

テキストエディタ (vi) の使い方と簡単な csh の実習

1 vi

vi はテキストエディタの名前である。テキストエディタとは、コンピュータ上でテキスト(文章やプログラム)を編集するためにつかう。

エディタには色々ある(例えば Windows のメモ帳もエディタ)が、ここで vi の練習をする理由は、vi は UNIX(LINUX)系の OS には必ず付いているので、これを覚えていれば何処に行っても困らないから、である。

vi はモードエディタである。モードエディタとは、例えば“文字入力モード”の時にはキーボードから打った文字はそのまま編集中のテキストに挿入されるが、文字入力モードを終了すると、同じ文字を打ってもエディタに対するコマンドとして解釈される、など、エディタが複数の状態で制御されることを指す。面倒なようだが、慣れればどうということはない。

2 vi の起動

以下のコマンドを入力して vi を起動する。

```
$ vi (ファイル名)
```

なお、コマンドの先頭にある“\$”はプロンプト(コンピュータが入力を待っている状態であることを示す印)なので入力しないこと。(ファイル名)のファイルが既に存在すればそれを編集する。存在しなければ新規ファイルとして作成する。

3 vi のコマンド

vi は、入力された文字に対応する動作を行う。アルファベットの大文字と小文字は、vi に区別されて認識されるので注意。

文字入力

- | | |
|---|-------------------------------------|
| a | カーソルの後に文字を挿入する“入力モード”に移行する |
| i | カーソルの前に文字を挿入する“入力モード”に移行する |
| o | カーソルのある行の後ろに一行新しい行を挿入して“入力モード”に移行する |

O	カーソルのある行の前に一行新しい行を挿入して“入力モード”に移行する
(Esc)	“Esc”キーを入力。“入力モード”を終了する
x	カーソルの場所の1文字を消去する
dd	カーソルのある行を1行消去する
D	カーソルの有る位置から行末までを消去する
dG	カーソルのある行からテキストの終わりまでの行を消去する。
<u>n</u> dG	テキストの先頭から n 行目ーカーソルのある行までを消去。
yy	カーソルのある行をヤンク (コピー) する。
p	消去もしくはヤンクした文字列を貼付ける。
<u>n</u> r <u>x</u>	カーソルの位置から n 文字を x で置き換える。n=1 の時(カーソルのある位置の1文字だけを置き換える)時は、n を省略できる。

カーソル移動

h	カーソルを1文字左に移動(カーソルキー←でもできる)
j	カーソルを1文字下に移動(カーソルキー↓でもできる)
k	カーソルを1文字上に移動(カーソルキー↑でもできる)
l	カーソルを1文字右に移動(カーソルキー→でもできる)
<u>n</u> G	n 行目に移動
G	テキストの末行に移動
\$	カーソル行末に移動
^	カーソル行頭に移動
%	(), { }, [] などの上でキーを押すと対応する括弧に移動する

コマンドの取り消し、繰り返し

u	undo。一つ前に実行したコマンドを無効にする。
(Ctr)+R	コントロールキーを押しながら R を押す。u で undo したことをもう一度やり直す。
.	一つ前に実行したコマンドをもう一度繰り返す。

ファイルの保存 + vi の終了

ZZ	現在のファイルを保存して、同時に vi を終了する。
----	----------------------------

練習 1

以下のコマンドを実行すると、schedule10 というファイルに 10 月のカレンダーが出来る。できた schedule10 というファイルに次の i) から iv) を実行してみよう。

```
$ cal > schedule10
```

- i) vi で schedule10 を開いてみる
- ii) この実習のある日に印”*” を付けてみる (カーソル移動+一文字置換)
- iii) 自分で 11 月のカレンダーを付け加えて見る (文字入力モード)
- iv) 10-11 月のカレンダーをセーブする (ファイルの保存+vi の終了)

文字列検索

/ /の後入力した文字列を探してその位置にカーソルを移動する。n で次の一致する文字列に、N で一つ前の一致する文字列に移動する。

ed コマンド

: これを入力すると。カーソルがウィンドウの下に移動し、以下のコマンドが実行できる。(Esc)でこのモードを終了できる。

(文字列置換)

1, \$s/string/STRING/g [テキスト中の文字列”string”をすべて”STRING”に置き換えることができる。文字列にはワイルドカードを使うことができる。(ワイルドカード: . = 任意の 1 文字、* = 任意の文字列, ^ = 行の頭、\$ = 行の終わり)]

(ファイルの保存)

w! [現在のテキストを現在のファイルに上書き保存する]

w name [ファイル名 name のファイルに上書き保存する]

q! [ファイルを保存せずに vi を終了する]

(文字列置換)

set number [すべての行に行番号を表示する]

set nonumber [行番号を非表示にする]

set nohlsearch [検索にマッチした文字列が high light されないようにする]

練習 2

以下のようにすると 2007 年 1 年分のカレンダーができる。できた schedule2007 というファイルに次の i) から iv) を実行してみよう。

```
$ cal 2007 > schedule2007
```

- i) schedule2007 を vi で開いてカレンダーに縁取りをつけてみる (置換)
- ii) 特定の日付を検索してカーソルを移動してみる (文字列検索)
- iii) 特定の日付 (例えば 13 日) だけ塗りつぶす (検索置換)
- iv) 中旬の日付だけを塗りつぶす (検索置換)

