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

Practical 03

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.

Name: Pranav Mohan Paunikar

USN:CM 24032

Semester / Year: IV/II

Academic Session: 2025-26

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.
   - Display calendar of current month.
   - Display today's date and time.
   - Display usernames those are currently logged in the system.
   - 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
   - Creation of file.
   - Write content in the file.
   - Upend file content.
   - 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. Marksheet 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 n 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.

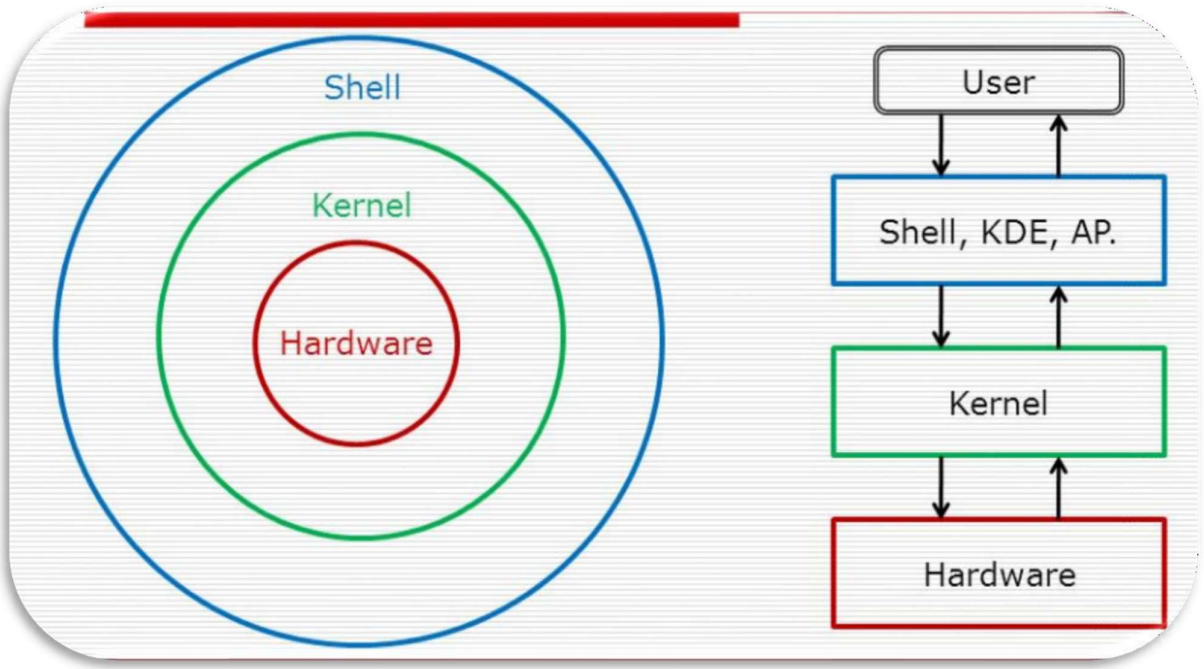
5. Menu-Driven File Management

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**



