ESSENTIAL COMMANDS

Fedora Linux 対応

LINUX ハンドブック

機能引き コマンドガイド

DANIEL J. BARRETT 著
猪俣 敦夫、岡本 健、田淵 貴昭 訳

O'REILLY®
オライリー・ジャパン

tools@manhouse.org

Linuxハンドブック
機能引きコマンドガイド

Daniel J. Barrett 著

猪俣 敦夫
岡本 健 訳
田淵 貴昭

O'REILLY®
オライリー・ジャパン

This PDF is prepared for tools@maruhouse.org, personal use only.

本書で使用するシステム名、製品名は、それぞれ各社の商標、または登録商標です。
なお、本文中では™、®、©マークは省略しています。

This PDF is prepared for tools@maruhouse.org, personal use only.

Linux

Pocket Guide

Daniel J. Barrett

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

This PDF is prepared for tools@maruhouse.org, personal use only.

©2005 O'Reilly Japan, Inc. Authorized translation of the English edition ©2004 O'Reilly Media, Inc. This translation is published and sold by permission of O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

本書は、株式会社オライリー・ジャパンがO'Reilly Media, Inc.の許諾に基づき翻訳したものです。日本語版についての権利は、株式会社オライリー・ジャパンが保有します。

日本語版の内容について、株式会社オライリー・ジャパンは最大限の努力をもって正確を期していますが、本書の内容に基づく運用結果について責任を負いかねますので、ご了承ください。

This PDF is prepared for tools@maruhouse.org, personal use only.

訳者まえがき

本書は、D. J. Barrett 氏によって、米国 O'Reilly 社から出版された『Linux Pocket Guide』の翻訳書である。Linuxについては、これまでにも多くの解説書が出版されているが、Linuxが持つ膨大な機能を全て説明しようとするあまり、焦点がぼやけてしまったものや、ある機能に限定して解説しているものの、説明不足と感じられる書籍もあったように思われる。

これに対して、原著者はLinuxの「コマンド」に焦点をあて、これをいかにうまく扱うかに全力を注いだ。といっても、単にコマンドを羅列するのではなく、原著者の実体験によって得られたノウハウが惜しみなく紹介されている。このため、読者は逆引きによりコマンドを調べるといった使い方の他に、読み物として最初から読み進めるといった楽しみ方もできる。また、Linuxの利用方法について、非常にバランスよく解説されているので、本書を一読することにより、Linuxのコマンド操作に関する主要な技術を一通りマスターできるだろう。

本書を執筆するにあたって我々は、まず最初にLinuxのポケットガイドとして、「どのような解説書であるべきか」について十分な議論をした。原著の英文を一字一句正確に翻訳するのも、もちろん大切であるが、それよりもまず「読者が何を望んでいるか」を理解し、それをいかに反映させるかを最優先して考えるべきであろう。このために、原著において不明瞭に思われる箇所については、文章を加筆・修正するなどして、大幅に変更を加えた。章立てについても、再構成を行い、見通しよく配置している。また原著には記載されていないが、あると便利なコマンドについても追加した。これらの作業によって、本書は原著の日本語版というよりは、第2版に近い構成になっている。本文中の表現については、十分吟味したつもりであるが、わかりにくい箇所などあれば、それは我々翻訳者の責任である。

最後に、オライリー・ジャパンの担当編集者の高恵子さんに感謝したい。本書の編集や構成について、我々は高さんと何度も相談した。また本書の執筆を応援していただいた、同僚、家族、友人など、全ての支援者にも感謝したい。読者の皆様が、本書を通じて「Linux ワールド」という素晴らしい世界を体験できることを、我々は願って止まない。

2005 年 8 月

翻訳者一同

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

まえがき

Linux ワールドにようこそ。本書は Linux を初めて扱うユーザ(特に Fedora Linux のユーザ)に、すぐに役立つ知識、常識とされているコマンドを簡単に理解できるよう、構成している。あなたがLinuxのエキスパートであれば、まえがきを飛ばして読み進めてもかまわない。

本書の活用法

本書は、幅広い知識を与えるリファレンス本ではなく、いつでもすぐに利用できるガイドブックとして執筆したつもりである。もちろん、Linux についての役立つ知識や、重要なことは全てカバーしており、本書を実践的に活用していただけるはずである。しかし、全てのコマンドについて取り上げるわけではなく、また、各オプション全てを網羅しているわけではないことをあらかじめ述べておきたい(もし、お気に入りのコマンドが掲載されていなければ、これは我々の力不足である)。これ以外にもオペレーティングシステム内部に関する情報についても取り上げていないこともあらかじめ了承していただきたい。あくまでも、本書のモットーは、使いやすく、わかりやすく、そして簡潔に、である。

本書は、いわゆるコマンド(command)に焦点を当てて執筆されている。コマンドとは、あなたが実行したいこと(例えば、ファイルのリスト表示を行うコマンド`ls`、ファイル中の指定したテキストを検索する`grep`、オーディオファイル再生を行う`xmms`、ディスクの空き領域を調べる `df` など)は、コマンドラインを入力することによって Linux システムにその意図を伝える。我々にとって多少厄介な言葉のようなものである。また、本書では、グラフィカルなウィンドウシステム環境である X 上の GNOME や KDE のようなデスクトップ環境についても少しだけ取り上げる。「少しだけ」とした理由は、GNOMEやKDEの場合、それだけでPocket Guide が 1 冊作れてしまうほど複雑なシステムだからである。

ここで、本書の構成を述べておこう。本書では、読者の方がスムーズにコマンドを理解できるよう、順を追って解説を行う。例えば、ファイルの中身を参照するための方法として、まずファイル表示を行うコマンドを5つ紹介する。具体的には、ファイルサイズがあまり大き

This PDF is prepared for tools@maruhouse.org, personal use only.

くないテキストファイルを表示する場合に利用するcatコマンド、それよりも大きなファイルサイズを持つテキストファイルを表示するためのlessコマンド、バイナリファイルの表示を行うodコマンド、Postscriptファイル表示を行うghostviewである。これら全てのコマンドを取り上げた後に、各コマンドを解説するという流れをとる。この解説では、それらのコマンドの使用方法やオプションなどを簡潔に説明する。

本書では、Linuxシステム上にアカウントを持っていることを前提として話を進める。また、ユーザ名とパスワードを用いてログインする知識を有している、ということも前提とする。もしアカウントを持っていなければ、システム管理者と相談して、是非アカウントを作成してもらいたい。あなた自身がシステム管理者であるならば、Linuxのインストール後に、自分で作成したアカウントを使用してほしい。

Linuxとは

Linuxは、オペレーティングシステムという意味ではMicrosoft WindowsやApple Macintoshと同じであるが、この2つのオペレーティングシステムと大きく異なる点は、Linuxがオープンソースのソフトウェアである、ということである。Linuxは、おおまかに言って以下に示す4つの部分に分けられる。

カーネル(kernel)
ファイル、ディスク、ネットワーク、など基本的でローレベルな処理を全ての司るオペレーティングシステムの心臓部である。

プログラム(program)
ファイル操作、テキスト編集、数式処理、文章処理、オーディオ、ビデオ、コンピュータプログラミング、Webサイト作成、暗号化、CD書き込みなど、その実行目的に合わせて作られた様々なプログラムのことである。

シェル(shell)
コマンド入力、コマンド実行、コマンド実行による結果の表示に必要なユーザインタフェースのことである。なお、シェルは何種類か存在しており、有名なものとしては、bourne shell、korn shell、C shellなどがある。本書では、bash(bourne again shell)を対象としている。bashは、ユーザアカウントのデフォルトシェルである。bash以外のシェルについても、基本的な機能はいずれも同じである。

This PDF is prepared for tools@maruhouse.org, personal use only.

X Window System

ウィンドウ、メニュー、アイコン、マウスサポートなど、他のGUI機能群が提供されるグラフィカルユーザインタフェースシステムのことである。X Window System により、複雑なグラフィカル環境がX上で構築可能である。特に、その中でも有名なものが、GNOMEやKDEである。本書では、X Windows上において動作可能なプログラムを取り上げ、それについての解説を行う。

Fedora Linux

Fedora Linuxとは、Red Hat, IncとFedoraプロジェクト(詳細は、http://fedora.redhat.com/を参照のこと)によって開発されたLinuxディストリビューションであり、かつてはRed Hat Linuxと呼ばれていた。本書は、Fedora Core 3 (2004年11月8日 (EST) リリース)をベースとして執筆しており、提供されているプログラムとシェルに焦点を当て、X Window System やカーネルについて必要な情報を提供する。

コマンド

Linuxでは、シェルに入力するコマンドは、プログラムの名前と、その後に続くオプションや引き数から構成される。プログラムの名前は、ディスク上に存在するプログラムのファイル名である(シェルの役割は、そのプログラムの存在場所を検索して実行することである)。さらに、ダッシュ(-)に続いて指定するオプションで、そのプログラムの振る舞いを指定する。また、引き数には、入力と出力を指定することが多い。例えば、あるファイル中に記載されたテキストの行数をカウントするコマンドを取り上げてみよう。

```
$ wc -l myfile
```

このコマンドは、プログラム(wc、word count:文字数カウントプログラム)であり、さらにその行数をカウントするオプション(-l)と読み出しの対象とするファイルを指定する引き数(myfile)から構成されているのが分かるだろう。$マークは、シェルプロンプトを示し、これはシェルが入力を待機している状態であることを示す。なお、複数のオプションを指定する場合には、それらを個々に指定する方法がある。

```
$ myprogram -a -b -c myfile       3つの独立したオプション
```

あるいは、オプションを組み合わせて指定する方法もある。

```
$ myprogram -abc myfile           -a -b -cと同じ意味
```

This PDF is prepared for tools@maruhouse.org, personal use only.

しかし、これは全てのプログラムで統一されているわけではないため、このような組み合わせのオプションを認識しないプログラムも存在する。またコマンドは、単一のプログラムとして実行するだけでなく、複数のコマンドを組み合わせて実行することもできる。

- これは、同時に複数のプログラムを実行できることを意味する。すなわち、一度に実行させる、順番に実行させる、あるいは、コマンドの出力を以下に続けられるコマンドの入力として渡す「パイプ(|)」で繋げることも可能である。
- オプションは、何らかのルールが決められているわけではない。同一のオプション(例えば、-l オプション)でも、当然のことながら別のプログラムでは別の意味を示す。すなわち、wc -lコマンドの-lオプションは、「テキストの行数をカウントする」ことを意味するが、ls -l コマンドの-l オプションは、「ファイルリストを詳細に出力する」ことを意味する。当然その反対もありうる。すなわち2つのプログラムにおいて、違う2つのオプションが同じ意味を持つことがある。これは、“run quietly”を示す-qオプションと、“run silently”を示す-sオプションなどがある。このオプションは、「メッセージを非表示にする」といった意味である。
- 同様に、引き数にも何らかの規定が決められているわけではない。引き数は、一般的には入力と出力のファイル名が指定されることが多いが、それ以外の使用例として、ディレクトリ名や正規表現が指定されることもある。
- Linuxのコマンドラインユーザインタフェースであるシェル(Shell)は、シェル独自のプログラム言語を提供している。このプログラム言語とは、日常に使う言葉で、「このプログラムを実行しなさい」と伝える代わりに、「もし今日が火曜日ならば、このプログラムを実行しなさい、今日が火曜日でなければ、ファイル名が.txtで終わる全てのファイルに対して、別のもう一方のコマンドを6回実行しなさい」といった命令を伝えることができるようになる。

ユーザとスーパーユーザ

Linux は、マルチユーザ対応のオペレーティングシステムであり、各ユーザには、固有のユーザ名(username)が与えられる。例えば、smithやfunkyguyなどのように各々識別されるユーザ名を作成することができ、そのシステム上に適当なユーザ個別のプライベートな作業領域が与えられる。さらにユーザ名がrootとなる特別なユーザアカウントが存在する。このrootユーザは、システム上のあらゆる権限を実行する特権を持つ。なお、通常の一般ユーザ

This PDF is prepared for tools@maruhouse.org, personal use only.

は、権限が制限されているが、もちろん、一般ユーザもほとんどのプログラムを実行することは可能である。また、ユーザが自分で所有するファイルの書き換えも可能である。一方、スーパーユーザは、システム上の全てのファイルの作成、修正、削除、また全てのプログラムの実行、停止の権限を持つ。

本書で紹介するコマンドの中には、スーパーユーザでのみ実行可能なものがある。この場合には、シェルプロンプトとして(#)を使用する。

```
# command を実行する
```

以下では一般ユーザでのコマンド実行を示す場合にはプロンプトとして$を使用する。

```
$ command を実行する
```

一般ユーザからスーパーユーザに変更するには、一旦ログアウトして再度rootユーザでログインする、という手順を踏む必要はない。suコマンドを実行し(詳細は、「6.3　スーパユーザへの変更」を参照)、スーパーユーザのパスワードを入力すればよい。

```
$ su -l
Password: ********
#
```

本書の活用法

本書では、コマンドを取り上げる際に、最初に一般的な使用方法について解説する。例えば、wc(文字数カウント)プログラムの一般的な使用方法は以下の通りである。

```
wc [オプション] [ファイル名]
```

これは、コマンド名wcに続いて、選択するオプションとファイル名を入力する、ということを示す。当然ながら、かっこ[]を入力する必要はない。かっこは、その内容がオプションであることを示す。さらに、斜体で記載された英字は、任意の文字、たとえば、実際のファイル名などを指定する必要があることを示す。オプションと引数の指定で使われる縦棒の線 | は、いずれか選択することを示す。

```
ls (ファイル | ディレクトリ)
```

この例では、lsコマンドの引き数としてファイルとディレクトリの指定が可能であることを示す。

This PDF is prepared for tools@maruhouse.org, personal use only.

入出力

Linuxのプログラムは、データを標準入力(通常はキーボード)から受け付け、出力結果を標準出力(通常は画面)に出力する。さらに、エラーメッセージは、標準エラー出力(大抵は、画面に出力されるが、標準出力とは別に扱われる)に表示される。標準入力、標準出力、標準エラーへのリダイレクト、ファイルやパイプからのリダイレクトについては後述するが、まず用語について整理しておきたい。本書ではコマンド読み込みと示した箇所は、他の意味であると特に明示しない限り、標準入力からの入力を意味することとする。さらに、コマンド出力と示した箇所は、プリンターについて述べない限り、標準出力からの出力を意味する。

本書での記法

個々のコマンドについて、以下のように解説する。以下の例は、ls(ファイルのリスト表示)コマンドについて示したものである。

ls [オプション] [ファイル] coreutils

/bin stdin stdout - file -- opt --help --version

解説の冒頭には、まずコマンド名(ls)とその使用法を提示する。続いて、そのコマンドが格納されているディレクトリ(/bin)、そのコマンドが含まれるRPMパッケージ(coreutils)、そのコマンドの特性について、サポートされているならば黒色、サポートされていなければ灰色の文字で記載する。

stdin

このコマンドは標準入力、すなわちデフォルトでキーボードから入力する。

stdout

このコマンドは標準出力、すなわちデフォルトで画面上に出力する。

- file

引き数として入力するファイル名の代わりにダッシュを指定すると、コマンドは標準入力から読み込む。同様に、出力するファイル名の代わりにダッシュを指定すると、コマンドは標準出力に書き出す。以下の例では、wc(文字数カウント)コマンドは、file1とfile2からファイルを読み込み、文字数をカウントする。次に、標準入力で入力した文字をカウントする(^Dで終了)。最後にfile3を読み込みカウントする。

```
$ wc file1 file2 - file3
```

This PDF is prepared for tools@maruhouse.org, personal use only.

-- opt

コマンドラインのオプションとして-- を指定すると、これは「オプションの終わり」を示す。すなわち、これ以降の文字がオプションではないことを意味する。これは、ダッシュで開始されたファイル名を処理する場合など、それがオプションとして(誤って)処理されないために必要である。ここでは、ファイル名として-fooという例を取り上げてみる。この場合、wc -foo はエラーとなる。この理由は、-foo が(不正な)オプションとして認識されてしまうためである。すなわち、wc -- -fooと指定しないと正しく動作しない。また、コマンドによって–をサポートしていない場合には、カレントディレクトリのパスを示す./を記述することで解決できる。これにより、ダッシュ(-)を指定する必要がなくなる。

```
$ wc ./-foo
```

--help

オプション --help により、詳細な使用方法が記載されたヘルプメッセージを出力する。

--version

オプション --version により、コマンドのバージョン番号を出力する。

本書で取り扱う記号

本書では、キー入力を示すために特定の記号を使用する。他のLinuxディストリビューションのドキュメントと同様、本書においても^記号は、「コントロール(Ctrl)キーを押したまま」を意味する。例えば、^D(Control D) は、「コントロール(Ctrl) キーを押したままDを押す」ことを意味する。さらに、ESC は「エスケープ(Esc) キーを押す」、Enter、スペースについても同様である。

優れた echo コマンド

本書で取り上げる例は、通常、echoコマンドを用いて画面に情報を出力する、という方法を用いる。echo コマンドについての詳細は、「9.1　画面出力」にて取り上げるが、最も簡単なコマンドの1つであるので、是非この場で覚えてもらいたい。echo は、コマンドの後ろに指定した引き数をそのまま標準出力として出力するだけのコマンドである。引き数が複数指定されている場合には、1 度処理される。

```
$ echo My dog has fleas
My dog has fleas
$ echo My name shell is $USER USER はシェル変数
My name is smith
```

This PDF is prepared for tools@maruhouse.org, personal use only.

ヘルプ

本書で説明する内容より、さらに詳しい情報を得るための方法を示す。

man コマンド

man は、オンライン上でマニュアルページを表示するコマンドである。例えば、ls を利用してリスト表示されたファイル上のドキュメントを読むためには、以下のように実行する。

```
$ man ls
```

ある特定のトピックを得る方法として、マニュアルページに記載された内容に対してあるキーワード検索を実行するには、-k オプションの後に、そのキーワードを指定する。

```
$ man -k database
```

info コマンド

infoは、多くのLinuxプログラムをカバーしているハイパーテキストのヘルプシステムである。

```
$ info ls
```

リクエストしたプログラムのドキュメントが存在しない場合には、 infoコマンドは、指定したプログラムのマニュアルページを表示する。利用可能なドキュメントを表示するには、infoコマンドを単独で実行する。infoシステムの操作方法について知りたい場合には、info info と入力する。

--help オプション(利用可能な場合のみ)

多くのLinuxコマンドでは、ヘルプメッセージの表示方法として--helpオプションが提供されている。

```
$ ls --help
```

/usr/share/doc を確認する

このディレクトリには、サポートが行われている大変多くのプログラムのドキュメントが格納されており、通常はプログラム名とそのバージョン番号で分類されている。例えば、テキストエディタ Emacs Version 21.3 の場合には、/usr/share/doc/emacs-21.3 となる。

This PDF is prepared for tools@maruhouse.org, personal use only.

GNOME と KDE に関するヘルプ

GUI環境を提供するGNOMEあるいはKDEに関するヘルプは、メインメニューからHelpを選択する。

Fedora に関する Web サイトについて

Fedoraオフィシャルサイトは、http://fedora.redhat.comである。また、非公式ではあるがFAQとして有名な、http://fedora.artoo.netというサイトも存在する。また、本書を活用するためのサイトとして、http://www.oreilly.com/catalog/linuxpg/を提供している。

ネットニュース(usenet)

ネットニュースには、Linuxに関する話題が提供されているニュースグループとして、comp.os.linux.miscやcomp.os.linux.questionsなどのグループが数多く存在する。Red Hatに関するニュースグループとして有名なものは、alt.os.linux.redhat、comp.os.linux.redhat、linux.redhat、linux.redhat.miscなどがあり、是非、購読して最新情報を得てほしい。これ以外にも、Google Groups、http://groups.google.comを利用して、ニュースグループから探索して情報を得ることもできる。Google Groupsは、まさにトラブルシューティングのための宝の山といっても過言ではない。

Google

より詳細なドキュメントやチュートリアルなどが必要であれば、Google（http://www.google.com）で検索するとよいだろう。

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

目　次

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

1章 Fedora：さあ、はじめてみよう

それでは、Linux システムにログインしてみよう。さて、ログインすると、最初に図 1 で示すようなグラフィカルなデスクトップ画面が表示されて、あなたを迎え入れてくれるはずである。デスクトップには、以下に示すメニューが含まれる。

- デスクトップ画面の下部には Windows と同様のタスクバーが配置される。
  - ・一番左の“Red Hat”アイコン をクリックすると、多数のプログラムが収められたメインメニューがポップアップ表示される。
  - ・例えば、Mozillaウェブブラウザや、電子メールアプリケーション、プリンタ設定に必要なプリントマネージャなど、プログラムを実行するためのアイコンがデスクトップ上に置かれている。
  - ・デスクトップ切り替えスイッチャ(4つのボックスで区切られた長方形のボタン)を利用すると、現在利用しているデスクトップ画面を保存しつつ、複数のデスクトップ間を自由に移動できるようになる。
  - ・青色のチェックマーク表示は、システムが最新のアップデートが実施されていることを示し、赤色の！表示の場合には、アップデートが行われていないことを示す。後者の場合、最新の状態に更新されていないので、注意してほしい。
  - ・時計
- ファイルやフロッピーディスク、個別のファイルが格納されたディレクトリ(フォルダ)などを削除するためのゴミ箱の他様々なアイコンが置かれている。

Fedoraでは、似たようないくつかのGUIを提供しており、おそらく今、目にしているGUIは GNOME か KDE のいずれかである。Red Hat アイコンをクリックして、メインメニューの中のHelpを選択すると、ウィンドウが表示される。このウィンドウは、GNOMEあるいは KDE についての詳細を表示する。

This PDF is prepared for tools@maruhouse.org, personal use only.

図1 Fedoraのデスクトップ画面

1.1 シェルの役割

GNOMEやKDEのアイコンやメニューを利用して、GUI環境をいろいろと探索してみよう。これらのGUIは、ユーザがLinuxを扱う上で最も重要な機能となる。そして、Fedoraなどの様々なLinuxディストリビューションでは、ユーザがファイルを編集する、メールを読む、Webをブラウズする、といった場合に、面倒な作業を行わず簡単に操作できるインタフェースが多数提供されている。

それだけではない。Linuxの真のパワーは他の部分でも発揮される。Linuxが持つ広大な世界に飛び込むには、シェルを使いこなせるパワーユーザになる必要がある。もちろん、最初はアイコンやメニューを利用する方が容易だろう。しかし、少しでもシェルを使いこなせるようになれば、シェルがいかに簡単で、かつパワフルなものであるかに気づくはずである。ちなみに、本書で取り扱うLinuxコマンドのほとんどが、シェル上で実行できることをあらかじめ述べておきたい。

1.2 シェルの実行方法

Linuxが提供するGNOMEやKDEなどのGUI上でシェルを実行させるには、まずシェルウィンドウを起動する必要がある。シェルウィンドウは、xterm、gnome-terminal、konsole、

This PDF is prepared for tools@maruhouse.org, personal use only.

uxtermなどのプログラムを起動する必要がある。これらのプログラムは、基本的にどれも似たような動作をする。すなわち、シェル実行のためにウィンドウを開くと、コマンドの入力待ち状態になる。Fedoraでは、デフォルトで3つのウィンドウインタフェースを使用している。

グラフィカルインタフェース名	GUI 上での起動方法	シェルウィンドウプログラム名
GNOME	Menu : System Tools : Terminal あるいは、デスクトップ上でマウスを右クリックし、Open Terminal	gnome-terminal
KDE	Menu : System Tools : Terminal あるいは、デスクトップ上でマウスを右クリックし、Open Terminal	konsole
twm	デスクトップ上でマウスを右クリックし、Right Mouse Button : XTerm	xterm

ウィンドウプログラム(例えばkonsoleなど)と、そのウィンドウ内部で実行されるシェルを、同一のシェルとして混同してはならない。ウィンドウは素晴らしい機能を提供してくれるが、あくまでも入れ物(あるいはコンテナ)のようなものに過ぎない。シェルこそが、まさにあなたがこれから入力するコマンドを待ち続け、そして実行する環境を提供してくれるのである。

GUI環境が起動されていない、ネットワーク経由でリモートログインしている、あるいはターミナル経由で直接ログインしている場合には、ログインと同時にシェルが起動されるようになっている。シェルウィンドウはもはや必要ない。

1.3 システムの再起動

本書では、ログイン方法については既知のものとする。なお、GNOME あるいは KDE などの環境からログアウトするには、タスクバーの Red Hat アイコンをクリックし、メインメニューから Logout を選択する。リモートシェルからログアウトするには、コマンドとして exit あるいは logout と入力する(単にシェルを閉じるという方法もある)。

非常に重要なことであるが、Linux システムの電源を落とす場合には、いきなり電源を切断してはならない。電源を落とす前に適切なシャットダウン手順を実行する必要がある。GNOME 環境からシャットダウンを行う場合には、メニュー内の Logout → Shut Down を選

This PDF is prepared for tools@maruhouse.org, personal use only.

択する。KDE環境からシャットダウンを行う場合には、まずログアウトしてから、ログイン画面上のShut Downアイコンをクリックする。また、シェルからシャットダウンを行う場合には、スーパーユーザに権限を変更してから shutdown コマンドを実行する。

shutdown [オプション] *time* [出力メッセージ] SysVinit

/sbin stdin stdout - file -- opt --help --version

shutdown は、Linux システムの停止、あるいは再起動を行うコマンドである。このため、shutdownコマンドが実行できるのはスーパーユーザのみである。以下に例をあげておこう。10分後にシャットダウンを実行し、アナウンスとして“scheduled maintainance”というメッセージをログインしている全ユーザに配信する方法を示す。

```
# shutdown -h +10 "scheduled maintenance"
```

引き数「時間」は、時間(分)を表し、+に続けて値を入力する。例えば、10分であれば+10というように指定する、あるいは16:25のように、時間と分といった絶対時間を指定することも可能である。nowと指定すると、即座にシャットダウンを実行する。オプション無しのまま実行すると、shutdownはシステムのログインモードを、通常のマルチユーザモードからシングルユーザモードへと変更する。 シングルユーザモードとは、特別なメンテナンスモードであり、(システムコンソール上に)1ユーザのみログイン可能で、必須とされない全てのプログラムが停止される。シングルユーザモードを終了するには、システムを停止、あるいは再起動を行うためにshutdownを実行する。また、システムを通常の状態、すなわちマルチユーザモードに復帰させるには ^D を入力する。

オプション

-r システム再起動(reboot)

-h システム停止(halt)

-k 実際にはshutdownを実行しない、これは、システムのシャットダウンを実行するかのように振舞い、全ユーザに警告メッセージを送信するオプションである

-c 進行中のシャットダウンプロセスをキャンセル(引き数「時間」は省略可能)

-f リブート時のファイルシステムチェック fsckプログラムのスキップを指定(詳細は、「4.13 ディスク操作」を参照のこと)

-F リブート時のファイルシステムチェック(fsck)の実行を指定

This PDF is prepared for tools@maruhouse.org, personal use only.

シャットダウン、シングルユーザモードに関する詳細な技術情報に関しては、init、inittab のマニュアルページを参照してほしい。

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

2章 ファイルシステム

Linuxの全ての機能を活用するには、Linuxのファイルシステムとその配置方法について学ばなければならない。Linuxが処理を行う個々のファイルはディレクトリ(directory)と呼ばれるフォルダに格納される。ディレクトリとは、WindowsやMacintoshのフォルダと同様に、階層構造あるいはツリー構造により形成される。ディレクトリには、また別のディレクトリを格納することができ、これをサブディレクトリと呼ぶ。さらに、サブディレクトリの中には、多数のファイル、また別のサブディレクトリが格納され、これは無限の広がりを作り出していく。なお、最上位のディレクトリのことをルート(root)ディレクトリと呼び、このディレクトリは、スラッシュ(/)で示される。

ファイルシステムは、「ファイル名とスラッシュ」で記されたパス(path)を使用して、ファイルやディレクトリの参照を行っている。その一例を以下に取り上げよう。

```
/one/two/three/four
```

これは、まずルート(/)ディレクトリを参照し、続いてルートディレクトリに格納されたoneディレクトリ、twoディレクトリ、threeディレクトリを参照し、それからそのディレクトリに存在するfourファイルを示す。なお、ルートディレクトリで開始されたパスのことを絶対パスと呼び、それ以外のパスを相対パスと呼ぶ。以下、若干補足しておく。

シェルを起動させた時点で、そのシェルは必ず(抽象的な表現で言えば)あるディレクトリに「存在する」ことになる。つまり、シェルはカレントワーキングディレクトリを持ち、シェル上でコマンドを実行させる時には、シェルはこのディレクトリから相対的にファイル参照を行って処理している。より厳密に言えば、そのシェル上で、ある相対ファイルパスを参照する場合には、このファイルパスはカレントワーキングディレクトリへの相対パスとなる。例えば、シェルが /one/two/three ディレクトリに「存在して」いれば、ファイル myfile を参照するコマンドを実行すると、実際には /one/two/three/myfile を参照する、ということになる。それと同様に、相対パス a/b/c、すなわち /one/two/three/a/b/c が実際のパスとなる。

This PDF is prepared for tools@maruhouse.org, personal use only.

さらに特別なディレクトリとして、.(ピリオド1つ)と ..(連続したピリオド2つ)で示されるディレクトリについて取り上げておこう。前者は、カレントディレクトリを示し、後者は、1つ上位のディレクトリである親ディレクトリを示す。カレントディレクトリが /one/two/three であれば、. はこのディレクトリを意味する、そして .. は、/one/two を意味する。

またシェル上で、あるディレクトリから他のディレクトリに移動したい場合には、cdコマンドを使用する。

```
$ cd /one/two/three
```

cdコマンドは、シェル上のカレントのワーキングディレクトリから/one/two/threeに対して移動することを意味する。これは絶対パス(すなわち/で開始されたディレクトリ)である。もちろん、相対パスについても同様である。

```
$ cd d              サブディレクトリ d への移動
$ cd ../mydir       1つ上の親ディレクトリに移動し、それから mydir への移動
```

ファイル名やディレクトリ名として、大文字と小文字、数字、ピリオド、ダッシュ(-)、アンダースコア(_)、それ以外のほとんどの記号(/はディレクトリを分割するため除かれる)を使用することができる。しかし、通常スペース、アスタリスク、丸かっこ()、それ以外にもシェルに何らかの意味を持つ記号を使用は避けたほうがよい。どうしても使用したければ、これらの記号に対して常に引用符をつける必要がある(詳細は、「3.2.10　クオート」を参照のこと)。

2.1 ホームディレクトリ

ユーザのホームディレクトリは、通常、一般ユーザであれば/home以下のディレクトリとなる。スーパーユーザのホームディレクトリは /root である。一般的には、ユーザのホームディレクトリは、/home/あなたのユーザ名 : /home/smith、/home/jonesなどとなる。ホームディレクトリを参照する方法がいくつか存在する。

cd

cdコマンドを引き数なしのまま入力すると、自分のホームディレクトリにシェルのワーキングディレクトリを移動する。

環境変数 HOME

環境変数HOME(「3.2.4　シェル変数」参照)には、ホームディレクトリの名前が指定さ

This PDF is prepared for tools@maruhouse.org, personal use only.

れる。

```
$ echo $HOME      echo コマンドは引き数をそのまま出力
/home/smith
```

チルダ ~ を指定すると、自分のホームディレクトリを意味する。

```
$ echo ~
/home/smith
```

チルダに続いてユーザ名(すなわち~smithなど)を指定すると、指定したユーザのホームディレクトリを示す。

```
$ cd ~smith
$ pwd              「現在のワーキングディレクトリを出力する」コマンド
/home/smith
```

2.2 システムディレクトリ

一般的に、Linux システムには多数のシステムディレクトリがある。これらのディレクトリには、オペレーティングシステムに関わるファイル、アプリケーション、ドキュメント、個々のユーザのファイル(典型的には /home 以下)などが保存されている。

システム管理者でない限り、システムディレクトリを参照することはめったにないと思われるが、システムディレクトリについて少しでも知識があると、ファイルの目的について、推測や理解ができるようになる。なお、これらのファイル名には、3つのパート、すなわちスコープ、カテゴリ、アプリケーションといった情報が含まれていることがある(これらの名前は標準ではないが、理解の手助けになるので本書では取り上げることとした)。例えば、テキストエディタEmacsが格納されたディレクトリ/usr/bin/emacsは、スコープとして/usr(システムファイル)、カテゴリとしてbin(プログラムの実行ファイル置かれるディレクトリ)、アプリケーションとしてemacs(テキストエディタ)であることが分かる。これを図2に示す。本書では、若干順番が変わるが、これらの3つパートについて解説する。

図2 ディレクトリにおけるスコープ、カテゴリ、アプリケーション

This PDF is prepared for tools@maruhouse.org, personal use only.

2.2.1 カテゴリ

カテゴリとは、ディレクトリに存在するファイルの形式を示すものである。例えば、カテゴリがbinであれば、このディレクトリにはプログラムが存在している、ということが容易に分かる。以下では、共通カテゴリとされているリストを取り上げる。

プログラムカテゴリ

bin	プログラム（通常バイナリファイル）
sbin	スーパーユーザ、すなわちrootユーザによって実行されることが意図されたプログラム（通常バイナリファイル）
lib	プログラムで使用されるライブラリ
libexec	通常、ユーザが起動せずに、あるプログラムによって実行が制御されるプログラムのこと。すなわち「実行可能なプログラムによるライブラリ」である

ドキュメントライブラリ

doc	ドキュメント
info	Emacsにビルドインされたヘルプシステムのためのドキュメントファイル
man	manプログラムによって表示されるドキュメントファイル（マニュアルページ）であり、これらは圧縮されたファイル、あるいは、manコマンドが解釈できるディレクトリに構成
share	プログラムに依存したファイルであり、使用例やインストールに関する説明など

コンフィギュレーションカテゴリ

etc	システム（それ以外にも多方面にわたる諸機能に関する）設定ファイル
init.d	ブート設定用ファイル
rc.d	ランレベル（runlevel）に応じたブート設定用ファイル、これ以外にもrc1.d、rc2.dなどが存在

プログラムカテゴリ

include	プログラムのヘッダファイル
src	プログラムのソースコード

This PDF is prepared for tools@maruhouse.org, personal use only.

Web カテゴリ

cgi-bin	Web システム上で実行可能なスクリプト／プログラム
html	Web ページコンテンツ
public_html	通常、ユーザのホームディレクトリに存在している Web ページディレクトリ
www	Web ページコンテンツ

表示関連カテゴリ

fonts	フォント
X11	X ウィンドウシステムファイル

ハードウェアカテゴリ

dev	ディスクやそれ以外のハードウェアへのインタフェースであるデバイスファイル
mnt	マウントポイント、これはディスクへのアクセスを提供するディレクトリ
misc	autofs が使用するディレクトリ

ランタイムファイルカテゴリ

var	システムの起動により作成、更新されたファイルでありシステム依存ファイル
lock	ロックファイル、すなわち「プログラム実行中」を通知するために作成されるファイルのこと。ロックファイルが存在すると、同一のプログラムが別に再度起動が行われるといった状況を避けることが可能
log	重要なシステムイベント、エラー・警告情報、それ以外にも何らかの情報が格納されるログファイル
mail	受信したメールを格納するメールボックス
run	実行中のプロセスのID番号が格納されるプロセスIDファイルのこと。このファイルはある特定のプロセスを追跡する、あるいは停止させる際などに使用される
spool	送信メールや、プリント用ジョブ、スケジューリングされたジョブなどに使用されるキューイングファイル
tmp	ユーザが実行するプログラムなどのためのテンポラリファイル

This PDF is prepared for tools@maruhouse.org, personal use only.

proc	オペレーティングシステムのシステム状態の管理。「2.3 ディレクトリ」参照のこと

2.2.2 スコープ

ディレクトリパスのスコープについて以下に説明する。これは、ディレクトリの階層構造についてより明確に示すことを目的とし、その共通の情報は以下のようになる。

/	Linux の「ルート」全般に関わるシステムファイル
/usr	Linux の「ユーザ」全般に関わるより多くのシステムファイル
/usr/games	ゲーム
/usr/kerberos	kerberos 認証システムに関わるファイル
/usr/local	「ローカル」、すなわち読者自身の計算機あるいは機関で独自にコンパイルされたシステムファイル
/usr/X11R6	X Windows System に関わるファイル

lib(ライブラリ)といったカテゴリに対し、Linuxシステムでは/lib、/usr/lib、/usr/local/lib、/usr/games/lib、/usr/X11R6/libなどのディレクトリが提供されていることが多い。もちろん、それ以外のスコープとして、例えば、システム管理者向けに/my-company/lib、/my-division/lib などのディレクトリが提供されていることもある。

実際には/と/usrに関して、その明確な区別というものは存在しない。しかし、/は、「ローレベル」なファイル、すなわちオペレーティングシステム依存のファイルであり、/bin には ls、cat、/usr/binといった重要なプログラムが置かれている。また、このディレクトリには、Linux ディストリビューションで提供された数多くの様々なアプリケーションが格納されている。/usr/local/binには、システム管理者がインストールしたアプリケーションがプログラムが格納されている。なお/usr/local 以下にrpmパッケージのファイルはインストールされていない。

2.2.3 アプリケーション

ディレクトリパスが格納されているアプリケーションについて、そのディレクトリ名は、通常(格納されている)プログラムの名前になる。スコープ、カテゴリ(例えば /usr/share/doc)以下に、プログラムが格納されたサブディレクトリ(/usr/share/doc/myprogram)を持つ。

This PDF is prepared for tools@maruhouse.org, personal use only.

2.3 ディレクトリ

/boot

このディレクトリは、システムブートに必要なファイルであり、カーネル(kernel)が存在するディレクトリである。通常、/boot/vmlinuz- バージョン - リリース名(例:/boot/vmlinuz-2.6.9-1.667)となる。

/lost+found

ディスクリカバリツールによって復旧したファイル

/proc

現在進行中のプロセスを管理する特別なファイル

/proc ファイルは、動作中のカーネルの情報を提供するなど、特殊な性質を持つファイルである。通常はサイズが0byte、読み出しのみ(read-only)のファイルであり、常に更新されている。

```
$ ls -l /proc/version
-r--r--r--   1 root   root    0 Nov  10 22:55 /proc/version
```

このファイルには、Linux カーネルに関する情報が格納される。

```
$ cat /proc/version
Linux version 2.6.9-1.667smp...
```

多くの場合、これらのファイルはプログラムによって使用される。以下に例をあげる。

/proc/ioports	計算機の入出力(I/O)ハードウェアのリスト
/proc/version	オペレーティングシステムバージョン。これはunameコマンドが出力する情報と同様
/proc/uptime	システムの稼働時間(uptime)であり、これはシステムが最後に起動してからの経過時間(秒)である。なおuptimeコマンドによって人間が読みやすい型式での表示が可能
/proc/nnn	nnn は整数であり、プロセス IDnnn のプロセスに関する情報を表示
/proc/self	実行中のプロセスに関する情報であり、/proc/nnnへのシンボリックリンクである。このファイルは自動的に更新される。何秒間か、連続して ls -l /proc/selfコマンドを実行してみてほしい。/proc/self自身が変化していることが分かる

This PDF is prepared for tools@maruhouse.org, personal use only.

