CL205 - Operating Systems Lab

Lab#13 – Shell Scripting

Introduction

Shell programming is a group of commands grouped together under single filename. After logging onto the system a prompt for input appears which is generated by a Command String Interpreter program called the shell. The shell interprets the input, takes appropriate action, and finally prompts for more input. The shell can be used either interactively - enter commands at the command prompt, or as an interpreter to execute a shell script. Shell scripts are dynamically interpreted, NOT compiled.

Shell Keywords

echo, read, if fi, else, case, esac, for , while , do , done, until , set, unset, readonly, shift, export, break, continue, exit, return, trap , wait, eval ,exec, ulimit , umask.

General things in Shell

The shbang line	The "shbang" line is the very first line of the script and lets the kernel know what shell will be interpreting the lines in the script. The shbang line consists of a #! followed by the full pathname to the shell, and can be followed by options to control the behavior of the shell. **EXAMPLE** #!/bin/sh
Comments	Comments are descriptive material preceded by a # sign. They are in effect until the end of a line and can be started anywhere on the line. **EXAMPLE** # this text is not # interpreted by the shell
Wildcards	There are some characters that are evaluated by the shell in a special way. They are called shell metacharacters or "wildcards." These characters are neither numbers nor letters. For example, the *, ?, and [ ] are used for filename expansion. The <, >, 2>, >>, and | symbols are used for standard I/O redirection and pipes. To prevent these characters from being interpreted by the shell they must be quoted. **EXAMPLE** Filename expansion: rm *; ls ??; cat file[1-3]; Quotes protect metacharacter: echo "How are you?"

Example 1:

Write a shell program to perform arithmetic operations

Steps
1: get the input
2: perform the arithmetic calculation.
3: print the result.
4: stop the execution.

Code

```
#!/bin/bash
echo "enter the a vale"
read a
echo "enter b value"
read b
c=`expr $a + $b`
echo "sum:"$c
c=`expr $a - $b`
echo "sub:"$c
c=`expr $a \* $b`
echo "mul:"$c
c=`expr $a / $b`
echo "div:"$c
```

FOR Loop

Example 2:

Write a sell program to check whether a number is even or odd.

Code

```
#!/bin/bash
num="1 2 3 4 5 6 7 8"
for n in $num
do
q=`expr $n % 2`
if [ $q -eq 0 ]
then
echo "even no"
continue
fi
echo "odd no"
done
```

Example 3:

Table of a given number.

Code

```
#!/bin/bash
echo " which table you want"
read n
for i in 1 2 3 4 5 6 7 8 9 10
do
echo $i "*" $n "=" `expr $i \* $n`
done
```

Example 4:

```
#!/bin/bash
echo " what do you want"
read n
for i in $(ls)
do
gedit i
done
```

What does the above program do?

WHILE Loop

Example 5:

```
#!/bin/bash
a=1
while [ $a -lt 11 ]
// -ge -gt -lt -le -eq -ne
//[ $a -ne 11 -a $a -ne 12 ]
//[ $a -ne 11 -o $a -ne 12 ]
do
echo "$a"
a=`expr $a + 1`
done
```

Interpret the output of the above program.

IF Statement

Example 6:

```
#!/bin/bash
for var1 in 1 2 3
do
for var2 in 0 5
do
if [ $var1 -eq 2 -a $var2 -eq 0 ]
then
continue
else
echo "$var1 $var2"
fi
done
```

Interpret the output of the above program.

ELSE-IF Statement

Example 7:

```
#!/bin/bash
for var1 in 1 2 3
do
for var2 in 0 5
do
if [ $var1 -eq 2 -a $var2 -eq 0 ]
then
continue
else if [ $var1 -eq 4 -a $var2 -eq
1 ]
then
echo "$var1"
else
echo "$var1 $var2"
fi
fi
done
done
```

Functions

Example 8:

```
#!/bin/bash
add()
{
c=`expr $1 + $2`
echo "addition = $c"
}
add 5 10
```

Switch Statement

Example 9:

```
#!/bin/bash
ch='y'
while [ $ch = 'y ' ]
do
echo "enter your choice"
echo "1 no of user loged on"
echo "2 print calender"
echo "3 print date"
read d
case $d in
1) who;;
2) cal 20;;
3) date;;
*) break;;
esac
echo "do you wish to continue (y/n)"
read ch
done
```

Exercise

Write a shell script to create a file with extension .c. Copy contents(c code) from another file to this file. Now use if statement to ask user if user enter 1 just compile it. If user enters 2 compile it and run it. If user enters 3 just print the contents of the original file. Otherwise print the contents of the current directory. Perform the same task using switch inside a function.