



K. J. Somaiya College of Engineering, Mumbai-77

Batch:A4 Roll No.:1821001

Experiment No. 02

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE: Shell Programming and system calls

AIM: To study the shell script and write the program using shell.

Expected Outcome of Experiment:

CO 1. Explain the fundamental concepts of operating system with extension to Unix and Mobile OS

Books/ Journals/ Websites referred:

1. Silberschatz A., Galvin P., Gagne G. “Operating Systems Principles”, Willey Eight edition.
2. William Stallings “Operating Systems” Person, Seventh Edition Edition.
3. Sumitabha Das “ UNIX Concepts & Applications”, McGraw Hill Second Edition.

Pre Lab/ Prior Concepts:

The shell provides you with an interface to the UNIX system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell Scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by a pound sign, #, describing the steps.

Steps to create a Shell Script:

create a file using any text editor say vi, gedit, nano etc



K. J. Somaiya College of Engineering, Mumbai-77

1.\$ vi filename

2.Insert the script/ commands in file and save the file to execute the file we need to give execute permission to the file

3.\$ chmod 775 filename

4.Now execute the above file using any of following methods:

\$ sh filename

OR

\$./filename

NOTE: Before adding anything to your script, you need to alert the system that a shell script is being started. This is done using the shebang construct. For example –
#!/bin/sh.

Description of the application to be implemented:

1. Write a shell Script that accepts two file names as command line arguments and compare two file contents and check whether contents are same or not. If they are same, then delete second file.

```
echo "Enter name of file 1"  
read f1  
echo "Enter name of file 2"  
read f2
```

```
cmp $f1 $f2 > error
```

```
total=`wc -c error | cut -f 7 -d " "`  
echo $t
```

```
if [ $t -eq 0 ]  
then  
    echo "Both file's contents are same"  
else  
    echo "Both file's contents are not same"  
fi
```

2. Write a shell script that accepts integer and find the factorial of number.

```
echo "Enter the number whose factorial you want to find"  
read input
```

```
fct=1
```

```
while [ $input -gt 1 ]
```



K. J. Somaiya College of Engineering, Mumbai-77

```
do
    fct=$((fct * input))
    input=$((input - 1))
done

echo $fct
```

3. Write a shell script for adding users.

```
if [ $(id -u) -eq 0 ]; then
    read -p "Enter username : " uname
    read -s -p "Enter password : " pwd
    egrep "^$uname" /etc/passwd >/dev/null
    if [ $? -eq 0 ]; then
        echo "$uname exists!"
        exit 1
    else
        pass=$(perl -e 'print crypt($ARGV[0], "password")' $pwd)
        useradd -m -p $pass $uname
        [ $? -eq 0 ] && echo "User has been added to system!" || echo
        "Failed to add a user!"
    fi
else
    echo "Only root may add a user to the system"
    exit 2
fi
```

4. Write a shell script for counting no of logged in users.
5. Write a shell script for counting no of processes running on system

Program for System Call:

1. Write a Program for creating process using System call (E.g fork())
Create a child process. Display the details about that process using
getpid and getppid functions. In a child process, Open the file using file
system calls and read the contents and display.

Implementation details: (printout of code / screen shot)

1. Number of logged in users using shell script.



K. J. Somaiya College of Engineering, Mumbai-77

Program:-

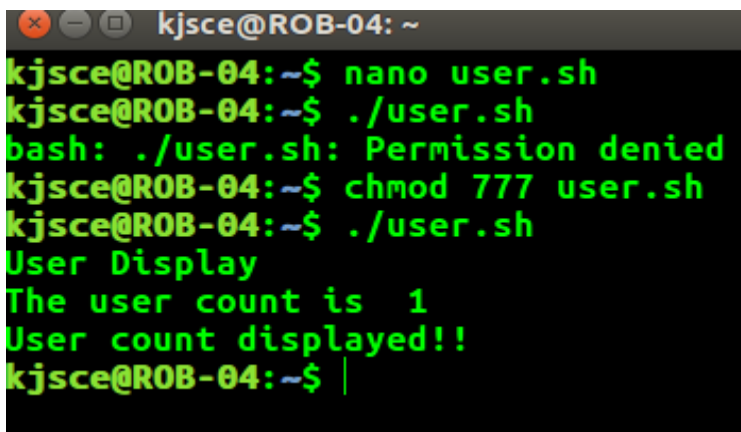
```
echo "User Display"

who > abc.txt
length=`wc -l < abc.txt`

echo "The user count is " $length

echo "User count displayed!!"
```

Output:-



```
kjsce@ROB-04: ~
kjsce@ROB-04:~$ nano user.sh
kjsce@ROB-04:~$ ./user.sh
bash: ./user.sh: Permission denied
kjsce@ROB-04:~$ chmod 777 user.sh
kjsce@ROB-04:~$ ./user.sh
User Display
The user count is 1
User count displayed!!
kjsce@ROB-04:~$ |
```

2. Counting number of foreground process.

Program:-

```
echo "Program starting to count foreground processes....."

num=`ps -e -o stat | grep "+" | wc -l`

echo "The count of foreground processes are :- " $num

echo "Program ends!!"
```