2.4 ファイル保護

Linux システムでは、ログインアカウントの管理により多くのユーザアカウントを作ることが可能である。このため、プライバシやセキュリティ保護のために、各ユーザはシステム上のいくつかのファイルに対してアクセス権限を設定することができる(全ファイルではない)。このアクセスコントロールには、2通りの保護によって構成される。それぞれを以下に説明する。

誰が権限を有しているのか？

各ファイルとディレクトリには、誰がアクセス権限を有しているかを示すオーナ(owner)が設定されている。通常、ファイル作成者がオーナであるが、それ以外の複雑な関係になっている場合もある。これに加えて、先に述べたユーザで構成されたグループ(group)に対してもアクセス権限を設定することもできる。なお、グループは、システム管理者によって設定される。詳細は、「6.4 グループの操作」を参照のこと。また、システム上にログインアカウントを持つ 全ユーザがアクセス出来るという権限がある。これは「ワールド(world)」あるいは「other」と呼ばれている権限である。

アクセス権限の種類にはどのようなものがあるのか？

ファイル所有者、グループ、ワールドは、特定のファイルに対して、それぞれ読み出し(read)、書き出し(write)、編集(modify)、実行(execute)パーミッション設定が可能である。さらに、パーミッションはディレクトリに対しても設定可能であり、read(ディレクトリ中のファイルへのアクセス)、write（ディレクトリ中のファイルの作成・削除)、execute(ディレクトリ内の参照許可)の設定が行える。

ファイルのオーナとパーミッションを参照するには、以下のように実行する。

```
$ ls -l filename
```

ディレクトリのオーナとパーミッションを参照するには、以下のように実行する。

```
$ ls -ld directory_name
```

ファイルパーミッションは、左から10文字で出力されr(read)、w(write)、x(execute)、それ以外の文字から構成される。以下、例を示す。

```
drwxr-x---
```

This PDF is prepared for tools@maruhouse.org, personal use only.

出力された文字と記号が何を意味するかを説明しておこう。

文字数	意味
1	ファイル形式：- ＝ファイル、d ＝ディレクトリ、l ＝シンボリックリンク、p ＝名前付パイプ、c ＝キャラクタデバイス、b ＝ブロックデバイス
2-4	ファイル所有者のread、 write、 execute権限
5-7	ファイルグループの read、write、execute 権限
8-10	それ以外の全ユーザの read、write、execute 権限

lsコマンドの詳細は、「4.3　ディレクトリ操作」を参照のこと。オーナ、グループオーナ、ファイルのパーミッションを変更するには、chown、chgrp、chmodの各コマンドを使用してほしい。動作の違いについては、「4.6　高度なファイル操作」を参照のこと。

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

3章 シェル

Linuxシステムでは、コマンドを実行する際に、そのコマンドを「何か」の上で実行する必要がある。その「何か」がまさにシェルである。シェルは、コマンドラインのユーザインタフェースであり、コマンドを入力してEnterキーを押すと、リクエストしたプログラム(もしくは複数のプログラム)を実行する。なお、シェルの実行方法については、「1章　Fedora:さあ、はじめてみよう」を参照のこと。

誰がログインしているか知りたい場合、シェル上で以下のコマンドを実行する。

```
$ who
barrett      :0    Sep 23 20:44
byrnes    pts/0    Sep 15 13:51
silver    pts/1    Sep 22 21:15
silver    pts/2    Sep 22 21:18
```

ドルマーク"$"は、シェルプロンプトを表しており、これはシェルがコマンドの実行準備が出来ていることを示す。1つのコマンドから同時に複数のプログラムを実行させることも可能であり、さらにそれらのプログラム同士でやり取りさせること、すなわち協調動作も可能である。ここでは、whoプログラムの出力結果を直接wcの入力にリダイレクトする例を取り上げてみよう。以下は、ファイル中のテキスト行数をカウントする例である。すなわちwhoの出力に対して、その行数をカウントするというコマンドである。

```
$ who | wc -l
4
```

このコマンドを実行することで、何人のユーザがログインしているかが出力される。縦の棒線のことをパイプ(pipe)と呼び、whoとwcコマンドをつなげる役目をする。

シェルは、それ自体がプログラムであり、Linuxにはいくつのかのシェルが存在する。有名なシェルとしてbash(bourne-again shell)を紹介しよう。bashは、Fedora Linuxの場合、デフォルトで/bin/bashにインストールされる。

This PDF is prepared for tools@maruhouse.org, personal use only.

3.1 シェル vs プログラム

コマンドを実行する、これは(whoのような)Linuxプログラムを実行すること、そうでなければシェル自身の機能として「ビルトインされたコマンド」を実行することに他ならない。type コマンドによる違いについては以下のことが言える。

```
$ type who
who is /usr/bin/who
$ type cd
cd is a shell builtin
```

この例から、シェルが何を提供し、そしてLinuxが何を行うのかについて手がかりを得ることができるだろう。次節では、シェルの機能について解説する。

3.2 bash の特徴

シェルは、簡単にコマンドを実行するための環境である。それだけでなく、コマンド実行のタスクを容易に行うための強力な機能も提供している。ここでも、いくつかの例を示そう。具体的には、ファイル名のパターンマッチングとして、ワイルドカード、コマンド出力のリダイレクト、ファイルからの入力やコマンドからの出力を別のコマンドへの入力として渡すパイプ、よく使用するコマンドを即座に実行するための方法として便利なエイリアス、シェルによって使用される環境変数などについて取り上げる。さらに、本書では、有用なツールを紹介する目的でこれらの説明を行う。完全なドキュメントを参照するには info bash を実行してみてほしい。

3.2.1 ワイルドカード

ワイルドカードは、類似した名前を持つファイルの集合を容易に指定する手法である。例えば、a*は、小文字のaで始まる名前を持つ全てのファイルを意味する。ワイルドカードは、名前が一致するファイル名へと、シェルによって展開されている。例えば、以下のように入力したとしよう。

```
$ ls a*
```

シェルは、まずa*を、カレントディレクトリ中のaで始まるファイル名に対して展開する、すなわち、ユーザが以下のコマンドを入力したことと同じである。

```
ls aardvark adamantium apple
```

This PDF is prepared for tools@maruhouse.org, personal use only.

lsコマンドには、引き数にワイルドカードが使用されたかは分からない。lsコマンドは、シェルによって展開された後のファイル名を引き数として扱う。

ワイルドカード	意味
*	先頭がピリオドである文字を除いた全ての文字
?	文字一文字
[*set*]	*set*で指定した単一の文字。通例、すべての母音[aeiouAEIOU]である文字列、あるいは、全ての大文字[A-Z]であるダッシュで範囲が指定された文字列
[^*set*] [!*set*]	(上述した)*set* で指定したもの以外の単一の文字

setを使用する場合に、文字列にダッシュ(-)を含めたい場合は、その文字列の先頭か終端に記述する。同様に]を含めたい場合は、先頭に記述する。なお、^あるいは!を含めたい場合は、先頭に記述してはならない。

3.2.2 中かっこ{}

ワイルドカードと類似したものとして、中かっこを使用して複数の引数をコマンドに渡す方法がある。これは各要素をカンマで区切る。

```
{a,b,cc,ddd}
```

この指定により、以下の意味をなす。

```
a b cc dddd
```

中かっこは、いかなる文字列に対しても有効であり、ワイルドカードのようにファイル名に対して何らかの制限を持つことはない。例えば、sand{X,Y,ZZZ}wichは、以下のように展開される。

```
$ echo sand{X,Y,ZZZ}wich
sandXwich sandYwich sandZZZwich
```

当然ながら、これらのファイルはカレントディレクトリに存在するファイルを指す。

3.2.3 チルダ ~

シェルは、チルダ(~)を特別な文字として処理する。チルダのみで表示される場合とチルダの後に文字列が続く場合がある。

This PDF is prepared for tools@maruhouse.org, personal use only.

~	自分のホームディレクトリ
~smith	ユーザ smith のホームディレクトリ

3.2.4 シェル変数

シェルでは、シェル変数を定義できる。また、その変数に値を設定することができる。

```
$ MYVAR=3
```

値を参照するには、変数名の前にドルマーク($)を指定する。

```
$ echo $MYVAR
3
```

変数の中には、標準としてログイン時に定義されるものが存在する。

変数	意味
DISPLAY	X Windows System 表示画面名
HOME	ホームディレクトリ名
LOGNAME	ログイン名
MAIL	メールボックスへのパス
OLDPWD	以前のディレクトリ
PATH	サーチパス。なお、複数のディレクトリはコロン(:)で区切る
PWD	カレントディレクトリ
SHELL	自分のシェルへのパス。一例として /bin/bash など
TERM	ターミナルの種類。一例として xterm や vt100 など
USER	ログイン名

シェル変数を参照するには、以下を実行する。

```
$ printenv
```

変数のスコープ(変数が参照できる範囲)は、デフォルトで変数が定義されたシェルに依存する。変数を作成し、その変数に値をセットし、プログラムがその変数を利用できるようにするには、シェルはサブシェルを起動する必要がある。これには、以下のようにexportコマンドを実行する。

```
$ export MYVAR
```

This PDF is prepared for tools@maruhouse.org, personal use only.

もしくは、以下のように一度に指定する方法もある。

```
$ export MYVAR=3
```

この変数は、環境変数と呼ばれる。名前はシェル上の「環境」において、別のプログラムがその変数を利用できることからその名が付けられている。なお、指定したプログラムに対してのみ特定の値を参照出来るようにするには、*variable=value* をコマンドラインに追加する。

```
$ echo $HOME
/home/smith
$ HOME=/home/sally echo "My home is $HOME"
My home is /home/sally
$ echo $HOME
/home/smith                    オリジナルの値は影響無し
```

3.2.5 コマンドサーチパス

ところで、最も重要な変数がPATH変数である。これは、シェルがプログラムを見つけるための場所を指示する変数である。例えば、以下のコマンドを入力したとしよう。

```
$ who
```

この場合、シェルはそのリクエストに対して何とかしてこのプログラムを探し出そうとする。まさにこれを解決するのがPATH変数である。この変数は、シェルが参照する複数のディレクトリをコロンで区切って指定する。

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/smith/bin
```

この場合、上述したディレクトリからwhoコマンドを探索する。whoコマンドを見つけ出せれば(/usr/bin/who)、シェルはコマンドを実行するというわけである。コマンドが見つけ出せなければ、シェルは以下のような返答をする。

```
bash: who: command not found
```

シェルのサーチパスに一時的にディレクトリを追加する場合には、PATH変数を修正する必要がある。ここでは、その一例を取り上げる。シェルのサーチパスに /usr/sbin パスを一時的に追加するには、以下のように行う。

```
$ PATH=$PATH:/usr/sbin
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/smith/bin:/usr/sbin
```

This PDF is prepared for tools@maruhouse.org, personal use only.

また、この変更を永続的に受け入れるには、シェルのスタートアップファイル~/.bash_profileに記述されているPATH変数を修正する。なお、詳細は、「3.6 シェル動作の設定」を参照のこと。この変更を行ったら、一旦ログアウトして、再度ログインすることに注意する。

3.2.6 エイリアス

ビルトインコマンドであるエイリアス(alias)は、長い名前を持つコマンドをより簡単かつ便利に実行出来るように、短い名前、あるいは覚えやすい名前として定義するコマンドである。以下に例を示す。

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

すなわちllという入力により、ls -lコマンドを実行できるように設定したことになる。

```
$ ll
total 436
-rw-r--r--    1 smith     3584 Oct 11 14:59 file1
-rwxr-xr-x    1 smith       72 Aug  6 23:04 file2
...
```

ログイン時には常にエイリアスを有効にするには~/.bashrcファイルにエイリアスを定義する必要がある。詳細は、「3.6 シェル動作の設定」を参照のこと。全てのエイリアスを参照するには、引き数を付けずにaliasと入力する。「8章 シェルスクリプトによるプログラミング」を参照のこと。また、info bashを実行し、その中の「シェルの関数」について理解を深めてほしい。

3.2.7 入出力リダイレクト

シェルは、ファイルに対して標準入力、標準出力、標準エラー出力、各々をリダイレクトすることができる。言い換えれば、標準入力から読みこむ内容を、シェルの<演算子によって、ファイルからの入力を扱うことができる。

```
$ mycommand < infile
```

同様に、標準出力への書き出しも可能である。

```
$ mycommand > outfile              outfileの作成／上書き
$ mycommand >> outfile             outfileへ追加
```

また、標準エラー出力への書き出しを行うコマンドは、その出力結果をファイルにリダイ

This PDF is prepared for tools@maruhouse.org, personal use only.

レクトすることも可能である。

```
$ mycommand 2> errorfile
```

標準出力と標準エラー出力の両方の結果をファイルにリダイレクトするには、以下のようにする。

```
$ mycommand > outfile 2> errorfile   別々のファイルへ出力
$ mycommand > outfile 2>&1           同じファイルへ出力
```

3.2.8 パイプ

シェルのパイプ(|)機能を使用することで、あるコマンドの標準出力の結果を別のプログラムの標準入力としてリダイレクトすることもできる。一例を紹介する。

```
$ who | sort
```

この例は、whoの結果をsortプログラムに渡すことを示しており、結果としてログインユーザの一覧をアルファベット順にソートした結果を出力する。

3.2.9 コマンドの組合せ

1行のコマンドで、複数のコマンドを実行させるには、各コマンドをセミコロン(;)で区切り指定する。

```
$ command1 ; command2 ; command3
```

連続した複数のコマンドを実行する場合に、前のコマンドの実行が失敗した時に、以下次のコマンドが停止するようにするには、&&(and)記号を各コマンドの区切りとして指定する。

```
$ command1 && command2 && command3
```

同様に、連続した複数のコマンドを実行する際に、ある1つのコマンド実行が成功すれば実行を停止する、といったようにしたければ、||(or)記号を各コマンドの区切りとして、指定する。

```
$ command1 || command2 || command3
```

3.2.10 クオート

通常、シェルは空白を単にコマンドライン上の文字間のスペースとして扱う。空白の文字

This PDF is prepared for tools@maruhouse.org, personal use only.

を含めたいのであれば(例えば、空白を含むファイル名など)、シェルがその空白を扱えるように、シングルクオート(')あるいはダブルクオート(")で囲む。シングルクオートの場合には、文字をそのまま処理するが、ダブルクオートの場合には、変数などを展開する。

```
$ echo 'The variable HOME has value $HOME'
The variable HOME has value $HOME
$ echo "The variable HOME has value $HOME"
The variable HOME has value /home/smith
```

バッククオート(`)は、その内容をコマンドとして処理する。すなわち、その内容が、コマンドの標準出力によって置き換えられる。

```
$ /usr/bin/whoami
smith
$ echo My name is `/usr/bin/whoami`
My name is smith
```

3.2.11 エスケープ

ある特定の文字をシェルに対して特別な意味を持たせたい場合がある。例えば、そのまま文字通りに処理したい(ワイルドカードとしてでなく、文字+アスタリスク*として表現したい)場合には、その文字の前にバックスラッシュ(\)を指定する。これは、エスケープと呼ばれており、特別な文字である。

```
$ echo a*                  ワイルドカードとして a ファイル名と一致
aardvark  agnostic  apple
$ echo a\*                 アスタリスクそのものを出力
a*
$ echo "I live in $HOME"   ドルマークは変数の値を示す
I live in /home/smith
$ echo "I live in \$HOME"  ドルマークそのものを出力
I live in $HOME
```

さらに、制御文字(タブ、改行、^Dなど)についても、コマンドライン上で文字通りに使用することも可能である。この場合、制御文字^Vを表示する文字の前に指定する。これは、bashが補完で使用するタブ文字(^I)の出力に非常に有用である。詳細は、「4.5 ファイルの作成と編集」を参照のこと。

```
$ echo "There is a tab between here^V^Iand here"
There is a tab between here     and here
```

This PDF is prepared for tools@maruhouse.org, personal use only.

3.2.12 コマンドライン編集

bashは、テキストエディタであるemacsやviでのキー入力と同様の方法で、動作中のコマンドラインを修正できる(詳細は、「4.5 ファイルの作成と編集」を参照のこと)。emacsによるキー入力でコマンドライン編集を行うには、以下のコマンドを実行する(永続的に設定したい場合は、以下の行を~/.bash_profileファイルに追加する)。

```
$ set -o emacs
```

viによるキー入力で編集を行うには以下のように実行する。

```
$ set -o vi
```

emacsキー入力	viキー入力(最初にESCを入力)	意味
^Pあるいは↑	k	前の行へ
^Nあるいは↓	j	次の行へ
^Fあるいは→	l	1文字右へ
^Bあるいは←	h	1文字左へ
^A	0	文頭へ
^E	$	文末へ
^D	x	次文字削除
^U	^U	1行削除

3.2.13 ヒストリ

既に実行したコマンドを再び呼び出すことが可能であり、これがシェルのヒストリと呼ばれる機能である。ヒストリを利用して、一度実行したコマンドを繰り返して実行することができる。ここでは、ヒストリに関連する有用なコマンドについて取り上げていこう。

コマンド	意味
history	ヒストリ出力
history N	ヒストリ中の最新の*N*個のコマンド出力
history -c	ヒストリのクリア(削除)
!!	直前のコマンド実行
!N	ヒストリ中の*N*番目のコマンド実行
!-N	入力したN個のコマンドよりも以前のコマンド実行
!$	直前コマンドの最終パラメータ $ ls a* $ rm !$

This PDF is prepared for tools@maruhouse.org, personal use only.

コマンド	意味
!*	直前コマンドの全てのパラメータ

3.2.14 ファイル名の補完

ファイル名を入力する際、その途中でTABキーを押してみてほしい。シェルは、自動的にファイル名の補完を行う。ファイル名の候補として複数の選択肢が存在している場合には、シェルはビープ音を発し、未確定状態であることを通知する。そのまま続けてTABを押すと、シェルはその候補を出力する。理解のためにまずは試してみてほしい。

```
$ cd /usr/bin
$ ls un<TAB><TAB>
```

3.3 ジョブコントロール

jobs	ジョブのリスト表示
&	バックグラウンドでのジョブ実行
^Z	カレント(フォアグラウンド)ジョブのサスペンド実行
suspend	シェルのサスペンド実行
fg	サスペンド中のジョブのアクティブ化：フォアグラウンドへの移行
bg	サスペンド中ジョブのバックグラウンドでのアクティブ化

全てのLinuxシェルは、ジョブコントロールを行っており、この機能はバックグラウンド(複数の動作が背後で行われるマルチタスク処理)とフォアグランド(シェルプロンプト上のアクティブなプロセス処理)のように、プログラム実行の制御を行う。ジョブとはシェルの仕事の単位である。コマンドをインタラクティブに実行した場合、シェルは実行したコマンドを1つのジョブとして扱う。コマンドが完了すると、それと関連するジョブは消失する。ジョブは、Linuxプロセスのようにローレベルの処理ではなく、ハイレベルでの処理が行われているため、Linuxのオペレーティングシステム側ではジョブについては全く関知していない。すなわち、ジョブは、単にシェルを構成する要素に過ぎない。ジョブに関する重要な用語のいくつかを以下に示す。

フォアグラウンドジョブ

シェル上で実行中のジョブであり、シェルプロンプトが利用中の状態である。このため他のコマンドの実行はできない

This PDF is prepared for tools@maruhouse.org, personal use only.

バックグラウンドジョブ
シェル上で実行中のジョブであるが、シェルプロンプトが使用中ではないため、別のコマンドを同一のシェルで実行できる

サスペンド
一時的にフォアグラウンドジョブを停止

レジューム
サスペンド中のジョブを再び開始

jobs

jobsは、カレントシェル上で実行中のジョブの一覧表示を行うビルトインコマンドである。

```
$ jobs
[1]-  Running                 emacs myfile &
[2]+  Stopped                 su
```

左側の数字はジョブ番号であり、プラス(+)は、fg(フォアグラウンド)コマンドとbg(バックグラウンド)コマンドから制御可能なデフォルトジョブであることを示す。

&

コマンドラインの終端に記述したアンパサンド(&)は、コマンドをバックグラウンドジョブとして実行することを示す。

```
$ emacs myfile &
[2] 28090
```

シェルの返り値としてジョブ番号(2)とコマンドのプロセスID（28090）が返される。

^Z

シェル上で^Zと入力すると、フォアグラウンドで実行中のジョブはサスペンドする。^Z入力は、単に実行を停止させるだけではなく、その状態を記憶し、再度開始することが可能となる。

```
$ mybigprogram
^Z
[1]+  Stopped                 mybigprogram
$
```

This PDF is prepared for tools@maruhouse.org, personal use only.

サスペンド(一時停止)中のジョブをバックグラウンドジョブにする場合にはbgと入力し、フォアグラウンドにレジューム(復旧)するのであればfgと入力する。

suspend

suspendは、可能な状態であればカレントシェルを一時停止するビルトインコマンドである。これは、シェル上で^Zと入力することと同一である。以下の例を見てほしい。この例は、はじめに、suコマンドを実行した後、元のシェルに復帰したい場合を示したものである。

```
$ whoami
smith
$ su -l
Password: *******
# whoami
root
# suspend
[1]+  Stopped                 su
$ whoami
smith
```

bg [% ジョブ番号]

bgは、バックグラウンド中のサスペンドジョブに対して実行リクエストを行うビルトインコマンドである。bgコマンドは、引数が指定されていない場合には、最後にサスペンドしたジョブに対してリクエストが行われる(jobsコマンドによって示される)。特定のジョブを指定する場合には、パーセント(%)に続けてジョブ番号を指定する。

```
$ bg %2
```

アクティブなジョブの中には、バックグラウンドのジョブとして渡すことが出来ないものも存在する。例えば、何らか入力を待機しているジョブなどである。このようなコマンドをバックグランウンドで動作させようとすると、シェルはジョブをサスペンドして以下のように表示する。

```
[2]+  Stopped                 コマンドライン
```

(fgを入力すると)ジョブをレジュームすることができて、継続される。

This PDF is prepared for tools@maruhouse.org, personal use only.

fg[% ジョブ番号]

fgは、サスペンドジョブあるいはバックグラウンドジョブからフォアグラウンドに移動するためのビルトインコマンドである。引き数無しの場合には、通常、最近のサスペンドジョブあるいはバックグラウンドジョブが選択される(jobsコマンドによって示される)。特定のジョブを指定する場合には、パーセント(%)に続いてジョブ番号を指定する。

```
$ fg %2
```

3.4 実行中コマンドの停止

フォアグラウンドで実行中のシェルからコマンドを起動している場合に、そのコマンドを即座に停止したいのであれば、^C と入力してみてほしい。シェルは、^C という入力を「カレントフォアグラウンドコマンドを今すぐに終了しなさい」と解釈する。あるいは、ファイルサイズが大きいものを表示している最中に(例えば、catコマンドを実行中に)これを停止させたい場合にも、同様に ^C と入力する。

```
$ cat bigfile
This is a very long file with many lines. Blah blah blah blah blah blah blahblahblah ^C
$
```

バックグラウンドで実行中のプログラムを削除するにはfgと入力し、フォアグラウンドに移動してから ^C を押す、あるいは、kill コマンドを使用してほしい。詳細は「5.2 プロセス制御」を参照のこと。

しかし、一般的には^Cは、プログラムを終了する方法として適切ではない。プログラム自体が終了する手段を提供されている場合は、できる限りその方法を使用して終了する。既に実行してみた方はお分かりだろうが、^Cによりプログラムを即座に停止することができるのは確かだが、実は正常に停止できたかどうかを確認する手段がない。フォアグラウンドのプログラムを削除(kill)することにより、キー入力したものが表示されずシェルが異常な状態、あるいは不安定な状態になることにもなりかねない。もし、このような状態に陥った時には、以下を実行してみてるとよいだろう。

1. シェルプロンプトを得るには ^J を入力する。この操作は、Enter キー(改行)と同一の意味であるが、正常に Enter が入力できない時などには有効な手段である。
2. 入力した文字が表示されない場合、resetと入力し、コマンドを実行するために再度

This PDF is prepared for tools@maruhouse.org, personal use only.

^J と入力してみて欲しい。この操作により、シェルがリセットされる。

^Cは、シェルに入力された時のみ有効である。なお、入力したはずのウィンドウが何も反応しない場合には、このシェルウィンドウではない可能性が高い。さらに、いくつかのプログラムの中には、^C を「別の意味として認識してしまう」ため、それが無視されてしまうこともある。その一例として、テキストエディタ emacs がある。

3.5 シェルの終了

シェルを終了するには、exit コマンド、あるいは ^D を押す。

```
$ exit
```

3.6 シェル動作の設定

シェルを特定の方法で動作するように設定するには、ホームディレクトリ上の.bash_profile と.bashrcを編集する。ログイン時には、(~/.bash_profile)ファイル、シェルのオープン時には(~/.bashrc)ファイルが実行される。これらのファイルでは、環境変数やエイリアス、プログラムの実行などを自由に設定することができる。

なお、この2つのファイルはシェルスクリプトの一例であり、シェルコマンドが含まれた実行可能な形式のファイルである。詳細は、「8章 シェルスクリプトによるプログラミング」を参照のこと。

This PDF is prepared for tools@maruhouse.org, personal use only.

4章 ファイル管理

4.1 パッケージのインストール

さて、Linuxを利用していくにつれて、新しいソフトウェアをインストールする必要が出てきたはずである。実はFedoraや、その他Linuxディストリビューション用には、多くのパッケージ化されたソフトウェアが存在する。以下に取り上げていこう。

***.rpmファイル**

これはRed Hat Package Manager(RPM)ファイルと呼ばれており、rpm(手動インストール)、あるいはup2date(自動インストール)プログラムによってインストール作業、およびソフトウェア管理が行われる。

***.tar.gzファイル、*.tar.Zファイル、*.tar.bz2ファイル**

これらは圧縮tarファイルと呼ばれ、tar、gzip(.gz)、compress(.Z)、bzip2(.bz2)で圧縮されたファイルである。

ほとんどの新しいソフトウェアについては、スーパーユーザ権限でインストールを行う必要があり、インストール作業を行う前にsuコマンド(あるいはそれと同等のコマンド)を実行する必要がある。以下にその例を取り上げる。

```
$ su -l
Password: ********
# rpm -ivh mypackage.rpm
...
```

新しいソフトウェアをインストールする場合には、Linux CD-ROMや以下のサイトを確認するとよいだろう。

```
http://www.fedora.jp/
http://freshmeat.net/
http://freshrpms.net/
```

This PDF is prepared for tools@maruhouse.org, personal use only.

```
http://rpmfind.net/
http://sourceforge.net/
http://jaist.dl.sourceforge.net/
```

up2date [オプション] [パッケージ]

up2date

/usr/bin stdin stdout - file -- opt --help --version

up2dateコマンドは、Fedoraシステムを最新状態にする上で、大変役立つツールである。これを実行するには、root ユーザに変更してから、以下を実行するだけでよい。

```
# up2date
```

これにより、グラフィカルなユーザインタフェースが表示される。コマンドライン上でもup2date は実行可能である。

```
# up2date -l
```

これにより、システム上で利用可能な全ての最新 RPM パッケージ が一覧表示される。提供されているパッケージをダウンロードするには、以下のように実行する。

```
# up2date -d packages
```

up2date -d を実行し、既にダウンロード済みの RPM パッケージをインストールするには、以下のように実行する。

```
# up2date -i packages
```

up2dateプログラムは、インターネット上のRed Hatサイトあるいは Fedoraと関連したサーバから RPM パッケージをダウンロードする。初めて up2date プログラムを実行する際には、システムの登録が必要になることもある。

Linuxユーザの中には、up2dateプログラム以外にも、yum（http://linux.duke.edu/projects/yum/）やapt（http://ayo.freshrpms.net/）といったプログラムを利用する人もいる。

rpm [オプション] [ファイル]

rpm

/bin stdin stdout - file -- opt --help --version

RPMパッケージをインストールする場合には、手動でrpmコマンドを利用してみてほしい。

This PDF is prepared for tools@maruhouse.org, personal use only.

実は、up2dateコマンドは、バックグラウンドでrpm(パッケージ管理)プログラムを実行しているにすぎない。なお、rpmコマンドは、単にソフトウェアのインストールを行うだけでなく、インストール条件が全て満たされているかの確認作業までも行う。例えば、パッケージsuperstuffが、インストールされていないパッケージotherstuffを必要とする場合、rpmコマンドはsuperstuffのインストールを行わない。このため、インストール準備の全て完了した時にのみ、rpmコマンドはソフトウェアのインストールを開始する。

RPMファイル名は、通常*name-version.architecture*.rpmといった形式を取ることが多い。例えば、emacs-21.3-17.i386.rpmであればemacsパッケージ、バージョン21.3-17、i386用(Intel 80386以降)である。場合によって、rpmは、引き数に(emacs-21.3-17.i386.rpmなどのように)ファイル名が必要なことがある。また、emacsのように、単にパッケージ名のみ指定する場合もある。

RPMパッケージツールの一般的な操作方法を示す。

`rpm -q` *package_name*
: これにより、名前が*package_name*であるソフトウェアがインストールされているかどうかを確認し、そのバージョン番号を出力する。その一例を取り上げよう。`rpm -q textutils`パッケージ名が分からない場合には、まず全パッケージリストを表示し、grepコマンドを用いてパッケージ名を探すという方法がよい。

```
$ rpm -qa | grep -i likely_name
```

`rpm -ql` *package_name*
: インストール済みのパッケージの中で、指定した名前を持つパッケージリストを表示する。以下に、例を取り上げる。

`rpm -qi` *package_name*
: パッケージに関する情報を参照する。

`rpm -qlp` *package_name*
: インストールされていないRPMファイルの内容のリストを表示する。RPMファイルに関する情報を出力するには、以下のように実行する。

`rpm -qa`
: インストール済みの全RPMパッケージを一覧表示する。パッケージ名を元に検索する場合には、パイプ経由でgrepコマンドを利用するのがよいだろう。

```
rpm -qa | grep -i emacs
```

This PDF is prepared for tools@maruhouse.org, personal use only.

`rpm -qf filename`
指定したファイル名を持つインストール済みのパッケージを出力する。

```
$ rpm -qf /usr/bin/who
coreutils-5.2.1-31
```

`rpm -ivh package1.rpm package2.rpm ...`
システムにインストールされていないパッケージのインストールを行う。

`rpm -Fvh package1.rpm package2.rpm ...`
システム上に既にインストールされているパッケージのアップデートを行う。

`rpm -e package_names`
システムからパッケージを削除する。パッケージ削除の実行には、そのバージョン番号は不要であり、単にパッケージ名を指定するだけでよい。例えば、GNU Emacsパッケージemacs-21.3-17.i386.rpmがインストールされている場合には、アンインストール時に、`rpm -e emacs-21.3-17.rpm` と指定するのではなく、単に `rpm -e emacs` と指定して実行すればよい。

tar.gz と tar.bz2

拡張子が .tar.gz または .tar.bz2 であるファイルの場合、その中身はインストール前のコンパイルに必要なソースコードであることが一般的である。

1. パッケージ内容について、1行ごとに1ファイルずつリスト表示する。システムに格納された大切なファイルを誤って上書きしてしまうことがないように、パッケージを解凍する際には細心の注意を払ってほしい。

```
$ tar tvzf package.tar.gz | less        gzipファイル用
$ tar tvjf package.tar.bz2 | less       bzip2ファイル用
```

2. 以下を実行すると、新しいディレクトリが作成され、そのディレクトリの中にファイルが解凍される。

```
$ mkdir newdir
$ cd newdir
$ tar xvzf package.tar.gz               gzipファイル用
$ tar xvjf package.tar.bz2              bzip2ファイル用
```

This PDF is prepared for tools@maruhouse.org, personal use only.

3. 解凍したファイルの中からファイル名がINSTALLあるいはREADMEファイルを参照する。これらのファイルには、そのソフトウェアのビルド方法などについて記載されている。以下、例を取り上げる。

```
$ cd newdir
$ less INSTALL
```

4. INSTALLあるいはREADMEファイルには、一時的に、カレントディレクトリで configure スクリプトを実行する、というように記載されている。それから指示に従って make、make installと実行する。なお、インストールするソフトウェアの動作を決定するために、configureスクリプトを実行する際にオプションを指定することも可能である。

```
$ ./configure --help
...  Then install the software:
$ ./configure options
$ make
$ su -l
Password: *******
# make install
```

4.2 ファイル操作の基本

ls	ディレクトリ中のファイルリストを表示
cp	ファイルをコピー
mv	ファイルのリネームあるいは移動
rm	ファイルの削除
ln	ファイルへのシンボリックリンクの作成

ファイル操作において実行することと言えば、例えば、ファイルをコピーする、ファイル名を変更する、ファイルを削除するなどがある。

ls [オプション] [ファイル]

coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

lsは、ファイルやディレクトリの属性などのリスト表示を行うコマンドである。カレントディレクトリでファイルのリスト表示を行うには、以下のように実行する。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ ls
```

ディレクトリの指定する。

```
$ ls dir1 dir2 dir3
```

ファイルの指定する。

```
$ ls file1 file2 file3
```

最も重要なオプションは-aと-lである。lsコマンドは、デフォルトでは、ドットで始まるファイル(ドットファイル)を表示しない。-aオプションを指定して実行するとドットファイルを含めた全ファイルを表示する。-lオプションにより、連続したリスト表示を行う。

```
-rw-r--r--    1 smith users       149 Oct 28  2002 my.data
```

これは、右の順で、ファイルパーミッション(-rw-r--r--)、所有者(smith)、グループ(users)、ファイルサイズ(149 bytes)、最終更新日(Oct 28 2002)、ファイル名を意味する。なお、パーミッションに関する詳細は、「2.4 ファイル保護」を参照のこと。

オプション

-a	ドットファイルを含めた全てのファイルリストを表示
-l	ファイル属性が含まれたリストを1行ごとに連続して表示する。なお、-hオプションにより、ファイルサイズをバイト表示する代わりにメガバイト、ギガバイトで表示
-F	ファイル形式を示す記号をファイル名に追加することで、詳細な情報を提供する。なお、/はディレクトリ、*は実行可能形式、@はシンボリックリンク、|は名前付パイプ、=はソケットを表す。これらの記号はファイル名の一部ではなく、あくまでも視覚的に示すための記号である
-i	ファイルのinode番号を出力
-s	ファイルのブロックサイズを出力する。サイズごとにソートする場合、非常に有効 `$ ls -s | sort -n`
-R	ディレクトリを再帰的に表示
-d	ディレクトリ内部を出力せずに、ディレクトリ自身のみを表示

This PDF is prepared for tools@maruhouse.org, personal use only.

cp [オプション] 元のファイル(ファイル|ディレクトリ) coreutils

/bin stdin stdout - file -- opt --help --version

cp は、ファイルのコピーを行うコマンドである。

```
$ cp file file2
```

あるいは、複数のファイルをディレクトリにコピーする。

```
$ cp file1 file2 file3 file4 dir
```

ディレクトリを再帰的にコピーする場合、-a あるいは -R オプションを指定する。

オプション

-p コピー対象のファイルに対して、コピーを実行する権限を持つ。すなわち、ファイルの所有者あるいはグループであれば、ファイル内容だけでなく、ファイルパーミッション、タイムスタンプについてもコピーが行われる(通常、コピーを行うことでファイルの所有者が自分に変更され、タイムスタンプが更新される。また、パーミッションは、元の所有者のパーミッションにセットされる)

-a 特別なファイル、パーミッション、シンボリックリンク、ハードリンク全ての関係を保持した状態で、ディレクトリ階層のコピーを再帰的に実行する。これは、以下のオプションを組み合わせて実行する場合と同一である。-R(特別なファイルを含めて再帰的にコピーを行う)、-p(パーミッション)、-d(リンク)

-i インタラクティブモードでコピーを行う。これにより、ファイルが既に存在していた場合、上書きしてよいか、確認できる

-f 強制的にコピーを行う。送り先にファイルが存在していた場合でも、無条件で上書きを実行

mv [オプション] 元のファイル 移動先のファイル|ディレクトリ coreutils

/bin stdin stdout - file -- opt --help --version

mv(move)は、ファイル名の変更を行うコマンドである。

```
$ mv file1 file2
```

あるいは、送り先ディレクトリへファイルの移動を行う。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ mv file1 file2 dir3 dir4 destination_directory
```

オプション

-i　インタラクティブモードで移動を行う。これにより、ファイルが既に存在していた場合、上書きしてよいか、確認できる

-f　強制的に移動を行う。送り先にファイルが存在していた場合でも、無条件で上書きを実行

rm [オプション] ファイル|ディレクトリ　coreutils

/bin　stdin　stdout　- file　-- opt　--hel　--version

rm(remove)は、ファイルの削除を行うコマンドである。

```
$ rm file1 file2 file3
```

あるいは、ディレクトリの削除を再帰的に実行する。

```
$ rm -r dir1 dir2
```

オプション

-i　インタラクティブモードで移動を行う。これにより、個々のファイルを削除してよいか、確認しながら削除する

-f　強制的に削除を行う。エラーあるいは警告を無視して実行

-r　ディレクトリとその内容について再帰的に削除を行う。なお、-fオプションと組み合わせて使用する場合には、注意して実行すること

ln [オプション] 元のファイル 移動先のファイル|ディレクトリ　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

リンク(link)とは、lnコマンドによって作成される特別なファイルである。これは、ファイルから別のファイルへの参照ポインタのようなものである。なお、Fedora Linuxには、2種類のリンクファイルが存在する。シンボリックリンク(symbolic link)は、Windowsのショートカットあるいは Macintoshのエイリアス機能のようなパスであり、まさに別のファイルへの参照を示すリンクである。

```
$ ln -s myfile softlink
```

This PDF is prepared for tools@maruhouse.org, personal use only.

しかし、リンク元となるファイルが削除された場合、そのファイルへのパスがなくなるため、シンボリックリンクは無効となる。一方、ハードリンク(hard link)を利用すると、ディスク上で2つめのファイルを持つことになるには同一のinode番号であるため、元のファイルを削除した場合でも、ハードリンクが無効になるということはない。

