ご利用になる前に必ずお読みください

このＰＤＦファイルの内容についてのご質問・お問い合わせは株式会社アスキー・メディアワークスでは一切お受けできません。ご自身の責任においてご利用ください。

クレジット：
●文/水野源（Ubuntu Japanese Team）
●まんが/瀬尾浩史
●デザイン/シオズミタロウ
●初出/株式会社アスキー・メディアワークス「Ubuntu Magazine Japan vol.03」（http://ubuntu.asciimw.jp/）2010 年 2 月 23 日発行

not scary!! コマンドなんて怖くない not scary

はじめてのシェルスクリプト入門

Introduction to shell script

コマンドって何？どう使うの？

CLIという便利ツールを使いこなそう

マウスの代わりにキーボードで命令！

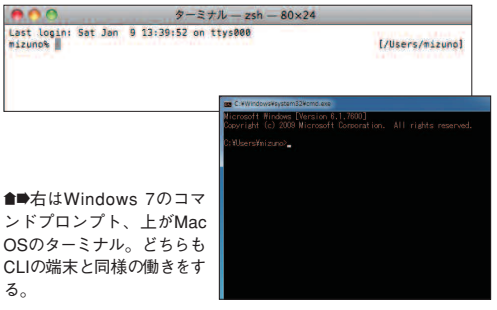
Windows、MacOS、そしてUbuntu。これらの先進的なOSは、標準的な操作としてマウスを使ったGUI（グラフィカル・ユーザ・インタフェース）での操作が想定されている。しかし一般的なユーザが目にする機会は少ないかもしれないが、これらのOSはすべて、キーボードからコマンドを入力して操作を行うCLI（コマンドラインインタフェース）の機能も備えている。

使いこなすには多少勉強が必要なCLIには「難しい」、「古くさい」、「マニア専用」といったイメージがあるようで、「コマンド操作が必要になるUbuntuは、初心者には使いこなせない」などといった批判につながることもある。

しかしそれは誤解だ。Ubuntuは基本的にマウスのみで操作ができ、一般的な用途では、ユーザが「端末」にコマンドを打ち込む必要はないように設計されているからだ。だが大量のファイルの処理など、マウスでの手作業では作業量が膨大になりすぎて現実的ではないことがある。そういったGUIが苦手とする分野の作業を楽に行うための便利ツールの一つが、CLIなのだ。

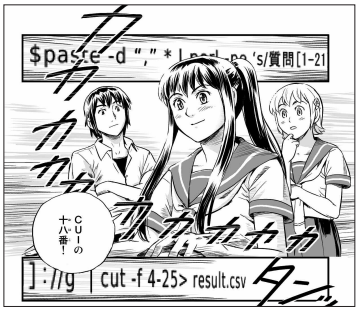
GUIとCLIにはそれぞれ、得意分野と苦手分野が存在する。全ての作業をマウスだけで行おうとするのはときに非効率的だし、またコマンドのみで生活するのもナンセンスだ。行う作業によってこれらを適切に使い分けてこそ、『うぶんちゅの一番星』（会長談）と言えるだろう！

ほかのOSでもコマンドは使う！



▲▶右はWindows 7のコマンドプロンプト、上がMac OSのターミナル。どちらもCLIの端末と同様の働きをする。

会長のようなうぶんちゅの一番星に！



◀複数のテキストファイルをあっと言う間にCSVにしてしまう、会長の腕を見習いたい!!

コマンド1行でこんなことも!!

```
$ for n in *.jpg; do convert ${n}
-resize 100 ${n%jpg}png; done
```

▲上の1行のコマンドを使うだけで、カレントディレクトリ内にあるJPEG画像をすべてに対し横100ピクセルに縮小したPNGサムネイル画像を追加できる（あらかじめ、ImageMagickをインストールしておく必要がある）。

コマンドって何？どう使うの？

端末とかコマンドを使って何ができる？

コマンドなら一発で処理できることも！

コマンドとは、コンピュータに仕事を行わせるための「命令」のことだ。ユーザはキーボードから、コマンドラインインタプリタ（シェル）を通してコマンドを入力する。Ubuntuでは『GNOME端末』で「bash」というシェルを通してコマンドを入力するのが標準だ。ちなみに「端末」は「端末エミュレータ」の略で、一般にコンソール、ターミナルなどと呼ばれることもある。

この特集を読んだら、さっそく「端末」を起動してコマンドを実行してみよう。GNOME端末は［アプリケーション］-［アクセサリ］-［端末］から起動できる。

●文/水野源（Ubuntu Japanese Team）、●マンガ/瀬尾浩史

コマンドなんて怖くない！

ファイルやフォルダの位置を指示するためのキホン

ディレクトリ構造の違いを知っておく

Ubuntuではルートが頂点の構造

ディレクトリとは、WindowsやMacで言うところの「フォルダ」と同じ、ファイルを整理するための入れ物と考えて差し支えない。Ubuntuを含むUnix系OSもWindowsと同じようにディレクトリがツリー構造の階層になっており、ファイルは決まったルールに従って分類され、それぞれのディレクトリに保存されている。Windowsとの大きな違いは、Windowsが「ドライブ」ごとに独立したディレクトリツリーを持つのに対し、Unixでは「／（ルート）」を頂点とした唯一のディレクトリツリーを持つところだ。複数のハードディスクが一つのツリーにマウントされることに、最初は違和感を覚えるかもしれない。しかし慣れれば「ドライブをまたぐ」必要がないのは非常にわかりやすく、シンプルであることに気づくはずだ。

ディレクトリの移動とカレントディレクトリ

端末を起動すると「ユーザ名@ホスト名:~$」という表示がされているだろう。このうち「$」の直前にある「~（チルダ）」が、現在自分のいる位置「カレントディレクトリ」を表している。「cd」コマンドを使ってディレクトリを移動し、カレントディレクトリの表記が変化するところを見てみよう。

コマンドを実行する時には、コマンドやファイルの位置をきちんと指示する必要がある。後述する相対パスで指定する場合には、現在位置であるカレントディレクトリが非常に重要になってくるため、自分が現在ツリー上のどこにいるかには常に気にかけておこう。

隠しファイルと隠しディレクトリ

ファイルやディレクトリの中には、通常では表示されないものがある。端末を起動して、ホームディレクトリ上でファイルのリストを表示する「ls」コマンドを実行してみよう。

次に「ls」コマンドに「-a」オプションをつけてみよう。「-a」オプションは、隠しファイルや隠しディレクトリを表示するためのオプションだ。ファイル名の先頭に「.（ドット）」がついたファイルが大量に表示されるだろう。これが隠しファイルだ。

