目次

1. Linuxとは.....5
予習編.....6
1.1 はじめに.....8
1.1.1 インターネットとは.....8
1.1.2 サーバーとは.....9
1.1.3 構築するサーバー群.....9
1.2 Linux とは.....11
1.2.1 OS とは.....11
1.2.2 Linux の特徴.....12
1.2.3 Linux の構成.....13
1.2.4 ディストリビューション.....14
1.3 Linux の基本的な利用法.....15
1.3.1 ユーザーアカウント.....15
1.3.2 ログイン・ログアウト.....15
1.3.3 管理者ユーザー root.....17
1.3.4 コマンドライン上での操作.....17
1.3.5 X Window System.....18
1.3.6 コンピューターの起動と停止.....19
1.4 確認問題.....20
column.....21
2. Linux の基本操作(1).....25
予習編.....26
2.1 ファイル操作コマンド.....28
2.1.1 ファイルとディレクトリ.....28
2.1.2 ファイル一覧とカレントディレクトリ表示.....30
2.1.3 ディレクトリの移動.....30
2.1.4 特殊なディレクトリ.....31
2.1.5 ファイルのコピー.....31
2.1.6 ファイルの参照.....32
2.1.7 ファイル名変更と移動.....32
2.1.8 ファイルの削除.....33
2.2 Linux のファイルシステム.....34
2.2.1 ファイルシステムとは.....34
2.2.2 ルートディレクトリ以下の主なディレクトリ.....34
2.3 ディレクトリ操作コマンド.....35
2.3.1 ディレクトリの作成.....35
2.3.2 ディレクトリの削除.....35
2.3.3 ディレクトリの移動・コピー.....36
2.4 確認問題.....37
3. Linux の基本操作(2).....39
予習編.....40
3.1 ユーザー・グループの管理.....41
3.1.1 ユーザー・グループとは?.....41
3.1.2 ユーザーの作成.....41
3.1.3 ユーザーの情報.....42
3.1.4 ユーザー ID.....43
3.1.5 ユーザーホームディレクトリ.....43
3.1.6 ユーザーの削除.....44
3.1.7 グループの作成.....44
3.1.8 グループへのユーザー追加と変更...46
3.1.9 グループの削除.....46
4. Linux の基本操作(3).....47
予習編.....48
4.1 ファイル.....50
4.1.1 ファイルの属性.....50
4.1.2 ファイルモード.....51
4.1.3 パーミッション.....52
4.1.4 リンク.....54
4.1.5 所有ユーザー・所有グループ.....55
4.2 確認問題.....56
column.....57
5. Linux の基本操作(4).....63
予習編.....64
5.1 vi.....65
5.1.1 vi とは.....65
5.1.2 編集モードとコマンドモード.....66
5.1.3 移動.....67
5.1.4 削除.....68
5.1.5 行挿入.....68
5.1.6 コピー・カット・ペースト.....69
5.1.7 保存と終了.....70
5.2 確認問題.....71
column.....72
6. Linux の基本操作(5).....77
予習編.....78
6.1 仮想コンソール.....79
6.2 プロセス.....80
6.2.1 マルチタスクの仕組み.....80
6.2.2 プロセスの管理.....81
6.2.3 プロセス管理に関するコマンド.....82
6.3 ジョブ.....86
6.3.1 ジョブ.....86
6.3.2 フォアグラウンドジョブとバックグラウンドジョブ.....86
6.3.3 jobs コマンド.....87
6.3.4 ジョブの操作.....87
6.4 確認問題.....89
7. Linux の基本操作(6).....91
予習編.....92
7.1 リダイレクション.....93
7.1.1 標準入出力.....93
7.1.2 出力リダイレクション.....94
7.1.3 入力リダイレクション.....96
7.2 パイプライン.....97
7.2.1 標準入出力の連結.....97
7.2.2 リスト.....98
7.2.3 標準入出力処理に関連するコマンド98
7.3 確認問題.....99
column.....100
8. Linux の基本操作(7).....109
予習編.....110
8.1 ファイルシステムの概要.....111
8.1.1 ファイルシステムの形式.....111
8.1.2 ファイルシステムのマウント.....112
8.2 ファイルシステムの操作.....113
8.2.1 mount と umount コマンド.....113
8.2.2 /etc/fstab ファイル.....114
8.2.3 df コマンド.....115
8.3 RPM パッケージ管理.....117
8.3.1 パッケージの照会.....118
8.3.2 アプリケーションのインストール.....118
8.3.3 アプリケーションのアンインストール.....118
8.4 yum を使った更新.....119
8.4.1 GPG key の入手.....119
8.4.2 リポジトリーファイルの修正.....119
8.4.3 yum の実行.....121
8.5 確認問題.....123
column.....124
付録 練習と確認問題の解答.....131
第 1 章.....132
第 2 章.....133
第 3 章.....137
第 4 章.....139
第 5 章.....142
第 6 章.....143
第 7 章.....146
第 8 章.....148
ダウンロード.....150

1 Linuxとは

本章のねらい

- Linuxとはどのようなものなのか
- Linuxはどのように利用するのか

予習編

スマートフォン、タブレット、パソコンといった情報デバイスは、もはや現代の生活には、なくてはならないものとなっています。家電店やパソコンショップに行けば、さまざまなデバイスがあります。皆さんの中にま複数のデバイスをお持ちの方も多いと思います。

そんなデバイスの中で、パソコンの主流は「Windows」でしょう。量販店にあるパソコンのほとんどは「Windows」対応です。他にも「Macintosh」（Mac OS）がありますし、タブレットやスマートフォンではAndroid や iPad/iPhone (iOS)があります。

ここで Windows や Mac OS、Android という言葉が出てきましたが、これらは「オペレーティングシステム：Operating Software」と呼ばれるソフトウェアを表します。OS は、「コンピューターを操作するための基本となるソフトウェア」で、キーボードやマウスといったコンピューターの基本操作や、アプリケーションの起動、ファイルの管理などを行ないます。

皆さんがこれから学ぼうとしている「Linux」も、OS の一つです。Windows や Mac は身近に感じると思いますが、Linux は聞きなれない方も多いと思います。しかしインターネットを扱う上では標準ともいえるOS で、Amazon、Google、Facebook、Yahoo!といったネット企業の多くで採用されています。

しかし Linux は Windows や MacOS とは全く異なる操作方法なので、はじめて触る人はほとんどが戸惑います。Linux は基本的に人とコンピューターのやりとりを「文字で行なう CUI[1] (

Character User Interface)」ですが、Windows や MacOS ではマウスを使って、クリックやドラッグという「図形で行なう GUI (Graphic User Interface）」です。

例えばファイルをコピーするとき Windows はアイコンを右から左に移動させるだけなのにたし、Linux では下記のようにキーボードから文字を入力しコマンド（命令）として指示します。

```
cp  file1.txt   dir1/file1.txt
```

つまり Linux を使いこなすには、この呪文のようなコマンドを覚える必要があります。

本章は Linux の入門として以下を解説しています。

Linux の特徴

Linux の特徴としては、インターネットと親和性の高い「強力なネットワーク機能」、同時に複数のユーザーが利用できる「マルチユーザーシステム」があります。これらの機能を生かして必要なサーバーが構築できます。また、Linux はプログラムの設計図であるソースコードを公開している「オープンソース」方式で、脆弱性・不具合の発見や修正が迅速にできます。

Linux の構成

旧来「Linux」とは OS の中核をつかさどる機能をさしていました。これを「カーネル（核）」と呼びます。カーネルだけでは機能不足なので、アプリ共通機能である「ライブラリ」および CUI でユーザとやり取りをする「シェル」は必須です。

初期の Linux では、これらをユーザーが手当てする必要があったのですが、現在ではこれらをまとめて提供する「ディストリビューション」があります。ディストリビューションは数多くあり「Red Hat Enterprise Linux」「CentOS」「Debian」「Ubuntu」「SUSE」などが有名で、特にソフトウェアアップデートに独自の工夫がなされています。

1　CLI : Command Line Interface とも言います。

1.1 はじめに

OECD によれば 20 世紀が工業品を製造・購入するといった「モノ社会」だったすれば、21 世紀はサービスによる経験の共感といった「コト社会（経験）」だと定義されます。従来は CD や雑誌をお店で買い、個人で所有していましたが今はネットでダウンロードまたはストリーミングする時代です。さらに、作家やアーチストの評判や、店舗ごとの価格などの比較情報もネットから入手している人が多いでしょう。

1990 年代中旬に公開されたインターネットは、今や情報インフラの中核として、必要不可欠です。インターネットのおかげで、いつでも、どこにでも必要な情報を得ることができます。

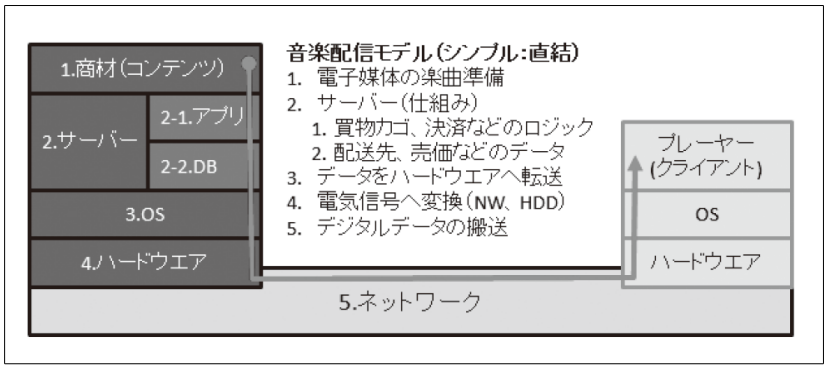
Linux ベーシック」や「Linux マスター」コースでは、この様なインターネットを介し、いろんな「サービスを利用できる仕組み」を解説します。

1.1.1 インターネットとは

1995 年まで、今のインターネットは一般公開されていませんでした。当時は FAX の仕組を使い音楽 1 曲をダウンロードするのに 20 分近くかかる低速でした。インターネットが公開されることで世界中が 1 つのネットワークでつながりました。

さてネットワークを高速道路に例えるならば、音楽という荷物を扱うトラックや倉庫・店舗が必要です。このように情報を作り、管理し、送り届けてくれるのサーバーです。

楽曲という商品を扱うネットビジネスを考えてみましょう。商材としてのコンテンツを「電子化する機能」、アーティスト名、曲名、ジャンルなどから欲しい曲を「検索する機能」、それをお客さんに届ける「お客さんの情報を蓄積する機能」が必要です。このような機能は応用ソフト(アプリケーション・ソフトウェア、略してアプリ)と呼ばれています。



1.1.2 構築するサーバー群の紹介

Linux ベーシックコース、マスターコースで構築するサーバーは以下の通りです。

WWW（World Wide Web）

ウエブサーバーは Apache HTTP Server を使用して簡単なホームページを表示します。応用として CGI プログラムをシェルスクリプトで作成したカウンター、ユーザー名とパスワードによる認証を実験します。ブラウザーは Firefox を使います。

DNS（Domain Name System）

インターネットの電話番号にあたる IP アドレスと、www.yahoo.co.jp といったドメイン名を変換するサーバーです。授業では BIND を利用します。

SSH（Secure Shell）

ネットワークを経由して他の Linux へログインや、ファイルの送受信を行うサーバーです。このときクライアントとサーバーの間は暗号化されるためセキュリティが高い設計になっています。授業では open-ssh を使用します。

電子メール

電子メールは送信サーバーと受信サーバーを構築します。実際に yahoo!メールや gmail のユーザーへ送信したり、クラスの中での送受信を行います。送信用に Postfix、受信用に Dovecot、クライアントとしては Thunderbird、Sylpheed を使います。

ファイル共有

ネットワークを介したファイル共有を行います。実習では Linux マシン相互でのファイル共有を行う NFS（Network File System）と、クライアントを選ばずアップロード・ダウンロード可能な FTP サーバーを構築します。授業では ProFTPD を使用します。

セキュリティ

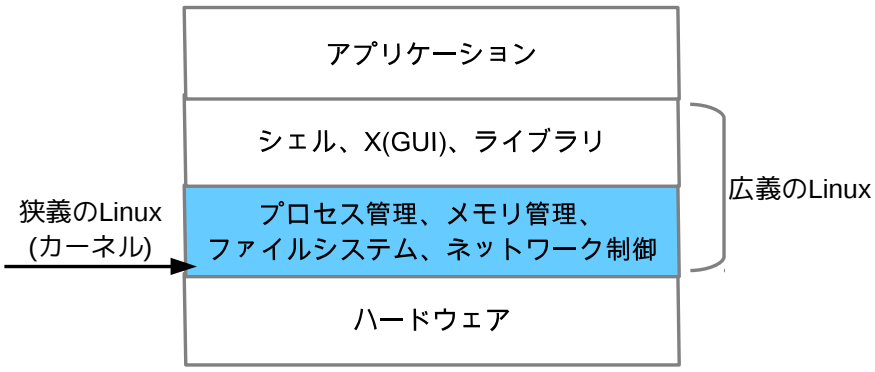
独立したサーバーではありませんが、上記のサーバーを安定稼働させ安全にサービスを提供するための仕組み（監視、不正アクセス防止、ウイルス対策など）も学びます。

1.2 Linux とは

1.2.1 OS とは

コンピューターを動かすための基本ソフトとして、OS（Operating System）があります。通常、OS がなければコンピューターを動かすことはできません。また、OS が異なれば、同じコンピューターを利用していても、異なる動作を示します。

OS は CPU、メモリー、ハードディスク、キーボード、ディスプレイ等のハードウェアの管理や、その上で動くアプリケーションの処理などを管理します。



Windows や Mac OS も OS の 1 つであり、Linux も同様です。他にも色々な OS が研究目的、工業用途など様々な分野で利用されています。

上図は Linux 上の機能分類を表したもので、OS の根幹部分はカーネルと呼ばれ無償で公開されています。これだけではアプリケーションを実行できないため、さらに機能を追加したものが一般に Linux と呼ばれ、その組み合わせによって RHEL や Debian のように名称が異なっています。

1.2.2 Linux の特徴

Linux の代表的な特徴を解説します。

CUI ベースのシステム

Windows と Linux の大きな違いとして、ユーザーインターフェースの違いが挙げられます。

Windows は、アイコンやメニューといった図形対象をマウスで操作します。「図形対象を介してユーザーと対話する」操作方式を GUI（Graphical User Interface）といいます。

一方 Linux は画面に文字が並んでいるだけです。操作はキーボードから命令（コマンド）を入力して行います。この「文字で操作方式」を CUI（Character-based User Interface）又は CLI（Command Line Interface）といいます。

```
 8:32pm  up 2 min,  1 user,  load average: 0.38, 0.18, 0.06
35 processes: 34 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  0.0% user,  0.2% system,  0.0% nice, 99.8% idle
Mem:   126704K av,   57732K used,   68972K free,       0K shrd,   19968K buff
Swap:  262072K av,       0K used,  262072K free                   21396K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
  808 wnn        9   0  3968 3968   612 S     0.0  3.1   0:00 jserver
  881 xfs        9   0  3460 3460   868 S     0.0  2.7   0:00 xfs
  781 root       9   0  2068 2068  1512 S     0.0  1.6   0:00 sendmail
  920 root      11   0  1316 1316  1036 S     0.0  1.0   0:00 bash
  731 root       9   0  1268 1268  1076 S     0.0  1.0   0:00 sshd
  565 root       9   0  1124 1124   448 S     0.0  0.8   0:00 klogd
  963 root      15   0  1112 1112   904 R     0.1  0.8   0:00 top
  912 root       9   0  1076 1076   856 S     0.0  0.8   0:00 login
  821 bin        9   0  1040 1040   692 S     0.0  0.8   0:00 cannaserver
  752 root       9   0  1004  988   816 S     0.0  0.7   0:00 xinetd
  601 rpcuser    9   0   840  840   724 S     0.0  0.6   0:00 rpc.statd
  501 root       9   0   804  804   688 S     0.0  0.6   0:00 dhcpcd
  560 root       9   0   788  788   664 S     0.0  0.6   0:00 syslogd
  833 root       9   0   660  660   576 S     0.0  0.5   0:00 crond
  893 root       9   0   600  600   532 S     0.0  0.4   0:00 anacron
  579 rpc        9   0   588  588   504 S     0.0  0.4   0:00 portmap
  905 daemon     9   0   568  568   500 S     0.0  0.4   0:00 atd
    1 root       8   0   520  520   452 S     0.0  0.4   0:05 init
```

本来 OS としての Linux には、GUI 機能ありませんが、アプリとしての X Window System により、GUI が実現できます。

マルチユーザーシステム

個人向け PC の OS は1台のコンピューターを利用できるユーザーは1人です。しかし、Linux では1台のコンピューターを複数のユーザーがネットワークを介して同時に利用できます。

強力なネットワーク機能

Linux は高度なネットワーク機能を持っています。先祖である UNIX がインターネット創世記から、標準 OS として採用され、現在でもインターネット上の標準的 OS という位置付けになっています。

オープンソースソフトウェア

Linux はソースコード（ソフトウエアの設計図）が公開されています[*3]。これを OSS（Open Source Software）と呼びます。それにより、バグの発見と修正を素早く行うことが可能です。Linux 以外にも、PostgreSQL や Maria-DB といったデータベースエンジン、Apache Web サーバーや FireFox ブラウザ、クラウド・コンピューティングの基板となる OpenStack など数多くの OSS が存在します。

1.2.3 Linux の構成

広義の Linux は以下の構成要素からなります。

カーネル（OS の中核）

一般的に OS に必要とされる重要な役割としては、

- プログラムの起動と停止（プロセス管理）
- ハードウエアの管理（リソース管理）
- データの保存と管理（ファイルシステム）

などがあります。他にも重要な機能がありますが、このような OS としての基本的な機能を「カーネル」（kernel、核）と呼びます。狭義に「Linux」は、このカーネルを[*4]を指します。

ライブラリ（アプリケーション間の共有機能群）

上記のカーネルだけでは「音楽を再生する」、「絵を描く」といった機能が不足していて、アプリケーションを動かすことができません。アプリケーションを動かすために「ライブラリ」（library）と呼ばれる共有機能群必要です。OS とアプリケーションを結びつける「のり」のような働きをします。ライブラはプログラム部分集合で、アプリケーションは実行時にライブラリを呼び出し、OS と連係した動作を行うことができるのです。

*3 ソースコードは公開されていますが、二次利用については制限もあるためライセンス規約に注意してください。
*4 実際には、ディストリビューション全体を指して Linux という場合が多々ある。

シェル（ユーザーと OS が対話するためのプログラム）

ライブラリはアプリと OS との連携に使われますが、これだけではユーザーが利用する環境が揃ったとはいえません。たとえば、メニューがなかったら、アプリを起動できません。

そこで用いられるのがユーザーからの要求を受け付け、アプリを起動するプログラムです。これを「シェル」（shell、貝殻）と呼びます。カーネルを核とし、それを包むシェル、さらにその外にユーザーを見立てています。シェルはキーボードからユーザーが「コマンド」を入力し操作します。様々なコマンドを駆使して、Linux を操作します。

1.2.4 ディストリビューション

Linux カーネルは、OS の中核ですが、日常利用するにはカーネル以外に、前述おライブラリ、ツール類が必要となります。他にも、必須ではありませんが、あると便利なアプリもあります。また、そういった機能やアプリを管理するツールも必要です。これらを個人で全て揃えることも可能ですが、かなりの労力を要します。

そこで、Linux は通常「ディストリビューション」という固まりで配布されています。ディストリビューションは、Linux カーネル ＋ 必要（便利）なライブラリ・アプリの集合を指します。ディストリビューションは、いろんな形態があり「商用」と呼ばれるものは有償で利用時のサポート（保守・運用）を提供してくれます。

授業では、商用で最も普及している RHEL（RedHat Enterprise Linux）と 96%以上互換がある無償の CentOS（Community Enterprise OS、https://www.centos.org）を利用します。

1.3 Linux の基本的な利用法

1.3.1 ユーザーアカウント

Linux では同時に複数の人が利用できます。この利用者を「ユーザー」と呼びます。

システムを利用できる人は、予め登録されたユーザーに限定されます。この利用する権利のことを「アカウント」といいます。ユーザーごとにパスワードが設定され、ユーザー名とパスワードによって本人確認を行います。

1.3.2 ログイン・ログアウト

ログイン（login）

Linux のシステムを利用するためにはログイン（ログオン）が必要になります。Linux では予め登録されたユーザーしか使えないようにしており、ログインすることで「私は登録された正規のユーザーで、今から利用を開始します。」と宣言します。

リナックスアカデミーで PC の電源を投入すると、以下の画面が表示されます。

登録されたユーザ（student）が青く表示されるので、その箱をクリックします。

パスワード欄に「himitu」と入力し（文字は表示されず●が並びます）、［サインイン］をクリックします。

コンソールと呼ばれる、以下の様な文字だけの画面の場合は「login」に対し、「student」と入力します。

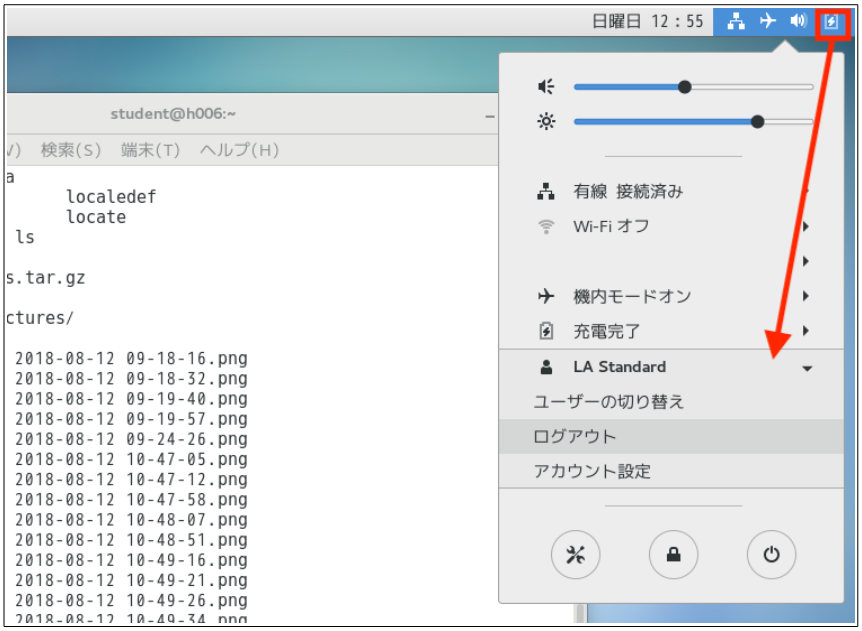
```
localhost login： student
Password： himitu        ←表示されません
```

続いて Password の項目には「himitu」と入力しますが、非表示です。これは盗み見により「なりすまし」されないための対策です。正確に入力できれば、ログインに成功し、システムを利用することができるようになります。途中で入力ミスに気づいた時は、[Ctrl]+[U]で最初から入力し直すことができます。

ログアウト（logout）

利用を終えるときにはログアウトという手続きが必要になります。

メニューの一番右側の「▼」ボタンをクリックし、メニューを表示します。ユーザー名（この例では LA Standard）の「▼」をクリックし、ログアウトを選択します。



文字だけの画面の場合は、端末上でキーボードから以下を入力します。

```
$ exit
あるいは
$ logout
```

もしくは[Ctrl]+[D]を同時に押します。なお logout はコンソールからログインした場合のみ有効です。

1.3.3 管理者ユーザー root

全ての UNIX/Linux では root というユーザーが存在します。これは管理者で、スーパーユーザー[*6]と呼ばれます。

スーパーユーザーは、一般ユーザーにはできない特殊なコマンドの利用や、システムの設定を変更することができます。例えば、コンピューターの停止作業はスーパーユーザーしかできません。

root と一般ユーザーを区別する簡単な方法としては、画面に表示されている文字が挙げられます。一般ユーザーでは、

```
$
```

と入力を待機しているプロンプトが表示されますが、root の場合は、

```
#
```

というように$ではなく、#が表示されます。

［練習］

1. ログアウトします。
2. root でログインします。

```
localhost login: root
Password: himitu      ←表示されません。
```

3. プロンプトが#になっていることを確認します。
4. ログアウトします。

1.3.4 コマンドライン上での操作

CUI 操作は基本的に、

コマンドを入力して、［Enter］キーを押す。
コマンドの実行結果が表示される。

という一問一答の対話形式で行います。この対話に用いるのが［Enter］を押すまでの一連の文字列です。コマンドから始まる一連の文字列をコマンドラインと呼、空白で区切られます。

```
コマンド　オプション　引数　　[Enter]
```