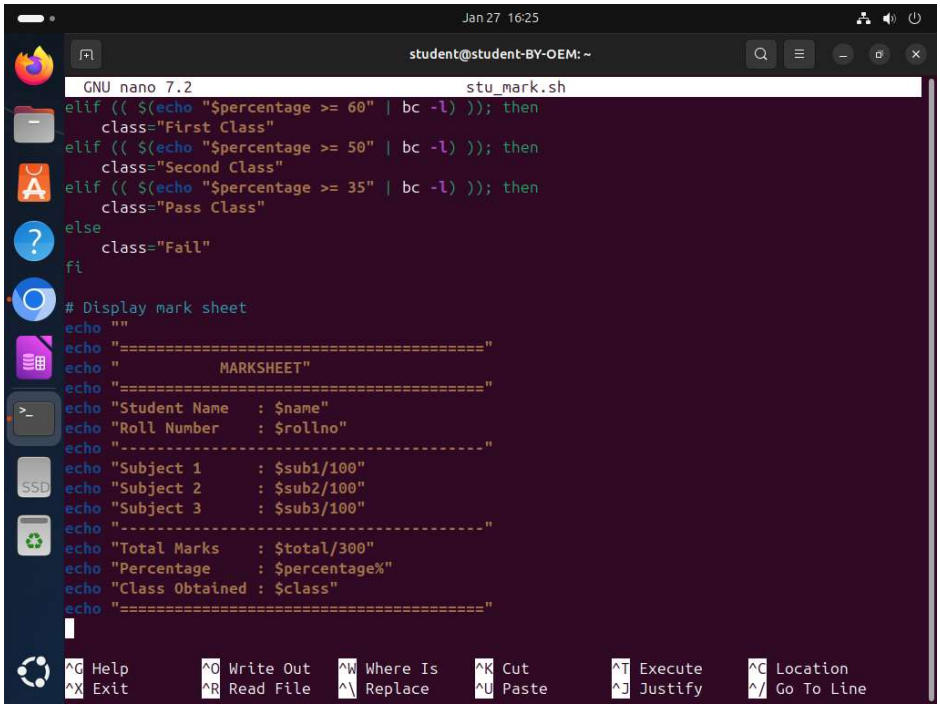
❖ CODES

1. **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 1:

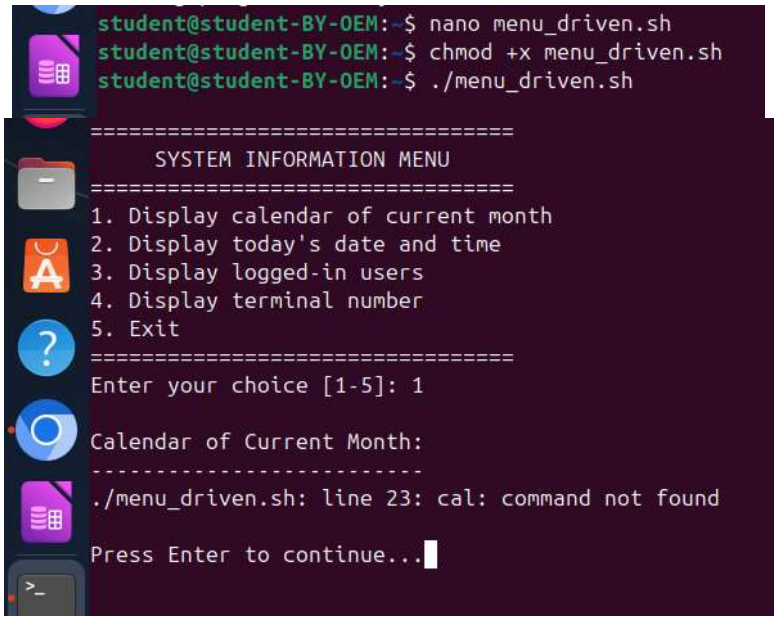
```
student@student-BY-OEM:~$ nano stu_mark.sh
student@student-BY-OEM:~$ chmod +x stu_mark.sh
student@student-BY-OEM:~$ ./stu_mark.sh
STUDENT MARK SHEET GENERATOR
Student Name: Pranav
Roll Number: CM24032
Enter marks for 3 subjects
Subject 1: 50
Subject 2: 70
Subject 3: 60

=======================================
          MARKSHEET
=======================================
Student Name   : Pranav
Roll Number    : CM24032
---------------------------------------
Subject 1      : 50/100
Subject 2      : 70/100
Subject 3      : 60/100
---------------------------------------
Total Marks    : 180/300
Percentage     : 60.00%
Class Obtained : First Class
=======================================
student@student-BY-OEM:~$ S
```