隠しファイルはなにも特別なファイルではなく、自分で作成したファイルであっても、先頭にドット（ピリオド）が付いた名前にすることで隠しファイルにできる。隠しファイルは主にプログラムの設定などが記述されているファイルなので、意味がわからないなら編集や削除はするべきではない。そのうえ大量に存在するため、普段から目に見えると邪魔になってしまう。そのため「ls」コマンドや、デフォルト状態のGNOMEのファイルブラウザでは表示されないようになっている。

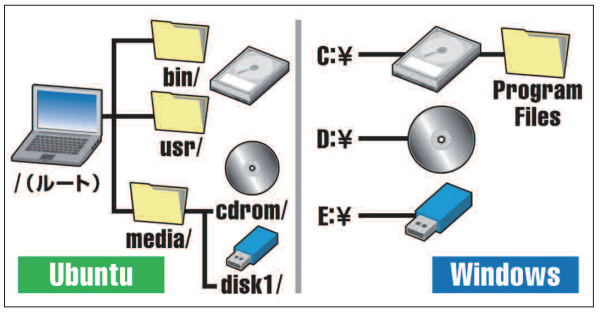
位置指定の方法 絶対パスと相対パス

ファイル名だけでは、そのファイルがディレクトリツリー上のどこに存在するか特定できない。そこで「絶対パス」「相対パス」という、ファイルの位置を表す記法を使おう。

絶対パスとは、ディレクトリツリーの頂点である「／（ルート）」からの道筋（パス）で表す記法だ。それに対し相対パスとは、カレントディレクトリからの相対位置で表す記法だ。

絶対パスはツリーの頂点からのパスを表すため、カレントディレクトリに影響されず、どこから呼び出しても同じファイルを一意に指定できるという安定性がある。逆に相対パスは、カレントディレクトリと対象の位置関係が変化しなければ、ディレクトリツリー上のどのパスにあっても構わないという柔軟性がある。いつどこから呼び出されるかわからない処理で、特定のファイルを確実に指定したい場合は絶対パスを、呼び出される場所は不定だが、対象との位置関係は崩れないことが確実な処理などでは相対パスを使うとよいだろう。またパスの指定には左のような特殊記号を使用できる。よく使うので覚えておこう。

Unix系OSのディレクトリ構造



プロンプトの変化

ディレクトリの移動

```
ユーザ名@ホスト名:~$ cd /usr
```

```
ユーザ名@ホスト名:/usr$
```

■「/usr」に移動したためカレントディレクトリが変更され、プロンプトの表示が変化した。

ファイルのリストを表示する

lsコマンドの実行例

```
$ ls
Desktop     Downloads  Pictures   Templates   examples.desktop
Documents   Music      Public     Videos
```

ls -aコマンドの実行例

```
$ ls -a
.               .fbtermrc        .mozilla        .viminfo
..              .folders         .nautilus       .wapi
.ICEauthority   .gconf           .navi2ch        .wl
(以下略)
```

絶対パスと相対パス

絶対パスの例

```
$ /usr/bin/firefox
```

相対パスの例

```
$ ../../usr/bin/firefox
```

■「/home/usr」からファイル「firefox」を指定する場合。絶対パス、相対パスの2通りで指定できる。

パスの指定に使える特殊記号

~（チルダ）	ユーザのホームディレクトリを表す
.（ドット）	カレントディレクトリを表す
..（ドットドット）	一つ上の階層のディレクトリ（カレントディレクトリが/usr/binなら/usr）を表す

■端末を起動した時点でのカレントディレクトリはユーザのホームディレクトリ「~」だ。

コマンドなんて怖くない!! not scary!!

権限って何？管理者権限で何ができるの？

ファイルに対してできる操作を知ろう

権限＝パーミッションでファイルは守られる

Ubuntuをはじめ、Unixではファイルやディレクトリへのアクセスを設定できる。具体的にはファイルの「読み込み」「書き込み」「実行」の権限を「ファイルの持ち主」「グループ」「その他」のそれぞれに対して設定できる。これをパーミッションと呼び、ファイルにどのようなパーミッションが設定されているかは「ls」コマンドに「-l」オプションをつけることで表示できる。

左図では、一番左の文字列が、対象への「読み込み(r)」「書き込み(w)」「実行(x)」の許可を表している。読み込み権限のないファイルを読もうとしたり、書き込み権限のないファイルを編集しようとすると「Permission denied」エラーが発生する。もしこのエラーに遭遇したら、アクセス権限が適切に設定されているか確認しよう。なおアクセス権限は「chmod」、ファイルの持ち主やグループは「chown」や「chgrp」コマンドで変更ができる。

ユーザだけでなくグループも設定できる

権限の管理で、グループという概念が出てきたので簡単に説明しよう。Unixはマルチユーザで、ファイルには持ち主と権限が設定されている。個人でデスクトップOSとしてUnixを使っているなら「持ち主」と「その他」だけでも特に困ることはないのだが、これでは多人数が使用するサーバなどでは柔軟なアクセス制限が行えない。

具体的な例として会社の帳簿を考えてみよう。経理部のメンバならば帳簿を閲覧できなければ困るが、経理に関係ない社員が帳簿を閲覧できてしまっては問題だ。そのような場合は経理部のグループを作成し、経理部の各ユーザアカウントをグループに所属させる。そして帳簿ファイルのアクセス権限を設定することで「グループメンバは全員が読み書きできるが、グループに所属していない人は読み書きできない」というようなアクセス制限をかけられる。なおグループの作成やユーザの所属グループの変更には、「groupadd」コマンドや「usermod」コマンドが使用できる。

システムの変更は管理者権限で実行

コマンドの中には、システム管理者でなければ実行できないコマンドが存在する。システム全体に影響を及ぼすコマンド、具体的にはパッケージのインストールやユーザの削除、システムのシャットダウンなどがそれだ。例えばシステムをアップデートする「apt-get update」コマンドを実行してみると、左のようなエラーが出るはずだ。

エラーメッセージを手がかりにロックファイルのパーミッションを確認してみると「root」ユーザのみが読み書きできることになっている。つまり、一般ユーザではロックの取得ができないため、「apt-get update」コマンドは実行できないのだ。

rootユーザとは、システムに対してあらゆる変更や操作が行える、管理者ユーザのことだ。rootユーザは自由にシステム設定の変更が行えるため、このユーザのパスワードが盗まれるとシステムが完全に乗っ取られてしまう危険性もある。

Ubuntuでは一時的にrootを使う

Ubuntuでは、「sudo」というコマンドを用いて一時的にroot権限を取得することになっている。「sudo」はコマンドを別のユーザとして実行するためのコマンドだ。rootとして「apt-get」コマンドを実行するには左上図のようにしよう。

