



S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT & RESEARCH, NAGPUR

Practical 03

Aim: Automate student marksheets generation, system information display, Fibonacci and prime number generation, and file management operations using shell scripts to enhance computational efficiency and user Interaction.

Name: Akansha Wanjari

USN: CM24004

Semester / Year: 4th/2nd

Academic Session: 2025-2026

Date of Performance:

Date of Submission:

- ❖ Aim: Automate student marksheet generation, system information display, Fibonacci and prime number generation, and file management operations using shell scripts to enhance computational efficiency and user interaction.

- Tasks to be done in this Practical.

- a) Write a shell script to generate mark- sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.
- b) Write a menu driven shell script which will print the following menu and execute the given task.
 - c) Display calendar of current month.
 - Display today's date and time.
 - CDisplay usernames those are currently logged in the system.
 - L]Display your terminal _ number
- c) Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3, 5, 13.
- d) Write a shell script which will accept a number b and display first n prime numbers as output.
- e) Write menu driven program for file handling activity
 - C]Creation of file.
 - Write content in the file.
 - Upend file content.
 - D Delete file content

- Objectives:

1. Automate marksheet generation with total, percentage, and class classification.
2. Develop menu-driven scripts for system information and file operations.
3. Generate Fibonacci and prime numbers for user-defined inputs.

- ❖ Requirements:

- ✓ Hardware Requirements:

- Processor: Minimum 1 GHz

- RAM: 512 MB or higher

- Storage: 100 MB free space

- Software Requirements:

- Operating

System: Linux/Unix-based

Shell: Bash 4.0 or higher

Text Editor: Nano, Vim, or any preferred editor

❖ Theory:

Shell scripting is a powerful way to automate repetitive tasks and manage system operations efficiently. It allows users to write programs using shell commands and scripting constructs. Shell scripts are interpreted line-by-line by a shell interpreter, making them ideal for administrative tasks, file management, and system automation. This practical encompasses a variety of real-world scenarios that demonstrate the utility of shell scripting for computing tasks and resource management.

1. Marksheets Generation

This script takes input marks for three subjects, calculates the total marks, percentage, and determines the class of the student based on predefined conditions. Conditional statements (if-else) are used to classify the performance into distinction, first class, second class, or fail. This exercise emphasizes the use of arithmetic operations and decision-making constructs. Key concepts include:

Reading user input using read

Arithmetic operations with \$((expression))

Conditional statements for decision-making

2. Menu-Driven Script for System Information

Menu-driven scripts enhance user interaction by presenting a list of options for performing different tasks. In this practical, options are provided to display the calendar of the current month, the current date and time, logged-in users, and the terminal number. The script utilizes looping constructs (while) and case statements for structured flow control. Commands used:

cal for displaying the calendar date for showing
current date and time who to list logged-in users • tty
to identify the terminal

3. Fibonacci Number Generation

Fibonacci numbers are a sequence where each term is the sum of the two preceding ones. The script uses iterative constructs (for loop) to generate terms based on user input. This practical illustrates the use of loop control and variable swapping to generate series data efficiently.

4. Prime Number Display

This script accepts an integer n and outputs the first n prime numbers. A nested loop checks divisibility to determine if a number is prime. The practical demonstrates logic building for number-theoretic operations using loops and conditionals.

5. Menu-Driven File Management

The file handling script enables users to create, write, append, and delete file content. The case construct _ manages different file operations. Commands include:

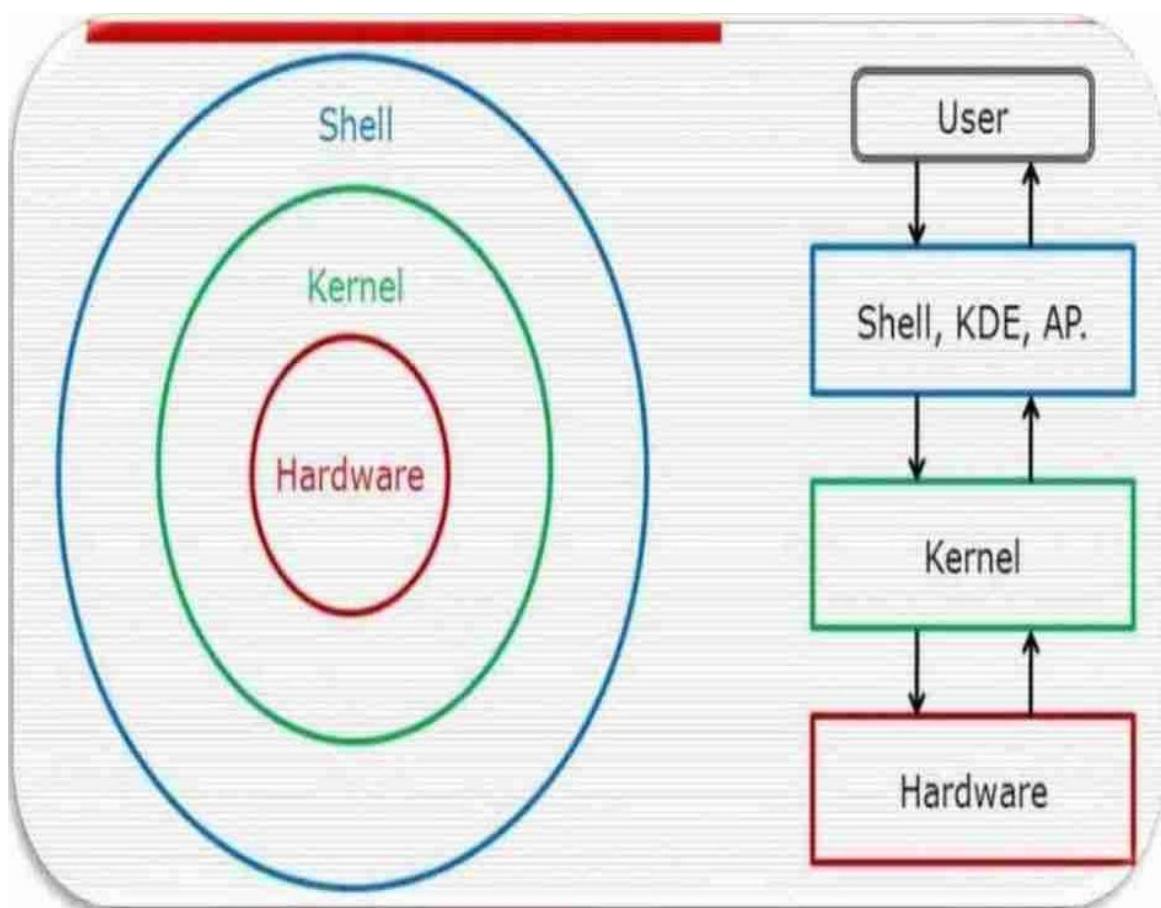
touch to create files cat for writing

and appending content

rm for deleting files

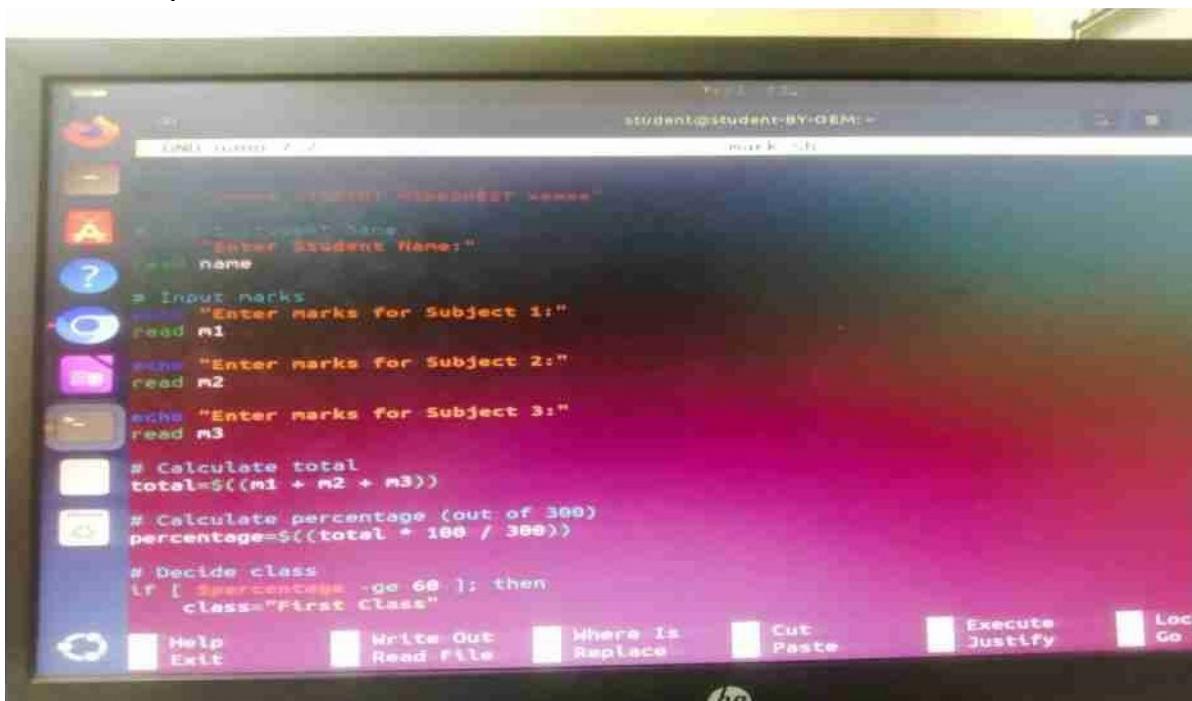
This exercise emphasizes text manipulation, input handling, and file control mechanisms in Unix-like environments.

Diagrammatical View of Shell



- I. Write a shell script to generate mark- sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.

Output I:



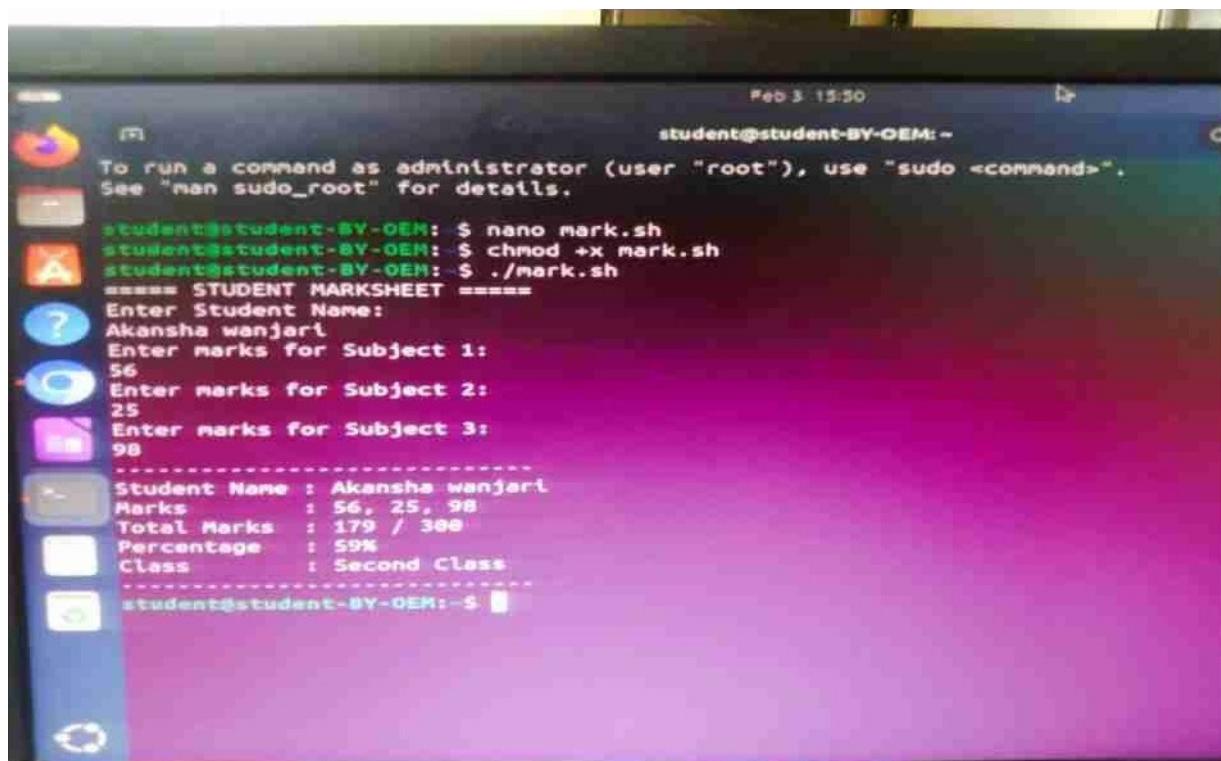
```

student@student-BY-OEM:~$ nano mark.sh
=====
# STUDENT MARKSHEET
=====
#!/bin/bash
# Enter Student Name:
name
# Input marks
# Enter marks for Subject 1:
read m1
# Enter marks for Subject 2:
read m2
# Enter marks for Subject 3:
read m3
# Calculate total
total=$((m1 + m2 + m3))
# Calculate percentage (out of 300)
percentage=$((total * 100 / 300))
# Decide class
if [ $percentage -ge 60 ]; then
    class="First Class"
else
    class="Second Class"
fi
=====

student@student-BY-OEM:~$ chmod +x mark.sh
student@student-BY-OEM:~$ ./mark.sh
===== STUDENT MARKSHEET =====
Enter Student Name:
Akansha wanjari
Enter marks for Subject 1:
56
Enter marks for Subject 2:
25
Enter marks for Subject 3:
98
=====
Student Name : Akansha wanjari
Marks        : 56, 25, 98
Total Marks  : 179 / 300
Percentage   : 59%
Class        : Second Class
=====

student@student-BY-OEM:~$ 