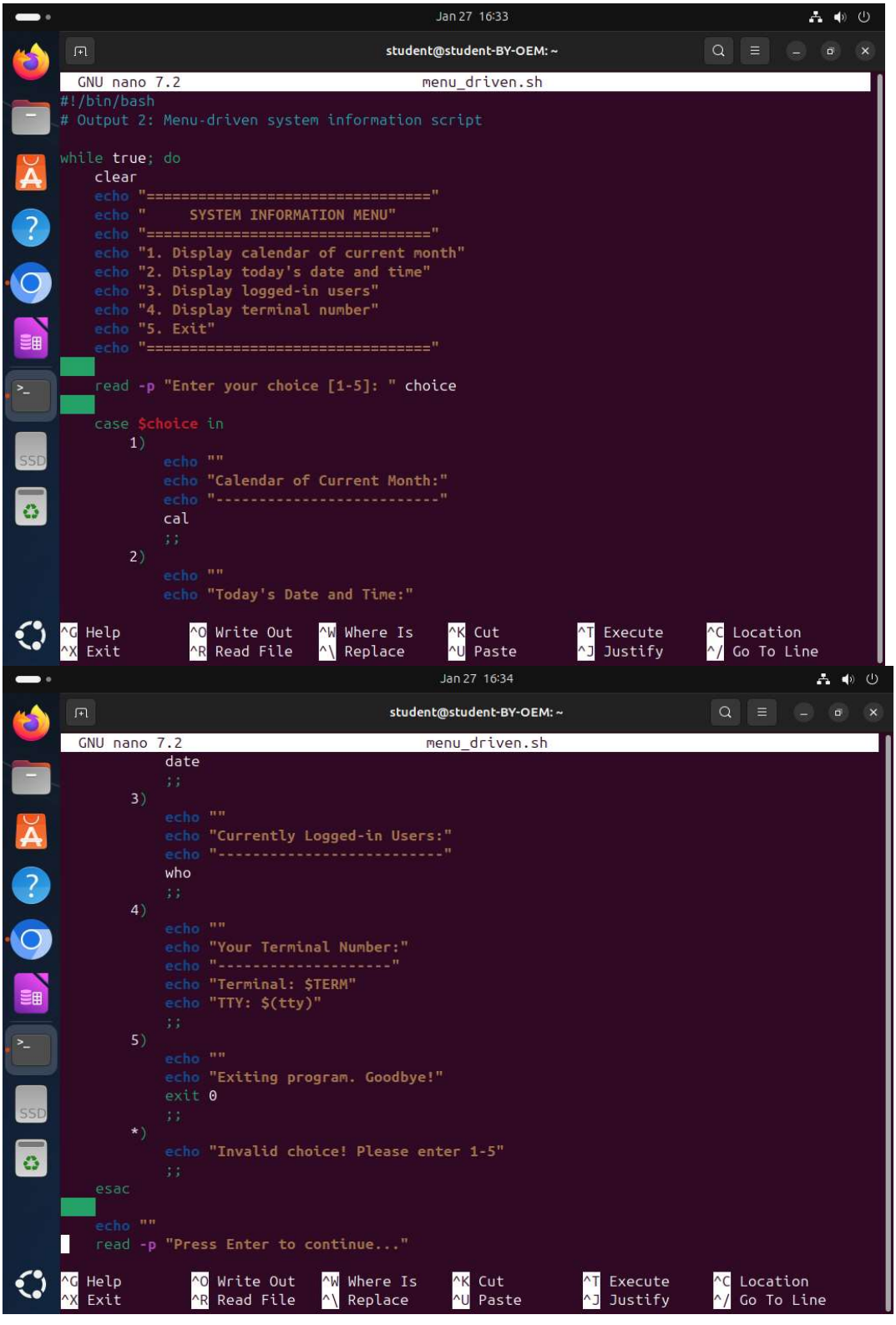
```
#!/bin/bash
# Output 1: Student Mark Sheet Generator

echo "STUDENT MARK SHEET GENERATOR"

# Input student details
read -p "Student Name: " name
read -p "Roll Number: " rollno

# Input marks for 3 subjects
echo "Enter marks for 3 subjects"
read -p "Subject 1: " sub1
read -p "Subject 2: " sub2
read -p "Subject 3: " sub3

# Calculate total and percentage
total=$((sub1 + sub2 + sub3))
percentage=$(echo "scale=2; $total / 3" | bc)

# Determine class
if (( $(echo "$percentage >= 75" | bc -l) )); then
    class="Distinction"
elif (( $(echo "$percentage >= 60" | bc -l) )); then
    class="First Class"
elif (( $(echo "$percentage >= 50" | bc -l) )); then
    class="Second Class"
elif (( $(echo "$percentage >= 35" | bc -l) )); then

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```



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:




Jan 27 16:33
student@student-BY-OEM: ~
GNU nano 7.2 menu_driven.sh
#!/bin/bash
# Output 2: Menu-driven system information script
while true; do
clear
echo "================================"
echo " SYSTEM INFORMATION MENU"
echo "================================"
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 "5. Exit"
echo "================================"
read -p "Enter your choice [1-5]: " choice
case $choice in
1)
echo ""
echo "Calendar of Current Month:"
echo "-------------------------"
cal
;;
2)
echo ""
echo "Today's Date and Time:"
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line
Jan 27 16:34
student@student-BY-OEM: ~
GNU nano 7.2 menu_driven.sh
date
;;
3)
echo ""
echo "Currently Logged-in Users:"
echo "-------------------------"
who
;;
4)
echo ""
echo "Your Terminal Number:"
echo "--------------------"
echo "Terminal: $TERM"
echo "TTY: $(tty)"
;;
5)
echo ""
echo "Exiting program. Goodbye!"
exit 0
;;
*)
echo "Invalid choice! Please enter 1-5"
;;
esac
echo ""
read -p "Press Enter to continue..."
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

3. **Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3, 5, 13**

Output 3:

