第 1 回 UNIX ゼミ

ゼミ担当者　：　吉形 允晴，澁谷 翔吾
指導院生　：　山川 望，山崎 弘貴，木田 清香，渡辺 崇文
開催日　：　2007 年 4 月 14 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，我々が利用・管理することになるバージョン管理システム「Subversion」や並列計算機において，UNIX 系 OS（Linux）でのコマンド操作をはじめとする UNIX のスキルが必要不可欠となる．そのため，本ゼミではそれらの利用が可能となるように，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を学ぶ．

2　UNIX(Linux)

2.1　UNIX とは

UNIX は 1960 年に米国 AT & T ベル研究所で Ken Tompson と Dennis Ritchie によって開発された OS である．アセンブリ言語で作られていた UNIX は，可読性と移植性等の欠点から 1973 年に C 言語に移植された．ソースコードが比較的コンパクト，ライセンスが安価で配布される等といった理由から，大学・研究機関を中心に普及し，その当時急速に広まった OS である．現在では，コンピュータの性能も上がったことから，一般的な PC で UNIX を使用されることも多くなっている．

2.2　Linux とは

Linux とは，1991 年にヘルシンキ大学の Linus Torvalds 氏により開発された UNIX 互換の OS である．Linux は既存の UNIX のソースを流用せず，何もないところから書きおこされたものであり，オープンソースソフトウェアとして，自由に改変，再配布ができるようになっている．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はではない．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．我々の研究室で主に利用されているのは，Debian GNU/Linux というデストリビューションである．

2.3　オープンソースとは

オープンソースとは，あるソフトウェアのソースコードを無償で公開し，世界中のプログラマの誰もが自由にそのソフトウェアを改良して再配布することを許すソフトウェア開発方式のことである．そのため，公開されているソフトウェアの類似品や技術の転用が可能であり，またそれを基盤に自分の環境に合わせたり，新たな機能を追加するといったことも可能である．

2.4　Cygwin とは

Cygwin は WindowsOS 上で UNIX 的環境を提供するソフトウェアの 1 つである．UNIX で使用されるシェルやコマンドなどのプログラムを，Windows 上でソースコードからコンパイルを可能にしたものである．つまりは UNIX コマンド (プログラム) を，Cygwin が仲介することにより WindowsOS 対応の機械語に翻訳し，それを WindowsOS が処理するといった形を採っている．このようにして Cygwin は Windows 下での UNIX 環境を提供する役割を果たしている．

3　UNIX のコマンド操作

UNIX の基礎的なコマンドを以下に紹介する.

3.1　コマンドの基本文法

UNIX におけるコマンドの基本的な書式を以下に示す．ここでオプションとは，コマンドの付加的な機能を実行させるもので,「-(ハイフン) アルファベット」で表現されるものが多く，また引数とは，コマンドやオプションに与える値やファイル名などのことを示す.

1. コマンド

   例)pwd

2. コマンド 引数

   例)mkdir dir

3. コマンド 引数 1 引数 2

   例)cp file1 file2

4. コマンド [オプション]

   例)ls -l

3.2　ディレクトリ操作

コマンド操作では，カレントディレクトリ (ユーザが現在作業を行っているディレクトリ) やディレクトリ内の情報を把握するため，次のようなコマンドが必要である.

- pwd

  カレントディレクトリを絶対パスで表示する.

```
Shogo@Shogo  $ pwd
/home/Shogo
```

  この表示は現在「home」ディレクトリの中の「Shogo」がカレントディレクトリであることを示している.

- ls

  ファイルやディレクトリの情報を表示する.

```
Shogo@Shogo   $ ls
dir1 dir2
```

  「ls」の入力に対して「dir1」と「dir2」が表示されている．これは,「Shogo」ディレクトリには「dir1」と「dir2」が存在することを示している．また，このコマンドと共によく用いるオプションを Table1 に示す.

Table 1　ls の主なオプション

オプション	機能
-a	ドットで始まる隠しファイルも表示
-l	ファイルやディレクトリの詳細を表示
-F	名前の後ろにファイルの型を表示

- mkdir

  ディレクトリを作成する.

```
Shogo@Shogo   $ mkdir dir3
Shogo@Shogo   $ ls
dir3
```

  「mkdir dir3」と入力することでカレントディレクトリに「dir3」というディレクトリを作成している.

