

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A2

Roll No.: 1911027

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. To introduce basic concepts and functions of operating systems.

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.

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

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

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.
2. Write a shell script that accepts integer and find the factorial of number.
3. Write a shell script for adding users.
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

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

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)

Shell Programming:

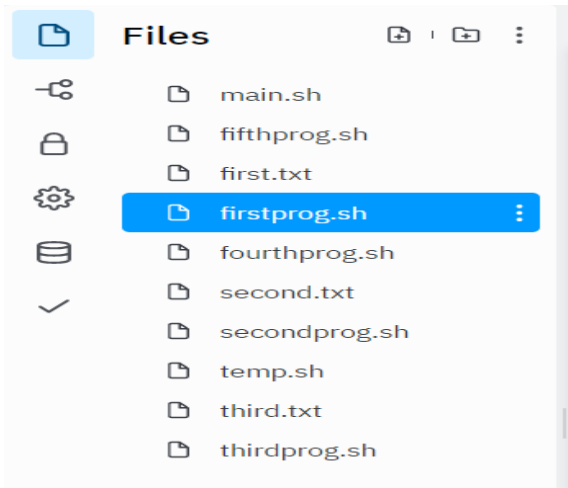
- 1) Write a shell Script that accepts two file names as command line arguments and compare two file contents and check whether contents are same or not. If they are same, then delete second file.

Code:

```
echo "-----"
echo "File 1 content : "
cat "$1"
echo "-----"
echo "File 2 content : "
cat "$2"
echo "-----"
if cmp -s $1 $2
then
    echo "Both the files have equal contetn."
    echo "Deleting file 2 ....."
    rm -f $2
    echo "$2 deleted....."
else
    echo "Both the files have different content"
fi
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Project Structure:



First.txt:

```
first.txt x
1 This is the first file
2
```

Second.txt:

```
second.txt x
1 |This is the second file.
2
```

Third.txt:

```
third.txt x
1 |This is the first file
2
```

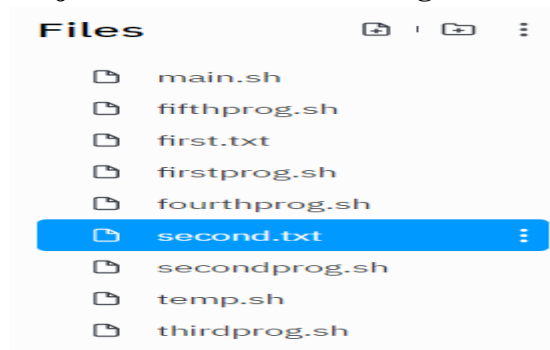


K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Output:

```
Console Shell
~/OvalYawningWireframe$ bash firstprog.sh first.txt second.txt
-----
File 1 content :
This is the first file
-----
File 2 content :
This is the second file.
-----
Both the files have different content
~/OvalYawningWireframe$ bash firstprog.sh first.txt third.txt
-----
File 1 content :
This is the first file
-----
File 2 content :
This is the first file
-----
Both the files have equal content.
Deleting file 2 .....
third.txt deleted.....
~/OvalYawningWireframe$
```

Project structure after running the shell script:



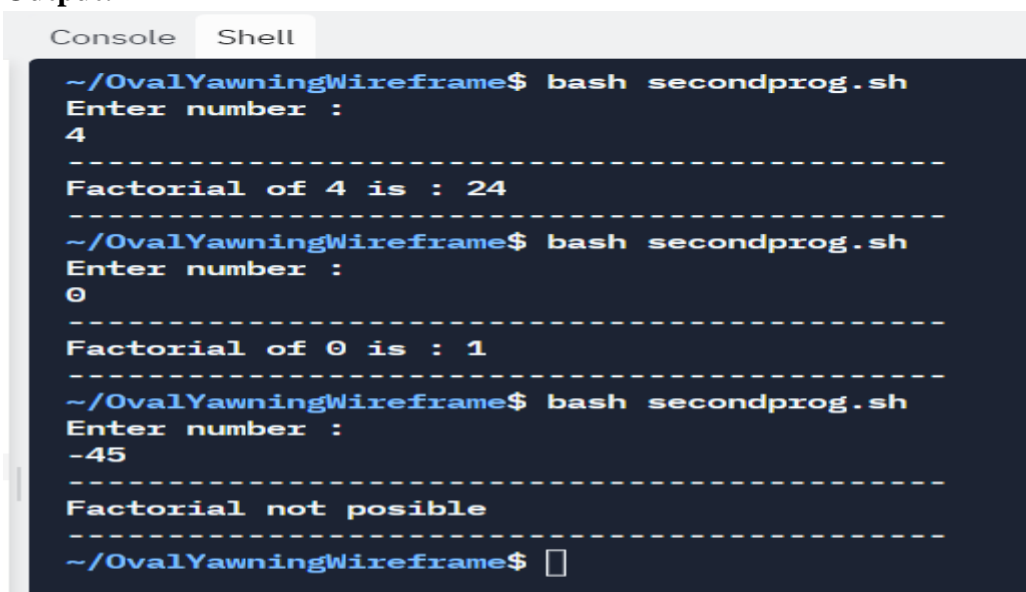
K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

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

Code:

```
echo "Enter number : "  
read N  
F=1  
echo "-----"  
if test $N -gt 0  
then  
    for((i=2;i<=N;i++))  
    do  
        F=$((F * $i))  
    done  
    echo "Factorial of $N is : $F"  
elif test $N -lt 0  
then  
    echo "Factorial not posible"  
else  
    echo "Factorial of $N is : 1"  
fi  
echo "-----"
```

Output:



```
~/OvalYawningWireframe$ bash secondprog.sh  
Enter number :  
4  
-----  
Factorial of 4 is : 24  
-----  
~/OvalYawningWireframe$ bash secondprog.sh  
Enter number :  
0  
-----  
Factorial of 0 is : 1  
-----  
~/OvalYawningWireframe$ bash secondprog.sh  
Enter number :  
-45  
-----  
Factorial not posible  
-----  
~/OvalYawningWireframe$
```

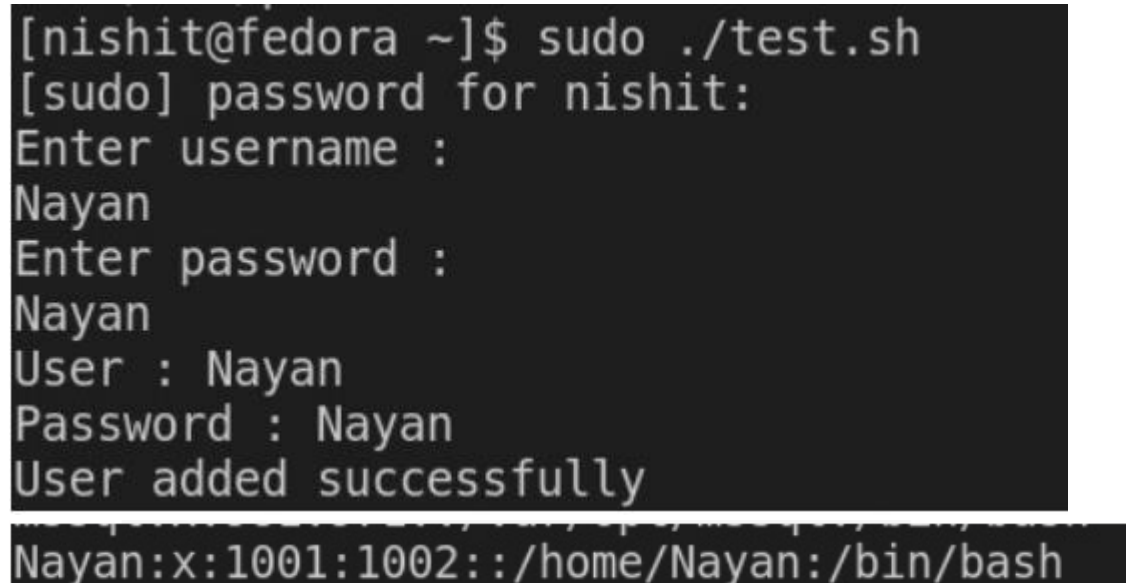
K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

3) Write a shell script for adding users.

Code:

```
echo "Enter username : "  
read username  
echo "Enter password : "  
read password  
adduser "$username"  
echo "User : $username"  
echo "Password : $password"  
echo "User added successfully"  
cat /etc/passwd
```

Output:



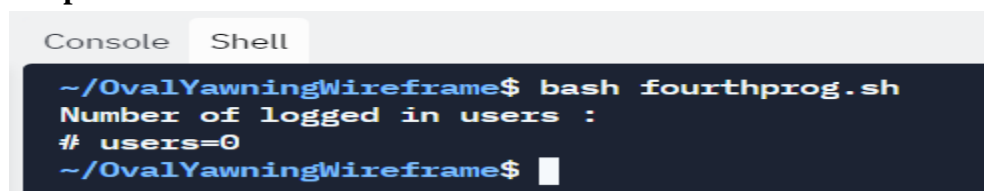
```
[nishit@fedora ~]$ sudo ./test.sh  
[sudo] password for nishit:  
Enter username :  
Nayan  
Enter password :  
Nayan  
User : Nayan  
Password : Nayan  
User added successfully  
Nayan:x:1001:1002::/home/Nayan:/bin/bash
```

4) Write a shell script for counting no of logged in users.

Code:

```
echo "Number of logged in users : $(who -q)"
```

Output:



```
Console Shell  
~/OvalYawningWireframe$ bash fourthprog.sh  
Number of logged in users :  
# users=0  
~/OvalYawningWireframe$
```


K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

5) Write a shell script for counting no of processes running on system.

Code:

```
echo "Number of processes running on system : "  
ps -aug | wc -l
```

Output:



```
~/OvalYawningWireframe$ bash fifthprog.sh  
Number of processes running on system :  
6  
~/OvalYawningWireframe$
```

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.

Code:

```
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <unistd.h>  
int main(void)  
{  
    printf("-----\n");  
    printf("Process ID before fork invocation : %d\n",getpid());  
    printf("-----\n");  
    int k=fork();  
    if(k==0)  
    {  
        printf("Child process created successfully\n");  
        printf("-----\n");  
    }  
}
```




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

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

```
printf("Child process ID is : %d\n",getpid());
printf("-----\n");
printf("Parent process ID is : %d\n",getppid());
printf("-----\n");
FILE *file;
file=fopen("test.txt","r");
if(file==NULL)
{
    printf("There is some internal error while opening the file!!!");
    printf("-----\n");
    exit(0);
}
else
{
    char c=fgetc(file);
    while(c!=EOF)
    {
        printf("%c",c);
        c=fgetc(file);
    }
    fclose(file);
    printf("\n-----\n");
}
else if(k<0)
{
    printf("Child process is not created\n");
    printf("-----\n");
}
else
{
    printf("Inside parent process\n");
    printf("-----\n");
    printf("Process ID is : %d\n",getpid());
    printf("-----\n");
    printf("Parent process ID : %d\n",getppid());
    printf("-----\n");
}
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

```
printf("Child process ID : %d\n",k);  
printf("-----\n");  
}  
return 0;  
}
```

Output:

```
Console Shell  
~/FinancialSorrowfulBlogclient$ gcc main.c -o fork  
~/FinancialSorrowfulBlogclient$ ./fork  
-----  
Process ID before fork invocation : 202  
-----  
Inside parent process  
-----  
Process ID is : 202  
-----  
Parent process ID : 39  
-----  
Child process ID : 203  
-----  
Child process created successfully  
-----  
Child process ID is : 203  
-----  
Parent process ID is : 202  
-----  
This is a text file created for EXP2 of operating system and system software subject.  
-----  
~/FinancialSorrowfulBlogclient$
```

2) Write a program which executes a loop from 1 to 10 and for every even number call fork and print something and for every odd number print something.

Code:

```
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>
```



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

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

```
#include <unistd.h>
int main()
{
    int i,count=1;
    for(i=1;i<=10;i++)
    {
        if(i%2==0)
        {
            int k=fork();
            if(k==0)
            {
                printf("Number : %d PID: %d",i,getpid());
            }
            else if(k<0)
            {
                printf("Child process not created");
            }
        }
        else
        {
            printf("%d is a odd number\n",i);
        }
    }
    return 0;
}
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Output:

```
Console Shell
~/CrimsonPunctualRoutine$ gcc main.c -o fork
~/CrimsonPunctualRoutine$ ./fork
1 is a odd number
3 is a odd number
5 is a odd number
7 is a odd number
9 is a odd number
Number : 2 PID: 463 is a odd number
5 is a odd number
~/CrimsonPunctualRoutine$ 7 is a odd number
9 is a odd number
Number : 10 PID: 54Number : 4 PID: 475 is a odd number
7 is a odd number
9 is a odd number
Number : 6 PID: 487 is a odd number
9 is a odd number
Number : 8 PID: 499 is a odd number
Number : 10 PID: 50Number : 4 PID: 515 is a odd number
7 is a odd number
9 is a odd number
Number : 10 PID: 63Number : 6 PID: 527 is a odd number
9 is a odd number
Number : 8 PID: 539 is a odd number
Number : 10 PID: 66Number : 6 PID: 557 is a odd number
9 is a odd number
Number : 10 PID: 68Number : 8 PID: 569 is a odd number
Number : 10 PID: 57Number : 8 PID: 589 is a odd number
Number : 10 PID: 70Number : 10 PID: 59Number : 10 PID: 60Number : 6
PID: 617 is a odd number
9 is a odd number
Number : 8 PID: 719 is a odd number
Number : 8 PID: 629 is a odd number
Number : 8 PID: 649 is a odd number
Number : 10 PID: 73Number : 10 PID: 65Number : 8 PID: 679 is a odd nu
mber
Number : 10 PID: 69Number : 10 PID: 72Number : 10 PID: 74Number : 1
PID: 75Number : 10 PID: 76
```

Conclusion : By performing this experiment learned how to write shell scripts and also learned how to execute shell scripts. Also understood how different unix commands are implemented in shell scripts. Implemented fork system call as well. Using fork system call understood how processes are created when we call it. System calls are well understood by performing some additional programs as well.

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Post Lab Descriptive Questions

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

ANS) Metacharacters: Metacharacters are special characters that are used to represent something other than themselves. As a rule of thumb, characters that are neither letters nor numbers may be metacharacters. Like grep, sed and awk the shell has its own set of metacharacters, often called shell wildcards. Shell metacharacters can be used to group commands together, to abbreviate filenames and pathnames, to redirect and pipe input/output, to place commands in the background, and so forth.

a. The Full Stop as a Metacharacter (.): The Full Stop (.) indicates the current position when running commands such as cd, find, or sh. In applications such as awk, grep, and sed, it's a wildcard that denotes a specific number of any character. As an example, the following command finds all txt files in the current folder and its subfolders.

```
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls
File.txt Temp1/ test.sh timepass.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ find . -type f -name '*.txt'
./File.txt

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ |
```

b. The Asterisk as a Metacharacter (*): The asterisk (*) is a universally known metacharacter. It means zero or more of any character when searching for a pattern. For example: The *.sh portion of the command returns a match for any filename that ends in .sh.

```
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls
File.txt Temp1/ test.sh timepass.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls *.sh
test.sh timepass.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ |
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

c. The Carat as a Metacharacter (^): The carat (^) is used to denote the start of a line or a string. If you want to list the files in a folder that begin with a certain string, for example, t, the carat can be used to specify that string. For example:

```
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls
File.txt  Temp1/  test.sh  timepasss.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls -a | grep ^t
test.sh
timepasss.sh
```

d. The Dollar Symbol as a Metacharacter (\$): The dollar symbol (\$) has multiple meanings as a metacharacter in Linux. When used to match patterns, it means the opposite of carat and denotes any pattern that ends with a particular string. For example:

```
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls
File.txt  Temp1/  test.sh  timepasss.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls | grep sh$
test.sh
timepasss.sh
```

Positional Parameters: 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.

```
echo "First file name is $1 and content is : "
cat "$1"
echo "Second file name is $2 and content is : "
cat "$2"
~
```

Here in the program \$1 and \$2 will be replaced by the positional parameters passed while running the shell script.

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

```
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ ls
File.txt File2.txt Temp1/ prog.sh test.sh timepasss.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ vi prog.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ chmod u+x prog.sh

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ bash prog.sh File.txt File2.txt
First file name is File.txt and content is :
This is first file.Second file name is File2.txt and content is :
This is second file.
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$
```

Command substitution: Command substitution is an operation with dedicated syntax to both execute a command and to have this command's output hold (stored) by a variable for later use. Command substitution allows the output of a command to replace the command itself. Bash performs the expansion by executing command and replacing the command substitution with the standard output of the command, with any trailing newlines deleted. A command substitution, i.e. the whole `$(...)` expression, is replaced by its output, which is the primary use of command substitutions. The command that the command substitution executes, is executed in a subshell, which means it has its own environment that will not affect the parent shell's environment. 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.

```
Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ echo "Current path is [ "$(pwd)" ]"
Current path is [/c/Users/Nayan Mandliya/Documents/OSSS LAB UNIX COMMANDS]

Nayan Mandliya@LAPTOP-BLBOBL3S MINGW64 ~/Documents/OSSS LAB UNIX COMMANDS
$ echo "Hostname is : $(hostname)"
Hostname is : LAPTOP-BLBOBL3S
```

Date: 12 / 9 / 2021

Signature of faculty in-charge

Department of Computer Engineering