情報工学実験 Report3

学籍番号 045713C：大城 和也

平成 17 年 5 月 17 日

1 Level 1

指定したディレクトリ・ファイルの内容を表示するコマンド myls を作成せよ。但し、ディレクトリと通常のファイルは分けて表示しディレクトリの後ろには”/” を表示すること

1.1 myls.sh

```
#!/bin/sh

if [ $# -ne 1 ] ; then   #引数が一つかどうかを判断
 if [ $# -eq 0 ] ; then  #引き数がなかった場合
  echo
 else #引き数が二つ以上のとき
  echo choice the one directory
 fi

else
 # 引き数が一つでさらにディレクトリであるとき
 if [ -d "$1" ] ; then

 # 引き数を代入、ディレクトリ内の表示を files に代入
 dir=$1
 files=‘ls -1 $dir‘

 for directory in $files ;
 do
# ディレクトリである場合
  if [ -d $dir/$directory ] ; then
# ディレクトリの表示
```

```
    echo $directory/
  fi
 done

 for directory in $files ;
 do
# ディレクトリでない場合
  if [ ! -d $dir/$directory ] ; then
# ファイルの表示
    echo $directory
  fi
 done
 else
  echo 引き数がディレクトリではありません。
 fi
fi
```

1.1.1 実行結果

```
[Kazuya-OSHIRO:~] j04013% ./myls.sh robocode
battles/
compilers/
javadoc/
robots/
templates/
compiler.properties
license.html
robocode.bat
robocode.ico
robocode.jar
robocode.properties
robocode.sh
robot.database
versions.txt
window.properties
```

1.2 考察

このシェルスクリプトはディレクトリを引数にとりそのディレクトリ内を表示するものである。表示する際、ディレクトリを先に表示するようにして

いる。

2 level 2

ファイル名を一括で小文字から大文字 (myupper)、拡張子の変更 (mymv) が可能なコマンド myrename を作成せよ。

2.1 myupper.sh

```
#!/bin/sh

#全ての引き数を取得
files="$*"

if [ $# -eq 0 ] ; then
 echo ’引数がありません’
fi

#引き数の数だけ繰り返す
for filename in $files
do
 #変換してその文字列を代入
 newnames=‘echo $filename | tr ’[a-z]’ ’[A-Z]’‘
 #mv でファイル名を変更
 mv $filename $newnames
 #変更したことを表示
 echo "$filename -> $newnames"
done
```

2.1.1 実行結果

```
[Kazuya-OSHIRO:~/a] j04013% ls
euro_rock.jpg  mymv.sh*  myupper.sh*  test.txt  test2.txt
[Kazuya-OSHIRO:~/a] j04013% ./myupper.sh *.txt
test.txt -> TEST.TXT
test2.txt -> TEST2.TXT
[Kazuya-OSHIRO:~/a] j04013% ls
TEST.TXT  TEST2.TXT  euro_rock.jpg  mymv.sh*  myupper.sh*
```

2.2 mymv.sh

```
#!/bin/sh

# 引数が二つ以外のとき
if [ $# -ne 2 ] ; then
 echo 引き数は二つです
 echo 例 : ./mymv.sh jpg gif
fi

# after
after=$1
before=‘ls *.$2‘

for rename in before ;
do
 # 拡張子変更後の名前を格納
 name=‘basename $before .$2‘.$1
 # 拡張子の変更
 mv $before $name
done
```

2.2.1 実行結果

```
[Kazuya-OSHIRO:~/a] j04013% ls
TEST.TXT  TEST2.TXT  euro_rock.gif  mymv.sh*  mymv.sh~*  myupper.sh*
[Kazuya-OSHIRO:~/a] j04013% ./mymv.sh jpg gif
[Kazuya-OSHIRO:~/a] j04013% ls
TEST.TXT  TEST2.TXT  euro_rock.jpg  mymv.sh*  mymv.sh~*  myupper.sh*
```