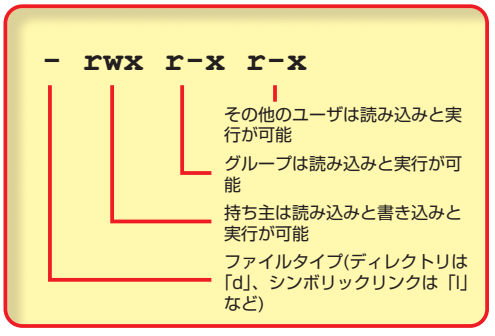
「sudo」コマンドを実行すると、パスワードの入力を求められる。ここで入力すべきパスワードは「自分自身のログインパスワード」だ。Windowsのように、パスワード入力中に「*」文字は表示されないが、入力は行われているので安心してほしい。一度「sudo」を実行すると、以降5分間はパスワードなしで「sudo」が実行できる。一見便利だがこの状態は「認証なしで管理者権限を取得できる危険な状態」であるとも言える。この認証済み状態をただちに終了させたい場合は、「sudo」に「-k」オプションをつけて実行しよう。

なお「sudo」コマンドでroot権限を得られるのはadminグループに所属するユーザのみに限定されている。インストール時に作成したユーザは自動的にadminグループに所属しているが「useradd」コマンドなどで後から追加したユーザは必要ならadminグループに追加しておこう。

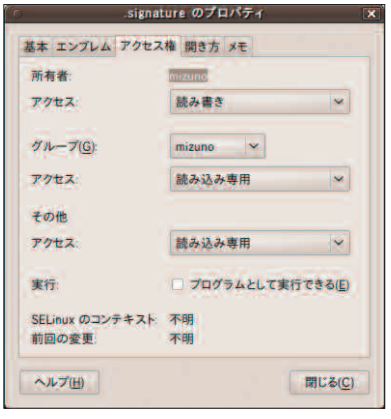
ファイルのパーミッションを調べる

```
$ ls -l /bin/cat
-rwxr-xr-x 1 root root 50876 2009-10-06 20:07 /bin/cat
```

パーミッションの読み方



ファイルのプロパティで確認



➡ファイルを右クリックし、プロパティを開いて「アクセス権」タブを見ても同様に確認できる。

apt-get updateの実行

```
$ apt-get update
E: ロックファイル /var/lib/apt/lists/lock をオープンできません - open (13: Permission denied)
E: list ディレクトリをロックできません
```

ロックファイルのパーミッションを確認

```
$ ls -l /var/lib/apt/lists/lock
-rw-r----- 1 root root 0 2009-10-29 05:56 /var/lib/apt/lists/lock
```

sudoで管理者権限を得てコマンドを実行

```
$ sudo apt-get update
```

端末（シェル）にコマンドを省入力するコツ

【コマンドの基本を知っておこう】

オプションとパラメータを付けてみる

コマンドには「オプション」と「パラメータ」を渡すことができる。たとえば「firefox」コマンドの後に「-P」をつけて実行してみよう。

$ firefox -P

ユーザプロファイルの選択画面が表示されるはずだ。この「-P」のようなものを「オプション」といい、オプションによってコマンドの挙動を変えられる。

次に「firefox」コマンドの後にURLをつけて実行してみよう。

$ firefox http://ascii.jp

起動したFirefoxが、指定したURLのページを開いたはずだ。このようにプログラムに与える値のことを「パラメータ」や「引数」などと呼ぶ。なおコマンド名、オプション、パラメータはそれぞれ半角スペースで区切る必要があるので気をつけて欲しい。

入力中のコマンドはエディタで文章を編集するように、追加、削除、編集を行うこともできる。カーソルキーの［←］や［→］でカーソルを移動し、［BackSpace］キーで削除、キーをタイプすればその場に文字が挿入される。

Tabキーでコマンドを補完しよう

コマンドやパラメータは一文字でも間違えると正常に動作しない。また大文字と小文字は厳密に区別されるため、正確にタイプする必要がある。しかし時には非常に長くなることもあるコマンドやパラメータを、間違えずタイプすることは困難だ。そんな時は［Tab］キーを用いたシェルの補完機能を使ってみよう。コマンド入力中に［Tab］キー、または［Ctrl］＋［i］を押すと、後に続く文字を自動的に補完してくれるため、素早くかつ正確にコマンドを入力することができるようになる。

まず端末から、「apt-」と入力した状態で［Tab］キーか［Ctrl］＋［i］を押してみよう。

$ apt-（［Tab］キーを押す）

すると左図のように「apt-」ではじまるコマンドの一覧が表示される。Ubuntuには「apt-」ではじまるコマンドが複数存在するため、シェルがコマンドの候補を表示してくれたわけだ。次は、「apt-g」と入力した状態で［Tab］キーか［Ctrl］＋［i］キーを押してみよう。

$ apt-g（［Tab］キーを押す）

apt-で始まるコマンドのリストを見るとわかるように、「apt-g」で始まるコマンドは「apt-get」しか存在しない。つまり「apt-g」までタイプすれば、実行したいコマンドは「apt-get」であると特定できるわけだ。その場合、シェルはコマンドの末尾までを一気に補完してくれるのだ。

bashの基本的な操作

キーバインド	別のキー操作	意味
Ctrl+b	←	カーソル左移動
Ctrl+f	→	カーソル右移動
Ctrl+p	↑	直前の履歴呼び出し
Ctrl+n	↓	直後の履歴呼び出し
Ctrl+i	Tab	コマンドの補完
Ctrl+a	HOME	カーソル行頭移動
Ctrl+e	END	カーソル行末移動
Ctrl+h	BackSpace	カーソルの前の文字を削除
Ctrl+d	Del	カーソル位置の文字を削除
Ctrl+r	なし	履歴の後方検索
Ctrl+s	なし	履歴の前方検索
Ctrl+k	なし	カーソル位置から行末までをカット
Ctrl+w	なし	カーソル位置から直前の1ワードをカット
Ctrl+y	なし	カーソル位置にヤンク(ペースト)
Ctrl+l	なし	端末のクリア

［Tab］キーでコマンドを補完

```
$ apt-（［Tab］キーを押す）
apt-cache                apt-file
apt-mark
apt-cdrom                apt-ftparchive
apt-sortpkgs
apt-config               apt-get
apt-extracttemplates     apt-key
```

```
$ apt-g （gを追加して［Tab］キーを押す）
```

```
$ apt-get
```

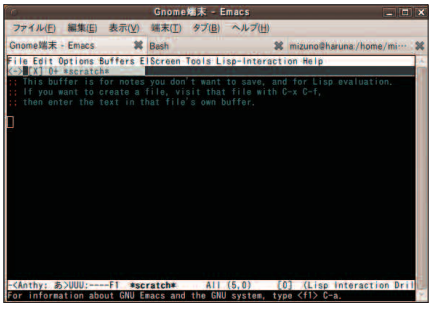
仮想端末って、いつ使うの？

GNOMEデスクトップが表示されている状態で、［Ctrl］＋［Alt］＋［F1〜F6のどれか］を押してみよう。真っ黒な画面に白い文字でログインプロンプトが出ているだけの、殺風景な端末画面に切り替わるのが確認できるだろう。実はLinuxには1から7までの仮想端末が存在し、GNOMEデスクトップは7番目に表示されている。［Ctrl］＋［Alt］＋［F7］でデスクトップ画面に戻ることができる。ちなみに現在接続している端末番号は「tty」コマンドで確認ができる。

