LING 408/508: Programming for Linguists

Lecture 8

Today's Topic

- Homework 3 graded
- Ungraded Homework 4 Review
- Homework 5
- Bash shell:
  - loops
  - positional parameters (more on)
  - globbing
  - string manipulation
    - an example, renaming files using a loop

Upgraded Homework Review 4

- Modify the shell script so it computes BMI (without using bc) and prints to two decimal places

```
1#/bin/bash¶
2if [ $# -ne 2 ]; then¶
3    echo "Usage: weight in kg, height in cm"¶
4    exit 1¶
5fi¶
6((bmi = $1 * 1000000 / ($2 * $2)))¶
7((bmi2 = $bmi / 100))¶
8echo $bmi2.${bmi#$bmi2}¶
```

```
ling508-20$ bash bmi5.sh 68 170
23.52
ling508-20$ bash bmi5.sh 68 175
22.20
ling508-20$ bash bmi5.sh 68 180
20.98
ling508-20$
```

Upgraded Homework Review 4

- echo

```
[ling508-20$ echo -e "\n\n"


[ling508-20$ echo "\n\n"
 \n\n
```

echo

`echo [-neE] [arg …]`

Output the *arg*s, separated by spaces, terminated with a newline. The return status is 0 unless a write error occurs. If `-n` is specified, the trailing newline is suppressed. If the `-e` option is given, interpretation of the following backslash-escaped characters is enabled. The `-E` option disables the interpretation of these escape characters, even on systems where they are interpreted by default. The `xpg_echo` shell option may be used to dynamically determine whether or not `echo` expands these escape characters by default. `echo` does not interpret `--` to mean the end of options.

Homework 5

- Modify the BMI calculator to:
  1. accept either command line arguments or read from the terminal if they're missing
  2. print the weight status message according to the following table:
  3. modify the calculator to accept input in both metric and traditional units
- make sure you supply examples of your program working!

BMI	Weight Status
Below 18.5	Underweight
18.5 – 24.9	Normal
25.0 – 29.9	Overweight
30.0 and Above	Obese

```
ling508-20$ bash hw5.sh
weight in kg (lbs): 68
height in cm (in): 172
units kg/lbs: k
22.98
normal
ling508-20$ bash hw5.sh 160 68 lbs
24.32
normal
ling508-20$
```

Homework 5

- Instructions:
  - email me
  - submit everything in one PDF file!
  - subject of email: 408/508 Homework 5 your name
  - cite any discussion or source
  - due date: next Monday by midnight

loops

- For-loop:
  - for VAR [in LIST]; do …; done
- Example:
  - create backup copies of files
  - (*let's create a few empty .jpg files first using* `touch`)
  - `for i in *.jpg; do echo $i ; done`
  - `for i in *.jpg; do cp $i $i.orig ; done`

```
touch f1.jpg f2.jpg f3.jpg
```
Brace expansion:
```
touch f{1,2,3}.jpg
```

```
sandiway@sandiway-VirtualBox:~$ touch f1.jpg f2.jpg f3.jpg
sandiway@sandiway-VirtualBox:~$ for i in *.jpg
> do echo $i
> done
f1.jpg
f2.jpg
f3.jpg
sandiway@sandiway-VirtualBox:~$ for i in *.jpg; do echo $i; done
```

loops

- while-loop:
  - while COND; do …; done
  - break
    (*out of while-loop*)

```
sandiway@sandiway-VirtualBox:~$ bash linebreak
Next word: this
Next word: is
Next word: nothing
Next word: like
Next word: what
Next word: i
Next word: want
33: this is nothing like what i want
```

- until COND; do …; done

```
#!/bin/bash
line=""
while true
do
     read -p "Next word: " word
     line="$line $word"
     if [ ${#line} -gt 30 ]
      then break
     fi
done
echo ${#line}:$line
exit 0
```

${#*var*} length of string stored in *var*

Positional Parameters

- command line to script (or function):
  - $0, $1, etc.
  - $# number of parameters passed
  - $* all parameters as a single word
  - $@ each parameter is a quoted string
  - shift removes one parameter (use with $#)
  - (quoted variables below just in case there are spaces in the values)

```
for arg in "$*"
do
  echo $arg
done
```

```
for arg in "$@"
do
  echo $arg
done
```

Positional Parameters

```
1 for arg in "$@"
2 do
3   echo $arg
4 done
5 for arg in $@
6 do
7   echo $arg
8 done
```

```
ling508-20$ bash cmd.sh 1 2 3 "this a"
1
2
3
this a
1
2
3
this
a
ling508-20$
```

Example

- Sum all numbers supplied on command line
- (sum.sh)

```
#!/bin/bash
sum=0
while [ $# -ne 0 ]
do
    ((sum=sum+$1))
    shift
done
echo $sum
exit 0
```

```
bash sum.sh 1 2 3 4 5
15
bash sum.sh 1 2
3
bash sum.sh 1
1
bash sum.sh
0
```

Expansion

- Arithmetic expansion:
  - $((… expression ..))
    - `x=$(($x+1))`
- Pathname expansion (aka ***globbing***):
  - similar (but not the same) as regular expressions
    - `*` (*wild card string*)
    - `?` (*wild card character*)
    - [ab] (*a or b*)
    - [^ab] (*not* (*a or b*))
- (curly) Brace expansion:
    - `mkdir ~/class/examples/{ex1,ex2}`
    - `echo x{1,2,3,4}`

Use command `ls`
in conjunction with globbing

String manipulation

- String length:
  - ${#var}
- Substring:
  - ${string:position} — starting at position (0,1,2…), (-N) from the end
  - ${string:position:length}

```
units=${3:0:1}
```

- Delete prefix:
  - ${string#substring} — shortest match
  - ${string##substring} — longest match
- Delete suffix:
  - ${string%substring} — shortest match
  - ${string%%substring} — longest match

```
cp $file ${file%.*}.bak
```

- Substring substitution:
  - ${string/substring/replacement} — replace first match
  - ${string//substring/replacement} — replace all matches
- Prefix substitution: ${string/#substring/replacement}
- Suffix substitution: ${string/%substring/replacement}

File extension renaming

Shell script:

```
#!/bin/bash
if [ $# -ne 2 ]; then
echo “usage: ext1 ext2”
exit 1
fi
for filename in *.$1
# Traverse list of files ending with 1st argument.
do
  mv "$filename" "${filename%$1}$2"
done
exit 0
```

create a subdirectory
with some .JPG files

```
sh ../rename_ext.sh JPG jpg
```

Delete suffix: ${string%substring}

File renaming

- Example:
  - append a suffix -1 to all jpg files
  - `for f in *.jpg; do mv $f ${f/./-1.}; done`
- Levels of quoting:

Substring substitution:
${string/substring/replacement}

```
$ echo text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
text /home/me/ls-output.txt a b foo 4 me
$ echo "text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER"
text ~/*.txt {a,b} foo 4 me
$ echo 'text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER'
text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
```

http://linuxcommand.org