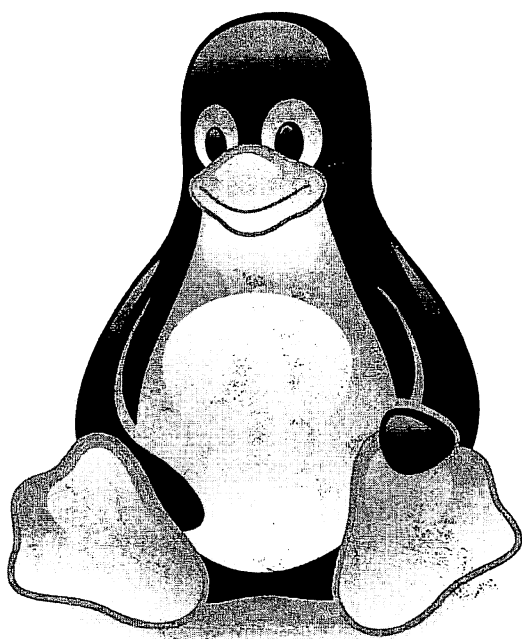


# Linux 入門の入門



暗黒通信団

# 1 はじめに

Windows や macOS<sup>\*1</sup>等の OS<sup>オペレーティングシステム</sup>（オペレーションシステム）に慣れ親しんだ方が学校や仕事で初めて Linux<sup>\*2</sup>に触れた時、まずコマンドの使い方で当惑するのではないのでしょうか。Windows でのコマンドプロンプトや MacOSX のターミナルを使いこなしてきた方はともかく、今まで主にマウスやタッチパネルでの操作で様々なファイル処理、アプリケーションの実行などを行ってきた方には、何をどうすればいいのか当惑するでしょう。

Linux では ターミナル（Terminal, 端末）と呼ばれるウインドウを開いて、様々な コマンド を実行することにより多種多様な処理ができます<sup>\*3</sup>。最近はマウスやタッチパネルで操作を行う GUI<sup>ジョーユーアイ</sup> によるシステム管理が容易にはなってきましたが、各々の処理ではターミナルでのコマンド実行が欠かせません。ところが、Linux は様々なコマンドがあり、どのコマンドをどう使えばいいのかが入門者にとっての壁になっているように思われます。Linux のコマンドに関する本は出版されていますが、詳しく書かれているためにページ数が多く、持ち歩くには不便です。電子書籍版ですと携帯は楽ですが、いずれにせよ一冊数千円します。また、「入門」という割には最初から難しい本もあります。一方、インターネット上にも Linux に関する様々な解説はありますが、ちょっとしたコツやポイントなどが欠けていることがあります。

そこで、「Linux のコマンドを、ポイントを押さえて最小限だけ説明する」持ち歩ける、入門のための入門書を目指して執筆してみました。単にコマンドを列挙するだけでなく、利用上便利なオプション、注意すべきオプションなども挙げています。書店で販売されている他の本よりもずっと安価ですので、もし Linux を実際に操作していて気になった所があれば、余白に自由にメモしてもいいでしょう<sup>\*4</sup>。まずは基本操作に必要なコマンドを使いこなして、Linux の世界を実感してみましょう。

本書ではターミナルで入力するコマンドは、フォントを変えています。1 と l, 0 と O と o などを間違えないよう、以下のように記すことにします。

数字：0123456789

ローマ字の大文字：ABCDEFGHIJKLMOPQRSTUVWXYZ

ローマ字の小文字：abcdefghijklmnpqrstuvwxyz

---

<sup>\*1</sup> 当初は Mac OS X, バージョン 10.8 (Lion) より OS X, バージョン 10.12 (Sierra) より macOS と名称が変わりました。

<sup>\*2</sup> 「リナックス」と読まれる事が一般的ですが、リヌックス、ライナックスと読まれることもあります。

<sup>\*3</sup> CUI<sup>シーユーアイ</sup> と呼ばれる操作方法です。

<sup>\*4</sup> ボロボロ、ぐちゃぐちゃになったら、新たに買っていただけると幸いです。

記号：< > | . + - \_ \* / ~ @ ' "

キー入力で **ESC** と記したところはそれぞれ、エスケープ (ESC) キーを表します。**Ctrl** が現れた場合、例えば **Ctrl**-c と記したところは、コントロール (control) キーを押しながら c キーを押してください。↵ はエンターキーを表します。本文でコマンド実行の際にエンターキーの入力が必要なところは ↵ を記していますが、一部省略しているところがありますので、入力しても反応が無い場合はエンターキーを入力するなど、適宜操作をお願いします。

(filename) や (directory name), (command) などのように斜字で書かれているところは、具体的に操作する際のファイル名、ディレクトリ名、コマンドなどを入力して下さい。

## 2 Linux で知っておいてもらいたいこと

### 2.1 Linux のディストリビューション

たとえば Windows には Home 版や Professional 版などの違いがあり、機能も大きく違います。Linux の場合には、OS の核心となるカーネル (kernel) と呼ばれる部分は共通ですが、基本的なツールやユーティリティ、そしてアプリケーションの管理方法に様々な種類があります。これらのツール、ユーティリティ、アプリケーションなど一式を利用できるようにまとめあげたものを ディストリビューション といいます。現在、世界中には非常に多くの種類のディストリビューションがあります。それらは主に Red Hat <sup>レッド ハット</sup>系、Debian <sup>デビアン</sup>系、その他に分類されます。Red Hat 系には、Red Hat Enterprise Linux, <sup>フェドラー セントオーエス</sup>Fedora, CentOS などがあります。Debian 系には Debian GNU/Linux, <sup>グニユー</sup>Ubuntu <sup>サブントウ</sup> などがあります。

ディストリビューションの違いで、いくつかのコマンドや設定に違いが現れます。本書では基本的なコマンドについて、なるべく Red Hat 系と Debian 系の両方で利用できるように掲載しています\*5。ご利用のディストリビューションにより使えるコマンド、使えないコマンドがありますので、注意しながらご利用下さい。

### 2.2 管理者と一般ユーザ

Linux は一台のパソコンで、複数のユーザが同時に利用できる OS です。さらに、複数の処理が同時並行になされます。たとえパソコンの前で操作していなくても、遠隔操作が可能です。中にはファイルを無意味に生成したり、CPU に無駄に大きな負荷をかける処 <sup>シービーユー</sup>

---

\*5 本書の執筆にあたり、CentOS 7.3, CentOS 6.9, Fedora 25, Debian 8.8, Ubuntu 17.04 での動作を検証しております。

理を行う、行儀の悪いユーザもいるかもしれません。

Linux ではユーザは主に、管理者（スーパーユーザ）と一般ユーザに分けられます。管理者は Linux 上で全ての権限を持ち、ユーザの作成や削除、他のユーザの処理の停止などを行うことができます\*6。管理者のユーザ名は root です。

一般ユーザは基本的に、自分自身の実行している処理や作成したディレクトリ、ファイルへの操作権限を持ちます。他のユーザの処理を停止したり、他のユーザのファイルを変更、削除することは一般にはできません。一般ユーザは「グループ」分けられ、グループごとにファイルの読み書きなどの権限を設定することができます。

Linux のシステムとして動作するプログラムの中には、OS の起動時点で動作を開始するものがあります。ただ、管理者の権限で実行してしまうと、万が一プログラムに問題があった場合に、悪意を持った第三者がプログラムの弱点について管理者権限を奪い取ってしまう恐れがあります。

実は、一般ユーザは実際に存在するユーザと結びついてなくてもいいようになっています。特定のプログラムを起動するために「一般ユーザ」を作成することもできます。起動時に実行されるプログラムに対して、そのプログラム専用の一般ユーザを作成しておけば、万が一プログラムの弱点をつかれても、管理者権限を奪い取られないようにできます。

## 2.3 絶対パスと相対パス

ターミナルを操作している時、ユーザはどこかのディレクトリ上にいます。ディレクトリはツリー上に連なって存在します。そのトップディレクトリを/と表します。現在、ユーザ自身がいるディレクトリを カレントディレクトリ といいます。ユーザ専用のディレクトリは ホームディレクトリ といいます。一般ユーザがファイルを操作するディレクトリは、たいていはホームディレクトリおよびそのサブディレクトリです。

