エディタとスクリプト

2019/02/21

作業場所

- 以降の作業は、以下のディレクトリで行います。

```
~/data/2_editor/
```

cd コマンドを用いてディレクトリを移動し、

pwd コマンドを利用して、カレントディレクトリが
上記になっていることを確認してください。

テキストエディタ

- 簡単なコマンドを連続して行いたいときには、シェルスクリプトを書くと便利である。
- シェルスクリプトは中身にUnixコマンドを書いたテキストファイルである。
- Windowsの「メモ帳」や、MacOSの「テキストエディット」など、GUIでテキストファイルを作成する機会は多い。
- リモートログイン先等、GUIが使えないこともある。その場合でも便利に使用できるのが、Unixでも使用できるテキストエディタである。

Unixで使えるテキストエディタ

- **emacs**
  - 「Control+文字キー」または「Esc+文字キー」によってカーソル移動等の操作を行う
  - 「Esc x コマンド」によって、豊富なコマンドを利用可能
- **vi (vim)**
  - 文字入力モードとコマンド入力モードの2つのモードを切り替えて使う
  - Unixでは最も標準的なエディタ

本講で扱うコマンド

- `$ emacs`　　Emacsエディタの起動

Emacsを扱う際の注意

- マウスは使えません。

　メモ帳やテキストエディットとは異なり、
マウスを使用しようとしてもできません。
（そのため、範囲選択などもできません）

→代わりにキーボードを使用します。

Emacs エディタのコマンド入力

- コントロール＋文字キー
  - コントロールキーを押しながら文字キーを押す
  - C-x（Control+x）などと表記
- エスケープ+文字キー （メタキー）
  - Esc キーを押してから文字キーを押す
  - M-x (Meta+x)などと表記

Emacs の起動と終了

- 起動
  `$ emacs [ファイル名]`

  (存在しないファイル名を指定すると、
  その名前のファイルを作成する)
- 終了
  - C-x C-c 保存するか聞かれるのでy (yes)またはn (no)と入力して終了
  - C-z 編集を中断してシェルに戻る。`fg` で再開

ファイルの読み込みと保存

- ファイルの読み込み
  - C-x C-f　で読み込むファイル名を入力
- ファイルの保存
  - C-x C-s 現在のファイル名で保存
  - C-x C-w 別名で保存

実習

1） `$ emacs` と入力し、emacs を起動してください。

2） emacs を終了してシェルに戻って下さい。

3） `$ emacs text.txt`
と入力し、emacs でファイルを読み込んでください。。

4） C-x C-w を使用し、このファイルを別名　(text2.txt) で保存してください。

5） emacs を終了してシェルに戻り、さきほど保存した名前のファイルがあることを確認してください。

Emacs の基本的なコマンド

カーソル移動

- C-p（または ↑ ）上に移動
- C-n（または ↓ ）下に移動
- C-b（または←）左に移動
- C-f （または→）右に移動
- C-a 行の先頭に移動
- C-e 行の最後に移動
- C-v 1ページ分下に移動
- M-v 1ページ分上に移動
- M-< ファイルの先頭へ移動
- M-> ファイルの最後へ移動
- C-SPC 現在の位置をマーク
- C-x C-x マーク位置へ移動（カーソル位置との入れ替え）

編集と検索

- DEL 左隣の文字を削除
- C-d カーソルの文字を削除
- C-k 現在の行のカーソル以降を削除してカットバッファへ
- C-w マーク位置からカーソル位置の間を削除してカットバッファへ
- C-y カットバッファの内容をペースト
- C-s 順方向に文字列検索
- C-r 逆方向に文字列検索
- C-_ 取り消し(Undo)
- C-g コマンド入力のキャンセル

カーソル移動

			M-< ファイル先頭へ			
			M-v 1ページ上へ			
			C-p（↑） 1つ上へ			
C-a 行の先頭へ		C-b（←） 1つ左へ		C-f（→） 1つ右へ		C-e 行の末尾へ
			C-n（↓） 1つ下へ			
			C-v 1ページ下へ			
			M->>ファイル最後へ			

編集

[DEL]	左隣の文字を削除
C-d	カーソル位置の文字を削除

C-@ または C-[SPACE]	現在の位置をマーク
C-w	マークした位置からカーソル位置までを削除してカットバッファへ （＝カット）
M-w	マークした位置からカーソル位置までをコピーしてカットバッファへ （＝コピー）
C-y	カットバッファの内容をペースト

検索・アンドゥ・キャンセル

C-s	順方向に文字列検索
C-r	逆方向に文字列検索
C-_ または C-u	直前の編集を取り消し（アンドゥ）
C-g	コマンド入力のキャンセル （例：C-sによる検索の中断）

シェルスクリプト

シェルスクリプトとは？

- シェルスクリプトとは、あらかじめ実行したい一連のコマンドを記述したテキストファイルである。
- 実行権を持たせることによってシェルから実行できる。
- 実行するコマンド群を記録しておいて再利用することができる。
- 記述したスクリプトを変更することによって、作業を簡略化できる。