2.3 考察

myupper は小文字を大文字にするものである。[a-z] というのは小文字アルファベット全てを意味し、同じように [A-Z] は大文字アルファベットを示している。これらを tr を用いることで文字を置き換え、それにより、この変換を行っている。

mymv は拡張し変換のシェルスクリプトであり、その内容は初めの引数を二つとり、一つめの引数を変更後の拡張子、二つ目の引数を変更前の拡張子

とする。basename をつかうことでファイルから拡張子を抜いたファイル名をとりだし、mv を用いて拡張子を変更する。

3 level 3

thumbnail3.sh をベースに、1 ページ内に納める写真数 (photo_num) を指定し、自動的にページ分割するように拡張せよ。

3.1 thumbnail_ex.sh

```
#!/bin/sh

# file: thumnail_ex.sh
# syonopsis: thumnail_ex.sh SUFFIX RESIZE
# comment: SUFFIX で指定した拡張子を持つ画像ファイルの
#          縮小画像を生成し,
#          表示する HTML ファイルを生成.

# 引数チェック、引き数が 3 つ以外の場合に表示する
if [ $# -ne 3 ] ; then
echo "Usage: $0 SUFFIX RESIZE"
echo "   RESIZE: percent"
echo "   PHOTO_NUM: number of Image"
echo "   example) thumnail3.sh jpg 30 4"
exit
fi

# リストアップ処理
SUFFIX=$1
RESIZE=$2
PHOTO_NUM=$3

# 同ディレクトリ内に存在する画像ファイルの数
# NUM=‘ls *.$SUFFIX | wc -l‘
# 画像ファイルのリストを変数に代入
files=‘ls *.$SUFFIX‘

#  縮小画像生成
## 縮小画像保存用の一時ディレクトリ生成
```

```
DIR=resized
if [ -d $DIR ] ; then
    echo " temporal directory ($DIR) exsits..."
else
    mkdir $DIR
    echo " temporal directory ($DIR) CREATED."
fi

## 一時ディレクトリに縮小画像を生成
cd $DIR
for filename in $files
do
  convert -resize $RESIZE%x$RESIZE% ../$filename ${filename}_s.$SUFFIX
  echo "convert $filename to $DIR/${filename}_s.$SUFFIX"
done
cd ..

# 生成する HTML ファイルの数
PAGES=1
# html を作成させるためのフラグ
FLAG=1
#表示した画像数
IMGS=1

#縮小したファイルのリストを変数に代入
resized_files=‘ls $DIR/*_s.$SUFFIX‘

#HTML ファイルを生成するループ
for filename in $resized_files
do
if [ $FLAG -eq 1 ] ; then
# HTML ファイル生成
## HTML 開始
echo "<html>" > page$PAGES.html
echo "<head>" >> page$PAGES.html
echo "<title>thumbnail page$PAGES.html</title>" >> page$PAGES.html
echo "</head>" >> page$PAGES.html
echo "<body>" >> page$PAGES.html
FLAG=0
```

```
fi

#画像を表示するタグを生成
echo "<img src=\"$filename\">" >> page$PAGES.html

#表示した画像数が指定した画像数に達したら終了
if [ $IMGS -eq $PHOTO_NUM ] ; then
 FLAG=1
 IMGS=0
fi

if [ $FLAG -eq 1 ] ; then
 #html 終了
 echo "</body>" >> page$PAGES.html
 echo "</html>" >> page$PAGES.html
 echo "page$PAGES.html makes."
 PAGES=‘expr $PAGES + 1‘
fi

#表示した画像数を更新
 IMGS=‘expr $IMGS + 1‘

done

#キレイに写真数がページごとに等分できないとき
if [ $IMGS -ge 2 ] ; then
 echo "</body>" >> page$PAGES.html
 echo "</html>" >> page$PAGES.html
 echo "page$PAGES.html makes."
fi
```

