

## 1. Marksheets Generation

A screenshot of a terminal window titled "sheet.sh". The script content is as follows:

```
#!/bin/bash
echo "Enter marks of Subject 1:"
read m1
echo "Enter marks of Subject 2:"correct Usage;
read m2
echo "Enter marks of Subject 3:"add execute permissions to a script, you would run:
read m3
total=$((m1 + m2 + m3))
percentage=$((total * 100 / 300))
echo "Total Marks = $total"
echo "Percentage = $percentage%"his command tells the system to add execute permissions to the file sheet.sh. After running it, you should
if [ $percentage -ge 75 ]      be able to execute the script like so:
then
    echo "Class: Distinction"
elif [ $percentage -ge 60 ]
then
    echo "Class: First Class"
elif [ $percentage -ge 50 ]
then
    echo "Class: Second Class"
else
    echo "Class: Fail"
fi
```

The terminal window has a dark theme. The status bar at the bottom shows various keyboard shortcuts for file operations like "Attach", "Search", "Study", "Create Image", "Voice", and "Help". A message from ChatGPT is visible at the bottom of the screen.

### Output:-

A screenshot of a terminal window titled "(kali㉿kali)-[~]". The session starts with a few commands:

```
$ echo $0
/usr/bin/zsh
$ nano sheet.sh
```

Then, the user runs the script:

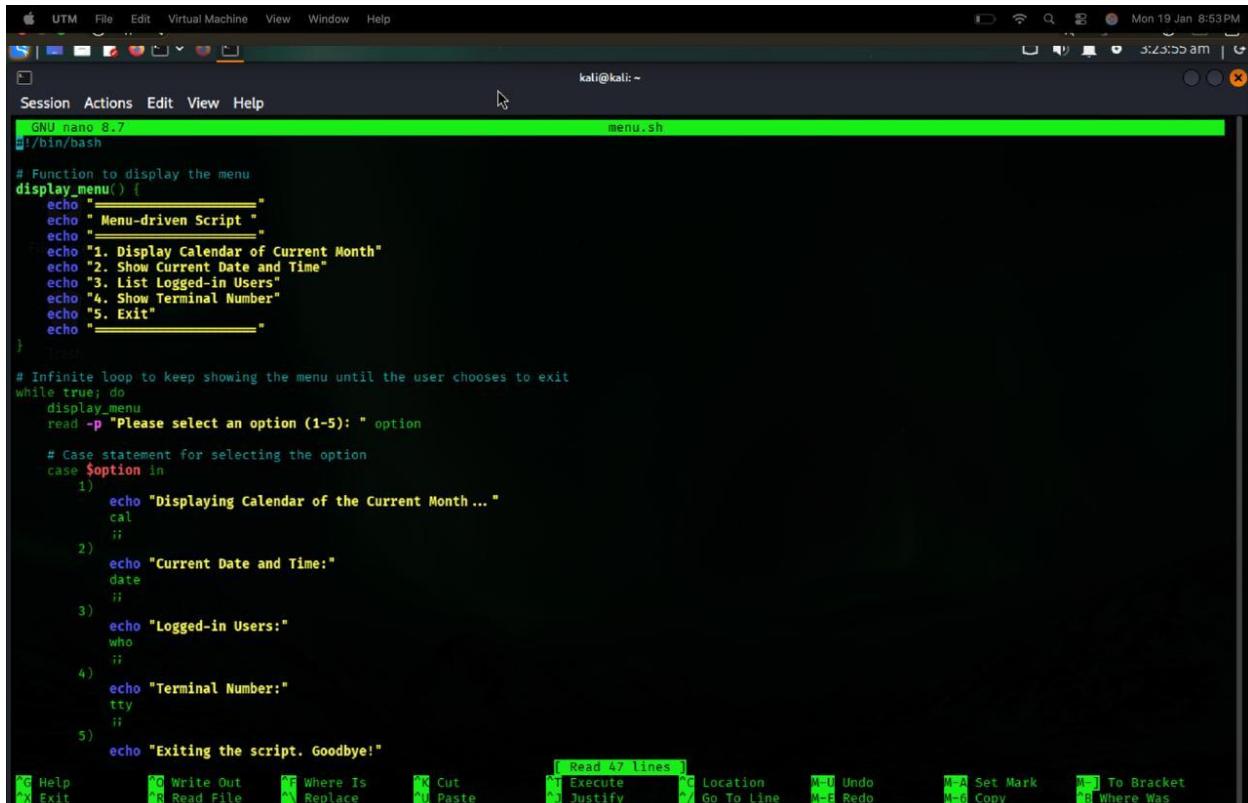
```
Correct Usage:
$ bash sheet.sh
```

The script prompts for subject marks and calculates the total and percentage:

```
Enter marks of Subject 1:
65
Enter marks of Subject 2:
78
Enter marks of Subject 3:
89
Total Marks = 232
Percentage = 77%
Class: Distinction
```

The terminal window has a dark theme. The status bar at the bottom shows various keyboard shortcuts for file operations like "Attach", "Search", "Study", "Create Image", "Voice", and "Help". A message from ChatGPT is visible at the bottom of the screen.

## 2. Menu-Driven Script for System Information.



```
GNU nano 8.7
menu.sh

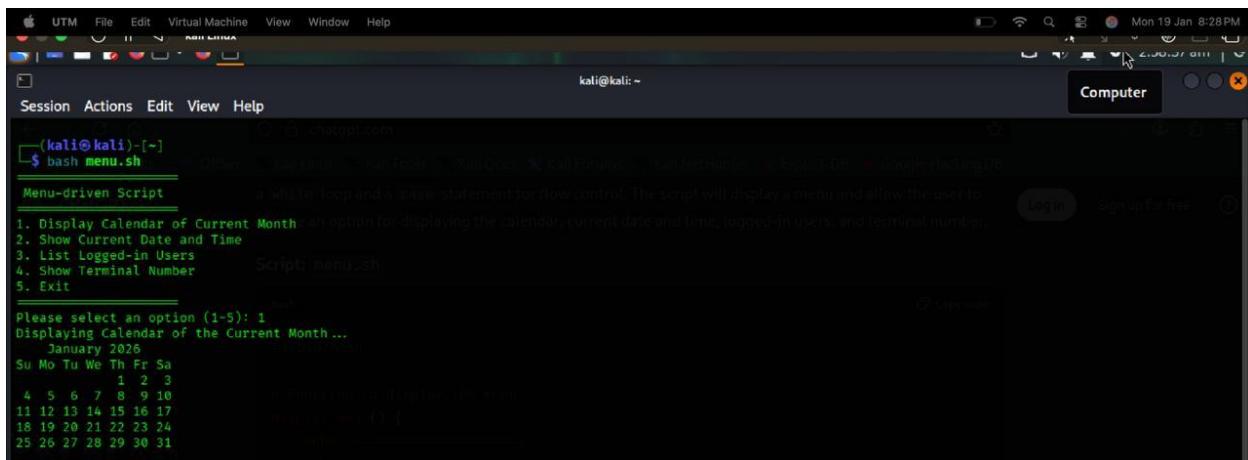
# Function to display the menu
display_menu() {
    echo "____"
    echo " Menu-driven Script "
    echo "____"
    echo "1. Display Calendar of Current Month"
    echo "2. Show Current Date and Time"
    echo "3. List Logged-in Users"
    echo "4. Show Terminal Number"
    echo "5. Exit"
    echo "____"
}

# Infinite loop to keep showing the menu until the user chooses to exit
while true; do
    display_menu
    read -p "Please select an option (1-5): " option

    # Case statement for selecting the option
    case $option in
        1)
            echo "Displaying Calendar of the Current Month ..."
            cal
            ;;
        2)
            echo "Current Date and Time:"
            date
            ;;
        3)
            echo "Logged-in Users:"
            who
            ;;
        4)
            echo "Terminal Number:"
            tty
            ;;
        5)
            echo "Exiting the script. Goodbye!"
            exit 0
            ;;
        *)
            echo "Invalid option! Please select a valid option between 1 and 5."
            ;;
    esac
done
```