シェル

- シェルは、OSのユーザーに対してカーネルへの操作機能を提供するためのソフトウェアである。
- 通常は　/bin/sh　を指す。
- OSによって指しているシェルの種類は様々。
  例）MacOS : bash
  　　CentOS : /bin/bash へのシンボリックリンク
  　　FreeBSD : ash

シバン (shebang)

- シェルスクリプトのファイルには、先頭の行に下記のような記述が見られる。

```
#!/bin/sh
```

- 「#!」をシバンと呼ぶ。
- シバンで指定されるパスにより、2行目以降の文字群を実行するべきプログラム（インタプリタ）を指定する。

```
例）Perlの場合
#!/usr/bin/perl
```

復習:コマンド入力

- 以下のコマンドを順番に入力してください。

```
grep 'GO' 1433T_HUMAN.sprot > line.tmp
```

```
echo 'GO matchs;'
```

```
wc -l line.tmp
```

```
rm line.tmp
```

シェルスクリプト例

- 実際にシェルスクリプトを見てみよう
  （以下、シバンは省略します）

example1.sh

```
grep 'GO' 1433T_HUMAN.sprot > line.tmp
echo 'GO matchs;'
wc -l line.tmp
rm line.tmp
```

演習）
$ ./example1.sh
と入力し、このシェルスクリプトを実行せよ。

変数

- シェルでは「変数」というものを使用できる。
- プログラマーが指定したある名前の「変数」に、プログラムの実行状況によって、文字列や数値などさまざまなデータを記憶しておく。

```
name=yamada
```

変数 name に ‘yamada’ という文字列（値）を記憶させる。
変数、イコール、入れる値の間にスペースは入れない。

```
echo ${name}
```

変数の中の値を呼び出すには、$を使う。
$のあとに続く文字列が変数であることを示すため、
{} を使用して区別するとよい。

クォーテーションの違い

- シングルクォーテーション［'］
  中身は単に文字列として扱われる。
  シェルに解釈されたくない場合に有効。
- ダブルクォーテーション ［"］
  中身に変数がある場合、シェルはその中の値を採用する。

```
name='yamada'
echo '${name}'   // ${name} と表示される。
echo "${name}"   // yamada   と表示される。
```

変数の使用

- 変数を使用したシェルスクリプトの例を見てみよう。

example2.sh

```
param='GO'
grep ${param} 1433T_HUMAN.sprot > line.tmp
echo "${param} matchs;"
wc -l line.tmp
rm line.tmp
```

演習）
$ ./example2.sh
と入力し、このシェルスクリプトを実行せよ。

変数の変更

- 先程の example2.sh では、“GO”という文字列でしか検索できない。他の文字列を検索したい場合、変数の中身を変えれば良い。

example3.sh

```
param='DE'
grep ${param} 1433T_HUMAN.sprot > line.tmp
echo "${param} matchs;"
wc -l line.tmp
rm line.tmp
```

演習）
example2.sh をEmacs で開き、上のようにparam= の値を編集した後、
example3.sh という別名で保存せよ。 example3.sh を実行すると何が起きるか？

スクリプトの実行権

先ほどemacsで作成した example3.sh を実行してみよう。

```
$ ./example3.sh
```

“Permission denied” と出ただろうか？
ファイルに実行権がない場合、このようなエラーメッセージが出る。
chmodコマンドを使用して、実行権を与えよう。

```
$ chmod +x example3.sh
$ ./example3.sh
```

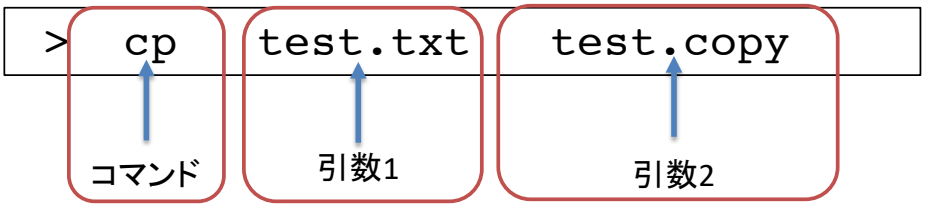
実行結果はどうなるか？

（参考） 変数と引数

- 先ほどの例では、変数に入れる値としてGOという文字列を使用した。
- しかし、別の文字列を使用したい場合、毎回スクリプトを直さないといけない。
- そのような手間を省くために、引数を使用することもできる。

（演習） 引数

- 引数とは、コマンド実行時にコマンドラインから渡される値である。



- シェルスクリプトの中でも引数を利用できる。その場合、スクリプト内で$1などと表記する。

```
$1 1番目の引数 $2 2番めの引数 … $9 9番目の引数
$0 コマンド自身 $* 引数全部 （などなど）
```

（演習） 引数の使用

- 検索に使用する文字列を引数として使用するシェルスクリプトを見てみよう。

example4.sh

```
#!/bin/sh
param=$1
grep ${param} 1433T_HUMAN.sprot > line.tmp
echo "${param} matchs;"
wc -l line.tmp
```

このスクリプトは、引数を1つとる。引数は 変数 param に代入される。
このスクリプトを実行するためには、下記のような文法で入力を行う。

$ ./example4.sh GO