この仮想端末の切り替えを使うことで、同時に複数の作業を行うことができる。例えば仮想端末1ではエディタでメールを書きながら、仮想端末2ではファイルのコピーを行う、などだ。

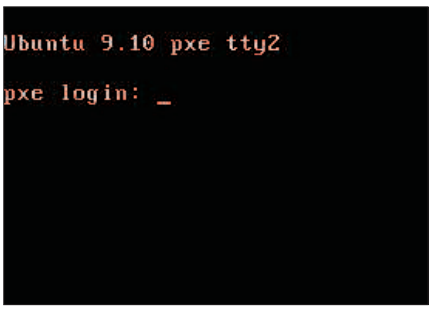
しかし仮想端末はXウィンドウシステムを使用していないので、GNOME端末のようにウィンドウサイズを変更したり、マウスホイールでバックログを閲覧するようなことはできないし、GNOMEデスクトップ上では複数のGNOME端末を立ち上げられるので、わざわざこんな不便なことをする必要はない。しかしXがデフォルトではインストールされないサーバ版Ubuntuや、Xが使えない状態（NVIDIAのドライバをリポジトリからではなく、手動でインストールする場合）などでは非常に有効だ。参考程度に覚えておくと、役に立つことがあるかもしれない。

GNOME端末は便利



➡タブで切り替えたり、［編集］-［プロファイルの設定］で背景画像やフォント、配色を変更したりできる。

仮想端末の画面



➡Xを使わないぶん軽量だが、一時的な作業のほかはあえてメインの端末として使う理由はない。

さらにTab補完を使いこなそう!!

コマンドに渡すパラメータも補完!

例に挙げた「apt-get」コマンドはパッケージの管理を行うためのコマンドだが、アップデートやインストールなどを行うためにサブコマンドを指定する必要がある。またパッケージをインストールするためにはパッケージ名の指定も必要だ。補完機能はこういった「コマンドに渡すパラメータ」でも有効になっている。「apt-get」コマンドに続いて［Tab］キーを2回押してみよう。すると「apt-get」コマンドで使えるサブコマンドの一覧が表示される。当然、パッケージ名も補完できる。

このように、Tab補完を使うことでコマンドやパラメータを間違えず、かつ素早く入力することが可能になる。コマンドは手で入力するのではなく、可能な限り補完機能に任せるように心がければ、コマンド入力の効率も大幅に上がるはずだ。

［Tab］キーでサブコマンド補完

```
$ apt-get（［Tab］キーを2回押す）
autoclean       clean           purge
upgrade
autoremove      dist-upgrade    remove
build-dep       dselect-upgrade source
install           update
```

絞り込んで一発補完！

```
$ apt-get i  （［Tab］キーを押す）
  ↓
$ apt-get install
```

■「i」ではじまるサブコマンドはinstallだけなので、一発で補完される。

［Tab］キーでパッケージ名補完

```
$ apt-get install thund  （［Tab］キーを押す）
  ↓
$ apt-get install thunderbird
```

■インストールできるパッケージ名も補完された！

過去に実行したコマンドを再利用

コマンドヒストリを使ってさらにラクに！

Tab補完を使えば、長いコマンドでも比較的楽に入力できるようになったはずだ。しかしどれだけTab補完が便利でも、似たようなコマンドを何度も入力するのは無駄というもの。そこで過去に実行したことのあるコマンドは、コマンドヒストリを使って再実行しよう。

まず「history」コマンドを実行してみよう。左の例のように、過去に実行したコマンドの履歴が表示されるはずだ。このように、シェルは実行したコマンドを記録しており、この記録を呼び出すことで過去に実行したコマンドを再利用できるようになっている。

プロンプトの状態でカーソルキーの［↑］か［Ctrl］＋［p］を押すと、最後に実行したコマンドから順にコマンドの履歴を過去に遡ることができる。行きすぎてしまった場合は、カーソルキーの［↓］か［Ctrl］＋［n］で戻れることを覚えておこう。［p］は「Previous」、［n］は「next」の略と考えると覚えやすい。ちなみにシェル上でのこれらのキー操作はプログラマに人気のエディタ「Emacs」のキーバインドがベースになっている。

historyで入力履歴を見る

```
$ history
    1  lsusb
    2  sudo apt-get update
    3  sudo apt-get dist-upgrade
    4  sudo lshw -short
    5  tail -f /var/log/messages
    8  xinput list-props
```

入力履歴を呼び出そう！

```
$ echo "hoge"
$ ls /usr/bin（［↑］キーを押す）
$ ls /usr/bin
最後に実行したコマンド入力が呼び出される
↓
（さらに［↑］キーを押す）
$ echo "hoge"
```

■2つ前に実行したコマンド入力が呼び出された。

履歴は時系列にさかのぼるだけでなく、検索もできる。検索を行う場合は［Ctrl］＋［r］を押そう。プロンプトが（reverse-i-search）に変化し、検索のためのキーワードの入力を受け付ける状態になる。ここで検索したいキーワードを入力すると、リアルタイムにインクリメンタルサーチを行ってくれる。キーワードにマッチするコマンド履歴が複数ある場合はさらに［Ctrl］＋［r］を押すことで、さらに後方にさかのぼってマッチするコマンドを呼び出すことが可能だ。また［Ctrl］＋［s］を押せば、前方に対して検索することができる。後方に行きすぎてしまった場合に使ってみよう。

端末上で［Ctrl］＋［s］を入力しても、前方検索が行われないことがある。これは「stty -a」コマンドを実行するとわかるが、［Ctrl］＋［s］が「端末の停止」に割り当てられているのが原因だ。このような場合は次のコマンドで、［Ctrl］＋［s］での端末の停止を無効にできる。

```
$ stty stop undef
```

入力履歴を検索しよう！

```
$ sudo apt-get update
$ ls /usr/bin
$ apt-cache stats
$ echo "hoge"（［Ctrl］＋［r］を押す）
(reverse-i-search)`apt':（プロンプトが変化するのでaptと入力）
(reverse-i-search)`apt': apt-cache stats
aptにマッチする、一番最近のコマンドが検索された
↓
（さらに［Ctrl］＋［r］を押す）
(reverse-i-search)`apt': sudo apt-get update
aptにマッチする2番目に新しいコマンドが検索された]
```