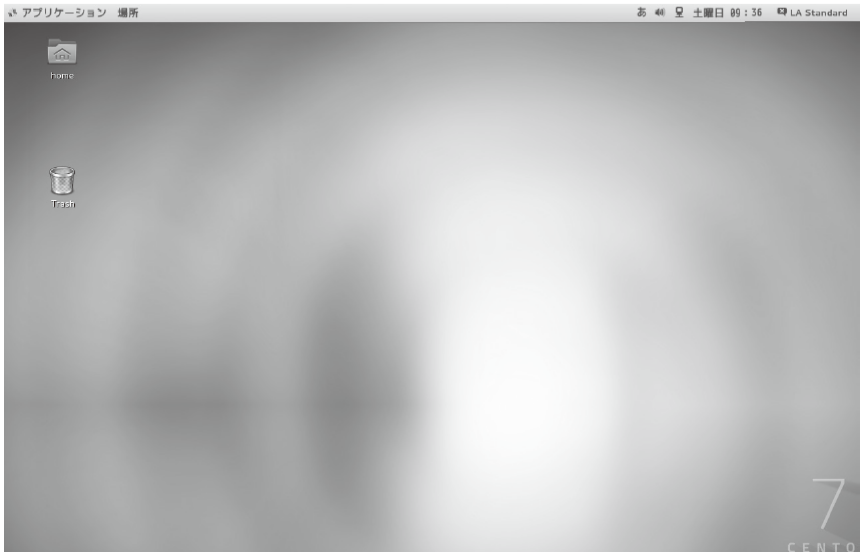
コマンドに続き、ハイフン（－）で始まるオプションや、引数（引数）が続きます。オプションや引数は複数ある場合や、省略される場合があります。オプションはコマンドの動作を指定し、引数はコマンドが操作する対象を表します。

*6　root は、システムを破壊することが可能です。システム管理には、そのくらい強力な権限が必要なのです。

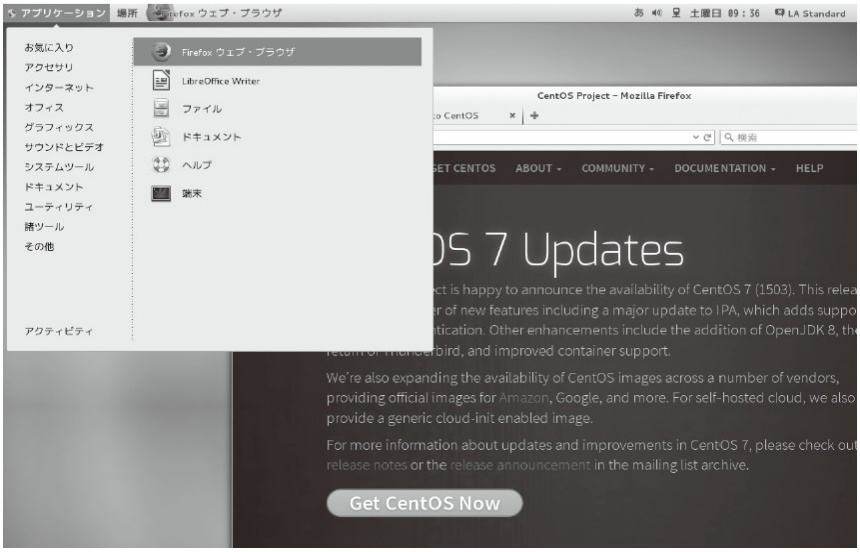
1.3.5 X Window System

X Window System（X と略される）によって、Linux でも GUI が利用できます[8]。コンソールから X を起動するには startx コマンドを利用します。

```
$ startx
```



ブラウザ Firefox を起動するには、下の図のようにメニューのアプリケーション＞お気に入り」から選択します。



なお Linux では、背景やメニューと行ったディスクトップ環境を複数から選択することができます。Cent OS では GNOME と KDE が選択できます。

*8 X 起動中に、CUI の画面（コンソール）を利用したいときは、[Ctrl]+[Alt]+[F2] などで利用できます。X に戻るときは、[Alt]+[F1]とします。

1.3.6 コンピューターの起動と停止

コンピューターの起動

コンピュータの電源を投入すると、Linux が起動します。様々なメッセージが表示された後、ログイン画面が表示されます。特に操作は必要としません。

コンピューターの停止

システム自体を停止するためには shutdown コマンドを root ユーザーで実行します。

```
shutdown　[オプション]　時刻
```

主なオプションとしては、

-h　　システムの停止（Halt）
-r　　システムの再起動（Reboot）
-c　　すでに実行した shutdown コマンドの取り消し（Cancel）

があり、時刻は次の形式となります。

now　　いますぐ
+n　　n 分後
hh:mm　時刻予約　hh 時 mm 分

例

直ちにシステムを停止する

```
# shutdown -h now
```

直ちに再起動する

```
# shutdown -r now
```

5分後にシステムを停止する

```
# shutdown -h +5
```

投入した shutdown をキャンセルする。

```
# shutdown -c
```

1.4 確認問題

1. ［Ctrl］+［Alt］+［F2］を同時に押し、コンソールを切り替えます。
2. ユーザー student でシステムにログインします。
3. who コマンドを実行します。

   who コマンドは、誰が現在ログインしているか、ユーザ名、端末、ログイン時刻を表示するコマンドです。

   ```
   $ who
   ```
4. ログアウトします。
5. root でログインします。
6. who コマンドを実行します。
7. ログアウトします。
8. 再度、student でログインします。
9. プロンプトに「su -」と入力し、パスワード要求に、root のパスワードを入力します。

   su（switch user）は他のユーザーを切り替えるコマンドです。

   ```
   su  [-]  [ユーザー名]
   ```

   オプション「-」はユーザー切替時に初期化を行うことを指示しています。ユーザー名を省略すると、root が仮定されます。
10. プロンプトが「#」に変わったことを確認し、shutdown コマンドでシステムを再起動します。
11. 再起動後、再び root ユーザーでログインします。
12. システムが直ちに停止するように shutdown コマンドを入力します。

column

Linux の簡単な歴史

Linux の手本となった UNIX は 1970 年代に AT&T 社ベル研究所により開発が始まった OS で、当初は無料で大学などの研究機関に配布されましたが、1980 年代に入り商用化が進むと、ライセンス管理も厳しくなり、当初の魅力であったオープン性が失われてしまいました。

そこで、同様の OS の必要性を感じた Linus Torvalds(リーナス・トーバルズ)という1人の大学生(当時)によって 1991 年から作り始められました。現在では世界中の開発者により改良されています。
Linus は PC/AT 互換機(1980 年代初めて Windows が稼働したパソコン)で動く MINIX という UNIX と機能面で互換性を持ったほぼフリーの OS を参考に、UNIX 互換カーネルを開発しました。Linux という名称は「Linus が作った UNIX」を省略したものです。
カーネル以外のコマンド類は FSF（Free Software Foundation）の GNU プロジェクトの成果物を採用しています。同様に FreeBSD プロジェクトもあります。

UNIX と Linux の違い

Linux の歴史からも分かるように、現在 UNIX はオープンではありません。過去 UNIX には認定基準が定められていて、この認定基準に準拠したものだけが「UNIX」と名乗ることができました。同様に米国政府が定めた「UNIX」相当の「POSIX」規格もあります。
これに対して Linux は、UNIX を模して作られた完全にオープンな OS です。GPL というライセンス形態に基づいて、誰でも自由に入手でき、改変や再配布が可能です。更に、Linux は、ネットワークやセキュリティの面で優れているため、スーパーコンピュータからスマートフォンや自動車などの「組み込み系」に至るまで幅広い分野で使用されている OS です。

カーネルのバージョン

カーネルは日々改良されていて、その変化は非常に早く2～3カ月で更新されています（http://www.kernel.org）。改良に伴う内容を正確に把握するために、バージョン管理が導入されています。
カーネルのバージョンは 3 つの数字を小数点で区切って表わし、例えば 4.18.4 や 2.6.18 といった表記になります。各数字は左から順に、メジャーバージョン、マイナーバージョン、リビジョンと呼んでいます。

メジャーバージョン	大幅な改修を表し過去のプログラムが動作しない可能性がある
マイナーバージョン	機能追加を表し新機能を使わなければ、およそ問題がない
リビジョン	バグ・脆弱性の対応を行った

複数バージョンのカーネルが同時に公開されているため、更新中のバージョンを mainline、メジャーやマイナーを更新が停止された時点で Stable（安定版）とよばれます。また利用者が多く Stable 以降もリビジョンが更新されているものは longterm（長期保証）と呼ばれますが、最後は EOL（End of Life、退役）となり更新されなくなります。

ディストリビューション

Linux カーネルの無償で提供され、いつでも・どこでもインターネットから入手できますが、ライブラリやア

プリを用意する手間が大量になり使いこなすのが大変です。それらソフトウェアの適切な組み合わせを選び、個々の設定を行い、動作確認するなど膨大な作業が必要になります。実際にベーシックコースで実習に使っている環境は、千を超えるソフトウェアがインストールされています。
そこで登場したのがディストリビューションという方法でカーネルと普段よく使うであろうライブラリ、アプリ群を組み合わせて、セット提供しています。
ディストリビューションには、システム管理ツール（特にパッケージ管理ツール：ソフトウェアの構成管理、インストール、アンインストールを行うためのツール）の違いから、大きく Red Hat 系と Debian 系という、2 つの流派に分かれます。

Red Hat 系

Red Hat Enterprise Linux
ディストリビューションの老舗かつ最大手が提供する Linux です。安定性や保守運用サービスなど企業向けの色合いが強く製品名 Red Hat Enterprise Linux（略して RHEL）は事実上の業界標準となっています。
開発元：Red Hat 社（http://www.jp.redhat.com/）

Fedora
RHEL の試作（最新技術の取込）と位置づけられ、更新サイクルが早いのが特徴です。バージョン 6 までは Fedora Core 6 と呼称していましたが、Red Hat 社が無償版の Red Hat Linux の開発をバージョン 9 で終了し、代わりに Fedora Project に支援する形で乗り入れています。更新サイクルが同様に速い Ubuntu に比べ、ライセンスフリーをより厳密に徹底しています。
開発元 :Fedora Project（http://fedoraproject.org/）

CentOS
Community Enterprise Operating System は「コミュニティ（有志）が作る企業向け OS」で、RHEL の完全互換を目指す多機能・高安定な企業向け Linux です。サーバー構築向け OS としての採用も多く、Red Hat 社からの支援も得ています。
開発元：CentOS/Lance Davis（http://www.centos.org/）

Debian 系

Debian GNU/Linux
中立性、柔軟性が高く、完全なフリーソフトウェアを目指しボランティアが開発を進めています。FSF の GNU プロジェクトが開発しており、対応する CPU やアプリの豊富さも特徴です。GNU プロジェクトでは、GNU/Hurd も開発していますが正式リリースには至っていません。
開発元：Debian Project（http://www.debian.org/）

Ubuntu
年2回のマメなバージョンアップと使いやすさ、インストールしやすさ（必要な機能すべてを 1 枚の CD にコンパクトにまとめている）を主眼に設計され、多言語対応と幾つかの廉価ノート PC で採用され急速に普及しています。
開発元：Canonical Ltd. / Ubuntu Foundation（http://www.ubuntulinux.jp/）

KNOPPIX
Debian をベースにしたドイツ製の軽量 Linux で、Live CD を採用しています。周辺機器の認識力が高くインストール時に指示する手間が少ないという特徴があります。
開発元：NOPPER.NET（http://www.knopper.net/knoppix/）

その他

Slackware
古くからあるディストリビューションで、非常にコンパクトな作りになっています。また他のディストリビューションと違いパッケージ管理ツールはありません。そのため他のディストリビューションの素材として利用される事もあります。
開発元：Patrick Volkerding, Slackware team（ http://www.slackware.com/）

openSUSE
ドイツ発祥の Linux で欧州でのシェアが大きく、独自の YaST2 パッケージ管理機能を搭載しています。安定性について評価が高く仮想化ソフトの Xen との相性が良いと言われています。もともと S.u.S.E でしたが、Novell が支援し始めた頃によりフリーな製品を目指すため openSUSE と名称変更されました。
開発元：openSUSE project（http://www.novell.com/ja-jp/linux/）

μClinux
組込み向け（携帯デバイス、マイクロコントローラ）の Linux でメモリー管理ユニットを含まない構成であるため、小さなハードウエアでも動作します。
開発元:http://www.uclinux.org

2. Linux の基本操作(1)

2

Linux の基本操作(1)

～ファイルとディレクトリ～

本章のねらい

- ファイル・ディレクトリ操作のためのコマンドを習得する

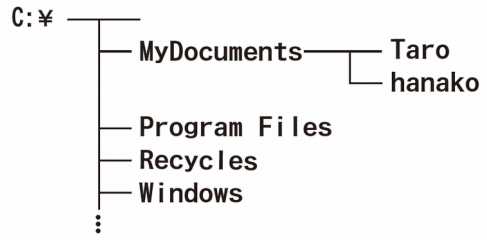
予習編

今回は、Linux の基本的な操作として、ファイルの操作について学びます。

Linux ではファイルとディレクトリの考え方はとても重要です。「ファイル」とは、Windows などのファイルと考え方は同じで、OS がデータを管理するときの基本単位です。Linux を扱う上で、ファイルは非常に重要です。なぜなら、Linux ではアプリの設定など多くの手続きがファイルを介して行なわれるからです。ファイルを自由自在に操れるようになることが、Linux を操れるようになるための第一歩です。ここでは、ファイルをコピーする cp コマンド、中身を参照する cat コマンドや less コマンドを学びます。

ファイルにかかわるもう一つ重要な概念として、「ディレクトリ」があります。ディレクトリは Windows の「フォルダ」に相当するもので、ファイルを格納する場所と形容されます。

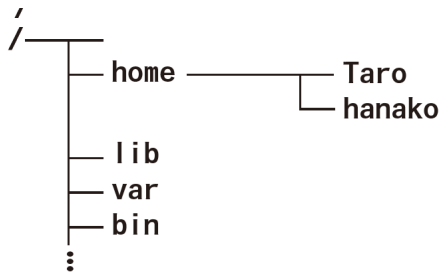
Windows の「エクスプローラ」などで次のようなフォルダの表示方法を見たことがあるでしょう。



Windows では MyDocuments フォルダ内の Taro フォルダの中にある hello.txt というファイルは次のように表されます。

```
C:\MyDocuments\Taro\hello.txt
```

これが Linux ではルートディレクトリ（/）を頂点とした以下の形式になります。



同様に Linux
では hello.txt は次のように表されます。

```
/home/Taro/hello.txt
```

このようにファイルの場所を表記する方法を「パス」と呼びます。ファイルを操作する上で、パスは非常に重要です。

またディレクトリで、大変重要な用語として「カレントディレクトリ」があります。ユーザーはディレクトリを移動しながら Linux の操作を行います。この「ユーザーが今いるディレクトリ」をカレントディレクトリといいます。ディレクトリを移動する cd コマンド、カレントディレクトリを表示する pwd コマンド、ファイルの一覧を表示する ls コマンドは今後頻繁に使います。授業が進行すると当たり前のように使うことになりますので、しっかりと身に付けましょう。

Linux のファイルシステムとディレクトリ構造

「Linux のファイルシステム」では、Linux が採用しているファイルを管理する「ファイルシステム」について解説します。Linux は ext2、ext3、XFS など複数のファイルシステムを採用しています。どれもルートディレクトリ（/）を頂点とした木構造をしています。Windows などの「ドライブ」という概念がなく、全てのファイルはルートディレクトリ傘下に保存されます。

代表的な Linux のディレクトリ構造をイメージで書くと次のようになっています。

```
/ ---+--- bin/
     +--- dev/
     +--- home/
     +--- lib/
     +--- root/
     +--- tmp/
     +--- boot
     +--- etc/
     +--- media/
     +--- proc/
     +--- sbin
```

上に示したディレクトリはルートディレクトリの直下にあるもので、さらにこれらの下にディレクトリが連なった構造をしています。それぞれのディレクトリには、役割の決まったファイルがまとめて置かれます。たとえば、bin ディレクトリの下にはコマンドなどのバイナリファイルがまとめて置かれます。主なディレクトリは、以下の通りです。

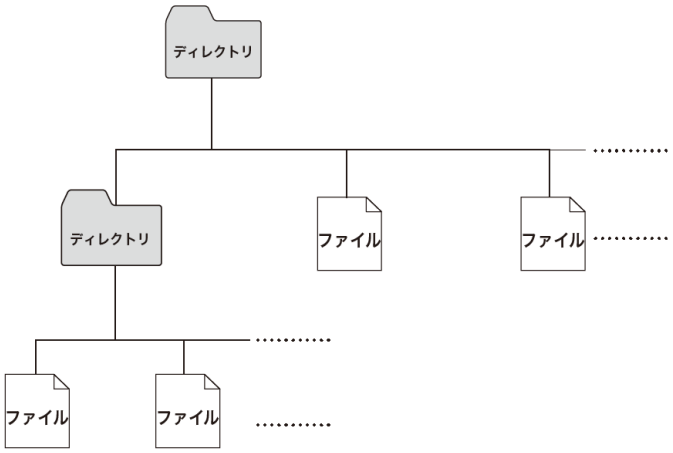
/home ディレクトリ	一般ユーザーのホームディレクトリが置かれる
/root ディレクトリ	root のホームディレクトリ
/etc ディレクトリ	各種プログラムの設定ファイルが置かれる
/bin ディレクトリ	様々なコマンドが置かれる

2.1 ファイル操作コマンド

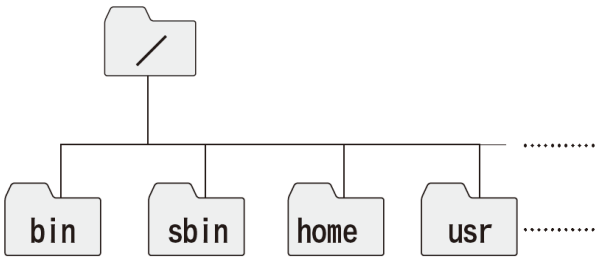
2.1.1 ファイルとディレクトリ

プログラムやユーザーが作成する文書や電子メールなどはすべてファイルという単位でディスク上に保存されます。ファイル名にはアルファベット・数字・記号を用いることができます。アルファベットの大文字・小文字は区別されます*1。記号は「_」「-」「.」だけを使うのがよいでしょう*2。

ディレクトリはファイルを納めるための入れ物、他の OS では「フォルダ」と呼ばれることがあります。ディレクトリの中にもディレクトリ(子ディレクトリ、サブディレクトリと呼ばれる) を作ることができます。ディレクトリ A の中にディレクトリ B があったとき、A から見た B を A ／ B という風に／ (スラッシュ) で区切って表現します。



ファイルやディレクトリは階層構造で管理されています。階層構造の頂点にあるディレクトリをルートディレクトリと呼び、／（スラッシュ）で表します。



*1 Windows などの OS では大文字と小文字は区別されません。hello.txt と HELLO.TXT は同一です。これに対して Linux では hello.txt と Hello.txt、HELLO.txt、HELLO.TXT などは全て区別されます。
2 「」「?」「!」などはメタキャラクタと呼ばれ、特殊な意味を持っているからです。

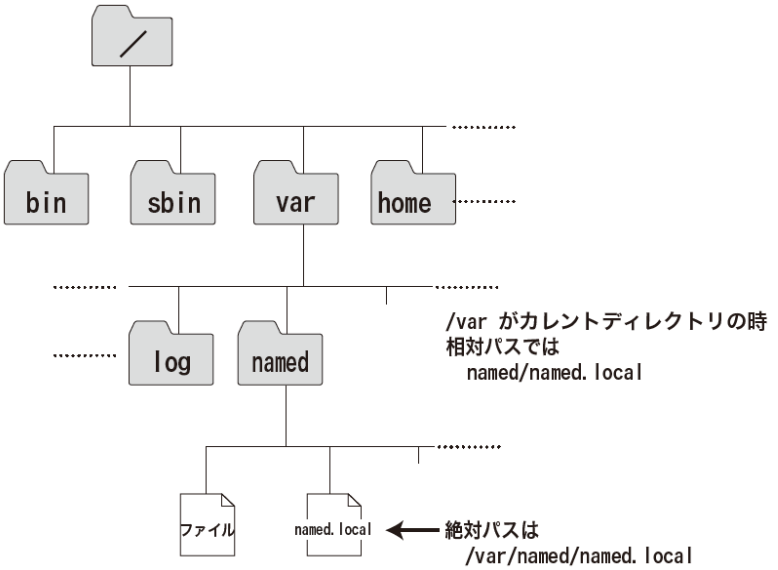
シェル上で作業する時、ユーザーはいずれかのディレクトリの中に所属しています。そのディレクトリのことを「カレントディレクトリ」といいます[*3]。ファイルやディレクトリの位置を var/named/named.local のように／（スラッシュ）で区切って表す表記法をパスと呼びます。パスには、相対パスと絶対パスという 2 つの使い方があります。

相対パス

カレントディレクトリを基点としたディレクトリの指定

絶対パス

ルートディレクトリ（/）を基点としたディレクトリの指定



ユーザーには、作業用スペースとして使える自分専用のディレクトリが用意されていて、これをホームディレクトリといいます。通常、ホームディレクトリはディレクトリ/home の下のディレクトリの下にあるユーザー名と同じ名前のディレクトリとなります。

例

ユーザー student の場合・・・ /home/student

*3　カレントワーキングディレクトリ(Current Working Directory)とも呼ばれます。

2.1.2 ファイル一覧とカレントディレクトリ表示

カレントディレクトリ内のファイルを表示するには ls (list) コマンドを使用します。またカレントディレクトリの絶対パスを表示するには pwd (print working directory) コマンドを用います。

［練習］

1. pwd コマンドでカレントディレクトリの絶対パスを確認します。

```
$ pwd
```

2. ls コマンドで、カレントディレクトリ内になるファイル名を確認します。

```
$ ls
```

3. ディレクトリ/etc/内のファイルの一覧を確認します。

```
$ ls /etc
```

2.1.3 ディレクトリの移動

ディレクトリを移動するには cd（change directory）コマンドを利用します。

```
cd <ディレクトリ名>
```

ディレクトリ名の指定には相対パス・絶対パスのどちらも使えます。ディレクトリ名を指定されていない時は、ホームディレクトリに移動します。

［練習］

1. ディレクトリ /home に移動します。

```
$ cd /home
```

2. pwd コマンドでカレントディレクトリの絶対パスを確認します。
3. ディレクトリ /home/student に移動します。
4. ディレクトリ /etc に移動し、ls コマンドを用いてカレントディレクトリ内のファイル一覧を表示させます。
5. ディレクトリ名を指定せず、「cd」とだけ入力します。pwd コマンドを用いて、どこに移動したかを確認します。

2.1.4 特殊なディレクトリ

パス名称を指定する代わりに、よく使うディレクトリには別名が用意されています。

.	カレントディレクトリ（現在のディレクトリ）
. .	1階層上のディレクトリ（親ディレクトリ）
～[*4]	自身のホームディレクトリ
～ユーザ名	指定ユーザーのホームディレクトリ

［練習］

1. 「cd ..」と入力して、pwd コマンドでカレントディレクトリがどこに移動したかを確認します。
2. 「cd ~」を用いて、ホームディレクトリに戻り、カレントディレクトリを確認します。

2.1.5 ファイルのコピー

cp (copy) コマンドは元のファイルを、別の名前で同じ内容のファイルを作成することができます。

```
cp   <コピー元のファイル名>   <コピー先のファイル名>
cp   <コピー元のファイル群...>   <コピー先のディレクトリ>
```

［練習］

1. cp コマンドを用いて、/etc ディレクトリ内のファイル resolv.conf（/etc/resolv.conf）をホームディレクトリにコピーします。

```
$ cp /etc/resolv.conf ~
```

2. ホームディレクトリに移動して、コピーされたファイルを ls コマンドで確認します。
3. ホームディレクトリ内の resolv.conf を resolv.conf.bak というファイル名でコピーします。
4. ls コマンドを実行して、コピーした2つのファイルがあることを確認します。

*4 「 ~ 」はチルダと読みます。

2.1.6 ファイルの参照

ファイルの内容を表示するには、cat（concatenate、連結の意味）コマンドを用います。

```
cat <ファイル名...>
```

cat コマンドではファイルの中身を表示します。しかし、行数の多いファイルだと表示が早すぎて、とても目で追うことができません。この場合は、less コマンド[*5]を用います。

```
less <ファイル名>
```

[↑][↓] キーで、好きな行へ移動できます。[q]で less を終了させることができます。

［練習］

1. ディレクトリ /etc に移動します。
2. ファイル resolv.conf の中身を参照します。
3. ホームディレクトリに戻り、先ほどコピーしたファイル resolv.conf.bak の中身を cat コマンドを用いて、参照しましょう。コピー元のファイル/etc/resolv.conf と内容が同じであることを確認します。
4. cat コマンドを用いて、ファイル /etc/bashrc を参照します。
5. less コマンドを用いて、ファイル /etc/bashrc を参照します。

2.1.7 ファイル名変更と移動

ファイルを移動するときには、mv（move）コマンドを利用します。cp コマンドと異なり、もとのファイルを残しません。

```
mv   <元のファイル名>     <新しいファイル名>
mv   <移動元ファイル群...>     <移動先ディレクトリ>
```

移動するときにファイル名を変更することもできるので、ファイル名を変更するためにも mv コマンドを利用できます[*6]。

［練習］

1. ホームディレクトリに戻ります。
2. 先ほどホームディレクトリにコピーしてきたファイル resolv.conf をファイル resolv.conf2 に移動します。
3. ls コマンドでファイル resolv.conf がなくなり、ファイル resolv.conf2 に変わったことを確認します。
4. cat コマンドでファイル resolv.conf2 の内容が元のファイルと同じであることを確認します。

