第 1 回 UNIX ゼミ

ゼミ担当者　：　木田 清香，千田 智治，村上 耕平
指導院生　：　梶原 広輝，柴田 優，山川 望
開催日　：　2005 年 4 月 17 日

1　はじめに

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の 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．

2　UNIX(Linux) とは

2.1　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．

2.2　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．

2.3　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となっている．そのため，ユーザは自由にソースを変更して自分の環境にあわせたり，新しい機能を組み込むことが容易である．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．

3　UNIX 上でのコマンド群

3.1　基礎的コマンド

以下より基礎的なコマンドを紹介する．

- pwd

  「pwd」と入力すると，自分のカレントディレクトリのパスがわかる．

```
Kiyoka@kiyoka ~$ pwd
/home/Kiyoka
```

- 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．

```
Kiyoka@kiyoka ~$ ls
Hello.java Maildir foo test.c
```

- mkdir

  ディレクトリを作成するコマンドである.「mkdir 作成したいディレクトリ名」と入力すると，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．

```
Kiyoka@kiyoka ~$ mkdir sample
Kiyoka@kiyoka ~$ ls
Hello.java Maildir foo sample test.c
```

- cd

  ディレクトリを移動するコマンドである.「cd 移動したいディレクトリ名」と入力すると，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．

```
Kiyoka@kiyoka ~$ cd sample
Kiyoka@kiyoka ~/sample$ pwd
/home/Kiyoka/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．

```
Kiyoka@kiyoka ~$ ls
Hello.java Maildir foo sample test.c
Kiyoka@kiyoka ~$ rm -r sample
Kiyoka@kiyoka ~$ ls
Hello.java Maildir foo test.c
```

  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が削除でき，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 行ずつスクロールし，Space キーで 1 ページずつスクロールできる．また「q」を押すと処理を中断することができる．more コマンドの機能を強化したのが，less コマンドであり，比較的小さなファイルの閲覧に適しているのが cat コマンドである．

```
Kioka@kiyoka ~$ more Hello.java
public class Hello{
    public static void main(String
args[]) {
        System.out.println("Hello!");
    }
}
```

- cp

  コピーをするコマンドである.「cp file1 file2」と入力すると，ファイル file1 をファイル file2 にコピーすることができる．ディレクトリ dir1 をディレクトリ dir2 にコピーする場合には,「cp -r dir1 dir2」を用いる．

```
Kiyoka@kiyoka:~$ ls
Hello.java Maildir foo test.c
Kiyoka@kiyoka ~$ cp Hello.java
Hello2.java
Kiyoka@kiyoka ~$ ls
Hello.java Hello2.java Maildir foo
test.c
```

- 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．

  Hello.java ファイルを foo ディレクトリに移動させる場合は下記のコマンドで実行できる．

```
Kiyoka@kiyoka ~$ ls
Hello.java Hello2.java Maildir foo
test.c
Kiyoka@kiyoka ~$ mv Hello.java foo
Kiyoka@kiyoka ~$ ls
Hello2.java Maildir foo test.c
Kiyoka@kiyoka ~$ cd foo
Kiyoka@kiyoka ~/foo$ ls
Hello.java
```

  Hello2 というファイル名を Hello3 というファイル名に変更する場合は下記のコマンドで実行できる．

```
Kiyoka@kiyoka ~$ ls
Hello.java Hello2.java Maildir foo
test.c
Kiyoka@kiyoka ~$ mv Hello2.java
Hello3.java
Kiyoka@kiyoka ~$ ls
Hello.java Hello3.java Maildir foo
test.c
```

3.2 プロセスに関するコマンド

プロセスに関する基本的なコマンドとして，ps，top，kill を紹介する．

- ps

  現在動作中の自分のプロセスを確認するコマンドである．オプションをつけることで，Table 1 に示しているような機能を果たす．

Table 1　コマンドの ps の主なオプション

オプション	機能
-a	全ユーザが実行したプロセスを表示する
-u	ユーザ名と開始時刻を表示する
-x	制御端末のないプロセスを表示する
-w	1 プロセスあたりの表示行数を増やす

- top

  実行中のプロセス情報を表示するコマンドである．メモリ消費量，スワップ消費量，各プロセスのプロセス ID，メモリ消費量などがわかり，現在動作しているプロセスをリアルタイムに表示して確認することができる．

- kill

  ジョブ・プロセスを終了させるコマンドである．「kill プロセス番号」を用いることで，指定したプロセスを停止することができる．またプログラムの異常により，kill コマンドを実行しても終了できない場合がある．そのときは，-KILL もしくは -9 のオプションを使うことで強制終了することができる．オプションにはシグナルを指定することができ，それはシグナル名でもシグナル番号でもよい．シグナル番号はオプション -l で表示される順番に割り振られている．また「killall プロセス」を用いることで，指定したプロセスを全部停止することができる．

ps または top コマンドでプロセス ID を調べ，kill コマンドでプロセスを停止する，といったように使用する．

```
Kiyoka@kiyoka ~$ ps
 PID   COMMAND
2976   /usr/bin/bash
2676   /cygdrive/c/WINDOWS/system32/java
 276   /usr/bin/bash
3036   /usr/bin/ps
Kiyoka@kiyoka ~$ ps aux |grep java
2676   /cygdrive/c/WINDOWS/system32/java
Kiyoka@kiyoka ~$ kill 2676
Kiyoka@kiyoka ~$ ps
 PID   COMMAND
2976   /usr/bin/bash
 276   /usr/bin/bash
3256   /usr/bin/ps
```

3.3 ファイル・ディレクトリに関するコマンド

ファイル・ディレクトリに関する基本的なコマンドとして，find，du，df を紹介する．

- find

  ファイルを検索するコマンドである．基本的には次の書式を使う．