```
student@student-BY-OEM:~$ nano fibo.sh
student@student-BY-OEM:~$ chmod +x fibo.sh
student@student-BY-OEM:~$ ./fibo.sh
=== FIBONACCI SEQUENCE GENERATOR ===

Enter the number of terms (n): 7
Fibonacci sequence (7 terms):
1, 1, 2, 3, 5, 8, 13
student@student-BY-OEM:~$
```

```
student@student-BY-OEM: ~
  GNU nano 7.2                                  fibo.sh *
#!/bin/bash
# Output 3: Fibonacci Sequence Generator

echo "=== FIBONACCI SEQUENCE GENERATOR ==="
echo ""
read -p "Enter the number of terms (n): " n

# Validate input
if ! [[ "$n" =~ ^[1-9][0-9]*$ ]]; then
    echo "Error: Please enter a positive integer!"
    exit 1
fi

if [ $n -eq 1 ]; then
    echo "Fibonacci sequence (1 term):"
    echo "1"
elif [ $n -eq 2 ]; then
    echo "Fibonacci sequence (2 terms):"
    echo "1, 1"
else
    echo "Fibonacci sequence ($n terms):"

    a=1
    b=1
    echo -n "1, 1"

    for (( i=3; i<=n; i++ ))
    do
        c=$((a + b))
        echo -n ", $c"
        a=$b
        b=$c
    done
    echo ""
fi

