情報工学実験
スクリプトプログラム

担当教員名：
氏名：澤岨　千明　　学籍番号：075730G

実験日：5 月 16 日（金）
提出期限：5 月 23 日
提出日：5 月 23 日 (金)

1　level1

以下の機能を満たすシェルスクリプト（level1.sh）を作成せよ。
　a=6,b=3 とし、四則演算を計算する。

level1.sh

```
#!/bin/sh
#四則演算。level1.sh
a=6
b=3

# a=6,b=3 の表示
c="a=$a,b=$b"
echo $c

# a+b
d=`expr $a + $b`
echo a+b=$d

# a-b
e=`expr $a - $b`
echo a-b=$e

# a*b
f=`expr $a * $b`
echo a*b=$f

# a/b
g=`expr $a / $b`
echo a/b=$g
```

実行結果

```
sh-2.05b$ ./level1.sh
a=6,b=3
a+b=9
a-b=3
a*b=18
a/b=2
```

考察

　単純な四則計算を行うスクリプト。#行はコメントである。
最初に a=6, b=3 を入れておき、c に"a=$a,b=$b"を代入することで、表示させている。ここでの a , b はただの記号であり、$a, $b が上記の a, b の値を持っている。
　d では expr コマンドを使って計算を行っている。 e, f も同様である。

2　level2

以下の機能を充たすシェルスクリプト（level2.sh）を作成せよ。
　level1.sh を拡張（複製して新しくファイルを作成）する。
　level1.sh では演算対象の a,b をスクリプト内で設定していたが、
　この２つの値を実行時の引数から設定するようにする。

level2.sh

```
#!/bin/sh
#特殊変数の利用。 level1.sh の拡張。level2.sh

echo "level2.sh の実行結果"

# a,b
a=$1
b=$2
c="a=$a,b=$b"
echo  $c

# a+b
d=`expr  $a +  $b`
echo a+b=$d

# a-b
e=`expr  $a -  $b`
echo a-b=$e

# a*b
f=`expr  $a *  $b`
echo a*b=$f

# a/b
g=`expr  $a /  $b`
echo a/b=$g
```

実行結果

```
sh-2.05b$  ./level2.sh 8 2
level2.sh の実行結果
a=8,b=2
a+b=10
a-b=6
a*b=16
a/b=4
```

考察

level1.sh を拡張したものである。　引数として２つの値を入力し、その値を使った計算を行う。
a = $1, b=$2　は引数を代入している。これも level1 同様に計算を行っている。

3 level3

以下の機能を満たすシェルスクリプト（level3.sh）を作成せよ。
int 型の引数を 2 個取り、それぞれ int1, int2 として設定 (保存/代入) する。
引数が 2 個以外の場合（0 個，1 個や，3 個以上）には使い方を出力して終了する。
int1 と int2 を数値比較し、出力すること。

level3.sh

```
#!/bin/sh
#数値の比較。level3.sh

# 入力数が 2 か否か
if [ $# -eq 2 ] ; then
        int1=$1
        int2=$2
else
        echo " > ./level3.sh int1 int2 "
        exit 1
fi

# int1 = int2 同じ値の場合
if [ $int1 -eq $int2 ] ; then
        echo int1=$int1, int2=$int2
        echo "int1 is equal to int2 !"
        exit 1
fi

# int1 > int2 int1 が大きい場合
if [ $int1 -gt $int2 ] ; then
        echo int1=$int1, int2=$int2
        echo "int1 is greater than int2."
        exit 1
fi

# int1 < int2 int2 が大きい場合
if [ $int1 -lt $int2 ] ; then
        echo int1=$int1, int2=$int2
        echo "int1 is less than int2."
        exit 1
fi
```

実行結果

```
sh-2.05b $ ./level3.sh
> ./level3.sh int1 int2

sh-2.05b $ ./level3.sh 10 20
int1=10, int2=20
int1 is less than int2.

sh-2.05b $ ./level3.sh 10 10
int1=10, int2=10
int1 is equal to int2 !

sh-2.05b $ ./level3.sh 20 10
int1=20, int2=10
int1 is greater than int2.
```

考察

数値の比較を行うスクリプトである。

最初に引数が２つあるかどうかを判断する。ここに出ている $ # とは引数の総数であり、-eq は左右に示されている値が等しいかどうかを判断するものである。ここでは、引数が２つ入力されていなければ、再度入力するようにコメントがでるようになっている。

