# 第 1 回 UNIX ゼミ

ゼミ担当者 ： 河本 敏孝，梶原 広輝，濱地 優希
指導院生 ： 谷口 義樹，折戸 俊彦
開催日 ： 2004 年 4 月 27 日

**ゼミ内容:** 本ゼミでは，研究活動での UNIX の使用において，最低限必要となる知識および操作のスキル取得を目的とする．本研究室では最適化の研究に並列計算機を用いることが多いが，その際に，並列計算機の利用および管理のために Linux の知識が必要不可欠となる．そこで，本ゼミではそれらの利用が可能となるように，Linux 上でディレクトリ操作，ファイル操作，パーミッション設定，エディタの利用方法などについて学ぶ．

## 1 UNIX(Linux) とは

### 1.1 UNIX とは

UNIX とは，1969 年に米国 AT & T Bell 研究所で Dennis Ritchie と Ken Thompson によって開発された OS である．後に，Ritchie が開発した C 言語によって 1972〜1974 年ごろに書き直された．UNIX はソースコードが比較的コンパクトであったのと，ライセンスが安価に配布されたために，大学や研究機関などを中心に普及していった．ちなみに現在，商標としての「UNIX」は，The Open Group が所有しており，一定の仕様を満たした OS のみが「UNIX」の名称を使うことができる．

### 1.2 Linux とは

Linux とは，1991 年にヘルシンキ大学の Linus B. Torvalds 氏によって開発された UNIX クローン (互換) の OS である．Linux は既存のオペレーティングシステムのコードを流用せず，なにもないところから書き起こされたものであり，GPL というライセンス体系に基づき，自由に改変，再配布を行うことができるようになっている．知的システムデザイン研究室で主に利用されているのは，Debian GNU/Linux というディストリビューションである．Linux と呼んでいるのはカーネル自体であり，そのカーネルは基本的にディストリビューションに依存しているわけではない．各ディストリビューションの違いは，日本語化されている度合いやインストーラの完成度，各ソフトの初期設定状態などにある．

### 1.3 Windows や MacOS との違い

UNIX 系 OS は，Windows や MacOS と異なる．これらの OS は根本的に，シングルユーザの OS であり，それに対して UNIX は，マルチユーザ，マルチタスクの OS である．UNIX においては同一のシステム上で同時に複数の人が作業可能な設計になっている．マルチユーザ，マルチタスクのシステムにはセキュリティの概念が必要になってくるが，セキュリティも過度に複雑にならないようシンプルさゆえに十分なセキュリティ管理のしやすさを UNIX では実現している．UNIX では GUI(Graphical User Interface) には X を利用しているが，X は UNIX にとって必要不可欠なものではないので，取り外すことができる．その場合には，CUI(Character User Interface) によるオペレーションを行う．ちなみに，MacOS X は内部構造に UNIX 系のシステムが組み込まれており，UNIX のディレクトリ構造やマルチユーザを実現している．

### 1.4 オープンソースとは

Linux のアプリケーションは，一般的に GPL というライセンスに基づいて無料で配布されているため，ユーザは Web から公開されたアプリケーションを持ってきて，自分の扱うマシンに組み込むことができる．このことは，Linux が発展していくのに大きな役割を果たしている．Windows や MacOS に関する無料のソフトウェアも存在するが，それらはフリーのソフトウェアと呼ばれるものであり，Linux が GPL により配布されるソフトウェアは，オープンソースソフトウェアである．オープンソースソフトウェアは，ソフトウェアと共にソフトウェアの設計図となるソースコードが公開されており，類似品を作成することや，そのソフトウェアで使われている技術を転用することが可能である．また，ソースが公開されているので，ユーザはソースを変更して自分の環境にあわせることが容易に可能である．GPL の主な特徴として，以下のようなものが挙げられる．