- cd

  カレントディレクトリを移動する．ディレクトリの変更には絶対パスと相対パスを用いた方法があるが，相対パスを用いる場合は，現在のカレントディレクトリを基準として、指定したいディレクトリ名を用い「cd　[ディレクトリ名] 」と入力することで移動できる．なお，相対パス入力の場合は,「.」でカレントディレクトリ,「..」で 1 つ上のディレクトリへ移動できる．絶対パスを用いる場合は，ルートディレクトリを基準として，指定したいディレクトリまでを「／」区切りで示した絶対パスを用いて「cd [絶対パス]」と入力することで移動できる.

```
相対パス
Shogo@Shogo  $ ls
dir1
Shogo@Shogo  $ cd dir1
Shogo@Shogo  /dir1$ pwd
/home/Shogo/dir1
```

```
絶対パス
Shogo@Shogo  $ pwd
/home/Shogo
Shogo@Shogo  $ cd /home/Shogo Shibutani/dir1/
Shogo@Shogo  /dir1$pwd
/home/Shogo/dir1
```

3.3　ファイル操作

次にファイル名の変更，移動，コピー，削除といったファイル操作を行うコマンドを紹介する.

- mv

  ファイルの移動やファイル名の変更に使用する.「mv [移動するファイル名] [移動先ディレクトリ名]」と入力することでファイルをディレクトリに移動できる．また，ファイル名の変更は「mv [変更前ファイル名] [変更後ファイル名]」と入力することでファイル名を変更できる.

```
Shogo@Shogo   $ ls
dir1 sample1
Shogo@Shogo   $ mv sample1 dir1
Shogo@Shogo   $ ls
dir1
Shogo@Shogo   $ cd dir1
Shogo@Shogo  /dir1 $ ls
sample1
```

  ここでは， sample1 というファイルを dir1 というディレクトリに移動している.

- cp

  ファイルやディレクトリをコピーする.「cp [コピー元ファイル名] [コピー先ファイル名]」と入力することでファイルをコピー できる．ディレクトリもファイルのコピーと同様の操作でできる．”-r”のオプションをつけることで，ディレクトリもコピーできる

```
Shogo@Shogo   $ ls
sample1
Shogo@Shogo   $ cp sample1 sample2
Shogo@Shogo   $ ls
sample1 sample2
```

- rm

  ファイルやディレクトリを削除する.「rm [削除するファイル名]」と入力することでファイルを削除できる．ディレクトリの削除は「rm -r [削除するディレクトリ名]」で削除できるが，ディレクトリを削除する場合は，そのディレクトリに存在するファイルを全て削除しておく必要がある．

```
Shogo@Shogo  $ls
sample1 sample2
Shogo@Shogo  $rm sample1
Shogo@Shogo  $ ls
sample2
```

3.4 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コマンドを以下に紹介する．

- find

  ファイルを検索する.「find [開始ディレクトリ]　[検索条件]　[コマンド]」でファイルを検索できる．検索条件にはオプションを用いる．

```
Shogo@Shogo  /dir1$ ls
dir2 sample2
Shogo@Shogo  /dir1$ cd ../../
Shogo@Shogo /home
$ find /home -name sample2
/home/Shogo/dir1/sample2
```

- 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．このコマンドと共によく用いられるオプションを Table2 に示す.

Table 2　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
-h	読みやすい形式で表示
-s	指定ディレクトリについて表示

4　パーミッションの設定

本章では，パーミッションの設定を行う．パーミッションとは，ファイルやディレクトリに対するユーザのアクセス権のことである．

4.1　パーミッションの設定対象

UNIX におけるパーミッションではファイル，ディレクトリの所有者である Owner，同じマシンを利用できるユーザ全体を意味する Group，そしてその他の Other に対して，それぞれ「読み込み (r)」「書き込み (w)」「実行 (x)」の権限の設定ができる．以下にパーミッションの確認方法と変更方法を紹介する．

4.2　パーミッションの確認方法

ファイル・ディレクトリの情報を表示する「ls」コマンドに，ファイルの詳細を表示するオプション「-l」をつけて実行することで確認する．以下の例ではファイル「dir1」のパーミッションを確認している．

```
Shogo@Shogo  $ ls -l   sample1.txt
-rwxr-x--x 3 Shogo None 0 Apr 11 13:51 sample1.txt
```

