UNIX コマンドと vi エディタ

目 次

1 UNIX 機の利用開始と終了

この授業では，Windows が動作しているコンピュータから，UNIX 機に遠隔ログイン (remote login) して実習を行います。

UNIX では，コンピュータの利用開始の手続きを，ログイン (login) と呼びます。遠隔ログイン (remote login) とは，目の前 (**local**) のコンピュータから，ネットワークで接続されている遠隔 (**remote**) のコンピュータにログインすることです。

この授業で UNIX 機に遠隔ログインする方法については，別途連絡します。ログアウトするときには，`exit` コマンドを実行してください。

2 コマンド

2.1 コマンドの実行

UNIX が搭載されているコンピュータの操作の基本は，キーボードからコマンドを入力して実行することです。本章では，コマンド実行に関わる基本事項を学びます。

- **コマンド (command)** はコンピュータへの命令です。
- UNIX にログインして表示される

  [*user*@*host* ~]$

  を**プロンプト (prompt)** と呼びます[1]。プロンプトが存在する行を**コマンド行 (command line)** と呼びます。
- プロンプトが現れれば，キーボードからコマンドを入力できます。入力の誤りは `<BACKSPACE>` (Backspace キー) で修正できます。続いて `<ENTER>`（エンター・キーまたはリターン・キー）を押せば，入力したコマンドが実行されます。

  例) date コマンドで現在の日付と時刻を表示する

  ```
  date<ENTER>
  ```
- UNIX のコマンドにおける大文字と小文字は，異なるものとして区別されます[2]。
- ほとんどのコマンドには，**引数 (ひきすう; argument)** を使って，コマンドの動作に関わる指示を与えることができます。引数が必須のコマンドもあります。

  引数は，コマンド行に入力したコマンドの名前に続いて，空白で区切って記述します。

  引数として記述すべき事柄はコマンド毎に異なります。

  例) 引数 `+%D` を付けて date コマンドを実行する

  ```
  date +%D<ENTER>
  ```

[1] プロンプトの表示形式は設定や状況によって変わります。利用者が設定を変更することも可能です。
[2] Windows のコマンドには，大文字・小文字の区別がありません。

練習

1. `cal` コマンドを実行してください。今月のカレンダーが表示されましたか。
2. 大文字で `CAL` と入力して `<ENTER>` を押してください。Shift キーを押しながら英字を入力すると大文字になります。

   「コマンドが見つかりません (Command not found.)」としかられますね。UNIX に cal コマンドはありますが，CAL コマンドはありません。
3. `cal 10 1966` を実行してください。10 や 1966 は，cal コマンドの引数です。

   cal コマンドは，引数が無ければ今月の，引数があれば指定の月あるいは年のカレンダーを表示します。
4. 途中に空白を入れずに `cal101966` を実行してください。

   cal101966 というコマンドはありません。コマンドと引数，および複数の引数を区切るには空白が必要です。
5. `man date` を実行してください。man はコマンド等のマニュアルを表示するコマンドです。

   表示が一画面におさまりきらないため，最終行に何やら表示されて止まっています。`<SPACE>` (スペース・キー) を押すと次の画面に進みます。`q` を打つと終了します。
6. `man` を実行してください。

   man コマンドには，調べたいコマンドの名前（コマンド名）等を，引数として与える必要があります。引数が必要かどうかは，コマンドによって異なります。
7. `clear` コマンドを実行してみましょう。

   きれいになりましたか。

2.2 困ったときには

- キーボードから入力できない。

  CTRL-q を押してみてください。CTRL-q は Ctrl キーを押しながら q キーを押すことを意味します。
- プロンプトが出ない。

  CTRL-d を押してみてください。それでもだめなら CTRL-c を押してみてください。CTRL-c は現在実行しているコマンドを強制終了します。
- 英字のキーを押すと大文字で入力される。Shift キーと一緒に押せば小文字になる。

  キーボードの Caps Lock ランプが点灯していれば，大文字入力になります。Caps Lock キー，または Shift キーと同時に Caps Lock キーを押してください。

- exit コマンドでログアウトしようとすると「中断した job が残っています」と表示されてログアウトできない。

  fg とタイプしてエンターを押すと画面にアプリケーションが現れます。それを正しい方法で終了させてからログアウトを試してください。

2.3 ファイル

コンピュータで作業したことを残しておきたいときには**ファイル (file)** を利用します。ファイルを作るには，その中身に加えて，ファイルの名前（**ファイル名**）が必要です。ファイルはファイル名で区別され，ファイル名を使ってファイルの中身を見たり，中身を変更できます。

