UNIXのコマンドとEmacsの使い方

山本昌志*

2005年4月21日

1 UNIXのコマンド

1.1 使用頻度の高いコマンド

使用頻度の多いUNIXコマンドを以下に示す。これらのコマンドは、すべてターミナルから打ち込み、コンピューターに命令を与える。他に、ファイルマネージャーもあるので、組み合わせて使って欲しい。

表 1: UNIXでよく使われるコマンド。下の3つは、正確にはコマンドではないが、便利な機能である。

コマンド	機能
man	コマンドのオンラインマニュアル
pwd	現ディレクトリーの表示
ls	ファイルとディレクトリーの表示
cd	ワーキングディレクトリーの移動
mkdir	ディレクトリーの作成
rmdir	ディレクトリーの削除
cp	ファイルやディレクトリーの複製
mv	ファイルやディレクトリーの名前の変更や移動
rm	ファイルの削除
cat	ファイルの表示や連結
less	ファイルの内容を一画面単位で出力
↑又は↓	history　以前のコマンドの表示　編集可能
[Ctrl]+c	プロセスの強制終了
[Tab]	補完機能

*国立秋田工業高等専門学校　電気情報工学科

1.2 コマンドの動作

これから、コマンドの動作について記述するが、**形式**あるいは**機能・形式**のところのカギ括弧 [ ] 内は省略可能を示している。これはオプションなので、必要なときにカギ括弧内のそれを使えば良い。

1.2.1 コマンドの動作を調べる

指定されたコマンドのオンラインマニュアルを一画面分ずつ表示する。

コマンド	`man`
語源	**man**ual
形式	`man` *commandname*
注意	次の頁を見るときには f キーを、前の頁を見るときには b キーを押す。表示をやめるときには、q キーを押す。

1.2.2 現ディレクトリーの表示

自分が、今どこにいるか調べるコマンドである。

コマンド	`pwd`
語源	**p**rint **w**orking **d**irectory
機能	現在のワーキングディレクトリ（カレントディレクトリ）を絶対パス名で表示する。

1.2.3 ファイルとディレクトリーの表示

自分が、今いるディレクトリーの中にあるファイルやディレクトリーを調べるときに使うコマンドである。

コマンド　`ls`

語源　list

機能　現在のワーキングディレクトリのファイルやディレクトリーの情報を表示する。

形式　`ls [-adFgilostux] [`*filename*`···]`

オプション

なし　ファイル名のみ並べて出力する。

`-a`　.(ドット) で始まる隠しファイルも含めて、すべて出力する。

`-l`　ファイルの詳細管理情報をロング形式で出力する。

`-d`　filename がディレクトリの時、その名前のみ表示する。

`-F`　ファイルの種類を記号で表示する。

< 記号の意味 >

表示なし	プレーンなデータファイル
`/`	ディレクトリファイル
`*`	実行可能ファイル

`-I`　inode 番号を表示する。

`-R`　ディレクトリの階層構造を表示する。

`-s`　ファイルのサイズをプロック単位で表示する。

`-t`　最終更新時刻の新しいものから順に表示する。

`-u`　`l` オプションとの併用時、最終更新時刻の代わりに最終アクセス時刻を表示する。

`-o`　`l` オプションと同じだが、グループ名を表示しない。

`-x`　ファイル名を横に並べて出力する。

1.2.4　カレントディレクトリーの変更

今いるディレクトリーから他のディレクトリーに移るときに使うコマンドである。

コマンド　`cd`

語源　**c**hange **d**irectory

機能・形式　指定されたディレクトリーに移動

`cd` *directory*

ログインでレクトリーに移動

`cd`

親ディレクトリーに移動

`cd ..`

1.2.5 ディレクトリーの作成

ディレクトリを作成する。ディレクトリ自身を表す「.」と、親ディレクトリを表す「..」の2つが、自動的に作成される。

コマンド mkdir
語源 **m**a**k**e **dir**irectory
形式 mkdir *directory*　　指定されたディレクトリーを作成

1.2.6 ディレクトリーの削除

ディレクトリを削除する。

コマンド rmdir
語源 **rm**ove **dir**ectory
形式 rmdir *directory*
注意 削除するディレクトリは空でなければならない。ディレクトリ配下にファイルがある場合、配下のファイルごと削除するには rm コマンドを使う。いずれの場合も mkdir コマンド同様、親ディレクトリに書き込み権が必要。