ディレクトリ上のファイルを指定する際には 絶対パス と 相対パス があります。絶対パスはトップディレクトリである/からたどったパスの事を指します。相対パスは、ターミナルでの現在位置から見たパスの事を指します。具体例を挙げて考えてみます。ターミナルを開いたときのカレントディレクトリが/home/hoge だったとします。もし後述のエディタ vim が/usr/bin ディレクトリの中にある場合、

絶対パス: /usr/bin/vim


相対パス: ../../usr/bin/vim

となります。ドット 2 つは、1 つ上の階層のディレクトリ（親ディレクトリ）を表します。Linux の設定ファイルによっては、絶対パスでコマンドなどの位置を指し示さないとい

---

\*6 Linux ディストリビューションによっては、SELinux という追加機能により、管理者でも全ての権限を持たないことがあります。以下では例外とします。

けないことがありますので、ご注意ください。

ターミナル上でコマンドやプログラムを実行する際、例として挙げた vim を単に vim 

と入力するだけで実行できます。これはシステムで前もって「参照するディレクトリ」が設定されており、ターミナルを開く際にこれらのディレクトリの中に存在するコマンド、プログラムを探し出して実行するようになっているからです。自分で用意したプログラムを実行する場合、「参照するディレクトリ」に含まれていないならば、パスを記述しなければなりません。「参照するディレクトリ」に設定されているディレクトリは、以下のコマンドで確認できます。

echo \$PATH 

## 2.4 主なディレクトリ

絶対パスと相対パスについて説明しましたが、特に絶対パスでファイルなどを指定するときのため、Linux で使われる主なディレクトリについて知っておいたほうがいいでしょう。一般ユーザが用いることの多い、主なディレクトリを挙げておきます\*7。

- /home  
一般ユーザのホームディレクトリが存在するディレクトリです。ユーザ ID が foo の方は  
/home/foo  
がホームディレクトリになります。
- /bin, /usr/bin  
Linux で使用されるコマンド、プログラムが格納されるディレクトリです。
- /usr/local/bin  
Linux で管理者が追加したアプリケーションがインストールされることの多いディレクトリです。
- /var/tmp  
一時的にファイルを保存するディレクトリです。どのユーザでもファイルを置くことができますが、システムによっては一定期間後にファイルは削除されます。
- /etc  
OS やアプリケーションの設定ファイルが格納されるディレクトリです。格納されたファイルの大部分は一般ユーザは変更できず、一部は管理者しか閲覧できません。ファイルのアクセス権限については後述します。

---


\*7 初期設定の場合を挙げています。システムによっては管理者の方針により若干異なることがあります。

- `/proc`

CPU の型番やメモリの量など、ハードウェア情報が記録されているディレクトリです。


## 2.5 コマンドのマニュアルを読む

コマンドの使い方や「そのコマンドが一体何をするものか」ということを知るコマンドがあります。それが`man`です。“Manual” の略だと思ってもらえばいいでしょう。コマンドラインで


`man (command)` 

と入力すれば、そのコマンドが何をするものか、オプションとしてどういう文字を追加すれば何ができるかがわかります。表示されるマニュアルが長い場合にはスペースキーを送ればページが変わりますし、キーボードの↑↓ボタンでマニュアルを上下できます。q キーをせば終了します。何もわからなければ、まずは `man` で調べましょう。

基本操作は `man` で調べていただければいいのですが、詳しい情報が載っていないことがあります。man の出力結果として、「`info`中の‘`xxx`’<sup>インフォ</sup> エントリ」などと記述されることがあります。このとき、コマンドラインで

`info (command)` 

と入力すれば、`man` で表示されるものよりも詳しいマニュアルが表示されます。表示されたマニュアルの主な操作を挙げておきます。

- ↑↓：マニュアルを 1 行ずつスクロールします。
- \* (アスタリスク) にカーソルを移動して ：詳細な説明が表示されます。
- 1 : 1 画面戻ります。
- d: 先頭の画面に戻ります。
- ?: `info` は操作方法を表示します。
- q: `info` を終了します。

`info` で表示されるマニュアルは英語の場合があります。調べたいことについて日本語のマニュアルがなかったり、あるいはマニュアルそのものがない場合には、インターネット上で調べてみましょう。

## 2.6 複数のファイルなどを指定する方法

これから扱うファイルの操作、例えばファイルの移動やコピーでは、複数のファイルに対して行いたいことがあるのではと思います。100 個のファイル操作を行うのに、100 回もコマンドを入力するのは大変です。そこで役に立つのが 正規表現 と呼ばれる表現方法です。詳細は後の応用編のところで説明することにし、ここでは ワイルドカード と呼ばれる \* (アスタリスク) について説明します。\* は、「任意の 0 文字以上の文字列」を表

します。アスタリスクの位置は、先頭、途中、最後のいずれでも構いません。

例として、`hoge1.txt`、`hoge2.txt`、`hoge10.txt`、`Ahoge.txt`、`AhogeB.txt`、`foo.txt` という 6 つのファイルを考えます。\*の使い方で以下のようにファイルを指し示すようになります。

- `hoge*.txt`: `hoge1.txt`、`hoge2.txt`、`hoge10.txt` を指し示します。
- `*hoge.txt`: `Ahoge.txt` を指し示します。
- `*hoge*.txt`: `hoge1.txt`、`hoge2.txt`、`hoge10.txt`、`Ahoge.txt`、`AhogeB.txt` を指し示します。
- `*`: 全てのファイルを指し示します。

このようにワイルドカードは便利ですが、ファイル名の先頭が . (ドット) で始まるファイルについては、\*ではドットを指し示さないのをご注意下さい\*<sup>8</sup>。

ファイル名にスペースを入れてしまうとコマンドラインの操作で困ることになります。例として「`New year.txt`」というファイル名をつけた場合、後述のコマンドでファイルを操作しようとする、「`New`」というファイルをシステムは操作しようとして誤動作を起こします。スペースはコマンドなどの区切りとして使われてしまうためです。このような場合、' (シングルクォーテーション) や " (ダブルクォーテーション) でファイル名をはさみ、'`New year.txt`' や "`New year.txt`" と表すことで、操作が可能になります。

## 2.7 実行中のコマンドの強制終了

コマンドによっては処理に時間がかかったり、画面出力が膨大だったりします。そのような時に、コマンドを強制終了するキーがあります。[**Ctrl**]キーを押しながら **c** を押して下さい。これでたいていの場合は強制終了できますが<sup>9</sup>、できない場合には後述の `kill` などをご利用下さい。

## 3 よく使われるコマンド

ここでは、ユーザが使うことの多いコマンドを示します。オプションは複数を組み合わせることが可能なものが多いです。

### 3.1 ターミナルに関するコマンド

- ビードプリューディー  
1. `pwd` (現在のディレクトリを表示)

現在操作しているディレクトリの場所を表示するコマンドです。

---

<sup>8</sup>他にも、場合によっては\*が誤認識をされてエラーを起こしたりファイルを消去したりすることがありますので、注意が必要です。

`pwd` 


を実行すると、

`/home/foo`

のように表示されます。

2. <sup>シーディー</sup>`cd` (ターミナル上で操作するディレクトリを移動)


操作しているディレクトリを移動するコマンドです。

`cd (directory name)` 

で実行します。単に

`cd` 

を実行すると、ユーザのホームディレクトリに移動します。よく使うものは以下のようなものでしょう。

`cd ..` 

現在の親ディレクトリに移動します。

`cd ~/` 


ホームディレクトリに移動します。

3. <sup>エルエス</sup>`ls` (ファイル、ディレクトリ一覧を表示)


ファイルやディレクトリの情報を表示します。単に

`ls` 

を実行すると、数字、大文字、小文字で始まるファイル名の順に表示されますが、ファイル名の先頭がドットで始まるファイルは表示されません。Linux の設定ファイルはそれぞれのユーザのホームディレクトリの中に、先頭がドットで始まるファイル名で保存されているものがいくつかあります。これらのファイルも表示したい場合には