*5 類似コマンドに more があります。more は上から順に読むことができますが、逆に戻ることはできません。
*6 ディレクトリ名の変更にも利用できます。

2.1.8 ファイルの削除

ファイルを削除するには rm（remove）コマンドを用います。

```
rm <ファイル名...>
```

他の OS と異なり、削除前に確認や、削除を取り消せる「ゴミ箱」機能はシェルにはありません。ファイル削除のような破壊的操作は慎重に行ってください。

-i オプションを使うと、確認操作を行います（y か n で回答）

［練習］

1. 先ほど作成した resolv.conf.bak ファイルを削除します。
2. ls コマンドで削除されたことを確認します。
3. resolv.conf2 ファイルをホームディレクトリ内の resolv.conf ファイルにコピーし、resolv.conf2 ファイルは削除します。
4. ls コマンドで変更を確認します。

2.2 Linux のファイルシステム

2.2.1 ファイルシステムとは

OS にはファイルがハードディスク上やフロッピー上のどこに書き込まれているのかを管理する役割があります。これを実現するのがファイルシステムです。Linux のファイルシステムの構成はルートディレクトリを頂点として、その下にディレクトリやファイルがある形になっています。全てのファイルはどこかのディレクトリの中に配置されています。ファイルシステムの形式にはいくつかの種類があり、OS ごとに使える形式の種類が異なります[*7]。

2.2.2 ルートディレクトリ以下の主なディレクトリ

ルートディレクトリの下には様々なディレクトリがあらかじめ用意されています。

例

```
$ cd /
$ ls
bin  dev  home  lib64       media  opt   root  sbin  sys   usr
boot etc  lib   lost+found  mnt    proc  run   srv   tmp   var
```

それぞれのディレクトリには用途[*8]があります。

ディレクトリ名	用途
/bin	通常のコマンドなどのバイナリファイルが置かれる
/dev	特殊デバイスファイルが置かれる
/home	一般ユーザーのホームディレクトリが置かれる
/lib	ライブラリが置かれる
/root	root のホームディレクトリが置かれる
/tmp	一時的に作成されるファイルが置かれる
/var	ログファイルなど頻繁に変更されるファイルが置かれる
/boot	起動時に必要なファイル（カーネルのブートイメージなど）が置かれる
/etc	各種プログラムの設定ファイルなどが置かれる
/media	一時的なマウントポイントなどが置かれる
/proc	動作中の OS の情報などが置かれる（メモリー上に存在）
/sbin	管理者用のコマンドが置かれる

*7 Windows では、ファイルシステムとして FAT32 や NTFS が使われていますが、Linux においてもこれらのフォーマットのファイルシステムを読むことができます。FAT32 はファイルシステムとして用いることができます。
*8 ディレクトリの詳しい構成は FHS(Filesystem HierarchyStandard) と呼ばれる規格を参照してください。

2.3 ディレクトリ操作コマンド

2.3.1 ディレクトリの作成

ディレクトリの作成には、mkdir（make directory）コマンドを利用します。

```
mkdir  <作成するディレクトリ名...>
```

[練習]
次のコマンドが持つ意味を考えながら実行します。

```
$ cd ~
$ ls -R
$ mkdir schedule
$ mkdir diary
$ mkdir schedule/work
$ mkdir schedule/work/meeting
$ ls -R
```

2.3.2 ディレクトリの削除

ディレクトリの削除には、rmdir（remove directory）コマンドまたは、「r」オプション付の rm コマンドを用います。両者の違いは以下となります。

```
rmdir：空のディレクトリのみ、削除可能
rm -r：指定したディレクトリ以下のファイル・ディレクトリを再帰的に全て削除
```

「rm -r」は非常に強力なため、ディレクトリ全てを削除してよいか十分に確認して下さい。

[練習]
次のコマンドが持つ意味を考えながら実行します[*9]。

```
$ cd ~
$ ls -R
$ rmdir schedule/work    ← 失敗する
rmdir: `schedule/work/' を削除できません: ディレクトリは空ではありません
$ rm -r schedule/work    ← 成功する
$ ls -R
```

*9 ls コマンドに-R オプションをつけると、サブディレクトリのファイルやディレクトリまで表示します。

2.3.3 ディレクトリの移動・コピー

ディレクトリの移動には mv コマンドを用います。

```
mv　<元のディレクトリ>　<移動先ディレクトリ>
```

ディレクトリ 2 が既に存在する場合は、ディレクトリ 1 はディレクトリ 2 の下に中身ごと全て移動されます。
ディレクトリ 2 が存在しない場合は、ディレクトリ 1 は名前が変更されます。

[練習]
次のコマンドが持つ意味を考えながら実行します。

```
$ mv schedule  diary
$ ls -R
$ mv diary/schedule diary/dream
$ ls -R
```

ディレクトリのコピーには cp コマンドに -r オプションを付けて実行します。

```
cp　-r　<コピー元ディレクトリ>　　<コピー先ディレクトリ>
```

-r オプションはディレクトリ 1 を中身ごと全てディレクトリ 2 にコピーするためのものです。オプションを付けずに cp コマンドをディレクトリに対し実行すると、エラーが出て失敗します。

[練習]
次のコマンドが持つ意味を考えながら実行します。

```
$ cd ~
$ cp -r diary diary2
$ ls -R diary
$ ls -R diary2
```

2.4 確認問題

1. ルートディレクトリに移動します。
2. 次のコマンドを実行し、カレントディレクトリを確認する。

```
$ cd ./etc
```

3. 次のコマンドを実行し、カレントディレクトリを確認する。

```
$ cd
```

4. 次のコマンドを実行し、カレントディレクトリを確認する。

```
$ cd ../../etc
```

5. /etc/postfix/main.cf をホームディレクトリ内にコピーする。
6. コピーした main.cf の内容を参照する。

3. Linux の基本操作(2)

3

Linux の基本操作(2)

～ユーザとグループ～

本章のねらい

- ユーザーやグループの概念について理解する
- ユーザーやグループの管理方法を学ぶ

予習編

Linux には、複数の利用者が同時にログインすることができます。したがって、これらの利用者を明確に区別するために、利用者をユーザーという単位で管理しています。各ユーザーには、ユーザー名、ユーザー ID、パスワードなどが設定されています。ログイン時に入力していたのが、このユーザー名とパスワードです。

Linux は、ユーザーごとに適切な環境を提供し、各ユーザーがお互いの操作の影響を受けないように管理しています。

また、利用者を管理するために、ユーザーのほかにグループという考え方が導入されています。これは、複数の利用者を同時に管理するためのものです。各グループには複数のユーザーが所属できます。

本章では、ユーザーやグループの管理方法（作成・変更・削除）を学びます。また、これらの情報が記述されているファイルについても学ぶことにします。

3.1 ユーザー・グループの管理

3.1.1 ユーザー・グループとは?

Linux では、登録されているユーザーのみがシステムを利用することができます。このように利用ユーザーを限定しておき、ログイン、ログアウトをきちんと行うことによって、ユーザー別にさまざまな設定を行ったり、ログ（操作の記録）を保存したりできるようになります。

Linux は、複数のユーザーが同時に利用するシステムです。もし、システム利用者が複数存在した場合には、それぞれの人別にユーザーアカウントを発行してください。Linux では、各ファイルやディレクトリに対し、所有者・所有グループが決められます。会社の書類に誰もが好き勝手にアクセスできないように、Linux も、システム内に保管されているファイルは、誰でも自由にみたり、編集したりすることはできません。

また、複数のユーザーをグループとしてまとめて管理できます。一般的な会社でも「営業」「経理」というように、従業員の方々がグループとして管理されています。営業グループに所属しているユーザーのみ閲覧できるファイル、というように権限を設定していた方が、管理が簡素化します。

3.1.2 ユーザーの作成

Linux では登録されているユーザーのみがシステムを利用することができます。ユーザーアカウントを作成するには、useradd コマンドを用います。新規のユーザーアカウントを作成することができるのはスーパーユーザー(root) のみです。

```
useradd 〈ユーザー名〉
```

また、ユーザーにはそれぞれパスワードを設定する必要があります。passwd コマンドを用いて設定します。設定するパスワードの入力が求められるので、2 回入力します。

```
passwd 〈ユーザー名〉
```

［練習］

1. root ユーザーになって、ユーザー abe を作成し、パスワードを設定します。

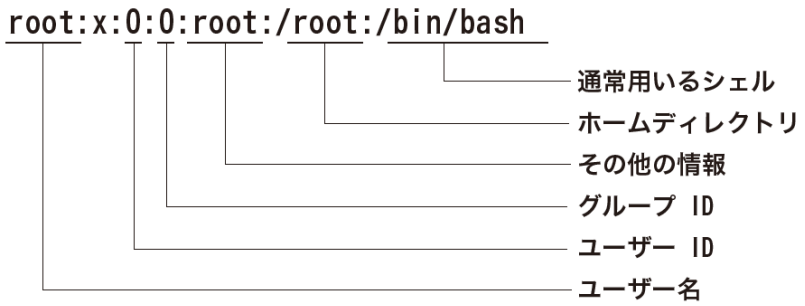
```
# useradd abe
# passwd abe
```

3.1.3 ユーザーの情報

作成したユーザー情報は /etc/passwd ファイルの中に記述されています。

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
            .
            .
            .
student:x:1000:1000:LA Standard:/home/student:/bin/bash
abe:x:1001:1001::/home/abe:/bin/bash
```

1 行目には root の情報が記述されています。最終行には作成したユーザー abe の情報があります。それぞれの行に 1 ユーザーごとの情報が書かれていて、以下のような形式[*1]になっています。



［練習］

/etc/passwd ファイルを参照し、先ほど追加したユーザー abe が記述されていることを確認します。

```
$ less /etc/passwd
```

*1　その他の情報にはユーザーのフルネームや電話番号を記述していましたが、セキュリティ上好ましくない為、現在はあまり利用されていません。

3.1.4 ユーザー ID

ユーザーには全て番号が振られています。これをユーザー ID（UID）と呼びます。最小のユーザー ID は root の 0 となっています。1 ～ 99 はシステム内のプログラムのために予約されていることが多いので、一般ユーザーに割り当てられるユーザー ID は 500 や 1000 以降など、大きな数がよく使われます。

useradd コマンドはデフォルトでは 999 より大きく、（すでに存在するユーザー ID の中の最大値）+ 1 を新規作成するユーザーのユーザー ID とします。ユーザー ID を指定してユーザーを作成するためには、以下のように useradd コマンドを用います。

```
useradd -u <ユーザー ID> <ユーザー名>
```

［練習］

1. ユーザー tanaka をユーザー ID=2000 として[*2]作成します。

```
# useradd -u 2000 tanaka
```

2. /etc/passwd ファイルで、作成されたユーザー tanaka のエントリを確認します。
3. ユーザー yoshida を追加します。

```
# useradd yoshida
```

4. ユーザー yoshida のユーザー ID を確認します。

3.1.5 ユーザーホームディレクトリ

ユーザーのホームディレクトリは、ディレクトリ /home 以下のユーザー名と同じ名前のディレクトリになります。ユーザー abe の場合、/home/abe となります。ログイン直後のカレントディレクトリは、ホームディレクトリになります。ユーザー作成時にホームディレクトリを明示的に指定することもできます。

```
useradd -d 〈ホームディレクトリ〉 〈ユーザー名〉
```

［練習］

1. ユーザー sato のホームディレクトリを/home/dir と指定して作成します。
2. /etc/passwd ファイル内に sato のエントリが記入されていることを確認し、ホームディレクトリがどこになっているかを調べます。
3. 「ls /home」として、/home ディレクトリの中に dir という名のディレクトリが作成されていることを確認します。

*2 実際には/etc/login.defs の中の UID_MIN で 1000 と設定されており、1000 より大きな値がユーザー ID として割り当てられます。

3.1.6 ユーザーの削除

ユーザーを削除するには userdel コマンドを用います。

```
userdel 〈ユーザー名〉
```

［練習］

1. ユーザー tanaka を削除します。
2. ユーザー tanaka のためのホームディレクトリ/home/tanaka は残っていることを確認します

通常、userdel コマンドではユーザーのホームディレクトリは削除されません。削除した後に/home ディレクトリの中を ls コマンドで表示させてみましょう。削除したユーザーの名前と同じ名前のディレクトリがまだ残っているはずです。ホームディレクトリも含めて削除するには ”-r” を付けます。

```
userdel -r 〈ユーザー名〉
```

［練習］

1. ユーザー sato をホームディレクトリごと削除します。

```
# userdel -r sato
```

2. 「ls /home」として、sato のホームディレクトリ「/home/dir」が削除されていることを確認します。

3.1.7 グループの作成

Linux では、複数のユーザーをひとまとめにして扱う時のために「グループ」が用意されています。グループを作成するには groupadd コマンドを用います。グループを作成できるのは root のみです。

```
groupadd 〈グループ名〉
```

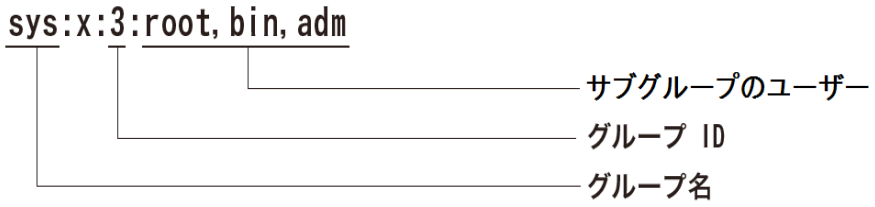
［練習］

1. eigyo という名のグループを作成します。
2. keiri という名のグループを作成します。
3. somu という名のグループを作成します。

この結果、/etc/group というファイルに新たなグループに関する行がファイルの最後に追加されます。/etc/group ファイルには以下のような形式で情報が書かれています。

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
       （省略）
eigyo:x:2002:
keiri:x:2003:
somu:x:2004:
```

各行の項目は次のようになっています。



useradd コマンドでユーザーを作成したときにはユーザー名と同じ名前のグループが作成され、そのグループにユーザーは自動的に追加されます。例えば、abe というユーザーを作成したときには、

```
abe:x:596:
```

などのようになっています。

ユーザーは複数のグループに参加することができます。groups コマンドを用いれば、自分がどのグループに参加しているかがわかります。

```
# groups
root
# groups postfix
postfix : postfix mail
```

ユーザーは必ず 1 つのグループに参加しています。このグループのことをプライマリグループと呼び、そのグループ ID（GID）が /etc/passwd の中に書かれています[*3]。また、プライマリグループ以外に参加しているグループをサブグループと呼び、サブグループに参加しているユーザー情報は、/etc/group の中に書かれています。

*3　id コマンドを使うと UID,GID も表示します。

3.1.8 グループへのユーザー追加と変更

ユーザーが useradd コマンドで作成されて自動的にユーザー名と同じ名前のグループが作成され、そのユーザーのプライマリグループとなります。ユーザーの作成時にプライマリグループを指定するには以下のようにします。

```
useradd -g <グループ名> <ユーザー名>
```

グループ名に指定できるのは既に存在しているグループのみです。

［練習］

1. ユーザー sato のプライマリグループを eigyo に指定して、作成しましょう。
2. /etc/passwd ファイルを参照し、ユーザー sato のプライマリグループが eigyo になっていることを確認しましょう。

また、既存のユーザーの参加するグループを変更するには usermod コマンドを利用します。プライマリグループを変えるには -g オプションを、サブグループの指定には -G オプションを用います。

```
usermod -g <プライマリグループ> -G <サブグループ> <ユーザー名>
```

-G オプションで複数のグループを指定するには、カンマ（,）で区切って指定します。

［練習］

次のコマンドが何を意味しているかを確認します。

```
# usermod -g keiri -G eigyo,somu sato
```

3.1.9 グループの削除

グループを削除するには、groupdel コマンドを利用します。

```
groupdel <グループ名>
```

但し、削除するグループをプライマリグループとしているユーザーが 1 つでもあると、そのグループは削除できません。これは、ユーザーは必ずプライマリグループに属しなければならないという、ルールが保てなくなるためです。

［練習］

1. somu グループをプライマリグループとするユーザー kato を作成します。
2. somu グループが削除できないことを確認します。
3. ユーザー kato を削除します。
4. somu グループを削除できることを確認します。

4. Linux の基本操作(3)

4

Linux の基本操作(3)

～ファイルの属性～

本章のねらい

- ファイルに関する知識を深める
- ファイルのパーミッションについて

予習編

第2章で、ファイルに関する基礎知識を学びました。本章ではファイルに関する知識を更に深め、ディレクトリの操作やファイルシステムなどについても深く掘り下げていきます。

ファイルの種類と属性

Linux はさまざまなファイルで構成されています。プログラムや、ソフトウェア・OS の設定、更に使用者が作成した文書や画像などもファイルになっています。

「ファイル」にはいくつかの種類があり、人間が文字として読める形の「テキストファイル」、文字として読めない形の「バイナリファイル」などがあります。また、ディレクトリも「ディレクトリファイル」と呼ばれるファイルの一種です。Linux では更にキーボードやディスプレイなどの入出力装置も「デバイスファイル」というファイルとして扱われます。そして、Linux にはもう一つ「リンクファイル」というものがあります。これは Windows の「ショートカット」に相当するものです。

ファイルはそれぞれ「属性」と呼ばれる情報を持っています。属性には、ファイル名のほか、ファイルのサイズ、最終更新日時、そして「ファイルの所有者」や「パーミッション」などの情報が書かれています。属性を表示するには、「ls」コマンドに「-l」オプションをつけて実行します。

```
$ ls -l /etc
合計 1848
-rw-r--r--     1     root root 2456   5月 13  2004 DIR_COLORS
-rw-r--r--     1     root root 2434   5月 13  2004 DIR_COLORS.xterm
drwxr-xr-x    15     root root 4096  12月 11 08:56 X11
-rw-r--r--     1     root root 2562   5月 13  2004 a2ps-site.cfg
-rw-r--r--     1     root root 1523   5月 13  2004 a2ps.cfg
.   .   .
```

「ls -l」はファイルの詳細な情報を得ることができるので、今後多用することになります。

ファイルの所有者とパーミッション

マルチユーザーシステムではファイルの所有者とパーミッション（権限）は重要な概念です。多くのユーザーが同じコンピューターを利用する上で、他人にファイルを書き換えられたり、読まれたり、勝手に実行されないよう制限する必要がります。これはプライバシーの問題だけではなく、設定ファイルを一般ユーザーが誤って書き換えシステム障害が発生することを回避するためでもあります。

Linux のファイルには、全て所有するユーザと所属するグループが割り当てられます。さらに誰がそのファイルを読み・書き・実行できるかを、許可・禁止することができます。これらの権限を「パーミッション」と呼びます。パーミッションを設定すると、「このファイルについて、自分は読み・書き・実行すべてが可能、同じ所属グループのユーザーは読込と実行のみ可能、第三者には実行だけ許可する」などといった制限が可能になります。「ls -l」実行結果の、左側に並んでいる英文字列がこのパーミッションを表しています。直前に挙げた例のパーミッションは、「ls -l」コマンドを使うと次のように表示されます。

```
-rwx r-x --x
```

1文字目はファイルの種類を表し、2文字目からの9文字がそのファイルのパーミッションを表しています。所有者・所有グループ・第三者に対するパーミッションはそれぞれ3つずつに区切られ、左から所有者、所有グループ、第三者に対するパーミッションを表します。各記号は、r が読込み（Read）、w は書き込み（Write）、x は実行（eXecute）を表します。ハイフン（-）は許可がないことを表します。

ファイルのパーミッションを設定するためには chmod コマンドを使います。chmod コマンドを使ったパーミッションの設定方法はいくつかあります。余裕のある方は本編の chmod コマンドの使い方を予め見ておくとよいでしょう。

ファイルに別名（リンクファイル）をつけるには「ln」コマンドを使います。リンクファイルは Windows のショートカットのようにディレクトリの移動の手間を省くだけでなくファイルの格納場所を変更するなど、システムの管理でも用いられます。

4.1 ファイル

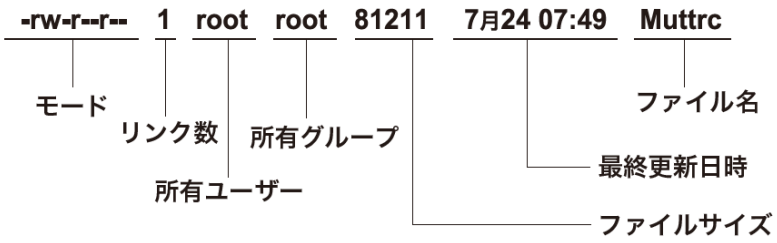
4.1.1 ファイルの属性

/etc ディレクトリの中に移動して、ls コマンドに -l（詳細）オプションを付けて実行します。

例

```
$ cd /etc
$ ls -l
合計 1576
-rw-r--r--.   1 root root 5090   1月 25   2014 DIR_COLORS
-rw-r--r--.   1 root root 5725   1月 25   2014 DIR_COLORS.256color
-rw-r--r--.   1 root root 4669   1月 25   2014 DIR_COLORS.lightbgco
-rw-r--r--    1 root root 94     3月   6  08:49 GREP_COLORS
drwxr-xr-x.   7 root root 4096   7月 12  13:15 NetworkManager
drwxr-xr-x.   3 root root 101    7月   5  09:56 PackageKit
（省略）
```

ファイルの属性情報が表示されます。



- モード
  ファイルの種別と、パーミッション（アクセス権限）
- リンク数
  ファイルが持つ名前の数
- 所有ユーザ
  このファイルの持ち主（ユーザ）
- 所有グループ
  持ち主が権限を委譲する参加グループ
- ファイルサイズ
  ファイルの大きさ（Byte 単位）
- 最新更新日時（タイムスタンプ）
  このファイルが変更された日時（mtime）。他にも参照日時（atime）、属性変更日時（ctime）も記録される。

4.1.2 ファイルモード

ファイルモードは、ファイルの種別とアクセス権を表します1番目の記号がファイル種別を表しています。

記号	ファイルの種類
-	レギュラーファイル
d	ディレクトリファイル
l	シンボリック　リンクファイル
b または c	特殊デバイスファイル

続く6文字（rwx の羅列）がパーミッションを表し、最後が（.や@）がセキュリティ上重要なものを表し、SELinux セキュリティ・コンテンツと呼ばれます。

レギュラーファイル（-）

コマンドなどのプログラム、その設定ファイル、ユーザーが作成する文書などは全てレギュラーファイルです。レギュラーファイルはテキストファイルとバイナリファイルにわけられます。テキストファイルは人が読める文字で書かれたファイルですが、バイナリファイルは人に読めるようには書かれていません。

ディレクトリファイル（d）

ファイルを格納する入れ物に当たります。ディレクトリファイルにはその中に属するファイルやディレクトリの大きさや更新日時といった情報（メタデータ）が含まれています。

シンボリック・リンクファイル（l）

ファイルの別名です。Windows では「ショートカット」、Mac OS では「エイリアス」と呼ばれます。リンクにはシンボリックリンクとハードリンクが存在します。リンクファイルを作成するには ln コマンドを使用します。

特殊デバイスファイル（c、b）

特殊デバイスファイルは入出力装置でキャラクタ、ブロックの2種類があります。

キャラクタデバイス（c、Character device）はキーボードのように1 Byte づつ即、入出力する装置です。ブロックデバイス（b、Block device）はハードディスクやネットワークのように1 KByte などある程度まとめて入出力する装置です。

デバイスファイルの場合、ls コマンドではファイルの大きさではなく、装置種別を表すメジャー番号とマイナー番号が表示されます。

4.1.3 パーミッション

Linux はマルチユーザーシステムです。自分が作ったファイルを他人に見られる恐れもあり、プライバシーや機密情報の保護のためのアクセス権が設定されています。適切な権限を持っているユーザーのみがファイルを見たり、書き込んだり（変更したり）することができるようになっているのです。この権限をパーミッションと呼びます。

パーミッションは読み込み（r）、書き込み（w）、実行（x） の3つの権限があります。

ファイル	読込み	書込み	実行
ディレクトリ	一覧（ls）	ファイル追加・削除	移動（cd）
許可	r	w	x
禁止	-	-	-

この権限を所有ユーザー、所有グループのメンバー、第三者に対し、それぞれ設定します。

	所有ユーザー	所有グループ	第三者
許可	r w x	r w x	r w x
禁止	- - -	- - -	- - -

例

rwx rwx rwx	全てのユーザーに全ての権限を許可
--- --- ---	全てのユーザーの全ての権限が禁止
rwx r-- r--	所有ユーザーは全権、他のユーザは読込み権のみ許可
rw- rw- rw-	全てのユーザーの読込み、書込み権を許可

パーミッションは基本的に所有ユーザーと root のみが設定・変更することができます。変更のために使うコマンドは chmod(change mode) です。

```
chmod （パーミッション） ファイル名...
```