```



```

student@student-BY-OEM:~$ nano mark.sh
=====
# STUDENT MARKSHEET
=====
#!/bin/bash
# Enter Student Name:
name
# Input marks
# Enter marks for Subject 1:
read m1
# Enter marks for Subject 2:
read m2
# Enter marks for Subject 3:
read m3
# Calculate total
total=$((m1 + m2 + m3))
# Calculate percentage (out of 300)
percentage=$((total * 100 / 300))
# Decide class
if [ $percentage -ge 60 ]; then
    class="First Class"
else
    class="Second Class"
fi
=====

student@student-BY-OEM:~$ chmod +x mark.sh
student@student-BY-OEM:~$ ./mark.sh
===== STUDENT MARKSHEET =====
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@student-BY-OEM:~$ nano mark.sh
student@student-BY-OEM:~$ chmod +x mark.sh
student@student-BY-OEM:~$ ./mark.sh
===== STUDENT MARKSHEET =====
Enter Student Name:
Akansha wanjari
Enter marks for Subject 1:
56
Enter marks for Subject 2:
25
Enter marks for Subject 3:
98
=====
Student Name : Akansha wanjari
Marks        : 56, 25, 98
Total Marks  : 179 / 300
Percentage   : 59%
Class        : Second Class
=====

student@student-BY-OEM:~$ 