`ls -a` 

を実行しましょう。ls のオプションはたくさんあります。よく使われるオプションの例は以下の通りです。


`ls -l` 

ファイルの権限、所有ユーザ、所有グループ、ファイルサイズ、変更日時が表示されます。以下のように表示されます。


```
-rwxrwxr-x 1 users foo 8512 5月 2 13:40 a.out
-rw-rw-r-- 1 users foo 58 5月 2 13:40 hello.c
```

8512 や 58 はファイルサイズ (バイト)、表示がやや変ですが「5月 2 13:40」は変更日時です。左端の文字列については後で現れる `chmod` のところで説明します。




`ls -t` 

変更日時の順に表示されます。

`ls -R` 


ディレクトリ内のファイルを全て表示します。膨大な量のファイルが表示される恐れがあるので注意が必要です。

`ls hoge*` 


ファイル名の先頭に `hoge` を含むファイルだけが表示されます。

#### 4. <sup>ファインド</sup>`find` (ファイルの検索)

条件に従うファイルを探します。よく使われるオプションの例を示します。

`find (directory name) -name hoge` 


ファイル名に `hoge` を含むファイルを、指定したディレクトリおよびそのサブディレクトリから探します。


`find (directory name) -mtime n` 

`n` 日前に変更されたファイルを探します。

#### 5. <sup>ウィッチ ウェアイズ</sup>`which`, `whereis` (コマンド、プログラムのパスを表示)

コマンドやプログラムのパスを表示します。


`which (command)` 

`whereis (command)` 

両者の違いですが、`which` は 2.3 で挙げた「参照するディレクトリ」に存在するコマンドやプログラムを探し出して、パスを表示します。一方で `whereis` は「標準的な Linux ファイル階層」でコマンドやプログラムを探します。後述の管理者向けのコマンドは、一般ユーザに対しては「参照するディレクトリ」に設定されていない事が多いので `which` では表示されませんが、`whereis` では表示されます。さらに、それらのコマンドやプログラムに加えて、`man` で表示されるマニュアルのファイルのパスを表示します。


#### 6. <sup>クリア</sup>`clear` (ターミナルをクリア)

ターミナルに表示されている画面をクリアし、1 行目から入力できるようになります。

`clear` 

#### 7. <sup>ログアウト</sup>`logout` (ターミナルを終了)


ターミナルを終了します。


`logout` 

## 3.2 ファイル操作に関するコマンド

### 1. <sup>エルエヌ</sup>ln (リンクの作成)

ファイルやディレクトリのリンクを作成します。よく用いられるのは、Windows のショートカットに相当するシンボリックリンクです。


`ln -s (filename) (リンク先)` 

`ln -s (directory name) (リンク先)` 


このようにすると、別名のファイルやディレクトリ (リンクファイル) が作成されます。

### 2. <sup>シービー</sup>cp (ファイル、ディレクトリのコピー)


ファイルやディレクトリをコピーします。

`cp (old filename) (new filename)` 

新しいファイル名が、すでに存在するファイル名と一致すると、上書きされてしまうので注意して下さい。

`cp -R (old directory name) (new directory name)` 


ディレクトリを丸ごとコピーします。


`cp -p (old filename) (new filename)` 

元のファイルの変更日時を保ったままコピーします。

### 3. <sup>エムヴィ</sup>mv (ファイル、ディレクトリの移動、名前変更)


ファイルやディレクトリを移動、もしくは名前の変更をします。

`mv (old filename) (new filename)` 

`mv (directory filename) (directory name)` 

新しいファイル名が、すでに存在するファイル名と一致すると、上書きされてしまうので注意して下さい。

`cp` とは異なり、`mv` ではディレクトリの移動、名称の変更でもオプションは不要です。

`mv (filename) (directory name)` 


として既に存在するディレクトリを指定すると、ディレクトリの中にファイルが移動されます\*<sup>9</sup>。

### 4. <sup>アールエム</sup>rm (ファイルの削除)


---

\*<sup>9</sup> ここで誤って `mv *` としてディレクトリを指定し忘れると、ファイル同士の上書きが発生するので注意です。

ファイルを削除します。

`rm (filename)` 


で指定したファイルを削除できます。主なオプションなどは以下の通りです。

`rm hoge*` 

ファイル名が `hoge` で始まるファイルがすべて削除されます。

`rm *` 

ディレクトリやファイル名の先頭がドットで始まるファイルを除いて、ファイルがすべて削除されてしまいます。


`rm -r (directory name)` 

指定されたディレクトリと、そこに含まれる全てのディレクトリ、ファイルが削除されます。

削除されたファイルやディレクトリの復元は困難なので、ファイル削除のオプションの取り扱いには注意して下さい。

5. <sup>タッチ</sup> `touch` (ファイルのアクセス時刻, 修正時刻の変更)


ファイルのアクセス時刻, 修正時刻の変更を行います。

`touch (filename)` 


もし、指定したファイルが存在しない場合には、0 バイトのファイルを作成します。

6. <sup>メイクディレクトリ</sup> `mkdir` (ディレクトリの作成)

ディレクトリを作成します。 `hoge` というディレクトリを作りたい場合、

`mkdir hoge` 

を実行して下さい。もし、 `hoge` というディレクトリの中にさらに `hoge2` というディレクトリを作ろうとして、

`mkdir hoge/hoge2` 


と入力した時、 `hoge` ディレクトリがなければ作成に失敗します。このような時は

`mkdir -p hoge/hoge2` 

のようにオプションをつければ、 `hoge` ディレクトリが作成され、さらにその中に `hoge2` ディレクトリが作成されます。

7. <sup>アールエムディレクトリ</sup> `rmdir` (ディレクトリの削除)

ディレクトリを削除します。ただし、ディレクトリの中は空でなければなりません<sup>\*10</sup>。

`rmdir (directory name)` 

---

<sup>\*10</sup> 特にファイル名の先頭がドットで始まる設定ファイルを見落としがちです。

8. ター **tar** (ファイルの格納, 展開)

複数のファイルをまとめて 1 つのファイルに格納したり, 逆に格納されたファイルを展開したりします。たとえば `text1.txt`, `text2.txt`, `text3.txt` を 1 つのファイル `text.tar` に格納したい場合には,

```
tar cf text.tar text1.txt text2.txt text3.txt
```

とします。拡張子 `tar` のファイル名の後に, 格納したいファイルすべてを列記します。格納はファイルのみならずディレクトリを指定することも可能です。大量のファイル, ディレクトリを 1 つのファイルに格納する場合には, ワイルドカードを使うと便利です。

逆に `text.tar` を展開したい場合には

```
tar xf text.tar
```

とします。展開したファイルと同名のファイルが存在する場合, 上書きされるので注意して下さい。

`tar` コマンドは下記の `gzip`, `gunzip`, `bzip2`, `bunzip2`, `xz`, `unxz` コマンドなどと一緒によく使われます。

9. ジーzip `gzip`, ジーアンzip `gunzip` (ファイルの圧縮, 展開)

ファイルの圧縮, 展開<sup>\*11</sup>を行います。 `gzip` で圧縮, `gunzip` で展開を行います。圧縮されたファイルは, 拡張子 `.gz` が付与されます。

```
gzip (filename)
```

```
gunzip (filename)
```

10. ビーzip2 `bzip2`, ビーアンzip2 `bunzip2` (ファイルの圧縮, 展開)

`gzip` に代わって普及してきている圧縮方法でのファイルの圧縮, および展開を行います。圧縮後のファイルサイズは `gzip` の場合に比べて小さくなりますが, 圧縮にかかる時間がやや長くなります。 `bzip2` で圧縮, `bunzip2` で展開を行います。圧縮されたファイルは, 拡張子 `.bz2` が付与されます。

```
bzip2 (filename)
```

```
bunzip2 (filename)
```


11. エックスzip `xz`, アンエックスzip `unxz` (ファイルの圧縮, 展開)


ファイルの圧縮, および展開を行います。圧縮後のファイルサイズは `gzip`, `bzip2` の場合に比べて小さくなりますが, 圧縮にかかる時間がやや長くなります。 `xz` で圧縮, `unxz` で展開を行います。圧縮されたファイルは, 拡張子 `.xz` が付与されま