パーミッションの表記方法は記号式と数値式の2つがあります。

記号式

記号を組み合わせることで権限の付与、剥奪を表します。

```
（ユーザー）（操作）（３つの権限）
```

ユーザー

- u ：所有ユーザー（ user ）
- g ： 所有グループのメンバー（ group ）
- o ：第三者（ other ）
- a ：全てのユーザー（ all ）

操作

- + ：続く権限を付与する
- - ：続く権限を剥奪する
- = ：続く権限にする

権限を表す記号は、以下の3種類です。

r ：読み込み権（ read ）
w ：書き込み権（ write ）
x ：実行権（ execute ）

例えば、ファイル test に対し、全てのユーザーに対し読込みを、所有者に書込みを付与する場合は次のようになります。

```
$  chmod  a+r,u+w  test
```

数字を用いる表現

rwx の許可されている部分に「1」、禁止する部分に「0」を割り当てます。

```
rwxrwxrwx → 111111111
```

これを 3 つに区切り、それぞれを 2 進数と考え、10 進数で表します。

```
rwx rwx rwx → 111 111 111 → 7 7 7
```

これは、許可されている権限に以下の重み付けをしたと考えることもできます。

```
r = 4, w = 2, x = 1
```

```
rw- r-- ---    110 100 000    (4+2+0) (4+0+0) (0+0+0)    640
 rwx の表記  →   2進数表記   →        10 進数で計算      →  結果
```

上記のように「rw- r-- ---」を数値で表すと640になります。

例

初期状態が rw-r----- であるファイル test を rwxr-xr-x にしたい場合、記号を用いた表現では

```
$  chmod  u+x,g+x,o+rx  test
```

もしくは

```
$  chmod  a+x,o+r  test
```

3 桁の数字を用いた表現では、

```
$  chmod  755  test
```

となります。

[練習]
次のコマンドが持つ意味を確認しながら実行します。

```
$ cd
$ cp /etc/fstab .
$ ls -l fstab
$ cat fstab
$ chmod a-r fstab
$ ls -l fstab
$ cat fstab
$ chmod 666 fstab
$ ls -l fstab
```

4.1.4 リンク

あるファイルへのリンクを作成するには ln コマンドを使用します。リンクには、ファイルの実態に直接名前をつけるハードリンクと、別のファイルへの飛び先を指定するシンボリックリンクがあります。リンク数はハードリンクされている名前の数を表します。

ハードリンクをファイルシステム（ハードディスク）をまたぐことはできませんが、シンボリックリンクはこの制限がありません。

$	ln	（リンク元のファイル）	（ハードリンク別名）
$	ln -s	（リンク元のファイル）	（シンボリックリンク別名）

[練習]
次のコマンドが持つ意味を確認しながら実行します。

```
$ cd ~
$ touch sample
$ ln sample sample.hard
$ ln -s sample sample.slink
$ ls -l sample*
```

4.1.5 所有ユーザー・所有グループ

全てのファイルはそのファイルを所有するユーザーと所属するグループの情報を持ちます。フ作成したユーザーが所有者で、プライマリグループが所有グループとなります。
ファイルの所有ユーザーを変更するには chown（change owner）コマンドを用います。root のみがこのコマンドを利用できます。chown コマンドは所有ユーザーと所有グループを同時に変更することもできます[*2]。

```
chown   <所有ユーザー>   <ファイル・ディレクトリ名>
chown   <所有ユーザー>:<所有グループ>   <ファイル・ディレクトリ名>
```

ディレクトリ内のファイルも再帰的に変更したいときは-R オプションを付けます。

[練習]
次のコマンドが持つ意味を確認しながら実行します。

```
# cp /etc/hosts /home/student/
# ls -l /home/student/
# chown student /home/student/hosts
# ls -l /home/student/
# chown student:student /home/student/hosts
# ls -l /home/student/
# chown -R root:root /home/student/
# ls -l /home/student/
# chown -R student.student  /home/student/
# ls -l /home/student/
```

所有グループを変更するには chgrp(change group) コマンドを用います。chgrp は root および、そのグループに所属するユーザーが利用できます。

```
chgrp <所有グループ> <ファイル・ディレクトリ名>
```

chown と同様に -R で、サブディレクトリ下の全ファイルを所有グループを変更できます。

[練習]
次のコマンドが持つ意味を確認しながら実行します。

```
# su – sato          ← su を使って abe に切り替えます。
$ cp /etc/hosts .
$ ls -l hosts
$ chgrp eigyo hosts
$ ls -l hosts
```

*2 所有ユーザと所有グループを同時に指定するとき、:（コロン）の代わりに .（ピリオド）を用いることができます。

4.2 確認問題

1. ユーザー student でログインします。
2. student のホームディレクトリ内に以下のようなディレクトリ階層を作成しパーミッションを設定します。

```
/home/student/ (755)
            +-- tmp/ (755)
                  +-- spring/ (750)
                      summer/ (751)
                      fall/   (752)
                      winter/ (754)
```

3. ユーザー abe になります。
4. ユーザー abe が/home/student/tmp/ 以下の各ディレクトリに対して、
   - cd コマンドでディレクトリ中に入れるか？
   - ディレクトリに対し ls コマンドで一覧表示できるか？
   - ディレクトリ内にファイルを作成・削除できるか？

   について確かめ、その理由を考察します。

column

Sticky ビット

全てのユーザーに書き込みが許可されているディレクトリでは、そのディレクトリ内の他のユーザーのファイルまで削除したり、名前を変更したりすることができてしまいます。しかし、sticky ビットをディレクトリに付けると、ファイルの所有者以外のユーザーが、その中のファイルについて名前変更・削除することが不可能になります。例として、/tmp ディレクトリがあります。/tmp はユーザー間で共有されていますが、sticky ビットが設定されています。

sticky ビットの表記は

```
rwxrwxrwt
```

のように第三者の実行権が x の代わりに t となります。また数によるパーミッションの指定では、sticky ビットは 1000 で表されます。

例

```
rwx rwx rwt = 1777
```

sticky ビットをディレクトリに付けるためには、「t」または4桁目に「1」を指定します。

```
chmod o+t <ディレクトリ名>
chmod 1777 <ディレクトリ名>
```

sticky ビットは本来、仮想記憶に配置されたプログラムが優先的にメモリーに居残る(張り付く=sticky)事を指定するために用いられてきました。最近はハードウェアの性能が向上したため、この機能は無用となり、代わりに上記のような削除制限に用いられることになりました。

SUID ビット・SGID ビット

一般ユーザーがプログラムを実行したとき、通常は実行したユーザーがそのプログラムのプロセスのユーザーとなります。SUID ビットを設定すると、実行したユーザーに関係なく、ファイルの所有ユーザーでプロセスを実行します。一般ユーザーが実行するプログラムであるが、管理者権限が必要な場合（passwd コマンドなど）がその例です。

SUID ビットは実行権 x に加え「s」を付与します、数字を用いたパーミッションの表記では 4000 で表します。SUID ビットを付けるときは「s」をユーザに付与します。

```
chmod u+s <ファイル名>
```

SGID ビットは SUID 同様、実行時のグループを実行者ではなく、ファイルのグループとします。実行権「x」に加えグループに「s」を付与します、数字を用いたパーミッションの表記では 2000 で表したりします。SGID ビットを付けるときは「s」をグループに付与します。

```
chmod g+s <ファイル名>
```

但し、SUID/SGID は本来与えられていない権限を許すものであるため、利用は最低限にとどめるようにすべきです。

ちなみに SUID、GUID ファイルを探すには、find コマンドが利用できます。（結果抜粋）

```
$ find /bin/ -perm -u+s -ls
12848955   32 -rwsr-xr-x   1 root     root        32048  4月 11 15:50 /bin/umount
12978448   28 -rwsr-xr-x   1 root     root        27832  6月 10  2014 /bin/passwd
$ find /bin/ -perm -g+s -ls
12632809   16 -r-xr-sr-x   1 root     tty         15344  6月 10  2014 /bin/wall
12848961   20 -rwxr-sr-x   1 root     tty         19624  4月 11 15:50 /bin/write
```

ハードリンクとシンボリックリンク

リンクファイルにはハードリンクとシンボリックリンクの 2 種類があります。2 つの違いはファイルの i-node（アイ・ノード）に関係があります。ハードディスク上のファイルの場所を指し示すのが i-node ですが、この i-node には重複しない数字が割り当てられています。i-node が指し示すディスク上のデータをファイルの「実体」と呼ぶことにします。

<i-node1> → < ファイル1の「実体」>
<i-node2> → < ファイル2の「実体」>

< ファイル1の「実体」> と< ファイル2の「実体」> は<i-node1> と<i-node2> の i-node の i-node ナンバーの違いにより区別される。

普段用いている「ファイル名」はファイルの「実体」を直接指し示すものではなく、ファイルの「実体」を指し示す i-node の i-node ナンバーを指し示すものです。

< ファイル名> → <i-node ナンバー>

ファイルの i-node ナンバーを調べるには、

```
$ ls -i <ファイル名>
```

とします。この結果、

<i-node ナンバー> < ファイル名>

という対応関係が表示されます。

例

```
$ ls -i /etc/httpd/conf/httpd.conf
63547 http.conf
```

あるファイルの実体を指し示す i-node に対応するファイル名の数をこのファイルの実体へのリンク数と呼びます。いくつのファイル名が同じファイルの実体を指し示しているのかがわかります。

例

```
$ ls -l /etc/httpd/conf/httpd.conf
-rw-r--r-- 1 root root 41551 Jul 17 03:16 /etc/httpd/conf/httpd.conf
           ↑ リンク数
```

/etc/httpd/conf/httpd.conf というファイルの場合はリンク数が 1 ですから、i-node ナンバー 63547 のファイルに対応するファイル名は/etc/httpd/conf/httpd.conf ただ 1 つということになります。

ハードリンクの場合、新しくできるリンクファイルは元のファイルの i-node を指し示します。つまり、同じ i-node ナンバーのファイルを直接指し示しているわけです。

/etc/httpd/conf/httpd.conf のハードリンクファイル httpd.conf.hdln を作成したとすると、

```
$ ln /etc/httpd/conf/httpd.conf httpd.conf.hdln
```

ファイル名	i-node
http.conf	63547
http.conf.hdln	63547

のように httpd.conf.hdln は httpd.conf と同じファイルの実体を指し示します。試しにここで httpd.conf のリンク数を見てみましょう。

```
$ ls -l /etc/httpd/conf/httpd.conf
-rw-r--r-- 2 root root 41551 Jul 17 03:16 /etc/httpd/conf/httpd.conf
           ↑ リンク数が１つ増えている
```

これに対し、シンボリックリンクを作成する場合は同じ i-node ナンバーの i-node を指し示すのではなくて、リンク先のファイル名を指し示します。

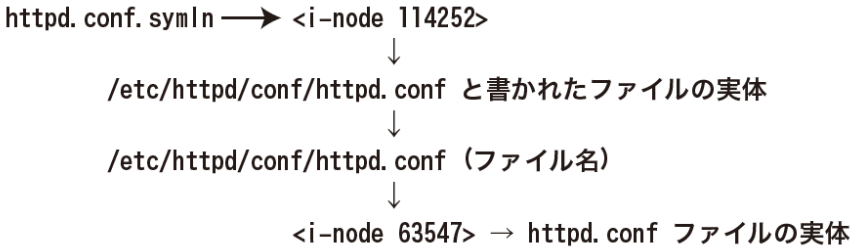
< シンボリックリンク> → < リンク先のファイル名> → <i-node>

/etc/httpd/conf/httpd.conf のシンボリックリンク httpd.conf.symln を作成したとすると

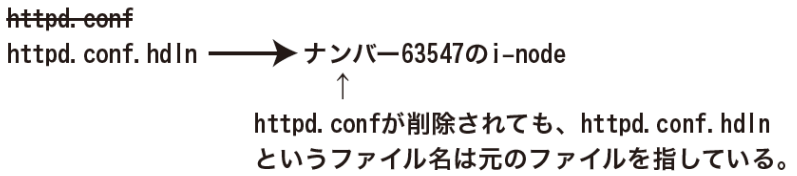
```
$ ln -s /etc/httpd/conf/httpd.conf httpd.conf.symln
$ ls -l /etc/httpd/conf/httpd.conf
-rw-r--r-- 2 root root 41551 Jul 17 03:16 /etc/httpd/conf/httpd.conf
           ↑ 先ほどハードリンクを作成したときから変化していない。

$ ls -i httpd.conf.symln
114252 httpd.conf.symln
```

ということからリンク数は増えず、httpd.conf と同じファイルの実体を指しているのではないことが分かります。i-node ナンバー 114252 が指しているファイルの実体には/etc/httpd/conf/httpd.conf というファイルの名前自体が書かれているのです。

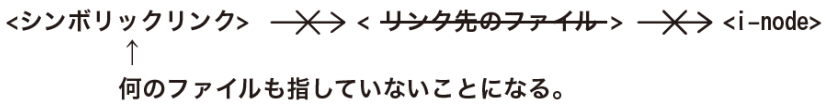


この違いはリンク先のファイルを削除したときに現れます。ハードリンクの場合、リンク先のファイルを削除したとしても、ハードリンク自身はもとの i-node を指し示しているのですから、削除前と同じように参照することができます。



このとき、httpd.conf.hdln のリンク数を調べると、「1」に戻っていることに気付くでしょう。これは i-node を指し示すファイル名の数が httpd.conf.hdln のみとなったからです。

これがシンボリックリンクの場合は、リンク先のファイルが消えてしまうと共に、このファイルが指し示していた i-node への対応も消えてしまうので、シンボリックリンクは何のファイルも参照することができないことになります。



5. Linux の基本操作(4)

5

Linux の基本操作(4)

～vi エディタ～

本章のねらい

- vi の使い方を学ぶ

予習編

vi とは

vi とは、Linux 標準のテキストファイルを編集する「エディタ」です。Windows でのエディタには「メモ帳」「秀丸エディタ」「WZ」などがあります。しかし vi はこれらのアプリケーションとは使い方が全く異なるため、使いこなすためにはある程度の練習を要します。

前回までに述べたとおり、Linux やソフトウェアの設定は、テキストファイルにその設定を書いて行われます。設定ファイルの編集は、vi を使って行うのが基本です。ですから、vi を使いこなせるようになることは、Linux を管理する上でとても重要です。

vi の特徴

vi の特徴として、Linux（に限らず UNIX 系の OS）であれば必ず実装されていること、コンソール上で動作すること、すべての動作をキーボードで行うこと（マウスを使わない）、が挙げられます。vi の操作方法は初心者にとっては難しいと思われることが多いようですが、その理由としては、「ファイルの保存、文字や行の削除、コピー、検索などの操作もすべてキーボードで行う」ことが大きな要因だと、考えられます。

Windows のような GUI を使用したエディタであればこれらの操作はすべてマウスを使用して行うことができますが、vi では「とにかくキーボードから」行います。このような操作もさることながら、当然文字を直接入力するにもキーボードを使用しますので、キーボードひとつではキーが足りません。そのため、vi では文字を入力する「編集モード」と、削除・書換え等といった操作をコマンドにより行う「コマンドモード」の二つがあり、両方のモードを適宜切り替えながら編集作業を行います。

5.1 vi

5.1.1 vi とは

vi(visual editor) はコンソール上で動作するエディタアプリケーションです。他にも多くのエディタがありますが、システム管理では vi がよく用いられます。

vi が Windows などで用いられる多くのエディタと異なる点は、

- 全ての操作をキーボードで行う
- 編集モードとコマンドモードの 2 つのモードがある

ことです。

vi でファイルを作成、編集するには

```
vi <ファイル名>
```

とします。

ファイルを参照のために開くときには、-R オプションを付けて起動すると、書き込みを不可にする Read-Only モードになります。

```
vi -R <ファイル名>
```

[練習]
vi を起動して、hello.txt という名前のファイルを作成します。

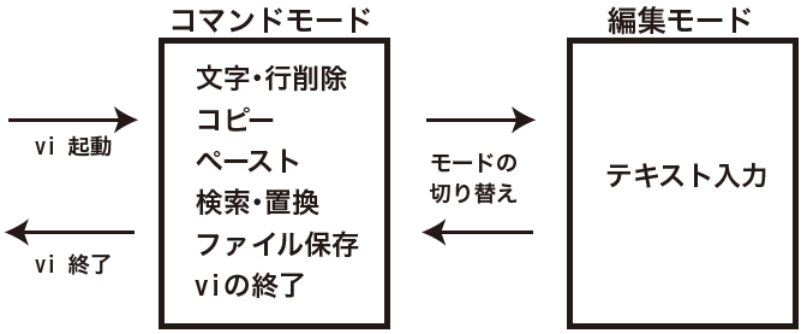
```
$ vi hello.txt
```

vi の画面で[:][q][!][Enter]と入力すると、終了できます。

5.1.2 編集モードとコマンドモード

ファイルを編集する上では文字を入力するという操作以外にも、カーソルの移動、文字や行の削除、コピー・ペーストといった操作が必要になります。vi ではこれらの操作を「コマンド」を用いて行います。

vi にはコマンドを入力するモード(コマンドモード) と文字を入力するモード(編集モード) があります。



vi での操作は上図のようにコマンドモードと編集モードの間でモードの切り替えを行うことにより、ファイルを編集します。起動直後はコマンドモードになっています。

編集モードは挿入、上書きなどが行えます。テキストを挿入するには、コマンドモードで［i］キーを入力します。画面の左下に -- 挿入 -- と出力されて、テキストを入力できる状態になります。コマンドモードに戻るためには [ESC] キーを入力します[*1]。

[練習]

下記の手順によって、ファイル hello.txt を作成します。

```
$ vi hello.txt
     [i] キーを入力して、編集モードに切り替える。


Hello!
I'm using vi on CentOS.
  上の二行を入力する。


[ESC]キーを入力して、コマンドモードに戻る。
[:][w][q][Enter]と入力する。  ←ファイルが保存されて vi が終了。


$ ls
```

*1 vi にはたくさんのコマンドがあり、間違えてキーを入力した際に何かのコマンドが実行されて、どうしていいか分からなくなることもあります。このようなときには、あせらず[ESC]キーを押しましょう。コマンドモードに戻ることができます。

5.1.3 移動

カーソルの移動は初期の vi ではコマンドモードでしか行うことができませんでした。キーボードにカーソルキーがないものが多かったため、別の文字で代替していました。

```
←（左）  :  [h] キー
↓（下）  :  [j] キー
↑（上）  :  [k] キー
→（右）  :  [l] キー
```

コマンドモードで[h][j][k][l]を用いて、1 文字上下左右に移動することができます。しかし、最近のバージョンの vi ではカーソルキーも利用できます。

また、

＜文字数＞＜[h][j][k][l]のいずれか＞

で指定した字数分だけ上下左右に移動します。

例 7文字右へ移動

```
This is a test.
I'm studying to use a vi!
It's not so hard.
     ↓ [7][l]
This is a test.
I'm studying to use a vi!
It's not so hard.
```

他にも行頭に移動や、行末に移動するためのコマンドも用意されています。

コマンド	操作
0	カーソル行の先頭に移動
$	カーソル行の末尾に移動
:1[Enter]	開いているファイルの先頭（1行目）に移動
:$[Enter]	開いているファイルの末尾に移動
w	1 つ右の単語に移動
b	1 つ左の単語に移動
:<行番号>[Enter]	指定された行番号に移動

5.1.4 削除

文字の削除

カーソル下の文字を削除するには[x]（小文字）を入力します。また、[X]（大文字）を入力すると、カーソル左の文字が削除されます。

例

```
abcde
  ↓ [x]
abde
  ↓ [X]
ade
```

行の削除

カーソルがある行を削除するには[d][d]と入力します。

例

```
abcde
  ↓ [d][d]
~           （行が存在しないときは「~」と表示される）
```

5.1.5 行挿入

カーソル行の下に新たに行を追加したい場合には[o]（小文字）を入力します。上に行を追加するには[O]（大文字）を入力します。」

例

```
abc
def
  ↓ [o]
abc
■
def
```

5.1.6 コピー・カット・ペースト

コピー・ペースト

[y][y]を使うと、カーソル行がバッファ（一時的にデータを記憶する場所のことで、Windows の「クリップボード」に相当）に格納されます。格納された行をペースト（貼り付け）するには[p]を入力します。

例

```
It's a show time!
This is a test of "COPY & PASTE!"
     ↓ [y][y]    ←「It's a show time!」がバッファにコピーされる。
It's a show time!
This is a test of "COPY & PASTE!"
     ↓           カーソルを下の行に移動する。
It's a show time!
This is a test of "COPY & PASTE!"
     ↓ [p]
It's a show time!
This is a test of "COPY & PASTE!"
It's a show time!       ←1行目が、最下行にコピーされた。
```

カット・ペースト

[d][d]は行を削除すると同時にその行をバッファに格納します。これを利用すれば、カット・ペーストが可能になります。

例

```
It's a show time!
This is a test of "COPY & PASTE!"
    ↓ [d][d]
This is a test of "COPY & PASTE!"
    ↓ [p]
This is a test of "COPY & PASTE!"
It's a show time!
```

5.1.7 文字列の検索

文字列を検索するには[/]を用います。

```
/検索文字列 [Enter]
```

[/]を入力すると画面最下で文字列入力待ちになります。検索したい文字列を入力し、[Enter]を押します。文字列が見つかると、その先頭にカーソルが位置付きます。

この例では、「/Directory [Enter]」を実行しています。キーワード Directory が強調表示され、先頭の「D」にカーソルが位置づいています。

次を検索するには、[n]と入力します。逆方向に検索を続けるには[N]とします。

最後まで検索すると、先頭に戻って再び検索します。

```
下まで検索したので上に戻ります                    100,4          26%
```

また同様に[?]も利用でき、現在のカーソル位置から先頭に向けて検索を行います。

```
?検索文字列 [Enter]
```

5.1.8 保存と終了

ファイルの保存

開いたファイルを保存するためには、コマンドモードで[:][w]を用います。

```
:w
```

この結果、

```
"<ファイル名>" <行数>L, <文字数>C 書込み
```

と、表示されます。

ファイル名を指定して、保存する際には以下のようにします。

```
:w <ファイル名>
```

vi の終了

編集が終了して、vi を終了したいときには、コマンドモードで

```
:q
```

と入力します。ファイルを保存しないで終了したい場合は

```
:q!
```

とします。ファイルを保存し、終了したい場合は

```
:wq
```

とします。コマンドモードで[Z][Z]と入力しても保存して終了することができます。

[練習]

簡単なアドレスメモを作成します。

```
<名前>    <ニックネーム>
birthday <生年月日>
email <メールアドレス>
hobby <趣味>
```

からなるアドレスデータをファイル address.memo に作成します。

5.2 確認問題

1. ユーザー student のホームディレクトリにある.bashrc を開きます。

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
  . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-
paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

2. このファイルの最後に以下を追加し、保存します。

```
# User specific aliases and functions
PS1="[\u@\h \W \t]\$ "
```

3. student でログインし直します。

```
[student@h200 ~ 00:30:49]$
```

4. プロンプトに時刻が表示されるようになります。

5. 再度、.bashrc を開き、付け加えた部分の先頭に「#」を追加し、プロンプトが元に戻ったことを確認します。
   （多くの Linux 設定ファイルは、#から行末までを注釈として無視します）

column

その他の便利なコマンド

本章で紹介した以外にも、vi には以下のような便利なコマンドもあります。

行番号の表示

行番号を表示するには、以下のように入力します。

```
:set number [Enter]
または
:set nu [Enter]          ← nu は number の省略形
```

行番号を非表示にするには、以下のように入力します。

```
:set nonumber [Enter]
または
:set nonu [Enter]← nonu は nonumber の省略形
```

[Ctrl]+[G]を入力すると、画面の左下にファイル名、ファイルの総行数、現在カーソルがある行が総行数の何%の位置なのか、という情報が表示されます。

```
".bashrc" 11 行 –54%--
```

この例は、ファイル名「.bashrc」、総行数 11 行、現在 54%の位置にカーソルがある、というとこを表します。

置換

文字列の置換（置き換え）をするには、以下のように置換前文字列と置換後文字列を指定し、[Enter]を押します。

```
:%s/置換前文字列/置換後文字列/g [Enter]
```

上記の各記号は以下のような意味になります。

記号	意味
%	置換対象を全ての行とする
s	置換する
g	同じ行に置換対象となる文字列が複数あった場合でも全て置換する

例えば、

```
:%s/user/usr/g [Enter]
```

上記を実行すると、ファイル内の”user”という文字列が全て”usr”に置換されます。

また、特定行を指定したい場合は開始行と終了行をカンマで区切って指定します。

```
:3,5s/user/usr/g [Enter]
```

上記を実行すると、3 行目から 5 行目にある”user”という文字列が全て”user”に置換されます。

コマンドの繰り返しと取り消し

直前に実行したコマンドを繰り返し実行したい場合、[.]（ピリオド）を入力します。
カーソルを移動した後も利用できます。

また、直前に実行したコマンドを取り消したい、あるいは、直前に入力した内容を取り消したい場合は
[u]「u」を入力します。

バッファ

本章で説明したように、[dd]で削除した行や[yy]でコピーした行はバッファに保存されます。このバッファに保存された内容は[p]で取り出せます。

バッファは過去 9 回分の削除内容を、「番号付きバッファ」（名前なしバッファ）保存されます。直前の内容をバッファ1とし、遡るように保存されており過去 9 回分の削除内容を貼り付けることが可能となります。