```

2. Write a menu driven shell script which will print the following menu and execute the given task.

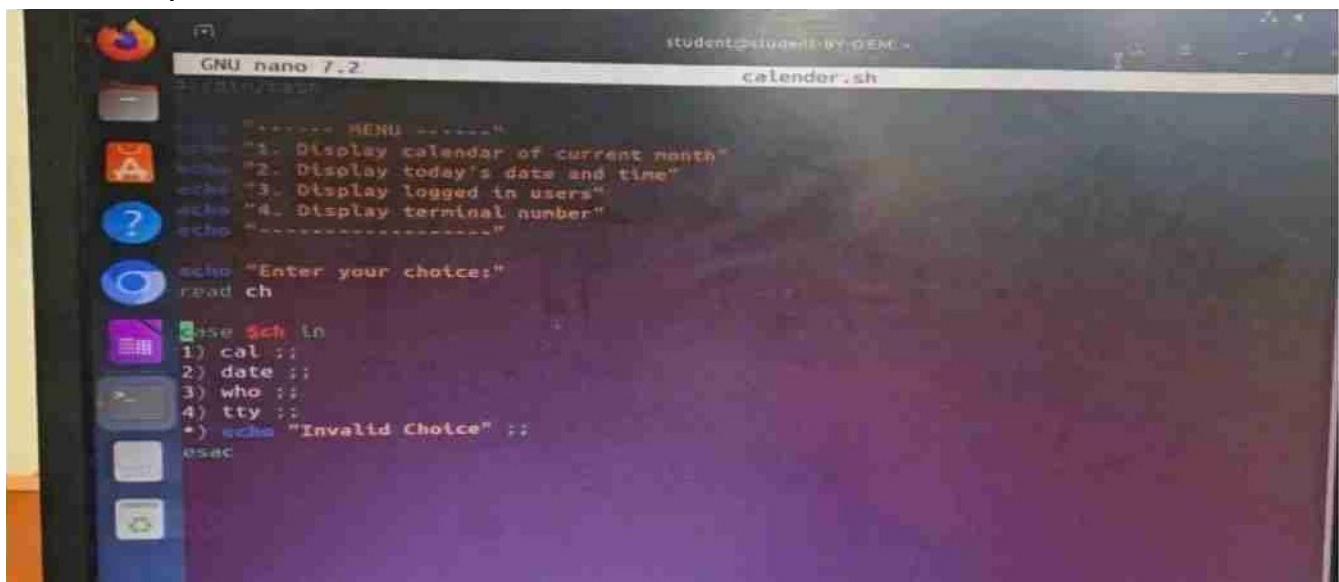
Display calendar of current month.

Display today's date and time.

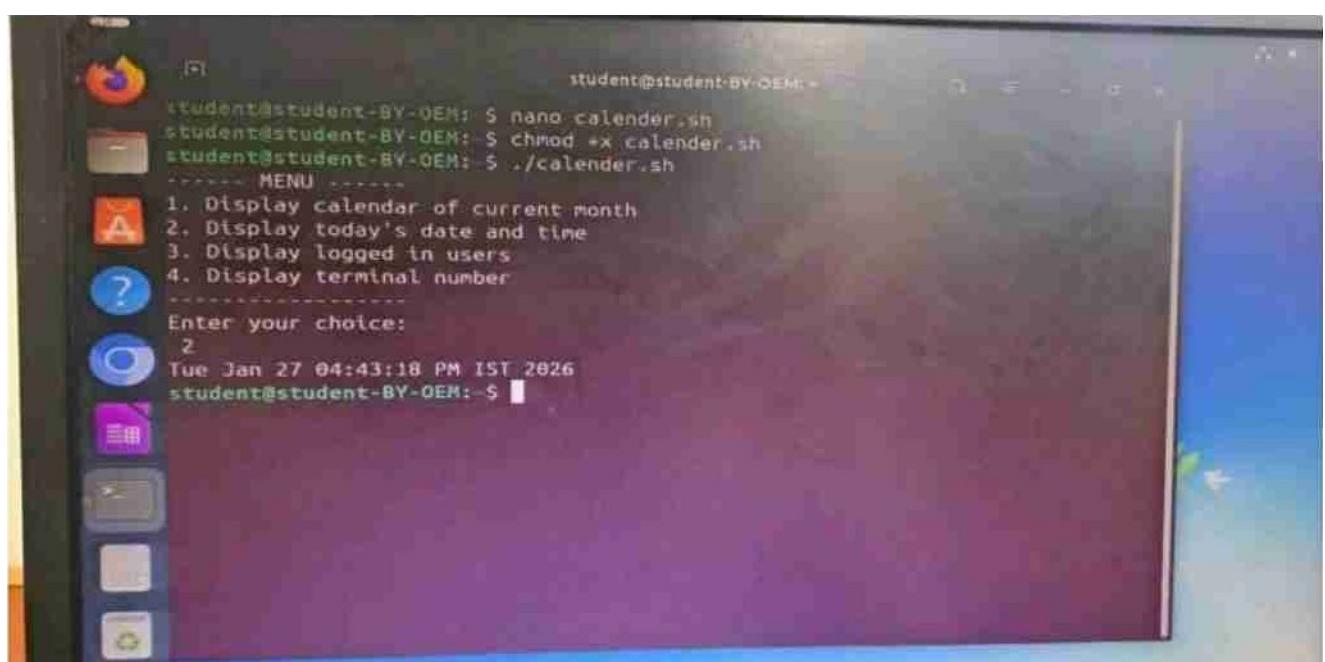
Display usernames those are currently logged in the system.