---

<sup>\*11</sup> 圧縮の反対の操作を「伸張」「解凍」ということもあります。


す。


**xz** (filename) 

**unxz** (filename) 

12. ジップ アンジップ **zip, unzip** (ファイルの圧縮, 展開)

Windows で用いられることの多い圧縮方法のファイルの圧縮, および展開を行います。圧縮の際にはファイル名を指定します。以下では例として圧縮後のファイル名を *hoge.zip* としています。一つのファイルに複数のファイルをまとめることもできます。圧縮されたファイルには, Windows でも判別できるよう拡張子 *.zip* を付けておきます。

**zip** hoge.zip (filename1 filename2 ...) 

**unzip** (filename) 


13. チェンジモード **chmod** (アクセス権の変更)

ファイルのアクセス権を変更します。例として, **ls -l** を実行したときに表示された以下のファイルのアクセス権を変更することを考えます。

```
-rwxrwxr-x 1 users foo 8512 5月 2 13:40 a.out
-rw-rw-r-- 1 users foo  58 5月 2 13:40 hello.c
```

左側の *rwX* のところを, 最初の *-* を除いて 3 つ区切りで見てください<sup>\*12</sup>。 *r* は読み出し可能, *w* は書き込み可能, *x* は実行可能な権限があることを示します。また, 左から 3 文字ずつでそれぞれ, 所有するユーザ, 所有するグループ, その他のユーザに対する権限を表します。例で挙げると, *a.out* は所有するユーザ, 所有するグループには全ての権限があり, その他のユーザには書き込み可能な権限はありません。 *hello.c* は誰にも実行権限がありません。

権限を変更する場合, 記述方法には 8 進数を用いるもの と, 文字で表現するもの があります。8 進数の場合には 3 桁目が所有するユーザ, 2 桁目が所有するグループ, 1 桁目がその他のユーザに対する権限を表します。そして, 1, 2, 4 がそれぞれ実行可能, 書き込み可能, 読み出し可能な権限を表します。例えば *hello.c* に対し, 自分以外の権限をすべて無しにするには, 所有ユーザの桁を書き込み可能と読み出し可能な足し算である  $4 + 2 = 6$  とし, 他を 0 として,

**chmod 600** hello.c 

と実行します。すると, **ls -l** の実行結果は

```
-rw----- 1 users foo  58 5月 2 13:40 hello.c
```

---

<sup>\*12</sup> 最初の文字は, ディレクトリの場合は *d* になります。

のように変わります。一方で誰でも読み書き可能<sup>\*13</sup>にしたい場合には先ほどと同じく  $4 + 2 = 6$  ですので

```
chmod 666 hello.c
```

と実行します。すると、`ls -l` の実行結果は

```
-rw-rw-rw- 1 users foo 58 5月 2 13:40 hello.c
```

のように変わります。

8進数だとわかりにくいという方は、`r`、`w`、`x` という権限を表す文字、`u`、`g`、`o` という対象 (user, group, others) と、`+`、`-` を使っても変更可能です。上記の操作の場合にはそれぞれ、

```
chmod go-r hello.c
```

```
chmod o+w hello.c
```

と実行します。

#### 14. チェンジオーナー `chown` (所有者の変更)

ファイルやディレクトリの所有者を変更します。

```
chown (username).(group ID) (filename)
```

ユーザ名、グループ名のいずれかは省略可能です。所有グループだけを変更する場合はドットが必要ですが、所有ユーザだけを変更する場合はドットは不要です。ディレクトリ以下のファイル全ての所有権を変更する場合には、以下のオプションを使います。

```
chown -R (username).(group ID) (directory name)
```

## 3.3 プロセスに関するコマンド

### 1. ピーエス `ps` (プロセスの表示)

現在実行されているプロセスを表示します。オプションをつけないと、`ps` を実行しているターミナルに関わるプロセスのみが表示されます。もしユーザ名が `foo` の場合、

```
ps -u foo
```


というオプションをつけて実行すると、`foo` が実行しているプロセスすべてが表示されます。例としては以下のようになります。

---

<sup>\*13</sup> 他人にファイルを変更されたり削除されたりする恐れがあるので、普通はこんな設定はしません。ちなみに映画『オーメン』で扱われたように、キリスト教の世界では 666 は悪魔の数字とされます。

PID	TTY	TIME	CMD
4523	pts/0	00:00:00	bash
4541	pts/0	00:00:00	ps

一番左の PID の列に現れる数字は「プロセス番号」を表します。次の kill コマンドで必要となります。


ps -ef 

全ユーザに対しユーザ名を含めた全プロセスが表示されます。表示される項目も大きく増えます。


UID	PID	PPID	C	STIME	TTY	TIME	CMD
(中略)							
foo	4523	4521	0	15:11	pts/0	00:00:00	/bin/bash
foo	4615	4523	0	15:17	pts/0	00:00:00	ps -ef

## 2. <sup>キル</sup>kill (プロセスの終了)

プロセスを終了させます。ps コマンドで確認したプロセス番号 (PID) を入力します。当然のことながら、kill を実行する権限のないプロセスに対する操作はできません。

kill (PID) 

kill コマンドを実行すると、指定されたプロセスは終了に向けた処理を行ってから終了しますが、中には終了しないプロセスもあります。直ちに終了せよという場合には


kill -9 (PID) 

のようにオプションをつけて下さい。

## 3.4 ファイルの中身を見たり操作したりするコマンド

### 1. <sup>ダブルユーシー</sup>wc (ファイルの行数などを表示)


ファイルの行数、単語数、バイト数を表示するコマンドです。オプションをつけないとわかりにくいので、

wc -l (filename) 

とするといいでしょう。行数とファイル名のみが表示されます。

### 2. <sup>モア レス</sup>more, less (ファイルの中身を表示)


ファイルの中身を見るためのコマンドです。more はテキストファイルを 1 画面ずつ表示します。

more (filename) 

画面のスクロールはスペースキー、終了の場合は q キーを押して下さい。

テキストファイルでないファイルを表示しようとすると、ターミナルの表示が以後おかしくなるので、注意して下さい。


less は more に加えて、スクロールの際に下ではなく上へのスクロールもできます。

`less (filename)` 

操作は more の場合と類似していますが、1 画面戻す場合には b キーを押して下さい。表示を終了させるときは q キーを押して下さい。


### 3. キャット cat (ファイルの中身を出力)

ファイルを連結して出力します。more や less とは異なり、一気に表示されます。ファイルを複数指定した場合には、複数のファイルが一度に出力されます。


`cat (filename 1) (filename 2) ...` 

### 4. ゼットキャット zcat , ビーゼットキャット bzcac (圧縮されたファイルの中身を出力)

前述の cat を拡張したコマンドで、圧縮されたファイルを連結して出力します。ファイルを複数指定した場合には、複数のファイルが一度に出力されます。

`zcat (filename 1) (filename 2) ...` 


gzip で圧縮されたファイルを展開して出力します。

`bzcat (filename 1) (filename 2) ...` 


bzip2 で圧縮されたファイルを展開して出力します。

### 5. ディフ diff (2つのファイルの比較)


2つのファイル間の違いを探します。プログラムや設定ファイルを更新した場合、過去のものとどこが食い違うのか調べる際に有用です。

`diff (filename 1) (filename 2)` 


ファイル名 1 のファイルからファイル名 2 のファイルに対し、どこが変更されたかが表示されます。削除された行は先頭に <、追加された行は先頭に > がついて表示されます。変更された行は変更前後の両方が表示されます。いくつかのオプションを挙げておきます。

`diff -b (filename 1) (filename 2)` 


スペースの数が違うだけの場合は、違いを無視します。

`diff -B (filename 1) (filename 2)` 

空の行の有無については、違いを無視します。


`diff -i (filename 1) (filename 2)` 

ローマ字の大文字と小文字の違いを無視します。


`diff -q (filename 1) (filename 2)` 



ファイルが違いかどうかだけを表示します。

`diff -u (filename 1) (filename 2)` 

まず 2 つのファイルの更新日時を出力し、削除された行は先頭に `-`、追加された行は先頭に `+` がついて表示されます。変更された行の場合は `+-` が入ります。