ディレクトリ移動もラクラク入力

ディレクトリスタックを活用してみよう

端末で実際に作業してみると、ひんぱんに「cd」コマンドを使う。しかし、毎回ディレクトリを指定するのは面倒だ。2つのディレクトリ、たとえば「~/」と「~/Desktop」の間などは、何度も往復することもある。そんな「直前のディレクトリに移動する」ための機能が、実は「cd」コマンドにある。「cd」コマンドの引数として「-（ハイフン）」を渡すと、そのシェルで直前にいたディレクトリに移動できるのだ。

これとは別に、ディレクトリスタックを用いてディレクトリを移動する「pushd」と「popd」コマンドを使う方法もある。

スタックとは、データを下から上へ順に積み上げて行き、データを取り出す際には上から順に取り出す、後入れ先出しのデータ構造。使い方は簡単で、「cd」コマンドの代わりに「pushd」コマンドでディレクトリを移動するだけだ。すると、移動前のディレクトリがスタックの一番上に保存される。その後「popd」を実行すると、スタックの一番上にあるディレクトリへ移動できる。深い階層のディレクトリをあちこち移動するような場合に効果的だ。

pushdでディレクトリ移動

```
$ pwd
/home/mizuno  ←カレントはホームディレクトリ
$ pushd /←pushdで「/」に移動
$ pushd /etc←pushdで「/etc」に移動
$ dirs
/etc / ~  ←「dirs」コマンドでスタックを表示できる
$ popd ←popdを実行
$ pwd
/  ←カレントが一つ前の「/」に移った
$ popd ←popdを実行
$ pwd
/home/mizuno  ←さらに一つ前のホームディレクトリに移った
```

cdで直前のディレクトリに移動

```
$ pwd
/home/mizuno  ←カレントはホームディレクトリ
$ cd Desktop ←デスクトップへ移動
$ pwd
/home/mizuno/Desktop ←カレントはデスクトップ
$ cd - ←直前のディレクトリへ移動
$ pwd
/home/mizuno  ←カレントはホームディレクトリ
$ cd - ←直前のディレクトリへ移動
$ pwd
/home/mizuno/Desktop  ←またデスクトップへ移動できた!
```

コマンドなんて not scary!! 怖くない!!

コマンドを組み合わせて複雑な処理をさせる！

パイプとリダイレクトを使って組み合わせる

コマンドの実行結果をファイルにしてみよう

一般的なUnixのコマンドは、標準入力からデータを受け取り、結果を標準出力に、エラーを標準エラー出力に出力するように作られている。

ユーザがふだんPCを使っているのと同様に、デフォルトの標準入力はキーボード、標準出力と標準エラー出力はディスプレイだが、これらは「パイプ」や「リダイレクト」と呼ばれる機能を利用して変更できる。この機能を利用すると、本来ならばディスプレイに出力されるコマンドの実行結果をファイルに書き出したり、別のコマンドに直接渡したりできるようになる。Unixには単純な機能のみを持ったコマンドが無数に用意されていて、パイプやリダイレクトを利用してコマンドを組み合わせれば、柔軟かつ複雑な処理をたった1行のコマンド入力で実行することもできるのだ。

ファイルのリストを表示する「ls」コマンドを例にとってみてみよう。

```
$ ls /usr/bin
```

これで/usr/binディレクトリに存在するファイルのリストが、ディスプレイ(標準出力)に出力される。今度は、このコマンドの出力をファイルにしてみよう。

```
$ ls /usr/bin > ~/usr.bin.list
```

ここでは、「/usr/bin」ディレクトリに存在するファイルのリストを、「~/usr.bin.list」というファイルに出力するように指示している。「>」が標準出力を切り替える指示で、この例は「標準出力をファイルにリダイレクト」していることになる。なおリダイレクトには「>」と「>>」という2種類の指示があり、前者を使用した場合は出力先のファイルを上書きし、後者を使用した場合はファイルの末尾に出力が追記されるということも、あわせて覚えておこう。次はちょっとおもしろい例だ。

```
$ ls /usr/bin > /dev/null 2>&1
```

「/usr/bin」ディレクトリに存在するファイルのリストと、エラー出力を「/dev/null」に出力する。リダイレクトの指示にある「1」や「2」という数字は、1が標準出力、2が標準エラー出力を表しており、「2>&1」は「2の出力先を1と同じにする」という指示だ。そして出力先の「/dev/null」は特別なファイルで、ここに出力されたデータはすべて捨てられてしまう。つまりこのコマンドを実行しても何も表示されない。「コマンドは実行したいが、記録はいらない。エラー出力もさせたくない」というような場合に利用されるテクニックだ。

出力を別のコマンドに渡してみよう！

今度は2つのコマンドを組み合わせて実行させてみよう。

```
$ ls /usr/bin | head
```

これは「/usr/bin」ディレクトリに存在するファイルのリストを、「head」コマンドに渡すという意味。コマンドの出力を他のコマンドの入力につなぐ「|」を「パイプ」と呼ぶ。「head」コマンドによって、「ls」コマンドの出力のうち、先頭の10行のみがディスプレイに表示されるはずだ。

```
$ ls /usr/bin | wc
```

「/usr/bin」ディレクトリに存在するファイルのリストを、「wc」コマンドに渡す。「wc」によって「ls」コマンドの出力がカウントされ、出力された行数、単語数、バイト数がディスプレイに出力される。

複数のパイプでコマンドをつなぐ

もちろん複数のコマンドをつなぐことも複数のパイプでつなぐことも可能だ。たとえば「sort」コマンドでソートしたファイルから「uniq」コマンドで重複を削除し、「grep」コマンドで対象を絞り込んだものを「less」コマンドで表示するなどは、常套手段だ。このように入力データに何らかの加工をして出力するコマンドは「フィルタ」と呼ばれ、フィルタを何重にもつなぎ合わせて処理を行うことは珍しくない。

複数のファイルを検索・指定するには

ワイルドカードを使ってファイルを指定

ワイルドカードとは「任意の文字列にマッチする」特殊な文字のことだ。古いPCユーザなら、MS-DOSで使った経験があるかもしれない。例えばファイル名を指定する際にワイルドカードを使うことで、特定のパターンをもった複数のファイルをまとめて指定することが出来る。

以下の例では「ls」コマンドに「/usr/bin/apt-*」というパラメータを与えているが、これは「/usr/bin」ディレクトリに存在する、apt-ではじまり、その後に任意の文字列が続くファイル」を意味している。「*（アスタリスク）」が「任意の文字列」を表すわけだ。次のようにすると、拡張子「.bak」を持つファイルを全て削除できる。ワイルドカードは大量のファイルを一括で指定したい場合に役に立つ。是非覚えておきたい機能だ。

```
$ rm *.bak
```