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

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

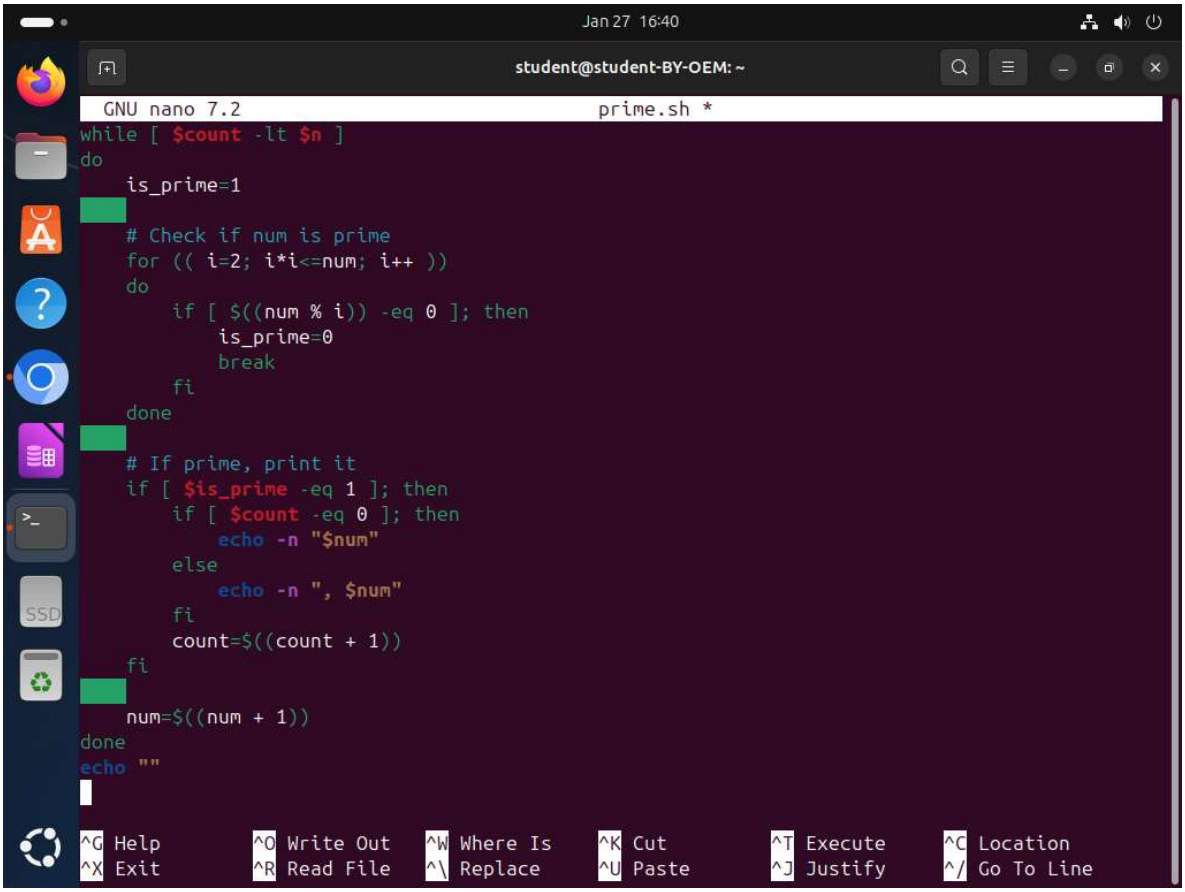
Output 4:

```
student@student-BY-OEM:~$ nano prime.sh
student@student-BY-OEM:~$ chmod +x prime.sh
student@student-BY-OEM:~$ .prime.sh
.prime.sh: command not found
student@student-BY-OEM:~$ ./prime.sh
=== PRIME NUMBERS GENERATOR ===

Enter the number of primes to generate (n): 8
First 8 prime numbers:
2, 3, 5, 7, 11, 13, 17, 19
student@student-BY-OEM:~$
```