`diff -c (filename 1) (filename 2)` 


オプション `-u` の場合と似ていますが、変更された行の場合は `!` になるなど若干異なります。

オプション `-u`、`-c` を用いた場合の利点は応用編で説明します。

さらに、3 つのファイル間の違いを比較する `diff3` というコマンドもありますが、詳細は `man` コマンドなどで調べてみて下さい。

## 6. <sup>ファイル</sup>file (ファイルのタイプを表示)

ファイルのタイプを表示します。


`file (filename)` 

Windows ではファイルを作成すると拡張子が自動的に付けられますが、Linux のエディタでファイルを作成すると拡張子をユーザがつけることになります。プログラムを作成していてソースファイルなのか実行ファイルのかなど、分からなくなったときに利用します。


## 7. <sup>エスケープ</sup>nkf (テキストファイルの文字コードを変換)

テキストファイルの漢字コードを変換します。PC で日本語を取り扱う場合には、歴史的経緯からいくつかの漢字コードが存在します。例えば Windows で扱われるものは Shift JIS と呼ばれます。一方で Linux などでは JIS や EUC と呼ばれる漢字コードが使われてきました。最近では Unicode (UTF-8) と呼ばれる規格が広まっています。このような様々な漢字コードで書かれたテキストファイルを変換するために `nkf` は用いられます。


主なオプションは以下の通りです。

`nkf -j (filename)` 


JIS コードを出力します。

`nkf -e (filename)` 

EUC コードを出力します。

`nkf -s (filename)` 

SHIFT JIS コードを出力します。


`nkf -w8 (filename)` 

Unicode を出力します。


`nkf` の動作時には入力されるテキストファイルの文字コードは自動判定されます。この判定を明示的に行うオプションも存在します。

8. <sup>ソート</sup> sort (ファイルの中身を並べ替え)


ファイルの中身を並べ替えます。具体的にはそれぞれの行を記号、数字、大文字、小文字の順に並べ替えます。

`sort (filename)` 


いくつかオプションがあります。

`sort -b (filename)` 


行頭の空白を無視します。

`sort -f (filename)` 

大文字と小文字の区別を無視します。

`sort -r (filename)` 


逆順に出力します。

`sort (filename) -o (output filename)` 


並べ替えの結果を (output filename) で指定したファイルに出力します。

9. <sup>グレップ</sup> grep (ファイル内の文字列検索)


指定した文字や単語などが含まれている行を表示します。例えば printf という単語が含まれる行を探す場合には

`grep printf (filename)` 


を実行します。こちらにもいくつかオプションがあります。以下、printf を検索対象の単語として例示します。

`grep -C n printf (filename)` 


printf を含む行だけでなく、前後 *n* 行も表示します。

`grep -c printf (filename)` 


printf を含む行数を表示します。

`grep -i printf (filename)` 

大文字と小文字の区別を無視します。つまり、printf が大文字で含まれていても表示します。

`grep -n printf (filename)` 


printf を含む行数も表示します。

`grep -v printf (filename)` 


printf を含まない行を表示します。

10. <sup>テイル</sup> tail (ファイルの末尾を表示)


ファイルの末尾部分を表示します。通常は

`tail (filename)` 

を実行すると、指定されたファイルの最後の 10 行を表示します。

`tail -n n (filename)` 

指定されたファイルの最後の行数  $n$  を表示します。


`tail -f (filename)` 

指定されたファイルの末尾が追加されるたびに表示されます。


`tail` とは逆にファイルの先頭部分を表示させる <sup>ヘッド</sup>`head` というコマンドもありますが、`more` や `less` でも同様の処理を行うことになりますので、利用頻度は低いでしょう。

#### 11. <sup>コンバート</sup>`convert` (画像ファイルの処理)


画像ファイルのフォーマットを変えたりサイズを変えたりします。

`convert sample1.jpg sample2.png` 


`sample1.jpg` を `sample2.png` に変換します。

`convert sample1.jpg -resize 300x sample2.png` 


変化するとき画像の横幅を 300 ピクセルに縮小します。x300 とすれば縦幅を 300 ピクセルに合わせます。

`convert -size 128x128 xc:#0000ff blue.jpg` 

128 ピクセル四方の青い画像を生成します。

`convert sample1.jpg -rotate +45 sample2.png` 

時計回りに 45 度回転した `sample2.png` を作ります。-flip なら垂直反転、-flip なら水平反転します。


`convert sample1.jpg -colorspace gray sample2.png` 

画像をグレースケールにします。


他にも画像合成、切り出し、ぼかし処理などもできます。

#### 12. <sup>ビーディーエフティーケー</sup>`pdftk` (PDF ファイルの処理)

PDF ファイルをつなげたりページを抜き出したりします。

`pdftk 1.pdf 2.pdf cat output 1-2.pdf` 

`1.pdf` と `2.pdf` を結合して `1-2.pdf` にします。


`pdftk 1-2.pdf cat 2 output 2.pdf` 

`1-2.pdf` から 2 ページ目を抜き出して `2.pdf` に保存します。cat の後が 1-10 なら 1 から 10 ページまで、3 5 なら 3 ページと 5 ページを抜き出します。

## 3.5 ネットワークを扱うコマンド

#### 1. <sup>ホスト</sup>`host` (ネットワーク経由でホスト名を用いた検索)

指定されたホスト名について、ホスト名を入力して IP アドレスなどを表示します。例として

host ankokudan.org 


を実行すると、

ankokudan.org has address xxx.xxx.xxx.xxx  
のような結果が表示されます。

## 2. ダブリューゲット wget, カー curl (ネットワーク経由でのファイルのダウンロード)


Web サーバなどで公開されているファイルのダウンロードを行うときに使います。ダウンロードの際はファイルの URL を指定します。wget と curl で動作がいくらか異なります。単一のファイルをダウンロードする場合は

wget (URL) 

curl -O (URL) 

です。curl にはオプションが必要です。

wget の場合、指定先の URL からリンクが貼られたファイルを一括ダウンロードすることができます。サイトによっては、一括ダウンロードが禁止されていることがありますので、ルールを守ってご利用下さい。

wget -r (URL) 

また、ダウンロード先が複数の場合には、前もってテキストファイルにダウンロード先の一覧を記録して、ダウンロード時にテキストファイルを指定すれば一括ダウンロードできます。テキストファイル名が URL.txt の場合は、以下のようになります。


wget -i URL.txt 

一方、curl の場合には後述の正規表現を利用して、複数のファイルを指定してダウンロードすることができます。

## 3. フーイズ whois (ドメインを運用している団体の表示)

インターネットに接続されているコンピュータには、どのような組織に属しているかを示す ドメイン名 が存在します。ドメイン名は世界中で重複しないように登録されています。whois はこのドメイン名を登録している組織がどのような組織かを示すコマンドです。

例えば暗黒通信団のドメイン ankokudan.org から、登録している組織の情報を知りたい場合は以下のように入力します。

whois ankokudan.org 

ディストリビューションによっては、whois コマンドを実行すると ジェイフーイズ jwhois が実行されることがあります。これは、まず日本国内のドメイン (.jp で終わるもの) を検索してから、海外のものを検索するようになっているわけで、動作結果に大きな差はありません。

### 3.6 時間指定での実行で行うコマンド

1. <sup>エーティー</sup>at (実行時間を指定してコマンドを実行) コマンドを今すぐではなく、決まった時間 (たとえば真夜中) に実行したいときに使います。まず

```
at MMDDYY hh:mm ↵
```

のように記述します。MM, DD, はそれぞれ月, 日, YY は西暦で書いた年の下 2 桁を入れてください。たとえば 2017 年 5 月 30 日の 15:00 にあるコマンドを実行したい場合には,

```
at 053017 15:00 ↵
```

と入力してください。するとターミナルは

```
at &>
```

のような表示に代わるので、ここで実行したいコマンドを入力してください。入力が終わったらエンターキーを押して、最後に **Ctrl** キーを押しながら d キーを押すと,

```
job 2 at 2017-05-30 15:00
```

のようにジョブ番号 (2) と実行日時が表示されます。

コマンドが複雑でいちいち入力ができないという場合には、事前に処理したい項目をファイルに列挙し、at コマンドを実行するときにファイルを入力するという方法もあります。

```
at -f (filename) MMDDYY hh:mm ↵
```