ファイルの作成や利用の方法はいろいろとありますが，この章では，コマンド行の操作だけでファイルを扱います。

2.3.1 ファイルを作って見てみよう — ls, cat, >

- ls（エル・エス）コマンドは所持するファイルの名前 (ファイル名) を表示するコマンドです。
- 「`cat` ファイル名」を実行すれば，ファイルの内容が表示されます[3]。
- コマンド [と引数] の後ろに「`>` ファイル名」を指定すると，コマンドの実行結果は画面に表示されません。その代わりに，指定したファイル名の新しいファイルができ上がり，そこに結果が書き込まれます。

では，実際に試してみましょう。

1. `ls` コマンドを実行してみましょう。

   まだファイルを作っていないので，何も表示されません。

2. `date` コマンドを実行してみましょう。

   現在の時刻が表示されましたね。

3. `date > now` を実行しましょう。

   date コマンドの実行結果は画面に表示されません。その代わりに，now という名前のファイルに書き込まれました。

4. 再度，`ls` を実行してみましょう。

   now という名前のファイルがありますね。

5. `cat now` を実行して，now の中身を見てみましょう。

[3] cat コマンドで内容を表示できるのは，テキストファイルと呼ばれるファイルのみです。

練習

1. `now cat` を試してみましょう。

   しかられましたね。一般に，プロンプトに続いては，ファイル名 (now) ではなく，コマンド名 (cat) を打つ必要があります[4]。

2. `cat dog` を実行してみましょう。

   cat コマンドの引数には，既存のファイルの名前を与えなければいけません。

3. 引数を与えずに `cat` を実行してみましょう。

   プロンプトが現れませんね。第 2.2 節を読んで，正しい対処をしてください。

2.3.2 ファイル名の付け方

- UNIX では / をファイル名に含めることはできません。
- 小文字と大文字は区別されます[5]。特に理由がなければ，小文字のファイル名を使うのが慣例です。
- UNIX では，.（ドット; ピリオド）で始まる名前のファイルは，多くのコマンドで特別に扱われます。

当面，英数字と . (ドット)，_ (アンダーバー) のみを組み合わせたファイル名の利用を推奨します。ただし，ドットで始まるファイル名は避けるのが無難です。

ファイルはファイル名で識別して操作するものですから，ファイル名はファイルの内容を連想できるものにしましょう。ドットやアンダーバーはファイル名を構成する単語を区切るために使われます。

練習

1. `date > a/b` を実行してみましょう。

   エラーメッセージが表示されましたね。エラーの意味は，ディレクトリの学習をすればわかります。

2. `ls` を実行しましょう。続いて `date > Now` を実行して，`ls` を実行しましょう。

   now と Now は区別されていますか。

3. now と Now の内容を見比べてください。

4. `cal` を実行してください。次に `cal > this_month` を実行してから，ファイルができたか，確認してください。

[4] 例外はありますが，細かなことは無視しましょう。

[5] Windows のファイルシステムでは，ファイル名の大文字・小文字は区別されません。そのため，大文字・小文字のみが異なる名前のファイルを，同じフォルダに作ることはできません。

2.3.3 ファイル名の変更とファイルの複写・削除 — mv, cp, rm

コマンドを使って，作成したファイルの名前を変更したり，ファイルの内容を別のファイルに複写したり，ファイルを削除することができます。そのためのコマンドは mv, cp, rm です。ファイル名を引数として次の形式で使います。

mv *file1 file2* — *file1* の名前 (ファイル名) を *file2* に変更

cp *file1 file2* — *file1* の内容を *file2* に複写 (*file1* と同じ内容のファイル *file2* を作る)

rm *file* — *file* を削除

注意

- 削除したファイルは二度と戻りません。rm コマンドを実行するときには心を落ち着かせてから`<ENTER>` を押すことを習慣づけましょう[6]。
- cp や mv の最後の引数 (上で *file2* と表記したもの) に既存のファイルの名前を指定すると，元のファイルは上書きされて無くなります[7]。このことは「コマンド > ファイル名」でファイルにコマンドの実行結果を書き込むときも同様です[8]。誤って必要なファイルを消さないように，既存のファイルをよく確認してから実行しましょう。