1.2.7 ファイルやディレクトリーのコピー

ファイルやディレクトリーのコピーを作成したい場合に使う。

コマンド cp
語源 **c**o**p**y
機能・形式 *filename1* を *filename2* という名前でコピーを作成する。

cp [-ip] *filename1 filename2*

directory2 の配下に *directory1* をサブディレクトリとして、配下のファイルごとコピーする。

cp -r[-ip] *directory1 directory2*

各 *filename*（サブディレクトリ指定可）を、最後に指定した *directory* 配下にコピーする。

cp -r[-ipr] *filename*··· *directory*

オプション

-I　コピー先ファイルが既存の場合、置き換えを行うかどうか確認してくる。(置き換える場合=y　、置き換えない場合=n と入力する）
-p　内容だけでなく、最終修正時刻・アクセス許可もコピーする。
-r　ディレクトリ配下のファイルごとコピーする。（ファイルがサブディレクトリの場合は、その配下のファイルごとコピーする。）

1.2.8 ディレクトリー・ファイルの移動と名前の変更

ディレクトリーやファイルを移動させるときに使う。また、名前を変えるときにも使う。

コマンド mv

語源 **m**ove

機能・形式 *filename1* を *filename2* に名前を変える。

mv [-i] *filename1 filename2*

directory1 を *directory2* に名前を変える。*directory2* が既存の場合、*directory2* の配下に *directory1* を移動する。

mv [-i] *directory1 directory2*

各 *filename*（サブディレクトリ指定可）を、最後に指定した *directory* 配下に移動する。

mv [-i] *filename* ··· *directory*

オプション

-i 移動先ファイルが既存の場合、置き換えを行うかどうか確認してくる。置き換える場合=y、置き換えない場合=n と入力する。

1.2.9 ファイルの削除

ファイルを削除するときに使う。また、オプションを付けて、ディレクトリーごと削除することもできる。

コマンド rm

語源 re **m**ove

機能・形式 ディレクトリから各 *filename* を削除する。

rm [-i] *filename1*

directory 配下のファイルから順に削除していき、*directory* 自身も削除する。

オプション

rm -r[i] *directory*

-i 削除していいかどうか確認のメッセージを出す。削除する場合=y、削除しない場合=n と入力する。

-r *directory* 配下のファイルを削除し、*directory* 自身も削除する。

-f ファイルを強制的に削除する。

1.2.10 ファイルの表示や連結

ファイルの連結と表示を行う。指定したファイルを順次読み取り、標準出力に出力する。この機能を利用して、複数ファイルの結合（concatenate 鎖状につなぐ）を行うことができる。次の例は、file1 と file2 を結合して、file3 を作る。

```
cat file1 file2 > file3
```

>はリダイレクトと呼ばれる機能である。

コマンド　cat
語源　con**cat**enate
形式　cat [filename···]
注意　cat file1 file2 > file1 および cat file1 file2 > file2 は、まず出力領域を確保しようとするために、読み取り前に入力データが破壊されてしまうので要注意。また、cat ではデータの出力は画面単位ではない。一画面分以上の大きさのファイルを表示しようとすると先頭部分はスクロールして見えなくなってしまう。その場合は less コマンドを使用する。

1.2.11　ファイルの内容を一画面単位で出力

ファイルの内容を見るときに使う。

コマンド　less
語源　**less**
形式　less *filename*
注意　次の頁を見るときには f キーを、前の頁を見るときには b キーを押す。表示をやめるときには、q キーを押す。