2. <sup>エーティーキュー</sup>atq (未実行の at コマンド一覧を表示)

at で予約されているコマンドの一覧が表示されます。もし管理者が実行すると、すべてのユーザが予約したコマンドの一覧が表示されます。

```
atq ↵
```

3. <sup>エーティーアールエム</sup>atrm (時間指定実行の取り消し)

時間指定しているコマンドの取り消しを行います。

```
atrm (ジョブ番号) ↵
```

4. <sup>クrontab</sup>crontab (定期的に時間指定したコマンドの実行)

コマンドを 1 回限りでなく、毎日同じ時刻に行うなど、定期的に実行するときに使用します。

設定の際には以下のようなコマンドを使います。

```
crontab -e ↵
```

コマンドを実行すると後述の vim が起動し、設定を記述できるようになります。

設定の書式は以下のようになります。

\* \* \* \* \* (command)

ここで \* はそれぞれ分、時、日、月、曜日に対応し、数字で指定します。曜日に  
関しては、0 または 7 が日曜日で、後はカレンダーの右側に行くにつれて数字が  
大きくなり、月曜日が 1、土曜日が 6 になります。コマンドは絶対パスで記述す  
る必要があります。スペースを含む場合は、シングルクォーテーションやダブル  
クォーテーションで囲む必要があります。


以下、いくつか例を示します。

毎時 0 分に実行	0 * * * * (command)
毎日 1:00 に実行	0 1 * * * (command)
毎時 0 分から 10 分ごとに実行	*/10 * * * * (command)
日曜日の 2:20 に実行	20 2 * * 0 (command)
毎日 6:00 と 20:00 に実行	0 6,20 * * * (command)
毎日 9:15 から 18:15 まで 1 時間おきに実行	15 9-18 * * * (command)
毎月 1 日の 0:10 に実行	10 0 1 * * (command)

## 3.7 その他のコマンド

### 1. パスワード passwd (パスワードの変更)

パスワードを変更します。

passwd (username) 

ユーザ名を省略した場合は、自分自身のパスワード変更を行います<sup>\*14</sup>。古いパス  
ワードを 1 回、新しいパスワードを 2 回入力します。入力したパスワードはター  
ミナルに表示されないので誤入力に注意して下さい。最近の Linux では、簡単す  
ぎるパスワードへの変更は拒否されます。ローマ字の大文字、小文字、数字、記  
号を混在させ、かつ辞書に載っている単語をそのまま含まないように、パスワ  
ードは設定すべきです。

### 2. デイト date (現在時刻を表示)

現在の時刻を表示します。通常は

date 


と入力すると、現在の日付および時刻を秒単位まで表示します。オプションとし  
て、日付や時刻を表示させるときの書式を指定することができます。書式に関す  
る説明は省略します。

---


<sup>\*14</sup> 一般的に、ユーザ名を指定してパスワードを変更できるのは管理者だけです。

3. <sup>キャル</sup>cal (カレンダーを表示)


今月のカレンダーを表示します。便利なオプションがいくつかあります。

cal -3 

先月、今月、来月のカレンダーを表示します。

cal -y 

今年のカレンダーを表示します。

cal (year) 


(year) 年 ((year) のところには数字を入れて下さい) の年間カレンダーを表示します\*<sup>15</sup>。

4. <sup>フー</sup>who (ログインしているユーザのユーザ名を表示)

どのユーザ名のユーザが、現在ログインしているかを表示します。

5. <sup>ラスト</sup>last (最近ログインしたユーザのユーザ名を表示)


最近ログインしたユーザのユーザ名を、ログイン時間の一覧を表示します。オプションをつけずに実行すると、ログイン記録が残るファイルに記録されているすべてが表示されます。過去 *n* 回のログイン状況を確認する場合には、

last -n 


を実行します。

6. <sup>ディユー</sup>du (ディレクトリのディスク使用量を表示)

ディレクトリのディスク使用量を表示します。オプションがない場合には、実行したディレクトリおよびそのサブディレクトリのディスク使用量を表示します。単位は 1 キロバイトのことが多いです。オプションがないと、サブディレクトリが多い場合には出力が多くわかりにくいです。また、ファイルサイズもわかりにくいです。そこで以下のようなオプションが有用です。

du -h 

人間が読みやすいよう、K (キロ)、M (メガ) などの単位で表示します。

du -s 

ディレクトリの合計使用量だけを表示します。


7. <sup>ディエフ</sup>df (ファイルシステムのディスク使用量を表示)

ファイルシステムのディスク使用量を表示します。du はディレクトリのディスク

---

\*<sup>15</sup> このオプションでは数字を変えることにより、西暦 1~9999 年の年間カレンダーを表示しますが、現在使用されているグレゴリオ暦が導入された時期をまたぐので、現在から大きく離れた年のカレンダーは不正確になります。詳細は man などでご確認下さい。

使用量を表示しますが、df はシステムのディレクトリやホーム全体のディレクトリなどの全体のサイズ、使用量、残りの量、使用率などを表示します。df のオプションも du のオプションに類似しています。

df -h 

人間が読みやすいよう、M（メガ）、G（ギガ）などの単位で表示します。このオプションをつけて実行すると、例として以下のようになります。

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/mapper/VolGroup- lv_root					
	18G	7.0G	9.3G	43%	/
tmpfs	504M	372K	503M	1%	/dev/shm
/dev/sda1	477M	32M	420M	8%	/boot


df を実行して、使用率が高い（Use% が 100 パーセントに近い）場合には、不要なファイルを削除して下さい。100 パーセントになるとファイルの書き込みなどが出来なくなります。特にシステムが含まれているディレクトリ（/）の使用率が高い場合には、管理者は速やかに不要なファイルを削除して下さい。最悪の場合、Linux が動作を停止します。

## 4 vi でテキストファイルを編集する

ヴァイ  
vi はテキストエディタとして Linux で長年使われてきました。Linux には他にも イーマックス emacs、ジーエディット gedit など様々な使いやすいテキストエディタがありますが、vi はたいていのシステムに最初からインストールされていますので、最初に紹介します。最近は vi の機能 ヴァイアイエム を大幅に拡張した vim（Vi IMproved）が使われることが多いです。

### 4.1 vi の起動









テキストエディタ vi を起動します。vim がインストールされているシステムでも、vi と入力すれば vim が起動されるようになっています。以下、vim も含めて vi と記すことにします。

vi (filename) 

で起動します。指定したファイル名のファイルがある場合には編集、無い場合には新規作成になります。



表 1 vi での主なキー操作

操作	キー入力
カーソルを 1 字左へ移動	h
カーソルを 1 字右へ移動	l
カーソルを 1 字上へ移動	j
カーソルを 1 字下へ移動	k
カーソルを行頭へ移動	^
カーソルを行末へ移動	\$
カーソルを後方へ 1 画面分移動	<b>Ctrl</b> -b
カーソルを前方へ 1 画面分移動	<b>Ctrl</b> -f
直前の操作を取り消す	u
直前の操作を繰り返す	.
操作の中止	<b>ESC</b>
カーソルの位置からテキスト入力	i
カーソルの位置の右からテキスト入力	a
カーソルの上の行にテキスト挿入	O
カーソルの下の行にテキスト挿入	o
カーソルの位置の文字削除	x
カーソルの位置の左の文字削除	X
行末の改行を削除	J
文字列を下へ検索	/文字列 
文字列を下へ再検索	N
文字列を上へ検索	?文字列
文字列を上へ再検索	n
文字列を全置換	:%s/置換前文字列/文字列/g 
文字列を確認しながら置換	:%s/置換前文字列/文字列/gc 
カーソルのある行をカット	dd
カーソルのある場所から範囲指定しカット	v の後でカーソルを移動し d
カーソルのある行をコピー	yy
カーソルのある場所から範囲指定しコピー	v の後でカーソルを移動し y
カーソルの前方にペースト	P
カーソルの後方にペースト	p
ファイルを上書き保存	:w 
ファイルを上書き強制保存	:w! 
ファイルを保存して vi を終了	:wq  または ZZ
vi を終了	:q 
vi を強制終了	:q! 

