**Batch: D - 1 Roll No.: 16010122096**

**Experiment No. 02**

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

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Shell Programming |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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.**

1. **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

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.

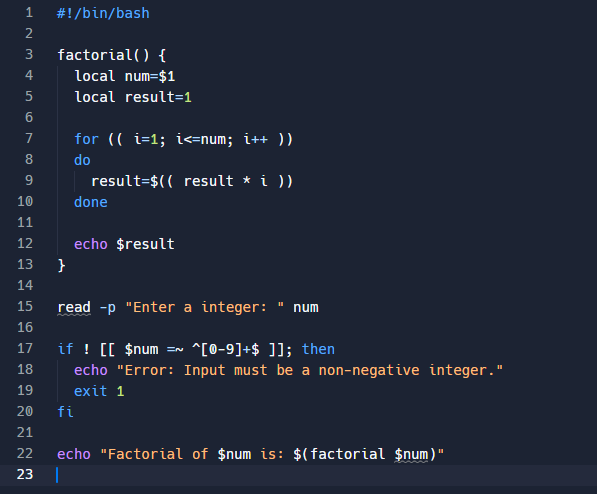
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Description of the application to be implemented**:

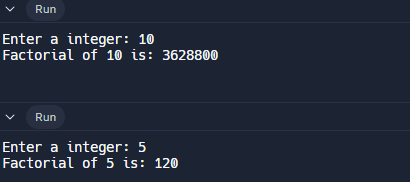
1. Write a shell Script that accepts two file names as command line arguments and compare two file contents and check whether contents are same or not. If they are same, then delete second file.
2. Write a shell script that accepts integer and find the factorial of number.
3. Write a shell script for 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

**Implementation details:** (printout of code / screen shot)

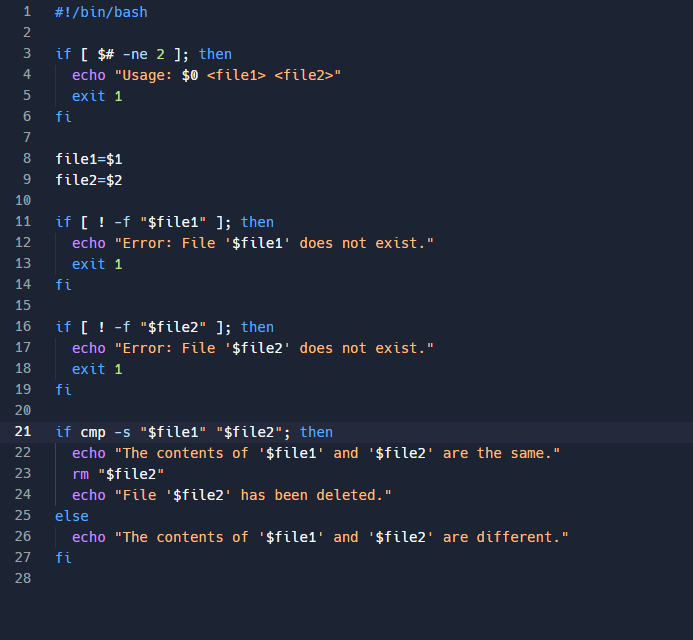
**1] Factorial**

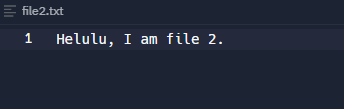
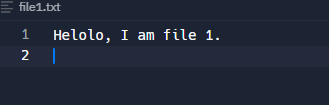


**Output:**



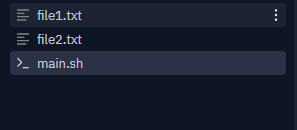
**2] File Comparison**

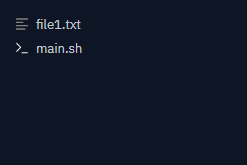
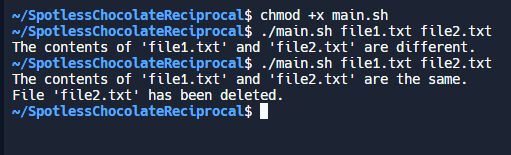
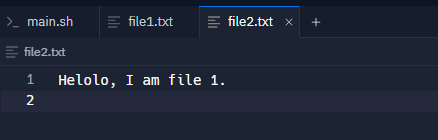
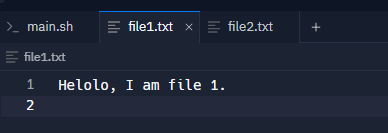