1.2.12　history　以前のコマンドの表示

前に入力したコマンドを確認したり、取り出して再実行、編集して実行する。キーを叩く回数が減り、タイプミスを減らせるので、かなり便利である。というか、ほとんどの場合、これを使うことで済むので、作業が軽減される。

使用方法　↑と↓

1.2.13　プロセスの強制終了

ターミナルで実行したプロセスを強制的に終了させる。プログラムが、暴走したときに使う。

使用方法　[Ctrl]+c
注意　[Ctrl] キーを押したまま、c キーを押す。

1.2.14　補完機能

補完機能は、最初の何文字かを入力して [Tab] キーを押すと、自動的にディレクトリ名やファイル名、あるいはコマンドを表示してくれる機能である。キーを叩く回数が減り、タイプミスを減らすことができる。

使用方法　[Tab]

2　プログラムの作成と実行

2.1　一連の流れ

それでは、実際に、プログラムを作成して、実行させて見ましょう。初心者が最初に作るものとして、最も有名な'Hello World' というプログラムを作ります。

1. ターミナルの起動
   - ターミナル (端末) のアイコンをクリックして、ターミナルを立ち上げる。
2. 作業ディレクトリーの作成と移動
   - 「`mkdir hello`」とタイプして、作業用ディレクトリー `hello` を作る。
   - 作業用ディレクトリーができているか、「`ls`」コマンドで確認します。すると「`hello`」が表示される。
   - 「`cd hello`」とタイプして、作業用ディレクトリーに移動する。
   - 「`pwd`」コマンドで、作業用ディレクトリーに移れたことを確認する。
3. エディターの起動
   - 「`emacs hello_world.c&`」とタイプする。
   - すると、プログラムのソースを書くウインドウが現れる。
4. プログラムの記述
   - エディターのウィンドウに以下のプログラムを書きましょう。

```
#include <stdio.h>

int main(){

   printf("Hello World !!\n");

   return 0;
}
```

5. プログラムの保管
   - プログラムを書き終わったならば、[file] メニューの [Save(current buffer)] を選択する。あるいは、フロッピーディスクアイコンをクリックして、ソースファイルを保管する。
   - ターミナル上で「`ls`」コマンドを打ち、ソースファイルが保管されていることを確認する。
6. コンパイル
   - ターミナル上で「`gcc -o aisatsu hello_world.c`」と打ち込み、先ほど作成したソースファイルをコンパイルする。