Jan 27 16:40

student@student-BY-OEM: ~

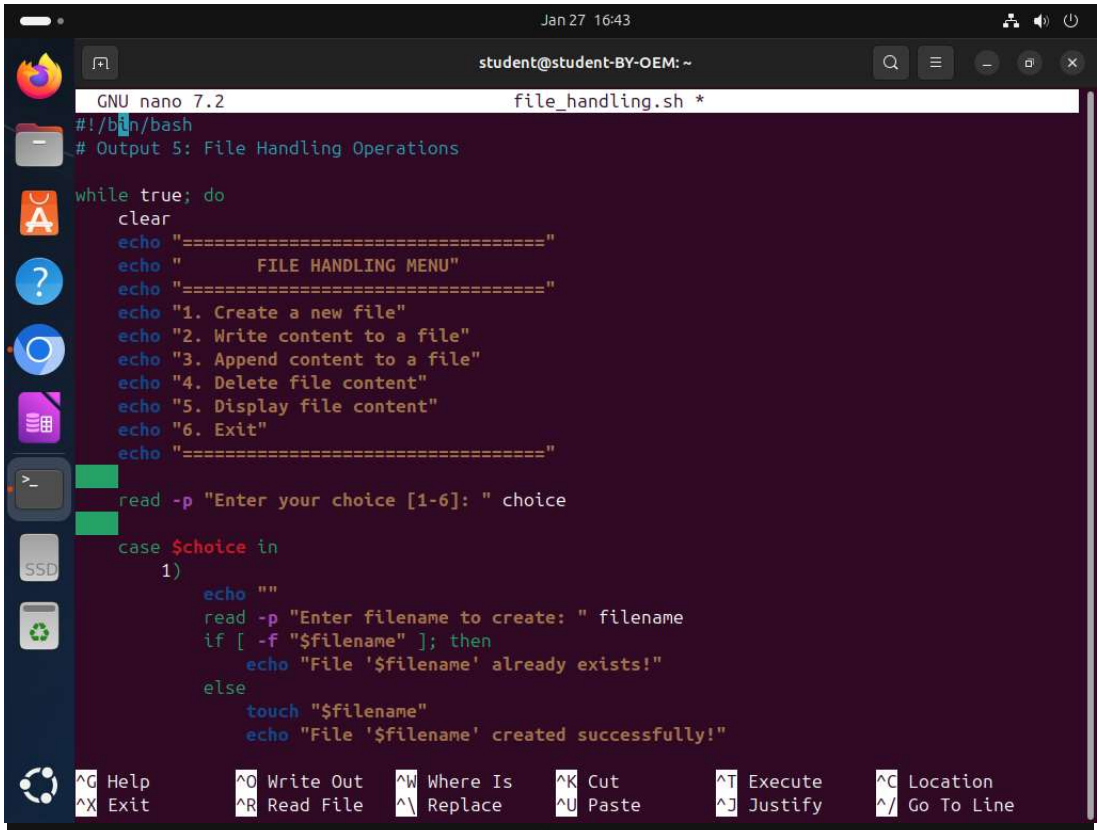
```
 GNU nano 7.2                                  prime.sh *
#!/bin/bash
# Output 4: First n Prime Numbers Generator

echo "=== PRIME NUMBERS GENERATOR ==="
echo ""
read -p "Enter the number of primes to generate (n): " n

# Validate input
if ! [[ "$n" =~ ^[1-9][0-9]*$ ]]; then
    echo "Error: Please enter a positive integer!"
    exit 1
fi

echo "First $n prime numbers:"

count=0
num=2

while [ $count -lt $n ]
do
    is_prime=1

    # Check if num is prime
    for (( i=2; i*i<=num; i++ ))
    do
        if [ $((num % i)) -eq 0 ]; then
            is_prime=0

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