-gt は int1 が int2 より大きいかどうかを判断するものであり、 -lt は int1 が int2 より小さいかどうかを判断するものである。

4 level4

以下の機能を満たすシェルスクリプト（myls.sh）を作成せよ。
指定したディレクトリの内容を表示するコマンド myls.sh を作成せよ。
指定されたディレクトリが存在しない場合には、
「Not found: $dir」等のようにエラーを出力して終了する事。
指定されたディレクトリ内に存在するディレクトリと通常のファイルは，出力を分けて表示すること。
ディレクトリの後ろには ”/” を表示すること。

myls.sh

```
#!/bin/sh
# level4 myls.sh
# ディレクトリ内容を出力。

# $1 はディレクトリか
if [ -d $1 ] ; then
        dir=$1
else
        echo ”Nothing...”
                exit 1
fi

# $dir はファイルか
if [ -f $1 ] ; then
        echo ”Not directory”
        exit 1
fi

files=‘ls -1 $dir‘
for file in $files
do
        if [ -d $dir/$file ] ; then
                echo $file/
        fi
done

for file in $files
do
        if [ -f $dir/$file ] ; then
                echo $file
        fi
done
```

実行結果

```
sh-2.05b $ ./myls.sh /Users/e075730/source/
dir1/
dir2/
LaTex.rtf
level1.sh
level2.sh
level3.sh
level4-a.sh
myls.sh
mymv.sh
myupper.sh
実験シェルスクリプト
実験シェルスクリプト.rtf

sh-2.05b $ ./myls.sh /User/e075730/source/
Nothing...
```

考察

指定したディレクトリの内容を出力するスクリプトである。

[-d $1] の -d はディレクトリかどうかを判断し、[-f $1] の -f はファイルか否かを判断するものである。引数として入力された内容がディレクトリな場合、 files=‘ ls -1 $dir‘ よりそのディレクトリの一覧を取得する。そしてそのディレクトリ内のファイル名が file としてへ入る。そして、一つ一つディレクトリか否かと判断し、それによって$file/と出力する。

5　level5

以下の機能を満たすシェルスクリプト（myupper.sh）を作成せよ。
　指定されたファイル名を小文字から大文字に変換するスクリプト myupper.sh を作成せよ。

myupper.sh

```
#!/bin/sh
# level5 myupper.sh
#指定されたファイル名を小文字から大文字へ変換

# 引数が 1 より小さい時 -lt は ＜ のときの評価式
if [ $# -lt 1 ] ; then
        echo ” ＞ ./myupper.sh file1 file2 ... ”
        exit
fi

# 引数の入力を得ての動作
for arg in $*
do
        if [ -f $arg ] ; then
                filename=‘echo $arg — tr [a-z] [A-Z]‘
                mv $arg $filename

# ファイルが無い場合の動作
        else
                echo ”Nothing...”
        fi
done
```

実行結果

```
sh-2.05b$ ls
file1.txt file2.tex file2.txt file3.dat file3.txt myupper.sh myupper.sh
sh-2.05b$ ./myupper.sh
＞ ./myupper.sh file1 file2 ...
sh-2.05b$ ./myupper.sh file1.txt
sh-2.05b$ ls
FILE1.TXT file2.tex file2.txt file3.dat file3.txt myupper.sh myupper.sh
```

考察

　指定されたファイル名を小文字から大文字に変換するスクリプトである。
　この時、ファイル名として入力された引数が 1 以下である場合は、正しい入力法というコメントが出力される。ここでてる $* とはスクリプトに手渡された引数全体のリストである。ここでは引数は 1 つ以上あれば実行できるので、この変数が必要となる。
　引数を得た後、それがファイルならば filename より小文字が大文字に変換され、mv によって名前が書き換えられる。

ここで、オプションとして大文字を小文字に変換するスクリプトを次に表す。

mylower.sh

```
#!/bin/sh
# level5 option mylower.sh
#指定されたファイル名を大文字から小文字へ変換

# 引数が 1 より小さい時 -lt は < のときの評価式
if [ $# -lt 1 ] ; then
        echo " > ./mylower.sh file1 file2 ... "
        exit
fi

# 引数の入力を得ての動作
for arg in $*
do
        if [ -f $arg ] ; then
                filename=`echo $arg — tr [A-Z] [a-z]`
                mv $arg $filename

# ファイルが無い場合の動作
        else
                echo " Nothing ... "
        fi
done
```