- もし、コンパイルエラーが発生したら、ソースファイルを修正する。
- 実行ファイルができているか、「ls」コマンドで確認する。

7. 実行

- ターミナル上で「./aisatsu」と打ち込み、プログラムを実行させます。
- 'Hello World' と表示されれば、プログラムは動作は完璧です。

以上のプログラム作成の手順をまとめると、図 1 のようなフローチャートになる。

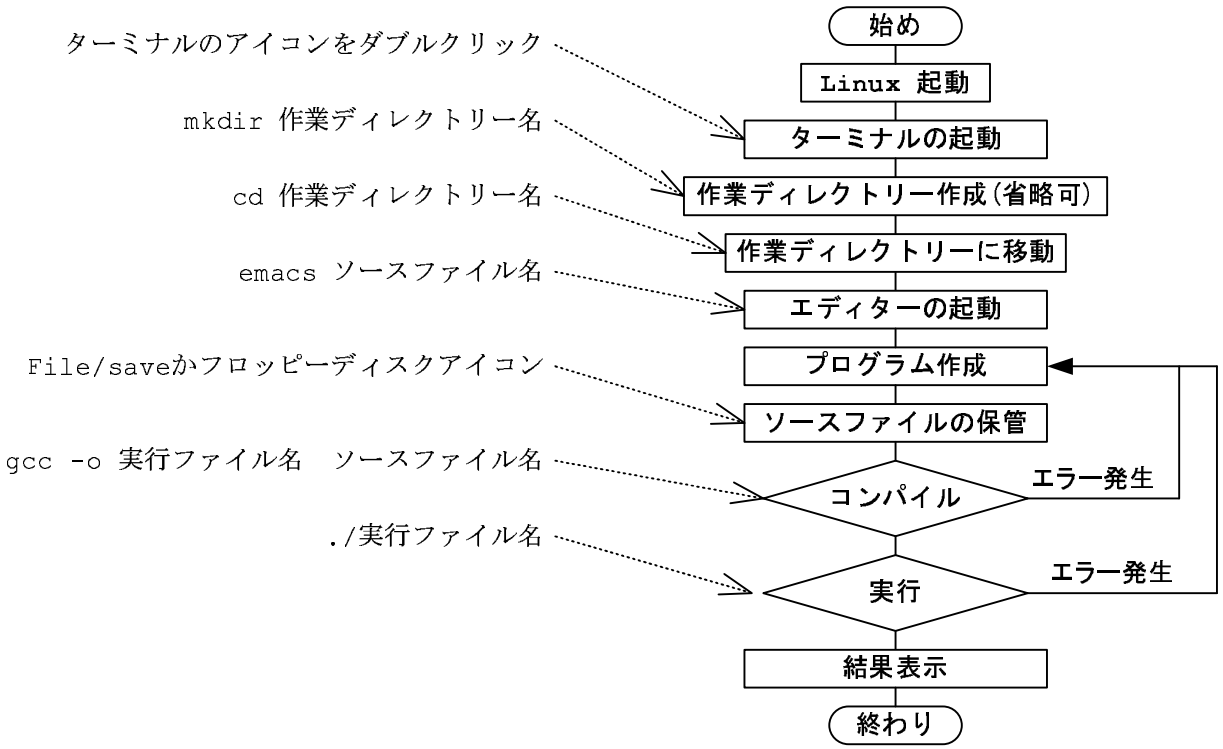

図 1: プログラムの作成のフローチャート

2.2 コンパイルとは

プログラムを作成する場合、コンパイル[1](gcc コマンド) という不思議なステップがある。このステップについて簡単に説明しておく。

一般に、コンパイルというと、ソースファイル (ここでは C 言語) から実行ファイルを作る動作のことを指す。ターミナル上で、

```
gcc -o 実行ファイル名 ソースファイル名
```

とタイプすれば、ソースファイルから実行ファイルができあがる。エラーが無ければちゃんと実行可能なファイルが作成され、「./実行ファイル名」とタイプすればプログラムムが実行できる。

さて、この動作がなぜ必要なのであろうか?。諸君が学習している C 言語のソースファイルの内容は人間が理解できるが、それはコンピューターは理解できない。そのため、人間が理解できる言葉からコンピューターが理解可能な言葉に翻訳が必要で、コンパイルとはそのために必要なのである。要するに、プログラミング言語を機械語に翻訳しているのである。

以上の説明で大体良いが、正確には、機械語に翻訳するためには、コンパイルとリンクと言う作業が必要である。それらの作業は、コンパイラーとリンカーが受け持っており、コマンド「gcc」の作業は、図 2 のようになっている。通常、コンパイルと言っているが、実際にはコンパイルとリンクを行っている。

余談ではあるが、オプションの「-o」を付けない、すなわち実行ファイル名無しで「gcc」を動作させた場合、「a.out」という実行ファイルができあがる。この a は、assembler out の略らしいが、実際には linker out である。