- ソフトウェアは必ずソースプログラムとともに配布，複製される．もしソースプログラムを付けずに配布する場合は，ソースプログラムを確実に入手できる手段を提供することが義務付けられる．
- ソフトウェアを，使用，複製，変更，配布したり，新しいフリーソフトウェアの一部として利用できる．
- 変更，改良されたソフトウェアは GPL に従って配布される．

- プログラムの全部あるいは一部を用いて作られたソフトウェアは GPL に従って配布される．
- 基本的に無保証であり，そのソフトウェアが原因でトラブルが生じても作者に責任はない．

## 2 UNIX 上での基本コマンド群

- whoami
  whoami と入力すると自分が誰なのかわかる．

```
hama@mikilab:~$ whoami
hama
```

- pwd
  pwd と入力すると自分が今いるディレクトリがわかる．

```
hama@mikilab:~$ pwd
/home/hama
```

- ls
  ls と入力すると，カレントディレクトリにどんなファイルがあるのかを知ることができる．

```
hama@mikilab:~$ ls
Maildir    foo    test.c    xyz.java
```

- mkdir
  ディレクトリを作成するコマンドである．作成したいディレクトリ名を入力する．カレントディレクトリに新しいディレクトリが作成される．

```
hama@mikilab:~$ mkdir sample
hama@mikilab:~$ ls
Maildir    foo    sample    test.c    xyz.java
```

- cd
  別のディレクトリに移動するコマンドである．「cd 移動したいディレクトリ名」と入力する．指定したディレクトリへ移動できる．移動したかどうかは pwd で確認できる．

```
hama@mikilab:~$ cd sample
hama@mikilab:~/sample$ pwd
/home/hama/sample
```

- rm
  「rm ファイル名」でファイルの削除ができる．「rm -r ディレクトリ名」でディレクトリの削除ができる．削除できたかどうかは ls コマンドで確認できる．

```
hama@mikilab:~$ ls
Maildir    foo    sample    test.c    xyz.java
hama@mikilab:~$ rm -r sample
hama@mikilab:~$ ls
Maildir    foo    test.c    xyz.java
```

  他に rmdir というディレクトリを削除するコマンドもあるが，このコマンドでは空のディレクトリのみ削除ができ，ファイルがディレクトリ内にあるときはこれらのファイルを削除した後でないとディレクトリは削除できない．

- more

  ファイルの１ページスクロール表示を行うコマンドである．「more 表示したいファイル名」というように入力すると表示することができる．Enter キーで１行ずつの表示が行え，Space キーで次の１ページの表示を行うことができる．また”q”を押すと処理を中断できる．more コマンドの他に高機能な less というコマンドもある．

```
hama@mikilab:~$ more xyz.java
import java.lang.*;
import java.io.*;
```

- cp

  ファイルのコピーをするコマンドである．「cp file1 file2」と入力すると，file1 を file2 にコピーすることができる．

```
hama@mikilab:~$ ls
Maildir    foo    test.c    xyz.java
hama@mikilab:~$ cp foo foo2
hama@mikilab:~$ ls
Maildir    foo    foo2    test.c    xyz.java
```

- mv

  ファイルを移動したり，ファイル名を変更するコマンドである．「mv test sample」と入力すると，test というファイル名を sample というファイル名に変更することができる．

```
hama@mikilab:~$ ls
Maildir    foo    foo2    test.c    xyz.java
hama@mikilab:~$ mv foo2 foo3
hama@mikilab:~$ ls
Maildir    foo    foo3    test.c    xyz.java
```

- ln -s

  ファイルにシンボリックリンクを貼るコマンドである.「ln -s foo bar」と入力すると foo というファイルへ bar というシンボリックリンクを貼ることができる.

```
hama@mikilab:~$ ls
Maildir    foo    foo3    test.c    xyz.java
hama@mikilab:~$ ln -s foo bar
```

# 3 ファイルディレクトリについて

## 3.1 ファイル情報の意味

ls コマンドで-l オプションをつけると各ファイルについて詳細な情報を見ることができる.