## 4.2 vi のモード切替

vi には テキスト入力モード と コマンドモード があります。前者は文字の入力のみを行い、後者は、ファイルの操作、ファイルの保存などを行います。vi の起動時にはコマンドモードになっています。テキスト入力モードへ切り替えるには「i」キーを押して下さい。テキスト入力モードからコマンドモードへの切り替えは **ESC** キーを押して下さい。**ESC** キーは、コマンドモードで複数のキー操作による操作の途中で、入力中のコマンド実行をリセットできる便利なキーです。

テキスト入力モードの状態の場合、ウインドウの左下に「- 挿入 -」というようなメッセージが表示されます。

## 4.3 vi のコマンドモードでのキー操作

テキストモードは基本的に文字を入力するだけなので、キー操作は複雑ではありません。システムによっては日本語の入力ができない場合があるのでご注意ください。

コマンドモードのキー操作は独特です。主な操作を表 1 にまとめました。

# 5 emacs でテキストファイルを編集する

イーマックス

emacs も vi と同様に、テキストエディタとして長年使われてきました。emacs は vi とは異なり最初からインストールされていないシステムもありますが、利用する機会が多いと思われるので紹介します。

## 5.1 emacs の起動

`emacs (filename)`

です。指定したファイル名のファイルがある場合には編集、無い場合には新規作成になります。

上記のように起動すると、新しいウインドウが開いてファイルが開きます。画面が狭く新しいウインドウを開くことが困難で、自分が操作している端末でウインドウでファイルを開きたい場合には、

`emacs -nw (filename)`

とオプションをつけることで、端末内にウインドウが開きます。

## 5.2 emacs でのキー操作

emacs は vi とは異なり、文字入力の際のモード切替はありません。操作は、**Ctrl** キーを押しながら別のキーを押したり、**ESC** キーを押してから別のキーを押したりするものが多いです。emacs でファイルを上書き保存すると、以前に保存していたファイルは、ファイル名の末尾にチルダ ( ~ ) がついて保存されます。編集したファイルを保存せずに emacs を終了すると、作業中のファイルはファイル名の前後を # で挟まれたファイル名をつけられて保存されます。後にこのファイルを編集の際から開いて編集を再開するときは、

emacs (filename)

で起動後に

**ESC** x recover-this-file

を実行してください。

emacs の画面で操作を行う場合、**Ctrl** キーと同時に押したキーが例えば x ならば C-x、**ESC** の後に押したキーが例えば x ならば M-x というように入力経過が、ウィンドウの一番下の行 (エコーエリアと呼ばれます) に表示されます。

最初の方にも書きましたが、表 2 で、**Ctrl** キーの後に “-” が記述されているものは、**Ctrl** キーを押しながら操作することを意味します。

emacs にはこの他にも **ESC** キーを押した後に x キーを押し、様々なコマンドを入力すると様々な機能を利用することができます。さらに、Lisp と呼ばれる言語を使って自分の手で機能を拡張することもできます。

emacs の機能の詳細は、emacs を起動後に

**ESC** x help-with-tutorial

を実行すると、非常に詳しい入門ガイドが表示されますので、そちらをご覧ください。ページのスクロールは上記の表をご覧ください。さらに詳細な説明は、emacs を起動後に







**Ctrl**-h r

を入力すれば膨大なマニュアルが表示されます。このマニュアルの操作は、info コマンドで表示されるマニュアルに対するものと同じです。ただし、英語のマニュアルなのであまりお勧めではありません。インターネット上で操作方法を検索した方がいいでしょう<sup>\*16</sup>。

---

<sup>\*16</sup> 上記の膨大なマニュアルが日本語訳されたサイトが <https://ayatakesi.github.io/> にあります。

表 2 emacs での主なキー操作


操作	キー入力
カーソルを 1 字左へ移動	<b>Ctrl</b> -b
カーソルを 1 字右へ移動	<b>Ctrl</b> -f
カーソルを前の行へ移動	<b>Ctrl</b> -p
カーソルを次の行へ移動	<b>Ctrl</b> -n
カーソルを指定した行へ移動	<b>ESC</b> gg 行番号 
カーソルを行頭に移動	<b>Ctrl</b> -a
カーソルを行末に移動	<b>Ctrl</b> -e
カーソルを後方へ 1 画面分移動	<b>Ctrl</b> -v
カーソルを前方へ 1 画面分移動	<b>ESC</b> v
直前の操作を取り消す	<b>Ctrl</b> -.
操作の中止	<b>Ctrl</b> -g
カーソルの位置の文字削除	<b>Ctrl</b> -d
カーソルの位置の左の文字削除	<b>DEL</b>
文字列を下へ検索	<b>Ctrl</b> -s 文字列 
文字列を下へ再検索	<b>Ctrl</b> -s
文字列を上へ検索	<b>Ctrl</b> -r 文字列 
文字列を上へ再検索	<b>Ctrl</b> -r
検索を終了	
文字列を確認しながら置換	<b>ESC</b> % 置換前文字列  文字列  以後, y/n で対話的に置換
カーソルのある場所から行末まで削除	<b>Ctrl</b> -k
マークを設定	<b>Ctrl</b> -@
マークからカーソルの位置までをカット	<b>ESC</b> w
マークからカーソルの位置までをコピー	<b>Ctrl</b> -w
ペースト	<b>Ctrl</b> -y
編集中のファイルをリカバリ	<b>ESC</b> x recover-this-file
ファイルを挿入	<b>Ctrl</b> -x i ファイル名
ファイルを別名で保存	<b>Ctrl</b> -x <b>Ctrl</b> -w
ファイルを上書き保存	<b>Ctrl</b> -x <b>Ctrl</b> -s
emacs を終了	<b>Ctrl</b> -x <b>Ctrl</b> -c

## 6 応用編


コマンドを実行した後で、画面に表示された結果をさらに活用したいということはありませんか。いちいちファイルに出力するのは面倒です。また、出力結果が多すぎて、一気に表示されて何が何だかわからないということがありませんか。このような問題を解決できる応用方法を紹介します。

### 6.1 複数のコマンドをつなぐ

まず、コマンドを複数つなぎ合わせて実行する方法があります。一般にパイプと呼ばれる方法です。使い方は簡単です。最初に実行するコマンドの後に“|”（日本語キーボードだったら、¥マークのところにあります）を入れて、次のコマンドをその後に書きます。例えば、大量にファイルが存在するディレクトリで `ls -R` を実行すると、結果がわからないうちにターミナルがスクロールしてしまいます。これを防ぐために、`more`、`less` などのコマンドを使って、部分的に結果を表示しながら `ls -R` を実行することができます。


```
ls -lR | more 
```

他にも、たとえば実行中のプロセスのうち、`hoge` という名前を含むプログラムを実行しているプロセスだけを表示したければ、

```
ps -aux | grep hoge 
```

と実行すれば、該当するプロセスだけが表示されます。

複数のコマンドをつなぐ際に有用なコマンドとして、<sup>ユニーク</sup>`uniq` があります。

```
sort (filename) | uniq 
```

上記のコマンドを実行すると、指定されたファイルを並べ替えた後に重複した行を削除して出力します。`uniq` コマンド単独では、たとえ1つのファイルに重複した行があっても、連続した行でなければ削除してくれません。`sort` コマンドとつなぎ合わせて威力を発揮します。`uniq` コマンドの有用なオプションもいくつか挙げておきます。

```
uniq -c
```

それぞれの行が何回現れたかを出力します。

```
uniq -i
```

比較の際にローマ字の大文字と小文字の違いを無視します。

```
uniq -d
```

重複した行だけを出力します。

```
uniq -u
```

一度しか現れない行だけを出力します。


## 6.2 正規表現

最初の方でワイルドカードについて説明しました。任意の文字列を含むファイル名を指定できます。この他にも、複数のファイルなどをまとめて取り扱いたい時や、`grep`であいまいな検索をしたい時に便利な方法として、正規表現という記述方法があります。正規表現には様々な記法がありますが、主なものを挙げておきます。