```
$ ln myfile hardlink
```

シンボリックリンクは、ディスクパーティションをまたいでリンクを作成できる。これは、どのようなファイルに対してもファイルへのパスを作成できることを意味する。一方、ハードリンクはディスクパーティションをまたいだリンクを作成することはできない。これは、ディスク上の同一のinode番号を持つため、別のディスク上ではそのinode番号が意味を持たないためである。なお、シンボリックリンクは、ディレクトリへのリンクについても作成できるが、ハードリンクの場合は、それができない。ただし、これを実行するにはスーパーユーザに変更してから、-dオプションを使用する必要がある。

オプション

-s	シンボリックリンクの作成。デフォルトではハードリンクが作成される
-i	インタラクティブモードで行う。これにより、ファイルが既に存在していた場合、上書きしてよいか、確認を行う
-f	強制的にリンクを作成する。送り先にファイルが存在していた場合でも、無条件で上書きを実行
-d	ディレクトリ参照用のハードリンクを作成(スーパーユーザのみ)

シンボリックリンクがどのディレクトリを参照しているかを確認するための手法を説明しておく。

```
$ readlink linkname
$ ls -l linkname
```

4.3 ディレクトリ操作

cd	カレントディレクトリの変更
pwd	カレントディレクトリのパスを表示。すなわち、ファイルシステム上で「私は今どのディレクトリにいるのか？」を知らせる

This PDF is prepared for tools@maruhouse.org, personal use only.

basename	ファイルパスの最後の箇所のみ出力
dirname	ファイルパスの最後の箇所を取り除き出力
mkdir	ディレクトリを作成
rmdir	空のディレクトリを削除
rm -r	ディレクトリとその内容全てを削除

既に、「2章　ファイルシステム」にてLinuxのディレクトリ構造について解説済みである。本節では、ディレクトリの作成、変更、削除、操作に関するコマンドについて取り上げる。

cd [ディレクトリ]　bash

shell built-in　stdin　stdout　- file　-- opt　--help　--version

cd（change directory）により、ワーキングディレクトリの移動を行うコマンドである。引数としてディレクトリが指定されていない場合には、デフォルトで自分のホームディレクトリに移動する。

pwd　bash

shell built-in　stdin　**stdout**　- file　-- opt　--help　--version

pwd は、現在のワーキングディレクトリの絶対パスを出力するコマンドである。

```
$ pwd
/users/smith/mydir
```

basename パス　coreutils

/bin　stdin　**stdout**　- file　-- opt　**--help**　**--version**

basename は、ファイルパス中の最後の箇所のみを出力するコマンドである。

```
$ basename /users/smith/mydir
mydir
```

This PDF is prepared for tools@maruhouse.org, personal use only.

dirname パス

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

dirname は、ファイルパス中の最後の箇所を取り除いて出力するコマンドである。

```
$ dirname /users/smith/mydir
/users/smith
```

このコマンドはディレクトリ名が示す文字列を簡単に操作できる。

mkdir [オプション] ディレクトリ

coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

mkdir は、１つ、あるいは複数のディレクトリ作成を行うコマンドである。

```
$ mkdir d1 d2 d3
```

オプション

-p　ディレクトリパスを指定した場合には、自動的にその親ディレクトリも作成される。mkdir -p /one/two/three を実行すると、親ディレクトリである /one と /one/two が存在していない場合でも、/one/two/three ディレクトリが作成される

-m *mode*　指定したパーミッション権限を持つディレクトリを作成

```
$ mkdir 0755 mydir
```

デフォルトでは、シェルがパーミッションのモードを決定する。詳細は、「4.6 高度なファイル操作」および「2.4　ファイル保護」を参照のこと

rmdir [オプション] ディレクトリ

coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

rmdir（remove directory）は、指定した１つあるいは複数の空のディレクトリを削除するコマンドである。空でないディレクトリとその中身を削除する場合には、以下のようにオプションを指定して使用する。rm -r *directory* インタラクティブにディレクトリ削除を行う場合には、rm -ri *directory* を実行。

エラーメッセージや確認メッセージ無しで強制的にディレクトリ削除を行う場合には、rm -rf *directory* を実行。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-p　ディレクトリパス(単なるディレクトリ名ではないことに注意)を指定した場合、指定されたディレクトリだけでなく、自動的にその親ディレクトリも削除される。このとき、その親ディレクトリの全てが空でなければならない。すなわち、rmdir -p /one/two/three は、/one/two/three ディレクトリが削除されるだけでなく /one/two と /one の双方のディレクトリが削除される

4.4 ファイル表示

cat	ファイル内容の表示
less	ページごとにファイルを表示
head	ファイルの先頭部分の表示
tail	ファイルの最終部分の表示
nl	行番号を付けてファイルを出力
od	ファイルを 8 進数(または他の形式)で出力
xxd	ファイルを 16 進数で出力
gv	Postscript あるいは PDF ファイルの表示
xdvi	TeX DVI ファイルの表示

Linuxを利用していると、様々なファイル形式、例えばテキストファイル、Postscript、バイナリデータなどを扱う機会が増える。本節では、これらのファイルを表示するための方法を説明する。なお、画像ファイルを表示するコマンドについては、「9.5　画像とスクリーンセーバ」、音楽・音声ファイルの再生については「9.6　音と動画」を参照のこと。

cat [オプション] [ファイル]　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

catは、最も簡単な表示コマンドである。このコマンドは、あるファイルを標準出力する。なお、ファイルサイズが大きい場合には、1 画面に表示できずそのままスクロールアウトしてしまう。このような場合、lessコマンドを使用するとよい。catコマンドは、ファイルそのものをシェルのパイプ(|)に渡す場合に大変便利である。また、オプションを指定することで、ファイルに対して行番号を追加(この目的では nl コマンドの方がより優れた機能を持つ)、空白の削除といったことも可能である。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-T	タブを ^I と表示
-E	改行を $ と表示
-v	それ以外の制御文字を ^ と表示
-n	行番号を追加して表示
-b	空白でない行に行番号を追加して表示
-s	連続した空白行を、1 行の空白行にまとめて表示

less [オプション] [ファイル]

less

/usr/bin stdin stdout - file -- opt --help --version

1ページ(1画面)ごとにテキストを表示したい場合には、lessコマンドを使用する。lessは、何ページにもわたるテキストファイルを参照する場合や、パイプでつないだときの最終オプションとして利用すると、大変便利である。

```
$ command1 | command2 | command3 | command4 | less
```

lessの実行中に、使用法に関するヘルプメッセージを表示したい場合、hキーを入力する。ここでは、いくつか役立つ方法を取り上げておく。

キー入力	意味
h、H	ヘルプメッセージの参照
スペースキー、f、^V、^F	1 ページ(1 画面)先へ移動
Enter	1 行先へ移動
b、^B、ESC-b	1 ページ(1 画面)前に移動
/	検索したい文字列を入力し、Enter キーを押すと、less は検索文字列とマッチした最初の行を表示
?	/ と同等であるが、前に戻り文字列検索(前検索)を実行
n	最後に検索した文字列の検索を繰り返し実行
N	最後に検索した文字列の前検索を繰り返し実行
v	デフォルトのテキストエディタ(環境変数 VISUAL で定義されたエディタ、もしこれが存在しなければEDITOR変数で定義されたエディタ、いずれも存在しなければvi) を利用してファイル編集を行う
<	ファイルの先頭に移動
>	ファイルの最後に移動

This PDF is prepared for tools@maruhouse.org, personal use only.

キー入力	意味
:n	次のファイルへ移動
:p	前のファイルへ移動

lessには驚くほど多くの機能が用意されているため、本書ではその中で最もよく利用される機能のみを取り上げる。もちろん、マニュアルページ(manpage)を参照することも有益である。

オプション

-c　次ページを表示する前に画面をクリア

-m　ファイルの詳細なパーセント表示を出力

-N　画面の各行の先頭に行番号を追加

-r　制御文字を文字通りそのまま表示する。なお、less は制御文字を可読なフォーマットに変換して表示を実行

-s　連続した空白行を 1 行の空白行にまとめる

-S　画面幅よりも長い行を折り返さず切り捨てて表示

head [オプション] [ファイル]　coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

headは、ファイルの最初から10行目までを出力するコマンドである。これは、ファイルの概要のみ知りたい場合に、有益なコマンドとなる。

```
$ head myfile
$ head * | less          カレントディレクトリの全ファイルのプレビューを実行
```

オプション

-*N*
-n *N*　ファイルの先頭から *N* 行目までを出力

-c *N*　ファイルの *N* バイトまでを出力

-q　サイレントモードで実行。複数ファイルを処理している際に、各ファイルのバナー表示を行わない。デフォルトの場合、headはファイル名のバナーを出力

This PDF is prepared for tools@maruhouse.org, personal use only.

tail [オプション] [ファイル]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

tail は、ファイルの最後の 10 行を出力するコマンドである。

```
$ tail myfile
```

オプション

-N
-n *N*　ファイルの最後から *N* 行目までを出力

+*N*　最初の *N* 行を除き、全ての行を出力

-c *N*　最後の *N* バイトを出力

-f　ファイルをオープンした状態で、新たに行が追加された場合、その行を出力する。この機能は非常に有益である。実行する前にファイルが存在していない、またファイルが存在する(作成される)まで待機させる場合--retryオプションの使用を推奨する

-q　サイレントモードで実行。複数のファイルを処理している際に、各ファイルのバナー表示を行わない。デフォルトの場合、tailはファイル名のバナーを出力

nl [オプション] [ファイル]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--versionnl

nl は、行番号を付けてファイルの内容を出力するコマンドである。nl コマンドは、cat コマンドに -n と -b オプションを付けて実行する場合と比べて、ナンバリングの方法を変更できるため、より柔軟な操作が行える。さらに、通常のテキストファイル、あるいは、あらかじめ定義済みのヘッダやフッタを持つマークアップテキストファイルに対しても実行できる。

オプション

-b [a|t|n|p*R*]　全ての行(a)、空白の無い行(t)、行無し(n)、あるいは、正規表現が含まれた行のみ *R* に対して番号を追加(デフォルトでは a)

-v *N*　整数 *N* 番目からナンバリングを開始(デフォルトでは 1)

-i *N*　行番号のナンバリングを *N* 番ごとに行う。奇数番号のみ(-i2)あるいは偶数番号のみ(-v2 -i2)カウントするような設定が可能(デフォルトでは 1)

This PDF is prepared for tools@maruhouse.org, personal use only.

-n [ln|rn|rz] 行番号の出力フォーマットを指定する。左詰めにし、先頭の0埋めをしない(ln)、右詰めにし、先頭の0埋めをしない(rn)、右詰めにして、先頭は0で埋める(rz)というような指定が可能(デフォルトではln)

-w*N* 行番号表示に用いる文字数を*N*に設定(デフォルトでは6)

-s *S* 出力ファイルの行番号とテキスト行の区切りに*S*を使用(デフォルトではTAB)

またテキストファイルを仮想ページに分割することができる。このページにはそれぞれヘッダ、本文(ボディ)、フッタを持ち、異なったナンバリング機能をつけることができる。しかし、これを実行するには、ファイルにnlコマンド特有のデリミタ文字を挿入する必要がある。これには\:\:\:(ヘッダ開始)、\:\:(ボディ開始)、\:(フッタ開始)がある。デリミタ文字は、行の先頭に表示され、ファイルのヘッダやフッタの行番号を変更するために追加オプションを使用できる(詳細は、マニュアルを参照こと)。

od [オプション] [ファイル] coreutils

/usr/bin stdin stdout - file -- opt --help --version

バイナリファイルを表示する際には、od(octal dump)コマンドを使用する。このコマンドは、指定したファイルから、標準出力へ書き出しを行うが、そのデータをASCII、8進数、10進数、16進数、浮動小数点、といったように様々な形式(byte、short、long)で表示することができる。このコマンドの一例を取り上げる。

```
$ od -w8 /usr/bin/who
0000000 042577 043114 000401 000001
0000010 000000 000000 000000 000000
0000020 000002 000003 000001 000000
0000030 106240 004004 000064 000000
…
```

この例では、バイナリファイル/usr/bin/whoを1行に8バイト毎に、8進数で表示している。左のカラムに入力ファイルでのオフセットが書き出され、ファイルのデータのグループが続けられる。なお、オフセットを8進数で、ファイルのデータのグループは2バイトずつ8進数で表示される。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-N *B*
: ファイルごとに最初の *B* バイトまでを、指定した 10 進数、16 進数(0xあるいは0Xを追加)、512バイトブロック(bを追加)、キロバイト(kを追加)、メガバイト(mを追加)にて表示(デフォルトでは全体のファイルが表示)

-j *B*
: ファイルごとに *B*+1 バイトを出力して開始する。整形されたフォーマットは、-N オプションを使用した場合と同じ(デフォルトでは 0)

-w [*B*]
: 1行あたり *B* バイトの入力を表示する。整形されたフォーマットは、-Nオプションを使用した場合と同じ。-wオプションをそのまま使用することは、-w32 と等価(デフォルトでは 16)

-s [*B*]
: それぞれバイト列を空白で区切り、連続した *B* バイトずつにグループ化する。整形されたフォーマットは、-Nオプションを使用した場合と同じ。-sオプションをそのまま使用することは、-s3と等価(デフォルトでは 2)

-A (d|o|x|n)
: 最も左のカラムに表示されるオフセットの基数を、10進数(d)、8進数(o)、16 進数(h)、なし(n)から選択(デフォルトでは o)

-t (a|c)[z]
: エスケープシーケンス文字(a)あるいは文字の名前(c)として出力された英数文字によりその出力を表示。zについては、以下参照のこと

-t (d|o|u|x)[SIZE[z]]
: 整数フォーマット、すなわち、8進数(o)、符号付き10進数(d)、符号無し10進数(u)、16進数(x)にて出力を表示(バイナリ出力においては、このコマンドの代わりにxxdコマンドを使用のこと)。*SIZE*は、整数ごとの何バイト分かを示しており、それは正整数あるいはC,S,I,Lのいずれかであり、それぞれchar型、short型、int型、long型のサイズを表す。z については、以下を参照のこと

-t f[SIZE[z]]
: 浮動小数点数にて出力を表示。*SIZE*は、整数ごとの何バイト分かを示している。これは、正整数あるいはF、D、Lのいずれかであり、それぞれfloat型、double型、long型のサイズを表す。zについては、以下を参照のこと。-tが除かれた場合には、デフォルトでは-to2となる。-tパラメータにzを追加することにより、右側に新たなカラムが出力され、xxdコマンドのデフォルト出力と類似したような出力可能な文字が表示される

xxd [オプション] [ファイル]

vim-common

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

xxdは、いくつか異なるフォーマットでファイルの16進数表示やバイナリダンプ表示を行うコマンドである。さらに、xxdは、16進数ダンプのフォーマットから元のデータへの変換

This PDF is prepared for tools@maruhouse.org, personal use only.

も行うことが出来る。使用例について以下に示す。

```
$ xxd /usr/bin/who
0000000: 7f45 4c46 0101 0100 0000 0000 0000 0000 .ELF............
0000010: 0200 0300 0100 0000 a08c 0408 3400 0000 ............4...
0000020: 6824 0000 0000 0000 3400 2000 0600 2800 h$......4. ...(.
0000030: 1900 1800 0600 0000 3400 0000 3480 0408 ........4...4...
...
```

この例では、バイナリファイル /usr/bin/who の 16 進数表示であり、列ごとに 16 バイトずつ表示されている。左のカラムに入力ファイルでのオフセット、以下の8個のカラムにはデータが書き出される。最後のカラムは、可能であればその列の出力可能な文字の表示を行う。xxd は、デフォルトで3カラム、ファイルオフセット、データの16進数表記、データのテキスト表記(出力可能な文字のみ)の出力を行う。

オプション

-l *N*	最初の *N* バイトのみ表示(デフォルトでは、ファイル全体の表示)
-s *N* -s -*N*	ファイルの先頭バイト以外の場所から開始。すなわち、先頭 *N* バイトをスキップして開始される。次に、(-*N*)はファイルの最後*N*バイト目から開始することを示す(これ以外にもよりテクニカルなスキップ方法として+*N*という表記も存在する。詳細はマニュアルを参照のこと)
-c *N*	列ごとに *N* バイトずつ表示(デフォルトでは 16)
-g *N*	od -s と同様に、それぞれの列を空白で区切られた連続した *N* バイトごとにグループ化して実行(デフォルトでは 2)
-b	16 進数の代わりにバイナリで出力
-u	小文字 16 進数表記の変わりに大文字 16 進数表記で出力
-p	1 行ごとに 60 バイト連続した 16 進数ダンプとして出力
-i	C言語のデータ構造として出力。コマンドがファイルから読み出しを行う際に、データが含まれた unsigned char 型配列、および配列長が含まれた unsigned int 型配列を生成する。コマンドが標準入力から読み出しを行う際には、16 バイトごとにコンマで区切られたリストのみ出力
-r	リバース操作：xxd16 進数ダンプから元のファイルフォーマットへの変換を実行。-p オプションを追加していた場合には、デフォルトの 16 進数表記のフォーマットで出力が行われる。操作を理解する上で、標準出力に対して元のファイルを再生成するためにパイプラインを使用し、ファイルの変換とその逆変換を行うために、これらのコマンドを実行してみてほしい

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ xxd myfile | xxd -r
$ xxd -p myfile | xxd -r -p
```

gv [オプション] ファイル

gv

/usr/X11R6/bin　stdin　stdout　- file　-- opt　--help　--version

GhostView は、X Window 上で Adobe Postscript や PDF ファイルを表示するためのプログラムである。これを実行するには、gv あるいは ghostview と指定すればよい。このプログラムの基本的な操作方法は大変シンプルであり、単に表示したいページ番号をクリックするだけでよい。少しの訓練で、すぐに使いこなせるようになるだろう。GhostView は、Linux の優秀な Postscript ビューワであるが、それ以外にもフリーの PDF ビューワとして acroread（http://www.adobe.com/）や xpdf（http://www.foolabs.com/xpdf/）がある。

オプション

-page *P*　ページ番号 *P* から開始（デフォルトでは 1）

-monochrome
-grayscale
-color　指定した表示モードを使用

-portrait
-landscape
-seascape
-upsidedown　ページオリエンテーションの選択、通常 gv 自動的に選択

-scale *N*　表示サイズ（拡大）の設定を行う。*N* の値として、拡大する場合には正の値を指定し、縮小する場合には負の値を指定

-watch
-nowatch　変更が行われた際には、自動的に Postscript ファイルのリロード（再読み込み）

xdvi [オプション] ファイル

tetex-xdvi

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

ドキュメント生成システムである TeX は、拡張子が .dvi である dvi ファイルと呼ばれるバイナリファイルを出力する。そのビューワである xdvi は、X Window 上で dvi ファイルの表示を行う。もちろん、dvips コマンドを使用して DVI ファイルから Postscirpt ファイルへの変換をしてから、GhostView（gv）で表示するという方法もよいだろう。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ dvips -o myfile.ps myfile.dvi
$ gv myfile.ps
```

xdviは次ページへの移動を示すNextボタンなどいくつかのボタンが右側に配置されている(-expertオプションを指定することにより、これらのボタンを隠すことも可能)。さらに、キー入力によってファイル操作を行うことも可能である。

キー入力	意味
q	中止
n、スペースキー、リターン(Enter)キー、Pagedown キー	次ページへ移動。*N*ページ先に移動するには、番号*N*を追加すればよい
p、Backspaceキー、Delete キー、Pageup キー	前ページへ移動。*N*ページ前に移動するには、番号*N*を追加すればよい
<	先頭ページへ移動
>	最終ページへ移動
^L	ページを再表示
R	dvi ファイル修正後に、ファイルの再読み込みを実行
マウスボタン	マウスカーソルで選択した範囲の拡大

xdvi は、色、配置、ズームなど、様々な設定が行える機能を有しているツールである。

4.5 ファイルの作成と編集

emacs	Free Software Foundation 提供のテキストエディタ
vim	vi エディタの拡張版
umask	ファイルとディレクトリのモード設定
soffice	Microsoft Word、Excel、PowerPoint の編集を行う Office suite
abiword	Microsoft Word を編集
gnumeric	Excel ファイルを編集

Linuxを使いこなすには、最低限、1つのテキストエディタを習得してほしい。主なエディタとして、Free Software Foundationが提供するemacsや、UNIXでよく使われるエディタviの後継となるvimがある。両者ともオンラインチュートリアルが提供されているので、本書

This PDF is prepared for tools@maruhouse.org, personal use only.

では、これらの内容について全てを取り上げない。よく使用する操作方法を表1に掲載する。エディタを用いたファイル編集は、それぞれ以下を実行して行う。

```
$ emacs myfile
$ vim myfile
```

myfileファイルが存在しない場合、自動的にその名前のファイルが作成される。touchコマンドを用いて、空ファイルを作成することもできる（詳細は、「4.6　高度なファイル操作」を参照のこと）。

```
$ touch newfile
```

また、プログラムの出力をリダイレクトして、新しいファイルにデータを書き込むこともできる（詳細は、「3.2.7　入出力リダイレクト」を参照のこと）。

```
$ echo anything at all > newfile
```

Microsoft Windows とファイル共有を行う場合は、Microsoft Word や Excel、PowerPoint のドキュメントを編集する Linux プログラムにも精通しておく必要があるだろう。

デフォルトのエディタ

Linuxのプログラムでは、いつでも必要な時に、エディタを起動することができる。デフォルトのエディタはvimであるが、例えば、利用するメールソフトで新しいメッセージを編集する際は、指定されたエディタが起動される。また、“v”と入力すると、lessが起動される。ここで、vim以外のプログラムをデフォルトのエディタに設定するにはどのようにすればよいだろうか？

これには、環境変数 VISUAL と EDITOR を以下のように設定する。

```
$ EDITOR=emacs
$ VISUAL=emacs
$ export EDITOR VISUAL                    オプション
```

プログラムによって参照する環境変数が異なるため、上記のように2つの環境変数を設定する必要がある。常に環境変数を設定しておく場合は、起動ファイルである~/.bash_profileにEDITORとVISUALを設定しておく。これにより、任意のプログラムをデフォルトエディタとして呼び出すことができる。

環境変数の設定の有無に関わらず、システム管理者は少なくとも vim と emacs の基本的な操作方法を習得しておく必要がある。これにより、システムツールが突然エディタを起動し、

This PDF is prepared for tools@maruhouse.org, personal use only.

システムの設定ファイルの編集を要求してきた場合などに備えることができる。

emacs [オプション] [ファイル]

emacs

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

emacsは、強力な編集環境が提供されている。想像できないほど多くのコマンドが提供されており、またユーザに固有の編集機能を定義するプログラム言語(emacs lisp)が組み込まれている。emacs チュートリアルを起動するには、

```
$ emacs
```

を実行し、^h t を入力する。

emacsのコマンドは、コントロールキー、あるいは通常Escキーや Altキーとなる metaキーを利用する。metaキーは、emacsのドキュメントではM- と表記されている(M-F は「メタキーを押したまま、F を押す」の意味)。基本的な操作方法については、 表1を参照のこと。

vim [オプション] [ファイル]

vim-enhanced

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

vimは、旧来からのUnix標準エディタである viの拡張バージョンである。vimチュートリアルを起動するには、以下を実行する。

```
$ vimtutor
```

vimは2つのモードを持ったエディタである。すなわち、挿入モードと通常モードのいずれかのモードを用いて文章を編集する。 通常モードは「行を消す」あるいは「コピー/貼り付け」というコマンド実行で用いられる。挿入モードは、テキストの入力を行う。通常モードでの基本的な操作方法については、表1を参照のこと。

表1　emacs と vim の基本的な操作方法

操作	emacs	vim
現在のウィンドウでエディタを起動	$ emacs -nw[ファイル]	$ vim [ファイル]
新しいX Windowでエディタを起動	$ emacs [ファイル]	$ gvim [ファイル]
文章を入力	文章	i 文章 ESC
保存と終了	^x^s の後 ^x^c	:wq

This PDF is prepared for tools@maruhouse.org, personal use only.

表 1　emacs と vim の基本的な操作方法（続き）

操作	emacs	vim
保存せずに終了	^x^c、バッファを保存するかと聞かれたとき“no”と入力	:q!
保存	^x^s	:w
ファイル名を指定して保存	^x^w	:w ファイル名
アンドゥ（元に戻す）	^_	u
エディタを一時停止（X を使用していない場合）	^z	^z
編集モードに切り替え	（何もしない）	ESC
コマンドモードに切り替え	M-x	:
操作をキャンセル	^g	ESC
1 文字進む	^f あるいは右カーソルキー	l あるいは右カーソルキー
1 文字戻る	^b あるいは左カーソルキー	h あるいは左カーソルキー
上に移動	^p あるいは上カーソルキー	k あるいは上カーソルキー
下に移動	^n あるいは下カーソルキー	j あるいは下カーソルキー
1 単語進む	M-f	w
1 単語戻る	M-b	b
行の先頭に移動	^a	0
行の最後に移動	^e	$
1 画面進む	^v	^f
1 画面戻る	M-v	^b
バッファの先頭へ	M-<	gg
バッファの末尾	M->	G
カーソルの右側の文字を消去	^d	x
カーソルの左側の文字を消去	バックスペース	X
カーソルの単語を削除	M-d	de
カーソルの左側の単語を削除	M- バックスペース	db
現在の行を削除	^a^k^k	dd
カーソル位置から行末までを削除	^k	d$
リージョンの定義（リージョンの先頭をマークするためにこのキーをタイプし、その後カーソルを移動してリージョンの最後を指定する）	^ スペースキー	v
リージョンの切り取り	^w	d
リージョンのコピー	M-w	y
リージョンの貼り付け	^y	p

This PDF is prepared for tools@maruhouse.org, personal use only.

表1 emacs と vim の基本的な操作方法(続き)

操作	emacs	vim
ヘルプの取得	^h	:help
ユーザマニュアルを取得	^h i	:help

umask [オプション] [ファイル] bash

shell built-in stdin stdout - file -- opt --help --version

umaskは、ファイルやディレクトリの作成時に、デフォルトのモードを設定、あるいは表示を行うコマンドである。ここでいうモードとは、ユーザ(user)、グループ(group)、それ以外(other)に対して、それぞれ読み出し、書き込み、実行可能の権限を与えることである(詳細は「2.4 ファイル保護」を参照のこと)。

```
$ umask
0002
$ umask -S
u=rwx,g=rwx,o=rx
```

まず、技術的な説明をしよう。umaskが使用する変数は「マスク」である。すなわち、ファイルに対しては0666、ディレクトリに対しては0777を基準として(2進表記の)XOR操作を行い、デフォルトの保護モードを指定する。例えば0002 XOR 0666は、0664となり、ファイルの保護モード、0002 XOR 0777は0775となり、これがディレクトリの保護モードとなる。

この説明が難しいと感じるのであれば、以下に簡単な例を取り上げるので参考にしてほしい。以下の例は、umaskを0022に設定することで、あなた自身に全ての権限を与え、その他のユーザにはファイル読み出し権限とディレクトリの読みだし/実行権限のみを与える場合である。

```
$ umask 0022
$ touch newfile && mkdir dir
$ ls -ld newfile dir
-rw-r--r--    1 smith smith        0 Nov 11 12:25 newfile
drwxr-xr-x    2 smith smith     4096 Nov 11 12:25 dir
```

以下の例は、umaskを0002に設定し、あなた自身とあなたが属するグループに全ての権限を与え、その他のユーザにはファイルの読み出し権限とディレクトリの読みだし/実行権限のみを与える。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ umask 0002
$ touch newfile && mkdir dir
$ ls -ld newfile dir
-rw-rw-r--    1 smith smith        0 Nov 11 12:26 newfile
drwxrwxr-x    2 smith smith     4096 Nov 11 12:26 dir
```

最後の例は、umaskを0077に設定し、あなた自身にすべての権限を与え、その他のユーザには何も権限を与えない。

```
$ umask 0077
$ touch newfile && mkdir dir
$ ls -ld newfile dir
-rw-------    1 smith smith        0 Nov 11 12:27 newfile
drwx------    2 smith smith     4096 Nov 11 12:27 dir
```

soffice [ファイル]　openoffice.org

/usr/lib/openoffice/programs　stdin　stdout　- file　-- opt　--help　--version

OpenOffice.org は、Microsoft の Word、Excel、PowerPoint のファイルを編集できる便利な統合ソフトウェアである。起動は単に、

```
$ soffice
```

と実行するだけである。

1つのプログラムで3種類のファイルが編集できる。なお、OpenOffice.orgの実行には、非常に多くのメモリとディスク容量を使用する。

OpenOffice.org は描画（sdrawコマンド）、FAX（sfaxコマンド）、メールラベル（slabelコマンド）などが扱える。詳細は、http://www.openoffice.org/、sofficeのヘルプメニューを参照のこと。

abiword [オプション] [ファイル]　abiword

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

abiword は、Micrrosoft の Word 文章を編集するプログラムである。このプログラムは、Openoffice.orgほど強力ではなく、複雑な編集作業には適さないが、sofficeよりも軽くて高速である。引き数としてファイルを指定する場合、そのファイルが存在している必要がある。すなわち、abiword は新規にファイル作成を行わない。

This PDF is prepared for tools@maruhouse.org, personal use only.

gnumeric [オプション] [ファイル] gnumeric

/usr/bin stdin stdout - file -- opt --help --version

gnumeric は、Microsoft の Excel 文章を編集できる表計算ソフトである。gnumeric は、大変強力で高速に動作するため、Excelを使ったことがあるユーザはすぐに操作に慣れるだろう。引き数でファイルを指定する場合、そのファイルは存在している必要がある。すなわち、gnumeric は、abiword と同様、新規にファイルを作成しない。

4.6 高度なファイル操作

stat	ファイルとディレクトリの属性を表示
wc	ファイル内のバイト、単語、行をカウント
du	ファイルとディレクトリのディスク使用量の表示
file	ファイルのタイプを見分ける（あるいは推測）
touch	ファイルとディレクトリのタイムスタンプを変更
chown	ファイルとディレクトリの所有者を変更
chgrp	ファイルとディレクトリのグループの所属を変更
chmod	ファイルとディレクトリの保護モードを変更
chattr	ファイルとディレクトリの拡張された属性を変更
lsattr	ファイルとディレクトリの拡張された属性を表示

Linux でファイルを調査する場合、ファイルの中身を見ただけでは、半分しか調べたことにならない。全てのファイルとディレクトリは、所有者、サイズ、アクセス権限、その他の属性情報を持つ。ls -lコマンドは、これらの属性のいくつかを表示する（詳細は、「4.2ファイル操作の基本」を参照のこと）が、ここで紹介するコマンドは、さらに詳細な情報を提供する。

stat [オプション] [ファイル] coreutils

/usr/bin stdin stdout - file -- opt --help --version

statコマンドは、ファイルやファイルシステム（-fオプションの場合）の重要な属性を表示する。

ファイルの情報は、

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ stat myfile
  File: "myfile"
  Size: 1264       Blocks: 8         Regular File
Access: (0644/-rw-r--r--)         Uid: (  600/smith)  Gid: (  620/users)
Device: 30a        Inode: 99492      Links: 1
Access: Fri Aug 29 00:16:12 2003
Modify: Wed Jul 23 23:09:41 2003
Change: Wed Jul 23 23:11:48 2003
```

というように、ファイル名、バイト単位のサイズ(1264)、ブロック単位のサイズ(8)、ファイル形式(Regular File)、8進数表記によるパーミッション(0644)、ls -l形式によるパーミッション(-rw-r--r--)、オーナーのユーザID (600)、オーナーの名前(smith)、オーナーのグループID (620)、オーナーのグループ名(users)、デバイスのタイプ(30a)、iノード番号(99492)、ハードリンクの数(1)、そしてアクセス、修正、ステータス変更に関するファイルのタイムスタンプが表示される。

ファイルシステムの情報は、

```
$ stat -f myfile
  File: "myfile"
    ID: bffff358 ffffffff Namelen: 255     Type: EXT2
Blocks: Total: 2016068    Free: 876122     Available: 773709     Size: 4096
Inodes: Total: 1026144    Free: 912372
```

というように、ファイル名(myfile)、ファイルシステムID (bffff358 ffffffff)、ファイルシステムが許容するファイル名の限度(255 bytes)、ファイルシステムのタイプ(EXT2)、ファイルシステムの全容量、残りの容量、利用可能な容量(ブロックサイズ表記になっており、それぞれ2016068, 876122, 773709)、ファイルシステム(今回の例ではext2)のブロックサイズ(4096)、そして全てのiノードと残りのiノードの数(それぞれ1026144と912372)を表示する。

-tオプションは、上記と同じ内容を、ヘッダを表示せず、全ての情報を1行が表示される。これは、シェルスクリプトやその他のプログラムを用いて処理する場合に便利である。

```
$ stat -t myfile
myfile 1264 8 81a4 500 500 30a 99492 1 44 1e 1062130572 1059016181 1059016308
$ stat -tf myfile
myfile bffff358 ffffffff 255 ef53 2016068 875984 773571 4096 1026144 912372
```

オプション

-l	シンボリックをたどり、実体ファイルを通知
-f	ファイル自身ではなく、ファイルが含まれたファイルシステムの通知
-t	簡潔(terse)なモード。1行で全情報を表示

This PDF is prepared for tools@maruhouse.org, personal use only.

wc [オプション] [ファイル]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

wc(word count)は、テキストファイルのバイト数、単語数、行数を表示するプログラムである。バイナリファイルもテキストファイルとして扱う。

```
$ wc myfile
   24      62     428 myfile
```

以下の例では、ファイル内のテキストが24行、空白で区切られた62個の単語、428バイトであることを示す。

オプション

-l	行数のみを表示
-w	単語数のみ表示
-c	バイト(文字)数のみ表示
-L	各ファイルの最も長い行を示し、バイト単位でその長さを表示

du [オプション] [ファイル|ディレクトリ]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

du(disk usage)は、ファイルやディレクトリが使用しているディスクスペースを計算して表示するコマンドである。デフォルトでは、カレントディレクトリと、以下にある全てのサブディレクトリを対象として、それぞれのディレクトリについて、ブロック単位の容量を表示する。最後の行では、全てのディレクトリを合計した容量を表示する。

```
$ du
8    ./Notes
36   ./Mail
340  ./Files/mine
40   ./Files/bob
416  ./Files
216  ./PC
2404 .
```

また、複数のファイルを指定することも可能である。

```
$ du myfile myfile2
4    ./myfile
16   ./myfile2
```

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-b -k -m　バイト単位(-b)、キロバイト単位(-k)、メガバイト単位(-m)でサイズを計算

-B *N*　定義したブロック単位でサイズを表示。1ブロック＝Nバイト(デフォルトは1ブロック＝1024バイト)

-h -H　可読な形式で表示(human readable)。それぞれのサイズについて、もっとも適当な単位を選択。例えば、2つのディレクトリが、1ギガバイトと25キロバイトだった場合、du -hはそれぞれ1G、25Kと表示。-Hオプションは、1000の倍数を用いるが、-h オプションは1024の倍数を用いる

-c　最後の行で総計を表示。これはディレクトリを計算する際のデフォルトの振る舞いであるが、ファイルを計算する時に総計が必要な場合は -c を用いる

-L　シンボリックリンクをたどって、実体のファイル容量を計算

-s　総容量のみを表示

file [オプション] [ファイル]　file

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

file は、ファイルの種類を表示するコマンド。

```
$ file /etc/hosts /usr/bin/who letter.doc
/etc/hosts:    ASCII text
/usr/bin/who:  ELF 32-bit LSB executable, Intel 80386 ...
letter.doc:    Microsoft Office Document
```

とある有名なオペレーティングシステムとは異なり、Linux の場合ファイルの種類はファイル拡張子の名前からではなく、ファイルの内容などからも推測を行う。

オプション

-b　ファイル名を省略

-i　ファイルに関する情報を MIME 形式で出力(例：text/plain や audio/mpeg など)

-f *name_file*　*name_file* を読み込み検査する。*name_file* には、通常の file コマンドで検査するファイル名を、1行につき1つ明記する。*name_file* の後に、通常の場合と同様にファイル名を指定することもできる。この場合は、指定したファイルの形式を順に表示

-L　シンボリックファイルをたどって、実体ファイルの形式を表示

This PDF is prepared for tools@maruhouse.org, personal use only.

-z　ファイルが圧縮されている場合、(詳細は、「4.11　ファイルの圧縮と解凍」を参照のこと)「圧縮されたデータ」と通知する代わりに、ファイルを解凍した後のファイルの種類を識別して表示

touch [オプション] [ファイル]　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

touchは、ファイルに関する2つのタイムスタンプ、修正時間(ファイルのデータが最後に変更された時間)とアクセス時間(ファイルが最後にアクセスされた時間)を変更するコマンドである。

```
$ touch myfile
```

タイムスタンプは自由に設定できる。例えば、

```
$ touch -d "November 18 1975" myfile
```

のようにユーザが任意の時刻を指定して実行できる。

引き数の myfile が存在しない場合、touch は空の(つまり、サイズ 0 の)ファイルを作成する。これは空ファイルを作成するのに最も簡単な方法である。

オプション

-a　アクセス時間のみを変更

-m　修正時間のみ変更

-c　ファイルが存在しないのであれば、ファイルを作成しない(通常、touchは空ファイルを作成する)

-d timestamp　ファイルのタイムスタンプを設定。タイムスタンプは、“12/28/2001 3pm”から“28-May”(年は現在の年が使用され、時間は0:00に設定される)、“next tuesday 13:59”(次の火曜日の13時59分)、“0”(今日の0:00)など、非常に多彩な形式で表現可能である。statを使ってファイルの現在の状況を確認してみよう。詳細は、info touchを参照のこと

-t timestamp　[[CC]YY]MMDDhhmm[.ss]という形式を使ってファイルのタイムスタンプを設定できるが、あまり多機能ではない。ここで、*CC* は 10 進数で表記された世紀表示、また *YY* は 10 進数表記の年、MM は 10 進数の月、DD は 10 進数の日、hh は 10 進数の時間、mm は 10 進数の分、ss は 10 進数の秒を表す。例えば、-t20030812150047 は 2003 年 8 月 12 日 15 時 00 分 47 秒を表す

This PDF is prepared for tools@maruhouse.org, personal use only.

chown [オプション] [所有者 ファイル]　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

chown（change owner）は、ファイルとディレクトリの所有者を設定するコマンドである。

```
$ chown smith myfile myfile2 mydir
```

「所有者」部分のパラメータは、以下のいずれかの方法が使用できる。

- 「ユーザ名」で、所有者を設定
- 「ユーザ名:グループ名」で、所有者とグループを設定
- 「ユーザ名:」で、所有者とユーザのログイングループに関連したグループを設定
- 「:グループ名」で、グループのみを設定
- 「--reference= ファイル名」で、指定されたファイル自身の所有者とグループを設定

ユーザ名とグループ名には、それぞれ数値表記のユーザ ID とグループ ID が指定できる。

オプション

--dereference　シンボリックファイルをたどって、実体のファイルを操作
-R　ディレクトリ階層内の所有権を再帰的に変更

chgrp [オプション] [属性 ファイル]　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

chgrp（change group）は、ファイルやディレクトリのグループの所有権を設定するコマンドである。