”ls -l”というコマンドを実行した場合，Fig.1 のような画面が表示される.

はじめの行の”合計 16”というのは一覧に表示されたファイルやディレクトリの合計ブロック数を表す.ブロック数というのは，ディスク中に占める割合で，ファイルサイズとしてはその容量を満たしていなくても，実際のディスク割り当てではそのよう領分の場所を占めていることになる.

それでは，表示の中の各意味について左側から順に説明する.

- ファイルの種類

  そのファイルがディレクトリ (d) か，シンボリックリンク (l)，それとも普通のファイル (-) かを表す.

- パーミッション

  ”r”が読み込み権，”w”が書き込み権，”x”が実行権を表す.これは，所有者，グループ，その他に分かれていて，それぞれに権利を設定することができる.この部分が，”-”の場合にはその権利が与えられていないことを意味する.また，通常のファイルとディレクトリでは多少意味が異なり，ディレクトリの場合は，読み込み権とはそのディレクトリの中にファイルを作成する権利であり，実行権とはそのディレクトリをカレントディレクトリとする権利である.

- ハードリンク数

  次の数字はハードリンク数と呼ばれるものである.これは，このファイルに対して何個のハードリンクされているかを表している.

- ファイルの所有者

  ファイルの所有者を表す.通常はファイルを作成したユーザになる.

- 所有グループ

  ファイルを所有しているグループを示す.通常はファイルを作成したユーザが属するグループになる.

- ファイルのサイズ

  バイト単位でのファイルのサイズを示す.

- 更新日時

  ファイルが更新された日時を表す.

- ファイルの名前

  ファイルの名前を表す.

## 3.2 所有権とパーミッション

### 3.2.1 所有権の設定方法

UNIX では，すべてのファイルについて，そのファイルを所有しているユーザとグループを設定できる.

ファイルの所有者を変更するには，chown コマンドを使用する.

```
chown 新しい所有者 変更するファイル
```

所有グループを変更するには，chgrp コマンドを使用する.

```
chgrp 新しい所有グループ 変更するファイル
```

## 3.3 パーミッションの設定方法

UNIX でファイルのパーミッションを設定するには，chmod では，ファイルのパーミッションを設定する場合に 2 つの方法がある.

一つ目の方法は，どのユーザに対してどのパーミッションを割り当てるということを文字で指定する方法である.この方法では，Table 1 に挙げられている文字を組み合わせることでパーミッションの設定を行う.たとえば，下記のコマンドで所有者以外の読み込みを禁止できる.

```
chmod go-r file
```

もう一つの方法は，8 進法を用いてパーミッションを一括設定する方法である.ファイルのパーミッションに

```
合計 16
-rwxr-xr-x  1 hama    mikilab     2998    Apr 15  22:58   a.out
lrwxrwxrwx  1 hama    mikilab        3    Apr 15  21:59   bar -> foo
-rw-------  1 hama    mikilab       23    Apr 15  23:49   foo
-rw-r--r--  1 hama    mikilab       63    Apr 15  22:03   hello.c
drwxr-xr-x  2 hama    mikilab     4096    Apr 16  09:25   testdir
```

Fig. 1　ディレクトリ内のファイル

Table 1　chmod で用いる文字

| | |
|---|---|
| 対象 | u(所有者), g(グループ), o(その他), a(全て) |
| 操作 | +(追加), -(削除), =(設定) |
| モード | r(読み込み), w(書き込み), x(実行) |

は，”rwx”の 3 種類がある．そこで，”rwx”を 2 進数に見立てる．たとえば，rw-の場合は，’-’ を 0，それ以外を 1 に対応させると，110 となるので，8 進数では 6 になる．これが，所有者，グループ，その他についてのパーミッションがあるので，全体で 8 進数 3 桁となる．Table 2 に，数値指定での数値の意味を示す．

