

Shell Scripts	LABORATORY	FOUR
<b>OBJECTIVES</b>		
1. Shell Script Programming		
2. File Transfer Protocol		
3. Web Pages Publishing		

## Shell Script Programming

**Try out** the shell script programs covered / to be covered in the lecture and follow through the statements / commands to understand how they work. Try the programs using different inputs. The symbol **\*** is a *wildcard character* that leads to the matching of filenames, etc. Note that you may need to make some changes to the program before it can be executed, if you merely copy it from the pdf file. Watch out for difference in quotation symbols and for **^M** characters when you make the copy, before you try to run it. You can use the command **dos2unix** to perform removal of those **^M** characters for file *filename*:

```
dos2unix filename
```

The **dos2unix** command is not available in Mac OS X installation, but you can do the same trick using the following approach, by creating a new file called *newfilename*:

```
tr -d '\r' < filename > newfilename
```

```
#!/bin/csh -f
echo Hello $USER, time is `date`
echo You must be `whoami`
echo Your home is $HOME
echo Your calendar for this month is:
cal
echo The processes you are running are:
ps -ef | grep $USER
echo "Thanks for coming. See you soon!!!"
```

```
#!/bin/csh
# There is a bug here, could you debug it?
echo -n "What was your grade? "
set grade = $<
if ( $grade >= 85 && $grade <= 100 ) then
    echo "You got an A!"
else if ( $grade >= 70 ) then
    echo "You got a B"
else
    echo "Better study"
endif
```

```
#!/bin/csh
echo -n "Which color do you like? "
set color = $<
switch ($color)
case red:
    echo "$color is the color of Chinese New Year."
    breaksw
case yellow:
    echo "The sun is sometimes $color."
    breaksw
default:
    echo "$color is not one of those that I know."
    breaksw
endsw
```

```
#!/bin/csh
set courselist = (201 305 304)
foreach course ($courselist)
    if ($course < 300) then
        echo level 2 course $course
    else
        echo level 3 course $course
    endif
end
```

```
#!/bin/csh
echo -n "Please enter a number: "
set num = $<
set ctr = 1 sum = 0
while ($ctr <= $num)
  @ sum = $sum + $ctr
  echo $ctr
  @ ctr++
end
echo "The sum of $num numbers is $sum."
```

```
#!/bin/csh
# note that if this does not work for some of you due to change of file system in PolyU,
# try to make use of /tmp, which would still work under apollo or apollo2
# note also that /tmp is shared between everyone, so please create your subdirectory
# cd /tmp
# mkdir itismell1234567
# cd itismell1234567
# copy some files to here
foreach file (*)
  if (-d $file) echo $file is a directory
  if (-f $file && -x $file) then
    echo $file is a plain executable file
    if (!( -w $file )) echo $file could not be modified
  endif
end
```

```
#!/bin/csh
echo "This script is called $0"
if ( $#argv == 0 ) then
  echo "No argument"
else
  echo "The number of arguments : $#argv"
  echo "The first argument : $1"
  echo "We use argv to print the second argument : $argv[2]"
endif
```

```
#!/bin/csh -f
cp $1 $2
if ( $status == 0 ) then
  echo File copied successfully
else
  echo Fail to copy file
endif
```

## File Transfer Protocol

### [Optional]

The **File Transfer Protocol** or FTP allows you to upload files from PC to computer system (at an ISP or to Unix or Linux) or download files from computer system to your PC. A popular program on the PCs is the WS\_FTP program implementing FTP. However, it is not very secure and has recently been replaced by **Secure File Transfer Program** (SFTP) in the department. You can download the freeware WinSCP offering SFTP from <http://winscp.net/eng/download.php> and install it. You need to fill in [csdoor.comp.polyu.edu.hk](http://csdoor.comp.polyu.edu.hk) as *hostname* and select *port 22* when connecting from home. An equivalent software for FTP on the Mac is Cyberduck. You can invoke Cyberduck to transfer files around.



Note that if you are sending plain text files via FTP, you should use *text* format (this will automatically correct those ^M monsters on Unix/Linux created by an editor on PC so that you do not need to use the **dos2unix** command). For other files, you should use *binary* format. If you are not clear about the nature of your files, it is much safer to send them using *binary* format. Binary files include program executables, Word files, Excel files, images, etc.