```
$ chgrp smith myfile myfile2 mydir
```

属性パラメータは、以下のいずれかの方法が使用できる。

- グループ名あるいは、数値表記のグループ ID
- 「--reference= ファイル名」で指定されたファイル内のグループと同じ設定

グループに関するより詳しい説明は、「6.4　グループの操作」を参照のこと。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

--dereference　シンボリックファイルをたどって、実体のファイルを操作
-R　ディレクトリ階層内の所有権を再帰的に変更

chmod [オプション] [パーミッション ファイル]　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

chmod（change mode）は、ファイルとディレクトリのアクセス権を設定するコマンドである。一般ユーザの場合、アクセスできるファイルは制限される。

chmodは、アクセス権も設定できる。ファイルには所有者（user）、グループ（group）、その他（other）が、それぞれパーミッション（属性）を持っており、通常よく利用されるパーミッションは、読み込み、書き込み、実行である。パーミッションの引き数には、以下の3つの形式が使用できる。

- 数値指定：ビットを8進数で表記することによりファイルの絶対的なパーミッションを指定。長さは最大4桁。1桁目は特別な属性権を表し（後述）、2桁目、3桁目、4桁目は、それぞれ、ファイルのオーナー、ファイルのグループ、その他のユーザを表す。例として、モード 0640 の場合を、図 3 に示す。
- 文字指定：絶対的あるいは相対的なパーミッションを示す、1つあるいはそれ以上の文字の組み合わせ。所収者と権限はカンマによって区切る。
- ファイル指定：「--reference=ファイル名」によって、指定したファイルの中身と同じパーミッションを設定する。

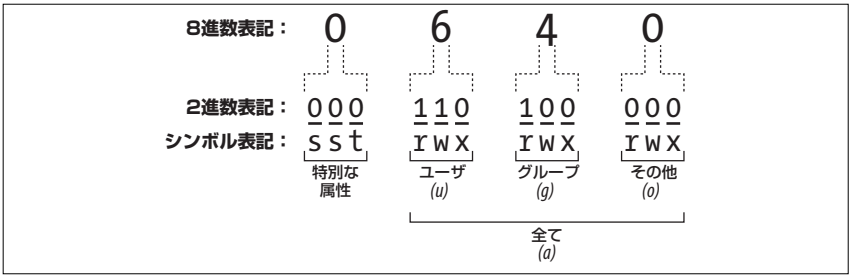

図 3　ビット操作によるファイルのパーミッション

This PDF is prepared for tools@maruhouse.org, personal use only.

2番目の形式の場合、それぞれの文字列は以下に示すスコープ、コマンド、パーミッションという3つの項目からなる。

スコープ(オプション)

uはユーザ、gはグループ、oはその他のユーザ、aは全てのユーザを表す。デフォルトはa。

コマンド

パーミッションを追加する場合には+、削除する場合には-、絶対表記で設定する場合には=を用いる。何も指定しなければ、無視される。

パーミッション

読み込みはr、書き込み/修正にはw、実行にはx(ディレクトリの場合、xはcdによる、これはそのディレクトリへの進入許可を意味する)。条件付実行(後述する)にはX、ユーザのパーミッションを複製するにはuを、グループのパーミッションを複製するにはgを、「その他のユーザ」のパーミッションを複製するにはoを指定する。setuidあるいはsetgidにはsを、スティッキービットにはtをそれぞれ用いる。

例えば、ug+rwはユーザとグループに読み込みと書き込みのパーミッションを与え、a-xは全てのユーザに対して実行権を削除し、u=rは存在する全てのパーミッションを削除し、その後、所有者によってのみ読み込み可能なファイルを作成する。ug+rw,a-xのように、カンマを用いることで、複数の指定が可能である。

setuidとsetgidは、実行可能なファイル(プログラムやスクリプト)に対して設定を行う。ここでは、ユーザ“smith”、グループ“friends”が所有している実行可能なファイルFがあると仮定する。ファイルFがsetuid(set user ID)のパーミッションを持っている場合、Fを実行するユーザは誰でもプログラムを実行する間は、ユーザsmithに「成りきる」ことができ、プログラムに関する全ての権限を持つ。同様に、Fがsetgid(set group ID)のパーミッションを持っていれば、Fを実行する誰もが、プログラムを実行する間は、メンバーに成りきることが可能となる。

以上から容易に想像できるように、setuidとsetgidはシステムの安全性に深刻な影響を与えかねないため、これについて十分に理解できない場合、これらのコマンドを使用しない方がよいだろう。例えばchmod +sを実行すると、システムがある攻撃に対し脆弱性を持つ可能性がある。条件付の実行権限(X)はファイルが既に実行可能、もしくはそのファイルがディレクトリという場合を除いてxと同じ意味を持つ。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-R　　ディレクトリ階層内の所有権を再帰的に変更

chattr [オプション] [+ -=] 属性 [ファイル]　　e2fsprogs

/usr/bin　　stdin　　stdout　　- file　　-- opt　　--help　　--version

Linux以外のUnix環境を使用してきた人ならば、Linuxのファイルがアクセス権限以外の付加的な属性を持つということを聞いて、驚くかもしれない。ext2、あるいはext3(Fedoraのデフォルト)のファイルシステムを使用している場合には、chattr(change attribute)コマンドを用いて、これら拡張された属性を設定し、lsattrを用いて、その属性を表示できる。chmodを用いた場合のように、属性を追加(+)、削除(-)、あるいは絶対的な設定(=)を行うことも可能である。

属性	意味
a	追加のみ。ファイルへの追加が許可される。しかし、編集は不可能である。スーパーユーザのみ使用可能
A	タイムスタンプ無しのアクセス。このファイルへのアクセスは、アクセスに対するタイムスタンプ(atime レコード)を更新しない
c	圧縮される。データは自動的に書き出し時に圧縮、読み込み時に解凍
d	ダンプしない。ダンププログラムはバックアップを作成する際、このファイルを無視(詳細は、「4.15　バックアップとリモートディスク」を参照のこと)
i	不変にする。ファイルは変更、あるいは削除されない(スーパーユーザのみ)
j	ジャーナルデータ(ext3 ファイルシステムのみ)
s	安全な削除。削除する際、ファイルデータは、割り当てられていたブロックの中身が0となり、上書きされる
S	同時アップデート。ファイルの変更は即座にディスク上に反映される。ファイルを保存した後、syncを実行した場合と同様(詳細は、「4.13　ディスク操作」を参照のこと)
u	削除不可

オプション

-R　　サブディレクトリを再帰的に実行

This PDF is prepared for tools@maruhouse.org, personal use only.

lsattr [オプション] [ファイル] e2fsprogs

/usr/bin stdin stdout - file -- opt --help --version

lsattr(list attributes)は、chattrを用いて拡張された属性を参照することが可能である。出力は、chattrとして同じ文字を使用する。例えば、以下のファイルは不変であり削除不可である。

```
$ lsattr myfile
-u--i--- myfile
```

オプション

-R	ディレクトリやその内容の属性を再帰的に表示
-a	ドットから開始されるファイル名を含め、全ファイルを表示
-d	ディレクトリを表示する際に内容を表示せず、そのディレクトリのみ表示

ファイルを指定しない場合、lsattrは、現在のディレクトリにある全ファイルの属性を表示する。

4.7 ファイルの場所の表示

find	階層ディレクトリ内のファイルの場所を表示
slocate	ファイルのインデックスを作成し、引き数のインデックスを検索
which	コマンドサーチパス内の実行ファイルの場所を単数表示
type	コマンドサーチパス内の実行ファイルの場所を複数表示（bashビルトイン）
whereis	実行ファイル、ドキュメント、ソースファイルの場所を表示

Linux システムは、10万もの数のファイルを容易に保存することができる。このとき、膨大なファイルの中から、ある特定のファイルを発見するにはどうすればよいだろうか？まず最初に行うべきことは、見つけたいファイルが、どこのディレクトリに分類されているのかを論理的に推測することである。探索したいファイルによって、複数の方法が存在する。

findは、任意のファイルを発見するために、ディレクトリ階層をファイル単位で調べ上げる力まかせのプログラムである。

slocateは、事前にインデックスを生成し、それからファイルを検索するため、非常に高速である。Fedoraの場合、デフォルトではcron（/etc/cron.daily/slocate.cron）によって、1日1

This PDF is prepared for tools@maruhouse.org, personal use only.

回インデックス生成を行っている。

特定のファイルを検索するプログラムとして、whichとtypeがある。このコマンドは、シェルのコマンドサーチパスに含まれる全ディレクトリをチェックする。typeはbashシェルのビルトインコマンドである(そのため bash 使用時のみ利用可能)。which は、(通常、/usr/bin/whichにある)外部プログラムである。typeは、高速であり、またシェルのエイリアスを展開することができる。

find [ディレクトリ] [条件]

findutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

find は、1つもしくは複数のディレクトリから(および、そのサブディレクトリを再帰的に)、指定した条件と一致するファイルを検索するコマンドである。このコマンドは、オプションが50個以上もあり、非常に強力であるが、残念ながらあまり使用されていない構文を用いる。以下、ルートディレクトリから全ファイルシステムを検索する例を示す。

この例では、myfile という名前のファイルを検索する。

```
$ find / -type f -name myfile -print
```

ディレクトリの名前を全て表示する。

```
$ find / -type d -print
```

オプション

-name *pattern*
-path *pattern*
-lname *pattern*
-iname *pattern*
-ipath *pattern*
-ilname *pattern*
: ファイルの名前 (-name)、パス名(-path)、あるいは、シンボリックリンク名(-lname)を表示。これらはシェルのワイルドカード*、 ?、[]が使用できる。パス名は検索されたディレクトリ名に含む。-iname、-ipath、-ilnameオプションは、それぞれ、-name、-path、-lnameと同じであるが、大文字小文字を区別しない

-regex *regexp*
: 検索するディレクトリツリーのパスが正規表現に一致

-type f|d|l|b|c|p|s
: 通常のファイル(f)、ディレクトリ(d)、シンボリックリンク(l)、ブロックデバイスファイル(b)、キャラクタデバイスファイル(c)、名前付きパイプ(p)、ソケット(s)を表示

-atime *N*
-ctime *N*
-mtime *N*
: ファイルの変更時間で指定。単位は「時間」。ファイルが最後にアクセスされた時間(-atime)、ファイルが最後に変更された時間(-mtime)、ファイルが *N* 日間で、ステータスの変化があった(-ctime)。「*N* 日より大きい」

This PDF is prepared for tools@maruhouse.org, personal use only.

	場合は *+N* を使用。また「*N* 日より小さい(以内)」という場合は、*-N* を使用
-amin *N* -cmin *N* -mmin *N*	ファイルの変更時間での指定。単位は「分」。ファイルが最後にアクセスされた(-amin)、最後に変更された(-mmin)、*N*分前にステータスの変化があった(-cmin)場合に表示。「*N* 分より大きい」場合は、*+N* を使用。または「*N* より小さい(以内)」という場合は、*-N* を使用
-anewer *other_file* -cnewer *other_file* -newer *other_file*	ファイルの時間を比較。ファイルが*other_file*より後に(新しく)アクセスされた(-anewer)、変更された(-newer)、ステータスの変化があった(-cnewer)場合に表示
-maxdepth *N* -mindepth *N*	ディレクトリツリー内の最大(-maxdepth)*N* レベルの深さまでファイルを検索。あるいは、少なくとも(-mindepth)*N* レベルの深さまでファイルを検索
follow	シンボリックリンクの参照先を検索
-depth	深さ優先で検索を実行。このためディレクトリ本体より先に、ディレクトリの内容を再帰的に検索
-xdev	他のファイルシステムにあるディレクトリを探索しない
-size N[bckw]	ファイルのサイズ*N*を指定。サイズは、ブロック(b)、1バイト文字(b)、キロバイト(k)、2バイト文字(w)単位で与えられる。「*N* より大きい」場合は、*+N* を、「*N* より小さい」場合は *-N* を使用
-empty	ファイルサイズが空である、通常のファイルあるいはディレクトリを検索
-user *name*	ファイルのユーザ名、グループ名を指定
-group *name* -perm *mode* -perm *-mode* -perm *+mode*	ファイルのパーミッション(8進または文字により)を指定。与えられたビットの「全て」がセットされていることを確認する場合は *-mode* を、「どれか一つ」でもビットがセットされていることを確認する場合は *+mode* を指定

以下に示す方法により、条件式の評価が可能となる。

条件式1 -a 条件式2

AND条件 (2つの条件式が並んで指定されていれば、デフォルトでAND条件になる。このため、“-a”は使用しなくともよい)

条件式1 -o 条件式2

OR条件

This PDF is prepared for tools@maruhouse.org, personal use only.

! 条件式

-not 条件式

条件式の否定

(条件式)

代数クラスのような優先順位の指定。かっこ内を先に評価する。"\" を用いて、シェルによる解釈をエスケープしなければならない場合がある

条件式 1 , 条件式 2

C 言語のコンマ操作と同様。両方の式を評価し、2 番目の式の返り値を渡す

条件式に一致したファイルを操作するオプションを以下に示す。

オプション

-print	ファイルのパスを表示
-printf *string*	書式指定文字列で表示形式を指定。Cのライブラリ関数であるprintf()と同様の規則。指定方式については、man ページを参照
-print0	-print と同様だが、改行文字の代わりにヌル文字(ASCII 0) を使用する。これは、find の出力を他のプログラムにパイプを用いて渡す際に有用である。この理由は、ファイル名に空白文字が含まれている場合があるためである。当然、パイプで渡されたプログラムでは、xargs -0などを用いて、ヌル文字を読み込み、パース(解析)する必要がある
-exec *cmd* ; -ok *cmd* ;	シェルコマンド *cmd* を起動。当然ながら、最後の;を含めて、シェルのメタ文字はエスケープする必要があるので、コマンドラインで実行するコマンドの内容をそのまま指定することはできない。また、{}はファイル名に置換される(これもクォートやエスケープが必要)。-ok は、シェルコマンドの実行前に、ユーザに問い合わせを行う。一方、-exec は問い合わせを行わずにコマンドを実行
-ls	ls -dils を実行

findは、ファイルの一覧を標準出力に書き出すが、xargsコマンドを組み合わせて使用することで、複雑な処理を行うことが可能。xargsは、標準入力からファイルの一覧を読み込み、そのファイルにコマンドに実行する(詳細は、man xargsを参照のこと)。例えば、カレントディレクトリ以下にある、"myxomatosis"という単語を含むファイルを検索するには、以下のコマンドを実行する。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ find . -print0 | xargs -0 grep myxomatosis
```

slocate [オプション]

slocate

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

slocate(secure locate)は、ファイルのインデックス(データベース)を作成することによって、高速な検索を行うコマンドである。slocateは、それほど変更されることが無いディレクトリにある多数のファイルを検索するのに非常に適している。ファイル数が少ない場合や、ファイルを探索するために複雑な条件を指定する場合には、find を使用する事が望ましい。

Fedora Linux は、1 日に 1 回、全てのファイルシステムのインデックスを自動的に作成するが、ユーザ自身でインデックスを作成したい場合(データベースを/tmp/myindexに収納する場合)には、以下のように実行する。

```
$ slocate -u -o /tmp/myindex
```

ディレクトリを指定し、その下のサブディレクトリ全てのインデックスを作成するには、以下を実行する。

```
$ slocate -U directory -o /tmp/myindex
```

そして、インデックスの中の文字列を検索するには、以下を実行する。

```
$ slocate -d /tmp/myindex string
```

slocate はファイルの読み出しについて、「安全」と言われるが、それはなぜか？ 理由は、あるユーザに読み出し許可が与えられていないファイルは、検索時に表示されないためである。このため、スーパーユーザがこのプロテクトされたディレクトリのインデックスを作成した場合、スーパーユーザではないユーザも検索は実行可能ではあるが、プロテクトされたファイルは表示されない。

インデックスのオプション

-u	ルートディレクトリ以下の全ディレクトリのインデックスを作成
-U ディレクトリ	指定したディレクトリ以下のインデックスを作成
-l(0|1)	セキュリティをオフ(0)、あるいはオン(1)。デフォルトは 1
-e ディレクトリ	インデックスから指定したディレクトリを除外 複数のディレクトリを指定する場合には、カンマによってパスを区切る

This PDF is prepared for tools@maruhouse.org, personal use only.

-o *outfile*　　*outfile* ファイルをインデックスファイルとして指定

検索のオプション

-d *index*　　使用するインデックスファイルの指定（先ほどの例では /tmp/myindex）

-i　　大文字小文字の区別なしに検索

-r *regexp*　　正規表現 regrexp に一致するファイルを検索

which [ファイル]　　which

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

which は、シェルのコマンドサーチパス中の実行ファイルを表示するコマンドである。

```
$ who
```

というコマンド名を入力して、プログラムが起動できる場合、whichコマンドは、このコマンドがどの場所に存在するかを通知する。

```
$ which who
/usr/bin/who
```

当然ではあるが、which プログラム自身についても発見できる。

```
$ which which
/usr/bin/which
```

コマンドサーチパスの中に、同じ名前で複数のプログラムが存在する場合、（例えば、/usr/bin/whoと/usr/local/bin/whoが存在した場合）、whichは最初に検索されたパスのみを表示する（PATH 変数に依存）。

type [オプション] コマンド　　bash

shell built-in　　stdin　stdout　- file　-- opt　--help　--version

typeは、シェルのコマンドサーチパスの中で実行可能なファイルを表示するコマンドである。

```
$ type grep who
grep is /bin/grep
who is /usr/bin/who
```

which は外部コマンドであるが、type は bash シェルのビルトインコマンドである。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ type which type rm if
which is /usr/bin/which
type is a shell builtin
rm is aliased to `/bin/rm -i'
if is a shell keyword
```

typeは、ビルドインコマンドであるため、whichより高速である。しかし、bashが起動している場合しか実行できない。

whereis [オプション] ファイル　util-linux

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

whereisコマンドは、あらかじめ設定されたディレクトリを検索して、指定したファイルを表示する。このコマンドによって、実行ファイル、ドキュメント、ソースコードを発見することが可能である。whereisは、検索したいファイルが、設定したディレクトリの中に含まれていない場合もあるため、若干風変わりなコマンドと言えるかもしれない。

オプション

-b -m -s	実行ファイルのみ表示(-b)、man ページのみ表示(-m)、ソースコード(-s)のみ表示
-B ディレクトリ -f ファイル -M ディレクトリ -f ファイル -S ディレクトリ -f ファイル	与えられたディレクトリの中でのみ、実行ファイル(-B)、manページ(-M)、ソースコード(-S)を検索する。検索を開始する前に、-f オプションでディレクトリリストの作成を終えている必要がある

4.8 テキストファイルの操作

grep	ファイルから正規表現にマッチした行を表示
cut	ファイルから選択した部分を切り出す
paste	列を追加
tr	文字を他の文字に変換
nkf	日本語の文字コード変換
sort	評価基準に従ってテキスト行をソート
uniq	ファイル中の重複していない行を表示
string	ファイル中の文字列を表示
tee	ファイルのコピーと標準出力への表示を同時に実行

This PDF is prepared for tools@maruhouse.org, personal use only.

Linuxが最も得意とする作業の1つとして、テキスト操作がある。これはテキストファイル(あるいは標準入力)を希望する形式に変換することを意味する。標準入力を読み込み、標準出力に書き出すプログラムは、全てこのカテゴリになる。ここでは、非常によく使用され、かつ強力なものについて、いくつかを取り上げる。

grep [オプション] パターン [ファイル] grep

/bin stdin stdout - file -- opt --help --version

grepは、Linuxコマンドの中で最も便利で強力なコマンドの1つである。その理由は、簡単であり、1つ以上のファイルから、正規表現のパターンにマッチした行を全て出力できるからである。例えば、

```
The quick brown fox jumped over the lazy dogs!
My very eager mother just served us nine pancakes.
Film at eleven.
```

という内容が記述されたファイルから、pancakeを含む行を全て検索したい場合には、以下のようにする。

```
$ grep pancake myfile
My very eager mother just served us nine pancakes.
```

grepは、基本型(basic)と拡張型(extended)と呼ばれる2つの異なった正規表現を使用できる。この2つはどちらも同様に強力であるが、表現が全く異なる。他のgrep(fgrep、egrep)を使用した経験に基づいて、どちらか一方を選択すればよい。基本的な構文を表2と表3に示す。

オプション

-v	正規表現でマッチしない行のみ表示
-l	マッチする行を持つファイル名のみを表示。行自身は表示しない
-L	マッチする行を持たないファイル名のみを表示
-c	マッチした行数を表示
-n	マッチした行の前に行番号を付けて表示
-b	マッチした行の前に入力ファイルの先頭からのオフセットを(バイト単位で)表示
-i	マッチングにおいて大文字と小文字を区別しない

This PDF is prepared for tools@maruhouse.org, personal use only.

-w	完全な単語(すなわち単語全体と正規表現が正確にマッチする単語)とマッチした行のみ選択
-x	行全体と正確にマッチする行のみを選択。-w を無効にする
-A *N*	マッチした行の後の *N* 行を表示
-B *N*	マッチした行の前の *N* 行を表示
-C *N*	-A *N* -B *N* と同様。マッチングした行の前後 *N* 行を表示
-r	各ディレクトリ下の全てのファイルを再帰的に読み取る
-E	拡張正規表現を使用。詳細は、egrep を参照のこと
-F	正規表現のかわりに固定文字列のリストを使用。詳細は、fgrepを参照のこと

egrep [オプション] パターン [ファイル]

grep

/bin　stdin　stdout　- file　-- opt　--help　--version

egrep は、grep コマンドと類似しているが、正規表現に「拡張された」言語を使用している。その他の部分は、grep と全く同様である。egrep は、grep -E と同じ働きをする。

表 2　grep の正規表現：通常型と拡張型

正規表現	意味
.	任意の 1 文字
[...]	リスト中の任意の一文字にマッチ
[^...]	リスト中にはない任意の一文字にマッチ
(...)	文字列を一まとめに扱う
^	行の開始
$	行の終了
\<	単語の開始
\>	単語の終了
[:alnum:]	アルファベットと(十進)数字
[:alpha:]	アルファベット
[:cntrl:]	制御文字
[:digit:]	10 進数字
[:graph:]	印字可能かつ表示可能な文字(スペースは印字可能だが表示可能ではない)
[:lower:]	アルファベットの小文字
[:print:]	印字可能な文字(制御文字以外の文字)

This PDF is prepared for tools@maruhouse.org, personal use only.

表2 grep の正規表現：通常型と拡張型(続き)

正規表現	意味
[:punct:]	句読点(「通常の文字、数字、制御文字、スペース」のいずれでもない文字)
[:space:]	ホワイトスペース文字(スペース、タブ、改ページ)
[:upper:]	アルファベットの大文字
[:xdigit:]	16 進文字
*	正規表現の 0 回以上の繰り返し
\c	*c*が特別な正規表現であった場合でも、文字通り、文字*c*と解釈する。例えば、アスタリスクをマッチするためには*を、バックスラッシュをマッチするには、\\を用いる。あるいは、[*]や[\]のように、大カッコの内側にリテラル文字を入れる

表3 通常型と拡張型の正規表現の違い

通常型	拡張型	意味
|	|	OR 条件
\+	+	正規表現の 1 つあるいはそれ以上の繰り返し
\?	?	正規表現の 0 あるいは 1 つ以上の出現
\{n\}	{n}	正規表現の *n* 回の繰り返し
\{n,\}	{n,}	正規表現の *n* 回あるいはそれ以上の繰り返し
\{n,m\}	{n,m}	正規表現の *n*～*m* 回(*n* < *m*)の(帰納的な)繰り返し

fgrep [オプション] 固定文字列 [ファイル]　grep

/bin　stdin　stdout　- file　-- opt　--help　--version

fgrep は、grep と類似したコマンドである。しかし、fgrep は正規表現の代わりに、改行で区切られた固定文字列のリストを扱う点が異なる。これは、grep -F と同じ動作になる。例えば、ファイルから one、two、three という文字列を検索するには以下を実行する。

```
$ fgrep 'one          末尾に改行を入力している点に注意
two
three' myfile
```

fgrep は、通常 -f オプションを指定する。これは、ファイルからパターンを読み込むことを意味する。例えば、文字列が記載された辞書ファイルがあり、一行ごとの文字列が以下のようであったとしよう。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ cat my_dictionary_file
aardvark
aback
abandon
...
```

このファイルを用いて指定したファイルから、これらの文字列にマッチする行を簡単に検索することが可能である。

```
$ fgrep -f my_dictionary_file *
```

fgrep も、正規表現中のメタ文字を文字通りに扱えるので、* や{のようにアルファベットでない文字列の検索に適している。

cut -(b|c|f) 範囲 [オプション] [ファイル] coreutils

/usr/bin stdin stdout - file -- opt --help --version

cut は、ファイルからテキストのコラムを抽出するコマンドである。「コラム」は文字単位のオフセット、もしくはバイト単位のオフセットのいずれかで指定する。例として、各行の19番目の文字を抽出する場合を考えてみよう。この場合において前者は、

```
$ cut -c19 myfile
```

となる。一方、後者の場合は、

```
$ cut -b19 myfile
```

となる。一般的にバイト単位のオフセット指定は、マルチバイト文字を使用していない場合、文字と同等に扱える。次に、デリミタ文字で区切られたフィールドを抽出する場合を考える。例として、カンマで区切られたファイルにおいて、各行の5番目のフィールドを抽出する場合は、以下のようになる。

```
$ cut -d, -f5 myfile
```

コラムの指定には様々な方法が使用できる。例えば、範囲指定(例:3-16) や、コンマで区切る指定(例:3,4,5,6,8,16)、あるいはその両方(例:3,4,8-16) を使用するといった指定が可能である。範囲について最初の数字が省略された場合(例:-16) は、最初に1が入っていると想定される(例:1-16)。反対に最後の数が省略された場合(例:5-)、行の最後までと想定される。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-d *C*	-fに対するフィールド間の入力デリミタ(区切り)文字として、文字 *C* を使用。デフォルトではタブ文字
--output-delimiter=*C*	-f に対するフィールド間の出力デリミタ文字として、*C* を使用。デフォルトではタブ文字
-s	デリミタ文字が含まれていない行を出力しない

paste [オプション] [ファイル]　　coreutils

/usr/bin　　stdin　　stdout　　- file　　-- opt　　--help　　--version

pasteは、cutコマンドと反対の機能を持つコマンドである。すなわち、いくつかのファイルを縦方向に並ぶコラムとして取り扱い、それらを結びつけ、標準出力に書き出す。

```
$ cat letters
A
B
C
$ cat numbers
1
2
3
4
5
$ paste numbers letters
1  A
2  B
3  C
4
5
$ paste letters numbers
A  1
B  2
C  3
   4
   5
```

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-d デリミタ　デリミタ文字をコラム間のデリミタ(区切り)文字として使用する。デフォルトはタブ文字。(-d:)の様に一文字(:)をデリミタ文字と指定した場合、:をデリミタ文字として使用する。(-dxyz)のように複数の文字列(xyz)を指定すると、各行において、最初のデリミタ文字をx、次のデリミタ文字はy、その次のデリミタ文字はz、x、y... と順に続けられる

-s　出力の行と列を入れ換え

```
$ paste -s letters numbers
A   B   C
1   2   3   4   5
```

tr [オプション] 文字集合 1 [文字集合 2]　coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

trは、ある文字集合を他の文字集合に変換するコマンドである。このコマンドは、シンプルであるが非常に有用である。例えば、全ての母音をアスタリスクに変換するには、以下を実行する。

```
$ cat myfile
This is a very wonderful file.
$ cat myfile | tr aeiouAEIOU '*'
Th*s *s * v*ry w*nd*rf*l f*l*.
```

全ての母音を削除するには、以下を実行する。

```
$ cat myfile | tr -d aeiouAEIOU
Ths s  vry wndrfl fl.
```

ファイル中の全ての小文字を大文字にするには、以下を実行する。

```
$ cat myfile | tr 'a-z' 'A-Z'
THIS IS A VERY WONDERFUL FILE.
```

trは、*charset1*で指定された最初の文字を*charset2*で指定された最初の文字に変換し、2番目のものは2番目に変換、3番目のものは3番目に変換 ... と続く。*charset1* の文字列の長さが *N* 文字である場合、*charset2* で指定して文字列の最初の *N* 文字のみが置換される(文字列 *charset1* の長さが、文字列 *charset2* の長さより長い場合は、-t オプションを参照のこと)。

文字集合としては、以下の指定が可能である。

This PDF is prepared for tools@maruhouse.org, personal use only.

形式	意味
ABCD	A, B, C, D という文字の連続
A-B	A から B までの文字の範囲
[x*y]	文字 x を y 回繰り返す
[:class:]	grep と同じ文字クラス([:alnum:]、[:digit:]など)

trもprintf(「9.1 画面出力」を参照)と同様にエスケープ文字“\a”(^G=ビープ音)、“\b”(^H=バックスペース)、“\f”(^L=前進)、“\n”(^J=改行)、“\r”(^M=改行)、“\t”(^I=タブ)、“\v”(^K=垂直タブ)などを使用することができる。これは*nnn*という指定で、8進数値*nnn*を指定した場合と同様である。

trは、単純な変換が高速にできる点で優れている。より強力な編集を行う場合は、sedやawkやperlの使用を検討してみてほしい。

オプション

-d 指定した文字列*charset1*を入力文字列から削除

-s 指定した文字列*charset1*で見つかった隣り合う文字の重複を文字列から取り除く。例えば、tr -s aeiouAEIOUは、隣り合って重複している母音を削除する(この結果、reeeeeeally は really になる)

-c 指定した文字列*charset1*で発見できなかった全ての文字を扱う

-t 文字列*charset1*が文字列*charset2*より長い場合、文字列*charset1*の端を切り詰めて、長さを同一にする。-tを使用しない場合、文字列*charset2*の最後の文字が、文字列*charset2*が文字列*charset1*と同じ長さになるまで繰り返される

nkf [オプション] [ファイル] nkf

/usr/bin/nkf stdin stdout - file -- opt --help --version

nkfは、様々な文字コードを変換するためのフィルタコマンドである。nkfが認識できる入力の文字コードは、「JISコード」(ISO-2022-JP準拠)、MS漢字コード(シフトJIS)、日本語EUC(AT&Tコード)、UTF-8のいずれかである。nkfは、Linuxコマンドの中で最も有益で強力なコマンドの一つである。これは、1つあるいは複数のファイルの文字コードを容易に変換できるためである。特にテキスト処理やメール処理などにおいて有用である。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-d	JIS コードを出力(デフォルト)
-e	EUC コードを出力
-s	シフト JIS コードを出力
-w	UTF-8 コードを出力
-m[BQSNO]	MIME を解釈(デフォルト)するが、ISO-2022-JP(base64)と ISO-8859-1 (Q encode)のみ。ISO-8859-1 (Latin-1) の場合には、-l フラグが必要。-mO では MIME を解釈しない。-mQ、-mB により、Q encode、B encode されているものとして処理
	-mB MIME base64 stream を解釈
	-mQ MIME quoted stream を解釈
	-mS MIME の詳細なチェック(デフォルト)
	--mN MIME のチェック
	-mO MIME を解釈しない
-J	入力を ISO-2022-JP と想定
-E	入力を日本語EUC (AT&T) と想定
-S	入力を MS 漢字、X0201 仮名と想定
-W	入力を UTF-8 と想定
-X	入力を MS 漢字中に X0201 仮名と想定
-B	壊れた JIS コード。ESC 無しと想定
-Z	X0208 中の英数字と若干の記号を ASCII に変換
-b	バッファリング出力(デフォルト)
-u	出力時にバッファリングしない
-t	何も変換しない
-d	改行コード CR を削除
-c	改行コード CR を追加

sort [オプション] [ファイル] coreutils

/bin stdin stdout - file -- opt --help --version

sortは、テキストをアルファベット順に出力するコマンドである。もしくは、指定したルールを用いてソートする。指定した全ファイルは連結され、結果がソートされて出力される。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ cat myfile
def
xyz
abc
$ sort myfile
abc
def
xyz
```

オプション

-f	大文字小文字を区別しないソート
-n	アルファベット順(10は先頭が1なので、9より前となる)の代わりに数値順にソート(9が10の前となる)
-g	数学的記法を解釈し、数値順にソート(例えば、7.43eは7.4*10^3、あるいは7400と解釈する)。記法に関する詳細は、info sortを参照のこと
-u	同一の行が存在する場合、その中の最初の行だけを表示。-cオプションと共に使用した場合、連続した行について同じものがないか確認
-c	ソートせずに、ファイルが既にソートされているかのみを確認。ソートされていた場合には何も行わない。それ以外の場合、エラーメッセージを表示
-b	各行の行頭の空白を無視
-r	出力を逆にする。降順(キー値の大きいものから順に出力)
-t *X*	*X*をデリミタ文字(フィールドを分離する文字)として扱う
-k F1[.C1][,F2[.C2]]	ソートキーの選択

ソートキーとは、ソートの対象となる(行中の)箇所のことである。例えば、「各行に5文字が並んでいる」場合を考えると、通常sortは以下の行は既にソートが完了していると判断する。

```
aaaaz
bbbby
```

もし、ソートキーが「各行の5番目の文字」ならば、yはzの前であるので行の順番が入れ替わる。構文について、以下の表に示す。

This PDF is prepared for tools@maruhouse.org, personal use only.

アイテム	意味	明記されていない場合のデフォルト
F1	開始フィールド	必須
C1	フィールド1内にある開始位置	1
F2	最終フィールド	最後のフィールド
C2	最終フィールド内にある開始フィールド	1

このため、sort -k1.5は、最初のフィールドの5番目の文字を基点としてソートする。sort -k2.8,5は、「2番目のフィールドの8番目の文字から5番目のフィールドの最初の文字まで」を意味する。複数キーを指定するために、-kオプションを複数使用することが可能である。この複数キーは、コマンドライン上で指定して、ファイルの最初から最後まで適用することができる。

uniq [オプション] [ファイル]

coreutils

/usr/bin　stdin　stdout　-file　--opt　--help　--version

uniqは、テキストの連続、あるいは重複を整理するコマンドである。例えば、myfileが以下のように記載されているとしよう。

```
$ cat myfile
a
b
b
c
b
```

このとき、uniqは、連続した2つのbを発見し、指定された処理(削除)を行う。しかし、3つめのbは、3文字目のbとの間にcが挟まれていて連続していないため削除を行わない。

```
$ uniq myfile
a
b
c
b
```

uniqは、sortコマンドの後で使用されることが多い。

```
$ sort myfile | uniq
a
b
c
```

This PDF is prepared for tools@maruhouse.org, personal use only.

この場合、bは1文字のみが残る。また、-cオプションを用いて、文字を削除する代わりに連続した行をカウントすることができる。

```
$ sort myfile | uniq -c
      1 a
      3 b
      1 c
```

オプション

-c	隣り合う重複している行をカウント
-i	大文字小文字を区別しない
-u	1回のみ現れる行を出力
-d	重複した行のみを表示
-s *N*	重複を発見した場合、各行の最初の*N*文字を無視
-f *N*	重複を発見した場合、最初の*N*個の空白で区切られたフィールドを無視（フィールドとは、空白とタブ以外の文字からなる文字列のこと）
-w *N*	重複を発見した場合、各行の最初の*N*文字のみ考慮。オプションとして-sや-fを用いた場合、sortは、最初に記載された数の文字のやフィールドを無視し、その後、次の*N*文字を考慮

strings [オプション] [ファイル] strings

/usr/bin/strings stdin stdout - file -- opt --help --version

stringsは、ファイル中の表示可能な文字列を表示するコマンドである。stringsは、指定されたファイルから表示可能なキャラクタ文字列を表示する。デフォルトでは4文字以上の長さのものを可能な限り表示する。ファイルがオブジェクトファイルの場合、初期化された箇所からの文字のみ表示する。それ以外のファイルの場合、ファイル全体から文字列を探索して表示する。stringsは、主としてテキストファイル以外のファイルの内容を調査する場合などに使用する。

オプション

-a --all	オブジェクトファイルからファイル全体を参照
-f --print-file-name	文字列の前にファイル名を表示
-n -min-len	min-len以上の長さを持つ文字のシーケンスを表示（デフォルトは4）

This PDF is prepared for tools@maruhouse.org, personal use only.

-t {o,x,d} --radix={o,x,d} --o　各文字列の前にファイル中のオフセットを表示。引数として指定された一文字によりオフセットの基数を(8, 16, 10)進数として指定

tee [オプション] [ファイル]　coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

tee は、cat コマンドと同様、標準入力をそのまま標準出力にコピーするコマンドである。しかし、teeの場合、同じ標準入力を1つあるいは複数のファイルへ同時にコピーすることが可能である。teeは、パイプラインの途中でよく使用され、teeの次のコマンドにデータを渡すと同時に、そのデータを中間データとしてファイルに保存する。

```
$ who | tee original_who | sort
```

これは、標準出力で wh コマンドの出力をソートして表示し、ファイル original_who に who のオリジナル(つまりソートされていない)をコピーする。

オプション

-a　ファイルを上書きせず、標準入力をファイルに追加
-i　割り込みシグナルを無視

4.9　より強力なテキスト操作

これまでの説明は、Linux のテキストフィルタリング操作に関する氷山の一角に触れたに過ぎない。Linux には、さらに複雑なデータ処理を行える何百ものフィルタコマンドが存在する。しかしこれらのコマンドを理解するにはさらなる学習が必要であり、本書で扱うにはあまりにも負担が大きい。このため以下では、重要なフィルタコマンドについて、簡単な説明をするだけに留めておく。

awk

awkは、パターンマッチング言語である。awkは、正規表現に従ってデータをマッチさせ、そのデータを用いて命令を実行する。以下にテキストファイルmyfileを処理する簡単な例をいくつか示す。

各行の 2 番目と 4 番目の単語を表示する。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ awk '{print $2, $4}' myfile
```

60 文字より短い全ての行を表示する。

```
$ awk '{length($0) < 60}' myfile
```

sed

awk と同様、sed は、テキスト行を処理するパターンマッチングするためのツールである。sed の構文は、vim や、ラインエディタ ed の構文と密接に関連している。以下にいくつかの簡単な例を示す。

テキスト内の文字列“red”の全てを“hat”に変換して表示する。

```
$ sed 's/red/hat/g' myfile
```

テキストの最初の 10 行を削除して表示する。

```
$ sed '1,10d' myfile
```

m4

m4は、マクロ処理言語である。m4はファイル内にあるキーワードの場所を表示し、キーワードを指定した値に置換する。例えば、以下のファイルについて考えてみよう。

```
$ cat myfile
My name is NAME and I am AGE years old
ifelse(QUOTE,yes,No matter where you go... there you are)
```

m4 は NAME、AGE、QUOTE に対する置換を行う。

```
$ m4 -DNAME=Sandy myfile
My name is Sandy and I am AGE years old

$ m4 -DNAME=Sandy -DAGE=25 myfile
My name is Sandy and I am 25 years old

$ m4 -DNAME=Sandy -DAGE=25 -DQUOTE=yes myfile
My name is Sandy and I am 25 years old
No matter where you go... there you are
```

perl と python

PerlやPythonは、非常に強力なスクリプト言語である。開発環境も整備されており、質の

This PDF is prepared for tools@maruhouse.org, personal use only.