### Output:-



```
kali@kali: ~
$ bash menu.sh
_____
Menu-driven Script
_____
1. Display Calendar of Current Month
2. Show Current Date and Time
3. List Logged-in Users
4. Show Terminal Number
5. Exit
_____
Please select an option (1-5): 1
Displaying Calendar of the Current Month...
January 2028
Su Mo Tu We Th Fr Sa
      1  2  3
4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

```
(kali㉿kali)-[~] $ bash menu.sh
Menu-driven Script
1. Display Calendar of Current Month
2. Show Current Date and Time
3. List Logged-in Users
4. Show Terminal Number
5. Exit

Please select an option (1-5): 2
Current Date and Time:
Mon Jan 19 09:59:00 AM EST 2026
```

```
(kali㉿kali)-[~] $ bash menu.sh
Menu-driven Script
1. Display Calendar of Current Month
2. Show Current Date and Time
3. List Logged-in Users
4. Show Terminal Number
5. Exit

Please select an option (1-5): 3
Logged-in Users:
kali      seat0    2026-01-19 09:04 (:0)
```

```
(kali㉿kali)-[~] $ bash menu.sh
Menu-driven Script
1. Display Calendar of Current Month
2. Show Current Date and Time
3. List Logged-in Users
4. Show Terminal Number
5. Exit

Please select an option (1-5): 4
Terminal Number:
/dev/pts/0
```

```
(kali㉿kali)-[~] $ bash menu.sh
Menu-driven Script
1. Display Calendar of Current Month
2. Show Current Date and Time
3. List Logged-in Users
4. Show Terminal Number
5. Exit

Please select an option (1-5): 5
Exiting the script. Goodbye!
```

### **3. Fibonacci Number Generation.**

The screenshot shows a terminal window titled "fabonacci.sh \*". The script, named "fabonacci.sh", is a shell script that generates a Fibonacci sequence. It starts by prompting the user for the number of terms. If the input is not a valid positive integer, it exits with an error message. Otherwise, it initializes variables "a" and "b" to 0 and 1 respectively, and then enters a loop. In each iteration of the loop, it prints the current value of "a", then calculates the next term as the sum of "a" and "b", and updates "a" and "b" for the next iteration. The loop continues until the user-specified number of terms is reached. The script concludes with an empty line.

```
#!/bin/bash

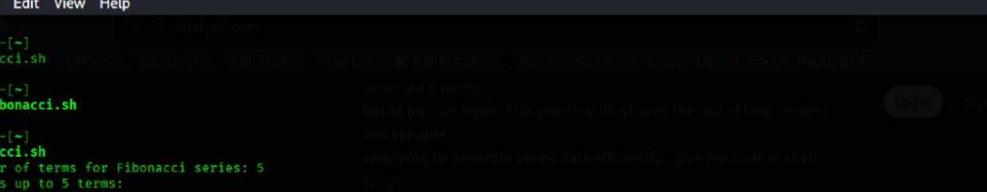
generate_fibonacci()
{
    read -p "Enter the number of terms for Fibonacci series: " n
    if ! [[ "$n" =~ ^[0-9]+ ]] || ! [ "$n" -le 0 ]; then
        echo "Please enter a valid positive integer."
        exit 1
    fi

    a=0
    b=1

    echo "Fibonacci Series up to $n terms:" simple shell script that generates the Fibonacci sequence using an iterative approach with a
    for ((i = 0; i < n; i++)); do
        echo -n "$a "
        next=$((a + b))
        a=$b
        b=$next
    done
    echo ""
}

generate_fibonacci
```

## **Output:-**



A screenshot of a terminal window titled "UTM" on a Mac OS X desktop. The terminal shows a user named "kali" running a shell script named "fibonacci.sh". The script prompts the user for the number of terms and then prints the Fibonacci series up to that number. The terminal window has a dark theme with green text and a black background.