更に「名前付きバッファ」に保存することもできます。「名前付きバッファ」とは、”a”から”z”までの 26 種類の名前が付いたバッファで、最大 26 種類の内容を保存することができます。

・番号付きバッファ

「番号付きバッファ」に保存された内容を貼り付けるには、「”」の後にバッファ番号と[p]を入力します。

```
"バッファ番号 p
```

例えば、2 回前に削除した内容を貼り付けるには、以下のように入力します。

```
"2p
```

・名前付きバッファ

「名前付きバッファ」に内容を保存するには、以下のように入力します。

```
"バッファ名 yy        ←指定したバッファ名（a から z のいずれか）にコピーした行を保存
"バッファ名 dd        ←指定したバッファ名（a から z のいずれか）に削除した行を保存
```

例えば、バッファ c にヤンクした行を保存するには、以下のように入力します

```
"cyy
```

次に、「名前付きバッファ」に保存された内容を貼り付けるには、以下のように入力します。

```
"バッファ名 p
```

例えば、バッファ c に保存された内容を貼り付けるには、以下のように入力します。

```
"cp
```

便利なテキストエディタ

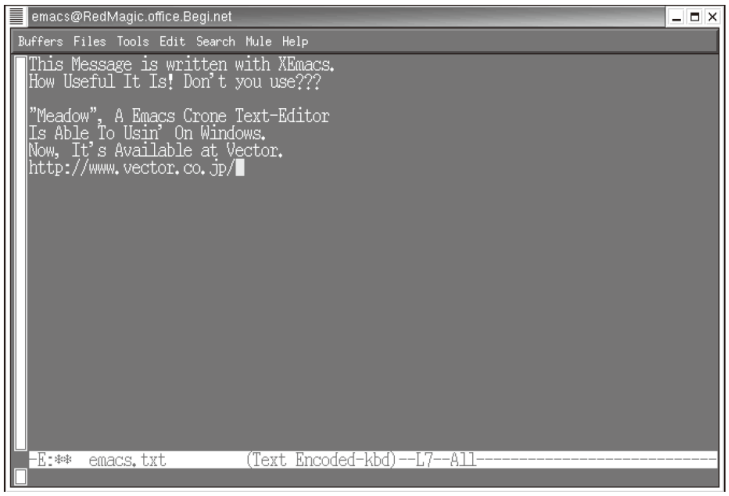
設定ファイルの編集、プログラミング、文書作成など幅広い分野で必要不可欠なのがテキストエディタです。今回は vi を用いましたが、それ以外にも様々なものが Linux では利用できるようになっています。

vi

vi は今回学んだもの以外にもたくさんのコマンドを持つ、機能的なテキストエディタです。Linux に限らず、UNIX や UNIX 互換の OS でも利用でき、システム管理の上では非常に重宝するエディタです。
多くの Linux では改良版の vim が採用されていますが、LPIC 範囲は vi となっています。

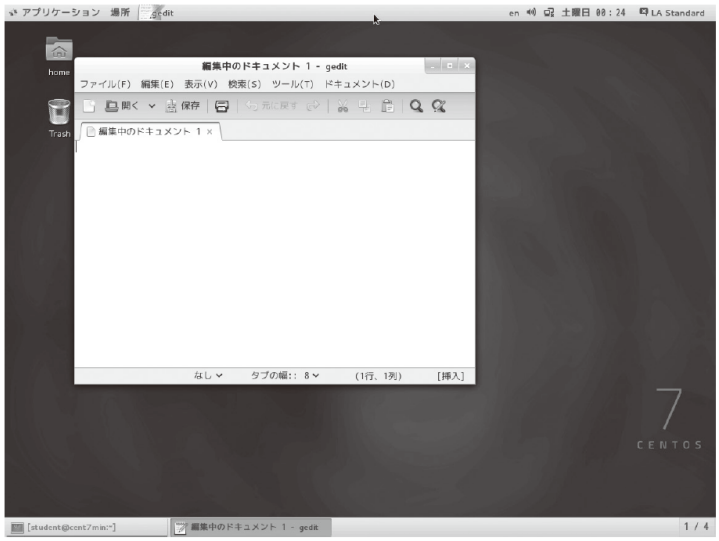
GNU Emacs

FSF の設立者であり、GCC を開発した Richard Stallman 氏が中心となって開発された高機能エディタで、自身を拡張（Emacs-Lisp 言語）できるという特徴を備えています。そのため、多くのプラグインがあり、ソフトウェア開発環境としても利用されています。



gedit

gedit は、X のデスクトップ環境 GNOME で標準装備されたエディタです。



6. Linux の基本操作(5)

6

Linux の基本操作(5)

～プロセスとジョブ～

本章のねらい

- 仮想コンソールの利用について学ぶ
- プロセス・ジョブについて学ぶ

予習編

プログラムとその実行

今までの章で学んだようなコマンドやワープロソフトやブラウザなどのアプリケーションは、Linux に限らず、あらかじめ人の手によって作られたファイル（プログラム）です。ユーザーがコマンドやアプリケーションを実行しようとすると、コンピューターはプログラムを読み込みます。そして、コンピューター内の CPU は読み込まれたプログラムに従って処理を行います。

たとえば、Windows で Web ページを閲覧する場合、ユーザーは Internet Explorer などのアイコンをダブルクリックして Web ブラウザを起動します。この時、コンピューターはハードディスクに書き込まれている Web ブラウザのプログラムを読み込みます。CPU はここで読み込まれたプログラムにしたがって「Web ページのデータを取り寄せる」「取り寄せたデータを整形し表示する」などさまざまな処理を行います。

ジョブとプロセス

Linux はマルチタスク OS であり、1 つのコンピューターで同時に複数の処理をこなすことができます。複数のプログラムを効率よく同時に実行するために「ジョブ」や「プロセス」という考え方が用いられています。

Linux のマルチタスク方式では、プログラムを実行しようとするとき、まず複数のプログラムを「ジョブ」や「プロセス」という単位に分けて実行します。そして、CPU が細かく分けられたプロセスを一定時間ごとに交互に処理することにより、複数の処理を並列にこなします。これを「タイムシェアリング方式」とよんでいます。

6.1 仮想コンソール

Linux では 1 つのマシン上で 6 つのコンソール（端末）を利用することができます。1 台のマシンには 1 組のキーボードとディスプレイがありますが、ここで 6 つのコンソールを利用するために仮想コンソールという仕組みが用意されています。これは物理的には 1 つの端末をあたかも 6 つあるかのように見せるものです。

6 つのコンソールには

```
tty1 tty2 tty3 tty4 tty5 tty6
```

という名前[*1]が付いています。このうち tty1 は X のグラフィック画面を兼任しており、切り替えるには、

```
[ALT} + [F1] ～ [F6]
```

を押します。最初の状態では tty1 の端末を利用していますが、[Alt]+[F2]キーを押すと画面が切り替わり、ログイン画面が表示されます。この様に一つのディスプレイ装置で複数のコンソールを切り替えて利用可能にしている仕組みが仮想コンソールです。X Window System が起動している場合は[Ctrl]+[Alt]+[F2]キーの組み合わせで仮想コンソール tty2 に移ることができます。X の GUI 画面に戻るためには[Alt]+[F1]キーを押します。

[練習]

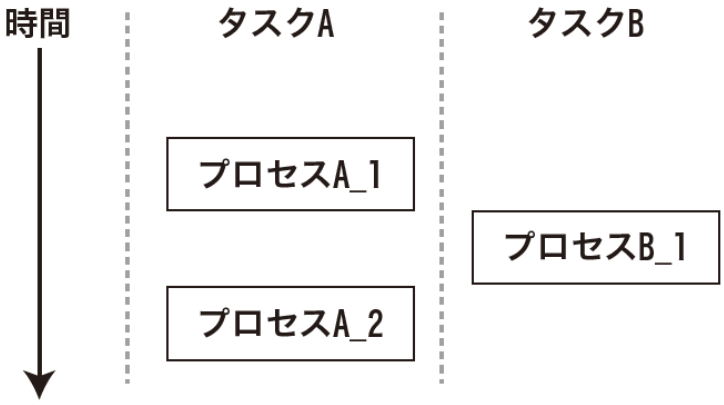
1. X の画面から、[Ctrl]+[Alt]+[F2]キーを押して、コンソール tty2 に移ります。
2. tty2 から、[Alt]+[F3] キーを押して、tty3 に移ります。
3. [Alt]+[F1]キーを押して、X に戻ります。

*1 tty は TeleTYpewriter の略で CUI 端末を表す、歴史的な用語です。

6.2 プロセス

6.2.1 マルチタスクの仕組み

Linux はマルチタスクな OS です。複数のアプリケーションを同時に実行することができるのはマルチタスク OS の大きな特徴です。1 つの処理（タスク）を処理の単位であるプロセスに分けて、一定時間ごとに処理するプロセスを切り替えることで、あたかも複数の処理を同時に行っているよう見せかけています。



このように、処理するプロセスを一定時間ごとに切り替える方式を「タイムシェアリング方式 TTS : Time Sharing System」と呼びます。プロセスにはそれぞれプロセス ID（PID）という一意の数字が割り振られており、区別できるようになっています。

6.2.2 プロセスの管理

個々のプロセスの実行は OS のカーネル（核）により管理されており、ツリー状の階層形式がとられています。1 つのプロセスを頂点にフォーク（fork）と呼ばれる手続きを経て、別のプロセス（子プロセス）を起動し、そのプロセスがまた別の子プロセスを起動し…、といった具合にした階層構造になっているのです。pstree コマンドを利用すると、個々のプロセスの関係を見ることができます。

例

```
$ pstree
systemd-+-ModemManager---2*[{ModemManager}]
          |-NetworkManager---2*[{NetworkManager}]
          |-2*[abrt-watch-log]
        |-abrtd
        |-accounts-daemon---2*[{accounts-daemon}]
        |-alsactl
        |-atd
        |-auditd-+-audispd-+-sedispatch
        |        |    `-{audispd}
        |        `-{auditd}
        |-avahi-daemon---avahi-daemon
        |-chronyd
        |-colord---2*[{colord}]
        |-crond
        |-cupsd
        |-login---bash---pstree
                 ...
        |-upowerd---2*[{upowerd}]
        `-wpa_supplicant
```

これを見ると、systemd が全ての頂点となるプロセスであり、その下に多数のプロセスが並んでいることが分かります。

```
systemd-+-login---bash---pstree
```

この行は1つのプロセスがフォークし、子プロセスを生成している事を端的に表しています。

systemd プロセスは、OS が最初に生成するプロセスで、全てのプロセスの創始となりあらゆるサービスを起動する、非常に重要なプロセスです。

[練習]
pstree コマンドを実行します。

6.2.3 プロセス管理に関するコマンド

実行中のプロセスを一覧表示するのが ps（process status）コマンドです。

例

```
$ ps
PID     TTY               TIME   CMD
11295   tty1          00:00:00   bash
11545   tty1          00:00:00   ps
```

オプションなしで ps コマンドを実行すると、ユーザー自身の権限で動作するプロセス（例えばアプリケーションやコマンドなど）の一覧が表示されます。

ps コマンドの主な表示項目

USER	実行したユーザー名
PID	プロセス ID
TTY	プロセスを制御している端末名
TIME	実行時間（CPU を使用した時間）
COMMAND(CMD)	実行コマンド（プロセス名）
%CPU	CPU 利用率
%MEM	メモリーの使用率
VSZ	使用仮想メモリーのサイズ
RSS	常駐セットの大きさ
STAT	状態（S:スリープ、R:実行中、T:一時停止）
START	プロセスの開始時刻

ps コマンドの主なオプション

オプション	説明
a	他のユーザーのプロセスも表示
u	人間に読みやすい形での表示
x	制御端末のないプロセスも表示

a, u, x オプションを付けた結果は次のようになります。

例

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.3  0.2 125276  3820 ?        Ss   16:36   0:00 /usr/lib/systemd/s
root         2  0.0  0.0      0     0 ?        S    16:36   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    16:36   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    16:36   0:00 [kworker/0:0]
root         5  0.0  0.0      0     0 ?        S<   16:36   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        S    16:36   0:00 [kworker/u2:0]
root         7  0.0  0.0      0     0 ?        S    16:36   0:00 [migration/0]
root         8  0.0  0.0      0     0 ?        S    16:36   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        R    16:36   0:00 [rcu_sched]
root        10  0.0  0.0      0     0 ?        S    16:36   0:00 [watchdog/0]
root        12  0.0  0.0      0     0 ?        S    16:36   0:00 [kdevtmpfs]
root        13  0.0  0.0      0     0 ?        S<   16:36   0:00 [netns]
root        14  0.0  0.0      0     0 ?        S    16:36   0:00 [khungtaskd]
root        15  0.0  0.0      0     0 ?        S<   16:36   0:00 [writeback]
root        16  0.0  0.0      0     0 ?        S<   16:36   0:00 [kintegrityd]
root        17  0.0  0.0      0     0 ?        S<   16:36   0:00 [bioset]
root        18  0.0  0.0      0     0 ?        S<   16:36   0:00 [kblockd]
root        19  0.0  0.0      0     0 ?        S<   16:36   0:00 [md]
root        20  0.0  0.0      0     0 ?        S    16:36   0:00 [kworker/0:1]
root        25  0.0  0.0      0     0 ?        S    16:36   0:00 [kswapd0]
root        26  0.0  0.0      0     0 ?        SN   16:36   0:00 [ksmd]
root        27  0.0  0.0      0     0 ?        SN   16:36   0:00 [khugepaged]
root        28  0.0  0.0      0     0 ?        S    16:36   0:00 [fsnotify_mark]
root        29  0.0  0.0      0     0 ?        S<   16:36   0:00 [crypto]
（省略）
```

[練習]

1. ユーザー student で 2 つ目の仮想コンソール（tty2）にログインします。
2. ps aux コマンドを使い端末 tty2 でユーザー student により、bash が起動されていることを確認します。
3. 次のコマンド[*1]を実行し、プロセスの依存関係を確かめます。

```
$  pstree -p
```

*1 「pstree -p」は、PID 付きでプロセスのツリー構造を表示するためのオプションです。

kill コマンド

動いているプロセスを止めることができるのが kill コマンドです。

```
kill <オプション> <プロセス ID>
```

kill コマンドを実行することができるのは、自分が実行しているプロセスのみです。他人のプロセスを止めることはできません。他人のプロセスを kill しようとしたときには、以下のようなエラーメッセージが現れます[*2]。

例

```
$ kill 2814
-bash: kill: (2814) - 許可されていない操作です
```

但し、root はプロセスの所有者が誰であろうとすべてのプロセスを kill できます。また、kill コマンドはただプロセスを止めるためだけのものではありません。このコマンドは元々、シグナルと呼ばれるメッセージをシステムに伝えるためのコマンドで、停止する以外にも様々なシグナルが用意されています。

-l オプションを付けて実行すると、定義されているシグナルの一覧が表示されます。この中で重要なのは、

値	シグナル名	意味
1	SIGHUP	一度、プロセスを停止し、再度起動する
9	SIGKILL	プロセスを強制的に停止する
15	SIGTERM	プロセスを停止する（デフォルトの操作）
18	SIGCONT	一旦停止中（SIGSTOP）のプロセスの再開
19	SIGSTOP	プロセスの一旦停止

です[*3]。

シグナルを指定しない時は、デフォルトで SIGTERM が適用されるため、

```
kill <PID>
kill -15 <PID>
kill -TERM <PID>
kill -SIGTERM <PID>
```

は全て同じ動作をします。kill コマンドの動作の指定にはシグナルの番号と名前のいずれを用いても構いません。

*2 動いてるプロセスの PID や所有ユーザは ps aux コマンドで参照できます。
*3 シグナル名の SIG は省略すことができます。

[練習]

1. ユーザー student で tty3 にログインします。
2. tty3 上で、次のコマンドを実行します。

```
$ less /etc/bashrc
```

3. tty4 に移り、ユーザー student でログインします。
4. tty4 上で次のコマンドを実行し、2. で実行したプロセスの PID を調べます。

```
$ ps aux
```

5. tty4 上で、4. で調べたプロセスを停止させます。

```
$ kill <調べた PID>
```

6. tty3 に戻って、プロセスが停止されていることを確認します。

6.3 ジョブ

6.3.1 ジョブ

プロセスは処理の単位でした。これに対し、端末ごとに見た一連の処理をジョブと呼びます。ジョブは 1 つもしくは複数のプロセスから成り立ちます。ジョブにはジョブ番号が付けられています。ジョブ番号は処理に対して端末ごとに付けられる番号です。プロセス ID がシステムの中で一意的に振られているのとは異なり、端末が違えば、同じジョブ番号のジョブが存在することもあります。

6.3.2 フォアグラウンドジョブとバックグラウンドジョブ

Linux では 1 つの端末で複数のコマンドを同時に実行することができます。1 組のキーボードとディスプレイしかない状態で複数のジョブを実行するためにフォアグラウンドジョブとバックグラウンドジョブという概念があります。

```
フォアグラウンドジョブ ： 現在ユーザーが直接操作できるジョブ
バックグラウンドジョブ ： ユーザーには直接見えない、裏で動いているジョブ
```

ジョブはフォアグラウンドからバックグラウンドに、バックグラウンドからフォアグラウンドに移ることができるので、これを用いて一つの端末上で複数のジョブを同時に実行することができます。

バックグラウンドでジョブを実行するには普通のコマンド入力のあとに&（アンパサンド）を付けて実行します。

```
<コマンド> &
```

[練習]

1. X Window System でターミナルを開きます。
2. 次のコマンドの持つ意味を考えて、実行します。

```
$ gedit &
[<ジョブ番号>] <プロセス ID> ジョブ番号と PID が表示される


$ ps aux
$ jobs
```

6.3.3 jobs コマンド

起動しているジョブの一覧を表示するには、jobs コマンドを用います。

```
jobs [-l]（-l オプションで PID を表示させることができます）
```

結果は

```
[<ジョブ番号>](+/-) <ジョブの状態> <ジョブの名前>
```

という形式で表示されます。ジョブ番号のあとに続くマークは

マーク	意味
+	直前に実行したジョブ
-	直前の1つ前に実行したジョブ
無印	それ以前に実行したジョブ

という意味を持ちます。

6.3.4 ジョブの操作

フォアグラウンドのジョブを一時停止にする

[Ctrl]+[Z]キーを押すと、現在のフォアグラウンドジョブを一時停止して、バックグラウンドジョブに切り替えることができます。

例

```
$ gedit
   [Ctrl]+[Z] キーを押す
$ jobs
```

停止中のバックグラウンドジョブをバックグラウンドのままで実行

bg コマンドを用いれば、バックグラウンドのままで停止中のジョブを再開することができます。

```
bg              ← 直前のジョブをバックグラウンドで実行
または
bg  %<ジョブ番号>
```

バックグラウンドジョブをフォアグラウンドにする

fg コマンドを使用すれば、バックグラウンドで動作もしくは停止しているジョブをフォアグラウンドで実行することができます。

```
$ fg                        ← 直前のジョブを実行
または
$ fg %<ジョブ番号>          ← ジョブ番号を指定して実行
または
$ fg +(-)                   ← + または - のついたジョブを実行
```

実行中のジョブを停止する

kill コマンドを用いれば、ジョブを停止することができます。プロセスの停止と同様に

```
kill [オプション] %<ジョブ番号>
```

を用いることで、ジョブは停止します。プロセスの停止と同様、他人のジョブを終了させることはできません（root ではない場合）。

[練習]

1. バックグラウンドで実行されているジョブを jobs コマンドで確認します。
2. バックグラウンドで実行しているジョブがあれば、kill コマンドを用いて、全て停止させます。

6.4 確認問題

1. top コマンド[*4]を実行します。

```
$ top
```

2. [Ctrl]+[Z]キーを押します。top コマンドによるプロセス情報の更新は停止します。top コマンドの結果表示が画面に残ってしまうため clear コマンドを使って、画面表示をクリアします。

3. jobs コマンドを実行します。

```
$ jobs
```

4. ps コマンドを実行します。

```
$ ps aux
```

5. top コマンドは今、バックグラウンドで停止しています。これを fg コマンドでフォアグラウンドに戻します。

```
$ fg %1
```

6. 再び、[Ctrl]+[Z]キーでバックグラウンドに戻します。

7. top のプロセスを正常終了させます。

```
$ kill %1
```

※ 正常終了できなかった場合、プロセスを強制終了させます。

```
$ kill -9 %1
```

*4 top コマンドは CPU への負荷が多い順にプロセスを表示するコマンドです。ps コマンドと違って、明示的に終了するまで情報を更新して表示し続けます。

7. Linux の基本操作(6)

7

Linux の基本操作(6)

～リダイレクトとパイプライン～

本章のねらい

- リダイレクション・パイプについて学ぶ
- 圧縮と展開について学ぶ

予習編

リダイレクション

多くのコマンドでは、キーボードからデータを入力し、ディスプレイに結果を出力します。しかし、コマンドの実行結果を何度も参照したい時はデータの出力先をファイルにした方が便利です。これを実現するのがリダイレクションです。たとえば ls コマンドの結果は標準ではディスプレイに表示されますが、これをファイルに書き出すよう変更することができます。ls コマンドの出力先をファイル ls.out にするには記号「>」を使って次のようにします。

```
$ ls > ls.out
```

こうすると、ls コマンドの出力結果がファイル ls.out に書き込まれます。

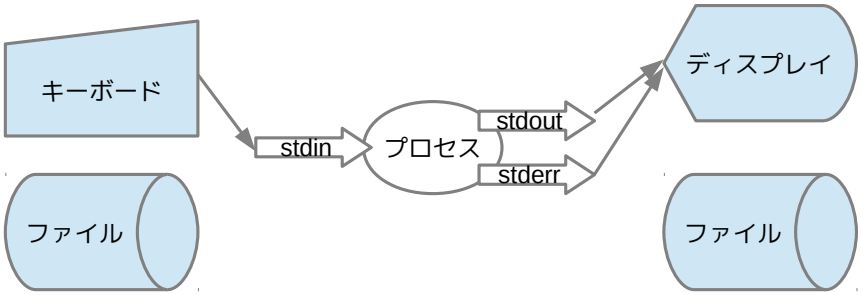
7.1 リダイレクション

7.1.1 標準入出力

今までいくつかのコマンドを学んできましたが、キーボードから入力し、結果が画面に表示されるというタイプのものが一般的です。このデータの入出力先を、Linux では予め3つ用意しています。

```
[stdin]  標準入力
[stdout] 標準出力
[stderr] 標準エラー出力、または標準診断出力
```

データの入力方法は、キーボードやファイルからの読み込み、フロッピーディスクや CD-ROM からの読み込みなど様々です。しかし、これらの装置を区別なく汎用的な入力元として定義されたのが標準入力です。標準出力、標準エラー出力も同様です。



デフォルトでは、この標準入出力に下記の装置が割り当てられています。

```
標準入力（0）        ：キーボードからの入力
標準出力（1）        ：ディスプレイへの出力
標準エラー出力（2）  ：ディスプレイへの出力
```

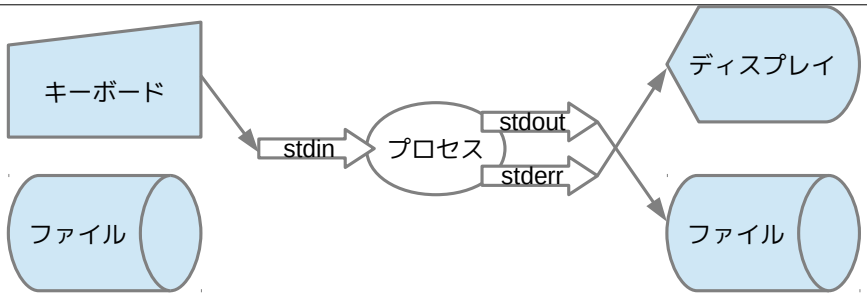
この入力元、出力先は必要に応じて変更することができます。例えば、ls コマンドの結果をディスプレイでなく、ファイルに切り替えることができます。これをリダイレクション[*1]とよ呼びます。

*1 リダイレクト(redirect) には元々、方向を変えるという意味があります。

7.1.2 出力リダイレクション

標準出力をディスプレイではなくファイルへ切り替えることを出力リダイレクションと呼び、コマンドラインから不等号を使って以下のように指定します。

```
[コマンド] > [ファイル名]
```



この時、出力先のファイルが存在すると上書きされ、元の内容は失われます。

例

```
$ ls
CF-3.7Wpl2 CF-3.7Wpl2.tar.gz gcc-3.0.2.tar.gz mbox

$ ls > list.txt
$ cat list.txt
CF-3.7Wpl2
CF-3.7Wpl2.tar.gz
gcc-3.0.2.tar.gz
list.txt
mbox
```

[練習]

次のコマンドの持つ意味を考えて、実行します。

```
$ echo "Hello" > hello.txt
$ cat hello.txt
```

また、大なり記号(>) を 2 つ重ねて使うと、追加書き込みになります。