また、アスタリスクは任意の長さの文字列を表すが、「?(クエスチョンマーク)」は任意の1文字を表すことができる。左の例では「ls」コマンドのパラメータに「/usr/bin/apt-???」を与えているが、これは「apt-」の後に任意の「3文字」が続くファイルを意味している。アスタリスクは任意の長さ(0文字も含む)を表すが、クエスチョンマークは文字数を厳密に指定できるところがポイントだ。

このようにワイルドカードは非常に強力な機能だが、ひとつ操作を間違うと、全てのファイルを削除してしまうような事故につながることも多い。ファイル削除の「rm」コマンドに確認を促す「-i」オプションをつけるなど、くれぐれも注意してほしい。またネットにはホームディレクトリのファイルをすべて削除する「rm -rf ~/*」のようなコマンドが書き込まれていたりするが、いたずらなので絶対に実行してはいけない。

apt-で始まる名前をもつファイルをリストする例

```
$ ls /usr/bin/apt-*
/usr/bin/apt-cache              /usr/bin/apt-file
/usr/bin/apt-mark
/usr/bin/apt-cdrom              /usr/bin/apt-
ftparchive  /usr/bin/apt-sortpkgs
/usr/bin/apt-config             /usr/bin/apt-get
/usr/bin/apt-extracttemplates   /usr/bin/apt-key
```

apt-で始まり、ハイフンの後に3文字の名前をもつファイルをリスト

```
$ ls /usr/bin/apt-???
/usr/bin/apt-get  /usr/bin/apt-key
```

シェルスクリプトってなんだ!?

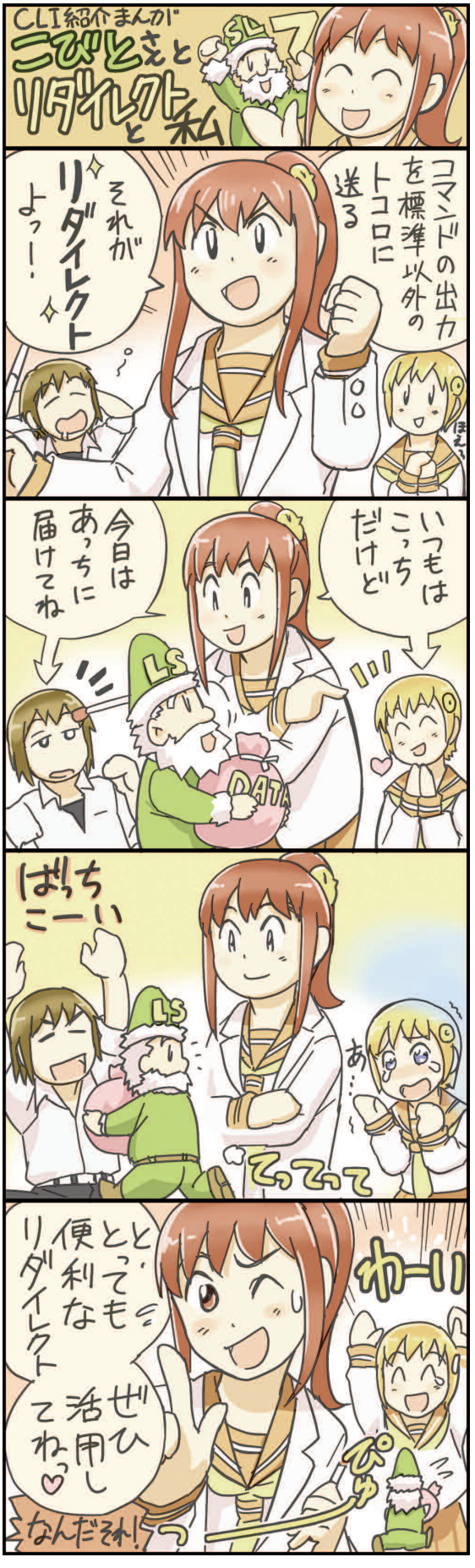
シェルを使ったプログラミング

シェルスクリプトとは、簡単に言えばWindowsのバッチファイルのようなもので、複数のコマンドをまとめて実行するための手続きを記述したファイルのことだ。さらにシェルが持つ「条件判定」「ループ」といった機能を用いて、複雑なプログラミングもできる。事前にスクリプトを作成しておけば、どんなに複雑で大量のコマンド処理でも、スクリプトを起動するだけで実行できる、CLIの真骨頂とも呼べる機能だ。

とは言っても、なんのことやらチンプンカンプンかもしれない。参考に日本語Remixに含まれる「日本語環境セットアップ・ヘルパ」を見てみよう。GUIでインストールするアプリを選択できる便利なツールだが、実はセットアップ・ヘルパの実体はシェルスクリプトで構成されているのだ。「/usr/bin/ubuntu-ja-setup-helper」を、ぜひ「less」や「cat」コマンドで見てみよう。シェルスクリプトでできることがイメージできるはずだ。次回はこういったシェルスクリプトや正規表現を中心とした使いこなしテクニックを紹介していこう。

実はシェルスクリプトなのだ

▲シェルスクリプトとzenityコマンドを組み合わせて、GUI操作を実現しているのだ。



シェルを変更して端末をもっと便利に!!

Ubuntuをはじめ最近のLinuxではGUIが標準的に使えるので、GNOME端末のような端末エミュレータのウィンドウを複数同時に開いて作業することも珍しくないだろう。その際に問題となるのが、端末間での履歴の共有だ。前述の通り「bash」はコマンドの履歴を記録しているが、複数の端末を同時に起動している場合、端末Aで実行したコマンドを、端末Bのヒストリから呼び出すことは標準ではできない。

また、bashのコマンドヒストリはホームディレクトリの「.bash_history」というファイルに記録されているが、このファイルはシェルの終了時に「上書き」されることになっている。つまり最後に閉じた端末以外の履歴は失われてしまうのだ。

しかし、zshというシェルには端末間で履歴を共有する機能が搭載されているため、この問題はシェルをzshに変更することで簡単に解決できる。まずzshパッケージをインストールしたら、zshの設定ファイルである「~/.zshrc」に「share_history」オプションを記述しよう。zshを起動するには、端末から

```
$ zsh
```

を実行すればよい。これ以降zsh上で実行したコマンドの履歴は、全てのzsh上で共通して呼び出せるようになるはずだ。

もしデフォルトのシェルをbashからzshに変更したいなら、「chsh」コマンドを実行してログインシェルに「/usr/bin/zsh」を指定するといいだろう。GNOME端末を起動した際に、デフォルトのbashにかわってzshが実行されるようになる。

zshには履歴の共有機能だけでなく、ファイル名のワイルドカード展開、複数行コマンドの編集機能など、ここでは説明できないほど多くの機能が用意されている。

[zshのインストール]

```
$ sudo apt-get install zsh
```

[.zshrcの設定]

```
setopt share_history
```

[ログインシェルの変更]

```
$ chsh
Login Shell [/bin/bash]:
/usr/bin/zsh
```

よく使われるコマンドを知ろう!!