練習

1. 再び `ls` を実行してから `cat now` を実行しましょう。
2. ファイル now の名前を old に変更します。`mv now old` を実行してください。名前が変わったことを ls コマンドで確認してください。cat コマンドでファイルの中身も見てみましょう。
3. old と同じ内容のファイル old.copy を作ります。`cp old old.copy` を実行してください。ファイルができたか ls で確認してから，old と old.copy の内容を見比べてみましょう。
4. old.copy を削除します。`rm old.copy` を実行してください。うまくいったか確認してください。
5. 続いて old と this_month を削除してください。結果を確認してください。

2.3.4 コマンドのオプション

多くのコマンドには，**オプション (option)** と呼ばれる，コマンド実行に必須ではない引数を指定できます。オプションは，コマンド動作の詳細を指定するもので，通常，- か -- に続いて指定します。各コマンドのオプションは man コマンドを使って調べることができます。

例) `ls -a`

ls のオプション -a は . (ドット) で始まるものを含めて，すべてのファイルの名前を表示するためのものです。

---

[6] rm 実行時，本当にファイルを削除してよいか，確認のメッセージを出すように設定しているシステムもあります。

[7] cp や mv の実行時に，既存のファイルを上書きしてよいかどうか，確認のメッセージを出すように設定されているシステムもあります。

[8] 設定によっては上書きされないこともあります。

注意) ここで表示される . (ドット) で始まる名前のファイルは，コンピュータを正常に使うために必要なファイルです。削除したり，名前を変えたりしないでください。

練習

1. `cal > month` を実行してから，`cat month` を実行してください。

   ファイル month の内容が表示されましたね。

2. `cat -n month` を実行してみましょう。

   cat は -n というオプションを受け付けます。-n は行番号を付して表示するために使います。

3. `ls` を実行してから，`ls -l`（エル）を実行してみましょう。

   ls コマンドのオプション -l は，ファイルのいろいろな情報を表示する指定です。ここでは，ファイルの最終変更日時がわかることにだけ，注目することにしましょう。

4. `ls -a -l` と `ls -al` の実行結果を比較してください。

   同じですね。a や l のような 1 文字のオプションは，– の後ろにまとめて指定できる場合が多いです。

3 vi エディタを使ってみよう

3.1 テキストファイルとエディタ

cat コマンドや more コマンド等で内容を表示できる，文字の書かれたファイルを**テキストファイル** (text file) といいます。一方で，cat コマンド等では中身を読めない通常のファイルもあります。これをバイナリファイル (binary file) と呼びます[9]。例えば，画像や音声の入ったファイルの多くや，Windows 上のワープロ等で “普通に” 保存したファイルの多くはバイナリファイルです。

テキストファイルを作成したり編集するためのプログラムを**テキストエディタ** (text editor) といいます。単にエディタと呼ぶことも多いです。UNIX にはさまざまなエディタがありますが，この章では UNIX の標準的なテキストエディタである vi の基本的な使い方を紹介します。

3.2 vi の基本操作

3.2.1 vi の起動法

vi を起動するには，コマンド行で

`vi` *file*

を実行します[10]。ここで *file* には，新規に作成するファイルの名前や，編集したい既存のファイル名を与えます[11]。

3.2.2 vi の編集モード

vi はモードという概念を持つエディタです。vi では，現在どのモードになっているかにより，使用可能な操作が異なります。

コマンド入力モード vi 起動直後には，このモードになっています。カーソル移動や，文字の消去，ファイルへの保存等の操作は，コマンド入力モードにおいて，vi のコマンドを入力することにより行います。vi のコマンドの一部を付録 B に挙げますので，参考にしてください。

コマンド入力モードでは文字の入力はできません。**i** や **a** 等の vi のコマンドを入力し，次に紹介する文字入力モードに移行してから，文字入力をします。

文字入力モード このモードは文字を入力するためのモードであり，基本的にそれ以外の操作はできません。カーソル移動等のためには，`<ESC>` を押して，コマンド入力モードに戻る必要があります[12]。

ただし，`<ENTER>` や `<ESC>` を押す前であれば，入力したばかりの文字は `<BACKSPACE>` (バックスペースキー) または CTRL-h で削除できます[13]。`<BACKSPACE>` を続けて押すと，さら

[9] より正確にいうと，文字コードと若干の制御コードのみを含むファイルがテキストファイルであり，それ以外のファイルがバイナリファイルです。cat コマンドを使うと，ファイル内の文字コードが文字に変換されて表示されますが，od というコマンドを使うとファイル内の文字コードそのものを見ることができます。