```
[コマンド] >> [ファイル名]
```

例

```
$ date > date.out
$ date >> date.out
$ cat date.out
2018年  8月  6日 月曜日 13:20:04 JST
2018年  8月  6日 月曜日 13:20:09 JST
```

出力リダイレクションによる上書きを制限するにはシェルのオプション noclobber を有効にします。シェルオプションは set コマンドを用います。-o でシェルオプションを有効にし、+o で無効にすることができます。

例

```
$ set -o noclobber
$ date > date.out
-bash: date.out: cannot overwrite existing file
$ date >> date.out
$ set +o noclobber
$ date > date.out
```

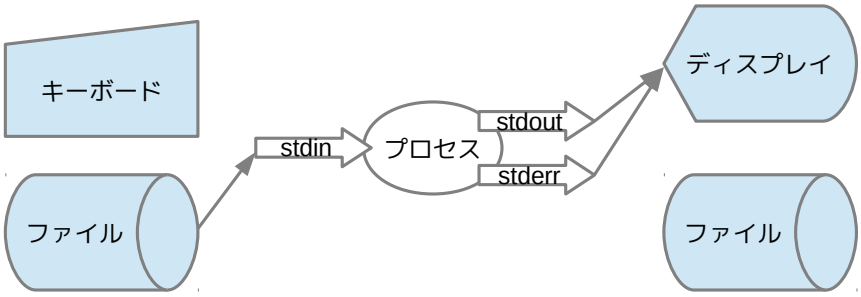
また、引数なしの-o では、現在の設定内容が表示されます。

```
[student@h006 ~]$ set -o
allexport       off
braceexpand     on
emacs           on
errexit         off
errtrace        off
functrace       off
hashall         on
. . .
```

7.1.3 入力リダイレクション

キーボードからデータを入力する代わりに、ファイルからまとめてコマンドへデータを投入することを入力リダイレクションと呼びます。コマンドラインからは不等号を使って以下のように指定します。

```
コマンド < ファイル名
```



入力リダイレクションのバリエーションとして、ヒアドキュメントがあります。キーボード入力とファイル入力の中間にあたる考え方で、入力する情報をスクリプト上に展開して表記します。

```
コマンド << 終了キーワード
データ
  .
  .
終了キーワード
```

終了キーワードは任意の文字列を指定できます。定形データを扱う場合に用いられます。

例

```
$ cat -n << EOD
> `date`
> Hello, $USER
> EOD
      1 Mon Mar 30 12:11:15 JST 2009
      2 Hello, student
$
```

キーボードからヒアドキュメントを起動した場合は、終了キーワードが入力されるまで継続プロンプト（'>'、コマンド入力が完了していない事を意味するプロンプト）が表示されつづけます。

入力リダイレクションは全てのデータをそのままプロセスに引き渡しますが、ヒアドキュメントは、一旦シェルの評価（特殊文字に関する処理）を行います。

7.2 パイプライン

7.2.1 標準入出力の連結

コマンドとコマンドを縦棒（|）で接続したものをパイプラインと呼び、前段の標準出力と後段の標準入力が接続された状態となります。

```
[コマンド] | [コマンド]
```

例

```
$ ps aux | grep systemd
root         1  0.0  0.2 125276  3824 ?        Ss   16:36   0:00 /usr/lib/systemd/sys
temd --switched-root --system --deserialize 21
root       357  0.0  0.1  36920  2568 ?        Ss   16:36   0:00 /usr/lib/systemd/sys
temd-journald
root       382  0.0  0.1  44360  2412 ?        Ss   16:36   0:00 /usr/lib/systemd/sys
temd-udevd
dbus       527  0.0  0.1  26764  2096 ?        Ss   16:36   0:00 /bin/dbus-daemon
--system –address=systemd: --nofork --nopidfile --systemd-activation
root       538  0.0  0.0  24188  1740 ?        Ss   16:36   0:00 /usr/lib/systemd/sys
temd-logind
student   2520  0.0  0.0 112648   964 tty1     S+   16:53   0:00 grep --color=auto
systemd
```

grep コマンドは与えられたファイルの中から、引数のパターンを持つ行を取り出すコマンドです。

[練習]

次のコマンドの持つ意味を考えて、実行します。

```
$ ps aux | grep bash
```

7.2.2 リスト

パイプ以外にも複数のコマンドを連結できます、リストには以下の種類があります。

コマンド１ ； コマンド２	コマンド1が終わり次第コマンド2を実行
コマンド１ ＆ コマンド２	コマンド1をバックグランド実行し、コマンド2を同時実行
コマンド1 ＆＆ コマンド2	コマンド1が正常の場合に限りコマンド2を実行
コマンド1 || コマンド2	コマンド１がエラー終了だった場合にコマンド2を実行

各コマンドの処理結果はシェル変数$?に格納され、0が正常、それ以外はエラーというルールになっています。

［練習］

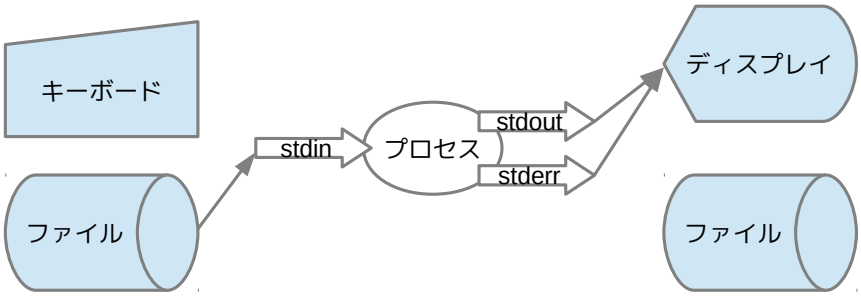
1.「ls ～」の結果が正常の場合に「echo “OK”」を実行する。

2.「ls /uso800 」がエラーの場合に「echo “NG”」を実行する。

7.2.3 標準入出力処理に関連するコマンド

リダイレクトを使うと、出力は全てファイルに向けられます。ディスプレイとファイルに対して同時に出力するためには tee コマンドが用いられます。

```
[コマンド] | tee [ファイル名]
```



[練習]

次のコマンドの持つ意味を考えて、実行します。

```
$ ps aux | grep syslogd | tee syslogd.txt
$ cat syslogd.txt
$ pstree |tee pstree.log
$ cat pstree.log
```

7.3 確認問題

1. ユーザー student でログインします。
2. top コマンドの結果をパイプにより、tee コマンドに渡します。

```
$ top | tee top.log
```

3. [Ctrl]+[C]で top を終了します。
4. 作成された top コマンドの結果のログを参照します。

```
$ cat top.log
```

column

エラーリダイレクション

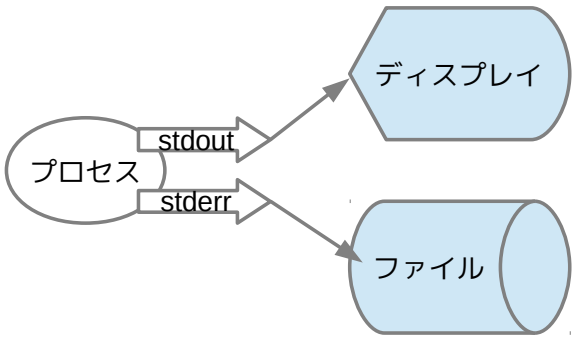
普段、コマンドの出力はディスプレイに表示されていますが、この出力には標準出力と標準エラー出力が混在している場合があります。この場合、出力リダイレクションを用いるとエラー出力の取りこぼしが発生してしまいます。

例

```
$ ls
date.out
$ ls -l date.out uso800
ls: cannot access uso800: No such file or dirctory
-rw-rw-r-- 1 student student 86 3 月 30 01:03 date.out
$ ls -l date.out uso800 > ls.out
ls: cannot access uso800: No such file or dirctory
$
```

標準エラー出力をリダイレクションするときは、不等号の直前にファイルディスクリプタである「2」を付加し、以下のように行います。

```
コマンド 2> ファイル
```

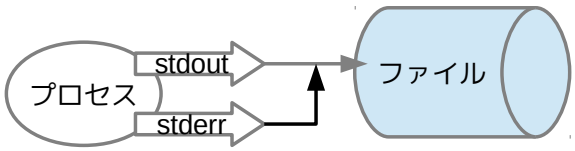


以下は、先の例を再度実行し、標準出力を ls.out、エラー出力を ls.err にリダイレクトした例です。ディスプレイに出力されていた両出力ともファイルに格納できましたが、別々のファイルになっています。(同じファイルへ出力した場合は、その内容が破損してしまいます)

```
$ ls -l date.out uso800 2> ls.err
-rw-rw-r-- 1 student student 43 3 月 30 13:58 date.out
$ ls -l date.out uso800 > ls.out 2> ls.err
```

2 つの出力を 1 つに片寄せするには、以下のようにファイルディスクリプタ番号と& を使って指示します。

```
コマンド  >  ファイル    2>&1
```



この例では、まず標準出力（1）をファイルへリダイレクションし、エラー出力（2）を標準出力へ連結しています。

圧縮・解凍・アーカイブ作成と展開

圧縮（gzip）

ディスク領域の節約やインターネットからのダウンロードの便利性を求めて、ファイルを圧縮することが必要になる場合があります。Linux で多く用いられているのは gzip（GNU zip）形式の圧縮です。gzip 形式の圧縮は gzip コマンドを用いて行われます。

```
gzip [圧縮するファイル1] [圧縮するファイル2] ...
```

gzip コマンドを実行すると、指定したファイルが圧縮され、元のファイル名に.gz という拡張子が付けられたファイルが作成されます。同時に複数のファイルを指定して、それぞれ圧縮ファイルにすることができます。

例

```
[student@h006 ~]$ cd
[student@h006 ~]$ cp /bin/bash .
[student@h006 ~]$ gzip bash
[student@h006 ~]$ ls -l /bin/bash ./bash.gz
-rwxr-xr-x  1 student student 470292  8月 15 12:55 ./bash.gz
-rwxr-xr-x. 1 root    root    964544  4月 11 09:53 /bin/bash
```

gzip の主なオプション

オプション	説明
-c	圧縮した結果をファイルに書き込まずに、標準出力に出力する
-v	処理情報を詳細に表示する
-d	圧縮ファイルを解凍する
-l	圧縮された個々のファイルについて、圧縮前・圧縮後のファイルサイズ、圧縮率、元のファイル名を表示する

gzip コマンドを使用すると、元のファイルは残されません。元のファイルを残して、ファイルを圧縮するには-c オプションの標準出力を別ファイルにリダイレクトします。

```
gzip -c [圧縮するファイル] > [圧縮したファイルの名前]
```

例

```
$ cd
$ gzip -c .bashrc > .bashrc.gz
$ ls -l .bashrc*
-rw-r--r-- 1 student student 256  8月  7 17:10 .bashrc
-rw-rw-r-- 1 student student 225  8月 15 12:57 .bashrc.gz
```

解凍（gunzip）

gzip 形式で圧縮したファイルは gunzip コマンド、もしくは gzip -d コマンドを実行することにより、解凍することができます。

```
gunzip [解凍するファイル(.gz)…]
gzip -d [解凍するファイル.gz…]
```

gunzip コマンドは gzip コマンドと同様に複数のファイルを指定することができます。

gunzip の主なオプション

オプション	説明
-c	解凍した結果をファイルに書き込まずに、標準出力に出力する
-v	処理情報を詳細に表示する

圧縮ファイルの表示

zcat コマンドや less コマンドを用いれば、圧縮ファイルの内容を直接表示することができます。

```
zcat [圧縮ファイル…]
less [圧縮ファイル…]
```

これは、

```
gunzip -c [解凍するファイル(.gz)…]
```

としたときと、同じ効果があります。

アーカイブの作成と展開（tar）

複数のファイルを一つにまとめたファイルにしたものをアーカイブ（archive、書庫）と呼びます。Linux ではアーカイブ作成に tar コマンドを使用します。

```
tar [オプション][ファイル…, ディレクトリ…]
```

tar コマンドの主なオプション

オプション	説明
c	アーカイブの作成
x	アーカイブファイルの展開
t	アーカイブファイルの内容を ls コマンド形式で一覧表示
r	既存のアーカイブファイルの最後に新たにファイルを追加
v	処理情報を詳細に表示する
f	作成するアーカイブファイルの名前を指定する
z	gzip 形式（LZ77 法）の圧縮・解凍を同時に行う
j	bzip2 形式（Burrows-Wheeler 法）の圧縮・解凍を同時に行う
J	xz 形式（LZMA 法）の圧縮・解凍を同時に行う

アーカイブファイルの作成

アーカイブファイルの作成には通常、c, v, f オプションを使用します。

```
tar cvf [アーカイブ名].tar [ファイル…, ディレクトリ…]
```

例

```
$ mkdir ~/archives
$ cd archives/
$ tar cvf bashfiles.tar ~/.bash*
tar: メンバ名から先頭の `/' を取り除きます
/home/student/.bash_history
/home/student/.bash_logout
/home/student/.bash_profile
/home/student/.bashrc
/home/student/.bashrc.gz
[student@h006 archives]$ ls -l
合計 20
-rw-rw-r-- 1 student student 20480  8月 15 13:01 bashfiles.tar
```

アーカイブファイルの展開

アーカイブファイルの展開には通常、x, v, f オプションを使用します。

```
tar xvf [アーカイブ名].tar
```

例

```
$ pwd
/home/student/archives
$ tar xvf bashfiles.tar
home/student/.bash_history
home/student/.bash_logout
home/student/.bash_profile
home/student/.bashrc
home/student/.bashrc.gz
$ ls -aR
.:
.  ..  bashfiles.tar  home

./home:
.  ..  student

./home/student:
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .bashrc.gz
```

tarball（ターボール）の作成・展開

アーカイブ作成後に gzip で圧縮してできた.tar.gz もしくは、.tgz という拡張子が付けられたファイルのことを tarball と呼びます。tarball は、アプリケーションを配布する際によく用いられる形式です。

tarball の作成

```
tar czvf [tarball 名].tar.gz [ファイル・ディレクトリ…]
tar czvf [tarball 名].tgz [ファイル・ディレクトリ…]
```

tarball の解凍

```
tar xzvf [解凍する tarball 名].tar.gz
tar xzvf [解凍する tarball 名].tgz
```

[練習]

次のコマンドの持つ意味を考えながら、実行します。

```
$ cd archives/
$ tar czvf www.tar.gz /var/www
tar: メンバ名から先頭の `/' を取り除きます
/var/www/
（省略）

$ ls -l ~/archives/
```

```
合計 28
-rw-rw-r--  1 student student 20480  8月 15 13:01 bashfiles.tar
drwxrwxr-x  3 student student    21  8月 15 13:02 home
-rw-rw-r--  1 student student   668  8月 15 13:05 www.tar.gz
$ tar xzvf www.tar.gz
var/www/
var/www/cgi-bin/
var/www/cgi-bin/sample.cgi
var/www/cgi-bin/count.cgi
var/www/cgi-bin/count.dat
var/www/html/
var/www/html/staff/
var/www/html/staff/index.html
var/www/html/staff/.htpasswd
```

BZIP2 に関する補足

最近では gzip(LZ 法による圧縮) だけでなく、bzip2(BW 法による圧縮) による圧縮も普及し始めています。使い方は gzip / gunzip と同様で、それぞれ bzip2 / bunzip2 コマンドを用います。拡張子は.bz2 となり、tar のオプションは z に代えて j を用います。

```
$ cp /bin/bash data0 ; cp data0 data1; cp data0 data2
$ gzip -v data1
$ bzip2 -v data2
$ ls -l data*
-rwxr-xr-x 1 student student 964544  8月  6 13:29 data0
-rwxr-xr-x 1 student student 470293  8月  6 13:29 data1.gz
-rwxr-xr-x 1 student student 445770  8月  6 13:30 data2.bz2
```

全てはファイル

1 つ目の端末（tty1）を使っているときに、

```
$ echo "Hello" > /dev/tty1
```

としてみましょう。

コンソールの画面に「Hello」と表示されたはずです。/dev ディレクトリ以下にあるこの tty1 というファイルは第 3 章で紹介した特殊デバイスファイルと呼ばれる種類のファイルです。上の例ではこのデバイスファイルに対して、” Hello” という文字列をリダイレクトしています。つまり、Linux ではディスプレイという物理的な装置をあたかもファイルと同じように扱っているわけです。/dev には同じような特殊デバイスファイルがたくさん並んでいます。

> ※デバイスをファイルと同じように扱っているとはいえ、それはシステムの側から見てということであって、ユーザーが直接デバイスファイルを操作するということはほとんどしません。闇雲にファイルを開いたりしないようにしてください。

また、パイプではコマンドの結果を別のコマンドの引数として渡していました。例えば、

```
$ cat /etc/httpd/conf/httpd.conf | wc
```

という例では、ファイル内容を cat コマンドで画面表示する代わりに wc コマンドの引数として渡しています。しかし、wc コマンドは

```
wc [オプション] [ファイル]
```

という風に引数にはファイルを渡さないといけないはずです。なぜこのようなことができるのでしょうか？実は Linux ではコマンドの結果も 1 つの「ファイル」として扱うのです。

直感的なイメージとしては、「ファイル」というものは名前が付けられていて、ハードディスクやフロッピーディスクに保存されているものとなるでしょう。しかし、Linux の世界でいうファイルはそうとは限らず、1 つのデータのかたまりのことを指します。コマンドの結果、例えば、ls コマンドによるファイルのリストなどは 1 つのデータの集合、つまりファイルということになります。普通のファイルと区別するために「中間ファイル」と呼ぶ場合もあります。

このように極めて抽象的にファイルを扱っているので、パイプやリダイレクションなどといった機能が利用できるのです。

名前付きパイプ

パイプラインではコマンド１とコマンド２が同時に実行され、先行するコマンド１が終了すると同時にコマンド２も終了するという同期処理になっています。

```
コマンド１ | コマンド２
```

パイプライン処理をコマンド１とコマンド２を別々のタイミングで実行するには、名前付きパイプを作成し、それをコマンド１とコマンド２がそれぞれ個別に読み書きすれば可能です。

```
$ ls -l | cat -n
     1total 12
     2-rw-rw-r-- 1 student student 58 Jul 27 16:50 date.out
     3-rw-rw-r-- 1 student student 52 Jul 27 17:01 ls.err
     4-rw-rw-r-- 1 student student 54 Jul 27 17:01 ls.out
$ mkfifo fifo
$ ls -l > fifo &
[1] 2701
$ date
Thu Jul 27 17:02:29 JST 2017
$ cat -n fifo
     1total 12
     2-rw-rw-r-- 1 student student 58 Jul 27 16:50 date.out
     3prw-rw-r-- 1 student student  0 Jul 27 17:02 fifo
     4-rw-rw-r-- 1 student student 52 Jul 27 17:01 ls.err
     5-rw-rw-r-- 1 student student 54 Jul 27 17:01 ls.out
[1]+ Done     ls –color=auto -l > fifo
$
```

8. Linux の基本操作(7)

8

Linux の基本操作(7)

～ファイルシステムと RPM～

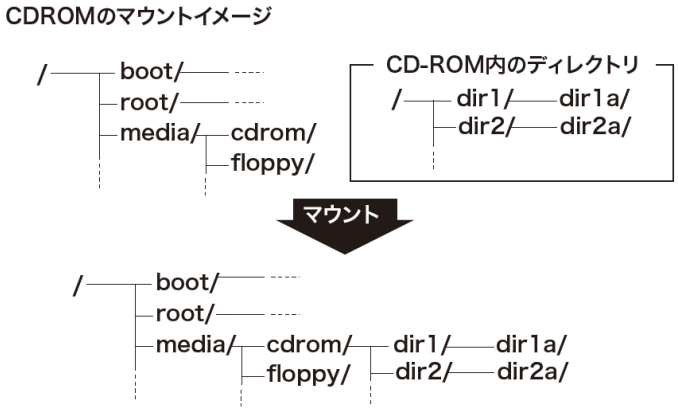
本章のねらい

- ファイルシステムのマウント
- RPM の利用

予習編

ファイルシステムのマウント

Linux は、すべてのディレクトリがルートディレクトリを頂点としたひとつの大きなツリー状の構造をとっています。そのため、CD-ROM やフロッピーなどの外部デバイスを使用する際には、このディレクトリのツリーの中に外部デバイス内のディレクトリを取り込む操作が必要になります。これが「マウント」と呼ばれる操作です。マウントは、外部デバイスなどを自分のファイルシステムの一部として「接ぎ木する」イメージで考えると理解できます。



マウントを行うには mount コマンドを使います。接ぎ木を行った先のディレクトリ(上図の/media/cdrom ディレクトリ) を「マウントポイント」と言います。

マウントを解除することを「アンマウント」といい、umount コマンドを使います。アンマウントを行わずにフロッピーや CD-ROM などを取り出すと、システムの誤動作の原因になりますので、必ずアンマウントしてから取り出すようにしましょう。

アプリケーションのインストール

Red Hat Linux 系列の Linux では、アプリケーションのインストールおよびアップデートに RPM(RedHat Package Manager) という独自のシステムを採用しています。従来は、アプリケーションのインストールを行う際は、ソースコードからコンパイルしていましたが、この方法ではアプリケーションの依存関係などを自分で管理しなくてはならず、高度で面倒なものでした。RPM ではこれらを自動的に行うため、初級者にも簡単にアプリケーションのインストールができます。また依存関係のチェックも行うためトラブルを減らすことができます。

RPM を使ったアプリケーションのインストールを行うには rpm コマンドを使います。rpm コマンドではインストールのほか、アップグレード、アンインストール、インストールされているアプリケーションの照会などができます。

8.1 ファイルシステムの概要

CD-ROM やフロッピーの外部記憶メディアに保存されたファイルを読み書きするためにはマウント (mount) という手続きが必要です。「マウントする」とは、外部のファイルシステムをあたかも自分のファイルシステムのように扱えるようにすることです。

8.1.1 ファイルシステムの形式

ファイルシステムには様々な形式があります。Linux では ext2 が標準採用されており、i-node を用いたファイルの管理が行われています。ext2 以外にも Linux は様々なタイプのファイルシステムをサポートしており、他の OS にあるファイルを読み込んだりすることもできます。

Linux でサポートされているファイルシステムの形式

名前	説明
ext2	Linux で標準的に採用されている形式
ext3	ext2 にジャーナル機能を追加した形式
ext4	ext3 に拡張性・セキュリティを追加した形式
iso9660	CD-ROM、DVD-ROM の形式（読込みのみ）
UDF	DVD の形式（読込みのみ）、iso9660 の後継
FAT	古い Windows、および USB メモリの形式
NTFS	Windows の形式
ReiserFS	小さなファイルを多数処理するのに優れた形式
XFS	SGI が開発したジャーナリング*1形式、成熟している
CIFS	Windows のネットワークファイル共有形式
NFS	UNIX のネットワークファイル共有形式

CentOS 5.x では ext3、CentOS 6.x は ext4、CentOS 7.x は XFS が標準のファイルシステムになっています。

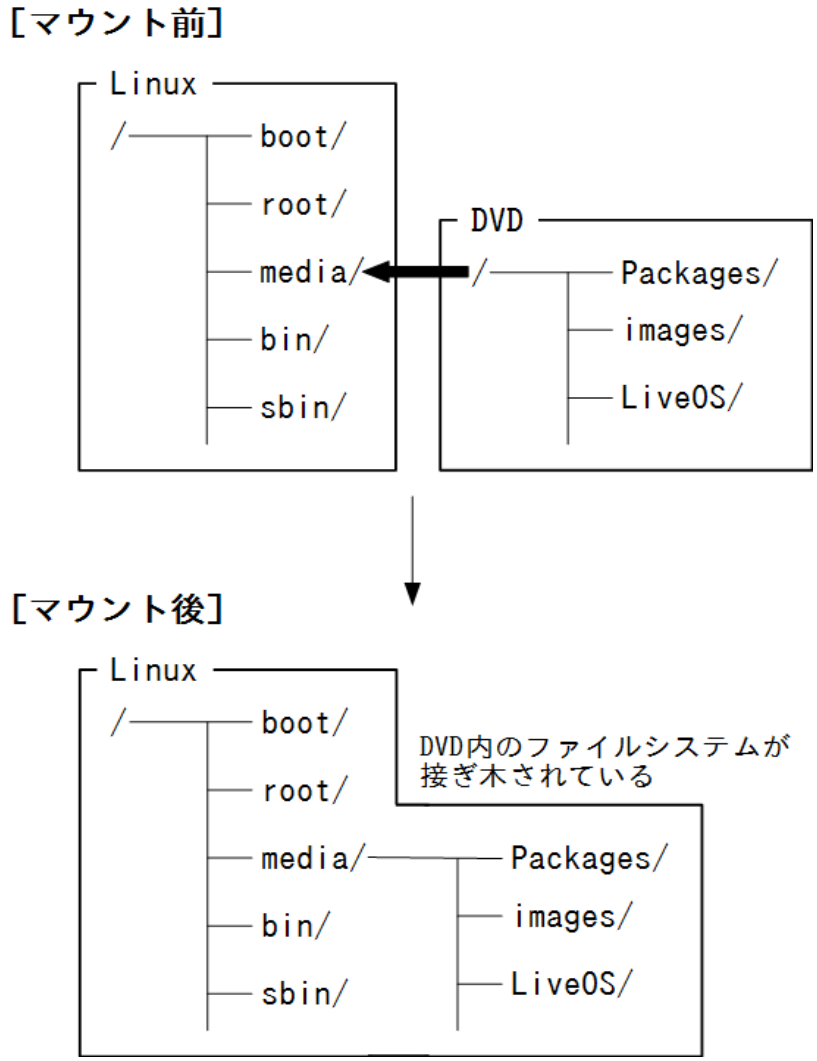
利用可能なファイルシステムは/proc/filesystems にあります。

