Shell Programming

1.1 Shell Preliminaries

What is a Shell?

When you log on, a shell process is created by the kernel to act upon your commands.

*You can skip this part and straight away proceed to Exercise 1 on page 2. Return to this if you need a refresher.

What are types of Shell?

1. Bourne Shell (sh)
2. C Shell (csh)
3. Bourne again Shell (bash)

What does the shell process do?

♦ It displays a prompt and waits for a user command

♦ The shell executes a user command when entered and continues until an eof character

(^D) is entered

♦ The shell terminates upon reading eof character

♦ If a command is longer than a line, terminate the line with a \ followed by ENTER and continue the command in the next line

1.2 Special Characters

Characters Description

> Output redirection >> Output redirection by appending < Input redirection	* File substitution wild card; matches zero or more characters ? File substitution wild card; matches any single char […] file substitution wild card; matches any char in bracket
'command' Command substitution; replaced by the output from command | pipe symbol ; Command sequencer || Conditional execution; executes next if previous failed && conditional execution; executes next if previous succeeded	(…) Group commands & Runs a command in the background # All characters that follow up to a new line are comment $ Accesses a variable value

1.3 Command Sequences

To run **more than one** command in a line, we put the ";" character in between the commands

Example:

date;pwd;ls

What is the output? What is the difference with pipe?

date | pwd | ls

1.4 Conditional Sequencing By

using && or ||

cc myprog.c && a.out

The line of codes above means that Execute a.out **only if** cc was successful

cc myprog.cc || echo compilation failed

The line of codes above means that Execute echo only if cc returns error

---

Exercise 1:

1.5 Shell Script

Now we will try to write our very first shell script

1. Make a directory first for this lab session. So like before type (remember StudentID is your own ID):

   mkdir OSStudentID

2. To write a shell script, we can use any text editor for example, pico and vi in linux mint, you can use **Gedit text editor**. Try and find the program through the search box.
3. In the Gedit text editor, type in the following (you can copy and paste **but you have to retype** all the special characters like -, +, " because it does not translate well after being copied):

```
#My first shell script
echo –n "The date today is "
date
```

To save the file, find your OSStudentID folder and the save the script. Save it as script1.

4. After we save the shell script, we need to change that file to be executable. Make sure you are currently inside the directory that you have made earlier. **cd OSStudentID** to change directory.
5. Then, on the command line type the following…
   chmod u+x script1
   This give execute permission to the owner (**you have to do this EACH time you want to make a file to become executable-One time per file**)

6. After that, to execute the file, type the file name on the command line:
   ./script1

   You will get somewhat similar date output displayed as below

   The date today is Tue Mar 6 12:05:57 SGT 2015

1.6 Subshells

Subshells are created when:

♦ You give a command to create a shell

- $ /bin/bash

♦ You give a group of command

- $(ls;pwd;date)

♦ You give a script to execute

♦ You give a background job to execute

If the command is not executed in the background, parent shell sleeps until shell terminates

1.7 Variables

There are two kinds of variables - local and environment

1.7.1 Predefined environment variables

$HOME full pathname of your home directory

$PATH list of directories to search for commands

$MAIL full pathname of your mail box

$USER your user ID

$SHELL full pathname of your login shell

$TERM type of your terminal

Example:

$echo home = $HOME

1.7.2 User-Created Variables

♦ Any sequence of alphanumeric data, beginning only with an alphabet can be used as a variable name

♦ When a value is assigned to a variable, do not embed the `=` with a space or tab. Use " " if needed.

♦ Remove variables by assigning null value or use unset command

♦ Use readonly command to retain the previous value of a variable

♦ Use export command for making local variables of the parent within the grabs of child processes

Now let's type the codes below into the command line. Try and see what difference does each command will produce.