「ls -l」コマンドによる出力結果を空白で区切るとすると，左端「-rwxr-xr-x 」において，一番左の文字が「-」ならファイル,「d」ならディレクトリという意味である．一番左の文字を除いた残りの文字は左から 3 桁ずつ，Owner，Group，Other に与えられた権限を示している．例えば,「-rwxr-x–x」という表示では，Owner には「読み込み (r)」「書き込み (w)」「実行 (x)」，Group には「読み込み (r)」「実行 (x)」，Other には「実行 (x)」の権限が与えられているということを示している．以降順に「3」がリンク数,「Shogo」が所有者,「None」がグループ所有者,「0」がファイル・ディレクトリサイズ,「Apr 11 13:51」が日付,「dir1」がファイル・ディレクトリ名を表している．

4.3　数値を用いたパーミッションの設定

- chmod [モード (数値)] [ファイル名]

  パーミッションを設定する際は 3 桁の数字が用いられる．左から順に Owner 権限，Group 権限，Other 権限を意味し，それぞれの対象に与えたい権限を，Table3 に示す数字を足し合わせて指定する．例えば「chmod 755 sample1」であれば，sample1 の Owner に全ての権限，Group と Other に読み込み，実行の権限が与えられる．

Table 3　数字と権限の対応

記号	数字	権限
r	4	読み込み
w	2	書き込み
x	1	実行
-	0	なし

```
Shogo@Shogo  $ls -l sample1.txt
-r-xr--r-- 1 Shogo ...  sample1.txt
Shogo@Shogo  $ chmod 755 sample1.txt
Shogo@Shogo  $ ls -l sample1.txt
-rwxr-xr-x 1 Shogo ...  sample1.txt
```

4.4　文字を用いたパーミッションの設定

- chmod モード (文字) ファイル名

  この方法は，どのユーザに対して，どのパーミッションを追加・削除するかを決める方法である．Table4 に示すように，権限を与える対象を Owner(u)，Group(g)，Other (o)，全て (a) とし，行う操作は+（追加），-（削

除)，= (設定) のいずれかである．オプションには,「対象・操作・権限」の順に続けて入力する．例えば「chmod go+rw sample1」と入力した場合は，sample1 に対して，Group と Other に「読み込み (r)」と「書き込み (w)」の権限を追加したことになる．

Table 4　文字による権限設定

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)
権限	r(読み込み)，w(書き込み)，x(実行)

```
Shogo@Shogo  $ls -l sample1.txt
---------- 1 Shogo ...  sample1.txt
Shogo@Shogo $chmod go+rw sample1,txt
Shogo@Shogo $ls -l dir1
----rw-rw- 1 Shogo ...  sample1.txt
```

5　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を以下に紹介する．

- ps

  現在動作中の自分のプロセスを確認する．Table5 に主なオプションと機能の対応を示す.

Table 5　オプションと機能の対応

オプション	機能
aux	全てのプロセスを表示
a	他のユーザが権限を持つプロセスも表示
u	ユーザ名と開始時刻を表示
x	制御端末のないプロセスについても表示

```
Shogo@Shogo  $ ps
PID PPID PGID WINPID TTY UID STIME COMMAND
S 4212 1 4212 4212 con 1000 21:12:46 /usr/bin/bash
S 3904 4212 3904 1528 con 1000 21:16:53 /usr/bin/top
S 6136 4212 6136 3108 con 1000 21:18:02 /usr/bin/ps
```

このように「ps」を使うことで，3 つのプロセスが動いていることが確認できる．

- top

  実行中のプロセス情報を表示する．メモリの消費量，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.「kill プロセス番号」と入力することによって，指定したプロセスを停止することができる．強制終了する際は，オプションに「-KILL」もしくは「-9」を用いる．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「kill -l」で確認できる．

6　テキストエディタ

UNIX では，ユーザの文書やプログラムソースだけでなく，システム設定ファイルまでもがテキスト形式となっている．このため，Linux をはじめとする UNIX 系 OS ではテキストエディタが必須となる．UNIX 上で使われる代表的なテキストエディタに vi と Emacs がある．本章では，その vi と Emacs の特徴と基本的な操作方法を紹介する．