```
$ cat /proc/filesystems
nodev    sysfs
nodev    rootfs
nodev    ramfs
nodev    bdev
nodev    proc
（省略）
```

*1 ジャーナリングはファイルの更新状態を記録しておく事で、不意な停電等によるシステムダウン後の復旧作業時間を短縮する機能です。

8.1.2 ファイルシステムのマウント

外部のファイルシステムにあるファイルを使用するには mount コマンドを利用します。「外部のファイルシステムを自分のファイルシステムの一部に接ぎ木する」と考えるとよいでしょう。



上のイメージは DVD を Linux からマウントした様子を表しています。

マウント前は外部にあったファイルシステムが、/media ディレクトリ以下に結びつけられ、あたかも/media ディレクトリ以下に Packages や images などといったディレクトリが存在しているように見えます。

/media のようなマウント先のディレクトリのことをマウントポイント（mount point）と呼びます。

8.2 ファイルシステムの操作

8.2.1 mount と umount コマンド

マウント

外部のファイルシステムをマウントするには mount コマンドを用います。先の例では DVD 用の ISO9660 という形式のファイルシステムをマウントしています。mount コマンドは以下の形式となります。

```
mount -t ファイルシステム形式  [-o オプション] デバイス名  マウントポイント
```

ファイルシステム形式は、自明であれば省略できます。オプションは必要に応じて付与します。マウントポイントは接ぎ木する場所となるディレクトリで予め作成しておきます。

実際に DVD をマウントする例を示します。

```
# mount -t iso9660 -o ro /dev/cdrom /media
```

オプションの ro は読み取り専用(read only) で、省略形の -r も利用できます。

また引数なしの mount は、現在マウントされているファイルシステムの一覧が表示されます。

```
# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,size=927556k,nr_inodes=231889,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
(中略)
/dev/sr0 on /media type iso9660 (ro,relatime)
```

この内容は /etc/mtab にも記録されています。

アンマウント

マウント済みディスクを取り外す事をアンマウントといい、umount コマンドで行います。引数はデバイス名またはマウントポイントを指定します。

```
umount デバイス名
umount マウントポイント
```

先の DVD を切り離すには次のようにします。

```
# umount /media
または
# umount /dev/cdrom
```

8.2.2 /etc/fstab ファイル

/etc/fstab へ必要な事項を記載しておけば、OS 起動時に自動でファイルシステムをマウントしてくれます。fstab は空白で区切られた6つの項目からなります。

例

```
# /etc/fstab
# Created by anaconda on Fri Jul 21 11:58:07 2017
UUID=9d0158c4-3126-4dd5-98c  /           xfs   defaults    0 0
UUID=8f8aca4d-1cbc-47fa-aac  /boot       xfs   defaults    0 0
UUID=b77bb006-2d9d-4c65-be7  /home       xfs   defaults    0 0
```

イゲタ（#、hash）記号はコメントで、空白で区切られた各項目は左側から以下の通りです。

1. デバイスファイル名またはハードディスクの固有 ID
2. マウントポイント
3. ファイルシステム形式
4. マウントオプション（既定値を用いる場合は defaults)
5. バックアップ可否（1 はバックアップ対象、0 は対象でない）
6. ファイルシステムチェック順序（0 はチェック対象外）

fstab に登録されている全ファイルシステムをマウントする場合は、mount -a を用います。

```
# mount -a
```

また、fstab に登録されているファイルシステムをマウントする際にはデバイスファイル名（あるいはハードディスクの固有 ID）かマウントポイントどちらか一方だけを指定するだけでかまいません。

```
# mount UUID=b77bb006-2d9d-4c65-be7
または
# mount /home
```

8.2.3 df コマンド

df コマンドを用いれば、現在マウントされているファイルシステムの一覧を表示することができます。

例

```
$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda5        2037760  286936   1750824  15% /
devtmpfs          927556       0    927556   0% /dev
（省略）
```

ファイルシステムとその容量、使用量とマウントポイントなどが表示されます。

-h オプションを付けて実行すると、容量や使用量が単位付きで表示されるのでわかりやすくなります。

例

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5       2.0G  281M  1.7G  15% /
devtmpfs        906M     0  906M   0% /dev
（省略）
```

-i オプションはファイル数（i-node の数）を表示します。

例

```
$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
/dev/sda5      1024000   4884 1019116    1% /
devtmpfs        231889    369  231520    1% /dev
tmpfs           235474      7  235467    1% /dev/shm
（省略）
```

[練習]

1. CentOS の DVD または CD を/media にマウントします。
2. df コマンドにより、マウントされたことを確認します。
3. /media 以下を参照しファイルが存在することを確認します。
4. /media をアンマウントします。
5. df コマンドにより、アンマウントされたことを確認します。

8.3 RPM パッケージ管理

RPM（Red hat Package Manager）は Red Hat 系 Linux ディストリビューション特有のアプリケーションパッケージ管理システムです。ディストリビューションが登場する以前は tarball 形式（必要なファイル群を1つにまとめたもの）でを配布し、ユーザーはそれを自らコンパイルして、適切なディレクトリに配置する必要がありました。これでは手間がかかり不便ですし、高度なスキルが必要でした。この不便さを解決下のが RPM です。RPM の長所は手軽なだけでなく、パッケージ間の依存関係を考慮し、インストールやアップグレードを確実にできるという点です。

RPM 形式のファイル(RPM パッケージ) は拡張子が.rpm であり、rpm コマンドによって管理されます。RPM ファイルは以下の形式で名づけられています。

```
[RPM パッケージ名]-[バージョン(数字)]-[リリース].[CPU アーキテクチャ].rpm
```

今後、パッケージ名とは上記の [RPM パッケージ名] を指します。

例

```
kernel-3.10.0-229.el7.x86_64.rpm
kernel-doc-3.10.0-229.el7.noarch.rpm
```

バージョンは数字が 3 つ小数点で区切った形式で順に、メジャーバージョン、マイナーバージョン、リビジョンと呼んでいます。リリースはディストリビュータが変更管理を行うために付けた管理記号です。
CPU アーキテクチャは CPU の種別を表していて、これが違う環境ではバイナリファイルを実行することができません。

アーキテクチャ	解説
i386	インテル社の 80386 以降、第 3 世代と呼ばれる CPU の総称。 Linux が誕生した頃の規格。
x86_64(amd64)	米 AMD 社が開発した i386 互換 64bit CPU。 Cent OS7 ではこの形式のみ対応。
arm	英 ARM 社が設計した低電力消費型 CPU。 殆どのスマートフォンやモバイルデバイスで採用されている。
ppc(powerpc)	IBM、Apple、Motorola が共同開発した CPU。 現在は主に IBM のサーバー製品に採用されている。
sparc	Sun Microsystems（現 Oracle）社が開発した CPU。 専用用途のサーバー機器に採用されている。
noarch	ソースコードやドキュメントなど CPU 非依存ファイル。

CentOS は、アーキテクチャを絞る（x86_64）方向ですが、Debian は今でも数多くのアーキテクチャをサポートしています。

8.3.1 パッケージの照会

パッケージに関する情報は rpm コマンドの「-q」オプションを使います。

主なパッケージ照会系オプオプション

コマンド	意味
rpm -qa	インストール済み全 RPM パッケージの一覧
rpm -qi [パッケージ名]	指定したパッケージに関する情報を表示
rpm -ql [パッケージ名]	指定したパッケージに関するファイルの一覧
rpm -qf [ファイル名]	指定したファイルを含むパッケージ名を表示

8.3.2 アプリケーションのインストール

RPM 形式を rpm を用いて、インストールするには「-i」オプショ n を使います。詳細情報を表示する「-v」や、進捗状況を表示する「-h」もよく用いられます。

```
rpm -ivh [RPM ファイル名]
```

例

```
# rpm -ivh xorg-x11-docs-1.6.7.el7.noarch.rpm
Preparing...            ################################# [100%]
Updating / installing...
xorg-x11-docs-1.6-7.el7   ################################# [100%]
```

さらに「-F」は指定したパッケージのみをバージョンアップ。「-U」はバージョンアップに必要な前庭パッケージも合わせてインストールします。

```
rpm -Uvh [RPM ファイル名]
rpm -Fvh [RPM ファイル名]
```

依存関係を無視するは「--nodeps」オプションを用います。また、インストール済みのパッケージを再度インストールするとエラーが出になりますが、強制的に上書きする「--force」オプションがあります。

8.3.3 アプリケーションのアンインストール

インストール済みのパッケージをアンインストールするためには-e オプションを用います。

```
rpm -e [RPM パッケージ名]
```

8.4 yum を使った更新

yum (Yellow dog Updater, Modified) はネットワークを経由して、必要なパッケージの更新を自動的に行うツールです。

8.4.1 GPG key の入手

初めて利用するには GPG(GNU Privacy Guard) の公開鍵を入手します。これは yum を初めて行う時に必要です。

```
# rpm --import http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-7
```

実際には、yum を始めた実行した時に、上記を自動的に行ってくれます。

8.4.2 リポジトリーファイルの修正

yum が最新のパッケージ情報を問い合わせるサイトを記述したファイルをリポジトリーファイルと呼びます。これは/etc/yum.repos.d/に格納され、用途によって複数のリポジトリーに分かれています。たとえば基本機能の Base や、追加機能を格納する CentOS Plus などがあり、いくつかは無効化されています。それらを有効にするには、当該ファイルを編集します。

```
# yum repolist
読み込んだプラグイン:fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.fairway.ne.jp
 * extras: mirror.fairway.ne.jp
 * updates: mirror.fairway.ne.jp
リポジトリー ID          リポジトリー名                      状態
!base/7/x86_64         CentOS-7 - Base                   9,911
!extras/7/x86_64       CentOS-7 - Extras                   368
!updates/7/x86_64      CentOS-7 - Updates                1,041
repolist: 11,320
```

```
# vi /etc/yum.repos.d/CentOS-Base.repo
```

```
#additional packages that extend functionality of existing packages
[centosplus]
name=CentOS-$releasever - Plus
mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=centosplus&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/centosplus/$basearch/
gpgcheck=1
enabled=1        ← 0から1に修正し、保存

# yum repolist
読み込んだプラグイン:fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.fairway.ne.jp
 * centosplus: mirror.fairway.ne.jp            ←追加されている
```

```
 * extras: mirror.fairway.ne.jp
 * updates: mirror.fairway.ne.jp
base                                            | 3.6 kB     00:00
centosplus                                      | 3.4 kB     00:00
extras                                          | 3.4 kB     00:00
updates                                         | 3.4 kB     00:00
centosplus/7/x86_64/primary_db                  | 2.2 MB     00:00
リポジトリー ID                  リポジトリー名                状態
base/7/x86_64                   CentOS-7 - Base              9,911
centosplus/7/x86_64             CentOS-7 - Plus                 63
extras/7/x86_64                 CentOS-7 - Extras              368
updates/7/x86_64                CentOS-7 - Updates           1,041
repolist: 11,383
```

EPEL リポジトリ

Fedora の成果物を提供する EPEL（Extra Packages for Enterprise Linux）は非常に多くのパッケージを提供していて、登録しておくと便利です。

```
[root@h006 ~]# yum -y install epel-release
読み込んだプラグイン:fastestmirror
（省略）

[root@h006 ~]# yum repolist
（省略）
epel                                                      | 3.2 kB  00:00:00
(1/3): epel/x86_64/group_gz                               |  88 kB  00:00:00
(2/3): epel/x86_64/updateinfo                             | 934 kB  00:00:00
(3/3): epel/x86_64/primary                                | 3.6 MB  00:00:00
epel                                                                12646/12646
リポジトリー ID          リポジトリー名                                    状態
base/7/x86_64            CentOS-7 - Base                                   9,911
centosplus/7/x86_64      CentOS-7 - Plus                                      63
epel/x86_64              Extra Packages for Enterprise Linux 7 - x86_64   12,646
extras/7/x86_64          CentOS-7 - Extras                                   370
updates/7/x86_64         CentOS-7 - Updates                                1,042
repolist: 24,032
```

8.4.3 yum の実行

yum を使って、更新可能なパッケージ一覧を表示するには、check-update を実行します。

```
# yum check-update
読み込んだプラグイン:fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.fairway.ne.jp
（中略）
postfix.x86_64       2:2.10.1-6.0.1.el7.centos        centosplus
python-perf.x86_64   3.10.0-862.9.1.el7.centos.plus   centosplus
（省略）
```

インストール済みパッケージ全てを更新する場合は、『yum update』を実行します。途中で全てを更新するかどうかの確認があります。そこで y を入力すると上記のリストを元に、インストールされている最新ではないパッケージが最新のものに更新されます。

```
[root@h006 ~]# yum update centos-release
読み込んだプラグイン:fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.tsukuba.wide.ad.jp
 * centosplus: ftp.tsukuba.wide.ad.jp
 * epel: ftp.tsukuba.wide.ad.jp
 * extras: ftp.tsukuba.wide.ad.jp
 * updates: ftp.tsukuba.wide.ad.jp
依存性の解決をしています
--> トランザクションの確認を実行しています。
---> パッケージ centos-release.x86_64 0:7-5.1804.1.el7.centos を 更新
---> パッケージ centos-release.x86_64 0:7-5.1804.4.el7.centos を アップデート
--> 依存性解決を終了しました。
依存性を解決しました
================================================================================
 Package            アーキテクチャー       バージョン                  リポジトリー       容量
================================================================================
更新します:
 centos-release       x86_64       7-5.1804.4.el7.centos       updates        25 k
トランザクションの要約
================================================================================
更新  1 パッケージ
総ダウンロード容量: 25 k
Is this ok [y/d/N]: y
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
centos-release-7-5.1804.4.el7.centos.x86_64.rpm             |  25 kB  00:00:00
Running transaction check
Running transaction test
```

```
Transaction test succeeded
Running transaction
  更新します              : centos-release-7-5.1804.4.el7.centos.x86_64        1/2
  整理中                  : centos-release-7-5.1804.1.el7.centos.x86_64        2/2
  検証中                  : centos-release-7-5.1804.4.el7.centos.x86_64        1/2
  検証中                  : centos-release-7-5.1804.1.el7.centos.x86_64        2/2
更新:
  centos-release.x86_64 0:7-5.1804.4.el7.centos
  完了しました!
```

主な yum のオプション

オプション	解説
check-update	更新可能なパッケージ一覧表示
update［パッケージ名］	指定したパッケージを更新します。省略すると可能な全てのパッケージを更新します。
info［パッケージ名］	パッケージの情報を表示（rpm -qi 相当）
clean all	ダウンロードしたファイルや作業データの削除
list	利用可能なパッケージの一覧
install［パッケージ名］	指定したパッケージをインストール
repolist	リポジトリとそこに格納されるパッケージ総数一覧

8.5 確認問題

1. CentOS の DVD または CD を/media にマウントします。
2. /media/Packages 以下にある

```
wireshark-1.10.14-10.el7.x86_64.rpm
wireshark-gnome-1.10.14-10.el7.x86_64.rpm
libsmi-0.4.8-13.el7.x86_64.rpm
```

   上記のパッケージをインストールします。オプションに何を指定すればよいか、考えて rpm コマンドを実行します。

3. インストールできたことを確認してください。
4. yum を使って、最新版のパッケージを確認し必要であれば更新します。

```
# yum _____________ wireshark
```

5. libsmi のみ削除を行ってみてください。

```
# rpm -e libsmi
```

   依存性チェックでエラーが発生しているので、--nodeps オプションよる強制削除を行ってください。

```
# rpm -e --nodeps libsmi
```

6. 再度、libsmi をインストールしておきます。

```
# rpm -ivh libsmi-0.4.8-13.el7.x86_64.rpm
```

column

Debian 系パッケージ管理

Debian GNU/Linux では Cent OS の rpm に相当する、dpkg コマンドによるパッケージ管理を行っています。

deb パッケージの照会

-l オプションで、インストール済みパッケージの一覧を表示します。

パッケージ照会に関するコマンド

オプション	意味
-l	インストール済みパッケージの一覧。 ワイルドカードを含むパッケージ名を指定できる。
-L パッケージ名	パッケージに含まれるファイルの一覧

```
$ dpkg -l
要望=(U)不明/(I)インストール/(R)削除/(P)完全削除/(H)保持
| 状態=(N)無/(I)インストール済/(C)設定/(U)展開/(F)設定失敗/(H)半インストール/(W)トリガ待ち/(T)トリガ保留
|/ エラー?=(空欄)無/(R)要再インストール (状態,エラーの大文字=異常)
||/ 名前            バージョン    アーキテクチ 説明
+++-==============-============-============-================================
ii  adduser         3.115        all          add and remove users and groups
ii  anacron         2.3-24       amd64        cron-like program that doesn't go
ii  apt             1.4.8        amd64        commandline package manager
. . .
$ dpkg -L adduser
/.
/etc
/etc/deluser.conf
/usr
/usr/sbin
/usr/sbin/adduser
/usr/sbin/deluser
/usr/share
/usr/share/adduser
```

deb パッケージのインストール

インストールは -i オプションにより行います。この時に --force-depends が指定されると依存関係のチェックを行いません。

```
# dpkg -i a2ps_4.14-2_amd64.deb
以前に未選択のパッケージ a2ps を選択しています。
(データベースを読み込んでいます ... 現在 30357 個のファイルとディレクトリがインストールされています。)
a2ps_4.14-2_amd64.deb を展開する準備をしています ...
a2ps (1:4.14-2) を展開しています...
dpkg: 依存関係の問題により a2ps の設定ができません:
 a2ps は以下に依存 (depends) します: psutils ...しかし:
  パッケージ psutils はまだインストールされていません。
 a2ps は以下に依存 (depends) します: libpaper1 ...しかし:
  パッケージ libpaper1 はまだインストールされていません。

dpkg: パッケージ a2ps の処理中にエラーが発生しました (--install):
 依存関係の問題 - 設定を見送ります
man-db (2.7.6.1-2) のトリガを処理しています ...
処理中にエラーが発生しました:
 a2ps

# dpkg -i --force-depends a2ps_4.14-2_amd64.deb
(データベースを読み込んでいます ... 現在 30679 個のファイルとディレクトリがインストールされています。)
a2ps_4.14-2_amd64.deb を展開する準備をしています ...
a2ps (1:4.14-2) で (1:4.14-2 に) 上書き展開しています ...
dpkg: a2ps: 依存関係の問題、しかし要求どおり設定を行います:
 a2ps は以下に依存 (depends) します: psutils ...しかし:
  パッケージ psutils はまだインストールされていません。
 a2ps は以下に依存 (depends) します: libpaper1 ...しかし:
  パッケージ libpaper1 はまだインストールされていません。

a2ps (1:4.14-2) を設定しています ...
man-db (2.7.6.1-2) のトリガを処理しています ...
```

deb パッケージのアンインストール

```
# dpkg -r a2ps
(データベースを読み込んでいます ... 現在 30679 個のファイルとディレクトリがインストールされています。)
a2ps (1:4.14-2) を削除しています ...
man-db (2.7.6.1-2) のトリガを処理しています ...
```

ネットワーク経由のインストール

また yum 同様、ネットワークを経由しパッケージファイルを取得、インストールするためのツール apt-get もあります。

パッケージのリスト取得

```
# apt-get update
無視:1 cdrom://[Debian..._Stretch_ - Official amd64...] stretch InRelease
無視:2 cdrom://[Debian..._Stretch_ - Official amd64...] stretch Release
無視:3 cdrom://[Debian..._Stretch_ - Official amd64...] stretch/contrib all Packages
. . . （省略）
取得:15 http://security.debian.org/debian-security...Packages [386 kB]
取得:16 http://security.debian.org/debian-security...Translation-en [182 kB]
820 kB を 1秒 で取得しました (455 kB/s)
パッケージリストを読み込んでいます... 完了
. . . （省略）
```

パッケージのインストール

```
# apt-get install zsh
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
以下の追加パッケージがインストールされます:
  zsh-common
提案パッケージ:
  zsh-doc
以下のパッケージが新たにインストールされます:
  zsh zsh-common
アップグレード: 0 個、新規インストール: 2 個、削除: 0 個、保留: 1 個。
4,270 kB 中 0 B のアーカイブを取得する必要があります。
この操作後に追加で 15.2 MB のディスク容量が消費されます。
続行しますか? [Y/n] n
中断しました。
```

パッケージのバージョンアップ

```
# apt-get upgrade
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
アップグレードパッケージを検出しています... 完了
以下のパッケージはアップグレードされます:
  linux-image-4.9.0-7-amd64
アップグレード: 1 個、新規インストール: 0 個、削除: 0 個、保留: 0 個。
39.1 MB のアーカイブを取得する必要があります。
この操作後に追加で 5,120 B のディスク容量が消費されます。
```

```
続行しますか? [Y/n] y
取得:1 http://security.debian.org/debian-security stretch/updates/main amd64 linux-
image-4.9.0-7-amd64 amd64 4.9.110-3+deb9u1 [39.1 MB]
39.1 MB を 6秒 で取得しました (6,025 kB/s)
changelog を読んでいます... 完了
(データベースを読み込んでいます ... 現在 30362 個のファイルとディレクトリがインストールされていま
す。)
.../linux-image-4.9.0-7-amd64_4.9.110-3+deb9u1_amd64.deb を展開する準備をしています ...
linux-image-4.9.0-7-amd64 (4.9.110-3+deb9u1) で (4.9.110-1 に) 上書き展開しています ...
linux-image-4.9.0-7-amd64 (4.9.110-3+deb9u1) を設定しています ...
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-4.9.0-7-amd64
/etc/kernel/postinst.d/zz-update-grub:
Generating grub configuration file ...
Linux イメージを見つけました: /boot/vmlinuz-4.9.0-7-amd64
Found initrd image: /boot/initrd.img-4.9.0-7-amd64
完了
```

ファイル検索

Linux ではコンピューターの全ての情報が「/」ルートを起点としファイルシステムに置かれています。設定ファイル、デバイスファイルを見失ってしまうと、本当に大変です。
では、ファイル（ディレクトリ）検索コマンド find を紹介します。

```
find [検索の基点となるディレクトリ] [検索条件] [[動作]]
```

使用できる主な検索条件と動作としては、以下のがあります。

検索条件	意味
-name［ファイル名］	指定したファイル名のファイルを探す
-user［ユーザー名］	指定した所有ユーザーのファイルを探す
-type　d / f / l	ファイル種別指定（d:ディレクトリ、f:ファイル、l:シンボリックリンク）

検索結果に対して、操作できる主なものとしては以下があります。

動作	意味
-ls	検索結果のファイル情報を詳細表示
-exec［コマンド］¥;	検索結果のファイルを引数として、コマンド実行

-name オプションを用いると、ファイル名には「*」や「?」などのワイルドカードを使用することもできます。「whi*」とすれば、「whi」で始まる全てのファイルを指し、「whi?」とすれば、「whi + 1 文字」を指します。なおワイルドカードを使う場合は、ファイル名全体をダブルクォーテーション（"）でくくります。

find コマンドの使用例

```
$ find /etc -name "*.cf"
find: 'etc/pki/CA/private': Permission denied
（中略）
find: 'etc/audit': Permission denied
/etc/postfix/main.cf
/etc/postfix/master.cf
find: 'etc/vmware-tools/GuestProxyData/trusted': Permission denied
find: 'etc/sssd': Permission denied
find: 'etc/sudoers.d': Permission denied
find: 'etc/cups/ssl': Permission denied
```

一般ユーザーで/etc ディレクトリの中身などを参照した場合はパーミッションが与えられていないため、エラーが出る場合があります。上の例では、.cf で終わるファイルを/etc ディレクトリの中から検索して参照しています。

付録
練習と確認問題の解答

第 1 章

1.3.3 練習

```
[student@h006 ~]$ exit

localhost login: root
Password: ← 入力した文字は画面に表示されません

[root@h006 ~]# logout
```

1.4 確認問題の結果例

```
1-2.
login: student
Password:               ← 入力した文字は画面に表示されません
3.
[student@h006 ~]$ who
student   tty1           2018-06-21 10:32
student   tty2           2018-06-21 11:55
4.
[student@h006 ~]$ exit
5.
login: root
Password:
6.
[root@h006 ~]# who
student   tty1           2018-06-21 10:32
root      tty2    2018-06-21 12:03
7.
[root@h006 ~]# logout
8.
login: student
Password:
9.
[student@h006 ~]$ su -
Password:
10.
[root@h006 ~]# shutdown -r now
11.
login: root
Password:
12.
[root@h006 ~]# shutdown -h now
```