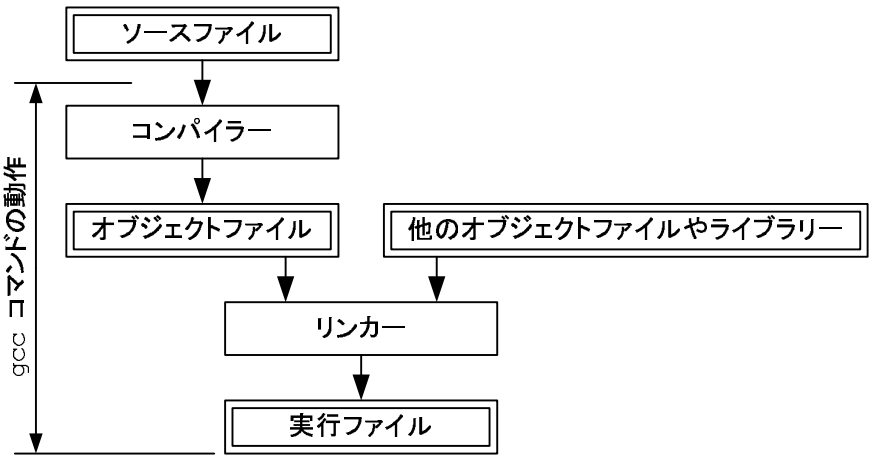

図 2: gcc コマンドの作業内容

[1]compile:収集する。編集する。機械語に翻訳する。

3 Emacsの使い方

3.1 Emacsとは

Emacs[2]はエディターで、テキストファイルを作成するソフトウェアーである。これの拡張機能は強力で、web閲覧、メール操作、ファイル操作までできほとんど作業環境と言うこともできる。私の場合は、この講義ノート作成にこのEmacsを用いている。諸君はプログラム作成のエディターとして用いることになる。

たいがいのUNIXには、Emacsは標準でインストールされているので、使い方を覚えておくと将来役に立つであろう。また、UNIXに限らず、windowsでもEmacsもどきがあるので気に入ったら使うとよいだろう。

このEmacsは、Richard Stallmanにより開発が始められた。かれは、著名なハッカーで魅力的な人に思えるので、興味のある人は調べてみると良い。

3.2 実行方法

端末(ターミナル)上で、「`emacs`」と打ち込めば、エディターが立ち上がり操作可能となる。しかし、このようにすると、後でファイル名を指定する必要が生じ面倒なので、通常は、ファイル名(例`hogehoge.c`)をつけて、

```
emacs hogehoge.c&
```

のようにする。こうすると、カレントディレクトリーに「`hogehoge.c`」の有無により次の動作を行う。

- ファイルが無い場合には、新規にファイルを作成する。
- ファイルが既にある場合には、そのファイルの編集モードに入る。

ファイル名の後の「&」は、Emacsをバックグラウンドで動作させるということを示している。こうすると、端末から次の命令を打ち込むことができて便利である。例えば、編集したソースファイルを同じ端末でコンパイルすることができる。

[2] イーマックスと読む

3.3 編集機能

表 2 に Emacs の操作を示すが、そのキー操作は次の約束に従う。

- 表中の C-x は、Ctrl キーを押したまま x キーをタイプする。
- 表中の Esc-x は、Esc キーを押し、離してから x キーをタイプする。

これを忘れないで、表の機能を使って、効率よくプログラムを作成しよう。

表 2: Emacs の基本的なキー操作

キー操作	機能
C-g	現在実行中のコマンドを中断
C-k	カーソルの位置から行末までカット。カットされた部分は、C-y でペーストできる。
C-Space	現在の位置を領域の開始としてマークする。後は、上下左右の矢印でカーソルを移動して、領域を決める。そして、Esc-w や C-w を使う。
Esc-w	領域をコピーする。コピーされた部分は、C-y でペーストできる。
C-w	領域をカットする。カットされた部分は、C-y でペーストできる。
C-y	コピーやカットした部分をペースト (張り付け) する。
C-s	順方向にサーチを開始する。
C-r	逆方向にサーチを開始する。
Esc-x replace-string	文字列を置換する。