amwnSZ/MZ/9kMKQ==
=g1jw
-----END PGP MESSAGE-----
```

`carol` は秘密鍵を持っていますから、`gpg` コマンドを `--decrypt` オプションと暗号化されたファイルのパスを指定して実行することで、このファイルを復号できます（秘密鍵のパスフレーズの入力を求められます）。

```
carol@debian:~$ gpg --decrypt encrypted-message
gpg: encrypted with 3072-bit RSA key, ID 0227347CC92A5CB1, created 2020-07-03
      "carol <carol@debian>"
This is the message ...
```

`--output` オプションとファイル名を指定すると、復号したメッセージを新しいファイルに保存できます。

```
carol@debian:~$ gpg --output unencrypted-message --decrypt encrypted-message
gpg: encrypted with 3072-bit RSA key, ID 0227347CC92A5CB1, created 2020-07-03
      "carol <carol@debian>"
carol@debian:~$ cat unencrypted-message
This is the message ...
```

ファイルの署名と検証

GPGは、ファイルを暗号化するだけでなく、署名するためにも使えます。`--sign` オプションとファイルパスを指定します。新しいメッセージを書いた `message` という名前のファイルを作成し、`--sign` オプションで署名します（秘密鍵のパスフレーズの入力を求められます）。（訳注：ここで言うところの署名は、ファイルのハッシュ値を署名者の秘密鍵で暗号化した証明書を添付し、受信者は署名を署名者の公開鍵で復号して、取り出したハッシュ値とファイルのハッシュ値を比較することで、秘密鍵の所有者が署名したこととファイルが改ざんされていないことを確認する仕組みのデジタル署名です。）

```
carol@debian:~$ echo "This is the message to sign ..." > message
```

```
carol@debian:~$ gpg --output message.sig --sign message
(...)
```

実行した gpg コマンドについて、一つずつ見ていきましょう。

--output message.sig
署名をしたファイルの名前を指定します（ここでは message.sig というファイル名を指定しています）。

--sign message
元ファイルのパスを指定します。

NOTE --sign オプションを指定すると、圧縮されてから署名されます。出力はバイナリ形式になります。

scp message.sig ina@halof:/home/ina コマンドで、署名したファイルを halof の ina に送ります。ina は gpg コマンドを --verify オプションとファイルパスを指定して実行することで、署名されたファイルを検証できます。

```
ina@halof:~> gpg --verify message.sig
gpg: Signature made Sat 04 jul 2020 14:34:41 CEST
gpg:                using RSA key D18FA0021F644CDAF57FD0F919BBEFD16813034E
gpg: Good signature from "carol <carol@debian>" [unknown]
(...)
```

ファイルの中身を読みたければ、--output オプションとファイル名を指定して、新しいファイルに保存します（ここでは message というファイル名にしました）。

```
ina@halof:~> gpg --output message --decrypt message.sig
gpg: Signature made Sat 04 jul 2020 14:34:41 CEST
gpg:                using RSA key D18FA0021F644CDAF57FD0F919BBEFD16813034E
gpg: Good signature from "carol <carol@debian>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: D18F A002 1F64 4CDA F57F  D0F9 19BB EFD1 6813 034E
ina@halof:~> cat message
This is the message to sign ...
```

GPGエージェント

gpg-agent に軽く触れてこのレッスンを終えます。gpg-agent は、gpg の呼び出しに応じて起動する、GPGの秘密鍵を管理するデーモンです。gpg-agent --help ないし gpg-agent -h を実行すると、概要が表示されます。

```
carol@debian:~$ gpg-agent --help
gpg-agent (GnuPG) 2.2.4
libgcrypt 1.8.1
Copyright (C) 2017 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Syntax: gpg-agent [options] [command [args]]
Secret key management for GnuPG

Options:

     --daemon                      run in daemon mode (background)
     --server                      run in server mode (foreground)
     --supervised                  run in supervised mode
 -v, --verbose                     verbose
 -q, --quiet                       be somewhat more quiet
 -s, --sh                          sh-style command output
 -c, --csh                         csh-style command output
(...)
```

NOTE

詳細は `gpg-agent` のマニュアルページに譲りますが、`ssh-agent` と同様に秘密鍵のパスフレーズを一定時間キャッシュして、ユーザーによる再入力回数を減らします。また、`ssh-agent` の代わりに使用することもできます。

演習

1. 以下の表の空欄を正しいファイル名で埋めてください。

説明	ファイル名
信用データベース	
失効証明書のディレクトリ	
秘密鍵のディレクトリ	
公開鍵の鍵束	

2. 次の問いに答えてください。
   - GnuPG で用いられている暗号化の種類は何ですか？
   - 公開鍵暗号の2つの主な構成要素は何ですか？
   - 公開鍵指紋 `07A6 5898 2D3A F3DD 43E3 DA95 1F3F 3147 FA7F 54C7` の KEY-ID は何ですか？
   - グローバルレベルで公開鍵を配布するための方法は何ですか？

3. 秘密鍵の失効に関して、以下のステップを適切な順番に並べ替えてください。
   - 失効させた公開鍵を配布します。
   - 失効証明書を作成します。
   - 失効証明書を鍵束にインポートします。

   正しい順序は次のとおりです。

Step 1:	
Step 2:	
Step 3:	

4. ファイルの暗号化に関するコマンド `gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-message` 中の `--armor` オプションはどういう働きをしますか？

発展演習

1. `gpg` コマンドのオプションにはロングオプションとショートオプションがあります。以下の表のロングオプションに対応するショートオプションを埋めてください。

ロングオプション	ショートオプション
`--armor`	
`--output`	
`--recipient`	
`--decrypt`	
`--encrypt`	
`--sign`	

2. 鍵のエクスポートに関する次の問いに答えてください。
   - すべての公開鍵を `all.key` という名前のファイルにエクスポートするコマンドは何ですか？
   - すべての秘密鍵を `all_private.key` という名前のファイルにエクスポートするコマンドは何ですか？

3. 鍵の管理作業をメニューで対話的に行う `gpg` コマンドのオプションは何ですか？

4. クリアテキストに署名する `gpg` コマンドのオプションは何ですか？

まとめ

このレッスンでは、ファイルを暗号化・復号・署名・検証する、GNU Privacy Guard を取り上げ、以下の事柄を説明しました。

- 鍵ペアの生成
- 鍵束の一覧表示
- `~/.gnupg` ディレクトリの内容
- `USER-ID` と `KEY-ID`
- 公開鍵を通信相手に配布する方法
- 公開鍵の鍵サーバーを通じた配布
- 秘密鍵の失効
- ファイルの暗号化と復号
- ファイルの署名と検証
- GPG-Agent の基礎

このレッスンでは、次のコマンドについて説明しました。

`gpg`
OpenPGP で暗号化や署名をします。

演習の解答

1. 以下の表の空欄を正しいファイル名で埋めてください。

説明	ファイル名
信用データベース	`trustdb.gpg`
失効証明書のディレクトリ	`opengp-revocs.d`
秘密鍵のディレクトリ	`private-keys-v1.d`
公開鍵の鍵束	`pubring.kbx`

2. 次の問いに答えてください。
   - GnuPG で用いられている暗号化の種類は何ですか？

     非対称暗号（公開鍵暗号）です。（訳注：データの暗号化には対称暗号（共通鍵暗号）も併用されます。）

   - 公開鍵暗号の2つの主な構成要素は何ですか？

     公開鍵と秘密鍵です。

   - 公開鍵指紋 `07A6 5898 2D3A F3DD 43E3 DA95 1F3F 3147 FA7F 54C7` の KEY-ID は何ですか？

     `FA7F 54C7` です。

   - グローバルレベルで公開鍵を配布するための方法は何ですか？

     鍵サーバーを利用します。

3. 秘密鍵の失効に関して、以下のステップを適切な順番に並べ替えてください。
   - 失効させた公開鍵を配布します。
   - 失効証明書を作成します。
   - 失効証明書を鍵束にインポートします。

     正しい順序は次のとおりです。

Step 1:	失効証明書を作成します。
Step 2:	失効証明書を鍵束にインポートします。
Step 3:	失効させた公開鍵を配布します。

4. ファイルの暗号化に関するコマンド `gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-message` 中の `--armor` オプションはどういう働きをしますか？

   ASCII armoredテキスト形式で出力し、メールの本文に貼り付けられるようにします。

発展演習の解答

1. `gpg` コマンドのオプションにはロングオプションとショートオプションがあります。以下の表のロングオプションに対応するショートオプションを埋めてください。

ロングオプション	ショートオプション
`--armor`	`-a`
`--output`	`-o`
`--recipient`	`-r`
`--decrypt`	`-d`
`--encrypt`	`-e`
`--sign`	`-s`

2. 鍵のエクスポートに関する次の問いに答えてください。
   - すべての公開鍵を `all.key` という名前のファイルにエクスポートするコマンドは何ですか？

     `gpg --export --output all.key` ないし `gpg --export -o all.key` です。

   - すべての秘密鍵を `all_private.key` という名前のファイルにエクスポートするコマンドは何ですか？

     `gpg --export-secret-keys --output all_private.key` ないし `gpg --export-secret-keys -o all_private.key` です。（`--export-secret-keys` の部分を `--export-secret-subkeys` にすると、微妙に異なる結果になります。詳しくは `man gpg` を参照してください。）

3. 鍵の管理作業をメニューで対話的に行う `gpg` コマンドのオプションは何ですか？

   `--edit-key` です。

4. クリアテキストに署名する `gpg` コマンドのオプションは何ですか？

   `--clearsign` です。

覚え書き

PDF生成日: 2024-11-05

LPI学習教材（Learning Materials）のイニシアチブは、Linux Professional Institute（https://lpi.org）が保持しています。学習教材とその翻訳は https://learning.lpi.org にあります。

この覚え書きおよびプロジェクト全体に関する質問やコメントは、learning@lpi.org 宛てに電子メールでお送り下さい。