たとえば，

Table 2　chmod での数字の意味

| | | | |
|---|---|---|---|
| 0 | 一切の権限なし | 4 | 読込 |
| 1 | 実行 | 5 | 読込＋実行 |
| 2 | 書込 | 6 | 読込＋書込 |
| 3 | 書込＋実行 | 7 | 読込＋書込＋実行 |

たとえば下記のコマンドで，所有者以外は読むことも書くこともできなくなる．

```
chmod 600 file
```

## 4　UNIX 上のエディタの使い方

UNIX 上の多くのプログラムはその設定をテキスト形式で保存している．そのため，その設定ファイルを編集することでプログラムのさまざまな設定を行うことができる．UNIX で使われている代表的なエディタとして vi と Emacs がある．この節では vi，Emacs の基本的な使い方を説明する．

### 4.1　vi の使い方

vi は多くの UNIX に標準で添付されているエディタである．vi は Windows に標準添付されているメモ帳や Emacs などのように一般的に使われているエディタと違い，操作方法が多少異なっている．

#### 4.1.1　コマンドモードと入力モード

vi には「コマンドモード」と「入力モード」という 2 つのモードがある．

コマンドモードではファイル中の編集したい位置 (以後，カーソルと呼ぶ) を移動したり，ファイルを保存する，テキストを検索するといった Windows のエディタではメニューバーやツールバー上から行う操作を行う．それに対して，入力モードでは実際に入力したい文字を入力する．

vi は起動したときにはコマンドモードになっている．この状態で’i’ と入力すると，入力モードに移り，カーソルのある位置に文字を入力できる．入力が終わったら，ESC キーを押すことで，コマンドモードに戻る．vi ではコマンドモードと入力モードを行き来することでテキストを編集していく．

### 4.2　Emacs の使い方

UNIX 上ではよく Emacs という非常に高機能なエディタが使われている．Emacs Lisp というプログラミング言語を用いることでエディタの機能をさらに拡張することができるが，この節では Emacs の基本的な使い方についてのみ紹介する．

#### 4.2.1　Emacs でのキー操作

Emacs では，コマンド操作を行うときにはほかのエディタとは違ったキー操作をする．これ以降の説明では，Table 3 のような表記ルールに従って基本となるファイル編集に必要となる操作を説明する．

#### 4.2.2　Emacs でのファイル編集

ここでは，Emacs でのデフォルトのキー操作を説明する．これらのキー操作は設定によって変更することも可能である．

Table 3　Emacs でのキー操作

| | |
|---|---|
| C-x | Ctrl キーを押しながら ‘x’ を押す |
| M-x | Alt キーを押しながら ‘x’ を押す |
| C-x C-f | Ctrl キーを押しながら ‘x’ を押し，次に Ctrl キーを押しながら ‘c’ を押す |

Emacs でテキストを開くためには，“C-x C-f” と打つ．そうすると，カーソルが画面の下の方に移るので，ここで編集したいファイル名を入力する．

編集したファイルを保存するためには”C-x C-s” と打つ．保存するファイル名を変えたい場合は “C-x C-w” と打ち，その後にカーソルが画面の下の方に移るので，保存したいファイル名を入力する．

コマンド操作をしようとして途中で操作を取り消したい場合は “C-g” と打つ．

文字を消去したい場合は消したい文字の上にカーソルを移動して “C-d” と打つ．DEL キーを押すと，カーソルの前の文字が消去されるので注意する．

文字列を検索したい場合は，“C-s” を入力し，続けて検索したい文字列を入力すると，1 文字ごとに入力文字列に一致する文字列にカーソルが移動する．“C-r” とすると，ファイルの先頭側に検索を実行する．

Emacs を終了するためには “C-x C-c” と打つ．この際，未保存のファイルは保存するか聞かれるが，‘y’ を押すと保存され，‘n’ を押すと破棄することができる．

### 4.3　vi と Emacs の比較

前項で紹介したように Emacs は非常に高機能なエディタであり，わざわざ変わった操作を行わないと使えない vi を使う必要性は感じられないかもしれない．しかし，UNIX では以下のような理由により，vi も使われてる．