3.1.1 実行結果

```
[Kazuya-OSHIRO:~/ie_lab/sample-script/thumbnail] j04013% ./thumbnail_ex.sh
Usage: ./thumbnail_ex.sh SUFFIX RESIZE
   RESIZE: percent
   PHOTO_NUM: number of Image
   example) thumnail3.sh jpg 30 4
[Kazuya-OSHIRO:~/ie_lab/sample-script/thumbnail] j04013% ./thumbnail_ex.sh jpg 30 4
 temporal directory (resized) exsits...
```

```
convert DSC00030.jpg to resized/DSC00030.jpg_s.jpg
convert DSC00057.jpg to resized/DSC00057.jpg_s.jpg
convert DSC00076.jpg to resized/DSC00076.jpg_s.jpg
convert DSC00149.jpg to resized/DSC00149.jpg_s.jpg
convert DSC00160.jpg to resized/DSC00160.jpg_s.jpg
convert DSC00204.jpg to resized/DSC00204.jpg_s.jpg
page1.html makes.
page2.html makes.
```

3.2 考察

このシェルスクリプトは引数をもとに html に貼る画像数、画像の縮小率、変換する画像の種類などを決めてそれに基づき画像を貼付けた html をいくつか作成するものであり、ほとんどがサンプルと同じような作りになっている。

プログラムの流れとしてはまず、初めに引数のチェックを行う。今回の引数は 3 つであるので 3 つ以外の場合はエラーとして記入例を表示するようにしている。

次にそれぞれの引数を変数に代入し、縮小した画像を保存するためのディレクトリを生成する。その際、既にファイルが存在している場合には作成しないようになっている。その後、画像を縮小してディレクトリに保存する。ちなみに convert は ImageMagick をインストールしていない場合には使えなかった。

それが終わると html を作成する、まず初めに html を生成するフラグを設定する。このフラグは html の開始部分を作成する時と指定した画像を超えたときに変動する。このフラグが 1 の時に新しく html を作成する。変数 IMGS は画像の数を表すものであり、それが引数で渡した値になったら 0 に初期化する。ちなみにこの時にフラグが 1 となる。

最後にループを抜けたところにも html の終了部分を記述するためのものがある。これは画像数を 4 や 5 のした時 (今回の場合) のようにページに乗る画像数が変わる場合、ループ文内では html の終了部分を通らないため html がちゃんと作成できなくなるのを防ぐためにある。

4 level 4

4.1 access_log に記載されているアクセス総数をカウントせよ。(Level 4.1)

4.2 日毎のアクセス数、IP 毎のアクセス数をカウントせよ。(Level 4.2)

4.3 Level 4.2 で求めたカウント数を棒グラフ作成せよ。

4.1 level4.sh

```
#!/bin/sh

# 全てのアクセス数を表示
echo "総アクセス数‘less access_log | wc -l‘"
echo ’’

# 日毎のアクセス数を表示
echo 日毎のアクセス数
cut -d ’ ’ -f 4 access_log | tr -d ’[’ | cut -d ’:’ -f 1 | uniq -c
echo ’’

# IP 毎のアクセス数を表示
echo IP 毎のアクセス数
cut -d ’ ’ -f 1 access_log | sort -n | uniq -c
echo ’’

# day という変数に日毎のアクセス数を代入する
day=‘cut -d ’ ’ -f 4 access_log | tr -d ’[’ | cut -d ’:’ -f 1 | uniq -c‘
# daily.data の初期化
echo ’’ > daily.data

FLAG=1
for daily in $day
do
# フラグ 1 の時
 if [ $FLAG -eq 1 ] ; then
  W=$daily                # アクセス数を確保
  FLAG=0                  # フラグを 0 にする
 else
# 日にち、アクセス数の順に代入する
  echo "$daily $W" >> daily.data
  FLAG=1                  # フラグを 1 にする
 fi
done

PLOT=daily.gnuplot        # daily.gnuplot を読み込む
# PLOT に設定を記入していく
echo "set terminal png" > $PLOT             # 出力端末を変更
```

```
echo "set output \"daily.png\"" >> $PLOT    # daily.pngに出力する
echo "set xdata time" >> $PLOT              # x軸のデータとして年月日を扱う
echo "set timefmt \"%d/%b/%Y\"" >> $PLOT    # 今回の場合、日付/月/年となっている
echo "set boxwidth 0" >> $PLOT             # 棒グラフの棒の幅の太さ
echo "set format x \"%d/%b\"" >> $PLOT      # 日と月だけを表示
echo "plot \"daily.data\" using 1:2 notitle with boxes" >> $PLOT
gnuplot < $PLOT                            # gnuplotに設定を渡す
```