高いアプリケーションを作成できる。

4.10　スペルチェック

look	簡単な単語スペルチェッカー
aspell	インタラクティブなスペルチェッカー
spell	バッチ処理のスペルチェッカー

Linuxでは、いくつかのスペルチェッカープログラムが提供されている。GUIを用いたスペルチェッカーに慣れたユーザであれば、CUIは使いにくいと思われるかもしれないが、パイプラインを使用することでかなり強力なチェックも可能となる。

look [オプション] 系列 [辞書ファイル]　　util-linux

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

lookは、指定した文字列で始まる単語を(標準出力に)表示するコマンドである。その単語は、辞書ファイル(デフォルトは /usr/share/dict/words)から選択される。
例えば、look biggと実行すると、以下のように表示される。

```
bigger
biggest
Biggs
```

辞書ファイル(単語がアルファベット順に並べられたテキストファイル)を指定する場合、lookは辞書ファイルから指定した文字列で始まる全ての行を出力する。

オプション

-f	大文字小文字を区別しない
-t *X*	指定した文字列において、文字 *X* が最初に出現する箇所まで(指定した文字を含む)を対象として検索する。例えば、look -t i bigは、“bi”で始まる全ての単語を表示

This PDF is prepared for tools@maruhouse.org, personal use only.

aspell [オプション] ファイル | コマンド　aspell

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

aspellは、何十ものオプションを提供する強力なスペルチェッカーである。有用なコマンド使用例を以下に示す。

aspell -c *file*
: file 内の全てのファイルを対話的にチェックし、オプションにより訂正

aspell -l < *file*
: 標準出力に *file* 内の誤ったスペルの単語のリストを表示

aspell dump master
: aspell の master ディレクトリを標準出力に出力

aspell help
: 簡潔なヘルプメッセージを表示。詳細は、http://aspell.net/ を参照のこと

spell [ファイル] ファイル | コマンド　aspell

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

spellは、辞書ファイルに従い、与えられたファイル内のミススペルを含む単語全てを表示するコマンドである。以下のコマンドと同様の結果を出力する。

```
$ cat files | aspell -l | sort -u
```

ファイルを指定しない場合、spell は標準入力から読み込む。

4.11 ファイルの圧縮と解凍

gzip	GNU zip 形式にファイルを圧縮
gunzip	GNU zip 形式のファイルを解凍
compress	伝統的な Unix の圧縮を使って圧縮（時代遅れ）
uncompress	伝統的な Unix の圧縮を使って解凍（時代遅れ）
zcat	標準入力 / 出力（gzip あるいは compress）を通してファイルを圧縮 / 解凍

This PDF is prepared for tools@maruhouse.org, personal use only.

bzip2	bzip 形式にファイルを圧縮
bunzip2	bzip 形式のファイルを解凍
zip	Windows zip 形式にファイルを圧縮
unzip	Windows zip 形式のファイルを解凍
uuencode	uuencode 形式にファイルを変換
uudecode	uuencode 形式のファイルをテキスト形式に変換

Linux には、ファイルを圧縮、解凍する多数のコマンドが提供されている。その中でも最も有名なファイル形式が GNU zip（gzip）であり、これは圧縮されたファイルに .gz という拡張子が付けられる。その他のよく見かけるフォーマットは、古典的な Unix の圧縮（拡張子 .Z）、bzip2 圧縮（拡張子 .bz2）、そして Windows システムのファイル（拡張子 .zip）である。

これらの関連技術としてバイナリファイルをテキスト形式に変換するコマンドがある。このコマンドを用いれば、バイナリを変換してメールとして送ることができる。最近、このような作業は、MIME 形式の添付メール使用することが多い。しかし、uuencode や uudecode などの古いプログラムは現在でも使用されているため、本書ではこれらを説明する。

Macintosh の hqx/sit ファイル、arc ファイル、 zoo ファイルなどの対応できないファイル形式に出会った場合は、http://www.faqs.org/faqs/compression-faq/part1/section-2.html を参照のこと。

gzip [オプション] [ファイル]　gzip

/bin　stdin　stdout　- file　-- opt　--help　--version

gzip と gunzip は、GNU zip 形式の圧縮、解凍を行うコマンドである。圧縮されたファイルは拡張子が .gz となる。

コマンドの例

gzip *file*	*file* を圧縮し、*file*.gz を作成。元の *file* は削除
gzip -c *file*	圧縮データを生成して標準出力へ
cat *file* | gzip	パイプを用いて *file* を標準入力から読み出し、圧縮ファイルを作成
gunzip *file*.gz	*file*.gz を解凍して *file* を作成。元の *file*.gz は削除
gunzip -c *file*.gz	データを解凍して標準出力へ
cat *file*.gz | gunzip	パイプを用いて標準入力から読み出された圧縮データを解凍
zcat *file*.z	データを解凍して標準出力へ

This PDF is prepared for tools@maruhouse.org, personal use only.

gzip を用いた tar ファイルを扱うコマンドの例

tar czf *myfile*.tar.gz *dirname*　ディレクトリ *dirname* を圧縮＋パック
tar tzf *myfile*.tar.gz　ファイルの一覧を表示
tar xzf *myfile*.tar.gz　アンパック＋解凍

ファイル名の一覧を詳しく出力するには、ta に v オプションを追加する。

compress [オプション] [ファイル]　ncompress

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

compress と uncompress は、通常の Unix 圧縮形式（Lempel Ziv）でファイルを圧縮／解凍するコマンドである。圧縮されたファイルは、拡張子 .Z が付けられる。

コマンドの例

compress *file*　*file* を圧縮し、*file*.Z を作成。元の *file* は削除
compress -c *file*　圧縮データを生成して標準出力へ
cat *file* | compress　パイプを用いて *file* を標準入力から読み出し、圧縮ファイルを作成
uncompress *file*.Z　*file*.Z を解凍して *file* を作成。元の *file*.Z は削除
uncompress -c *file*.Z　データを解凍して標準出力へ
cat *file*.Z | uncompress　パイプを用いて標準入力から読み出された圧縮データを解凍
zcat *file*.Z　データを解凍して標準出力へ

圧縮された tar ファイル：コマンド例

tar cZf *myfile*.tar.Z *dirname*　ディレクトリ *dirname* を圧縮＋パック
tar tZf *myfile*.tar.Z　ファイルの一覧を表示
tar xZf *myfile*.tar.Z　アンパック＋解凍

ファイル名の一覧を詳しく出力するには、tar に v オプションを追加する。

bzip2 [オプション] [ファイル]　bzip2

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

bzip2 と bunzip2 は、Burrows-Wheeler 形式のファイルを圧縮、解凍するコマンドである。

This PDF is prepared for tools@maruhouse.org, personal use only.

圧縮ファイルは拡張子 .bz2 となる。

コマンド例

bzip2 file	*file* を圧縮し、*file.bz2* を作成。元の *file* は削除
bzip2 -c file	圧縮データを生成して標準出力に出力
cat file | bzip2	パイプを用いて *file* を標準入力読み出し、圧縮ファイルを作成
bunzip2 file.bz2	*file.bz2* を解凍して *file* を作成。元の *file.bz2* は削除
bunzip2 -c file.bz2	データを解凍して標準出力に出力
cat file.bz2| bunzip2	パイプを用いて標準入力から入力された圧縮データを解凍
bzcat file.bz2	データを解凍して標準出力に出力

bzip2 を用いた tar ファイルを扱うコマンドの例

tar cjf myfile.tar.bz2 dirname	ディレクトリ *dirname* を圧縮＋パック
tar tjf -myfile.tar.bz2	ファイルの一覧を表示
tar xjf myfile.tar.bz2	アンパック＋解凍

ファイル名の一覧を詳しく出力するには、v オプションを追加する。

zip [オプション] [ファイル]　zip

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

zip と unzip は、Windows zip 形式でファイルを圧縮、解凍するコマンドである。圧縮されたファイルは拡張子 .zip となる。gzip、compress、bzip2 などのコマンドとは異なり、zip コマンドは元のファイルを削除しない。

zip myfile.zip file1 file2 file3...	file1 file2 file3... を myfile.zip にまとめる
zip -r myfile.zip dirname	direname 以下を再帰的に myfile.zip にまとめる
unzip -l myfile.zip	myfile.zip の内容を表示
unzip myfile.zip	myfile.zip を展開

This PDF is prepared for tools@maruhouse.org, personal use only.

uuencode [オプション] [ファイル] 名前 sharutils

/usr/bin stdin stdout - file -- opt --help --version

ファイルをMIME形式で添付するメールツールが開発される以前では、バイナリファイルをメールで送信する際には多少の手間が必要であり、このとき用いた方法は送信前にuuencode(「ユーユーエンコード」と発音)コマンドを実行して、ASCII 形式に変換しなければならなかった。

```
begin 644 myfile
M(R`N8F%S:%]P<F]F:6QE"B,@4G5N<R!F:7)S="!W:&5N(&QO9V=I;F<@:6X@
M=6YD97(@1TY/344*"G1R87`@)PH@('1E<WO@+6X@(B134TA?O4=%3E1?4$E$
...
end
```

すなわちこのデータを受け取った受信者が、元のデータに復元するには、uudecode(「ユーユーデコード」と発音)コマンドを実行する必要がある。ファイルmyfileをuuencode形式に変換するには、以下を実行してmyfile.uuを作成する。

```
$ uuencode myfile newfile > myfile.uu
```

2番目の引き数newfileは、エンコードファイルの展開時に使用される名前である。この指定された名前は、uuencode された出力の最初の行に、以下のように現れる。

```
begin 644 newfile
M(R`N8F%S:%]P<F]F:6QE"B,@4G5N<R!F:7)S="!W:&5N(&QO9V=I;F<@:6X@
...
```

この uuencode されたファイル myfile.uu をデコードすると、newfile が作成される。

```
$ uudecode myfile.uu
```

4.12 ファイルの比較

diff	2つのファイルあるいはディレクトリを行単位で比較
comm	2つのソートされたファイルを行単位で比較
cmp	2つのファイルをバイト単位で比較
md5sum	ファイルのチェックサム(MD5)を計算

Linux のファイルを比較する方法として、以下の3つがあげられる。

This PDF is prepared for tools@maruhouse.org, personal use only.

- 行単位で比較(diff、diff3、sdiff、comm)。テキストファイルを扱う場合に最も適している。
- バイト単位で比較 (cmp)。バイナリファイルの場合によく使用される。
- チェックサムで比較(md5sum、sum、cksum、sha1sum)。

上記のプログラムは、全てテキストファイルを扱う。画像のファイル比較ツールについては、http://xxdiff.sourceforge.net にある xxdiff を試してみてほしい。

diff [オプション] ファイル1 ファイル2 diffutils

/usr/bin stdin stdout -file --opt --help --version

diff は、2つのファイル、あるいは2つのディレクトリを行単位で比較するコマンドである。テキストファイルを比較する場合に、diff はそれらの差分について詳細な表示を行う。バイナリファイルに対して、diffは、単にそれらが異なるか否かを通知する。全てのファイルを比較し、全く差が存在しない場合、diffは何も出力しない。通常の出力形式は以下のようになる。

```
Indication of line numbers and the type of change
< Corresponding section of file1, if any
---
> Corresponding section of file2, if any
```

例として、ファイル fileA が、以下の内容である場合を以下に示す。

```
Hello, this is a wonderful file.
The quick brown fox jumped over
the lazy dogs.
Goodbye for now.
```

このファイルについて、最初の行を削除し、2行目の“brown”を“blue”に換え、最後の行を加えた、ファイル fileB を作成する。

```
The quick blue fox jumped over
the lazy dogs.
Goodbye for now.
Linux rOOlz!
```

ここで、diff fileA fileB を実行すると、以下のような出力を行う。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
1,2c1                              fileAの1-2行がfileBの1行になる
< Hello, this is a wonderful file. fileAの1-2行
< The quick brown fox jumped over
---                                diffの区切り
> The quick blue fox jumped over   fileBの1行
4a4                                4行目が追加されている
> Linux rOOlz!                     追加された行
```

行頭の記号<と>は、それぞれfileA、fileBを示す矢印である。上記の出力形式がデフォルトであるが、他にも多数の形式が利用可能である。出力形式のいくつかは直接他のツールに引き渡される。実際にコマンドを試して、それらがどのような出力になるかを確認してみてほしい。

オプション	出力形式
-n	rcsdiffが使用するRCS形式で出力(詳細は、man rcsdiffを参照のこと)
-c	patchコマンドが使用するcontext形式で出力(詳細は、man patchを参照のこと)
-D *macro*	Cのプリプロセッサ形式で出力する。#ifdef*macro* ... #else ... #endifを使用
-u	unified形式。追加の場合“+”、削除の場合“-”を使用して出力
-y	2つのファイルを横に並べて表示。横幅を調節するには-Wを使用
-e	edのスクリプトを生成。実行するとfileAをfileBに変換
-q	差分の詳細については、表示しない。ファイルが異なるかのみ表示

diffは、ディレクトリを比較することも可能である。

```
$ diff dir1 dir2
```

これは、2つのディレクトリdir1、dir2について、1つめのディレクトリdir1の内容を名前別にソートした後、dir2の同じ名前のファイルを対象として、デフォルトのdiffコマンドを実行する。ディレクトリ構造になっているファイルの全てを再帰的に比較したいような場合には、-rオプションを使用する。

```
$ diff -r dir1 dir2
```

このコマンドは、同一の名前のファイル全てに対して通知するため、出力結果が大きくなることもある。

オプション

-b　　空白文字を無視

This PDF is prepared for tools@maruhouse.org, personal use only.

-B　　空白行を無視
-i　　大文字と小文字の区別をしない
-r　　ディレクトリを比較する際、見つかったサブディレクトリも再帰的に比較

ファイルの違いを取り扱うプログラムは複数存在するが、diff はその中のプログラムの 1 つに過ぎない。他のプログラムとしては、3つのファイルを一度に比較する diff3 や、2つのファイルの違いを発見し、対話的にマージ作業を実行し、3つめのファイルを作成する sdiff がある。

comm [オプション] ファイル1 ファイル2　　coreutils

/usr/bin　　stdin　　stdout　　- file　　-- opt　　--help　　--version

comm は、2 つのソートされたファイルを比較し、以下のようなタブ文字によって区切られた 3 つのコラムを生成するコマンドである。

1. ファイル 1 で現れるがファイル 2 では現れない全ての行
2. ファイル 2 で現れるがファイル 1 では現れない全ての行
3. 両方のファイルに現れる全ての行

例えば、file1 と file2 が以下のような行を含んでいたと想定する。

```
file1:    file2:
apple     baker
baker     charlie
charlie     dark
```

このとき、comm は、以下を出力する。

```
$ comm file1 file2
apple
    baker
    charlie
  dark
```

オプション

-1　　1 番目の列を非表示
-2　　2 番目の列を非表示

This PDF is prepared for tools@maruhouse.org, personal use only.

-3　　3番目の列を非表示

cmp [オプション] ファイル1 ファイル2 [オフセット1] [オフセット2]

diffutils

/usr/bin　　stdin　　stdout　　-file　　--opt　　--help　　--version

cmpは、2つのファイルを比較するコマンドである。内容が同一の場合、cmpは何も通知しない。そうでない場合、cmpは最初に異なる場所を表示する。

```
$ cmp myfile yourfile
myfile yourfile differ: char 494, line 17
```

デフォルトの場合、cmpは内容の表示はせず、異なる行のみ表示する。このためバイナリファイルを比較する場合にも適しており、テキストファイルに対する操作が容易なdiffとは対称的である。

通常、cmpはそれぞれの行の1文字目から比較を行う。しかし、オフセットを指定すれば、それに従って動作する。

```
$ cmp myfile yourfile 10 20
```

この場合、myfileの10番目の文字と、yourfileの20番目の文字から比較を開始する。

オプション

-l　　長い出力。文字毎の比較内容を表示

```
$ cmp -l myfile yourfile
   494 164 172
```

結果はオフセット行が494（10進数表記）であり、myfileでは“t”（バイト表記で164）であるが、yourfileでは“z”（バイト表記で172）であることを示している

-s　　サイレント出力。ファイルが同一であれば0、異なれば1という返り値のみを返す。その他は何も表示しない（比較が何らかの理由で失敗した場合、0以外の返り値となる）

This PDF is prepared for tools@maruhouse.org, personal use only.

md5sum ファイル | --check ファイル　coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

md5sumは、MD5アルゴリズムを用いて与えられたファイルのチェックサムを表示するプログラムである。チェックサムの桁数は 32 バイトである（詳細は、http://www.faqs.org/rfcs/rfc1321.html を参照のこと）。

```
$ md5sum myfile
dd63602df1cceb57966d085524c3980f  myfile
```

異なる 2 つのファイルが、同一の MD5 チェックサムであることは事実上あり得ない。このため、2 つのファイルが同一であるかをチェックサムを用いて確認する方法は非常に信頼できる。

```
$ md5sum myfile1 > sum1
$ md5sum myfile2 > sum2
$ diff -q sum1 sum2
Files sum1 and sum2 differ
```

--check オプションを用いる比較方法を以下に示す。

```
$ md5sum file1 file2 file3 > mysum
$ md5sum --check mysum
file1: OK
file2: OK
file3: OK
$ echo "new data" > file2
$ md5sum --check mysum
file1: OK
file2: FAILED
file3: OK
md5sum: WARNING: 1 of 3 computed checksums did NOT match
```

md5sumと同様のプログラムとして、sumとcksumがあげられる。両者のプログラムは、チェックサムの計算に MD5 とは異なるアルゴリズムを利用する。sum は、他の Unix システム、特に BSD Unix のデフォルト、あるいは System V Unix（-s オプション）と互換性がある。cksum は CRC チェックサムを生成する。

```
$ sum myfile
12410     3
$ sum -s myfile
47909 6 myfile
```

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ cksum myfile
1204834076 2863 myfile
```

最初の整数値は、チェックサムであり、2つめの数値はブロック数である。これらのチェックサムは桁数が小さいため、複数のファイルが、偶然に同一のチェックサムを持つ場合があるので、信頼性が低くなる。これに対してmd5sumは、はるかに高い信頼性を持つと言うことができる。

4.13 ディスク操作

df	マウントされたファイルシステムの利用できる容量を表示
mount	ディスクパーティションをマウントしてアクセス可能状態にする
umount	ディスクパーティションをアンマウント(アクセス不能にする)
fsck	ディスクパーティションのエラーをチェック
sync	ディスクのキャッシュを全てのディスクに書き込みの同期

Linuxシステムは、複数のディスクやディスクパーティションを扱うことができる。本書では、これまでディスク、パーティション、ファイルシステム、ボリューム、ディレクトリなどを取り上げてきた。以下では、これらの意味についてより厳密に述べる。

ディスクは、パーティションという単位に分割できるハードウェアである。このパーティションは、独立したストレージとして扱うことができ、通常Linuxシステム上では、/devディレクトリ以下のスペシャルファイルとして表示される。例えば、/dev/hda7はマスターに接続された、IDEディスク上のパーティションであることを示す。

/dev以下のデバイス名と内容を紹介する。

hda	プライマリIDE busに接続されたマスターHDD。パーティションはhda1、hda2...と続く
hdb	プライマリIDE busに接続されたスレーブHDD。パーティションはhdb1、hdb2...と続く
hdc	セカンダリIDE busに接続されたマスターHDD。パーティションはhdc1、hdc2...と続く
hdd	セカンダリIDE busに接続されたスレーブHDD。パーティションはhdd1、hdd2...と続く
sda	1つめのSCSI HDD。パーティションはsda1、sda2...と続く

This PDF is prepared for tools@maruhouse.org, personal use only.

sdb　2つめのSCSI HDD。パーティションはsdb1、sdb2...と続く。同様にsdc、sdd...と続く

ht0　プライマリIDE busに接続された、IDEテープドライブ(ht1、ht2...と続く)。自動的に巻き戻しを行う

nht0　プライマリ IDE bus に接続された、IDE テープドライブ(nht1、nht2... と続く)。自動的に巻き戻しを行わない(not-rewind)

st0　1つめの SCSI テープドライブ(st1, st2... と続く)

scd0　1つめの SCSI CD-ROM ドライブ(scd1、scd2... と続く)

fd0　1つめのフロッピーディスクドライブ(fd1、fd2...と続く)。通常、/media/floppyにマウントする

パーティションにファイルを保存できるようにするには、「フォーマット」作業を行うことにより、ディスクにファイルシステムを書き込む必要がある。ファイルシステムは、ファイルの扱い方を定義する。例として、ext3(Linuxのジャーナルファイルシステム。Fedoraのデフォルト)、vfat (Microsoft の Windows ファイルシステム)を取り上げる。フォーマットは、通常 Linux のインストール時に実行される。

ファイルシステムの作成後は、空のディレクトリにマウントして、パーティションが使用可能となる。

例えば、ディレクトリ /mnt/win にWindowsのファイルシステムをマウントすると、これはシステムのディレクトリツリーの一部になり、/mnt/win/myfileのようなファイルの作成、編集が可能になる。メンテナンスなどの場合に、ファイルシステムにアクセス出来なくするためにアンマウントすることも可能である。通常、ハードディスクのマウントは、ブート時に自動的に行われる。

df [オプション] [ディスクデバイス|ファイル|ディレクトリ]

coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

df(disk free)は、指定したディスクのパーティションサイズ、使用容量、残容量を表示するコマンドである。引き数にファイルやディレクトリを指定すると、dfコマンドは、そのファイルやディレクトリが属するディスクデバイスに関して表示を行う。引き数を指定しない場合は、df はマウントされた全てのファイルシステムを表示する。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ df
Filesystem    1k-blocks      Used Available Use% Mounted on
/dev/hda7       1011928    225464    735060  24% /
/dev/hda9        521748    249148    246096  51% /var
/dev/hda8       8064272   4088636   3565984  54% /usr
/dev/hda10      8064272   4586576   3068044  60% /home
```

オプション

-k
-m　キロバイト単位(デフォルト)、メガバイト単位で全てのサイズを表示

-B *N*　定義したブロック単位でサイズを表示。1 ブロック = *N* バイトである(デフォルト = 1024 バイト)

-h
-H　可読可能な形式で表示。それぞれのサイズに対して最も適当な単位を選択。例えば、2 つのディスクが、それぞれ 1 ギガバイトと 25 キロバイトである場合、df-h は 1G、25K と表示。-H オプションは、1000 の倍数を使用するが -h オプションは 1024 の倍数を使用

-l　ローカルファイルシステムのみを表示。ネットワークファイルシステムを非表示

-T　ファイルシステム名(ext2、vfat など)をあわせて表示

-t *type*　指定したファイルシステムのパーティションのみ表示

-x *type*　指定したファイルシステム以外のパーティションのみ表示

-i　i ノード形式。ディスクのブロックの代わりに、ファイルシステムに対する合計、使用、空きの i ノードを表示

mount [オプション] デバイス|ディレクトリ　mount

/bin　stdin　stdout　- file　-- opt　--help　--version

mountは、ストレージデバイスをアクセス可能にするコマンドである。最もよく行う操作は、(/dev/hda1のような)ディスクデバイスを、(/mnt/mydirなどの)存在するディレクトリにマウントしてアクセス可能にすることである。

```
# mkdir /mnt/mydir
# mount /dev/hda1 /mnt/mydir
# df /mnt/mydir
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/hda1        1011928  285744    674780  30% /mnt/mydir
```

mountには、非常にたくさんのオプションがある。本書では最も基本的なもののみを取り上げる。

This PDF is prepared for tools@maruhouse.org, personal use only.

通常、mountを実行すると、どのようにマウントを行うか確認するために、/etc/fstabファイルの読み込みを行う。例えば、mount/usrというコマンドを実行する場合、mountコマンドは/etc/fstabにある“/usr”の行を調べる。ここでは、/etc/fstabには以下のように指定されていたとする。

```
/dev/hda8    /usr    ext3    defaults    1    2
```

ここでmountコマンドは、ディスクデバイス/dev/hda8を、Linuxのext3ファイルシステムとして/usrにマウントする、ということを行う。通常mountコマンドは、スーパーユーザ権限で実行する。しかし、フロッピーやCD-ROMなどの、頻繁に入れ換えを行うデバイスについては、任意のユーザがマウント／アンマウントできるように、/etc/fstabで設定されていることが多い。

```
$ mount /media/cdrom
$ mount /media/floppy
```

umount [オプション] [デバイス|ディレクトリ]　mount

/bin　stdin　stdout　- file　-- opt　--help　--version

umountは、mountコマンドと反対の動作を行うコマンドである。すなわち、ディスクのパーティションを利用できない状態にする。例えば、CD-ROMディスクをマウントした場合、umountコマンドを実行しない限りCD-ROMを取り出すことはできない。

```
$ umount /media/cdrom
```

このように、リムーバブルメディアを取り出すためには、アンマウントする必要がある。umountを実行しない場合、ファイルシステムを壊す可能性がある。マウントされている全てのデバイスをアンマウントするには以下のコマンドを実行する。

```
# umount -a
```

なお、使用中のファイルシステムをアンマウントしてはならない。使用中のファイルシステムをumountした場合、安全のためコマンドの実行が拒否される。

This PDF is prepared for tools@maruhouse.org, personal use only.

fsck [オプション] [デバイス]

e2fsprogs

/sbin　stdin　stdout　- file　-- opt　--help　--version

fsck(filesystem check) は、Linuxのディスクパーティションを検証してエラーを見つけ出し、要求に応じてエラー修復を行うコマンドである。fsckは、システム起動時に自動的に実行される。しかし、必要ならば手動で実行することも可能である。一般的に、fsckを実行する前には、デバイスをアンマウントして、他のいかなるプログラムも実行されないようにしておかなければならない。

```
# umount /dev/hda10
# fsck -f /dev/hda10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/home: 172/1281696 files (11.6% non-contiguous), 1405555/2562359 blocks
```

/sbin 以下には、“fsck”で開始される名前のファイルシステムをチェックするプログラムがいくつか存在しており、fsckはその中の1つである。これらコマンドでは、それぞれ1つのファイルシステムのタイプのみがサポートされている。以下のコマンドで、コマンドがサポートしているファイルシステムを表示できる。

```
$ ls /sbin/fsck.* | cut -d. -f2
```

オプション

-A	/etc/fstab に記載されている全てのディスクを順番にチェック
-N	実行が想定された内容のみの表示。実際には実行しない
-r	対話的にファイルシステム修復の実行
-a	自動的にエラーを修復(ファイルシステムを完全に理解している場合のみ使用すること。そうでない場合ファイルシステムの破壊を招くこともありうる)

sync

coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

syncは、全てのディスクキャッシュをディスクに書き込みするコマンドである。通常、カー

This PDF is prepared for tools@maruhouse.org, personal use only.

ネルはバッファ内の読み込み、書き込み、iノードの変化、その他のディスク関連の情報を蓄積している。このように、syncコマンドは、これらの変更点をディスクに書き込む。通常はこのコマンドを実行する必要はない。しかし、マシンがクラッシュするような危険な作業をする前にはsyncを実行するとよい。これを実行することにより、マシンを破壊する可能性を低くすることができる。

4.14 パーティション化とディスクのフォーマット

パーティションやフォーマットなどのディスク関連の操作が、Linux システムを複雑にしているといっても過言ではない。ここでは、関連するプログラムの概要を述べる（詳細は、manpage を参照のこと）。

parted、disk、sfdisk	ハードディスクにパーティションを区切る。この3つのどれを用いてもこの作業は可能である。異なるのはユーザインターフェースのみ
mkfs	ハードディスクをフォーマットする。新たにファイルシステムを作成
floppy	フロッピーディスクをフォーマット

4.15 バックアップとリモートディスク

mt	テープドライブを制御
dump	ディスクパーティションをテープに書き込む
restore	ダンプ結果を書き戻す
tar	テープ† のアーカイブの読み込み / 書き込み
cdrecord	CD-R を作成
rsync	他のデバイスやホストにファイルのミラーを作成

Linux 上の大切なファイルをバックアップする方法は多数存在する。

- テープドライブにコピーする
- CD-R を作成（書き込み）
- リモートマシンにミラーリングする

† 訳注：テープは元々の意味でありテープ以外にも使用可能

This PDF is prepared for tools@maruhouse.org, personal use only.

Linuxでは通常、バックアップに用いるテープドライブは、IDEドライブの場合は/dev/ht0、SCSIドライブ(もしくはide-scsiエミュレーションを用いたIDEドライブ)の場合は/dev/st0になる。/dev/tapeに使用するデバイスへのリンクを作成することは一般的である。

```
$ ln -s /dev/ht0 /dev/tape
```

本書ではバックアップに用いるLinuxコマンド全てについて取り上げていない。もちろん、tarよりもcpioを好む読者もいるかもしれない。また、ローレベルのディスクコピーコマンドddを用いてバックアップを行う場合もある。これらのコマンドを用いてバックアップを行いたい場合はマニュアルページ参照のこと。

mt [-f デバイス] コマンド

mt-st

/bin　stdin　stdout　- file　-- opt　--help　--version

mt(magnetic tape)は、テープドライブの巻き戻し、前後のスキップ、リテンションなど、テープドライブの簡単な操作を実行するコマンドである。よく用いる操作を以下に示しておこう。

status	ドライブの状態を表示
rewind	テープの巻き戻し
retension	テープのリテンション(テープを巻き戻した後、リールの最後まで進め、さらにもう一度巻き戻す)
erase	テープの消去
offline	テープドライブをオフラインにする
eod	データの終端に移動

例えば、以下のようなコマンドを考える。

```
$ mt -f /dev/tape rewind
```

mtを使用すると、ファイルからファイルへ、記録部分から記録部分へなど、テープ中の任意の場所に移動することが可能である。しかし、tarやrestoreのような、データの読み込み/書き出すコマンドをテープドライブに対して使用することも可能である。

This PDF is prepared for tools@maruhouse.org, personal use only.

dump [オプション] パーティションまたはファイル dump

/sbin stdin stdout - file -- opt --help --version

dumpは、ディスクのパーティション全体や選択したファイルを、テープなどのバックアップメディアに書き出すコマンドである。このコマンドは、完全バックアップをサポートする。また、バックアップが必要なファイルはどれであるか(すなわち、最近のバックアップから変更されたファイルはどれであるか)を自動的に判断するため、差分バックアップをサポートする。バックアップメディアからファイルを復元するには、restoreコマンドを使用する。指定したファイルシステム(ここでは/usrとする)の、完全バックアップをテープドライブ(/dev/tape) に実行するには、-0(ゼロ)と -u オプションを用いる。

```
# dump -0 -u -f /dev/tape /usr
```

これは、レベルゼロダンプと呼ばれる。-uオプションは、バックアップ内容を、ファイル/etc/dumpdates に記録する。差分バックアップはレベル 1 から 9 まで存在する。
レベル *i* バックアップは、最後のレベル *i-1* バックアップ以降の、新しくかつ更新された全てのファイルをバックアップする。

```
# dump -1 -u -f /dev/tape /usr
```

dump コマンドは、(マウントされている)使用中のファイルシステムでダンプを実行してはならない。可能ならば dump の実行前にそのファイルシステムをアンマウントする。

restore [オプション] [ファイル] dump

/sbin stdin stdout - file -- opt --help --version

restoreは、dumpコマンドによって作成されたバックアップを元に戻すコマンドである。このコマンドは、ディスクにファイルを復元し、ディスク上のファイルに対して比較するなどの動作を行う。restoreの-iオプションを用いることで、インタラクティブに動作を行う。このオプションは、テープの内容をファイルシステムと同様に表示し、ファイルやディレクトリを選択して、最後に復元を行う。

```
# restore -i -f /dev/tape
```

restore は、以下に示すコマンドを用いる。

This PDF is prepared for tools@maruhouse.org, personal use only.

help	ヘルプメッセージを表示
quit	ファイルを復元せずにプログラムを終了
cd *directory*	シェルの cd コマンドと同様、ダンプ中のカレントディレクトリを設定
ls	lsコマンドと同様、ダンプ中のカレントディレクトリ内の全てのファイルを表示
pwd	pwd コマンドと同様、ダンプ中のカレントディレクトリを表示
add	「抽出リスト」に復元したいファイルやディレクトリを追加する。引き数を指定しない場合、カレントディレクトリとそのディレクトリ以下の全ファイルをリストに追加
add *filename*	抽出リストにファイル *filename* を追加
add *dir*	抽出リストにディレクトリ *dir* を追加
delete	add の反対の動作。抽出リストからファイルを削除する。引き数を指定しない場合、deleteは抽出リストからカレントディレクトリ(以下の全ての内容)を削除
delete *filename*	抽出リストからファイル *myfile* を削除
delete *dir*	抽出リストからディレクトリ *dir* を削除
extract	抽出リストに記載されている全ファイルを復元する †

restore コマンドは、非インタラクティブ処理(バッチ処理)でも動作する。

restore -x	テープから存在するファイルシステムの全てを復元(最初に希望するファイルシステムのルート位置に cd コマンドを用いて移動する)
restore -r	テープから新しくフォーマットされたディスクパーティションへ全てを復元(最初に希望するファイルシステムのルート位置にcdコマンドを用いて移動する)
restore -t	ダンプの内容を表示
restore -C	オリジナルのファイルシステムとダンプの内容を比較

† 訳注:バックアップに複数のテープを使用している場合、最後に使用したテープから開始し先頭へと逆に進む

This PDF is prepared for tools@maruhouse.org, personal use only.

tar [オプション] [ファイル] tar

/bin stdin stdout - file -- opt --help --version

tar(tape archive)は、テープドライブからファイルを読み書きを行うコマンドである。

```
$ tar -cf /dev/tape myfile1 myfile2
```

tarコマンドは、テープ以外にもtarファイルの作成、展開が行える。これは、LinuxやUnixシステムでのファイルのパッケージングの標準的な手段である。

```
$ tar -czvf myarchive.tar.gz mydir      myarchive.tar.gzファイルを作成
$ tar -tzvf myarchive.tar.gz            myarchive.tar.gzの内容を表示
$ tar -xzvf myarchive.tar.gz            myarchive.tar.gzを展開
```

コマンドライン上でファイルを指定すると、そのファイルのみ展開される。

```
$ tar -xvf /dev/tape file1 file2 file3
```

ファイルを指定しない場合、アーカイブの全ファイルが展開される。

オプション

-c	アーカイブを作成。コマンドラインで入力ファイルやディレクトリを指定
-r	アーカイブにファイルを追加
-u	アーカイブに新規／更新されたファイルを追加
-A	アーカイブの最後に、他のアーカイブ(例えばtar ファイル)を追加。例：tar -A -f /dev/tape myfile.tar
-t	アーカイブのリストを表示
-x	アーカイブからファイルを展開
-f *file*	アーカイブとして読み込む／書き出すファイルを指定。これには(/dev/tapeのような)デバイスファイルを指定する方法と、tarファイルを作成／展開したい場合に、通常のファイルを指定する方法が存在
-d	ファイルシステムに対するアーカイブの差分を取得
-z	gzipを用いてデータを(書き出しながら)圧縮、あるいは(読み込みながら)解凍
-j	bzip2 を用いてデータを(書き出しながら)圧縮、あるいは(読み込みながら)解凍

This PDF is prepared for tools@maruhouse.org, personal use only.

-Z	compress を用いてデータを(書き出しながら)圧縮、あるいは(読み込みながら)解凍
-b *N*	*N**512 バイトのブロックサイズを使用
-v	冗長モード。その他の情報も合わせて表示
-h	シンボリックリンクが示すファイルを扱う
-l	ローカルなファイルシステムに限定してアーカイブを作成
-p	ファイル展開時に、元のパーミッションおよび所有権を復元

cdrecord [オプション] トラック　cdrecord

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

cdrecord は、Linux の ide-scsi エミュレーションを用いて、SCSI あるいは IDE の CD-R でディスクの書き込みを行うコマンドである。Linux、Windows、そしてMacintosh上で読み出し可能な CD-ROM を作成するには、以下を実行する。

1. CD を書き込むデバイスを表示する。

```
$ cdrecord --scanbus
...
0,0,0     0) *
0,1,0     1) *
0,2,0     2) *
0,3,0     3) 'YAMAHA  ' 'CRW6416S  ' '1.0d' Removable CD-ROM
...
```

   この場合のデバイスは 0,3,0 である。
2. CD-RあるいはCD-RWディスクを書き込む速度を確認する。CD-Rの書き込み速度が 6x と表示された場合、6 倍速ドライブであることを示す。
3. ディレクトリに書き込むファイル(ここでは、dir とする)を置く。CD 上の配置は自由である。ディレクトリ dir 自身はコピーされず、中身のみでコピーされる。
4. CD 書き込みを実行するには、以下のコマンドを実行する。

```
$ DEVICE="0,3,0"
$ SPEED=6
$ mkisofs -R -l dir > mydisk.iso
$ cdrecord -v dev=${DEVICE} speed=${SPEED} mydisk.iso
```

   使用しているシステムが充分に高速な場合、以下のように2つのコマンドをパイプで

This PDF is prepared for tools@maruhouse.org, personal use only.

繋いで実行することが可能である。

```
$ mkisofs -R -l dir \
  | cdrecord -v dev=${DEVICE} speed=${SPEED} -
```

cdrecord コマンドは、音楽 CD を作成することも可能である。もちろん、このコマンドを使用する代わりに xcdroast のような GUI を提供するプログラムを利用するのもよいだろう。xcdroast は、cdrecord を用いた GUI プログラムである。

rsync [オプション] 転送元 転送先 rsync

/usr/bin stdin stdout - file -- opt --help --version

rsync は、ファイルをコピーするコマンドである。このコマンドは、ファイルのパーミッションやその他の属性を含めて、全て正確なコピー(ミラーリング)を行うため、データを完全にコピーできる。

rsync はネットワーク経由、もしくは同一のマシン内でも実行できる。rsync は、多くの使用方法、および50以上のオプションが提供されているが、ここではバックアップを行う際の代表的なコマンド例のみを取り上げる。

ディレクトリ D1 と内容を、同じマシンの他のディレクトリ D2 にミラーする場合、以下を実行する。

```
$ rsync -a D1 D2
```

ディレクトリD1の内容を、ネットワーク経由で他のホストserver.example.com(アカウント名は smith、ディレクトリ ~/D2 にコピーする。盗聴を防ぐために SSH を用いて安全性を確保することを推奨する。