Display your terminal number

Output 2:



```
student@student-BY-OEM:~$ nano calender.sh
student@student-BY-OEM:~$ cat calender.sh
#!/bin/bash
echo "----- MENU -----"
echo "1. Display calendar of current month"
echo "2. Display today's date and time"
echo "3. Display logged in users"
echo "4. Display terminal number"
echo ""
echo "Enter your choice:"
read ch
case $ch in
    1) cal ;;
    2) date ;;
    3) who ;;
    4) tty ;;
    *) echo "Invalid Choice" ;;
esac
```



```
student@student-BY-OEM:~$ nano calender.sh
student@student-BY-OEM:~$ chmod +x calender.sh
student@student-BY-OEM:~$ ./calender.sh
----- MENU -----
1. Display calendar of current month
2. Display today's date and time
3. Display logged in users
4. Display terminal number
Enter your choice:
2
Tue Jan 27 04:43:18 PM IST 2026
student@student-BY-OEM:~$
```

3. Write a shell script which will generate first n fibonacci numbers like:

1 1 2 3 5 13.

Output 3:

The screenshot shows a terminal window titled "student@student-BY-OEM: ~". The file being edited is "calender.sh". The script contains the following code:

```
GNU nano 7.2
student@student-BY-OEM: ~
calender.sh

----- MENU -----
echo "1. Display calendar of current month"
echo "2. Display today's date and time"
echo "3. Display logged in users"
echo "4. Display terminal number"
echo "-----"

echo "Enter your choice:"
read ch

case $ch in
    1) cal ;;
    2) date ;;
    3) who ;;
    4) tty ;;
    *) echo "Invalid Choice" ;;
esac
```

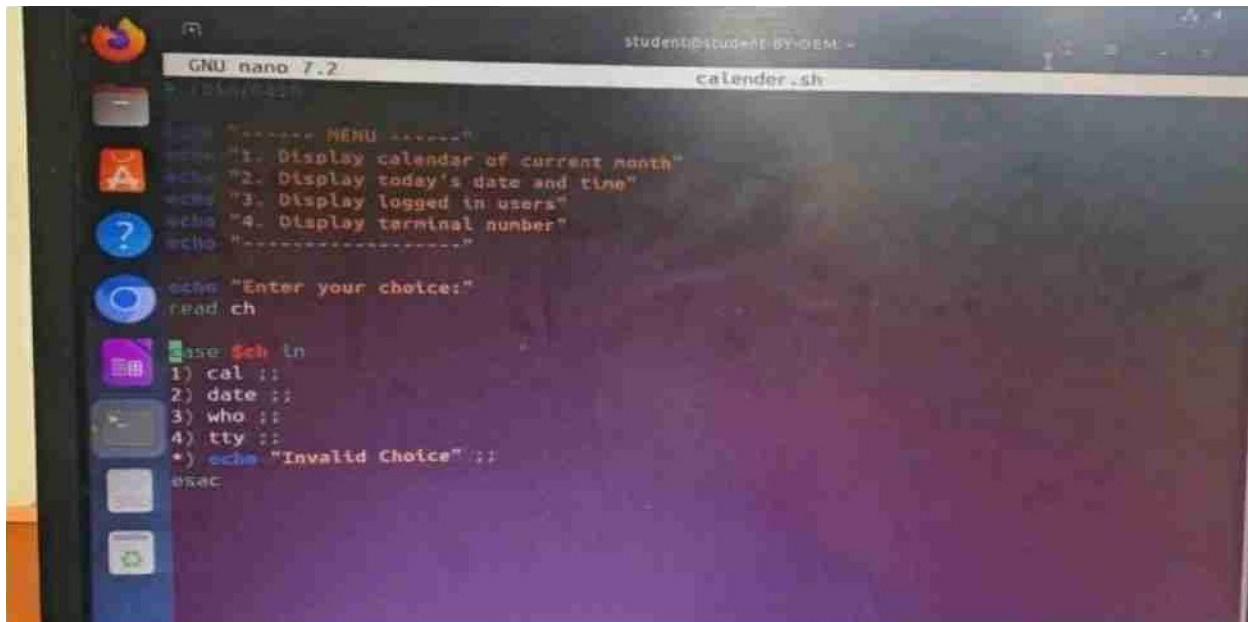
4.