実行結果

```
sh-2.05b$ ls
FILE1.TXT file2.txt level1.sh  level3.sh  mylower.sh myupper.sh
daily.eps file3.dat level2.sh level7-1.sh myls.sh myupper.sh
daily.png file3.txt level2.sh  level7.sh myls.sh
file2.tex level1.sh level3.sh level7.sh  mymv.sh

sh-2.05b$ ./mylower.sh FILE1.TXT
sh-2.05b$ ls
daily.eps file2.txt level1.sh  level3.sh  mylower.sh myupper.sh
daily.png file3.dat level2.sh level7-1.sh myls.sh myupper.sh
file1.txt file3.txt level2.sh  level7.sh myls.sh
file2.tex level1.sh level3.sh level7.sh  mymv.sh
```

考察

動作は myupper.sh と同じであるが、異なる点は filename の中のみである。
myupper.sh では　tr [a-z]　[A-Z] とあったが、ここでは tr [A-Z] [a-z] となっている。これによって大文字は小文字へと変換される。

6 level6

以下の機能を満たすシェルスクリプト（mymv.sh）を作成せよ。
第１引数で指定された拡張子を持つファイルに対し、
拡張子を第２引数に変更するスクリプト mymv.sh を作成せよ。

mymv.sh

```
#!/bin/sh
# level6 mymv.sh
#ファイルの拡張子を変更

#引数が 2 である時
if [ $# -eq 2 ] ; then
        name1=$1
        name2=$2
else
        echo " > mymv.sh name1 name2"
        exit 1
fi

files=`ls -1 *.$name1`

for file in $files
do
        newfile=`basename $file .$name1`
        mv $file $newfile.$name2
done
```

実行結果

```
sh-2.05b$ ls
daily.eps file2.txt level1.sh  level3.sh  mylower.sh mymv.sh
daily.png file3.dat level2.sh level7-1.sh myls.sh myupper.sh
file1.txt file3.txt level2.sh  level7.sh myls.sh  myupper.sh
file2.tex level1.sh level3.sh level7.sh  mymv.sh

sh-2.05b$ ./mymv.sh txt dot

daily.eps file2.tex level1.sh  level3.sh  mylower.sh mymv.sh
daily.png file3.dat level2.sh level7-1.sh myls.sh myupper.sh
file1.dot file3.dot level2.sh  level7.sh myls.sh  myupper.sh
file2.dot level1.sh level3.sh level7.sh  mymv.sh
```

考察

ファイルの拡張子を変更するスクリプトである。　ここでは引数を２つ必要としている。２つ入力されていなければ実行できないようになっている。
files=‘ ls -1 *.$name1‘　では最初に入力した引数を拡張子として見、ディレクトリ内のファイル名を取得している。そして newfile の basename によって $file . $name1 のファイルがとりだされ、次の行の mv によってファイルの拡張子は $name2　より、別の拡張子に書き換えられる。

7 level7

数値データをプロットする gnuplot スクリプトを作成せよ。
どのような数値でも構わないが、以下の条件を満足するものとする。
　数値データ数を１０件以上とする事。
　LaTeX の出力に含めるために、gnuplot の出力画像形式を EPS 形式とする事。
　軸の説明やグラフのタイトルを図内に含める事。

gnuplot.sh

```
#!/bin/sh
# level7 gnuplot.sh
# gnuplot グラフ。

# Level 4.3

#—— gnuplot 出力用スクリプトサンプル

# 出力画像の形式を設定．EPS の場合は 'set terminal postscript eps'
set terminal postscript eps

# 出力ファイル名を決定
set output "bird.eps"

# グラフのタイトルを設定
set title "The number of watched birds"

# x 軸のラベルを設定
set xlabel "Date (Dec 2006)"

# y 軸のラベルを設定
set ylabel "number"

# y 軸に 2 カラム目（日付），x 軸に 1 カラム目（アクセス数）をプロット
plot "level7-1.sh" using 2:1 with boxes
```

level7-1.sh

```
7 06/Dec/2006
1 08/Dec/2006
5 09/Dec/2006
2 10/Dec/2006
0 13/Dec/2006
2 15/Dec/2006
0 17/Dec/2006
8 21/Dec/2006
3 22/Dec/2006
5 25/Dec/2006
12 30/Dec/2006
```