4.1.1 実行結果

```
[Kazuya-OSHIRO:~/ie_lab/sample-script/log] j04013% ./level4.sh
総アクセス数       95

日毎のアクセス数
   7 06/Dec/2004
   1 08/Dec/2004
  76 09/Dec/2004
   2 10/Dec/2004
   1 13/Dec/2004
   2 15/Dec/2004
   1 17/Dec/2004
   2 21/Dec/2004
   3 22/Dec/2004

IP毎のアクセス数
   1 35.10.47.37
   3 60.34.136.173
   1 61.116.186.45
   2 61.199.170.156
   1 61.213.47.211
   6 61.78.61.166
   1 61.95.54.174
   1 64.53.90.33
   2 66.196.90.178
```

```
1 66.196.90.207
5 66.196.90.59
1 66.196.90.92
1 66.196.91.132
1 66.196.91.175
1 66.196.91.178
1 66.196.91.199
1 66.196.91.202
1 66.196.91.203
2 66.196.91.205
1 66.196.91.206
2 66.196.91.207
2 66.196.91.216
7 66.196.91.239
1 66.196.91.32
1 66.196.91.87
1 68.121.94.147
1 68.143.54.82
4 82.67.110.170
2 82.79.189.243
1 133.13.48.235
1 133.13.48.8
1 133.13.49.18
2 133.13.50.76
1 133.13.52.37
1 133.13.53.122
2 133.13.54.156
2 133.13.54.53
3 133.13.57.226
2 133.95.109.80
1 210.139.250.215
1 210.47.27.36
1 218.113.200.138
1 218.146.238.174
1 219.160.252.142
5 219.166.179.56
3 220.109.8.237
5 220.110.222.75
3 220.20.70.41
```

```
1 220.221.238.133
2 220.221.239.196
```

4.1.2 作成したグラフ

どうも、うまく画像を Latex に付けられなかったので最後にのせることにした … 。

4.2 考察

4.1 access_log にあるアクセス総数をカウントするために wc -l を記述しているこれは行数を表示するもので一つのアクセスには一行必要なのでこれでできるようになる。

4.2 まず、cut で' ' の 3 つめと 4 つめの間の文字列を抜き出す。この場合、年月日と時間の表示が選ばれる。次にこの表示には始めに'[' がついているのでこれを省くそして初めの':' までの間の文字列を抜き出す。これで後ろについていた時間の表示を消すことができる。そのあと uniq -c を使うことで同じ所をひとまとめにしてその数を追加している。

IP のときも同じように行う。ただ、IP の場合は見やすいように sort を用いている。

4.3: これは日毎のアクセス数を gnuplot をつかって表にするためのシェルスクリプトである。まず、先ほどの日毎のアクセス数を取得してその値をループ文を用いて年月日とアクセス数の表示を逆し、それを daily.data に加えている。ちなみにそれをやることにより後の plot 設定において日にちを取得できるようになる。

次に PLOT の作成に移る。プロットにはコメントを見れば分かるように出力時のファイル名や日にちの設定を受けとったり、棒グラフにすること等が記述されている。