  ```
  find 開始ディレクトリ 検索条件 コマンド
  ```

  このコマンドを用いれば，さまざまな条件から目的とするファイルを検索することができる．検索条件にはオプションを用いる．例えば，カレントディレクトリ以下から foo.txt という名前のファイルを探したい場合は，下記のコマンドで実行できる．

  ```
  find .  -name foo.txt
  ```

- du

  ディレクトリ以下のファイル・ディレクトリのサイズを表示するコマンドである．このコマンドでよく使うオプションを Table 2 に示す．

Table 2　コマンドの du の主なオプション

オプション	機能
-k	1 ブロック 1KB 単位で表示する
-s	指定ディレクトリについて表示する
-h	読みやすい形式で表示する

- df

  ファイルシステムの容量を表示するコマンドである．du コマンド同様に，オプション -h で読みやす

```
Kiyoka@kiyoka ~$ ls -l
total 3
10 17:39 test 10 18:00 unix.tex
-rwx------ 1 Kiyoka 394 Apr 10 20:26 Hello.class
-rwx------ 1 Kiyoka 165 Apr 10 20:16 Hello.java
-rwx------ 1 Kiyoka 165 Apr 10 22:20 Hello3.java
drwxr-xr-x 2 Kiyoka   0 Apr 10 22:06 Maildir
drwxr-xr-x 2 Kiyoka   0 Apr 10 22:18 foo
drwxr-xr-x 2 kIyoka   0 Apr 10 22:09 test.c
```

Fig. 1　ディレクトリ内のファイル

い形式で表示することができる．

4　ファイル・ディレクトリについて

4.1　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 otal“3” 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容量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

  最初の 1 文字は，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 (-) かを表している．

- パーミッション

  2 文字目以降の “rwx”は ‘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．これは，通常のファイルとディレクトリでは多少意味が異なり，通常のファイルではほぼそのままの意味だが，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の中身を見ることができる権利であり，実行権とはそのディレクトリの中に移動することができる権利である．

- 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表す．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表す．

- 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．

4.2　所有権とパーミッション

4.2.1　所有権の設定方法

UNIX では，全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，下記の 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，下記の 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
```

ファイルの所有者と所有グループを同時に変更するには，下記のコマンドを使用する．

```
chown 所有者:グループ 変更するファイル
```

4.3 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 コマンドを使用するが，その設定する場合に 2 つの方法がある．

1 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 3 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．例えば，下記のコマンドで所有者以外の読み込みを禁止することができる．

```
chmod go-r file
```

Table 3　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)

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は，“rwx” の 3 種類がある．そこで，“rwx” の 1 文字 1 文字を 2 進数に見立てる．例えば，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 4 に，数値指定での数値の意味を示す．

Table 4　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

例えば，下記のコマンドを入力すると，8 進数 600 は，2 進数では 110000000 であるので，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
```

5　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．よって，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 では一般的に vi と Emacs というそれぞれ特徴の異なるエディタが使われている．この節では vi および Emacs の基本的な使い方を説明する．

5.1 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この後で説明する 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．

5.1.1 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の操作と同等の操作を行う．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．

5.1.2 vi でのファイル編集

vi でファイル編集のための操作を行うときはコマンドモードになってから行う．現在のモードがコマンドモードなのか，入力モードなのかわからないときは，Esc キーを押すことで，コマンドモードになることができる．

この後の説明中のアルファベットは大文字小文字が区別されるので，注意してほしい．

vi でファイルを開くためには，“:e filename” と入力する．vi の起動時にコマンドライン上で “vi filename” とすることでファイルが読み込まれる．

ファイルを保存するためには，“:w” と入力する．“:w filename” と入力することで，別ファイルに保存することが可能である．

入力された文字を消去するには，消したい文字の上にカーソルを移動して ‘x’ を入力する．“dd” と入力すると，カーソルのある一行全体が消去できる．

“/検索したい文字列” とすることで文字列検索することができる．‘/’ の代わりに ‘?’ を使うとファイルの先頭の方に検索を実行する．

vi を終了するためには，“:q” と入力する．編集したテキストを破棄したい場合は，“:q!” と入力する．“:wq” あるいは “ZZ” とすると保存と同時に終了することができる．

5.2 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．

5.2.1 Emacs でのキー操作

Emacs では，コマンド操作を行うときには他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 5 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．

Table 5　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

5.2.2 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．

基本的には emacs ファイル名 と入力し，ファイルを作成・編成する．

Emacs でテキストを開くためには，“C-x C-f” と入力する．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” と入力する．保存するファイル名を変えたい場合は “C-x C-w” と入力し，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” と入力する．

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” と入力する．Delete 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” と入力し，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” と入力すると，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” と入力する．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．

5.3 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．
- 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．

最後に，Table 6 に vi と Emacs で対応する編集操作の方法を掲載する．

Table 6　vi と Emacs のコマンド対応表

vi	Emacs	機能
:e filename	C-x C-f	ファイルを読み込む
:w	C-x C-s	ファイルを保存する
:w filename	C-x C-w	ファイルを別名で保存する
	C-g	コマンド操作を取り消す.
x	C-d	文字を消去する
/string	C-s	ファイル末尾方向への検索
?strung	C-r	ファイル先頭方向への検索
:q	C-x C-c	エディタを終了する