```
(kali㉿kali)-[~] $ nano fibonacci.sh
(kali㉿kali)-[~] $ chmod +x fibonacci.sh
(kali㉿kali)-[~] $ bash fibonacci.sh
Enter the number of terms for Fibonacci series: 5
Fibonacci Series up to 5 terms:
0 1 1 2 3

[kali㉿kali)-[~]
```

#### 4. Prime Number Display.

A screenshot of a terminal window titled "prime.sh ~". The window shows the source code for a shell script named "prime.sh". The code defines two functions: "is\_prime()" and "generate\_primes()". The "is\_prime()" function checks if a given number is prime by attempting to divide it by all numbers from 2 to its half. The "generate\_primes()" function prompts the user for the number of primes to generate, then iterates, calling "is\_prime()" for each number until the count reaches the user-specified value. The terminal window has a dark background with green text and a black border. The menu bar at the top includes "UTM", "File", "Edit", "Virtual Machine", "View", "Window", and "Help". The status bar at the bottom shows "Mon 19 Jan 8:42PM" and "5:12:29 dm1".

```
GNU nano 8.7 prime.sh ~
#!/bin/bash

is_prime() {
    local num=$1
    if [ "$num" -le 1 ]; then
        return 1
    fi
    for ((i=2; i<=num/2; i++)); do
        if ((num % i == 0)); then
            return 1
        fi
    done
    return 0
}

generate_primes() {
    read -p "Enter the number of prime numbers you want to display: " n
    if ! [[ "$n" =~ ^[0-9]+\$ ]] || [ "$n" -le 0 ]; then
        echo "Please enter a valid positive integer."
        exit 1
    fi
    count=0
    num=2
    echo "First $n prime numbers:"
    while [ "$count" -lt "$n" ]; do
        if is_prime "$num"; then
            echo -n "$num "
            ((count++))
        fi
        ((num++))
    done
    echo ""
}

generate_primes
```

#### Output:-

A screenshot of a terminal window titled "kali Linux". The window shows the execution of the "prime.sh" script. The user first runs "nano prime.sh" to edit the file, then "chmod +x prime.sh" to make it executable. Finally, they run "bash prime.sh" and enter "5" when prompted for the number of primes. The script outputs the first 5 prime numbers: 2, 3, 5, 7, 11. The terminal window has a dark background with white text and a black border. The menu bar at the top includes "File", "Edit", "Virtual Machine", "View", "Window", and "Help". The status bar at the bottom shows "Mon 19 Jan 8:43PM".

```
(kali㉿kali)-[~]
$ nano prime.sh

(kali㉿kali)-[~]
$ chmod +x prime.sh

(kali㉿kali)-[~]
$ bash prime.sh
Enter the number of prime numbers you want to display: 5
First 5 prime numbers:
2 3 5 7 11
```

## 5. Menu-Driven File Management.

```
GNU nano 8.7                               file.sh *
#!/bin/bash

display_menu() {
    echo "_____"
    echo " File Management Menu "
    echo "_____"
    echo "1. Create a New File"
    echo "2. Write to a File"
    echo "3. Append to a File"
    echo "4. Delete a File"
    echo "5. Exit"
    echo "_____"
}

create_file() {
    read -p "Enter the name of the new file: " filename
    touch "$filename"
    echo "File '$filename' created."
}

write_to_file() {
    read -p "Enter the name of the file to write to: " filename
    if [ ! -f "$filename" ]; then
        echo "File '$filename' does not exist."
        return
    fi
    echo "Enter content to write to the file (Ctrl+D to finish):"
    cat > "$filename"
    echo "Content written to '$filename'."
}

append_to_file() {
    read -p "Enter the name of the file to append to: " filename
    if [ ! -f "$filename" ]; then
        echo "File '$filename' does not exist."
        return
    fi
    echo "Enter content to append to the file (Ctrl+D to finish):"
    cat >> "$filename"
    echo "Content appended to '$filename'."
}

append_to_file()