```
$ rsync -a -e ssh D1 smith@server.example.com:~/D2
```

オプション

-o ファイルの所有権をコピー(リモートホストでスーパーユーザの権限が必要)

-g ファイルのグループ所有権をコピー(リモートホストでスーパーユーザの権限が必要)

-p ファイルのパーミッションをコピー

This PDF is prepared for tools@maruhouse.org, personal use only.

-t	ファイルのタイムスタンプをコピー
-r	ディレクトリを再帰的にコピー。ディレクトリ内容全てコピー
-l	シンボリックリンクが指し示すファイルではなく、その実体をコピー
-D	デバイスファイルのコピーを許可(スーパーユーザーのみ)
-a	ミラーリング：オリジナルファイルの属性全てコピー。これはオプション -Dgloprt と同等
-v	冗長出力。ファイルのコピー時に、実行内容を表示。--progressオプションによって、進捗状況の表示が可能
-e *command*	安全性のために ssh などのプログラムを指定

4.16 ファイルの印刷

lpr	ファイルを印刷
lpq	プリントのキューを表示
lprm	プリントジョブをキューから削除

Linux には、CUPS と LPRng という 2 つの印刷システムがある。Fedora では、CUPS を採用している。

両システムは、lpr、lpq、lprm という同名のコマンドを使用する。しかし、このコマンドはシステムがCUPSとLPRngのどちらが使用しているかによってオプションが異なる。一般的な話として、以下では両システムで動く共通のオプションを述べておこう。

Fedora では、プリンターを使用するために以下のコマンドを実行する。

```
# redhat-config-printer
```

この後、コマンドの指示に従う。

lpr [オプション] [ファイル]　cups

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

lpr(line printer)は、プリンタにファイルを送信するコマンドである。

```
$ lpr -P myprinter myfile
```

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-P *printername* redhat-config-printer コマンドを用いて設定したプリンタ、*printername* にファイルを送信

-# *N* ファイルを *N* 部プリント

-J *name* (システムがプリントカバーページが設定されている場合)カバーページで印刷するジョブ名、*name* を設定

lpq [オプション] cups

/usr/bin stdin **stdout** - file -- opt --help --version

lpq(line printer queue)は、印刷を待っている全てのプリントのジョブを表示するコマンドである。

オプション

-P *printername* プリンタ *printername* のキューを出力

-a 全てのプリンタのキューを出力

-l 冗長表示。長いフォーマットで情報を表示

lprm [オプション] [ジョブ ID] cups

/usr/bin stdin **stdout** - file -- opt --help --version

lprm(line printer remove) は、1つあるいは複数の印刷ジョブを中止するコマンドである。中止したい印刷のジョブ ID 番号(以下の lprm コマンドで指定している 61 と 78) を参照するには、lpq コマンドを実行する。その後、以下の lprm コマンドを実行する。

```
$ lprm -P printername 61 78
```

ジョブID番号を指定しない場合、最新の印刷ジョブがキャンセルされる(他ユーザのジョブは、スーパーユーザのみキャンセル可能)。-Pオプションは、印刷キューを中止するプリンタ名を指定する。

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

5章 プロセス管理

5.1 プロセスの表示

ps	プロセスを表示
uptime	システムの負荷を表示
w	全ユーザに対してアクティブなプロセスを表示
top	負荷が大きいプロセスをインタラクティブに表示
xload	X Window System 上でシステムの負荷グラフを表示
free	空きメモリを表示

プロセスは、Linuxシステムにおけるジョブの単位である。起動したプログラムは、1つあるいは複数のプロセスとなり、Linuxはプロセスを表示、操作するコマンドを提供している。全てのプロセスは、プロセスID（PID）と呼ばれる番号が付与される。

プロセスは、ジョブと同一ではない（詳細は「3.3　ジョブコントロール」を参照のこと）。プロセスは、オペレーティングシステムの一部であるのに対して、ジョブは実行中のシェルのみに通知されているためである。実行中のプログラムは、1つもしくは複数のプロセスで構成されており、ジョブはシェルにコマンドとして実行された1つ、もしくは複数のプログラムから構成される。

ps [オプション]　　procps

/bin　　stdin　stdout　- file　-- opt　--help　--version

psは、実行しているプロセスの情報や、他のユーザのプロセスを表示するコマンドである。

```
$ ps
  PID TTY          TIME CMD
 4706 pts/2    00:00:01 bash
15007 pts/2    00:00:00 emacs
16729 pts/2    00:00:00 ps
```

This PDF is prepared for tools@maruhouse.org, personal use only.

psコマンドは少なくとも80のオプションを持っており、ここでは、いくつかの有用なオプションのみを取り上げる。Linuxのpsコマンドのオプションが多数提供されており、それが勝手で気まぐれであるように見えるのは、psコマンドが、GNU psや他のいくつかのUnixのps コマンドと互換性を持たせるようにしているためである。

プロセスを表示するには、以下を実行する。

```
$ ps -ux
```

ユーザ smith の全プロセスを表示する。

```
$ ps -U smith
```

指定したプログラムに関して、すべてを表示する。

```
$ ps -C プログラム名
```

ターミナル *N* のプロセスを表示する。

```
$ ps -tN
```

特定のプロセス(1と2と3505)を表示する。

```
$ ps -p1,2,3505
```

すべてのプロセスをスクリーン幅に合わせて表示する。

```
$ ps -ef
```

すべてのプロセスを広い幅で表示する。

```
$ ps -efww
```

すべてのプロセスを階層化して表示する。親とその子プロセスをインデント化して表示する。

```
$ ps -efH
```

grepや他のフィルタプログラムを用いて、psの出力から特定の情報を抽出することも可能である。

This PDF is prepared for tools@maruhouse.org, personal use only.

uptime

procps

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

uptimeは、最後の再起動からどれくらいの期間、システムが稼動しているかを表示するコマンドである。

```
$ uptime
 10:54pm  up 8 days,  3:44,  3 users,  load average: 0.89, 1.00, 2.15
```

この情報は、左から順に、現在の時間(10:54pm)、システムの稼働時間(8 days、3 hours、44 minutes: 8日と3時間44分)、ログイン中のユーザ数(3)、3つの間隔で計測されたシステムの平均負荷(1分(0.89)、5分(1.00)、15分(2.15)という)となる。平均負荷とは、一定時間内で起動の準備ができるプロセスの平均数を示す。

w [ユーザ名]

procps

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

wは、ログイン中のユーザの現在のプロセスを表示するコマンドである。wは、各ユーザの利用状況を詳細に表示する。

```
$ w
 10:51pm  up 8 days,  3:42,  8 users,
 load average: 0.00, 0.00, 0.00
USER     TTY   FROM  LOGIN@  IDLE   JCPU   PCPU  WHAT
barrett pts/0 :0    Sat 2pm 27:13m 0.07s  0.07s emacs
jones   pts/1 host1 6Sep03   2:33m 0.74s  0.21s bash
smith   pts/2 host2 6Sep03   0.00s 13.35s 0.04s w
```

先頭の行は、uptimeコマンドの表示と同様である。各列は、ユーザ名(USER)、ユーザの使用している端末(TTY)、ログイン元のホスト、もしくはX Window Systemの画面(上記の例では:0)(FROM)、ログイン時間(LOGIN@)、アイドル時間(IDLE)、CPU時間の2つの計測(JCPUとPCPU:詳細は、man wを参照のこと)、そして現在のプロセス(WHAT)を示す。特定のユーザの情報のみを表示するにはユーザ名を指定する。最も簡潔な出力を行う場合は、w -hfsを実行する。

オプション

-h　ヘッダ行を非表示

This PDF is prepared for tools@maruhouse.org, personal use only.

-f　　FROM 列を非表示

-s　　JCPU ／ PCPU 列を非表示

top [オプション]　　procps

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

topは、最もアクティブなプロセスを監視して、一定間隔(たとえば、毎秒)毎に表示を更新するコマンドである。

top は、スクリーンベースのプログラムであり、インタラクティブに表示が更新される。

```
$ top
116 processes: 104 sleeping, 1 running, 0 zombie, 11 stopped
CPU states: 1.1% user, 0.5% system, 0.0% nice, 4.5% idle
Mem: 523812K av, 502328K used, 21484K free, 0K shrd, 160436K buff
Swap:  530104K av,  0K used, 530104K free  115300K cached

PID   USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
26265 smith 10 0  1092 1092 840  R    4.7  0.2  0:00 top
    1 root   0 0   540  540 472  S    0.0  0.1  0:07 init
    2 root   0 0     0    0   0  SW   0.0  0.0  0:00 kflushd
...
```

top が起動している間のキー操作によって、top の振る舞いを変更することが可能である。その変更内容としては、更新速度の設定(s)、IDLEプロセスを隠す(i)、プロセスをkillする(k)などがある。オプションの完全なリストの表示はhを入力する。停止するにはqを入力する。

オプション

-n*N*　　*N* 回だけ更新を繰り返し、その後停止

-d*N*　　*N* 秒間隔で表示を更新

-p*N* -p*M*...　　プロセス ID が *N* と *M* であるプロセスのみを表示(最大 20 個まで指定可能)

-c　　プロセスのコマンド行と引き数をあわせて表示

-b　　バッチ形式で標準出力に表示する。書き換えを行わない。このオプションを用いて top -b -n1 > outfile と実行することにより、その時点での top の表示内容を素早く保存できる

This PDF is prepared for tools@maruhouse.org, personal use only.

xload

XFree86-tools

/usr/X11R6/bin　　stdin　stdout　-file　--opt　--help　--version

X Window System 上でシステムの負荷情報を表示するには、xload を実行する。時間ごとの(X 軸)、プロセッサの負荷(Y 軸)を表示する。

オプション

-update *N*　*N* 秒毎に表示を更新(デフォルトは 10 秒)

-scale *N*　Y軸を*N*個に分割する(デフォルトは1個)。xloadは負荷が増えた場合さらに多く分割。*N* は、最小の分割数

-hl *color*　スケール表示に *color* 色を使用

-label *X*　グラフ上にテキスト *X* を表示(デフォルトはホスト名)

-nolabel　グラフ上にテキストラベルを非表示

-jumpscroll *N*　グラフ右側の余白が無くなった場合には、左に *N* ピクセル分スクロールし、表示を続ける(デフォルトはウィンドウ幅の半分)

free [オプション]

procps

/usr/bin　　stdin　stdout　-file　--opt　--help　--version

free は、メモリの使用量をキロバイト単位で表示するコマンドである。

```
$ free
             total       used       free     shared    buffers     cached
Mem:        523812     491944      31868          0      67856     199276
-/+ buffers/cache:     224812     299000
Swap:       530104          0     530104
```

Linux kernel は、使用可能メモリを全てキャッシュとして使用する。上記の表示で使用可能 RAM の最大量は 299000 である。

オプション

-s *N*　コマンド実行後に終了せず、*N* 秒毎に表示を更新して表示

-b　バイト単位で表示

-m　メガバイト単位で表示

-t　最終行に合計の表示を追加して表示

-o　buffers/cache 行を非表示

This PDF is prepared for tools@maruhouse.org, personal use only.

5.2 プロセス制御

kill	プロセスを停止する(もしくはシグナルを送る)
nice	プログラムの優先度を指定して実行する
renice	プログラムの優先度を動作中に変更する

これらのコマンドで、一度プロセスが実行された後のプロセスの停止、再起動、削除、優先度の変更などを行える。この操作はシェルが取り扱う。ここではプロセスのkillと優先度の変更を説明する。

kill [オプション] [プロセスID] bash

shell built-in stdin **stdout** - file -- opt --help --version

killは、プロセスにシグナルを送るコマンドである。これでプロセスを終了(デフォルトの動作)、中止、一時停止、クラッシュする。プロセスの所有者だけがこのコマンドを実行できるが、スーパーユーザは全てのプロセスに対してコマンドを実行できる。

```
$ kill 13243
```

これでプロセスが停止しない場合には-KILLオプションを付ける。

```
$ kill -KILL 13243
```

これは、一見正常に停止できているように見えるが、プロセスは終了していない。動作していないプロセスに対してリソースが割り当てられたままプロセスが死んでしまうことになる。

プロセスのID番号(PID)が分からない場合は、pidofコマンドを使用する。

```
$ /sbin/pidof emacs
```

もしくはpsコマンドを用いる。

killコマンドは、大抵はシェルのビルトインコマンドであるが、/bin/killというプログラムが存在する事もある。2つのプログラムでは書式が異なるが、以下に示した書式はどちらでも使うことができる。

```
$ kill -N PID
$ kill -NAME PID
```

This PDF is prepared for tools@maruhouse.org, personal use only.

N は、シグナルの番号である。*NAME* は、シグナル名から冒頭の“SIG”を省いた名前(例えばSIGHUPならば-HUP)である。killコマンドによって送られるシグナルの一覧はkill -lで表示することができる。しかし、その出力は実行したkillコマンドによって異なる。シグナルの説明は、 man 7 signalを参照のこと。

nice [- 優先度] 実行するコマンド

coreutils

/bin stdin stdout - file -- opt --help --version

指定したプログラムを他のプログラムより低い優先度で実行したい場合、niceコマンドを用いる。大きいプログラムを優先度 7 で実行する例を以下に示す。

```
$ nice -7 sort VeryLargeFile > outfile
```

優先度を指定しない場合には10が用いられる。デフォルトの優先度(niceを使用しない場合の優先度)は、nice コマンドを単独で実行すると表示される。

```
$ nice
0
```

スーパーユーザの場合は、優先度を上げることができる(小さい数値を指定する)。

```
$ nice --10
```

これは「ダッシュ マイナス 10」であり -10 を指定した場合である。プロセスの nice 値を参照するには、ps コマンドを実行する。NI のカラムが nice の値である。

```
$ ps -o pid,user,args,nice
```

renice 優先度 [オプション] プロセスID

util-linux

/usr/bin stdin stdout - file -- opt --help --version

niceは、実行時に優先度を指定してプログラムを起動するコマンドである。reniceは既に実行中のプロセスの優先度を変更するコマンドである。プロセス ID 28734 の nice 値を大きくする(優先度を下げる)場合、以下のように実行する(優先度は 5)。

```
$ renice +5 -p 28734
```

一般ユーザは、優先度を下げる(数値を大きくする)操作しか許可されていない。スーパー

This PDF is prepared for tools@maruhouse.org, personal use only.

ユーザのみ、優先度を上げること(数値を小さくすること)ができる。優先度は-20から+20の間が指定できるが、あまりマイナス方向に大きく指定するとシステムプロセスの実行を妨害してしまうことになるため注意すること。

オプション

-p *pid*　プロセスIDの指定。-pは省略可能で単にプロセスIDのみを指定することができる（例：renice +5 28734）

-u *username*　指定したユーザの全てのプロセスの nice 値を変更することができる

This PDF is prepared for tools@maruhouse.org, personal use only.

6章 ユーザ管理

6.1 ユーザの表示

logname	login 名の表示
whoami	現在の実行中ユーザを表示
id	ユーザ ID と所属グループの表示
who	login ユーザの表示(詳細表示)
users	login ユーザの表示(詳細表示)
finger	ユーザ情報の表示
last	最後に login していたユーザを表示
printenv	ユーザの環境情報を表示

あなたは誰？　この答えはシステムだけが知っている。これらのコマンドはユーザ名、login 時間、環境のプロパティ(設定)など全ての情報を表示する。

logname

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

logname は、login 名を表示するコマンドである。とても単純なコマンドではあるが、シェルスクリプトを利用する上で大変便利なコマンドである。

```
$ logname
smith
```

whoami

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

whoamiは、現在の実行中のユーザを表示するコマンドである。なお、suコマンドを使用し

This PDF is prepared for tools@maruhouse.org, personal use only.

た場合にはlogin名(lognameコマンドの出力)と異なるので注意すること。以下、lognameコマンドとwhoamiコマンドの出力を比較した結果を示す。

```
$ logname
smith
$ whoami
smith

$ su
Password: ********
# logname
smith
# whoami
root
```

id [オプション] [ユーザ名]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

すべてのユーザには個別のユーザID番号が割り当てられており、またグループも個別のグループID番号が割り振られる。idは、ユーザが所属するユーザとグループのID番号を表示するコマンドである。

```
$ id
uid=500(smith) gid=500(smith) groups=500(smith),6(disk),490(src),501(cdwrite)
```

オプション

-u　ユーザIDを表示して終了

-g　グループIDを表示して終了

-G　ユーザの所属するグループを表示

-n　ユーザとグループを数字のIDではなく名前で表示。-u、-g、もしくは-Gと同時に使用する。例えば、id -Gnは、groupsコマンドと同等の結果を出力

-r　ユーザ／グループでなく実ユーザ/グループを表示する。-u、-g、もしくは-Gと同時に使用

who [オプション] [ファイル]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

whoは、ログインしているユーザを全て表示するコマンドである。1行に1ログインユーザを表示する。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ who
smith    :0       Sep  6 17:09
barrett  pts/1    Sep  6 17:10
jones    pts/2    Sep  8 20:58
jones    pts/4    Sep  3 05:11
```

who は、通常 /var/run/utmp の情報を表示に用いるコマンドである。引き数により、他のファイル名を指定できる。例えば、/var/log/wtmpを指定すると過去にログインしたユーザを参照できる。また、/var/log/btmpを指定するとログインに失敗したユーザを参照できる。

オプション

-H	1 行目に説明を表示
-l	リモートからログインしているユーザは、ホスト名を付加して表示
-u	ユーザのアイドル(idle)時間を表示
-T	ターミナルに書き込み可能かを表示(mesgコマンドを参照)。+はmesg y、-はmesg n、?は不明
-m	自ユーザの情報のみを表示。現在使用中のターミナルを表示
-q	ユーザ名とユーザ数のみを表示。usersコマンドに類似しているが、異なるのはユーザ数が表示される点

users [ファイル名] coreutils

/usr/bin stdin stdout - file -- opt --help --version

usersは、現在ログイン中のユーザ名を短い形式で表示するコマンドである。1ユーザが複数のシェルを起動している場合には、複数表示される。

```
$ users
barrett jones smith smith smith
```

usersコマンドもwhoコマンドと同様、デフォルトでは/var/log/utmpファイルを使用する。引き数を指定することにより、他のファイルを使用することができる。

finger [オプション] [ユーザ名 @[ホスト]] finger

/usr/bin stdin stdout - file -- opt --help --version

finger は、ユーザ情報を簡潔な形式で表示するコマンドである。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ finger
Login     Name            Tty      Idle  Login Time
smith     Sandy Smith     :0             Sep  6 17:09
barrett   Daniel Barrett  :pts/1     24  Sep  6 17:10
jones     Jill Jones      :pts/2         Sep  8 20:58
```

ユーザ名を指定すると長い形式で表示する。

```
$ finger smith
Login: smith            Name: Sandy Smith
Directory: /home/smith            Shell: /bin/bash
On since Sat Sep  6 17:09 (EDT) on :0
Last login Mon Sep  8 21:07 (EDT) on pts/6 from localhost
No mail.
Project:
Enhance world peace
Plan:
Mistrust first impulses; they are always right.
```

引き数のユーザの部分には「ユーザ名 @ ホスト名」の形式で localhost や remote host のユーザ名を指定する。リモートホストがfingerコマンドに返答するのは、これが設定されている場合のみである。

オプション

-l	詳細表示
-s	省略表示
-p	Project と Plan を表示しない。この情報は各ユーザの ~/.project と ~/.plan から読み込みを実行

last [オプション] [ユーザ] [*tty*]

SysVinit

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

last は、ログイン履歴を時間で逆順に表示するコマンドである。

```
$ last
barrett pts/3  localhost Mon Sep 8 21:07 - 21:08 (00:01)
smith   pts/6  :0        Mon Sep 8 20:25 - 20:56 (00:31)
barrett pts/4  myhost    Sun Sep 7 22:19 still logged in
...
```

ユーザ名や tty を指定して、出力を限定することができる。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-N	最後の *N* 行を表示。*N* は正の整数
-i	ホスト名でなく IP アドレスで表示
-R	ホスト名を非表示
-x	システムのシャットダウン時間とランレベルが変更された時間の双方を表示（例：シングルーユーザモードからマルチユーザモードに変更された時間）
-f *filename*	/var/run/utmp以外のファイルから情報を読み込む。詳細は、who コマンドを参照

printenv [環境変数] coreutils

/usr/bin stdin stdout - file -- opt --help --version

printenv は、シェルが設定している全ての環境変数を表示するコマンドである。

```
$ printenv
HOME=/home/smith
MAIL=/var/spool/mail/smith
NAME=Sandy Smith
SHELL=/bin/bash
…
```

指定した変数だけを表示することもできる。

```
$ printenv HOME SHELL
/home/smith
/bin/bash
```

6.2 アカウントの操作

useradd	アカウントの作成
userdel	アカウントの削除
usermod	アカウントの変更
passwd	パスワードの設定
chfn	個人情報の設定
chsh	シェルの変更

Fedora のインストーラは、2種類のユーザアカウントを作成するように通知し、その1つ

This PDF is prepared for tools@maruhouse.org, personal use only.

はスーパーユーザ、もう1つは一般ユーザ(まさにあなた自身)のアカウントである。もちろん、他のユーザも同様に作成することができる。

ユーザアカウントの作成は重要な仕事であり、この作業を軽々しく実施してはならない。この理由は、全てのアカウントが潜在的にシステムへの侵入の脅威があるためである。全てのユーザは、意味を持たない文字列であり、すなわち推測が困難なパスワードをつけ、定期的に変更すべきである。

useradd [オプション] ユーザ名 shadow-utils

/usr/bin stdin stdout - file -- opt --help --version

useradd は、スーパーユーザが新しいユーザアカウントを作成するコマンドである。

```
# useradd smith
```

useradd -D を実行すれば、デフォルト設定が参照できる。このコマンドは、デフォルトではカスタマイズされていないため、詳細設定はオプションで指定する必要がある。以下に例を示す。

```
# useradd -d /home/smith -s /bin/bash -g users smith
```

オプション

-d *dir*	ユーザのホームディレクトリを *dir* で指定
-s *shell*	ユーザのログインシェルを *shell* で指定
-u *uid*	ユーザ ID を *uid* で指定。何を指定すればよいか分からない場合には、これを省略してデフォルトに設定
-g *group*	ユーザのデフォルトグループを *group* で指定。存在するグループであれば、数字表記による ID 番号でもグループ名指定の両方が可能
-G *group1*,*group2*,...	存在するグループ *group1*、*group2* などにメンバーを追加
-m	スケルトンディレクトリ(/etc/skel)にあるファイルを、全て新規作成するホームディレクトリにコピーする。スケルトンディレクトリは伝統的に最低限の(見本の)初期設定ファイル(~/.bash_profileなど)を置くディレクトリである。/etc/skel以外のディレクトリからコピーしたい場合は、-k オプションを追加指定する(例:-k 指定したスケルトンファイルの置かれたディレクトリ)

This PDF is prepared for tools@maruhouse.org, personal use only.

userdel [-r] ユーザ名

shadow-utils

/usr/sbin　stdin　stdout　- file　-- opt　--help　--version

userdel は、ユーザアカウントを削除するコマンドである。

```
# userdel smith
```

アカウントを削除する前には注意が必要である。-r オプションを指定しない場合には、ユーザのホームディレクトリ以下の消去は行わない。削除する前には、再度リカバリさせるような場合に備えて、全てのファイルをバックアップしておくことを推奨する。また、アカウントを削除せずに、usermod -L コマンドでアカウントをロック(lock) することでアカウント削除の代用とすることも可能である。

usermod [オプション] ユーザ名

shadow-utils

/usr/sbin　stdin　stdout　- file　-- opt　--help　--version

usermodは、ユーザアカウントの変更を行うコマンドである。例えば、ホームディレクトリの変更などである。

```
# usermod -d /home/another smith
```

オプション

-d *dir*	ホームディレクトリを *dir* で指定
-l *username*	ユーザのログイン名を *username* で指定。システムで元のログイン名に依存しているものがないか、変更前に注意すること。なおシステムアカウント(root、daemon など)を変更してはならない
-s *shell*	ログインシェルを *shell* で指定
-g *group*	ユーザの初期(デフォルト)グループを *group* で指定。存在するグループならば、数字の ID でもグループ名の指定も可能
-G *group1,group2,...*	指定した既存グループ(*group1*、*group2* など)に追加される。ユーザが、現在あるグループに所属していて、新規に所属するグループとして指定しなかった場合、そのグループからの削除を実行
-L	指定したアカウントがログインできないようにロック
-U	-L で指定したアカウントロックを解除

This PDF is prepared for tools@maruhouse.org, personal use only.

passwd [オプション] ユーザ名

passwd

/usr/bin stdin stdout - file -- opt --help --version

passwdは、ログインパスワードを変更するコマンドである。引き数を指定しない場合には、自分自身のパスワードを変更する。

```
$ passwd
```

スーパーユーザは、他のユーザアカウントのパスワードを変更できる。

```
# passwd smith
```

passwdコマンドにはオプションがあるが、これはパスワードの有効期限に関するものが多い。これらはセキュリティポリシーを厳格に定義してから実施すること。

chfn [オプション] ユーザ名

util-linux

/usr/bin stdin stdout - file -- opt --help --version

chfn(change finger)は、ユーザの個人情報を更新するコマンドである。なお、個人情報とは名前、自宅の電話番号、オフィスの電話番号、オフィスの住所などのfingerコマンドで表示される情報である。chfnコマンドをユーザ名を指定せずに実行すると、自分のアカウント情報の変更を行うことができる。rootユーザがユーザ名を指定してchfnコマンドを実行すると、そのユーザ情報を変更する。chfnコマンドをオプションを指定しないで実行すると、全ての情報を表示して変更を促す。

```
$ chfn
Password: ********
Name [Shawn Smith]: Shawn E. Smith
Office [100 Barton Hall]:
Office Phone [212-555-1212]: 212-555-1234
Home Phone []:
```

オプション

-f *name* フルネームを *name* で指定

-h *phone* 自宅の電話番号を *phone* で指定

-p *phone* 職場の電話番号を *phone* で指定

-o *office* 職場の所在を *office* で指定

This PDF is prepared for tools@maruhouse.org, personal use only.

chsh [オプション] ユーザ名 util-linux

/usr/bin stdin stdout - file -- opt --help --version

chshは、ログインシェルを変更するコマンドである。ユーザ名を指定しないでchshコマンドを実行すると、自分のアカウントのシェルの変更を行う。rootユーザがユーザ名を指定してchshコマンドを実行すると、rootユーザのシェルの変更を行う。chshコマンドをオプションを指定しないで実行すると、現在のシェルを表示して変更を行うことができる。

```
$ chsh
Changing shell for smith.
Password: ********
New shell [/bin/bash]: /bin/tcsh
```

新しく設定されるシェルは、/etc/shells に記述されている必要がある。

オプション

-s *shell* シェルの指定
-l 使用できるシェルの一覧表示

6.3 スーパーユーザへの変更

通常一般ユーザは、自分の所有するファイルしか操作できない。一方、スーパーユーザあるいはrootと呼ばれる特別なユーザは、全てのファイルに対して操作可能である。スーパーユーザに変更する場合には、以下のコマンドを実行する。

```
$ su -l
Password: ********
#
```

これによりスーパーユーザのパスワードが質問される(インストール、あるいは変更時にシステム管理者が設定したパスワード)。ユーザがスーパーユーザに変更したということは、シェルのプロンプトが#になることでわかる。この方法は、スーパーユーザ権限を得る一番簡単な方法である。スーパーユーザでの作業を終えたら、^D(Control キー + d) か exit コマンドでスーパーユーザから一般ユーザに変更しなおすこと。

他にも、同様のコマンドが存在しており、その一例としてsudoコマンドがある。suコマンドにユーザ名を付けて実行すると、そのユーザに変更できる(ただし、パスワードを知っている必要がある)。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ su -l jones
Password: ********
```

オプション

-l　ログインシェルとして起動。通常サーチパスを設定するには、このオプションを使用

-m　新しいシェルに現在の環境変数を引き継がないように指定

-c *command*　(他のユーザとして)*command* で指定したコマンドを実行するだけで終了する。このオプションを多用する場合は、sudoコマンドのマニュアルを参照のこと

-s *shell*　実行するシェルを指定(例：/bin/bash)

6.4 グループの操作

groups　ユーザの所属するグループを表示

groupadd　新しいグループを作成

groupdel　グループを削除

groupmod　グループを変更

ユーザアカウントにはグループ情報が設定されている。(ファイルの変更など)ある操作の権限を特定のグループに与えることにより、グループのメンバー全てが同様の操作が可能となる。friends グループに /tmp/sample に関する read、write、execute 全ての権限を許可する例を以下に示す。

```
$ groups
users smith friends
$ chgrp friends /tmp/sample
$ chmod 770 /tmp/sample
$ ls -l /tmp/sample
-rwxrwx---  1 smith friends  2874 Oct 20 22:35 /tmp/sample
```

ユーザをグループに追加するには、root 権限で /etc/group に追加する。chgrp コマンドでグループの変更ができる。

This PDF is prepared for tools@maruhouse.org, personal use only.

groups [ユーザ名]

coreutils

/usr/bin stdin stdout - file -- opt --help --version

groups は、自分または他のユーザが所属しているグループを表示するコマンドである。

```
$ whoami
smith
$ groups
smith users
$ groups jones root
jones : jones users
root : root bin daemon sys adm disk wheel src
```

groupadd [オプション] [グループ名]

shadow-utils

/usr/sbin stdin stdout - file -- opt --help --version

groupadは、新しいグループを作成するコマンドである。既存のグループとの重複を避けるため -f オプションを使用することを推奨する。

```
# groupadd -f friends
```

オプション

-g *gid* 特定の数字の ID を指定する。デフォルトで、groupadd コマンドが自動的に設定

-f 指定したグループが存在する場合、エラーで終了

groupdel [グループ名]

shadow-utils

/usr/sbin stdin stdout - file -- opt --help --version

groupdel は、存在するグループを削除するコマンドである。

```
# groupdel friends
```

実行前に、削除するグループに属する全ファイルを別のグループに変更しておくと、グループ削除後にも取り扱うことが可能である。これは、作業内容次第では便利なことがある。

```
# find / -group friends -print
```

This PDF is prepared for tools@maruhouse.org, personal use only.

この理由は、groupdelコマンドはグループに所属するファイルは変更しないで、単に指定したグループをシステムから削除するのみだからである。

groupmod [オプション] [グループ名] shadow-utils

/usr/sbin stdin stdout - file -- opt --help --version

groupmodは、グループの名前やIDなどを変更するコマンドである。

```
# groupmod -n newname friends
```

groupmodコマンドは、グループに属するファイルの操作は行わない。単に、システムで設定済みのグループIDやグループ名を変更するだけである。そのグループに所属していたファイルは、nonexistentグループに所属する。

オプション

-g *gid* グループIDを*gid*で指定したIDに変更

-n *name* グループ名を*name*で指定した名前に変更

This PDF is prepared for tools@maruhouse.org, personal use only.

7章 ネットワーク管理

7.1 ホストの表示

uname	ホスト情報を表示
hostname	ホスト名を表示
dnsdomainname	DNS ドメイン名の表示(hostname -d と同様)
domainname	NIS ドメイン名の表示(hostname -y と同様)
nisdomainname	NIS ドメイン名の表示(hostname -y / domainname と同様)
ypdomainname	NIS ドメイン名の表示(hostname -y / domainname / nisdomainname と同様)
ifconfig	ネットワークインターフェイス情報の設定、および表示

Linuxマシン(ホスト)は、全てホスト名やIPアドレスなどが設定されているが、ここではそれらの情報を表示するコマンドを紹介する。

uname [オプション]

coreutils

/bin stdin stdout - file -- opt --help --version

uname は、コンピュータの基本的な情報を表示するコマンドである。

```
$ uname -a
Linux server.example.com 2.6.9-1.667 #1 Tue Nov 2 14:59:52 EST 2004 i686 i686 i386 GNU/Linux
```

表示の説明は、それぞれカーネル名(Linux)、ホスト名(server.example.com)、カーネルバージョン(2.6.9-1.667 #1 Tue Nov 2 14:59:52 EST 2004)、ハードウェア名(i686)、CPU の種類(i686)、ハードウェアプラットフォーム(i386)、OS名(GNU/Linux) を意味する。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-a　全ての情報を表示
-s　カーネル名のみ表示(デフォルト)
-n　ホスト名のみ表示
-r　カーネルバージョンのみ表示
-m　ハードウェア名のみ表示
-p　プロセッサタイプのみ表示
-i　ハードウェアプラットフォームのみ表示
-o　OS 名のみ表示

hostname [オプション] [ホスト名]　net-tools

/bin　stdin　stdout　- file　-- opt　--help　--version

hostnameは、コンピュータ名を表示するコマンドである。設定された情報によることもあるが、FQDN(fully qualified domain name)が表示される。

```
$ hostname
myhost.example.com
```

短い(ドメイン部分を省略した)ホスト名の場合は、以下のような表示になる。

```
$ hostname
myhost
```

root ではホスト名を設定できる。

```
# hostname orange
```

ホスト名とネームサーバについては本書の範囲外である。当然のことではあるが、やみくもにホスト名を変更しないように注意してほしい。

オプション

-i　ホストの IP アドレスの表示
-a　ホストの別名の表示
-s　ホストの短い名前の表示
-f　ホストの FQDN での表示
-d　ホストの DNS ドメイン名の表示

This PDF is prepared for tools@maruhouse.org, personal use only.

-y　　ホストの NIS/YP ドメイン名の表示
-F *hostfile*　　*hostfile* に指定されたホスト名を設定

ifconfig [ネットワークインターフェイス名]　　net-tools

/sbin　　stdin　stdout　- file　-- opt　--help　--version

ifconfigは、ネットワークインターフェイスの設定情報を表示するコマンドである。詳細は、本書の範囲を超えるため取り扱わないが、ここではいくつかテクニックを紹介する。

デフォルトのネットワークインターフェイス(eth0)を表示させる方法を以下に示す。

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:50:BA:48:4F:BA
          inet addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1955231 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1314765 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:2320504831 (2213.0 Mb)  TX bytes:152785756 (145.7 Mb)
          Interrupt:11 Base address:0x6000
```

ここでは、MACアドレスが00:50:BA:48:4F:BA、IPアドレスが192.168.0.10、ネットマスクが255.255.255.0、およびその他の情報が表示されている。全てのネットワークインターフェイスを表示するには以下に示したコマンドを実行する。

```
$ ifconfig -a
```

詳細は、ifconfig のマニュアルを参照のこと。

7.2 ホストの情報

host　　ホストネーム、IP アドレスと DNS 情報を表示
whois　　ドメイン名をドメイン管理レジストラから検索
ping　　リモートホストが到達可能か確認
traceroute　　リモートホストまでの経路を表示

リモートのコンピュータを使用する場合、リモートコンピュータに関する情報を知りたいことがある。例えば、誰の所有であるか、IPアドレスは、所在はどこか、どのネットワークを経由しているか、などである。

This PDF is prepared for tools@maruhouse.org, personal use only.

host [オプション] ホスト名 [サーバ]

bind-utils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

host は、リモートホストのホスト名や IP アドレスを DNS から検索するコマンドである。

```
$ host www.redhat.com
www.redhat.com has address 66.187.232.50
$ host 66.187.232.50
50.232.187.66.in-addr.arpa domain name pointer www.redhat.com.
```

以下のようにさらに詳細な情報を出力することもできる。

```
$ host -a www.redhat.com
Trying "www.redhat.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50419
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.redhat.com.      IN  ANY

;; ANSWER SECTION:
www.redhat.com.    196  IN  A  66.187.232.50

;; AUTHORITY SECTION:
redhat.com.    90535  IN  NS  ns2.redhat.com.
redhat.com.    90535  IN  NS  ns3.redhat.com.
redhat.com.    90535  IN  NS  ns1.redhat.com.