## Publishing Web Pages via Unix / Linux

### [Optional]

In Department of Computing, personal web pages accessible to the public are stored in Unix/Linux file system. They must be placed under a special directory called `public_html` under your home directory. Directories must be at least *executable* and files must be at least *readable* to other users. The default file accessed by web browsers is `index.html`. Once you have created a set of web pages using FrontPage or other software on PC, you may try to publish them to the public. You have to upload those related files from PC/Mac to Unix/Linux, perhaps using `sftp` / WinSCP / Cyberduck or move them via J: drive. The web page files must be stored under `public_html`, similar to the need that the web page files must be stored under `FPWeb/yourweb` in FrontPage98. We must do some preparation work to set the appropriate protection mode. **Note:** The preparation work only needs to be done *once* in Unix/Linux. You may or may not need to set protection mode for your own directory in PolyU system, depending on which year you are in (recall there is a recent change in file system), but that is the default approach in most other Unix/Linux.

- Change the protection mode of your own directory to `drwx--x--x` (i.e., 711) by typing  
`chmod 711 ../11234567d`
- Create a sub-directory called `public_html` by typing  
`mkdir public_html`
- Change the protection mode of the directory to `drwx--x--x` (i.e., 711) by typing  
`chmod 711 public_html`
- Now, invoke `sftp` or WinSCP to *upload* to Unix/Linux or *copy* the files to J: by yourself.

The files are now placed in Unix/Linux. There is one major subtle difference between PC and Unix/Linux. In Unix/Linux, all web pages have file names ending with `.html` but in PC, all web pages have file names ending with `.htm`. Depending on the web server, you may need to rename the file(s):

```
mv index.htm index.html
```

Now, you can explore your web pages using any browser. Your own homepage can be viewed at

```
http://www.comp.polyu.edu.hk/~11234567d
```

where `11234567d` should be replaced by your own username. You may need to make some adjustments to the contents of those web page files if you discover some minor problems, such as path names. Also, you need to make sure that files for web pages are readable (`chmod 644`) and directories are executable (`chmod 711`). It is now that the `pico/jpico` editor becomes useful. However, you need to work directly with the “terrible” HTML formats. To simplify, you may use `zip` to archive all the necessary files under FrontPage or other web tools into a *zip file* and place it under Unix/Linux. Finally, you can extract the files from the `zip file` and set the appropriate protection mode for files and directories.

## Laboratory Exercises

1. *Extend the shell script* above that prints out a grade (only grade **A** and **B** in that shell script) based on the input mark from a user according to the following mark distribution. Check the input to ensure that the mark is between 0 to 100 and print an appropriate error message when necessary. This is only for practice. No submission is required.

Mark	Grade
90 - 100	A+
80 - 89	A
75 - 79	B+
65 - 74	B
60 - 64	C+
50 - 59	C
45 - 49	D+
40 - 44	D
0 - 39	F

2. [Optional submission to BlackBoard for participation mark, deadline 18 February] Write a shell script that prompts two players to play the game “Pick the Turtle” (潛烏龜) in a simplified version, where there are nine pairs of cards, with face values 1 to 9. One card is randomly taken out to be the *turtle card*. The remaining 17 cards are dealt to the two players, as stored in the argument list. You can assume that there is no error in the input cards.

The first player is **cross** × and the second player is **circle** ○. Each player will look at his/her cards and remove the pairs. After that, the player with fewer remaining cards will draw a card from the player with more remaining cards. The card drawn will be placed *at the end for simplicity*. If a match is found, the pair is discarded and the other player will draw a card. This is repeated until one of the players gets rid of all cards from his/her hand to become the winner. You should report which player is the winner to the game.

Each player will enter a number  $d$  to indicate the position of the card to draw from the opponent, where  $d \in \{1, 2, 3, \dots, 9\}$ . A player will need to retry if  $d >$  number of cards held by the opponent. In the example, prompts are in black color. User inputs are in **red** color. System and error messages are in **blue** color. For instance, the program can be run like:

```
pickturtle 4 8 7 6 2 9 4 3 5 8 1 1 2 9 7 6 3
```

Here the turtle card is 3, and player **cross** and player **circle** get the cards 4,7,2,4,5,1,2,7,3 and 8,6,9,3,8,1,9,6 respectively. After removing duplicates, player **cross** and player **circle** hold the cards 5,1,3 and 3,1 respectively. You would be printing the configuration after each step. Your program should be called `pickturtle`. You may want to process an optional argument `-d` (e.g. `pickturtle -d 4 8 7 6 2 9 4 3 5 8 1 1 2 9 7 6 3`) to turn on the *debug mode* to print the cards held by the two players, as shown in **green** color.