6.1　vi とは

vi は UNIX システム標準のテキストエディタであり，emacs に比べてプログラムが小さく起動が速く負荷も少ない．Windows に標準添付されているメモ帳をはじめ，次節で説明する Emacs などの一般的に利用されているエディタと操作方法が異なる．

vi にはコマンド入力モードとテキスト入力モードがあり，それらを切り替えながらテキストを作成する．コマンド入力モードではファイルを編集したい位置にカーソルを移動したり，ファイルを保存したり，テキストを検索するといった操作を行う．それに対して，テキスト入力モードでは，実際に入力したい文字を入力する．

6.1.1　vi ファイル編集

まず，vi 起動をするために,「vi」とコマンド入力する．ファイル名を指定する場合は「vi ファイル名」と入力する．エディタ起動後にファイルを開くためには,「: e ファイル名」と入力する．ファイルを保存するためには,「:w」と入力する．また,「:w ファイル名」と入力することで，別ファイルに保存することができる．vi で用いるコマンドを Table6 に示す．

Table 6　vi コマンド表

コマンド	意味
i	カーソルの直前にテキストを入力
a	カーソルの直後にテキストを入力
o	改行してテキストを入力
ESC	コマンドモードへ戻る
:e	ファイルの読み込み
:w	ファイルの保存
:q	vi の終了
:q!	保存せずに強制終了
:wq	書込みと同時に vi を終了
h	カーソルを 1 文字左へ移動
l	カーソルを 1 文字右へ移動
j	カーソルを 1 文字下へ移動
k	カーソルを 1 文字上へ移動
x	カーソル位置の文字を削除
$	カーソルを行の末尾に移動
0	カーソルを行の先頭へ移動

入力された文字を消去したいときは，その文字の上にカーソルを移動して「x」と入力する．また,「dd」と入力すると，カーソルのある一行が全て消去される．vi を終了するときは「:q」と入力する．

6.2　Emacs とは

Emacs とは，vi と並び，UNIX でよく使われるエディタである．一般的に GUI で表示されるエディタで，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．本節では Emacs の基本的な使い方についてのみ説明する．

6.2.1　Emacs ファイル編集

まず，Emacs を起動するためには,「emacs」と入力することで起動する．ファイル名を指定する場合「emacs ファイル名」と入力し，エディタを開く．以前に保存したファイルを呼び出すには「C-x C-f」と入力し，画面下部で呼び出すファイル名を入力する．

編集したファイルを保存するには「C-x C-s」と入力する．新規ファイルの場合，画面の下部で保存ファイル名を入力することによって保存できる．また，そのファイルが以前に作成されたファイルならば，上書きされる．Emacs でのコマンドを Table7 に示す．なお，Table7 の「C-x」は Ctrl キーを押しながら'x' を押すという意味である．

Table 7　Emacs コマンド表

キー入力	意味
C-x C-f	ファイルを開く
C-x C-s	上書き保存
C-x s	すべてのファイルを保存
C-x C-c	ファイルを閉じる
C-x C-v	別のファイルを開く
C-x i	カーソル位置に別ファイルを挿入
C-x C-w	ファイルを別名で保存

6.3　vi と Emacs の違い

一般的に vi はテキストモードのコンソール上で，Emacs は GUI アプリケーションとして動作する．一般ユーザからすれば GUI で動作する Emacs の方が扱い易いが，コマンド操作に慣れている人にとっては，キー操作のみで処理でき，小型で高速な vi の方が使い勝手が良いとされている．

7　その他の UNIX コマンド

- man

  man コマンドは「man コマンド名」と入力することにより，コマンドの説明を表示するものである．下記の実行例は，ls のコマンドを man コマンドで調べた結果の一部である．

```
Shogo@Shogo  $man ls
NAME    ls - list directory contents
SYNOPSIS    ls [OPTION]...  [FILE]...
DESCRIPTION    List information about the FILEs (the current directory by default).
   Sort entries alphabetically if none of -cftuvSUX nor --sort.
   Mandatory arguments to long options are mandatory for short options too.
    -a, --all
       do not ignore entries starting with .
    -A, --almost-all
       do not list implied .  and ..
    --author
       with -l, print the author of each file
```

  「man」コマンドでコマンドを調べると，上記のように「NAME」，「SYNOPSIS」，「DESCRIPTION」の項目が表示される．それぞれの意味として，「NAME」がコマンド名の由来，「SYNOPSIS」がコマンドの書式，「DESCRIPTION」がコマンドの特徴とオプションを示している．