Ubuntuを使う上で、ぜひ知っておきたいコマンドを紹介しよう。ただしコマンドといっても膨大な数があるため、この誌面ですべてを説明するのは不可能だ。そのため、ファイル処理を中心に代表的なコマンドを厳選し、概略のみを載せるだけにとどめている。

grep テキスト操作

パターンにマッチする行を検索する

テキストファイルから、任意の文字列を探し出し、該当部分を表示する。検索には文字列だけでなく、正規表現を使うことも可能。人が書いたプログラムを読むときなどには、必須のコマンドだ。

diff テキスト操作

2つのファイル間の相違点を表示する

2つのファイルを比較し、相違点を表示することができる。diffコマンドで出力した差分は、patchコマンドでファイルに適用することができるので、ソフトウェア開発においてよく利用されている。

cat テキスト操作

テキストファイルの内容を出力する

ファイルの内容を、標準出力に出力する。また、複数のファイルの結合も行える。Unixはデバイスもファイルとして扱うため、「ディスク上のファイル」だけでなく、たとえばビデオキャプチャボードの出力なども扱えたりする。

uniq テキスト操作

テキストファイルから重複した行を削除

テキストファイルから重複する行を削除することができる。ただし、対象となるファイルはあらかじめソート済みである必要があるため注意が必要だ。sortコマンドとパイプで繋いで使用するのが一般的だろう。

tail テキスト操作

テキストファイルの末尾部分を表示する

headコマンドの逆で、テキストファイルの末尾の数行（デフォルトでは10行）を表示する。また-fオプションを使うと、ファイルに追記される文字列を監視し続けることができる。ログファイルの監視などによく使われる。

sort テキスト操作

テキストファイルを行ごとにソートする

テキストファイルを行ごとにソートする。また、ソート済みかのチェックや、複数のファイルをソートして一つのファイルにマージすることも出来る。データの集計などに役立つコマンドだ。

less テキスト操作

テキストファイルをページごとに表示

テキストファイルを表示するが、catコマンドのように垂れ流すのではなく、前後に移動したり、検索を行ったりという操作が可能なビューアを起動する。テキストを「読む」時に使用するとよいだろう。

head テキスト操作

テキストファイルの先頭部分を表示する

テキストファイルの先頭の数行（デフォルトでは10行）を表示する。catコマンドでは全体を表示してしまうため、巨大なファイルの先頭だけを見たいような場合に有効なコマンドだ。

cp ファイル操作

ファイルのコピーを実行する

ファイルやディレクトリのコピーを行う。ディレクトリを再帰的にコピーするためには、-Rオプションが必要になる。また複数のファイルを一つのディレクトリにまとめてコピーすることも可能だ。

chown ファイル操作

ファイルの所有者を変更する

ファイルを所有するユーザ、およびグループを変更する。-Rオプションを併用すれば、ディレクトリを下位にあるディレクトリやファイルにも変更を適用できる（これを「再帰的に適用する」という）。

chmod ファイル操作

アクセス権限を変更する

ファイルの「持ち主」「グループ」「その他全員」それぞれに対する、ファイルへのアクセス権限を設定する。アクセス権限はパーミッションのビットパターンを8進数で表すか、変更方法をシンボルで表すことが可能。

cd ファイル操作

カレントディレクトリを変更する

指定されたディレクトリに、カレントディレクトリを変更する。ディレクトリが指定されなかった場合は、カレントディレクトリをユーザのホームディレクトリに変更する。

wc テキスト操作

テキストのバイト数、単語数、行数を計算

テキストファイルから、スペースで区切られた単語数や行数を計算できるが、日本語は単語がスペースで区切られていないため、使いづらい。コマンドの出力をパイプで渡して、行数を数えるといった用途が多くなるだろう。

ls ファイル操作

ファイルやディレクトリの一覧を表示する

任意のディレクトリにある、ファイルやディレクトリをリストして表示する。指定がない場合は、カレントディレクトリが対象となる。タイムスタンプでソートしたり、サブディレクトリの中身を再帰的に表示することもできる。

ln ファイル操作

ファイルやディレクトリへのリンクを作成する

指定したファイルやディレクトリへのリンクを作成する。デフォルトではハードリンクが作成されるが、-sオプションを併用することでシンボリックリンクを作成することができる。

find ファイル操作

条件にマッチするファイルを検索する

引数に与えられたディレクトリ以下から、条件にマッチするファイルやディレクトリを検索する。条件にはファイルタイプや名前、更新時刻などが指定できる。また見つかったファイルに対し、別のコマンドを適用できる。

file ファイル操作

ファイルのタイプを判定する

fileコマンドは、対象のファイルタイプを判定することができる。バイナリのプログラムをテキストのつもりでcatしてしまったりすると面倒だ。何のファイルか解らない場合は、fileコマンドで調べてみよう。

dd ファイル操作

フォーマット変換とコピーを行う

ファイルを指定したブロックサイズでコピーする。HDD全体をバックアップしたり、逆にHDD全体を書きつぶしたり、または仮想マシンのディスクイメージのような巨大なファイルを作成したりといった用途によく使われる。

rmdir ファイル操作

ディレクトリを削除する

指定されたディレクトリを削除する。ただし対象となるディレクトリは、中身が空でなければならない。mkdirコマンドと同じく-pオプションを併用することで、深い階層のディレクトリを親ディレクトリごと削除もできる。

rm ファイル操作

ファイルを削除する

指定されたファイルを削除する。デフォルトではディレクトリの削除は行わない。-rオプションを併用することでディレクトリを再帰的に削除することもできるが、大変危険であるため注意して使用しよう。

pwd ファイル操作

カレントディレクトリを絶対パスで表示する

現在のカレントディレクトリを表示することができる。pwdコマンドには、シェルの組み込み版とGNU版(/bin/pwd)があり、通常使用されるのは組み込み版である。こちらはシンボリックリンクを展開しないため注意が必要だ。

mv ファイル操作

ファイルを移動、またはリネームする

ファイルやディレクトリを別の場所へ移動する。cpと同様、複数のファイルを同じディレクトリにまとめて移動することも可能だ。また、「同じ場所へ別の名前で移動する」ことによって、ファイルをリネームすることができる。

mkdir ファイル操作

ディレクトリを作成する

指定した名前でディレクトリを新規作成する。-pオプションを併用すると、深い階層のディレクトリを親ディレクトリごとまとめて作成することができる。また、作成時にパーミッションの指定もできる。