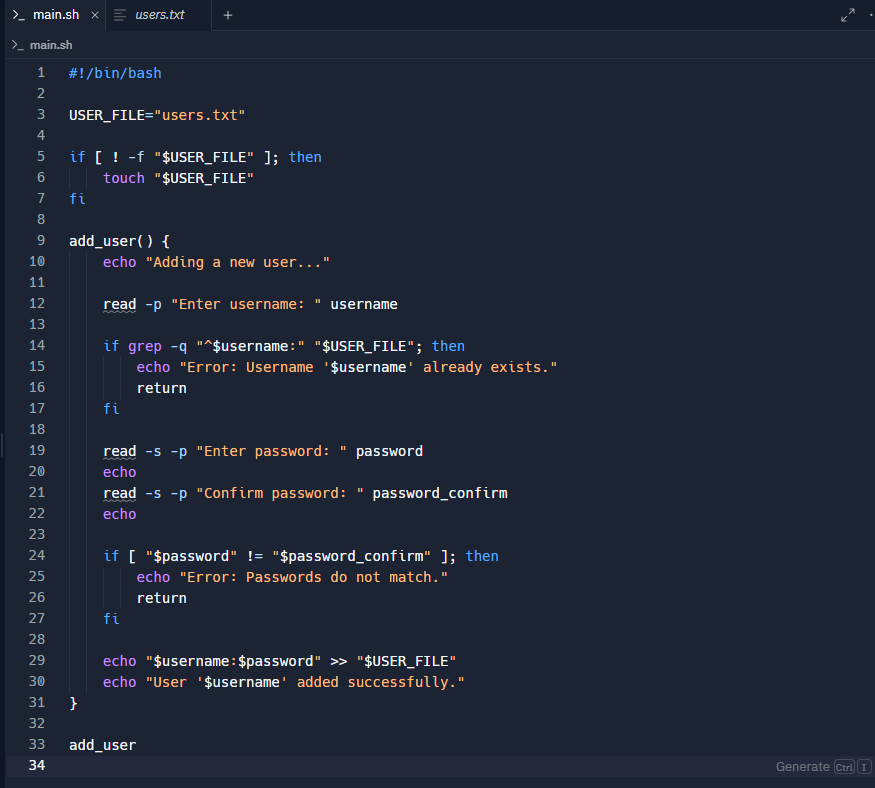
**Output:**

https://lh7-rt.googleusercontent.com/docsz/AD_4nXf0bTviaHjIwNNyUMmWclpnxzid86U6O40RS8SKGBWw9XRjHvEQ9RSIqCZMS8FIH4-Qx1Kn7QkbOts1kb5fE7udtRztYu1H89isWFvarLac9y1LwP0KTFC9bQy93x3ecduNQOattgz_-dF-xA_quh4XSS1O?key=LHTamgQxbKFMKlk4M7LKAA

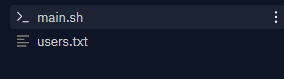


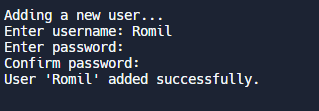


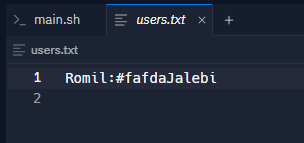
**3] Adding User**

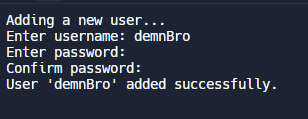


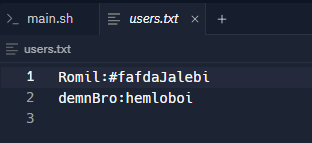
**Output:**



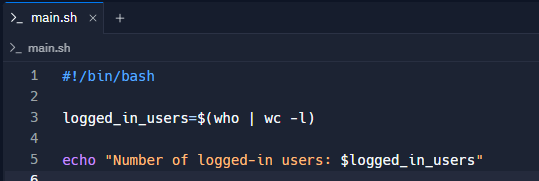




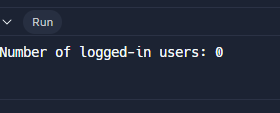




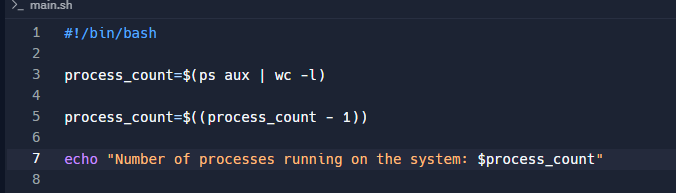
**4] Logged in users**



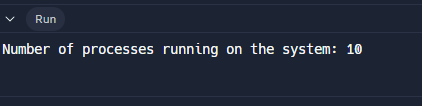
**Output:**



**5] No. of processes**



**Output:**



**Conclusion :**

Through this experiment, we explored fundamental shell scripting techniques, enhancing our understanding of operating systems and automation. We effectively implemented scripts for file comparison, factorial calculation, user management, and system monitoring.

**Post Lab Descriptive Questions**

1. What are the different types of commonly used shells on a typical linux system?

On a typical Linux system, several shells are commonly used:

1. **Bash (Bourne Again Shell)**: The most widely used shell. It is the default shell for many Linux distributions due to its features and compatibility with sh (the original Bourne shell).
2. **sh (Bourne Shell)**: An older shell that is compatible with older Unix systems. Many modern systems have /bin/sh as a symlink to /bin/bash or another compatible shell.
3. **zsh (Z Shell)**: Known for its powerful features, including improved tab completion, better globbing, and customizable prompts. It's popular among users who need advanced features.
4. How do you find out what’s your shell?

**Using the echo Command**:

echo $SHELL

1. List the advantages and disadvantages of shell scripting.

**Advantages:**

1. **Simplicity**: Shell scripts are relatively simple to write and understand, making them accessible for users who need to automate tasks quickly.
2. **Automation**: Shell scripts can automate repetitive tasks, reducing manual effort and the potential for human error.
3. **Integration**: Shell scripts can easily integrate and interact with other programs and commands available on Unix-like systems.

**Disadvantages:**

1. **Error Handling**: Shell scripts often have limited error handling and debugging features compared to more advanced programming languages.
2. **Limited Functionality**: For complex tasks, shell scripts might become cumbersome. They are generally not as powerful or flexible as languages like Python or Perl.
3. **Portability Issues**: While shell scripts are generally portable, scripts that use advanced features or non-standard commands might not work on all systems or shells.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**