4 csh の使いかた入門

csh(シーシェルと読む)は UNIX 系のコマンドインタプリタ(人間が打ち込むコマンドを解釈してコンピュータに実行させるプログラム)のことである(UNIX 系には他にも sh、ksh、bash などの類似のシェルがある)。csh スクリプトというのは、コマンドをまとめてシェルに実行させるための、一種のプログラムである。比較的簡単に書けて、まとまった仕事をさせることができるので、ちょっとした csh スクリプトが書けるようになると、とても便利である。

例えば *file1* と *file2* という二つのファイルがあったとき、これらの中身を入れ替えてみる。既に習った unix コマンドを 3 つ以下のように実行すれば入れ替えができる。

```
$ cp file1 filet
$ mv file2 file1
$ mv filet file2
```

以下の内容のファイル(これはすごく簡単なシェルスクリプトである)を作って実行すれば、これと同じことを一回でやることができる。

```
fileswap1
```

```
#!/bin/csh -f
cp file1 filet; mv file2 file1; mv filet file2
```

```
$ csh fileswap1
```

上の例では *file1*、*file2* という名前のファイルしか使えないが、引数をつかえばどんな名前のファイルでも入れ替えるシェルができる。

```
fileswap2
```

```
#!/bin/csh -f
cp $1 file1; mv $2 $1; mv file1 $2
```

下の様に *fileswap2* を実行すると、\$1 (一番目の引数) に *file1* が、\$2 (二番目の引数) に *file2* が代入されるので、*fileswap1* と同じように *file1* と *file2* の中身が入れかわる。

```
$ csh fileswap2 file1 file2
```

練習 3

ファイル名を 3 つ与えて中身を入れ替えるシェルを作ってみよう。これは以下の 4 つのコマンドを実行するシェルを書くことに相当する。

```
$ cp file1 file1
$ mv file2 file1
$ mv file3 file2
$ mv file1 file3
```

5 ループを使って何回も同じ仕事をさせる為のシェル

ホモロジーサーチをかける時など、何十回も同じ作業をするときはループを使ったシェルを使うと便利である。

```
loopcopy1
```

```
#!/bin/csh -f
@ i = 2
while ($i <= 10)
    @ j = $i - 1
    cp file$j file$i
    @ i++
end
```

i というシェル変数に 2 を代入する
 i (シェル中で参照するときは \$i と書く) が 10 以下の内は、
 while と end の間を繰り返す
 j に i-1 を代入
 file をコピー
 i を一つ増やす

下のシェルは *loopcopy1* を改造して、一万回ループしながら現在何番目のループかを表示し、ループが終わったらメールをして知らせる。

loopcopy2

```
#!/bin/csh -f
date > time_stp ←
@ i = 1
while ($i <= 10000) ← 一万回ループ
    @ j = $i + 1
    echo "This is $i th loop. Next $j" ← 現在の回数を表示
    @ i++
end
date >> time_stp
mail your@mail.address -s finished < time_stp ← 終わったらメール
```

練習 4

loopcopy2 を改造し、タイムスタンプ(**date**)を順次 *filen* (n=1~10000)に追加し、さらに古いファイルを消す(*filen+1* を作ったら *filen* を消去) 様に見よう。

練習の解答例

練習 1

- i) vi で schedule10 を開いてみる

```
$ vi schedule10
```
- ii) この実習のある日に印”*” を付けてみる (カーソル移動+一文字置換)
 印を付けたい位置にカーソルを移動して→r*
- iii) 自分で 11 月のカレンダーを付け加えて見る (文字入力モード)
 テキストの末尾にカーソルを移動して→i (または a, または o)→入力
- iv) 10-11 月のカレンダーをセーブする (ファイルの保存+vi の終了)
 入力モードから抜けて(Esc)→ZZ

練習 2

- i) schedule2007 を vi で開いてカレンダーに縁取りをつけてみる (置換)

```
$ vi schedule2006
```

 テキスト先頭で 0
 #####など入力
 テキスト末尾で o

#####など入力

```
:1,$s/^/# /g
```

```
:1,$s/$/ #/g (末尾はずれるので適当に修正)
```

ii) 特定の日付を検索してカーソルを移動してみる (文字列検索)

```
/15
```

iii) 特定の日付 (例えば 13 日) だけ塗りつぶす (検索置換)

```
:1,$s/13/XX/g
```

iv) 中旬の日付だけを塗りつぶす (検索置換)

```
:1,$s/1[0-9]/XX/g
```

練習 3

シェル変数を使わない例

```
#!/bin/csh -f
cp file1 file1; mv file2 file1; mv file3 file2; mv file1 file3
```

シェル変数を使う例

```
#!/bin/csh -f
cp $1 file1; mv $2 $1; mv $3 $2; mv file1 $3
```

練習 4

```
#!/bin/csh -f
date > time_stp
@ i = 1
while ($i <= 10000)
    @ j = $i + 1
    mv file$i file$j
    date >> file$j
    @ i++
end
date >> time_stp
mail your@mail.address -s finished < time_stp
```