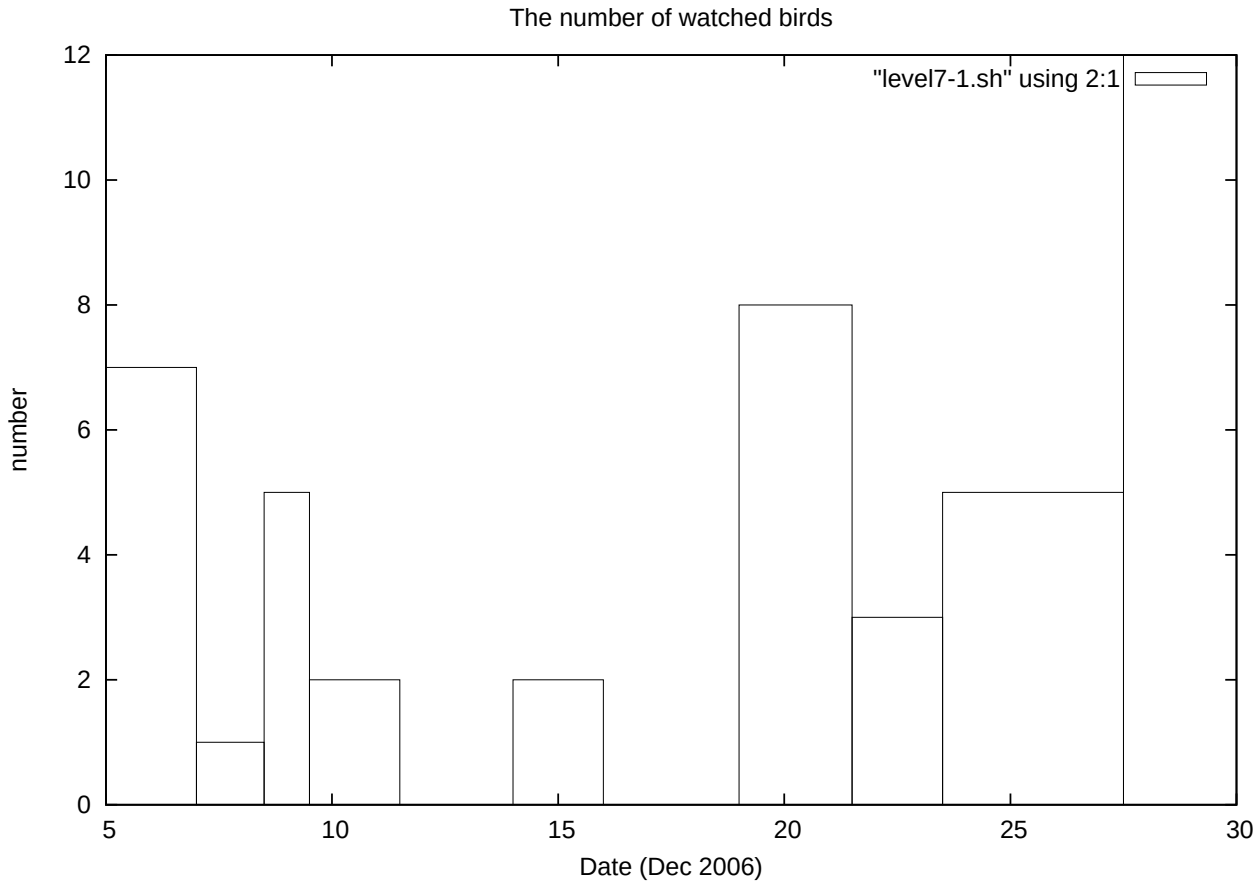

Figure 1: 鳥を見た数とその日付

実行結果

考察

gnuplot.sh を実行し、level7-1.sh の数値データを gnuplot を使いグラフにするスクリプトである。

ここで ”set terminal postscript eps ”と入力してあるのは出力を eps という拡張子にするため。LaTeX はこの拡張子のファイルを取り扱うためにこの拡張子にする必要があるからである。動作内容はスクリプト内に書かれているので割愛する。　私は level7-1.sh に、鳥を見た数とその日付のデータをという内容を作った。

参考 URL

http://itpro.nikkeibp.co.jp/article/COLUMN/20070824/280327/
http://www.atmarkit.co.jp/flinux/index/indexfiles/shellsindex.html