Output:-

```
kjsce@ROB-04:~$ nano foreground.sh
kjsce@ROB-04:~$ ./foreground.sh
bash: ./foreground.sh: Permission denied
kjsce@ROB-04:~$ chmod 777 foreground.sh
kjsce@ROB-04:~$ ./foreground.sh
Program starting to list foreground processes.....
The list of foreground processes are :- 7
Program ends!!
kjsce@ROB-04:~$ nano foreground.sh
kjsce@ROB-04:~$ nano background.sh
kjsce@ROB-04:~$ chmod 777 background.sh
kjsce@ROB-04:~$ ./background.sh
\Program starting to count background processes.....
The count of background processes are :- 216
Program ends!!
kjsce@ROB-04:~$ nano foreground.sh
kjsce@ROB-04:~$ |
```

3. Counting number of background process.

Program:-

echo "Program starting to count background processes....."

num=`ps -e -o stat | grep -v "+" | wc -l`

echo "The count of background processes are :- " \$num

echo "Program ends!!"

Output:-

```
kjsce@ROB-04:~$ nano foreground.sh
kjsce@ROB-04:~$ ./foreground.sh
bash: ./foreground.sh: Permission denied
kjsce@ROB-04:~$ chmod 777 foreground.sh
kjsce@ROB-04:~$ ./foreground.sh
Program starting to list foreground processes.....
The list of foreground processes are :- 7
Program ends!!
kjsce@ROB-04:~$ nano foreground.sh
kjsce@ROB-04:~$ nano background.sh
kjsce@ROB-04:~$ chmod 777 background.sh
kjsce@ROB-04:~$ ./background.sh
\Program starting to count background processes.....
The count of background processes are :- 216
Program ends!!
kjsce@ROB-04:~$ nano foreground.sh
kjsce@ROB-04:~$ |
```



4. Creating user account using shell script.

Program:-

```
echo "Creation of New User Account : "  
echo "Enter no of user acc req : "  
read numofu  
echo "User acc Req are : $nou"  
for i in $numofu  
do  
    echo "Enter New Username"  
    read username  
    sudo useradd $username  
    sudo passwd $username  
done  
echo "Task Done"
```

Output:-

```
ubuntu@ubuntu:~/Desktop$ ./useracc.sh  
Creation of New User Account :  
Enter no of user acc req :  
aaa  
User acc Req are :  
Enter New Username  
aaa
```

```
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
Task Done
```

4. Using system calls create process using extc and fork

Program:-



K. J. Somaiya College of Engineering, Mumbai-77

1.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

void main()
{
    fork();
    printf("\nHello World");
    printf("PPID : %u\n",getppid());
    printf("PID : %u\n",getpid());
}
```
2.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

void main()
{
    fork();
    printf("\nHello World");
    printf("PPID : %u\n",getppid());
    printf("PID : %u\n",getpid());
}
```

Output:-

```
kjsce@ubuntu: ~/Downloads
kjsce@ubuntu:~$ cd Down*
kjsce@ubuntu:~/Downloads$ gcc -o child child.c
kjsce@ubuntu:~/Downloads$ gcc -o parent parent.c
kjsce@ubuntu:~/Downloads$ gcc -o child child.c
kjsce@ubuntu:~/Downloads$ sudo ./child
[sudo] password for kjsce:
IN CHILD PRGM : PID : 2674
kjsce@ubuntu:~/Downloads$
```



Conclusion : Hence, we have implemented and understood the shell script concepts and we have written a script based on those concepts.

Post Lab Descriptive Questions (Add questions from examination point view)

1. Explain in brief: Meta Characters, Positional Parameters and command substitution with example.

A metacharacter (sometimes spelled meta character or meta-character) is a special character in a program or data field that provides information about other characters. A metacharacter can express an idea about how to process the characters that follow the metacharacter, as the backslash character sometimes is used to indicate that the characters following it are to be treated in a special way. A common metacharacter usage is the wildcard character , which can represent either any one character or any string of characters. In UNIX shell s, metacharacters include, but are not limited to these:

* ; |] [?

Each of these characters has a special meaning on the command line, and their use must be avoided for purposes other than their special meaning.

A positional parameter is an argument specified on the command line, used to launch the current process in a shell. Positional parameter values are stored in a special set of variables maintained by the shell.

Command substitution allows the output of a command to replace the command itself. Command substitution occurs when a command is enclosed as follows:

\$(command)



K. J. Somaiya College of Engineering, Mumbai-77

or

``command``

Bash performs the expansion by executing command in a subshell environment and replacing the command substitution with the standard output of the command, with any trailing newlines deleted. Embedded newlines are not deleted, but they may be removed during word splitting. The command substitution `$(cat file)` can be replaced by the equivalent but faster `$(< file)`.

When the old-style backquote form of substitution is used, backslash retains its literal meaning except when followed by `'$'`, ````, or `'\'`. The first backquote not preceded by a backslash terminates the command substitution. When using the `$(command)` form, all characters between the parentheses make up the command; none are treated specially.

Command substitutions may be nested. To nest when using the backquoted form, escape the inner backquotes with backslashes.

If the substitution appears within double quotes, word splitting and filename expansion are not performed on the results.

Date: 2/10/2019

Signature of faculty in-charge