Example 1:

economy=sustainable

echo economy

sustainable

echo $economy

sustainable

echo "$economy"

sustainable

echo '$economy'

$economy

echo \$economy

$economy

Example 2:

vision="firm, certain,bright"

echo “$vision”

firm, certain, bright

echo $vision

firm, certain, bright

echo ‘$vision’

vision

readonly vision

vision=“Can we become better?”

Bash: vision: readonly variable

Exercise 2.

Call by Value “grab” by child process

You have to create 3 files: doom1, doom, doom2. Again, **use gedit text editor** and make sure you save this file inside **OSStudentID** that you have created in Exercise 1

1. For file **doom1**, type in the following inside the text editor:

```
chaos=small
echo “doom1 1: $chaos”
./doom
echo “doom1 2: $chaos”
```

Save the file as **doom1** then type in the following to give doom1 executable rights

chmod u+x doom1

2. Next, for a file named **doom**, type in the following inside the text editor:

```
echo “doom 1: $chaos”
chaos=big
echo “doom 2: $chaos”
```

Save the file as **doom** then type in the following to give doom executable rights

chmod u+x doom

3. Type the following on the command line:

./doom1

What is the output? **Print screen the output and save it as EX02A**

4. Next create **doom2**. Type in the following inside text editor.

```
export chaos
chaos=small
echo “doom2 1: $chaos”
./doom
echo “doom2 2: $chaos”
```

Save the file as **doom2** and then type the following to give doom2 executable rights

chmod u+x doom2

5. Type the following on the command line:

./doom2

What is the output? Is it different from doom1, Why?

Write your explanation in a text file: save it as **explainDOOM.txt**

Print screen the output and save it as EX02B

1.8 Quoting

Single quotes (`) prevent:

♦ Wildcard replacement

♦ Variable substitution

♦ Command substitution

Double quotes (“) prevent wildcard replacement

Try this:

echo 3 * 4 = 12

What is the output?

echo ‘3 * 4 = 12’

What can you see?

echo “3 * 4 = 12”

How about this?

echo ‘$HOME knows today is `date` ’

Observe the output of the above command

echo “$HOME knows today is `date`”

Observe the output

1.9 Keyword Shell Variables

Variables that have special meaning.

Variable	**Description**
HOME	Home Directory
PATH	File search path
MAIL	Mailbox
PS1	Primary prompt
term	Terminal type

1.10 Input from keyboard: read

Variables received value from keyboard with the use of **read** command

Example:

1. Using the Gedit text editor, type in the following codes. Remember if you are copy and pasting, retype the special characters such as the -,$ and " to ensure that there will be no error when you execute

```
echo –n "Enter your name:"
read N1
echo "How are you today $N1?"
```

   Save the file as test

2. Give executables permission to the file:

   chmod u+x test

3. Execute the file

   ./test

Shell Programming – Part 2

For this part we are going to look at test statements. **You can skip this part and continue doing the exercise on page 10 and come back to this part if you need a refresher.**

1.2 test Command

1.1.2 String checks

Success = 0 (true) Failure = non-zero (false)
test "$s1" -> true if string s1 is non-null test –n
"$s1" -> true if length of string s1 is non-null test –z
"$s1" -> true if length of string is null test
"$s1" = "$s2" -> true if string are equals
test "$s1" != "$s2" -> true if strings are unequal

1.2.2 Numeric checks

test $n1–eq$n2 -> true if integers are equal
test $n1 –ne $n2 -> true if integers are unequal
test $n1 –gt $n2 -> true if integer n1 > n2
test $n1 –ge $n2 -> true if integer n1 >= n2
test $n1 –lt $n2 -> true if integer n1 < n2
test $n1 –le $n2 -> true if integer n1 <= n2

1.2.3 File Status checks test –option filename

Option: f, r, s, w, x, d

1.3 if Statement

1.3.1 if then statement

if test
expression
then
commands
fi

or

if test [expression] **then**
commands
fi

1.3.2 if then else statement

if test expression **then** commands
else
commands
fi

1.3.3 if then elif statement

if test expr1 **then**
commands
elif test expr2
then commands
else
commands
fi

Exercise 2.1:

Create the shell script called test1 and run it.
Save the file inside your **OSStudentID** folder
Observe the output. (Remember if you are copy and pasting, retype the special characters such as the -,$ and " to ensure that there will be no error when you execute)

As before, use the Gedit text editor to write the file and then save it as test1.
Remember to change the file permission $chmod u+x test1. (Remember this also for the following exercise)

```
echo -n "enter StudentID1: "
read StID1
echo -n "enter StudentID2: "
read StID2
if test $StID1 = $StID2
then
        echo "StudentID1 and StudentID2 are equal"
else
        echo "StudentID1 and StudentID2 are different"
fi
```

Exercise 2.2:

Create the shell script called test2 and run it. Observe the output.
Same as before, use Gedit text editor to write and save the file.
The code will search for file with the same name as you entered. So make sure when you test it, you test it with the name of the file that already exists inside your working directory

```
echo –e "enter filename: \c"
read FILE
if test –f $FILE
        then
        echo "$FILE exist and is a regular file"
elif test –d "$FILE"
        then
        echo "$FILE exists and is a directory"
else
        echo "$FILE does not exist OR is not a directory"
fi
```

1.4 case Statement

```
case string in pattern1)
commands1
        ; ;
        pattern2) commands2
        ; ;
        patern3)
commands3
        ; ;
        .
        .
        .
esac
```

Exercise 2.3:

Create the shell script called test3 and run it. Observe the output.

```
echo –e “Enter a character : \c”
read CH
case $CH in
        a) echo “You entered an ‘a’”
        ;;
        b) echo “You entered a ‘b’”
        ;;
        *) echo “You entered neither an ‘a’ nor ‘b’”
        ;;
esac
```

Exercise 2.4:

Create the shell script called test4 and run it. Observe the output.

```
echo –e “Enter a character: \c”
read CH
case $CH in
        a|e|i|o|u) echo “Vowel”
        ;;
        0|1|2|3|4|5) echo “0-5 entered”
        ;;
        *) echo “enter vowel or number 0-5”
        ;;
esac
```

1.5 The while LOOP

- Execute a set of statements while a condition remains *true*.
- Exit when the condition becomes *false* while command1 do command(s)
  done

Exercise 2.5:

Create the shell script called test5 and run it. Observe the output.

```
read VAR
while test –n “$VAR”
do
    echo $VAR >> file
    read VAR
done
    cat file
```

1.6 The until LOOP

- Execute a set of statements until a condition remains *false*.
- Exit when the condition becomes *true* until command1 do command(s)
  done

Exercise 2.6:

Create the shell script called test6 and run it. Observe the output.

```
read VAR
until test –z “$VAR”
do
   echo $VAR >> file
   read VAR
done
   cat file
```

Exercise 2.7:

Create the shell script called test7 and run it.
Observe the output. After observing the output, type ls to see the content of your directory **OSStudentID**. Screen capture the content and save it as **EX03.png**

```
for i in 1 2 3 4 5
do
echo -n “$i “
done
```

Checklist for submission

Exercise 1 (Script1) – 1 File

Exercise 2 (Doom, Doom1, Doom2 scripts)

Screen Capture of EX02A abdEX02B.png

Explaination: ExplainDOOM.txt

3 script files, 2 image files, 1 text file

Exercise 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7 (Scripts test1 until 7), Screen Capture EX03.png

7 script files, 1 image file

Value Added Exercise (OPTIONAL):

- Make a script that will ask for a name and Age
- If the age is higher than 13, it will prompt a message
  - "Hi name"
  - "You can watch PG13 movies"
- If the age is lower than 13, it will prompt a message
  - "Sorry name"
  - "Please bring your parents with you"

As always, submit to e-mail muhd.nabil@gmail.com with the subjet CSNB224_Lab7_studentID_SectID