- `^` 文字列：先頭部分が指定した文字列に一致するものを意味します。
- 文字列 `$`：終わりの部分が指定した文字列に一致するものを意味します。
- `.`：任意の 1 文字を表します。たとえば `'cou.t'` と表したら `count`, `court` のいずれも表します。
- `[文字]`：`[ ]` 内の文字のいずれか 1 字を表します。たとえば `'file[123]'` と表したら `file1`, `file2`, `file3` を表します。`file12` や `file123` は含まれないのでご注意ください。
- `[文字1-文字2]`：文字 1 と文字 2 の間にある 1 字を表します。よく使われるのは、数字の 1 字を表す `[0-9]` や、ローマ字の大文字、小文字を表す `[A-Z]`, `[a-z]`, あるいはいずれかのローマ字を表す `[A-Za-z]` です。

## 6.3 ファイルに出力する

コマンドの出力結果はターミナルに表示されますが、ファイルに保存された方が後で処理するのに都合が良いことがあります。たとえば `grep` コマンドで単語を含むファイルを探したとき、あまりに多く該当すると画面表示の場合には次の操作がしにくいです。このような場合には、

```
grep printf (filename) > printf.txt 
```

というように不等号をつければ、`grep` の結果が `printf.txt` に保存されます。不等号 1 個の場合は、既に存在するファイル名を指定して出力先にすると、上書き保存されます。既に存在するファイルの末尾に追記したい場合は不等号 2 個を使います。

```
grep printf (filename) >> printf.txt 
```


## 6.4 ファイルを更新する

ファイルに出力する方法を使って、さらにファイルを更新する方法を紹介します。それは <sup>パッチ</sup>`patch` というコマンドです。`diff` によって抽出された 2 つのファイルの違い（差分データ）をもとに、ファイルを変更します。このことを「パッチを当てる」ともいいます。




例えば、タイポ修正等のファイルのわずかな修正を伝えたい場合を考えてみましょう。修正したファイル全体を送らなくても差分データだけあれば十分なはずですし、修正箇所は差分データを見ればほぼ一目瞭然です。差分を受け取った人は `patch` を使うことで、簡

単にファイルを修正できます。

差分データ (diff-filename) を利用してファイル (filename) を修正するには、

`patch (filename) < (diff-filename)` 

とします。なお、patch は差分データと関係ない部分を無視してくれます。

ここで、差分データが `diff -u` や `diff -c` を使って作成されているとき、その差分データの最初のファイル名と変更したいファイル名が同じならば (filename) を省略して `patch < (diff-filename)`  とするだけで十分です。もっと横着するには、`patch`  としたあと、ターミナルに差分データをペーストし、必要に応じて  を押してカーソルを行頭位置に持っていき、`[Ctrl]-d` で patch への標準入力を終了するという方法もあります。


## 6.5 過去に実行したコマンドの再利用

上記のようにコマンドをつなぎ合わせたりファイルに出力したりしていくと、一度に実行するコマンドが長くなります。何度も打ち直すのが面倒という場合には、過去のコマンドを再実行するという手があります。↑を1回押すと、1回前に実行したコマンドを呼び出します。同じコマンドを実行するならば、そのままエンターキーを押して実行すればいいですし、少し違うコマンドを実行するならば書き換えていけばいいでしょう。ターミナルでのキー操作は以下の通りです。


- `[Ctrl]-a` : コマンドの先頭にカーソルが移動します。
- `[Ctrl]-e` : コマンドの最後にカーソルが移動します。
- `[Ctrl]-b` : カーソルを1文字前に移動します。
- `[Ctrl]-f` : カーソルを1文字後に移動します。
- `[Ctrl]-d` : カーソルが重なっている文字を消去します。

過去に実行したコマンドの記憶数はディストリビューションにより異なりますが、通常は1000回に設定されていることが多いようです。

過去に実行したコマンドの一覧を表示するには <sup>ヒストリー</sup> history コマンドを使います。

history 


このままだと1000回分のコマンドが表示されることになりますので、複数のコマンドをつなぐ方法で more などと組み合わせる他、以下のオプションで表示件数を変更できます。

history *n* 

ここで実行する history コマンドも含めて、最近実行した *n* 件のコマンドを表示します。

## 6.6 バックグラウンドで処理を進める


今までのコマンド実行では、終了するまでターミナルは次のコマンドを受け付けません。コマンドの実行に時間がかかるような場合には、時間のかかる処理をバックグラウンドで進めて、他に進めても差しつかえのないコマンドを実行したほうが、全体の作業時間が短くなります。バックグラウンドでコマンドを実行する場合には

`(command) &` 

のように、コマンドの後に&を付け加えてください。すると、たとえば

`[1] 4218`

のように実行番号と PID が表示されます。バックグラウンドで実行されているコマンドは

`jobs` 


で確認できます。バックグラウンドでの処理が終了すると、

`[1]+ 終了 2 (command)`

のようなメッセージが表示されます。

## 6.7 ログアウトしても処理を進める

コマンドなどの実行にとても長い時間がかかる場合、会社や学校の Linux を使っていたらログアウトして帰らなければならないかもしれませんが、このようにログアウトした場合でも、コマンドなどが実行できるようにするコマンドが ノーハングアップ **nohup** です。以下の様に実行します。

`nohup (command) &` 

これでログアウトしてもコマンドは実行されます。通常はターミナルに表示される出力は、nohup.out というファイルに出力されます。

Linux がシャットダウンされたり再起動されたりした場合には、処理は停止します。

## 7 あとがき

本書では Linux で必要になりそうなコマンドをなるべく少なめにし、ちょっとしたオプションを挙げています。「こういうことをやりたいけれど、いいオプションはないの?」ですとか、「応用編のさらに先を知りたい」という方も出てくることでしょう。そのような方には、Linux の様々な本が出版されていますので、次はそちらをお読みいただければと思います。

本書を読んだ後でさらに Linux について学びたい方に、お勧めの文献をいくつか挙げておきます。



1. 6 日間で楽しく学ぶ Linux コマンドライン入門 (大津真, インプレス R&D, 2015 年)  
6 冊構成になっています。1 冊が 4 セクションで各セクションの学習時間は 1 時間です。Kindle 版もあるので、ダウンロードして持ち歩くのにもお勧めです。
2. Linux コマンドブック ビギナーズ 第 4 版 (川口拓之, 田谷文彦, 三澤明, SB クリエイティブ, 2015 年)  
本書より一歩深くコマンドを使いこなしたい方にお勧めの本です。Kindle 版もあります。
3. 改訂第 3 版 Linux コマンドポケットリファレンス (杓名亮典, 技術評論社, 2015 年)  
「ポケット」という割には約 600 ページの、辞書のような本です。最近では Kindle 版も出版されているので、タブレット端末などにダウンロードするのもいいでしょう。
4. まんがでわかる Linux システム系女子 (結城洋志, 日経 BP Next ICT 選書, 2015 年)  
Linux の管理者を目指す人が読むとわかりやすいマンガです。1 巻がコマンド & シェルスクリプトの基礎, 2 巻がその応用になっています。応用まで理解できれば複雑なコマンドを繰り返し実行することができるでしょう。
5. 小悪魔女子大生のサーバエンジニア日記 (aico, 技術評論社, 2011 年)  
文学専攻の女子大生がアルバイトで、インターネットを支えるサーバや通信技術を学んだ日記をまとめた本です。文字が手書きだったり、少々クセがあったりなので、読者の向き不向きがあるかと思います。
6. LPIC  
<http://www.lpi.or.jp/>  
世界共通の「Linux 技術者認定制度」を実施している団体です。アンケートに答えることで Linux 標準教科書など、様々な教科書を無料でダウンロードできるのでおすすめです。じっくり学べば、世界で活躍する Linux エンジニアになるのも夢ではありません。

リナックス にゅうもん にゅうもん  
**Linux 入門の入門**

2017 年 8 月 25 日 初版発行

著 者 茗荷 さくら (みょうが さくら)

発行者 星野 香奈 (ほしの かな)

発行所 同人集合 暗黒通信団 (<http://ankokudan.org/d/>)

〒277-8691 千葉県柏局私書箱 54 号 D 係

本 体 300 円 / ISBN 978-4-87310-094-4 (C0004)



sort が正常実行されていない本は在庫がある限りお取り替え致します。

©Copyright 2017 暗黒通信団

Printed in Japan