はじめてのコマンド&端末入門

kill
ジョブ/プロセス管理

プロセスにシグナルを送信する

システム上で動作しているプロセスに任意のシグナルを送信する。デフォルトのシグナルはTERMで、プロセスを正常に終了させることを意味するため、プロセスを外部から終了させるためによく使用されるコマンドだ。

jobs
ジョブ/プロセス管理

ジョブの一覧を表示する

バックグラウンドで実行中のジョブや、サスペンド中のジョブの一覧をジョブ番号とともに表示する。カレントジョブには「+」、直前のジョブには「-」という記号が表示される。

fg
ジョブ/プロセス管理

ジョブをフォアグラウンドで実行

サスペンド中のジョブをフォアグラウンドで再開する。ジョブ番号を指定しなかった場合はカレントジョブが対象になる。なおシェルから単にジョブ番号を入力する(例:%1など)と「fg %1」を実行したのと同じ効果が得られる。

bg
ジョブ/プロセス管理

ジョブをバックグラウンドで実行

サスペンド中のジョブをバックグラウンドで再開する。ジョブ番号を指定しなかった場合はカレントジョブが対象になる。バックグラウンドで再開したジョブは、「&」をつけて起動した時と同様なふるまいをする。

tar
ファイル操作

ファイルの圧縮や展開を行う

Unixの一般的な圧縮ファイルである、tar.gzの作成や展開を行う。本来tarは複数のファイルをまとめてtar形式にするだけで、それを別途gzipコマンドで圧縮する必要がある。だがGNU版のtarはこの作業を同時に行える。

tee
シェル関連

標準入力を標準出力とファイルに出力

teeコマンドにパイプで出力を渡すと、標準出力とファイルの両方に同じ内容を出力できる。単にファイルにリダイレクトすると標準出力に結果が表示されないため、ディスプレイにも表示したい時に活用しよう。

history
シェル関連

コマンド入力の履歴を表示する

コマンドの入力履歴に、行番号をつけて表示する。また、履歴の削除や編集、追加、再読込を明示的に行うこともできる。この機能を応用することで、zshのコマンド履歴共有をbashで実現することも可能だ。

alias
シェル関連

コマンドに別名を割り当てる

任意のコマンドに対し、別名を割り当てることができる。たとえば「$ alias ll='ls -l'」とすると、以降、「$ ll」で-lオプションつきのlsを実行できる。引数なしで実行すると、定義済みのエイリアスの一覧を表示できる。

top
ジョブ/プロセス管理

稼働中のシステム状況を表示する

システムの状況をリアルタイムに表示し続ける。システムの稼働時間やロードアベレージ、CPU使用率といった情報に加え、動作中の各プロセス情報を、CPU負荷やメモリ使用量などの項目ごとにソートして表示できる。

ps
ジョブ/プロセス管理

プロセス情報を表示する

現在動作中のプロセスの情報を取得する。psコマンドは複数のオプション形式をもっており、混在させて使うことはできないので注意。例えば動作中の全プロセス情報を得るには「$ ps aux」か「$ ps -fe」となる。

fsck
ディスク管理

ファイルシステムの検査と修復を行う

ファイルシステムの状況を検査し、必要であれば修復を行うためのコマンド。マウント中のディスクに対してfsckすることはできないので、必要であればLive CDから起動して行おう。

fdisk
ディスク管理

パーティションテーブルを操作する

基本パーティション、拡張パーティション、論理ドライブの作成や削除、さらにパーティションタイプの変更などを行う。ただし、GPTのパーティションには対応していない。

du
ディスク管理

ファイルのディスク使用量を表示する

引数で指定したファイルのディスク使用量を表示する。ファイルサイズと「ファイルが実際に消費しているディスク容量」は異なるため、後者を知りたい場合はduコマンドが必要だ。

df
ディスク管理

ファイルシステムの使用状況を表示する

マウントされているファイルシステムの使用状況を表示する。引数としてファイルを与えた場合、そのファイルが存在するファイルシステムのみの状況を表示できる。-hオプションを併用するとバイト単位での表示が可能。

watch
シェル関連

コマンドを一定時間ごとに実行する

引数として指定されたコマンドを一定時間ごとに実行し、その結果をフルスクリーンで表示する。例えばduコマンドと組み合わせてファイルの容量を監視する、などといった用途に使用できる。

passwd
ユーザと権限

ユーザパスワードを変更する

自分自身、あるいは任意のユーザのパスワードを変更する。自分のパスワードを変更するためには、現在のパスワードの入力が必要。また他人のパスワードを強制的に変更するためには、当然ながら管理者権限が必要だ。

last
ユーザと権限

最終ログイン情報を表示する

そのコンピュータにログインしたユーザ名や端末名、接続元のホスト名、ログイン時間といった情報を表示することができる。なおログイン情報の実体は「/var/log/wtmp」ファイルに記録されている。

id
ユーザと権限

ユーザIDと所属するグループIDを表示する

自分自身、または任意のユーザのユーザIDと、所属するグループIDを表示することができる。デフォルトではIDだけでなく、ユーザ名やグループ名といった文字列も合わせて表示することができる。

mount
ディスク管理

ファイルシステムをマウントする

引数に指定したデバイスをディレクトリツリーの任意の位置にマウントし、アクセス可能な状態にする。引数なしで実行した場合は、現在のマウント情報を表示することができる。

mkfs
ディスク管理

ファイルシステムを作成する

ハードディスクのパーティションに対し、ファイルシステムの作成を行う。ファイルシステムタイプを明示的に指定しなかった場合は、デフォルトであるext2ファイルシステムが作成される。

uname
その他

カーネルの情報を表示する

動作中のカーネルの情報を表示できる。オプションをつけずに実行すると、Linuxというカーネルの種類のみが表示されるが、-aオプションを併用することで、アーキテクチャやバージョン、ホスト名なども表示できる。

man
その他

オンラインマニュアルを表示する

引数としてコマンドや設定ファイルの名前を渡すことで、各種コマンドのオンラインマニュアルを表示させられる。もちろん最初にやることは「$ man man」でマニュアルのマニュアルを読むことなのは言うまでもない。

date
その他

日付の表示や設定を行う

現在日時を指定されたフォーマットで表示させることができる。また、システム時刻を任意の時刻に設定することができる。当然システム時刻を設定する場合は管理者権限が必要だ。

sudo
ユーザと権限

別のユーザ権限でコマンドを実行する

自分以外の、任意のユーザの権限でコマンドを実行することができる。-uオプションを省略するとrootが対象となり、Ubuntuではroot権限でコマンドを実行する際に利用するのが一般的。

su
ユーザと権限

別のユーザに切り替える

自分以外の、任意のユーザにスイッチすることが出来る。スイッチするユーザ名を省略した場合はrootが対象となり、一般的にはrootに昇格するための目的で使われることが多い。