;; ADDITIONAL SECTION:
ns2.redhat.com.    143358  IN  A  66.187.224.210
ns3.redhat.com.    143358  IN  A  66.187.229.10
ns1.redhat.com.    143358  IN  A  66.187.233.210
```

ネームサーバについての議論は、本書の範囲外であるので説明は省略する。hostコマンドは、serverパラメータを指定することで、ネームサーバを指定できる。例えば、comcast.netのサーバを使用する場合は、以下のように実行する。

```
$ host www.redhat.com ns01.jdc01.pa.comcast.net
Using domain server:
Name: ns01.jdc01.pa.comcast.net
Address: 66.45.25.71#53
Aliases:
www.redhat.com has address 66.187.232.50
```

host コマンドを引き数無しで実行すると、全てのオプションを表示する。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-a　全ての情報を表示

-t　ネームサーバへのクエリとして、type を指定。A、AXFR、CNAME、HINFO、KEY、MX、NS、PTR、SIG、SOA などを指定

```
$ host -t MX redhat.com
redhat.com mail is handled by 20 mx2.redhat.com.
redhat.com mail is handled by 10 mx1.redhat.com.
```

hostコマンドでは実行できないリクエストの場合には、digコマンドの使用を推奨する。digは、強力なDNS検索コマンドである。また、nslookupコマンドも使用できるが、このコマンドは今では古いコマンドになりつつある。このため、いくつかのLinuxディストリビューションや UNIX システムでは、nslookup コマンドが付属しているがもはや使用されていないことが多い。

whois [オプション] ドメイン名　jwhois

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

whois は、登録されたドメイン名を検索するコマンドである。

```
$ whois redhat.com
Registrant:
Red Hat, Inc. (REDHAT-DOM)
   P.O. Box 13588
   Research Triangle Park, NC 27709
...
```

最近は、この情報の前後に、インターネットドメイン管理レジストラからの法的注意事項が表示される。

オプション

-h *registrar*　検索するレジストラのサーバを指定
（例：whois -h whois.networksolutions.com yahoo.com）

-p *port*　TCP ポート番号の指定。デフォルトは 43 番（whois サービス）

This PDF is prepared for tools@maruhouse.org, personal use only.

ping [オプション] ホスト名 iputils

/bin stdin stdout - file -- opt --help --version

pingは、リモートホストが到達可能であるかを特定するコマンドである。小さいパケット(ICMP パケット)をリモートホストに送信して、その返答を表示する。

```
$ ping google.com
PING google.com (216.239.37.100) from 192.168.0.10 : 56(84) bytes of data.
64 bytes from www.google.com (216.239.37.100): icmp_seq=0 ttl=49 time=32.390 msec
64 bytes from www.google.com (216.239.37.100): icmp_seq=1 ttl=49 time=24.208 msec
^C
--- google.com ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/mdev = 24.208/28.299/32.390/4.091 ms
```

オプション

-c *N*	実行回数を *N* 回に指定
-i *N*	待ち時間を *N* 秒で設定(デフォルトは 1 秒)
-n	ホスト名でなく IP アドレスで表示

traceroute [オプション] ホスト名 [パケット長] traceroute

/bin stdin stdout - file -- opt --help --version

tracerouteは、ローカルホストからリモートホストへのネットワーク経路を表示するコマンドである。また、その経路での経過時間を表示する。

```
$ traceroute yahoo.com
 1 server.example.com (192.168.0.20) 1.397 ms 1.973 ms 2.817 ms
 2  10.221.16.1 (10.221.16.1)  15.397 ms  15.973 ms 10.817 ms
 3  gbr2-p10.cb1ma.ip.att.net (12.123.40.190)  11.952 ms 11.720 ms  11.705 ms
...
16  p6.www.dcn.yahoo.com (216.109.118.69)  24.757 ms 22.659 ms  *
```

tracerouteは、各ホストに対して3回の経路調査を実施し、それぞれの時間を表示する。5秒以内に返答が無い場合は、*を表示する。また、ファイヤウォールやその他の理由で通過できない場合は、以下に示すシンボルを表示する。

This PDF is prepared for tools@maruhouse.org, personal use only.

シンボル	意味
!F	フラグメンテーションが必要(Fragmentation needed)
!H	ホスト到達不可(Host unreachable)
!N	ネットワーク到達不可(Network unreachable)
!P	プロトコル到達不可(Protocol unreachable)
!S	ソースルート失敗(Source route failed)
!X	管理上、通信が禁止されている(Communication administratively prohibited)
!*n*	ICMP type3 code[n]により到達不能(ICMP unreachable code n)

デフォルトでは40バイトのパケットを使用するが、最後の引き数でパケットサイズを変更することができる(例：`traceroute myhost 120`)。

オプション

-n　　ホスト名の代わりにIPアドレスで表示

-w *N*　　タイムアウト時間を *N* 秒で指定

7.3 ネットワーク接続

ssh　　リモートホストへのセキュアなlogin、およびコマンドの実行

telnet　　リモートホストへのlogin(telnet実行可能な環境は危険！)

scp　　リモートホストとのセキュアなファイルコピー(バッチ処理)

sftp　　リモートホストとのセキュアなファイルコピー(インタラクティブ処理)

ftp　　リモートホストとのファイルコピー(インタラクティブ処理、危険！)

Linuxでは、リモートホストへのネットワーク接続やファイル転送が容易に行える。リモートから接続する際にはセキュリティに気を付けること。

ssh [オプション] ホスト名 [コマンド]　　openssh-clients

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

ssh(secure shell)は、アカウントを持つリモートマシンに安全な方法でログインできるコマンドである。

```
$ ssh remote.example.com
```

This PDF is prepared for tools@maruhouse.org, personal use only.

上記以外にも、ログインせずにリモートマシンでコマンドを実行する使い方もある。

```
$ ssh remote.example.com who
```

sshは、やり取りが行われる全てのデータを暗号化する。このデータには、ユーザ名、(リモートマシンへのログインに必要な)パスワードが含まれる。SSHプロトコルでは他の認証もサポートしており、例えば、公開鍵認証やホストID認証がある。詳細は、man sshdを参照のこと。

オプション

-l *username* — *username* でリモートのリモートを指定。ssh は、ユーザ名を指定しない場合にはローカルのユーザ名が用いられる。*username@host* という書式で指定可能

```
$ ssh smith@server.example.com
```

-p *port* — *port* でポート番号の指定(デフォルトは 22 番)

-t — リモートの tty を割り当てる。リモート環境でテキストエディタのようなインタラクティブなユーザインターフェイスを持つコマンドを実行する場合に有用

-v — 冗長出力を行う。デバッグ時に有用

telnet [オプション] ホスト名 [ポート番号] telnet

/usr/bin stdin **stdout** - file -- opt --help --version

telnet は、アカウントを持つリモートマシンへログインするコマンドである。

```
$ telnet remote.example.com
```

なお、telnetを使用してリモートマシンにログインしてはならない。この理由は、telnetの実装が安全ではないためである。すなわち、パスワードをネットワーク越しに平文で(暗号化されてないまま)送信されるため、第三者に盗聴される危険性がある。telnet の代わりに ssh を使用すること。telnet を使用してもよい場合を以下に示す。

- Kerberos環境でクライアントとサーバの両方で拡張されたtelnetを使っている場合、Fedora Core の telnet は kerberos 環境でも動作する。kerberos に関する詳細は http://web.mit.edu/kerberos/ を参照のこと。
- リモートサーバのあるポートに接続して重要ではないデータを送信する場合。例え

This PDF is prepared for tools@maruhouse.org, personal use only.

ば、ウェブサーバ(80 番ポート)の返事を調査するなどである。

```
$ telnet remote.example.com 80
Trying 192.168.55.21...
Connected to remote.example.com (192.168.55.21).
Escape character is '^]'.
xxx          何か文字を打ち込んでリターンを押す
<HTML><HEAD>  # ウェブサーバが反応を返す
<TITLE>400 Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
Your browser sent a request that
this server could not understand.<P>
</BODY></HTML>
Connection closed by foreign host.
```

一般的にはtelnetの使用を推奨できない。そのため、ここではオプションの説明を行わないこととする。

scp ローカルの path リモートの path

openssh-clients

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

scp(secure copy)は、あるマシンから別のマシンへファイルやディレクトリ以下をコピーするコマンドである(インタラクティブコマンドについてはsftp を参照のこと)。なお、すべての通信情報が暗号化される。

```
$ scp myfile remote.example.com:newfile
$ scp -r mydir remote.example.com:
$ scp remote.example.com:myfile .
$ scp -r remote.example.com:mydir .
```

リモートではローカルと異なるアカウントを使用している場合、「ユーザ名@ホスト名」という書式で指定する。

```
$ scp myfile smith@remote.example.com:
```

オプション

-p　コピー時にファイルの属性(パーミッション、タイムスタンプ)を保持

-r　ディレクトリを階層的にコピー

-v　冗長出力を行う。デバッグ時に有用

This PDF is prepared for tools@maruhouse.org, personal use only.

sftp (ホスト名 | ユーザ名 @ ホスト名)　openssh-clients

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

sftp は、ファイルを 2 台のホスト間でインタラクティブにコピーするコマンドである(なお、scpコマンドはバッチ処理を行う)。ユーザインターフェイスは、ftpとほぼ同様である。

```
$ sftp remote.example.com
Password: ********
sftp> cd MyFiles
sftp> ls
README
file1
file2
file3
sftp> get file2
Fetching /home/smith/MyFiles/file2 to file2
sftp> quit
```

リモートのアカウント名がローカルでのアカウント名と異なる場合、「sftp ユーザ名 @ ホスト名」という書式で実行する。

```
$ sftp smith@remote.example.com
```

コマンド	意味
help	実行可能コマンドの表示
ls lls	ファイルのリストを表示。ls はリモート、(lls)はローカルのディレクトリ
pwd lpwd	ワーキングディレクトリの表示。(pwd)はリモート、(lpwd)はローカルのディレクトリ
cd *dir* lcd *dir*	ディレクトリの変更。(cd)はリモート、(lcd)はローカルのディレクトリ。ディレクトリは *dir* で指定
get *file1* [*file2*]	リモートのファイル *file1* をローカルにコピー。*file2* を指定すると、ローカルのファイル名を変更
put *file1* [*file2*]	ローカルのファイル *file1* をリモートにコピー。*file2* を指定すると、リモートのファイル名を変更
mget *file**	複数のリモートのファイルをローカルにコピー。ワイルドカード(*や?など)も使用可能
mput *file**	複数のローカルのファイルをリモートにコピー。ワイルドカード(*や?など)も使用可能
quit	sftp の終了

This PDF is prepared for tools@maruhouse.org, personal use only.

ftp [オプション] ホスト名 ftp

/usr/bin stdin stdout - file -- opt --help --version

ftp(file transfer protocol) は、ホスト間でファイルをコピーするコマンドである。ftp の通信は暗号化されていないため安全ではない。これは、パスワードをネットワーク越しに平文(暗号化されていない内容)で送信されるため、第三者に盗聴される危険性がある。この理由から ftp の代わりに sftp を使用すること。sftp は、ftp と同様のコマンド体系である。

7.4 電子メール

evolution	GUI ベース電子メールクライアント
mutt	テキストベース電子メールクライアント
mail	テキストベース電子メールクライアント(簡易メールプログラム)

Fedora では、多数の MUA (メールリーダ)を提供している。本書では、これらのうち3つのメーラを紹介する。もちろんこれら以外にも、pine、RMAIL、emacs に付属する vm、mozilla の「Mail & News」などが存在する。

送受信中のメールを確認するには、/var/log/maillog ファイルを参照のこと。また、ユーザが root の場合には、mailq コマンドを利用することでマシンのキューに存在する待機中の送信メールを確認することができる。

evolution evolution

/usr/bin stdin stdout - file -- opt --help --version

Ximian Evolution は、Microsoft Ourlook と似たインタフェースを提供する GUI ベースのメールリーダである。Evolutionをメインメニューから起動できるかはもちろん設定に依存するが、メニューからは Internet : Evolution Email を選択する。シェルからは evolution という名前で実行できる。

メールアカウントの設定方法は以下のとおり。

1. 「ツール」から「設定」を選択。
2. 設定画面で、メールアカウントがない場合は、「追加」を選択。既に存在する場合は、「編集」を選択。

This PDF is prepared for tools@maruhouse.org, personal use only.

3. アカウント設定画面で「身元情報」タブを選択、「名前」と「Eメール・アドレス」を設定。
4. メールの受信タブを選択。サーバ種別を以下から選択(IMAP、POP、「ローカルの配信」など)。mail serverの設定を選択。POPかIMAPの場合は、ISPのメールサーバのホスト名とユーザ名を入力。ローカルの場合はローカルメールボックスのパス(位置)を入力。
5. メールの送信タブを選択。送信メールサーバを選択。リモートのSMTPサーバを選択するとホスト名の入力が促される。ローカルの場合は`sendmail`を選択。
6. 残りのタブとオプションを設定する。OKを選択してアカウントエディタから抜ける。これでメールの基本設定は終了。

受信箱	メールを参照
新規	新規にメールを作成
送受信	新規メールを取得
返信	メッセージにリプライする。送信者のみ
全員へ返信	メッセージにリプライする。`To`と`Cc`に入っている全員に送信
転送	メールを転送

Evolutionには、多数の機能が存在する。詳細は、ツールバーのヘルプボタンをクリックして参照のこと。

mutt [オプション]

mutt

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

`mutt`は、テキストベースのメーラである。ターミナル内で実行される。そのためローカルのXのterminalやssh経由のリモートマシンでも実行できる。muttは多数のオプションを提供し、とても強力なツールである。起動方法は以下のとおり。

```
$ mutt
```

メイン画面が表示されたら、メールボックスにあるメッセージが表示される。1行が1通のメールを示す。コマンドは以下のとおり。

This PDF is prepared for tools@maruhouse.org, personal use only.

表 4 mutt のコマンド

キー入力	意味
上矢印	前のメッセージへ移動
下矢印	次のメッセージへ移動
PageUp	メッセージの 1 ページ上へスクロール
PageDown	メッセージの 1 ページ下へスクロール
Home	1 つ目のメッセージへ移動
End	最後のメッセージへ移動
m	新規メール作成。デフォルトのテキストエディタを起動する。メールの編集後、エディタを抜ける。そこで y と入力すると、メールを送信する。q で送信せずに終了
r	現在のメッセージにリプライ(reply)する。その他は上記 m と同様の動作をする
f	メールの転送を行う。その他は上記 m と同様の動作をする
i	メールボックスの内容を参照
C	現在のメールを別のメールボックスにコピー
d	メールの削除

メッセージの編集後、テキストエディタが終了した時点で使用可能なコマンドを以下に示す。

表 5 mutt のメッセージ編集用コマンド

キー入力	意味
a	ファイルの添付(attach)
c	Cc の設定
b	Bcc の設定
e	再度編集(edit)
r	Reply-To の設定
s	Subject の設定
y	メール送信
C	メールのファイルへのコピー
q	メールを送信せずに破棄

mutt で常時使用可能であるコマンドを以下に示す。

This PDF is prepared for tools@maruhouse.org, personal use only.

表6 mutt のコマンド

キー入力	意味
?	全てのコマンドを参照(スペースバーを押すと下へスクロール、qで終了)
^G	現在のコマンドのキャンセル
q	終了

muttの公式サイトは、http://www.mutt.org/である。muttのチュートリアルは、http://www.cs.utk.edu/~help/mail/mutt_starting.php である。

mail [オプション] 宛先 MailX

/bin stdin stdout - file -- opt --help --version

mailは、簡単なメールクライントである。多くのユーザはもっと強力なMUAを常用したいだろうが、簡単なメッセージをコマンドラインから送信する場合や、スクリプトに使用する時はmailコマンドは便利である。

簡単なメールの送信手順は以下のとおり。

```
$ mail smith@example.com
Subject: my subject
I'm typing a message.
To end it, I type a period by itself on a line.
.
Cc: jones@example.com
$
```

1つのコマンドで実行する方法は以下のとおりである。

```
$ echo "Hello world" | mail -s "subject" smith@example.com
```

ファイルを用いて以下のように送信する方法もある。

```
$ mail -s "my subject" smith@example.com < filename
$ cat filename | mail -s "my subject" smith@example.com
```

パイプ(|)を使ってメール送信することも容易である。

オプション

-s *subject* 件名(subject)の設定

This PDF is prepared for tools@maruhouse.org, personal use only.

-v	冗長モード: メールの配送状況を表示
-c *addresses*	Cc のアドレス指定。複数のアドレスはコンマ(,)で区切って指定
-b *addresses*	Bcc のアドレス指定。複数のアドレスはコンマ(,)で区切って指定

7.5 WWW

mozilla	多機能ウェブブラウザ
lynx	テキストベースウェブブラウザ
wget	ウェブページをローカルの HDD に保存
curl	ウェブページをローカルの HDD に保存

Linux にはWWW(World Wide Web)を見る伝統的なGUIのブラウザ、テキストブラウザ、ファイル取得ツールなどが存在する。

mozilla [オプション] [*URL*] mozilla

/usr/bin stdin stdout - file -- opt --help --version

mozillaは、Linuxで有名なウェブブラウザの一つである。X Window Systemを用いて、他の OS でも動作する。バックグラウンドで起動するコマンドを以下に示す。

```
$ mozilla &
```

Mozillaは、ブラウザの機能(ブラウズ、進む/戻るボタン、ブックマーク、履歴など)に加え、タブブラウズ、ポップアップ広告の制限などの機能を提供する。またメールリーダやネットニュースリーダの機能も提供している。ヘルプメニューのHelp Contentsを選択すると、全ての情報が掲載されている http://www.mozilla.org/ へのリンクが提供される。

Mozilla 以外の Linux で使用可能なウェブブラウザは、Mozilla を軽量化した Firefox (http://www.mozilla.org/products/firefox/)、他に Netscape (http://www.netscape.com/) 、Opera (http://www.opera.com/)、KDEのkonquere (http://www.konquerer.org/)、GNOME の Epiphany (http://www.gnome.org/)、Mozilla から派生した Galeon (http://galeon.sourceforge.net/)などが存在する。

This PDF is prepared for tools@maruhouse.org, personal use only.

lynx [オプション] [*URL*]

lynx

/usr/bin　　stdin　　stdout　　- file　　-- opt　　--help　　--version

lynxは、テキストベースのウェブブラウザである。最近では珍しいが、画像の表示を必要としない場合や遅いネットワークの場合は特に有用である。

```
$ lynx http://www.yahoo.com/
```

すべてのブラウズ操作はキーボードで行い、マウスは使用しない。表やフレームは未対応であるため、多くのページは正しく表示されないことがある。

キー入力	意味
?	ヘルプの表示
k	コマンドとキー割り当ての一覧表示
^G	現在のコマンドをキャンセル
q	lynx の終了
Enter	現在のリンクをクリック、もしくは現在のフィールドを終了
左矢印	前のページへ戻る
右矢印	次のページへ進む、もしくは現在のリンクをクリック
g	(URL を入力して)URL へ移動
p	保存して、印刷、もしくは現在のページをメール
スペースバー	下へスクロール
b	上へスクロール
下矢印	次のリンク、もしくは次のフォームのフィールドへ移動
上矢印	前のリンク、もしくは前のフォームのフィールドへ移動
^A	ページの一番上へ移動
^E	ページの一番下へ移動
m	メイン／ホームページへ移動
/	現在のページの検索
a	現在のページをブックマークへ登録
v	ブックマークのリストを表示
r	ブックマークを削除
=	現在のページとリンクのプロパティを表示
\	HTML ソースを表示（再度\を押すと、普通の表示へ戻る）

This PDF is prepared for tools@maruhouse.org, personal use only.

lynxは100個以上のコマンドラインオプションが提供されているため、manを実行し、詳細を確認してみてほしい。

オプション

-dump	ページを描画して標準出力へ出力して終了(-source と比較する)
-source	HTML ソースを標準出力へ出力して終了(wget や curl と比較する)
-emacskeys	lynx のキーを emacs のキーバインドに設定
-vikeys	lynx のキーバインドを vim(もしくは vi)のキーバインドに設定
-homepage=*URL*	*URL* をホームページに設定
-color	カラーモードを on
-nocolor	カラーモードを off

wget [オプション] *URL* wget

/usr/bin stdin stdout - file -- opt --help --version

wget は、URL を指定して情報をファイルや標準出力にダウンロードするコマンドである。あるウェブページを取得したり、ウェブページを階層ごと全て取得する場合に大変便利である。例えば、Yahoo のページを取得する例を以下に示す。

```
$ wget http://www.yahoo.com
--23:19:51--  http://www.yahoo.com/
          => `index.html'
Resolving www.yahoo.com... done.
Connecting to www.yahoo.com[216.109.118.66]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]

    [ <=>                                 ] 31,434        220.84K/s

23:19:51 (220.84 KB/s) - `index.html' saved [31434]
```

取得したファイルはカレントディレクトリのindex.htmlというファイルに保存される。wgetは、途中でネットワークの不調などでダウンロードが停止した場合などに対応するレジューム機能を提供している。レジューム機能を使用する場合には、wget -cオプションを使用する。

同様のコマンドにはcurlがある。このコマンドは、標準出力に出力するところがwgetとは異なる。wget はデフォルトでは元のファイル名と同一の名前で保存する。

```
$ curl http://www.yahoo.com/ > mypage.html
```

This PDF is prepared for tools@maruhouse.org, personal use only.

wgetは、70以上のオプションがあるので重要なもののみ取り上げる。curlのオプションは、man を参照のこと。

オプション

-i *filename*	URL を filename で指定したファイルから読み込みファイルを取得
-O *filename*	取得したすべてのファイルを 1 つのページに追記
-c	継続(continue)モード:前回のファイル取得が途中で停止した場合、wgetはファイルの一部分を取得して終了してしまうことがある。例えば、サイズが150Kバイトのファイルのうち100Kバイトまで取得した場合、-cオプションを付けると残りの 50K バイトを取得して、元のファイルと結合し、150Kバイトのファイルとして保存する。しかし、wgetは万能ではないため、レジューム前から今回実行するまでにリモートでファイルの更新が行われた場合などには対応できない。-c オプションは、ファイルが更新されていない場合のみ使用可能
-t *N*	試行回数を*N* 回に設定する *N* を 0 とすると永久に実行し続ける
--progress=dot --progress=bar	ダウンロードの進行状況のドット／バー表示
--spider	実際にダウンロードせずに、リモートのページが存在するかの確認のみ
-nd	取得したファイルをカレントディレクトリに保存する。リモートの階層ディレクトリ構造を無視(デフォルトでは wget はリモートの階層構造をそのまま複製)
-r	全てのディレクトリサブディレクトリのディレクトリ構造を階層的に取得
-l N	取得するディレクトリの深さを *N* 段に制限(デフォルトは 5 段である)
-k	取得したファイルを書き換え、リンクをローカルの構造に一致
-p	スタイルシートや画像など、ページの表示のために必要なファイル全ての取得
-L	ページ内の相対リンクのみ参照し、絶対リンクは非参照
-A pattern1,pattern2,pattern3,...	acceptモード:取得するファイル名のパターンを指定する。シェルと同様のワイルドカードが使用可能
-R pattern1,pattern2,pattern3,...	reject モード:取得しないファイル名のパターンを指定

This PDF is prepared for tools@maruhouse.org, personal use only.

-I pattern1,pattern2,pattern3,...	ディレクトリinclusion：取得するディレクトリ名の指定
-X	ディレクトリexclusion: 取得しないディレクトリ名の指定

7.6 ネットニュース

ネットニュースは古いネットワークコミュニケーションの一つである。その世界にはたくさんのニュースグループが存在し、議論を投稿し、その返事などのやり取りが行われている。Fedoraのニュースリーダとしては、slrnがある（rnやtrnやtinなどもある）。Mozillaでもネットニュースを閲覧できる。ネットニュースは、Google Groups（http://groups.google.com/）で検索可能である。

ネットニュースにアクセスするには、newsサーバに接続する必要がある。newsサーバはニュース記事の投稿／配送を行うサーバのことである。一度、newsサーバに接続すると（例：news.example.com）、購読するニュースグループが記録される。読んだ記事は、自動的にホームディレクトリの指定されたファイルに保存される。ニュースリーダの設定に依存するが、多くの場合 ~/.slrnrc に保存される。

slrn [オプション]　　slrn

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

slrnはニュースリーダである。実行前に、シェル変数NNTPSERVERにニュースサーバを設定する必要がある。

```
$ export NNTPSERVER=news.example.com
```

設定後、ニュースグループファイル作成を行う（初めてslrn起動した場合に限り実行する）。

```
$ slrn --create
```

ニュースの記事を閲覧する場合は、以下を実行する。

```
$ slrn
```

slrnを実行すると、購読中のニュースグループが表示される。オプションを以下に示す。

This PDF is prepared for tools@maruhouse.org, personal use only.

表7 slrnのコマンド

キー入力	意味
q	slrnの終了
Down	次のニュースグループ
Up	前のニュースグループ
Enter	選択したニュースグループを閲覧
p	選択したニュースグループに新たな記事を投稿
a	ニュースグループの追加（追加するニュースグループの名前を知っている必要がある）
u	選択したニュースグループの購読停止（slrnの終了後に設定が反映される）。sで再度購読

ニュースグループを選択してEnterを押すと、slrnは一連の議論（スレッドと呼ぶ）を表示する。ここでのオプションを以下に示す。

表8 slrnでの記事表示時のコマンド

キー入力	意味
q	スレッドの表示を終了。ニュースグループの表示ページに戻る
Down	次のスレッドに移動
Up	前のスレッドに移動
Enter	選択したスレッドを閲覧
c	スレッドに既読マークを付ける。ESCAPE uでアンドゥ（元に戻す）される

記事の閲覧時に使用可能なコマンドは以下のとおりである。

表9 slrnで記事閲覧時に使用可能なコマンド

キー入力	意味
q	記事の表示を終了。グループの表示ページに戻る
Spacebar	閲覧中の記事の次ページに移動
b	閲覧中の記事の前のページに移動
r	記事の投稿者へメールでリプライ
f	記事のフォローアップを投稿
P	新規記事を投稿

This PDF is prepared for tools@maruhouse.org, personal use only.

表 9　slrn で記事閲覧時に使用可能なコマンド（続き）

キー入力	意味
o	記事をファイルに保存
n	次の未読記事に移動
p	前の未読記事に移動

いつでも?を押すとヘルプが表示される。slrnは、多数のオプションが存在する。設定は、~/.slrnrcで行う。より詳細な説明は、/usr/share/doc/slrn*、あるいはhttp://www.slrn.org/を参照されたい。

7.7　メッセンジャー

gaim	インスタントメッセンジャー兼 IRC クライアント
talk	Linux/Unix でのチャットプログラム
write	ターミナルメッセージ記述プログラム
mesg	talk や write の許可・禁止
tty	ターミナルのデバイス名を表示

Linux では他のユーザにメッセージを送信する方法が多数存在する。これは同一マシンでも、インターネットに接続されている他のマシンでも同様である。ターミナルデバイス（terminal devices：ttys）で動作する、talkやwriteのような旧来のプログラムから、gaimのような最近のインスタントメッセージクライアントなどがある。

gaim [オプション]　　gaim

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

gaimは、インスタントメッセージクライアントである。AOL、MSN、Yahooなど多数のプロトコルをサポートしている。また、IRC（internet relay chat）のクライアントでもある。gaimは、X 上で以下のように起動する。

```
$ gaim &
```

IMサービスのアカウントを持っていない場合には、最初にアカウントを作成する必要がある。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-u *screenname*　*screenname* でデフォルトのアカウントを設定
-l　gaim の起動時に自動ログインの設定(パスワードが記憶されている場合)
-w [*message*]　離席 / 退席の設定。*message* でメッセージを設定

talk [ユーザ名 [@ ホスト名]] [*tty*]　talk

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

talkは、インスタントメッセンジャーが登場する以前のプログラムである。このコマンドは同一マシン、もしくは他のマシンにログインしている 2 ユーザ間で、1 対 1 の対話を実現する。talkコマンドは、LinuxやUNIXマシン(その他、移植されたプラットフォームを含む)で動作する。xtermなどのウィンドウで上下に画面が分割され、お互い入力した文字が表示される。

```
$ talk friend@example.com
```

talk の相手が複数の tty を使用している場合には、使用する tty を 1 つ指定する。

write ユーザ名 [*tty*]　util-linux

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

writeコマンドは、talkよりもさらに原始的なコマンドで、同一マシンにログインしているユーザから他のユーザに対してテキストを 1 行送信する。

```
$ write smith
Hi, how are you?
See you later.
^D
```

^Dで接続を終了する。writeは以下のようにパイプを使用して、メッセージを送信することもできる。

```
$ echo 'Howdy!' | write smith
```

This PDF is prepared for tools@maruhouse.org, personal use only.

mesg [yln]

SysVinit

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

mesg は、talk や write コマンドの制限を行うコマンドである。mesg y では、メッセージ書き込みを許可し、mesg n では拒否する。mesg コマンドを単独で実行すると、現在の状況(y か n)を表示する。なおmesgは、gaimなどのインスタントメッセンジャーの動作と連携していないため、独立で実行することができる。

```
$ mesg
is n
$ mesg y
```

tty

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

tty は、現在のターミナルデバイス名を表示するコマンドである。

```
$ tty
/dev/pts/4
```

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

8章 シェルスクリプトによるプログラミング

シェル(bash)を紹介した4章では、シェルにはプログラミング言語が組み込まれていると説明した。実際に、1つのコマンドでは実現できない操作を、プログラム(シェルスクリプト)として記述できるようになる。他のプログラミング言語と同様に、シェルスクリプトでは変数、条件分岐(if-then-else など)、ループ、入出力などが使用できる。しかし、1冊まるごとシェルスクリプトを説明した本が存在するほど説明する分量が多いため、本書では基本的な部分のみ紹介する。info bash を参照のこと。

8.1 空白文字と改行

bash のシェルスクリプトでは、空白文字と改行文字を注意深く扱う必要がある。これは、プログラムによってはシェルにコマンドとして解釈されるキーワードが存在し、コマンドと引き数の間は空白文字で区切る必要があるためである。同様に、コマンド途中での改行は、シェルの解釈ミスを引き起こし、不完全なコマンドとなる。以下、詳細を説明する。

8.2 変数

変数は使用前に定義する。

```
$ MYVAR=6
$ echo $MYVAR
6
```

変数は全て文字列として扱われるが、数字の場合シェルは数値として扱う。

```
$ NUMBER="10"
$ expr $NUMBER + 5
15
```

This PDF is prepared for tools@maruhouse.org, personal use only.

シェルスクリプトを使用する場合、変数を""(ダブルクォート)で囲み、実行時のエラーを避けることを推奨する。変数に未定義の値や空白文字が存在すると、予期しない結果を引き起こす可能性がある。

```
$ FILENAME="My Document"                  ファイル名の中にスペースが存在
$ ls $FILENAME                            lsを実行
ls: My: No such file or directory         なんと！lsは引き数が2つあると解釈してしまった
ls: Document: No such file or directory
$ ls -l "$FILENAME"                       ダブルクォートで囲んでみる
My Document                               引き数が1つとして扱われ、正しく表示された
```

変数名が別の名前として解釈される事を防ぐには、中かっこ{}を使用する。

```
$ HAT="fedora"
$ echo "The plural of $HAT is $HATs"
The plural of fedora is                   なんと！"HATs"という変数は無いと解釈されてしまった
$ echo  "The plural of $HAT is ${HAT}s"   中カッコで囲んでみる
The plural of fedora is fedoras           期待した通りの結果
```

8.3　入出力

スクリプトの出力は、echoやprintfコマンドなどが使用可能である(「9.1　画面出力」を参照のこと)。

```
$ echo "Hello world"
Hello world
$ printf "I am %d years old\n" `expr 20 + 20`
I am 40 years old
```

入力には、readコマンドを使用できる。このコマンドは標準入力から1行ずつ読み込み、変数に格納する。

```
$ read name
Sandy Smith <ENTER>
$ echo "I read the name $name"
I read the name Sandy Smith
```

8.4　真偽値と返り値

条件分岐とループの説明をする前に、真偽(真(true)／偽(false))を学ぶ必要がある。シェルでは、“0”を真あるいは成功、その他全てを偽あるいは失敗として処理される。

This PDF is prepared for tools@maruhouse.org, personal use only.

すべてのLinuxコマンドは、コマンドの終了時に整数のリターンコード、もしくは終了ステータス(exit status)(返り値)を持つ。この値は、シェルの特殊変数"$?"で参照することができる。

```
$ cat myfile
My name is Sandy Smith and
I really like Fedora Linux
$ grep Smith myfile
My name is Sandy Smith and          一致部分を見つけると
$ echo $?
0                                   成功の返り値0を返す
$ grep aardvark myfile
$ echo $?                           一致部分がみつからないと
1                                   失敗の返り値1を返す
```

コマンドのリターンコードはmanページに掲載されているので確認してみてほしい。

8.5 testコマンド

test(シェルビルトイン)は、数値や文字列を評価して真偽値を返すコマンドである。結果は、0が真(true)で1が偽(false)である。

```
$ test 10 -lt 5       10 は 5 より小さいか? (10 < 5は正しいか?)
$ echo $?
1                     正しくない
$ test -n "hello"     文字列 "hello" は、文字列長が 0 文字ではないか?
$ echo $?
0                     正しい
```

testの引き数は、表12を参照のこと。数値、文字列、ファイルの比較が行われる。

testは、別名として[(左大かっこ)というコマンドがあり、条件分岐やループで使用される。これを使用する場合には、評価式の最後に](右大かっこ)がセットで必要である。以下の評価式は、前述のものと同一である。

```
$ [ 10 -lt 5 ]
$ echo $?
1
$ [ -n "hello" ]
$ echo $?
0
```

[は、コマンドであることに注意すること。そのため、引き数との間に空白文字が必要である。空白文字を入れない失敗例を以下に示す。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
$ [ 5 -lt 4]          4 と ] の間に空白文字が無い
bash: [: missing ']'
```

testコマンドは最後の引き数を4]だと解釈する。このため、“]”が見つからないと判断して、メッセージを表示する。

表12 testコマンドの引き数

ファイルテスト	
-d *name*	ファイル *name* がディレクトリ
-f *name*	ファイル *name* が通常のファイル
-L *name*	ファイル *name* がシンボリックリンク
-r *name*	ファイル *name* が存在して読み込み可能
-w *name*	ファイル *name* が存在して書き込み可能
-x *name*	ファイル *name* が存在して実行可能
-s *name*	ファイル *name* が存在してサイズが0でない
f1 -nt *f2*	ファイル *f1* がファイル *f2* より新しい
f1 -ot *f2*	ファイル *f1* がファイル *f2* より古い
文字列テスト	
s1 = *s2*	文字列 *s1* が文字列 *s2* と等しい
s1 != *s2*	文字列 *s1* が文字列 *s2* と等しくない
-z *s1*	文字列 *s1* のサイズが0
-n *s1*	文字列 *s1* のサイズが0でない
数値テスト	
a -eq *b*	数値 *a* と *b* が等しい
a -ne *b*	数値 *a* が数値 *b* と等しくない
a -gt *b*	数値 *a* が数値 *b* より大きい
a -ge *b*	数値 *a* が数値 *b* 以上
a -lt *b*	数値 *a* が数値 *b* より小さい
a -le *b*	数値 *a* が数値 *b* 以下
組合せと否定	
t1 -a t1	AND条件：t1とt2の両方が真
t1 -o t2	OR条件：t1かt2のどちらかが真
! your_test	否定（つまりyour_testが偽の時に真となる）
\(your_test \)	グループとしてまとめる

This PDF is prepared for tools@maruhouse.org, personal use only.

8.6　真偽

bash にはビルトインコマンドとして、true と false がある。これは、単に終了コードをそれぞれ 0 と 1 に設定する。

```
$ true
$ echo $?
0
$ false
$ echo $?
1
```

これが便利なのは条件分岐とループの場合である。

8.7　条件分岐

if ステートメントは、複雑な条件分岐を実現する。単純な if-then ステートメントの例を以下に示す。

```
if command              コマンドの返り値が 0 なら、then 以下を実行する
then
  body
fi
```

例：

```
if [ `whoami` = "root" ]
then
  echo "You are the superuser"
fi
```

次は、if-then-else ステートメントを紹介する。

```
if command
then
  body1
else
  body2
fi
```

例：

```
if [ `whoami` = "root" ]
```

This PDF is prepared for tools@maruhouse.org, personal use only.

```
then
  echo "You are the superuser"
else
  echo "You are an ordinary dude"
fi
```

最後に、多くの分岐ができるif-then-elif-elseステートメントを紹介する。

```
if command1
then
  body1
elif command2
then
  body2
elif ...
  ...
else
  bodyN
fi
```

例:

```
if [ `whoami` = "root" ]
then
  echo "You are the superuser"
elif [ "$USER" = "root" ]
then
  echo "You might be the superuser"
elif [ "$bribe" -gt 10000 ]
then
  echo "You can pay to be the superuser"
else
  echo "You are still an ordinary dude"
fi
```

caseステートメントは、数値、および条件式、いずれも解釈する。

```
echo 'What would you like to do?'
read answer
case "$answer" in
  eat)
    echo "OK, have a hamburger"
    ;;
  sleep)
    echo "Good night then"
    ;;
  *)
    echo "I'm not sure what you want to do"
```

This PDF is prepared for tools@maruhouse.org, personal use only.

```
    echo "I guess I'll see you tomorrow"
    ;;
esac
```

一般的な型式を以下に示す。

```
case string in
  expr1)
    body1
    ;;
  expr2)
    body2
    ;;
  ...
  exprN)
    bodyN
    ;;
  *)
    bodyelse
    ;;
esac
```

この場合、*string* は、どのようなものでもかまわない。通常、$myvar のような変数であるが、*expr1* から *exprN* のパターン（詳細は、info bash reserved case を参照のこと）ということもある。最後の*（アスタリスク）は、“else”のような役割を担う。全てのコマンドの終わりには、“;;”が必要である。

```
case $letter in
  X)
    echo "$letter is an X"
    ;;
  [aeiou])
    echo "$letter is a vowel"
    ;;
  [0-9])
    echo "$letter is a digit, silly"
    ;;
  *)
    echo "I cannot handle that"
    ;;
esac
```

This PDF is prepared for tools@maruhouse.org, personal use only.

8.8 ループ

while ループは、条件式が真である限り一連のコマンドを繰り返す。

```
while command          command の返り値が 0 ならば以下を実行する。
do
  body
done
```

スクリプト myscript の例を以下に示す。

```
i=0
while [ $i -lt 3 ]
do
  echo "again"
  i=`expr $i + 1`
done

$ ./myscript
0
1
2
```

until は、条件式が 0（真）になるまで繰り返す。

```
until command          command の返り値が 0 以外ならば以下を実行する。
do
  body
done
```

例：

```
i=0
until [ $i -gt 3 ]
do
  echo "again"
  i=`expr $i + 1`
done

$ ./myscript
0
1
2
```

for ループはリストの要素まで繰り返す。

This PDF is prepared for tools@maruhouse.org, personal use only.

```
for variable in list
do
  body
done
```

例：

```
for name in Tom Jack Harry
do
  echo "$name is my friend"
done

$ ./myscript
Tom is my friend
Jack is my friend
Harry is my friend
```

forループは、ファイルリスト処理などに向いている。例えば、全てのカレントディレクトリのファイルを対象とする場合などである。

```
for file in *.doc
do
  echo "$file is a stinky Microsoft Word file"
done
```

無限ループは、while ループで条件に true を使用する。もしくは、until ループで条件に false を使用する。

```
while true
do
  echo "forever"
done

until false
do
  echo "forever again"
done
```

おそらく何らかの条件でループを停止したいことがあるかもしれないが、その場合には break か exit を使用する。

This PDF is prepared for tools@maruhouse.org, personal use only.

8.9　中止と継続

breakは、ジャンプして現在のループから抜け出すコマンドである。以下に示す簡単なスクリプト myscript で確認してみよう。

```
for name in Tom Jack Harry
do
  echo $name
  echo "again"
done
echo "all done"

$ ./myscript
Tom
again
Jack
again
Harry
again
all done
```

break を使用した場合、以下のようになる。

```
for name in Tom Jack Harry
do
  echo $name
  if [ "$name" = "Jack" ]
  then
    break
  fi
  echo "again"
done
echo "all done"

$ ./myscript
Tom
again
Jack            break が起こった
all done
```

continue は、それ以降のコマンドを実行しない。

```
for name in Tom Jack Harry
do
  echo $name
  if [ "$name" = "Jack" ]
  then
```

This PDF is prepared for tools@maruhouse.org, personal use only.

```
    continue
  fi
  echo "again"
done
echo "all done"

$ ./myscript
Tom
again
Jack          Continueしている
Harry
again
all done
```

breakとcontinueは、数字の引き数も使用可能である（例：break *N*、continue *N*）。これは多段ループを制御をする時に使用する（上記例は*N*段ループからの脱出例である）。しかし、この書式を用いると、スクリプトが絡まってほどけない「スパゲッティコード」になるので使用しないことを推奨する。

8.10　シェルスクリプトの作成と実行

シェルスクリプトを作成後、実行にはbashコマンドをファイルの先頭に記述すればよい。スクリプトの実行は、この他にも以下の3通りの方法がある。

#!/bin/bashと先頭に記述し、実行パーミッションを設定する

これはスクリプトを実行する際によく行われる方法である。以下の行をファイルの先頭に追加する。

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

スクリプトファイルの先頭行に左詰めで記述する。次に、以下のコマンドを用いて実行パーミッションを設定する。

```
$ chmod +x myscript
```

さらに、コマンドサーチパスが通っているディレクトリに置けば、コマンドを以下のように、コマンド名のみで実行できる。

```
$ myscript
```

This PDF is prepared for tools@maruhouse.org, personal use only.