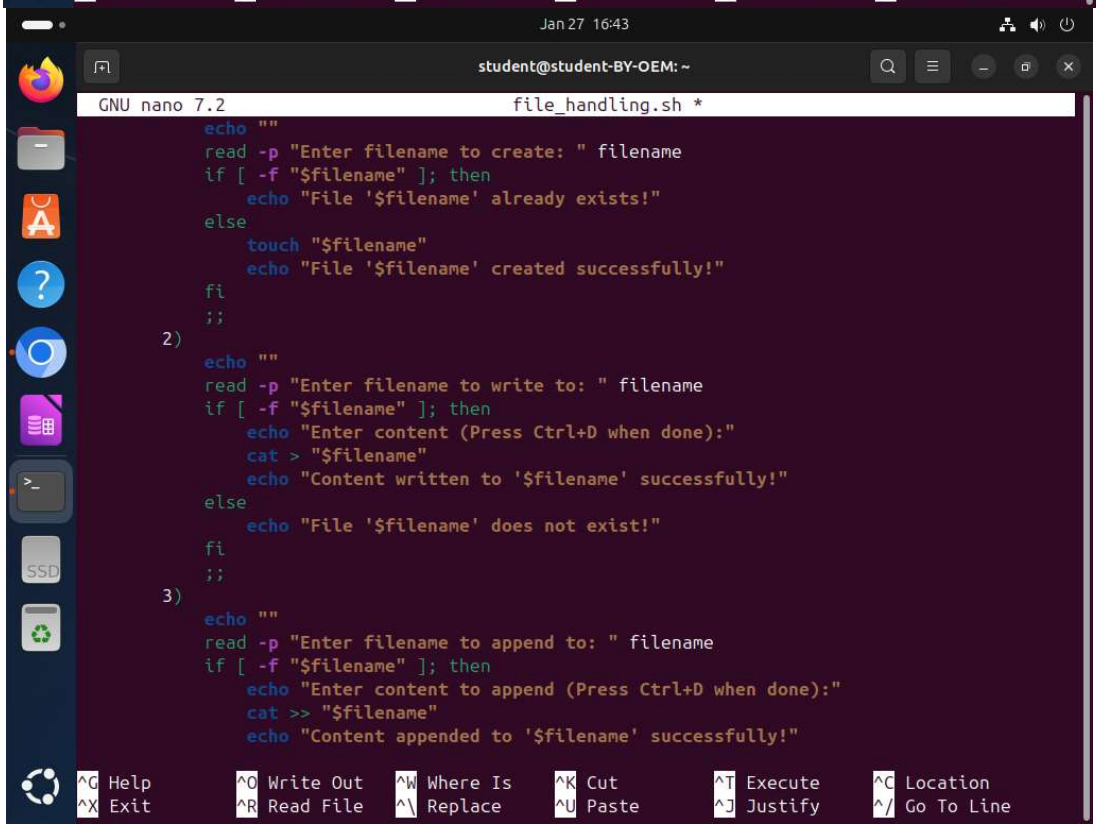

Jan 27 16:40
student@student-BY-OEM: ~
GNU nano 7.2 prime.sh *
while [$count -lt $n]
do
is_prime=1
# Check if num is prime
for ((i=2; i*i<=num; i++))
do
if [$((num % i)) -eq 0]; then
is_prime=0
break
fi
done
# If prime, print it
if [$is_prime -eq 1]; then
if [$count -eq 0]; then
echo -n "$num"
else
echo -n ", $num"
fi
count=$((count + 1))
fi
num=$((num + 1))
done
echo ""
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

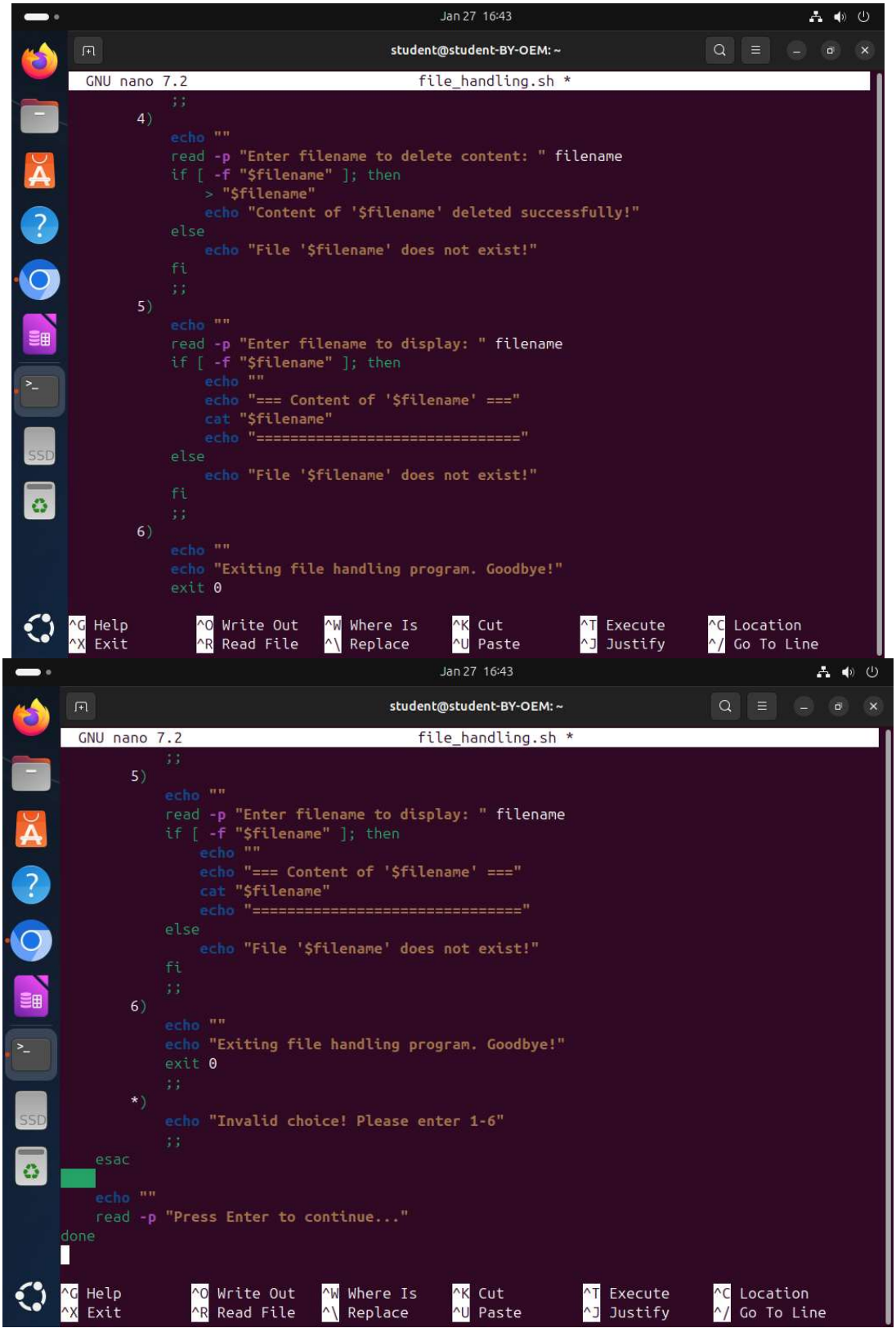
5. Write menu driven program for file handling activity
   - Creation of file.
   - Write content in the file.
   - Upend file content.
   - Delete file content

Output 5:

```
student@student-BY-OEM:~$ nano file handling
student@student-BY-OEM:~$ nano file_handling.sh
student@student-BY-OEM:~$
student@student-BY-OEM:~$ chmod +x file_handling.sh
student@student-BY-OEM:~$ ./file_handling.sh
================================
      FILE HANDLING MENU
================================
1. Create a new file
2. Write content to a file
3. Append content to a file
4. Delete file content
5. Display file content
6. Exit
================================
Enter your choice [1-6]: 1

Enter filename to create: pranavCM24032
File 'pranavCM24032' created successfully!

Press Enter to continue...
```


Jan 27 16:43
student@student-BY-OEM: ~
GNU nano 7.2 file_handling.sh *
#!/bin/bash
# Output 5: File Handling Operations
while true; do
clear
echo "================================="
echo " FILE HANDLING MENU"
echo "================================="
echo "1. Create a new file"
echo "2. Write content to a file"
echo "3. Append content to a file"
echo "4. Delete file content"
echo "5. Display file content"
echo "6. Exit"
echo "================================="
read -p "Enter your choice [1-6]: " choice
case $choice in
1)
echo ""
read -p "Enter filename to create: " filename
if [-f "$filename"]; then
echo "File '$filename' already exists!"
else
touch "$filename"
echo "File '$filename' created successfully!"
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line


Jan 27 16:43
student@student-BY-OEM: ~
GNU nano 7.2 file_handling.sh *
echo ""
read -p "Enter filename to create: " filename
if [-f "$filename"]; then
echo "File '$filename' already exists!"
else
touch "$filename"
echo "File '$filename' created successfully!"
fi
;;
2)
echo ""
read -p "Enter filename to write to: " filename
if [-f "$filename"]; then
echo "Enter content (Press Ctrl+D when done):"
cat > "$filename"
echo "Content written to '$filename' successfully!"
else
echo "File '$filename' does not exist!"
fi
;;
3)
echo ""
read -p "Enter filename to append to: " filename
if [-f "$filename"]; then
echo "Enter content to append (Press Ctrl+D when done):"
cat >> "$filename"
echo "Content appended to '$filename' successfully!"
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line


Jan 27 16:43
student@student-BY-OEM: ~
GNU nano 7.2 file_handling.sh *
;;
4)
echo ""
read -p "Enter filename to delete content: " filename
if [-f "$filename"]; then
> "$filename"
echo "Content of '$filename' deleted successfully!"
else
echo "File '$filename' does not exist!"
fi
;;
5)
echo ""
read -p "Enter filename to display: " filename
if [-f "$filename"]; then
echo ""
echo "=== Content of '$filename' ==="
cat "$filename"
echo "=============================="
else
echo "File '$filename' does not exist!"
fi
;;
6)
echo ""
echo "Exiting file handling program. Goodbye!"
exit 0
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

Jan 27 16:43
student@student-BY-OEM: ~
GNU nano 7.2 file_handling.sh *
;;
5)
echo ""
read -p "Enter filename to display: " filename
if [-f "$filename"]; then
echo ""
echo "=== Content of '$filename' ==="
cat "$filename"
echo "=============================="
else
echo "File '$filename' does not exist!"
fi
;;
6)
echo ""
echo "Exiting file handling program. Goodbye!"
exit 0
;;
*)
echo "Invalid choice! Please enter 1-6"
;;
esac
echo ""
read -p "Press Enter to continue..."
done
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line