- Emacs は UNIX の標準コマンドではないので，環境によっては Emacs が入っていないこともある．vi はほぼすべての UNIX システムに標準で用意されている．
- ちょっとした編集作業の場合は Emacs を起動するよりも，vi の方が処理が速い．

## 5　ファイル操作

一度に複数のファイルを扱うことが可能であることから，コマンドライン上で複数のファイルを扱う際にはワイルドカードを用いると便利である．

### 5.1　ワイルドカード

1 つのコマンドで複数のファイルやディレクトリを同時に操作したい場合はワイルドカードと呼ばれる特殊な記号を用いる．複数のファイルを指定できるコマンドであれば，すべてのワイルドカードを使用可能である．ワイルドカードには Table 4 のようなものがある．

Table 4　ワイルドカードの種類

| 記号 | 置き換え |
|---|---|
| * | 0 文字以上の任意の文字列 |
| ? | 任意の 1 文字 |
| [ ] | [ ] 内に指定された任意の 1 文字 |
| [^] | [ ] 内に指定されていない任意の 1 文字 |
| [!] | [ ] 内に指定されていない任意の 1 文字 |
| { } | 内に「,」で区切って指定された文字列のいずれか |

### 5.2　ワイルドカードの仕組み

ワイルドカードはシェルが展開する．つまり，コマンドの引数にワイルドカードを含むファイルを指定した場合，コマンドが引数を処理する前に，シェルがワイルドカードを解釈し，複数の引数に展開した上でコマンドに受け渡す．

### 5.3　ワイルドカード使用上の注意

cp のような「複数のファイルを引数とし，その並び順に意味があるコマンド」の場合や，find のようなコマンド自身もワイルドカードが処理できる場合には思わぬ動作をする可能性がある．

- cp での失敗例

  test で始まるファイルを別のディレクトリにコピーするつもりで「cp test*」したとする．このディレクトリに test1 と test2 があった場合，コマンドはシェルによって「cp test1 test2」に展開される．つまり，test1 を test2 にコピーするという展開になってしまう．

- find での失敗例

  test で始まるファイルを指定したい場合,「find . -name test*」としても実行できない．正規表現にマッチするカレントディレクトリ内の全ファイルを探したい場合は,「find . -name”test*”」する必要がある．

## 6　マウントとは

Windows にはドライブという概念があり，C ドライブ，D ドライブ，のように呼ぶが，Linux にはドライブという概念はない．すべてルートディレクトリを出発点としてアクセスできるようになっている．そのため，CD-ROM や複数のハードディスクを扱う場合，マウントという仕組みを用いる．

マウントとは，別のディレクトリツリーをルートディレクトリ以下のディレクトリに結合することを言う．例えば，CD-ROM ドライブを/mnt/cdrom というディレクトリにマウントした場合，CD-ROM の中にある「test」というディレクトリは「/mnt/cdrom/test」として扱うことができる．こうすることで，ユーザはファイルの物理的な場所を意識せず，統一した方法でファイルを扱うことができる．

### 6.1 マウント

ファイルシステムをマウントするときは mount コマンドを用いる．mount コマンドは「mount オプション デバイス名 マウントポイント」の書式でスーパーユーザのみが使用できる．

ただし，/etc/fstab で設定するとその設定に従ったマウント方法であれば，一般ユーザでもマウントできるようになる．一般ユーザが行う場合は「mount デバイス名 (またはマウントポイント)」のような書式になる．

「mount」のみで実行した場合は現在のマウント状況が表示される．一般ユーザでも実行できる．

### 6.2 アンマウント

FD，CD-ROM などの取り外し可能なデバイスは，マウント解除 (アンマウント) してから取り出す．アンマウントは unmount コマンドを用いる．

/etc/fstab の設定で，一般ユーザでもマウント可能になっている場合は，一般ユーザでアンマウントすることもできる．