Since there is no random number generator in C-shell, we have to either write a C program to generate the random numbers for the cards, or use the random number generator from **bash**. *Bonus* will be given if your program can *shuffle the cards randomly* before being drawn by an opponent. Here is an example of C-shell script that returns a random number (between 0 to 32767), making use of **bash**.

```
set rnum = `bash -c 'echo $RANDOM'`
echo random number is $rnum
```

```
Player cross has 3 cards.
Player cross cards: 5 1 3 # this line is optional debugging line with option -d
Player circle has 2 cards.
Player circle cards: 3 1 # this line is optional debugging line with option -d
Round 1, player circle to draw a card:
d = 1
card = 5
Player circle has 3 cards.
Player circle cards: 3 1 5 # this line is optional debugging line with option -d
Round 1, player cross to draw a card:
d = 4
sorry, invalid card, retry
d = 3
card = 5
Player cross has 3 cards.
Round 2, player circle to draw a card:
d = 3
card = 5
Player circle has 3 cards.
Round 2, player cross to draw a card:
d = 2
card = 1
Player cross has 1 cards.
Round 3, player circle to draw a card:
d = 1
card = 3
Player circle has 1 cards.
Player cross has 0 cards.
Game over!
Congratulation to player cross, winning in round 3!
```

3. [For interested students only] Create the following information in a file called **index.html** under directory **public\_html**

```
<html>
<head>
<title>Personal Home Page of Chan Tai Man</title>
</head>

<body text="#000000" link="#0000FF" bgcolor="#FFFFFF">
<hr>
<h1 align=center>My first home page in Unix</h1>
<p>
This is a description of Chan Tai Man (put some of your own information...).
</p>

<p>
My favorite links:
<ul>
<li><a href="http://www.hku.hk/">University of Hong Kong</a>
<li><a href="http://www.cuhk.edu.hk/">Chinese University of Hong Kong</a>
<li><a href="http://www.ust.hk/">Hong Kong University of Science and Technology</a>
<li><a href="http://www.polyu.edu.hk/">The Hong Kong Polytechnic University</a>
</ul>
</p>

<hr>
<address>

<a href="mailto:11234567d@polyu.edu.hk">11234567d@polyu.edu.hk</a>
</address>
</body>
</html>
```

Change the protection mode of your file to **-rw-r--r--** with

```
chmod 644 index.html
```

View your own homepage under

```
http://www.comp.polyu.edu.hk/~11234567d
```

Upload the file **mailbox.gif** (downloadable) to your directory (**public\_html**) and change the protection mode for the image file appropriately, so as to display the mailbox image in the web page. You may want to hunt for your own mailbox image and place it inside your home page.

4. [*For advanced students only*] CGI (Common Gateway Interface) is a simple way for *server side programming* to embed dynamic information to a web page. It can be considered the *first generation* internet programming. **PHP** and other mechanisms are adopted in newer generations of internet programming. We will try out simple CGI programming here. *Create* the following CGI program “**hello.cgi**” within the directory **public\_html**

```
#!/bin/csh
# create HTTP header
echo "Content-Type: text/html"
echo
# create HTTP Content
cat << EOF
<html>
  <head>
    <title>A Hello World CGI, created by a script program</title>
  </head>
  <body>
    Hello World CGI !<br>
    Welcome, $USER !<br>
    <hr>
    Your files stored under public_html are:<br>
  EOF
# this is a command
ls -l
echo and this completes the list.
# second part of content
cat << EOF
  <br>
  <br>
  Enjoy !
</body>
</html>
EOF
```

Change the protection mode accordingly. You can then access and run this CGI program at <http://www.comp.polyu.edu.hk/~11234567d/hello.cgi>

Observe how the variable **\$USER** gets *evaluated* and the command **ls -l** gets *executed*.