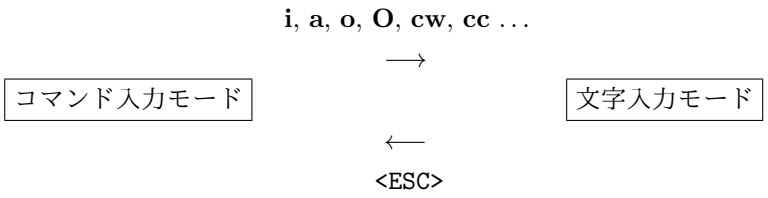
[10] 最近のシステムでは vi というコマンドを実行すると，実際には vi の機能拡張版である vim が起動するものが多いです。また，vi コマンドで vi が起動しない場合，代わりに vim と打ってみてください。

[11] *file* を指定せずに vi を起動してから，既存のファイルを読み込んだりもできますが，ここでは必ずファイル名を指定して起動することにします。また，ファイル名を複数指定して編集する機能も扱いません。

[12] 文字入力モードでもカーソルキー（矢印キー）でカーソル移動が可能な場合があります。

[13] これらと違うキーを使うシステムもあるかもしれません。

にその前に入力した文字を削除できます。なお，<BACKSPACE> を押しても，カーソルが左に動くだけで元の文字は表示されたままになっているかもしれませんが，これらは無視してください。<ESC> を押せばちゃんと消えます。



vi では，これらのモードを行ったり来たりしながら，文書を編集します[14]。現在，どちらのモードなのかが分からなくなったら，<ESC> を 1 回または 2 回押して，コマンド入力モードに移行してください。

3.3 例題

3.3.1 例題 1 —ファイルの新規作成

vi を使ってファイルの中身が

```
cal
date
who
```

である commands という名前のファイルを作成してみます。

1. vi の起動法に従って，コマンド行に `vi commands` と入力して，vi を起動してください。
2. **i** を押してください。

   この操作によって「コマンド入力モード」から「文字入力モード」に移行しました。これ以降，カーソル位置に文字を入力できます。
3. ファイルに書き込む内容 (cal<ENTER>date<ENTER>who) を入力してください。<ENTER> (エンターキー) は改行するためのキーです。タイプミスをしてしまっても，ここでは修正しないでおきましょう。
4. <ESC> (エスケープキー) を押してください。

   この操作によって,「文字入力モード」から「コマンド入力モード」に移行しました。
5. **:x** と打って <ENTER> を押してください。この vi のコマンドによって，3. で入力した内容がファイル commands に保存され，vi が終了してプロンプトが現れます。うまくいかない場合，再度 4. と 5. を行ってください。

以上を終えたら，commands という名のファイルが出来上がったことを，ls コマンドを使って確認してください。また，cat コマンドでファイルの内容を確認してください。

[14] より多くのモードを考えるのがよい場合もありますが，ここでは簡略化してモードは二つとします。

3.3.2 例題 2 —既存ファイルの編集

ファイル commands の内容を

```
cal     displays a calender
date    set date and time
```

に変更します。

1. vi の起動法に従って，コマンド行に `vi commands` と入力して vi を起動してください。既存のファイル commands の内容が表示されます。

   もし，画面にファイル commands の内容が表示されなければ，**:x** `<ENTER>` を打って一旦 vi を終了し，正しいファイル名で再度 vi を起動してください。

2. 文字を追加すべき位置 (cal の l の位置) までカーソルを移動します。カーソル移動はキーボード上のカーソルキー（矢印の書かれたキー）で行えますが，付録 B に記したように **l** (エル) 等のキーを使うこともできます[15]。

3. カーソルの右側に文字を追加するために，**a** を打ちます。

   これにより，文字入力モードに移行しました。コマンド i と a の違いを付録 B で確認してください。

4. 追加すべき文字「　　`displays a calender`」(4 個の空白に続いて display...) をタイプします。入力ミスは `<BACKSPACE>` で修正できます。

5. `<ESC>` を押してコマンド入力モードに戻り，date の行の行末までカーソルを移動します。

6. **a** を押して文字入力モードに入り，追加すべき文字を入力します。入力を終えたら `<ESC>` でコマンド入力モードに戻ります。

7. カーソルを who の行の行頭まで移動させてください。

   これから who を削除しますが，`<BACKSPACE>` では文字入力モードで打ち込んだばかりの文字しか消せませんので，ここでは文字削除のコマンドを使います。まず **x** を押してカーソルの乗っている文字 w を消してみましょう。