スクリプトがカレントディレクトリに存在し、“.”(カレントディレクトリ)がサーチパスに入っていない場合には、以下のようにファイルを指定する際に、ファイル名の前に、“./”を付ける必要がある。

```
$ ./myscript
```

カレントディレクトリは、セキュリティの問題があるため、通常サーチパスには入れない。

bashへの引渡し

bashは、引き数を実行するスクリプト名として解釈する。例を以下に示す。

```
$ bash myscript
```

現在のシェルに“.”コマンドで実行する。

上記で説明した実行方法は、現在実行中のシェルには何の影響も与えることはない。スクリプト内で、現在使用中のシェルで変更を行いたい場合(変数の設定、ディレクトリの変更など)、現在のシェルで、“.”コマンドを用いて以下のように実行する。

```
$ . myscript
```

8.11 コマンドライン引き数

シェルスクリプトは、コマンドライン引き数を使用可能である。また、他のLinuxのコマンドと同様に、オプションの指定も可能である(実際、Linuxで使用されているコマンドにもスクリプトは多数存在する)。自作のシェルスクリプトでも引き数を($1、$2、$3などで)用いることができる。

```
$ cat myscript
#!/bin/bash
echo "My name is $1 and I come from $2"

$ ./myscript Johnson Wisconsin
My name is Johnson and I come from Wisconsin
$ ./myscript Bob
My name is Bob and I come from
```

引数の個数は特殊変数$#で知ることができる。

```
if [ $# -lt 2 ]
then
```

This PDF is prepared for tools@maruhouse.org, personal use only.

```
  echo "$0 error: you must supply two arguments"
else
  echo "My name is $1 and I come from $2"
fi
```

シェルの特殊変数 $0 は、スクリプト自身の名前が設定されている。これは“Usage”の表示やエラーメッセージでよく使われる。

```
$ ./myscript Bob
./myscript error: you must supply two arguments
```

複数の引き数は、for ループで扱うことができる。引き数は、特殊変数“$@”に設定されている。

```
for arg in $@
do
  echo "I found the argument $arg"
done
```

8.12　終了と返り値

exitコマンドは、スクリプトを終了させて、返り値をシェルに送る。一般的には、シェルスクリプトは成功時に 0 を返し、失敗時に 1（もしくは 0 以外の数値）を返す。スクリプト内で exit を使用しない場合、自動的に 0 が返り値として設定される。

```
if [ $# -lt 2 ]
then
  echo "Error: you must supply two arguments"
  exit 1
else
  echo "My name is $1 and I come from $2"
fi
exit 0

$ ./myscript Bob
./myscript error: you must supply two arguments
$ echo $?
1
```

8.13　シェルスクリプトを超えて

シェルスクリプトは素晴らしいプログラミング言語であるが、Linux にはさらに強力なスクリプト言語が存在する。また、それ以外にもコンパイルプログラミング言語も存在する。代

This PDF is prepared for tools@maruhouse.org, personal use only.

表的な言語をいくつか取り上げておこう。

言語名	プログラム	情報源
Perl	perl	man perl http://www.perl.com/
Python	python	man python http://www.python.org/
C、C++	gcc	man gcc http://www.gnu.org/software/gcc/
Java	javac	http://java.sun.com/
FORTRAN	g77	man g77 http://www.gnu.org/software/fortran/fortran.html
Ada	gnat	info gnat http://www.gnu.org/software/gnat/gnat.html

This PDF is prepared for tools@maruhouse.org, personal use only.

9章 その他のツール

9.1 画面出力

echo	テキストを標準出力に表示
printf	書式を指定して標準出力に表示
yes	テキストを繰り返し標準出力に表示
seq	連続した数字を標準出力に表示
clear	スクリーン / ウィンドウのクリア

Linux では、自分自身で標準出力に文字を表示するコマンドがいくつか存在している。

```
$ echo hello world
hello world
```

それぞれのコマンドは、コマンド特有の特徴を有しており、これらのコマンドはLinuxを学ぶ上では必要不可欠である。例えば、プログラムのデバッグやシェルスクリプトなどで頻繁に使用される。

echo [オプション] 文字列 bash

shell built-in stdin stdout - file -- opt --help --version

echo は、単に引き数を表示するのみのコマンドである。

```
$ echo We are having fun
We are having fun
```

/sbin/echo とシェルのビルトインコマンドの echo は、若干ではあるが振る舞いが異なる。type echo を実行し、どちらの echo を実行しているかを確認してみてほしい。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-n　最後の改行を非表示

-e　エスケープキャラクタを有効にする。例えば、echo 'hello\a' とecho -e 'hello\a' では、前者は文字列として表示されるが、後者はビープ音で警告する

-E　エスケープキャラクタを無効にする。-e と逆

エスケープキャラクタを以下に示す。

\a　ビープ音
\b　バックスペース
\c　最後の改行文字を非表示(-n と同様)
\f　改ページ(コンソール上の場合、画面消去)
\n　ラインフィード (改行)
\r　キャリッジリターン
\t　水平タブ
\v　垂直タブ
\\　バックスラッシュ
\'　シングルクォート
\"　ダブルクォート
nnn　8 進数表記された *nnn* を ASCII で表示

printf フォーマット文字列 引き数 [引き数]　bash

shell built-in　stdin　stdout　- file　-- opt　--help　--version

printfコマンドは、echoコマンドの拡張であり文字列の書式を指定できる。文字列と引き数を C 言語の printf()関数とよく似た形式で指定できる。

```
$ printf "User %s is %d years old.\n" sandy 29
User sandy is 29 years old.
```

%s と %d は、書式の指定である。引き数は、sandy と 29 である。printf は、書式文字列にこの引き数を使用して表示する。書式指定では、浮動小数点数も使用できる。

```
$ printf "That\'ll be $%0.2f, sir.\n" 3
That'll be $3.00, sir.
```

This PDF is prepared for tools@maruhouse.org, personal use only.

printfは、bashのビルトインコマンドであるprintfと、/usr/bin/printfの2つのコマンドが存在する。2つのコマンドの違いは、%qである。これはbashのビルトインコマンドprintfのみで使用可能である。%qにより、エスケープ文字(\)を安全に表示することができる。以下に示した例を参照のこと。

```
$ printf "This is a quote: %s\n" "\""
This is a quote: "
$ printf "This is a quote: %q\n" "\""
This is a quote: \"
```

書式指定子(この例では%q)と引き数(この例では\)の個数が同一であることを確認してほしい。引き数の個数が多い場合、余分なものは無視され、不足している場合には、printfはデフォルト値(数字ならば0、文字列ならば""(空文字列))を表示する。printfコマンドはこのように処理を行うが、書式指定子と引き数と個数を合わせる必要がある。この理由は、シェルスクリプト内部でこの2つが合致していない場合にバグとなるためである。

書式指定子は、C言語のprintfのマニュアルで詳細に説明されている(man 3 printfを参照のこと)。以下によく扱われるものを示す。

%d	(単精度)10進整数
%ld	倍精度10進整数
%o	8進整数
%x	16進整数
%f	(単精度)浮動小数点数
%lf	倍精度浮動小数点数
%c	1文字
%s	文字列
%q	シェルのメタキャラクタ入り文字
%%	%(一文字)

%の後ろの数字は桁数の指定である。例えば、“%5d”の場合10進整数を5桁で表示、また、“%6.2f”の場合、浮動小数点数で6桁と小数点以下2桁で表示することを意味する。

n	*n*桁
On	*n*桁で0パディングあり
n.m	*n*桁で小数点以下*m*桁

This PDF is prepared for tools@maruhouse.org, personal use only.

printfは、“\n”や“\a”の様なエスケープ文字も解釈する(“\n”は改行文字、“\a”はベルを鳴らす)。エスケープ文字のリストは、echoコマンド部分を参照のこと。

yes [文字列]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

yesは、指定した文字列を出力するコマンドである(指定しない場合、デフォルトの“y”を出力)。1行に1出力を繰り返し続ける。

```
$ yes again
again
again
again
...
```

yesは、はじめのうちは何の意味も持たないコマンドのように思えるかもしれないが、バッチコマンド内のインタラクティブな処理を実現する際には最良のコマンドである。“Are you SURE you want to do that”というメッセージへの返答を自動的に処理した場合は、以下のようにyesコマンドの出力結果を入力する場合には、コマンドにパイプを使って入力することができる。

```
$ yes | my_interactive_command
```

my_interactive_commandが終了すると、yesが実行される。

seq [オプション] 定義

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

seqは、整数／実数の連番を表示するコマンドである。このコマンドを他のコマンドにパイプで入力することができる。なお、引き数は3種類ある。

上限の数を1つ

seqは、1から引き数まで連続で表示する。

```
$ seq 3
1
2
3
```

This PDF is prepared for tools@maruhouse.org, personal use only.

数2つ：上限と下限の指定

seq は、1つ目の引き数から2つ目の引き数までの数を表示する。

```
$ seq 5 2
5
4
3
2
```

3つの引き数：下限と差分と上限

seq 1つ目の引き数と、それに2つ目の引き数を足した数を表示する。そしてこれを3つ目の引き数まで続ける。

```
$ seq 1 .3 2
1
1.3
1.6
1.9
```

オプション

-w　先頭に 0 を付けて全行の桁を揃えて表示。

```
$ seq -w 8 10
08
09
10
```

-f *format_string*　printf に似た書式指定で出力。%g（デフォルト）か、%e か、%f を含む必要がある。

```
$ seq -f '**%g**' 3
**1**
**2**
**3**
```

-s *string*　指定した文字列を、数字の区切り文字として表示デフォルトでは改行文字が使われるため、1行に数字が表示される。

```
$ seq -s ':' 10
1:2:3:4:5:6:7:8:9:10
```

This PDF is prepared for tools@maruhouse.org, personal use only.

clear

ncurses

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

clear は、シェルの画面をクリアするコマンドである。

9.2 計算

xcalc	GUI ベースの電卓ツール
expr	CUI ベースの数値評価ツール
dc	テキストベースの電卓ツール

Linux は、グラフィカルな電卓アプリケーションだけでなく、コマンドラインの数式処理プログラムも提供している。

xcalc [オプション]

XFree86-tools

/usr/X11R6/bin　stdin　stdout　- file　-- opt　--help　--version

xcalc は、X で動作するシンプルな画面の GUI ベースの電卓ツールである。逆ポーランド記法を使用する場合には、-rpn オプションを指定する。

expr [評価式]

coreutils

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

expr は、入力した式を評価するコマンドである。

```
$ expr 7 + 3
10
$ expr '(' 7 + 3 ')' '*' 14   シェルのエスケープ文字はクォートする
140
$ expr length ABCDEFG
7
$ expr 15 '>' 16
0
```

引き数はそれぞれ、空白文字で区切られる。シェルの特殊文字にはクォートやエスケープが必要である。(エスケープされた)かっこは、グルーピングの指定として使用する。表10は、

This PDF is prepared for tools@maruhouse.org, personal use only.

expr の演算子である。

表 10　expr の演算子

演算子	数値表現での意味	文字表現での意味
+	加算	
-	減算	
*	乗算	
/	除算	
%	モジュロ（剰余）	
<	より小さい	辞書順で高速
<=	以下	辞書順で高速、もしくは同一
>	より大きい	辞書順で低速
>=	以上	辞書順で低速、もしくは同一
=	等しい	同一
!=	等しくない	同一でない
|	or 条件	or 条件
&	and 条件	and 条件
s : *regexp*		正規表現を *regexp* で指定 *s* と比較
substr *s p n*		*p* 文字目から始まる *n* 文字の文字列 *s*（*p*=1 は最初の文字）
index *s chars*		文字 *s* が文字列 *chars* に含まれているか、見つからない場合 0 を返す。C の関数 index() と同一

ブーリアンの評価は、数字が0、あるいは空文字列のとき偽、それ以外を真として扱う。評価の結果は、0 が偽で、1 が真である。expr は、万能ではないため複雑な用途には Perl などを使用することを推奨する。

dc [オプション] [ファイル]　bc

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

dc（desk calculator）は、逆ポーランド記法をサポートしたスタック式電卓ソフトである。このソフトは、標準入力から式を読み込み、結果を標準出力に書き出す機能を提供する。HP（Hewlett-Packard）の逆ポーランド電卓の使用法を知っていれば、dcはとても簡単に使えるはずである。逆ポーランド記法の電卓を使用したことがない場合には、慣れるまでdcは扱いにくいかもしれない。以下、基本コマンドを紹介する。

This PDF is prepared for tools@maruhouse.org, personal use only.

q　　dc を終了
f　　スタックの表示
c　　スタックの消去
p　　スタックの一番上の値を表示
P　　スタックの一番上の値を削除(pop)
*n*k　　計算の基数を *n* に変更(デフォルトは 0 で、整数の計算)

スタックの上 2 つの数値を pop して、指定された演算を行い、結果を push する。

+　　加算
-　　減算
*　　乗算
/　　除算
%　　モジュロ(剰余)
^　　指数(上から 2 つめの数値が基数となり、一番上の数値が指数の値になる)

スタックの上位 2 つの数値を pop して、指定された演算を行い、結果を push する。

v　　ルート(2 乗根)

例を示す。

```
$ dc
4 5 + p          4 と 5 の和を表示
9
2 3 ^ p          2 の 3 乗を表示
8
10 * p           スタックの最上位を 10 倍し、結果を表示
80
f                スタックを表示
80
9
+p               スタックの上 2 つの値を pop して、和を表示

89
```

This PDF is prepared for tools@maruhouse.org, personal use only.

9.3 日付と時間

xclock	GUI ベースの時計
cal	カレンダー表示
date	日付と時刻の表示／設定
ntpdate	システム時間と NTP サーバを同期

時間が知りたい場合など、以下のコマンドで時刻や日付の設定や表示が行える。

xclock [オプション]

XFree86-tools

/usr/X11R6/bin　stdin　stdout　- file　-- opt　--help　--version

xclock は、X 上で時計を表示するコマンドである。他の時計を利用したい場合には、例えば oclock（円時計）、t3d（3D のボールが表示される時計）も利用可能である。なお、サーチパスに存在しないかもしれない /usr/X11R6/lib/xscreensaver/t3d など、また GNOME、あるいは KDE 上のタスクバーに時刻を表示する時計などが存在する。

オプション

-analog	アナログ表示
-digital [-brief]	デジタル表示（日付と時刻を表示）。-brief オプションを付けると、時刻のみ表示
-update *N*	*N* 秒おきに更新

cal [オプション] [月 [年]]

util-linux

/usr/bin　stdin　**stdout**　- file　-- opt　--help　--version

cal は、カレンダーを表示するコマンドである。デフォルトでは今月のカレンダーを表示する。

```
$ cal
   September 2003
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

This PDF is prepared for tools@maruhouse.org, personal use only.

今月以外のカレンダーを表示するには、その月と年を西暦の4桁で指定する。

```
cal 8 2002
```

月の指定を省略すると(cal 2002)、その年のカレンダーが表示される。

オプション

-y　今年のカレンダーを表示

-m　ビジネスカレンダー(月曜始まり)を表示

-j　1月1日からの日数を表示。例えば9月1日は244、9月2日は245

date [オプション] [書式]　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

dateは、日付と時刻を表示するコマンドである。引き数を指定しない場合、ローカルタイムゾーンでのシステム日付と時間を表示する(デフォルト)。

```
$ date
Sun Sep 28 21:01:31 EDT 2003
```

表示形式を変更するには、+(プラス)を前に付けて、書式を指定する。

```
$ date '+%D'
09/28/03
$ date '+The time is %l:%M %p on a beautiful %A in %B'
The time is  9:01 PM on a beautiful Sunday in September
```

This PDF is prepared for tools@maruhouse.org, personal use only.

書式指定	意味	例
日付と時間		
%c	日付全部と 12 時間制の時間	Sun 28 Sep 2003、09:01:25 PM JST
%D	日付を数値で、年表示は 2 桁	09/28/03
%x	日付を数値で、年表示は 4 桁	09/28/2003
%T	時間を 24 時間制	21:01:25
%X	時間を 12 時間制	09:01:25 PM
名前		
%a	曜日（省略形）	Sun
%A	曜日（完全形）	Sunday
%b	月名（省略形）	Sep
%B	月名（完全形）	September
%Z	タイムゾーン	JST
%p	AM、あるいは PM	PM
数字		
%w	週の何日目（0-6、0 は日曜日）	0
%u	週の何日目（1-7、1 は月曜日）	7
%d	日付（0 パディングあり）	02
%e	日付（0 パディングなし）	2
%j	年の何日目（0 パディングあり）	005
%m	月の何日目（0 パディングあり）	09
%y	年。2 桁表示	03
%Y	年。4 桁表示	2003
%M	分（0 パディングあり）	09
%S	秒（0 パディングあり）	05
%l	時間。12 時間制（0 パディングなし）	9
%I	時間。12 時間制（0 パディングあり）	09
%k	時間。12 時間制（0 パディングなし）	9
%H	時間。24 時間制（0 パディングあり）	09
%N	ナノ秒	737418000
%s	UNIX Epoch（1970 年 1 月 1 日 00:00:00）からの秒数	1068583983
その他		
%n	改行文字	
%t	タブ文字	
%%	% を 1 つ	

This PDF is prepared for tools@maruhouse.org, personal use only.

上記オプションに加え、date は、以下のオプションも解釈する。

オプション

-d *date_or_time_string*　*date_or_time_string* で指定した書式で表示
-r *filename*　指定した書式で、filename ファイルの最終更新日時を表示
-s *date_or_time_string*　システムの日付／時刻を設定。root ユーザのみ実行可能

ntpdate ntpサーバ名　ntp

/usr/sbin　stdin　stdout　- file　-- opt　--help　--version

ntpdateは、ntpサーバと通信を行い、現在のシステム時間を設定するコマンドである。システム時間の設定には、root 権限が必要である。

```
# /usr/sbin/ntpdate timeserver.someplace.edu
7 Sep 21:01:25 ntpdate[2399]: step time server 178.99.1.8 offset 0.51 sec
```

定常的にシステム時計をntpサーバと同期するには、ntpdを用いる。詳細は、http://www.ntp.org/を参照のこと。ntpサーバのホスト名が分からない場合は、Googleで“public ntp time server”と検索してみてほしい。

9.4 ジョブ管理

sleep　指定した秒数の間、待機
watch　一定期間おきにプログラムを起動
at　指定した時間に 1 回ジョブを実行
crontab　指定した時間に定期的にジョブを実行

特定の時間にプログラムを実行させる必要がある、あるいは一定間隔で実行させたい場合など、Linux ではいくつかのツールを提供している。

sleep 時間指定　coreutils

/bin　stdin　stdout　- file　-- opt　--help　--version

sleepは、単に指定した時間(秒)待機するコマンドである。時間の指定は、整数(秒と解釈

This PDF is prepared for tools@maruhouse.org, personal use only.

される)、あるいは整数の後に s を付ける(同様に秒と解釈される)。m(分)、h(時間)、d(日)などが指定できる。

```
$ sleep 5m                    何もせずに 5 分間待機
```

sleep は、コマンドの実行をある一定時間遅らせるような場合に便利なコマンドである。

```
$ sleep 10 && echo 'Ten seconds have passed.'
(10 秒経過)
Ten seconds have passed.
```

watch [オプション] コマンド

procps

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

watchは、一定間隔でコマンドを実行するコマンドである。引き数を何も指定しない場合は、2秒おきにコマンドが実行される。このコマンドは、シェルに渡され(このためクォートやエスケープの扱いはシェルと同じ)、その結果はフルスクリーンモードで画面に表示されるため、出力結果を動的に確認することが可能である。

例えば、watch -n 60 date と実行すると、date コマンドが 1 分に 1 回実行されるため時計代りにもなる。なお、^C で停止する。

オプション

-n *seconds*　実行間隔を秒で指定

-d　変更された出力結果をハイライト表示する。前回と今回の実行において異なる箇所を強調表示

at [オプション] 時間指定

at

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

at は、特定の時刻にシェルコマンドを 1 回実行するコマンドである。

```
$ at 7am next sunday
at> echo Remember to go shopping | Mail smith
at> lpr $HOME/shopping-list
at> ^D
<EOT>
job 559 at 2003-09-14 21:30
```

This PDF is prepared for tools@maruhouse.org, personal use only.

atは、時間指定をとても柔軟に解釈する。一般的に、以下に示す指定が可能である。

- dateコマンド形式の時刻の指定(timeコマンド形式とは異なる)
- 日付のみ(現在の時刻と解釈する)
- 時刻のみ(直近の時刻、今日、あるいは明日と解釈する)
- nowやmidnightやteatime(16:00)などのキーワード指定
- 上記の指定と“+3 days”などでオフセットの指定

設定可能な日付の指定書式は様々である。december 25 2003、25 december 2003、december 25、25 december、12/25/2003、25.12.2003、20031225、today、thursday、next thursday、next month、next yearなどである。

月名は、3文字の省略記法でも指定可能である(jan、feb、marなど)。時間の指定も8pm、8 pm、8:00pm、 8:00 pm、20:00、2000など、柔軟に指定できる。これらは全て同一の意味である。

オフセットは、空白文字の後に+か-の記号を用いて指定する。その後に、時間の指定を行う。+ 3 seconds、+ 2 weeks、- 1 hourなどである。

日付と時間を指定しない場合、atは不足している情報をシステムの日付と時間から補完する。例えば“next year”は、現在から1年後と解釈される。“thursday”は次の木曜日となり、“december 25”は次の12月25日となる。“4:30pm”は一番近い次の午後4：30となる。

atコマンドの実行時には、指定したコマンドはシェルによって評価されず、実行時に初めて評価される。このため、ワイルドカードや変数などシェルが解釈する情報は、実行時まで展開されない。また、現在の環境情報(printenvで参照可能)は保持される。しかし、aliasはatコマンドで指定したコマンドでは使用できないため、atで実行するコマンド内では使用しないように注意してほしい。

atで指定したジョブの一覧は、atq(at queue)コマンドで表示できる。

```
$ atq
559  2003-09-14 07:00 a smith
```

atのジョブを削除するには、atrm(at remove)コマンドでジョブ番号を指定する。

```
$ atrm 559
```

オプション

-f *filename*　　指定したファイルからコマンドを読み込む
-c *job_number*　　ジョブで指定したコマンドを表示

This PDF is prepared for tools@maruhouse.org, personal use only.

crontab [オプション] [ファイル]

vixie-cron

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

crontab は、at と同様に時間を指定してコマンドを実行するコマンドである。しかし、crontab は繰り返しのジョブなどに向いており、例えば「毎月第二火曜日の 0：00 に特定のコマンドを実行する」場合などには有用である。この場合、crontab ファイルを編集する。

```
$ crontab -e
```

自動的にシステムディレクトリ（/var/spool/cron）にインストールが行われる。指定した時間になると、cron が起動し、指定したジョブが実行される。

$ crontab -e
　crontab ファイルの編集には、デフォルトエディタ（$EDITOR）が用いられる
$ crontab -l
　crontab ファイルの内容を標準出力に表示
$ crontab -r
　crontab ファイルの消去
$ crontab myfile
　crontab ファイルに myfile をインストール

スーパーユーザは、-u *username* オプションを用いて、他のユーザの crontab ファイルを操作できる。

crontab ファイルでは、1 行に 1 つのジョブを記述する（空行と # で始まるコメント行は無視される）。各行は、空白文字で区切られた6つのフィールドからなる。最初の5つは、ジョブの時間指定のフィールドである。最後の6つ目は、ジョブのコマンド名を設定するフィールドである。

分

整数 0 から 59 を指定する。1 つの指定（30）、複数の指定はコンマで区切る（0、15、30、45）、範囲指定（20 - 30）、複数＋範囲指定（0 - 15、50 - 59）、*（アスタリスク）は、全て指定することを示す。*n* 回ごとの指定については、/*n* という表記を用いる。例えば、*/12 と 0-59/12 は同じ指定で、0、12、24、36、48（12 分ごと）を表す。

This PDF is prepared for tools@maruhouse.org, personal use only.

時間

分と同一のシンタックスを用いる。

日

整数1から31で指定する。分と同様に、複数指定、範囲指定、複数＋範囲指定、*(すべて)が指定可能である。

月

整数1から12で指定する。分や日と同様に、複数指定、範囲指定、複数＋範囲指定、*(全て)が指定可能である。さらに、月の3文字省略表記(jan、feb、mar...)も指定可能であるが、この場合には複数指定、あるいは範囲指定はできない。

曜日

整数の0(日曜日)から6(土曜日)で指定する。同様に、複数指定、範囲指定、複数＋範囲指定、*(全て)が指定可能である。さらに、曜日の3文字省略表記(sun、mon、tue...)も指定可能であるが、同様にこの場合にも複数指定、あるいは範囲指定はできない。

実行コマンド

ログイン環境で実行可能な全てのシェルコマンドを記述できる。ここでは$HOMEなどの環境変数も使用できる。コマンドのパスは絶対パスで指定すること(例：whoではなく*/usr/bin/who*)。設定例を表11に示す。

表11　crontabの時間指定例

* * * * *	毎分
45 * * * *	毎時45分(1:45、2:45など)
45 9 * * *	毎日午前9:45
45 9 8 * *	毎8日の午前9:45
45 9 8 12 *	毎年12月8日の午前9:45
45 9 8 dec *	毎年12月8日の午前9:45
45 9 * * 6	毎週土曜日の午前9:45
45 9 * * sat	毎週土曜日の午前9:45
45 9 * 12 6	12月の毎週土曜日の午前9:45
45 9 8 12 6	毎年12月8日の午前9:45と、毎週土曜日の午前9:45

コマンド実行時に標準出力があるような場合には、cronはその内容をメールで送信する。

This PDF is prepared for tools@maruhouse.org, personal use only.

9.5　画像とスクリーンセーバ

eog	画像を表示
gqview	画像を表示、スライドショー
ksnapshot	スクリーンショット（画面キャプチャ）
gimp	画像ファイルの編集
gnuplot	グラフ作成
xscreensaver	スクリーンセーバー

Linuxでは、画像表示や編集ツールが多数提供されており、そのすべてを紹介することはできない。そこで、本書ではその中のいくつかのツールを紹介する。

eog [オプション] [ファイル]　　eog

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

eog（eye of Gnome）は、様々なフォーマットに対応したイメージビューアである。1つのファイルを指定すると、そのファイルを表示する。

```
$ eog file1.jpg file2.gif file3.pbm
```

上記のように2つ以上のファイルを指定すると、別々のウィンドウで表示する。

eogのオプションは、技術的なものがほとんどであるため説明は省略する。eogには、多数のオプションが提供されている。オプションについて知りたい場合、eog --help を実行すればよい。

gqview [オプション] [ファイル]　　gqview

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

gqviewは、様々なフォーマットに対応したイメージビューアである。スライドショーのように自動的に次のファイルを表示できる。デフォルトでは、カレントディレクトリに存在する画像ファイル名を全て表示し、その中から画像を選択できる。メニューはとても簡単であるためすぐに慣れるはずである。終了する際には、^q と入力する。

This PDF is prepared for tools@maruhouse.org, personal use only.

オプション

-f　　フルスクリーンモードで表示(v で通常のウィンドウモードとの切替え(トグル))
-s　　スライドショー表示(s でスライドショーの on ／ off)

ksnapshot [オプション]　　kdegraphics

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

ksnapshot は、多機能なスクリーンキャプチャを行うコマンドである。単に、

```
$ ksnapshot
```

と実行すると、スクリーンショットが取得され、そのサムネイルが表示される。その後、画像を保存する、あるいは別のスクリーンショットを取得できる。ファイルの出力形式は保存時にファイル名によって指定できる。一般的な .jpg という拡張子を指定すると JPEG ファイル、.bmp だと Windows ビットマップファイル、.pbm だと portable ビットマップファイル、.eps だと encapsulated Postscript、.ico だと Windows アイコンファイルとなる。サポートしているファイル形式一覧は、Save Snapshotボタンをクリックすると、フィルタ選択部分に表示される。

さらに詳しい情報は、ksnapshotのHelpボタンか、シェルからksnapshot --help-allと実行してみてほしい。

gimp [オプション] [ファイル]　　gimp

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

GIMP(GNU image manipulation program)は多機能画像編集パッケージである。非常に強力な画像編集機能を持ち、Adobe Photoshopがそのライバルである。使用法は非常に複雑であるが、提供される豊富な機能には驚くべきものがある。

全ての情報は http://www.gimp.org/ から参照できる。起動方法は以下のとおり。

```
$ gimp
```

ファイルを編集したい場合は、

```
$ gimp filename
```

と実行する。

This PDF is prepared for tools@maruhouse.org, personal use only.

個人用途で使用するにしてはGIMPが複雑すぎると思われる場合は、それ以外の簡単な編集ソフトも利用可能である。http://www.trilon.com/xv から xv をダウンロードして使う。画像ファイルを表示するには、

```
$ xv myfile.jpg
```

と実行する。画像上でのマウスの右ボタンクリックで、編集メニューとツールが表示される。

gnuplot [オプション] [ファイル]

gnuplot

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

gnuplotは、グラフ描画ソフトである。例えば点をプロットして、それらを直線や曲線で繋げることなどが可能である。そして、その内容をファイルやプリンタにプロッタ形式で保存する(例：Postscript)。gnuplot を実行するには、簡潔であるが強力なプログラミング言語を使用する。例として $y = x^2$ で x = 1 から 10 の曲線を X 上に表示する。

```
$ gnuplot
gnuplot> plot [1:10] x**2
gnuplot> quit
```

以下のコマンドで同じことが実行可能である。ファイルはポストスクリプト形式のファイルで保存される。

```
$ echo 'set terminal postscript; plot [1:10] x**2' | gnuplot > output.ps
```

詳細は、http://www.gnuplot.info/ を参照のこと。

xscreensaver

xscreensaver

/usr/X11R6/bin　stdin　stdout　- file　-- opt　--help　--version

xscreensaver は、多機能なスクリーンセーバである。多くのアニメーションが使用可能である。バックグラウンドで動作させる場合の制御方法が多数存在する。

何も操作をしない時間

何も操作をしない時間が一定時間(デフォルトでは 5 分)経過すると、Fedora の GUI(GNOMEやKDE)では、xscreensaverが自動的に実行される。この時間はメニューから

This PDF is prepared for tools@maruhouse.org, personal use only.

設定可能である。GNOMEの場合、メインメニューから個人設定/スクリーンセーバを選択する。KDEではControl Centerを起動して、Appearance & ThemesからScreen Saverを選択する。GNOME と KDE 以外ではデスクトップ上でマウスを右クリックすると、Configure Desktop が選択され、そこから Screen Saver を選択する。

スクリーンロック

GNOME、あるいはKDEでは、いつでもメインメニューからスクリーンロックを選択可能である。ログインパスワードを入力するまでスクリーンがロックされる。

コマンドラインで

xscreensaver-demoを実行すると、実行可能であるアニメーションのプレビューが行える。また、そこでは好みの設定を行うことができる。設定後に xscreensaver-command を実行して、詳細を設定する。

```
$ xscreensaver-command -activate  画面をクリア
$ xscreensaver-command -next      次のアニメーションを選択
$ xscreensaver-command -prev      前のアニメーションを選択
$ xscreensaver-command -cycle     ランダムにアニメーションを実行
$ xscreensaver-command -lock      画面をロック
$ xscreensaver-command -exit      終了
```

9.6 音と動画

grip	CD プレーヤ、リッパー、MP3 エンコーダ
xmms	オーディオファイル(MP3、WAV)プレーヤ
cdparanoia	CD から WAV ファイルへリッピング
audacity	音楽ファイルの編集
xcdroast	GUI ベースの CD 書き込みソフト

Linux では、オーディオファイルを扱うことができる。ここで説明するほとんどのプログラムは、多数の機能、かつ直感的なユーザインタフェイスを提供している。また丁寧なドキュメントも提供されているため、詳細は説明しない。ここでは主に、これらのソフトの概要についてのみ説明する。Linux のオーディオや MIDI に関して詳細は、以下のサイトを参照のこと。

```
http://linux-sound.org/
http://www.xdt.com/ar/linux-snd
```

This PDF is prepared for tools@maruhouse.org, personal use only.

Fedora coreのパッケージにはビデオプレーヤは含まれてないが、パッケージをダウンロードしてインストールすることができる。有名なものとしてmpg321（http://mpg321.sourceforge.net/）、smpeg（http://www.lokigames.com/development/）、mplayer（http://www.mplayerhq.hu/）などがある。

grip [オプション] grip

/usr/bin stdin stdout - file -- opt --help --version

gripは、CDプレーヤ兼オーディオリッピングソフトである。このソフトはCDの再生、CDからオーディオトラックのリッピング、wavファイルへの保存、mp3への変換などが行える。

cdparanoia [オプション] トラック番号 [出力ファイル名] cdparanoia

/usr/bin stdin stdout - file -- opt --help --version

cdparanoia は、CD からオーディオデータをリッピングして、wav ファイルに保存する（他の形式については man 参照のこと）。コマンド例を以下に示す。

```
$ cdparanoia N
  N 曲めをファイルに保存
$ cdparanoia -B
  CD の全曲を個別のファイルに保存
$ cdparanoia -B 2-4
  2、3、4 トラックを個別のファイルに保存
$ cdparanoia 2-4
  2 から 4 曲めを 1 つのファイルに保存
```

ドライブにうまくアクセスできない場合は、cdparanoia -Qvs と実行してみる（CD-ROM ドライブの検索を強制的に行うオプション）。wav ファイルを mp3 ファイルに変換するには、LAME（http://lame.sourceforge.net/）かNotLame（http://www.idiap.ch/~sanders/not_lame/）を使用すればよいだろう。

This PDF is prepared for tools@maruhouse.org, personal use only.

xmms [オプション] [ファイル]

xmms

/usr/bin　stdin　stdout　- file　-- opt　--help　--version

xmms（X multimedia system）は、優秀なGUIオーディオファイルプレーヤである。サポートしているファイル形式は、mp3、wav、Ogg Vorbis などである。プレイリストを作成するなどCDプレーヤと同じように制御できる。操作方法を学ぶ一番簡単な方法は試してみることである。引き数なしで以下のように実行する。

```
$ xmms
```

もしくは、引き数に再生するファイルを指定する。

```
$ xmms file1.mp3 file2.wav file3.ogg ...
```

操作方法を以下に示す。

操作	意味
タイトルバーを右クリック	メインメニューの表示
PL ボタンをクリック	プレイリストの表示（Add ボタンのクリックでファイルの追加）
EQ ボタンをクリック	グラフィックイコライザの表示
プレイリストのトラックをダブルクリック	指定したトラックを再生
プレイリストを右クリック	プレイリストメニューの表示

audacity [ファイル]

audacity

Fedora Core3 には含まれていない　stdin　stdout　- file　-- opt　--help　--version

audacity は、GUI ベースのオーディオファイルエディタである。WAV、MP3、Ogg をサポートしている。ファイルをロードすると、オーディオデータの波形を確認することができる。さらにオーディオデータのカットやペーストができ、またフィルタの設定を行える。それ以外にも、サウンドの特殊効果（エコー、ベースブースト、逆再生など）が可能である。audacity は、Fedora Core に含まれていないが http://audacity.sourceforge.net/ からダウンロードすることを推奨する。

This PDF is prepared for tools@maruhouse.org, personal use only.

xcdroast [オプション]

xcdroast

/usr/bin　　stdin　stdout　- file　-- opt　--help　--version

xcdroastは、GUIベースのCD作成ソフトである。CDドライブの情報は以下のようにして取得できる。

```
$ cdrecord -scanbus
```

詳細は、CD Writing HOTWO（http://www.tldp.org/HOWTO/CD-Writing-HOWTO.html）を参照のこと。さらに、xcdroastの実行前に、http://www.xcdroast.org/を読んでおくことをお勧めする。設定が特殊な場合は、特にFAQが役に立つ。その後、以下のように実行する。

```
$ xcdroast
```

設定をクリックして、全ての設定が正しいことを確認する。「設定を保存」をクリックして、OKを押してメイン画面に戻る。そして、「CD/DVDの複製」か「CD/DVDの作成」のどちらか操作したい方を選択する。システムの設定によっては、CDの書き込みにroot権限が必要なこともある。

This PDF is prepared for tools@maruhouse.org, personal use only.

This PDF is prepared for tools@maruhouse.org, personal use only.

索　引

記号

A

B

C

D

This PDF is prepared for tools@maruhouse.org, personal use only.

E

F

G

H

I

J

K

L

This PDF is prepared for tools@maruhouse.org, personal use only.

M

N

O

P

R

S

T

This PDF is prepared for tools@maruhouse.org, personal use only.

U

V

W

X・Y・Z

あ行

か行

This PDF is prepared for tools@maruhouse.org, personal use only.

さ行

た行

な行

は行

This PDF is prepared for tools@maruhouse.org, personal use only.

ま行

や・ら・わ行

This PDF is prepared for tools@maruhouse.org, personal use only.

●訳者紹介

猪俣 敦夫（いのまた あつお）

独立行政法人科学技術振興機構。専門は、光伝送制御技術、DWDM 光測定技術等。現在、主に情報セキュリティに関する研究に従事。訳書に『Linuxのためのsendmail』『Linuxセキュリティ全書』（ピアソンエデュケーション：共訳）等。博士（情報科学）。

岡本 健（おかもと たけし）

筑波大学システム情報工学研究科講師。専門は、暗号理論、情報セキュリティ。現在はプライバシ保護や暗号アルゴリズムの研究に従事。著書に「科学大辞典 第 2 版」（丸善出版：分担執筆）等。博士（情報科学）。

田淵 貴昭（たぶち たかあき）

株式会社インターネット総合研究所。現在、主にLinuxを用いたシステム構築に従事。訳書に『Linux のための sendamail』『Linux セキュリティ全書』（ピアソンエデュケーション：共訳）、『DNS&BIND クックブック』（オライリージャパン）等。修士（情報科学）。

This PDF is prepared for tools@maruhouse.org, personal use only.

Linuxハンドブック —機能引きコマンドガイド

2005年8月29日　初版第1刷発行
2005年12月8日　初版第2刷発行

著者　Daniel J. Barrett（ダニエル・J・ブレット）
訳者　猪俣 敦夫（いのまた あつお）
　　岡本 健（おかもと たけし）
　　田淵 貴昭（たぶち たかあき）
発行人　ティム・オライリー
印刷　株式会社ルナテック
製本　株式会社越後堂製本
発行所　株式会社オライリー・ジャパン
〒160-0002　東京都新宿区坂町26番地27　インテリジェントプラザビル1F
Tel　（03）3356-5227
Fax　（03）3356-5263
電子メール　japan@oreilly.com
発売元　株式会社オーム社
〒101-8460　東京都千代田区神田錦町3-1
Tel　（03）3233-0641（代表）
Fax　（03）3293-6224

Printed in Japan（ISBN4-87311-237-0）
乱丁、落丁の際はお取り替えいたします。

本書は著作権上の保護を受けています。本書の一部あるいは全部について、株式会社オライリー・ジャパンから文書による許諾を得ずに、いかなる方法においても無断で複写、複製することは禁じられています。

This PDF is prepared for tools@maruhouse.org, personal use only.