システム工学概論

2. シェルプログラミングの実践

山田泰司

taiji@aihara.co.jp

株式会社あいはら 研究開発チーム

シェルプログラミングに必要なツール

- **シェルスクリプトを書くためのエディタ** (**例えば、**GNU emacs)
- Bourne shell(sh, ksh, bash, zsh)
- **その他、様々なコマンドラインプログラム** (**例えば、**GNU coreutils)
  - ⇨ shellutils(basename, expr, false, test, true **など**)
  - ⇨ fileutils(chmod, mkdir, mv, rm, touch **など**)
  - ⇨ textutils(cat, sort, uniq **など**) **や** GNU sed

対立するツール

- csh(tcsh) **… これは学ぶ必要はありません**
- Microsoft command.com, cmd.exe, PowerShell **など**

シェルスクリプトの利点

- オペレーション環境がシェルであれば、もし、定型的な作業が発生した場合に、その作業を自動化、かつ、一般化することがシェルプログラミングにで容易にできる。
- また、正しく動作するシェルスクリプトを一度書いてしまえば、手作業で一連のコマンドを打鍵するよりも、間違いが無く目的を遂行できるようになる。
- Bourne shell スクリプトは、ほとんどのプラットフォームで同一に動作する。但し、真のマルチプラットフォームのためには、多少の方言や外部コマンドラインプログラムの有無などに配慮する必要はある。
- makefile, configure スクリプト、プログラミング言語 C からの外部アプリケーションの呼び出し等、シェルプログラミングが有効な場面が多く存在し、応用範囲が広い。

（例題０）シェルプログラミングの基礎

以下のような複数のファイルがあるものとする。

```
$ cat A.txt
This is A.txt.
これは A.txt です。

$ cat B.txt
This is B.txt.
これは B.txt です。
          :
```

これらすべてのテキストファイルについて文字列「A.txt」を「"A.txt"」、「B.txt」を「"B.txt"」(以下同様) のように置換せよ。

但し、まずはこの複数の例題用ファイルを自動的に作成せよ。

（例題０）の解答例ーシェルプログラミングの基礎０

for **文、ヒアドキュメント：**

```
for f in A.txt B.txt C.txt D.txt E.txt; do
  cat <<EOF > "$f"
This is $f.
これは $f です。
EOF
done
```

この手続きを「doexercise0.sh**」へ書き記したとすると、実行例は以下のようになる。**

```
$ sh doexercise0.sh
```

その結果、複数の例題用ファイルが自動的に作成される。

（例題０）の解答例ーシェルプログラミングの基礎１

ストリームエディタ sed **と正規表現：**

```
for f in *.txt; do
  sed -e 's|\(.\.txt\)|"\1"|' "$f" > "$f.new"
  mv -f "$f.new" "$f"
done
```

この手続きを「doexercise1.sh**」へ書き記したとすると、実行例は以下のようになる。**

```
$ sh doexercise1.sh
```

その結果、すべてのテキストファイルにおいて、文字列「A.txt**」が「**"A.txt"**」などのように置換される。**

（例題０）の解答の補足

先のシェルスクリプト「doexercise1.sh**」の** sed **命令はこのようにも書ける。**

```
        :
  sed -e 's| \([^ ]*\.txt\)| "\1"|' "$f" > "$f.new"
        :
```

但し、意味合いは相異なるので双方比較し、さらに他の書き方も考察してみよう。

（例題１）シェルプログラミングの実践１

以下のような２つのファイルがあるものとする。

```
$ ls
A.txt B.txt
```

これらのファイル名を「交換」するシェルスクリプトを書け。

（例題１）の解説

例題は、通常、コマンドラインで以下のようにオペレーションすることに他ならない。

```
$ mv A.txt A.txt.tmp
$ mv B.txt A.txt
$ mv A.txt.tmp B.txt
```

GUI で操作する場合も、ファイルが同一フォルダに格納されている場合は、やはりファイル名の交換には、ひとつのファイルの退避を要する。

（例題１）の解答例

変数、引数：
以下のようなシェルスクリプト「swapname」を作成する。

```
#!/bin/sh
tmpname="$1.tmp"
mv "$1" "$tmpname"
mv "$2" "$1"
mv "$tmpname" "$2"
```

そして、そのファイルに実行権を付ける。

```
$ chmod +x swapname
```

実行例は以下のようになる。

```
$ ./swapname A.txt B.txt
```

（例題２）シェルプログラミングの実践２

以下のような n 個のファイル群があるものとする。

```
$ ls
A.txt B.txt C.txt D.txt E.txt ...
```

これらのファイルの内容を変えずにファイル名を「逆順」にするシェルスクリプトを書け。

（例題２）の解説

n が偶数の場合、

```
$ ls
A.txt B.txt C.txt D.txt
$ ./swapname A.txt D.txt
$ ./swapname B.txt C.txt
```

n が奇数の場合、

```
$ ls
A.txt B.txt C.txt D.txt E.txt
$ ./swapname A.txt E.txt
$ ./swapname B.txt D.txt
```

のように、先ほどの「swapname」シェルスクリプトをオペレーションすることに他ならない。

つまり、ファイル名交換が、引数の数の半分の回数の繰り返しとなる。

（例題２）の解答

手続き（関数）、while **文、**eval**：**
以下のようなシェルスクリプト「swapnames**」を作成する。**

```
#!/bin/sh
swapapair(){
  tmpname="$1.tmp"
  mv "$1" "$tmpname"; mv "$2" "$1"; mv "$tmpname" "$2"
}
l=1
u=$#
while [ $l -lt $u ]; do
  eval lfile="\${$l}"
  eval ufile="\${$u}"
  swapapair "$lfile" "$ufile"
  l=`expr $l + 1`
  u=`expr $u - 1`
done
```

実行例は以下のようになる。

```
$ ./swapnames A.txt B.txt C.txt D.txt E.txt
```

問題の一般化とシェルプログラミング

よいプログラミングのコツ

- ▷ 問題を切り分けよ（ここではファイル名交換の手続き）
- ▷ 問題を一般化せよ（ここでは任意のファイル数へ拡張）
- ▷ 想定される使用場面に応じて、例外処理で事故を防げ

シェルプログラミングのコツ

- ▷ 失敗しても構わないテスト環境（想定するファイル群）を用意して「気軽に」
- ▷ 実行されるであろうコマンドを出力しながらシェルスクリプトを書いていく

（例題２）の解答の補足

先のシェルスクリプト「swapnames」を以下のように修正する。

```
#!/bin/sh
swapapair(){
  if [ -f "$1" -a -f "$2" ]; then
    name="`basename $1`"
    tmpname="`mktemp \"./$name.XXXXXX\"`"
    mv "$1" "$tmpname"; mv "$2" "$1"; mv "$tmpname" "$2"
  elif [ -f "$1" -a ! -f "$2" ]; then
    mv "$1" "$2"
  elif [ ! -f "$1" -a -f "$2" ]; then
    mv "$2" "$1"
  fi
}
        :
```

すると、万が一存在しないファイルを指定したとしてもエラーとはならない。

また、「～.tmp」が既に存在した場合、それが失われずに正常に動作する。

*（例題２）の解答の補足

先のシェルスクリプト「swapnames」をさらに以下のように修正する。

```
          :
u=$#
if [ $# = 0 ]; then
  cat <<EOF
$0 - swap filename(s)
usage: $0 file1 file2 ...
EOF
  exit
fi
while [ $l -lt $u ]; do
          :
```

すると、引数を指定しない (意図しない) 使い方をした場合に、簡単な使用方法が表示される。