read -p "Enter the name of the file to append to: " filename
if [ ! -f "$filename" ]; then
    echo "File '$filename' does not exist."
    return
fi
echo "Enter content to append to the file (Ctrl+D to finish):"
cat >> "$filename"
echo "Content appended to '$filename'."
```

The terminal window shows the initial state of a menu-driven file management script named 'file.sh'. The script defines several functions: 'display\_menu', 'create\_file', 'write\_to\_file', 'append\_to\_file', and 'delete\_file'. It then enters a loop where it displays the menu and reads user input to execute the selected option. The menu options are numbered 1 through 5, corresponding to creating a new file, writing to a file, appending to a file, deleting a file, and exiting the program respectively. The script uses the 'nano' text editor for the menu and file operations.

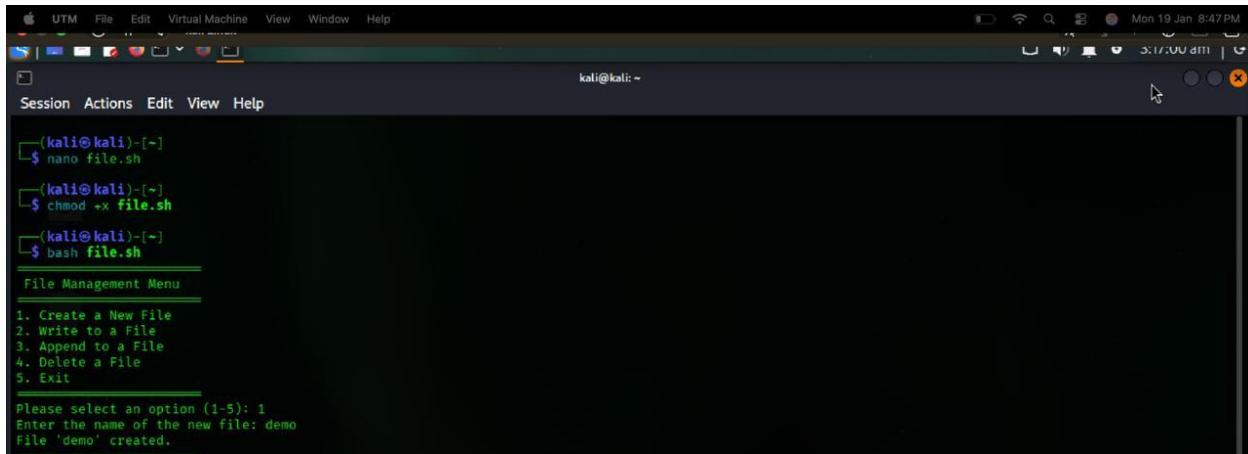
```
GNU nano 8.7                               file.sh *
echo "Content appended to '$filename'."
```

```
delete_file() {
    read -p "Enter the name of the file to delete: " filename
    if [ ! -f "$filename" ]; then
        echo "File '$filename' does not exist."
        return
    fi
    rm "$filename"
    echo "File '$filename' deleted."
}