The screenshot shows a terminal window titled "student@student-BY-OEM: ~". The command "nano calender.sh" was run, followed by "chmod +x calender.sh", and finally "./calender.sh". The output of the script is displayed:

```
student@student-BY-OEM: ~ nano calender.sh
student@student-BY-OEM: ~ chmod +x calender.sh
student@student-BY-OEM: ~ ./calender.sh
----- MENU -----
1. Display calendar of current month
2. Display today's date and time
3. Display logged in users
4. Display terminal number
-----
Enter your choice:
2
Tue Jan 27 04:43:18 PM IST 2026
student@student-BY-OEM: ~
```

4. Write a shell script which will accept a number b and display first n prime numbers as output.

Output 4:

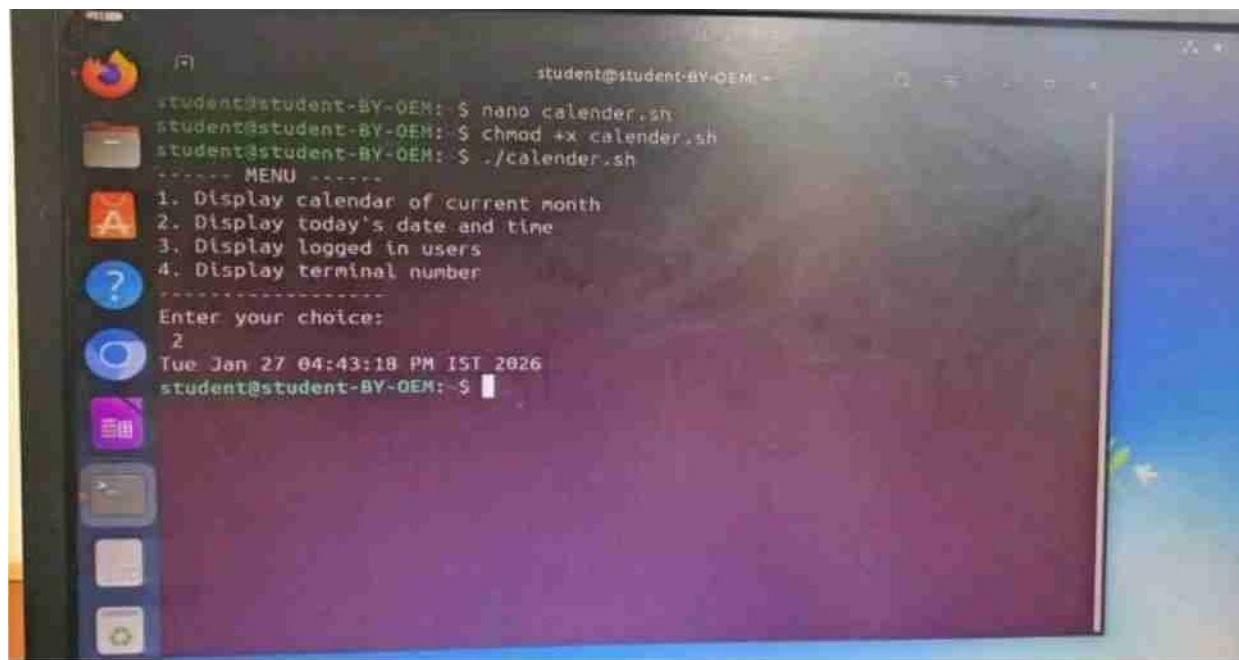


The screenshot shows a terminal window titled "student@student-BY-OEM ~". Inside the terminal, the command "nano calender.sh" is run, opening a file named "calender.sh". The script content is as follows:

```
GNU nano 7.2
student@student-BY-OEM ~
calender.sh

----- MENU -----
1. Display calendar of current month
2. Display today's date and time
3. Display logged in users
4. Display terminal number
echo "-----"
echo "Enter your choice:"
read ch

case $ch in
    1) cal ;;
    2) date ;;
    3) who ;;
    4) tty ;;
    *) echo "Invalid Choice" ;;
esac
```



The screenshot shows a terminal window titled "student@student-BY-OEM ~". The user runs the command "nano calender.sh" to edit the script. After saving and exiting nano, they run "chmod +x calender.sh" to make it executable. Finally, they execute the script with "./calender.sh". The output of the script is displayed, showing the menu options and the user's choice of 2, which outputs the current date and time.

```
student@student-BY-OEM ~
student@student-BY-OEM ~$ nano calender.sh
student@student-BY-OEM ~$ chmod +x calender.sh
student@student-BY-OEM ~$ ./calender.sh
----- MENU -----
1. Display calendar of current month
2. Display today's date and time
3. Display logged in users
4. Display terminal number
-----
Enter your choice:
2
Tue Jan 27 04:43:18 PM IST 2026
student@student-BY-OEM ~$
```

5. Write menu driven program for file handling activity

- Creation, of file, Write content in the fileUpend file content,
Delete file content

Output 5:

```
GNU nano 2.2
#!/bin/bash