   一行すべてを削除するコマンドもあります。**dd** と打って，カーソルの乗っている 3 行目を消してください。

8. 編集を終えたら **:x** `<ENTER>` を打ちます。変更した内容が元のファイルに保存され，vi が終了します。

3.4 練習

1. ファイル commands の内容を

   ```
   cal     displays a calender
   date    display or set date and time
   ```

[15] vi では，右手ホームポジション付近のキーだけでカーソル移動ができます。慣れると快適です。

に変更してください。

この練習では，ファイルの内容を，上記の通りにきれいに整えましょう。コマンド説明の出だしが揃っていなければ，揃えてください。不要な空白行があれば，行削除の vi コマンドで削除してください。

注意: vi は行単位でテキストを編集するエディタなので，改行文字を消すことで複数の行を一行にまとめたりはできません。その代わりにコマンド **J** を使います。

2. 実習用のコンピュータで vi を起動すると，実際には vi の拡張版である vim が起動します。vim には操作を学ぶためのチュートリアルがあります。vimtutor というコマンドで起動できますので，時間があれば試してください。

A UNIX コマンド

分類	コマンド	機能
オンラインマニュアル	`man` *command*	*command* のマニュアルを表示
ディレクトリ操作	`ls`	カレント・ディレクトリに存在するファイルの名前を表示
	`cd` *directory*	カレント・ディレクトリを *directory* に変更
	`pwd`	カレント・ディレクトリ名の表示
	`mkdir` *directory*	*directory* の作成
	`rmdir` *directory*	*directory* の削除
	`mv` *directory1 directory2*	*directory1* の名前を *directory2* に変更
ファイル操作	`cp` *file1 file2*	*file1* を *file2* に複写
	`cp` *files directory*	*files* を *directory* に複写
	`mv` *files directory*	*files* を *directory* に移動
	`mv` *file1 file2*	*file1* の名前を *file2* に変更
	`rm` *files*	*files* を削除
その他	`cat` *files*	*files* の内容を表示
	`more` *files*	*files* の内容を一画面毎に表示
	`file` *files*	*files* の種類を表示
	`diff` *file1 file2*	*file1* と *file2* の違いを表示

B　vi の主要コマンド

機能	コマンド	コメント
テキストを入力する	**i** *text* <ESC>	カーソルの位置に *text* を挿入する (insert)
	a *text* <ESC>	カーソルの右に *text* を追加する (append)
	o *text* <ESC>	今いる（カーソルがある）行の下に *text* を入れる
	O *text* <ESC>	今いる行の上に *text* を入れる
ファイルを保存する	**:w** <ENTER>	編集中のファイルを元の名前のまま保存する
	:w *file* <ENTER>	編集中のファイルを *file* として保存する
vi を出る	**:x** <ENTER> (**ZZ**)	ファイルを保存し vi を出る
	:q! <ENTER>	ファイルを保存せず vi を出る
カーソルを動かす	**h** (←)	左隣に移動する
	l エル (<SPACE> または →)	右隣に移動する
	k (↑)	上に移動する
	j (<ENTER> または ↓)	下に移動する
	0 ゼロ	行頭に移動する
	$	行末に移動する
	CTRL-f　CTRL-b	次ページ / 前ページに移動する
	G	最後の行に移動する
	n **G**	第 *n* 行に移動する
テキストを削除する	**x**	今いる文字（カーソルが乗っている文字）を削除する
	X	カーソルの左隣の文字を削除する
	dw	今いる単語を削除する
	dd	今いる行を削除する
行を連結する	**J**	今いる行に次の行を連結する
テキストを検索する	/ *string* <ENTER>	*string* が最初に現れる位置にカーソルを移動する
	n	一番最近行った検索を繰り返す
テキストを置換する	**r** *character*	今いる文字を *character* で置換する
	cw *text* <ESC>	今いる単語を *text* で置換する
	cc *text* <ESC>	今いる行を *text* で置換する
変更を繰り返す	**.**	直前のコマンドによる変更を繰り返す
変更を取り消す	**u**	直前のコマンドによる変更を取り消す (undo)
テキストをコピーする	**yy**	今いる行を名前なしバッファにコピー（ヤンク）する
	p	名前なしバッファ中のテキストを挿入（プット）する

括弧 () は当該操作を他のコマンドで行えることを表します。そのコマンドを括弧内に併記しています。
斜体字 (*italic*) の箇所は具体的なものに置き換えて記述します。例えば，*text* には入力テキストを，*file* には
ファイル名を書きます。