第 2 章

2.1.2 練習

```
[student@h006 ~]$ pwd
/home/student
[student@h006 ~]$ ls
mail        テンプレート  ドキュメント  音楽  公開
ダウンロード  デスクトップ  ビデオ       画像
[student@h006 ~]$ ls /etc
DIR_COLORS              hosts.allow             python
DIR_COLORS.256color     hosts.deny              qemu-ga
DIR_COLORS.lightbgcolor  httpd                  rc.d
GREP_COLORS             init.d                  rc.local
（後略）
```

2.1.3 練習

```
[student@h006 ~]$ cd /home
[student@h006 home]$ pwd
/home
[student@h006 home]$ cd /home/student
[student@h006 ~]$ cd /etc
[student@h006 etc]$ ls
DIR_COLORS              hosts.allow             python
DIR_COLORS.256color     hosts.deny              qemu-ga
（後略）
[student@h006 etc]$ cd
[student@h006 ~]$ pwd
/home/student
```

2.1.4 練習

```
[student@h006 ~]$ cd ..
[student@h006 home]$ pwd
/home
[student@h006 home]$ cd ~
[student@h006 ~]$ pwd
/home/student
```

2.1.5 練習（関係ないファイルは非表示）

```
[student@h006 ~]$ cp /etc/resolv.conf ~
[student@h006 ~]$ cd
[student@h006 ~]$ ls
resolv.conf
[student@h006 ~]$ cp resolv.conf resolv.conf.bak
[student@h006 ~]$ ls
resolv.conf  resolv.conf.bak
```

2.1.6 練習

```
[student@h006 ~]$ cd /etc
[student@h006 etc]$ cat resolv.conf
# Generated by NetworkManager
search s143.la.net
nameserver 192.168.10.1
nameserver 172.30.0.1
[student@h006 etc]$ cd
[student@h006 ~]$ cat resolv.conf.bak
# Generated by NetworkManager
search s143.la.net
nameserver 192.168.10.1
nameserver 172.30.0.1
[student@h006 ~]$ cat /etc/bashrc
# /etc/bashrc
# System wide functions and aliases
# Environment stuff goes in /etc/profile
  （省略）
[student@h006 ~]$ less /etc/bashrc
```

2.1.7 練習（関係ないファイルは非表示）

```
[student@h006 ~]$ cd
[student@h006 ~]$ mv resolv.conf resolv.conf2
[student@h006 ~]$ ls
resolv.conf.bak  resolv.conf2
[student@h006 ~]$ cat resolv.conf2
# Generated by NetworkManager
search s143.la.net
nameserver 192.168.10.1
nameserver 172.30.0.1
```

2.1.8 練習（関係ないファイルは非表示）

```
[student@h006 ~]$ rm resolv.conf.bak
[student@h006 ~]$ ls
resolv.conf2
[student@h006 ~]$ cp resolv.conf2 resolv.conf
[student@h006 ~]$ rm resolv.conf2
[student@h006 ~]$ ls
resolv.conf
```

2.3.1 練習（関係ないファイルは非表示）

```
[student@h006 ~]$ cd ~
[student@h006 ~]$ ls -R
.:
resolv.conf
[student@h006 ~]$ mkdir schedule
[student@h006 ~]$ mkdir diary
```

```
[student@h006 ~]$ mkdir schedule/work
[student@h006 ~]$ mkdir schedule/work/meeting
[student@h006 ~]$ ls -R
.:
diary  resolv.conf  schedule

./diary:

./schedule:
work

./schedule/work:
meeting

./schedule/work/meeting:
```

2.3.2 練習（関係ないファイルは非表示）

```
[student@h006 ~]$ cd ~
[student@h006 ~]$ ls -R
（結果は 2.3.1 と同じ）
[student@h006 ~]$ rmdir schedule/work
rmdir: `schedule/work/' を削除できません: ディレクトリは空ではありません
[student@h006 ~]$ rm -r schedule/work
[student@h006 ~]$ ls -R
.:
diary  resolv.conf  schedule

./diary:

./schedule:
```

2.3.3 練習（関係ないファイルは非表示）

```
[student@h006 ~]$ mv schedule diary
[student@h006 ~]$ ls -R
.:
diary  resolv.conf

./diary:
schedule

./diary/schedule:
[student@h006 ~]$ mv diary/schedule diary/dream
[student@h006 ~]$ ls -R
.:
diary  resolv.conf
```

```
./diary:
dream

./diary/dream:
```

```
[student@h006 ~]$ cd ~
[student@h006 ~]$ cp -r diary diary2
[student@h006 ~]$ ls -R diary
diary:
dream

diary/dream:
[student@h006 ~]$ ls -R diary2
diary2:
dream

diary2/dream:
```

2.4 確認問題

```
[student@h006 ~]$ cd /

[student@h006 /]$ cd ./etc
[student@h006 etc]$ pwd
/etc

[student@h006 etc]$ cd
[student@h006 ~]$ pwd
/home/student

[student@h006 ~]$ cd ../../etc
[student@h006 etc]$ pwd
/etc

[student@h006 etc]$ cp /etc/postfix/main.cf ~
[student@h006 etc]$ less ~/main.cf
```

第 3 章

3.1.2 練習

```
[student@h006 ~]$ su -
パスワード:
最終ログイン: 2018/08/07 (火) 11:10:19 JST 日時 pts/0
[root@h006 ~]# useradd abe
[root@h006 ~]# passwd abe
ユーザー abe のパスワードを変更。
新しいパスワード:
よくないパスワード: このパスワードは 8 未満の文字列です。
新しいパスワードを再入力してください:
passwd: すべての認証トークンが正しく更新できました。
```

3.1.3 練習と補足

```
[student@h006 ~]$ grep abe /etc/passwd
abe:x:1001:1001::/home/abe:/bin/bash
```

＊補足事項）grep コマンドは、指定したキーワードを抽出します。

3.1.4 練習

```
[root@h006 ~]# useradd -u 2000 tanaka
[root@h006 ~]# grep tanaka /etc/passwd
tanaka:x:2000:2000::/home/tanaka:/bin/bash
[root@h006 ~]# useradd yoshida
[root@h006 ~]# grep yoshida /etc/passwd
yoshida:x:2001:2001::/home/yoshida:/bin/bash
```

3.1.5 練習

```
[root@h006 ~]# useradd -d /home/dir sato
[root@h006 ~]# grep sato /etc/passwd
sato:x:2002:2002::/home/dir:/bin/bash
[root@h006 ~]# ls /home
abe  dir  student  tanaka  yoshida
```

3.1.6 練習

```
[root@h006 ~]# userdel tanaka
[root@h006 ~]# ls -l /home
合計 0
drwx------  2 abe     abe      62  8月  7 11:56 abe
drwx------  2 sato    sato     62  8月  7 12:02 dir
drwx------. 6 student student 214  8月  7 11:51 student
drwx------  2    2000    2000  62  8月  7 12:00 tanaka
drwx------  2 yoshida yoshida  62  8月  7 12:01 yoshida
```

```
[root@h006 ~]# userdel -r sato
```

```
[root@h006 ~]# ls /home
abe  student  tanaka  yoshida
```

3.1.7 練習

```
[root@h006 ~]# groupadd eigyo
[root@h006 ~]# groupadd keiri
[root@h006 ~]# groupadd somu
[root@h006 ~]# cat /etc/group
（省略）
apache:x:48:
abe:x:1001:
yoshida:x:2001:
eigyo:x:2002:
keiri:x:2003:
somu:x:2004:
```

3.1.8 練習

```
[root@h006 ~]# useradd -g eigyo sato
[root@h006 ~]# grep sato /etc/passwd
sato:x:2002:2002::/home/sato:/bin/bash
[root@h006 ~]# id sato
uid=2002(sato) gid=2002(eigyo) groups=2002(eigyo)
```

```
[root@h006 ~]# usermod -g keiri -G eigyo,somu sato
[root@h006 ~]# grep sato /etc/passwd
sato:x:2002:2003::/home/sato:/bin/bash
[root@h006 ~]# id sato
uid=2002(sato) gid=2003(keiri) groups=2003(keiri),2002(eigyo),2004(somu)
```

3.1.9 練習

```
[root@h006 ~]# useradd -g somu kato
[root@h006 ~]# groupdel somu
groupdel: ユーザ 'kato' のプライマリグループは削除できません。
[root@h006 ~]# userdel -r kato
[root@h006 ~]# groupdel somu
```

第 4 章

4.1.3 練習

```
[student@h006 ~]$ cd
[student@h006 ~]$ cp /etc/fstab .
[student@h006 ~]$ ls -l fstab
-rw-r--r-- 1 student student 475  8月  7 12:21 fstab
[student@h006 ~]$ cat fstab
#
# /etc/fstab
# Created by anaconda on Mon Jul 23 09:16:28 2018
（省略）
[student@h006 ~]$ chmod a-r fstab
[student@h006 ~]$ ls -l fstab
--w------- 1 student student 475  8月  7 12:21 fstab
[student@h006 ~]$ cat fstab
cat: fstab: 許可がありません
[student@h006 ~]$ chmod 666 fstab
[student@h006 ~]$ ls -l fstab
-rw-rw-rw- 1 student student 475  8月  7 12:21 fstab
```

4.1.4 練習

```
[student@h006 ~]$ cd ~
[student@h006 ~]$ touch sample
[student@h006 ~]$ ln sample sample.hard
[student@h006 ~]$ ln -s sample sample.slink
[student@h006 ~]$ ls -l sample*
-rw-rw-r-- 2 student student 0  8月  7 12:28 sample
-rw-rw-r-- 2 student student 0  8月  7 12:28 sample.hard
lrwxrwxrwx 1 student student 6  8月  7 12:28 sample.slink -> sample
```

4.1.5 練習

```
[root@h006 ~]# cp /etc/hosts /home/student
[root@h006 ~]# ls -l /home/student
合計 36
drwxrwxr-x 3 student student    19  8月  7 11:39 diary
drwxrwxr-x 3 student student    19  8月  7 11:42 diary2
-rw-r--r-- 1 root    root      185  8月  7 12:15 hosts
-rw-r--r-- 1 student student 27176  8月  7 11:51 main.cf
-rw-r--r-- 1 student student    95  8月  7 11:31 resolv.conf
[root@h006 ~]# chown student /home/student/hosts
[root@h006 ~]# ls -l /home/student
合計 36
drwxrwxr-x 3 student student    19  8月  7 11:39 diary
drwxrwxr-x 3 student student    19  8月  7 11:42 diary2
-rw-r--r-- 1 student root      185  8月  7 12:15 hosts
```

```
-rw-r--r-- 1 student student 27176  8月  7 11:51 main.cf
-rw-r--r-- 1 student student    95  8月  7 11:31 resolv.conf
[root@h006 ~]# chown -R root.root /home/student
[root@h006 ~]# ls -l /home/student
合計 36
drwxrwxr-x 3 root root    19  8月  7 11:39 diary
drwxrwxr-x 3 root root    19  8月  7 11:42 diary2
-rw-r--r-- 1 root root   185  8月  7 12:15 hosts
-rw-r--r-- 1 root root 27176  8月  7 11:51 main.cf
-rw-r--r-- 1 root root    95  8月  7 11:31 resolv.conf
[root@h006 ~]# chown -R student.student /home/student/
[root@h006 ~]# ls -l /home/student
合計 36
drwxrwxr-x 3 student student    19  8月  7 11:39 diary
drwxrwxr-x 3 student student    19  8月  7 11:42 diary2
-rw-r--r-- 1 student student   185  8月  7 12:15 hosts
-rw-r--r-- 1 student student 27176  8月  7 11:51 main.cf
-rw-r--r-- 1 student student    95  8月  7 11:31 resolv.conf
```

4.2 確認問題

```
[student@h006 ~]$ pwd
/home/student
[student@h006 ~]$ mkdir tmp
[student@h006 ~]$ chmod 755 ~ tmp
[student@h006 ~]$ cd tmp
[student@h006 tmp]$ mkdir spring summer fall winter
[student@h006 tmp]$ chmod 750 spring/
[student@h006 tmp]$ chmod 751 summer/
[student@h006 tmp]$ chmod 752 fall/
[student@h006 tmp]$ chmod 754 winter/
[student@h006 tmp]$ ln -s summer summer.slink
[student@h006 tmp]$ ls -l
合計 0
drwxr-x-w- 2 student student 6  8月  7 12:37 fall
drwxr-x--- 2 student student 6  8月  7 12:37 spring
drwxr-x--x 2 student student 6  8月  7 12:37 summer
lrwxrwxrwx 1 student student 6  8月  7 12:38 summer.slink -> summer
drwxr-xr-- 2 student student 6  8月  7 12:37 winter
[student@h006 tmp]$ su abe
パスワード:
[abe@h006 tmp]$ id
uid=1001(abe) gid=1001(abe) groups=1001(abe)

[abe@h006 tmp]$ cd spring/
bash: cd: spring/: 許可がありません
（省略）
```

```
[abe@h006 tmp]$ ls spring/
ls: ディレクトリ spring/ を開くことが出来ません: 許可がありません
（省略）

[abe@h006 tmp]$ touch spring/data
touch: `spring/data' に touch できません: 許可がありません
（省略）
```

確認結果まとめ（◯が成功、X が失敗）

ディレクトリ	cd	ls	touch
spring	X	X	X
summer	◯	X	X
summer.slink	◯	X	X
fall	X	X	X
winter	X	◯	X

考察）一見、fall ではファイル作成で来そうだが、実際には実行権が必要となる。

第 5 章

5.1.1 練習

```
[student@h006 ~]$ vi hello.txt
```

```
~
~
"hello.txt" [新ファイル]                              0,0-1         全て
```

```
:q!
```

```
[student@h006 ~]$ ls
diary   fstab  main.cf      sample       sample.slink
diary2  hosts  resolv.conf  sample.hard  tmp
```

5.1.2 練習

```
[student@h006 ~]$ vi hello.txt
[student@h006 ~]$ cat hello.txt
Hello!
I'm using vi on CentOS.

[student@h006 ~]$
```

5.1.7 練習

```
[student@h006 ~]$ vi address.memo
[student@h006 ~]$ cat address.memo
Akihito YAKOSHI  ycos
Birthday         1966/07/05
Email                    ycos001@yahoo.co.jp
Hobby                    Rock music and Drink
```

5.2 確認問題

```
[student@h006 ~]$ vi ~/.bashrc
（以下を最後に追加）
PS1="[\u@\h \W \t]\$ "
[student@h006 ~]$ su student
パスワード:
[student@h006 ~ 13:04:24]$
```

第 6 章

6.1 練習

自明なため割愛

6.2.2 練習

自明なため割愛

6.2.3 練習

[Ctrl] + [Alt] + [F2]

```
CentOS Linux 7 (Core)
Kernel 3.10.0-862.9.1.el7.x86_64 on an x86_64

localhost login: student
Password:
Last login: Tue Aug  7 13:04:24 on pts/0

[student@h006 ~]$ ps aux | grep tty2
student  5802  0.0  0.0 116100  2776 tty2 Ss  13:37  0:00 -bash
student  5931  0.0  0.0 151112  1852 tty2 R+  13:38  0:00 ps aux
student  5932  0.0  0.0 112708   960 tty2 S+  13:38  0:00 grep --color=auto tty2

[student@h006 ~]$ pstree -p
systemd(1)─┬─ModemManager(680)─┬─{ModemManager}(721)
           │                   └─{ModemManager}(723)
           ├─NetworkManager(679)─┬─{NetworkManager}(744)
           │                     └─{NetworkManager}(746)
           ├─abrt-dbus(5829)─┬─{abrt-dbus}(5832)
（省略）
```

[Ctrl] + [Alt] + [F3]

```
[student@h006 ~]$ tty
/dev/tty3
[student@h006 ~]$ less /etc/bashrc

[Ctrl] + [Alt] + [F4]
[student@h006 ~]$ tty
/dev/tty4
[student@h006 ~]$ ps aux |grep tty3
student  6168  0.0  0.0 116100 2836 tty3  Ss  13:52  0:00 -bash
student  6273  0.0  0.0 110308  984 tty3  S+  13:52  0:00 less /etc/bashrc
student  6384  0.0  0.0 112708  960 tty3  S+  13:54  0:00 grep --color=auto tty3
[student@h006 ~]$ kill 6273
```

```
[Ctrl] + [Alt] + [F3]


      if [ -e /etc/sysconfig/bash-prompt-xterm ]; then
          PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
      elif [ "${VTE_VERSION:-0}" -ge 3405 ]; then
          PROMPT_COMMAND="__vte_prompt_command"
/etc/bashrcTerminated
[student@h006 ~]$
```

6.3.2 練習

```
[student@h006 ~]$ gedit &
[1] 6675
[student@h006 ~]$ ps aux | grep gedit
student  6675  4.1  0.3 750996 28228 pts/0  Sl  14:06  0:00 gedit
student  6686  0.0  0.0 112724   972 pts/0  S+  14:06  0:00 grep --color=auto gedit
[student@h006 ~]$ jobs
[1]+  実行中              gedit &
```

6.3.4 練習

```
[student@h006 ~]$ jobs
[1]   実行中              gedit &
[2]-  実行中              gnome-clocks &
[3]+  実行中              gnome-calculator &
[student@h006 ~]$ kill %
[student@h006 ~]$ jobs
[1]   実行中              gedit &
[2]-  実行中              gnome-clocks &
[3]+  Terminated                gnome-calculator
[student@h006 ~]$ kill %2
[2]+  Terminated                gnome-clocks
[student@h006 ~]$ kill %1
[1]+  Terminated                gedit[student@h006 ~]$ gedit &
```

6.4 確認問題

```
[student@h006 ~]$ top
```

```
top - 13:40:20 up  2:30,  1 user,  load average: 0.00, 0.01, 0.03
Tasks:  90 total,   1 running,  89 sleeping,   0 stopped,   0 zom
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0
KiB Mem :  1015500 total,   823864 free,    68308 used,   123328
KiB Swap:   839676 total,   839676 free,        0 used.   805104

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+
  895 root      20   0  573852  17024   6036 S  0.7  1.7   0:01.36
    1 root      20   0  125336   3772   2568 S  0.0  0.4   0:00.88


   15 root      20   0       0      0      0 S  0.0  0.0   0:00.00
```

```
   16 root       0 -20       0      0      0 S  0.0  0.0   0:00.00
[1]+  停止                 top
[student@h006 ~]$ jobs
[1]+  停止                 top
[student@h006 ~]$ ps aux | grep top
student  2354  0.0  0.2 161840  2156 pts/0  T   13:40  0:00 top
student  2385  0.0  0.0 112720   976 pts/0  R+  13:46  0:00 grep --color=auto top
[student@h006 ~]$ fg %1
1]+  停止                 top
[student@h006 ~]$ kill %1

[1]+  停止                 top
[student@h006 ~]$ jobs
[1]+  停止                 top  ←停止しない
[student@h006 ~]$ clear
[student@h006 ~]$ kill -9 %1

[1]+  停止                 top
[student@h006 ~]$ [Enter]
[1]+  強制終了            top
```

第 7 章

7.1.2 練習

```
[student@h006 ~]$ echo "Hello" > hello.txt
[student@h006 ~]$ cat hello.txt
Hello
```

7.2.1 練習

```
[student@h006 ~]$ ps aux | grep bash
student  1452  0.0  0.2 115580  2140 pts/0  Ss  17:10  0:00 -bash
student  1476  0.0  0.0 112720   976 pts/0  R+  17:13  0:00 grep --color=autobash
```

7.2.2 練習

```
[student@h006 ~]$ ls ~ && echo "OK"
address.memo  hello.txt  main.cf     resolv.conf  time.sh  top.log
fstab         hosts      pstree.log  syslogd.txt  tmp
OK
[student@h006 ~]$ ls /uso800 || echo "NG"
ls: /uso800 にアクセスできません: そのようなファイルやディレクトリはありません
NG
```

7.2.3 練習

```
[student@h006 ~]$ ps aux|grep syslogd|tee syslogd.txt
root      895  0.0  0.3 214532  3640 ?      Ssl  16:23  0:00 /usr/sbin/rsyslogd -n
student  1628  0.0  0.0 112720   972 pts/0  R+   17:20  0:00 grep --color=auto syslogd
student  1629  0.0  0.0 108208   660 pts/0  S+   17:20  0:00 tee syslogd.txt
[student@h006 ~]$ cat syslogd.txt
root      895  0.0  0.3 214532  3640 ?      Ssl  16:23  0:00 /usr/sbin/rsyslogd -n
student  1628  0.0  0.0 112720   972 pts/0  R+   17:20  0:00 grep --color=auto syslogd
student  1629  0.0  0.0 108208   660 pts/0  S+   17:20  0:00 tee syslogd.txt
[student@h006 ~]$ pstree | tee pstree.log
systemd-+-NetworkManager-+-2*[dhclient]
        |                `-2*[{NetworkManager}]
        |-agetty
        |-auditd---{auditd}
（省略）
[student@h006 ~]$ cat pstree.log
systemd-+-NetworkManager-+-2*[dhclient]
        |                `-2*[{NetworkManager}]
        |-agetty
（省略）
```

7.3 確認問題と補足

```
[student@h006 ~]$ top | tee top.log
（１０秒程度放置）
```

```
[Ctrl] + [C]
[student@h006 ~]$ cat top.log
```

（補足）
Linux のコンソールは、curses と呼ばれる機能を使って、テキストのカーソルを上下左右、タブをエスケープシーケンスと呼ばれる特殊文字（[ESC]で始まるためこの名があります）に移動させています。

```
[student@h006 ~]$ vi top.log
```

```
^[[?1h^[=^[[?25l^[[H^[[2J^[(B^[[mtop - 18:10:58 up  1:47,  1 user,  load average:
0.00, 0.01, 0.01^[(B^[[m^[[39;49m^[(B^[[m^[[39;49m^[[K
Tasks:^[(B^[[m^[[39;49m^[[1m  91 ^[(B^[[m^[[39;49mtotal,^[(B^[[m^[[39;49m^[[1m   1
^[(B^[[m^[[39;49mrunning,^[(B^[[m^[[39;49m^[[1m  90
```

第８章

8.2.3 練習の結果例

```
[root@h006 ~]# mount -o ro /dev/cdrom /media
[root@h006 ~]# df
ファイルシス                1K-ブロック     使用   使用可 使用% マウント位置
/dev/mapper/centos_h006-root     6486016 1386788 5099228   22% /
devtmpfs                          495956       0  495956    0% /dev
tmpfs                             507748       0  507748    0% /dev/shm
tmpfs                             507748    6828  500920    2% /run
tmpfs                             507748       0  507748    0% /sys/fs/cgroup
/dev/sda1                        1038336  161724  876612   16% /boot
tmpfs                             101552       0  101552    0% /run/user/1000
/dev/sr0                         1354850 1354850       0  100% /media
[root@h006 ~]# ls /media/
LiveOS  isolinux
[root@h006 ~]# umount /media
[root@h006 ~]# df
ファイルシス                1K-ブロック     使用   使用可 使用% マウント位置
/dev/mapper/centos_h006-root     6486016 1386788 5099228   22% /
devtmpfs                          495956       0  495956    0% /dev
tmpfs                             507748       0  507748    0% /dev/shm
tmpfs                             507748    6828  500920    2% /run
tmpfs                             507748       0  507748    0% /sys/fs/cgroup
/dev/sda1                        1038336  161724  876612   16% /boot
tmpfs                             101552       0  101552    0% /run/user/1000
```

8.5 確認問題の結果例

```
[root@h006 ~]# mount -r /dev/cdrom /media
[root@h006 ~]# cd /media/Packages
[root@h006 Packages]# rpm -ivh libsmi-* wireshark-*
準備しています...              ################################# [100%]
更新中 / インストール中...
   1:libsmi-0.4.8-13.el7             ################################# [ 33%]
   2:wireshark-1.10.14-14.el7        ################################# [ 67%]
   3:wireshark-gnome-1.10.14-14.el7  ################################# [100%]
[root@h006 Packages]# rpm -qi wireshark
Name        : wireshark
Version     : 1.10.14
Release     : 14.el7
Architecture: x86_64
Install Date: 2018年08月08日 10時25分40秒
Group       : Applications/Internet
（省略）
[root@h006 Packages]# yum update wireshark
```

```
読み込んだプラグイン:fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.tsukuba.wide.ad.jp
 * centosplus: ftp.tsukuba.wide.ad.jp
 * extras: ftp.tsukuba.wide.ad.jp
 * updates: ftp.tsukuba.wide.ad.jp
No packages marked for update

[root@h006 Packages]# rpm -e libsmi
エラー: 依存性の欠如:
  libsmi.so.2()(64bit) は (インストール済み)wireshark-1.10.14-14.el7.x86_64 に必要とされてい
ます
[root@h006 Packages]# rpm -e libsmi --nodeps
[root@h006 Packages]# rpm -ivh libsmi-*
準備しています...              ################################# [100%]
更新中 / インストール中...
   1:libsmi-0.4.8-13.el7             ################################# [100%]
```

ダウンロード

本テキストの補足的な内容は、以下のサイトからダウンロードできます。

付録 1：vi コマンド一覧

`http://s.linuxacademy.ne.jp/linuxbasic_apdx1.pdf`