----- FILE MENU -----
1. Create File
2. Write Content
3. Append Content
4. Delete File Content
-----
echo "Enter choice"
read ch
echo "Enter file name"
read fname
case $ch in
    1) touch $fname
        echo "File Created"
        ;;
    2) echo "Enter content"
        cat > $fname
        echo "Content Written"
        ;;
    3) echo "Enter content to append"
        cat >> $fname
        echo "Content Appended"
        ;;
    4) rm $fname
        echo "File Content Deleted"
        ;;
    *) echo "Invalid Choice"
        ;;
esac
```

```
GNU nano 2.2
#!/bin/bash
----- FILE MENU -----
1. Create File
2. Write Content
3. Append Content
4. Delete File Content
-----
echo "Enter choice"
read ch
echo "Enter file name"
read fname
case $ch in
    1) touch $fname
        echo "File Created"
        ;;
    2) echo "Enter content"
        cat > $fname
        echo "Content Written"
        ;;
    3) echo "Enter content to append"
        cat >> $fname
        echo "Content Appended"
        ;;
    4) rm $fname
        echo "File Content Deleted"
        ;;
    *) echo "Invalid Choice"
        ;;
esac
```

```
student@student-BY-DEMI:~$ nano file.sh
student@student-BY-DEMI:~$ chmod +x file.sh
student@student-BY-DEMI:~$ ./file.sh
----- FILE MENU -----
1. Create File
2. Write Content
3. Append Content
4. Delete File Content
-----
Enter choice:
1
Enter file name:
dev.txt
File Created
student@student-BY-DEMI:~$ nano file.sh
```

- Conclusion: Automating tasks like student marksheet generation, system information display, and numerical computations using shell scripts enhances computational efficiency and user interaction reduce errors, and Improve productivity in Linux/UNIX environments. By leveraging shell scripting, users can create customized solutions for various administrative and computational needs.

'Discussion Questions:

1. How do shell scripts Improve efficiency in student marksheet generation?
2. What system information can be displayed using shell scripts?
3. How do you generate Fibonacci numbers using shell scripts?
4. What file management operations can be automated using shell scripts?
5. How to enhance user interaction in _shell scripts?

+ References:

https://www.tutorialspoint.com/lhmix/shell_scriptinghtml
<https://www.javatpoint.com/shell-scripting-tutorial>