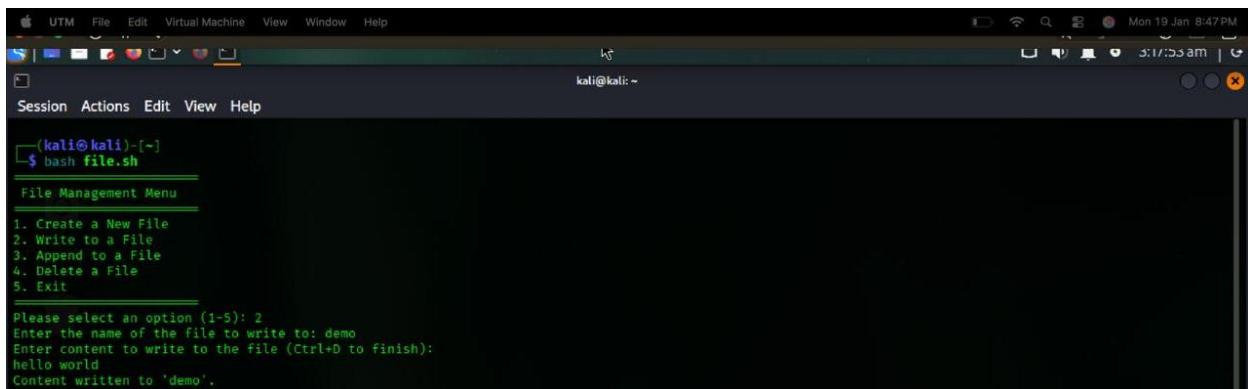
while true; do
    display_menu
    read -p "Please select an option (1-5): " option
    case $option in
        1)
            create_file
            ;;
        2)
            write_to_file
            ;;
        3)
            append_to_file
            ;;
        4)
            delete_file
            ;;
        5)
            echo "Exiting the program. Goodbye!"
            exit 0
            ;;
        *)
            echo "Invalid option! Please select a valid option between 1 and 5."
            ;;
    esac
done
```

The terminal window shows the execution of the menu-driven file management script. The 'append\_to\_file' function has been called, resulting in the message 'Content appended to '\$filename''. The script then enters a loop where it repeatedly displays the menu and reads user input. The 'case' statement handles five options: 1 for creating a new file, 2 for writing to a file, 3 for appending to a file, 4 for deleting a file, and 5 for exiting the program. The script uses the 'rm' command to delete files.

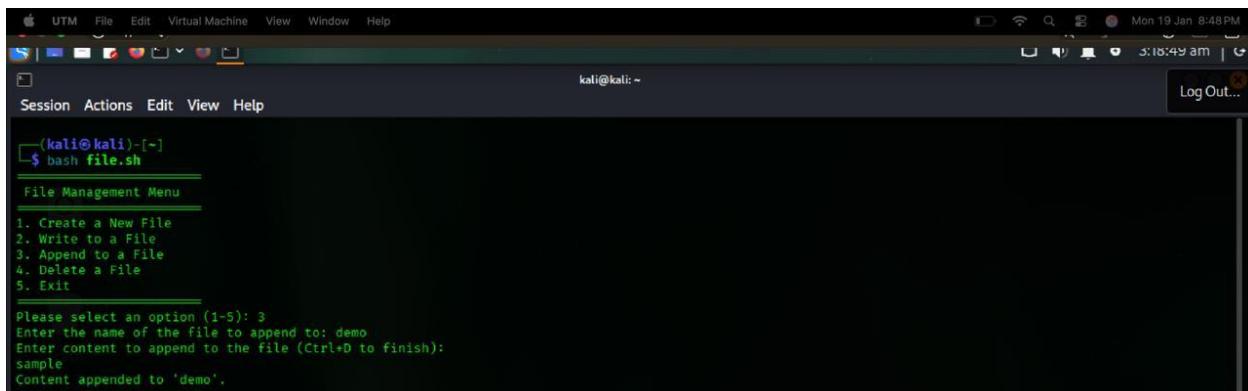
## Output:-



```
(kali㉿kali)-[~]
$ nano file.sh
(kali㉿kali)-[~]
$ chmod +x file.sh
(kali㉿kali)-[~]
$ bash file.sh
File Management Menu
1. Create a New File
2. Write to a File
3. Append to a File
4. Delete a File
5. Exit
Please select an option (1-5): 1
Enter the name of the new file: demo
File 'demo' created.
```



```
(kali㉿kali)-[~]
$ bash file.sh
File Management Menu
1. Create a New File
2. Write to a File
3. Append to a File
4. Delete a File
5. Exit
Please select an option (1-5): 2
Enter the name of the file to write to: demo
Enter content to write to the file (Ctrl+D to finish):
hello world
Content written to 'demo'.
```



```
(kali㉿kali)-[~]
$ bash file.sh
File Management Menu
1. Create a New File
2. Write to a File
3. Append to a File
4. Delete a File
5. Exit
Please select an option (1-5): 3
Enter the name of the file to append to: demo
Enter content to append to the file (Ctrl+D to finish):
sample
Content appended to 'demo'.
```

```
(kali㉿kali)-[~]
└─$ bash file.sh
File Management Menu
1. Create a New File
2. Write to a File
3. Append to a File
4. Delete a File
5. Exit
Please select an option (1-5): 4
Enter the name of the file to delete: demo
File 'demo' deleted.
```

```
(kali㉿kali)-[~]
└─$ bash file.sh
File Management Menu
1. Create a New File
2. Write to a File
3. Append to a File
4. Delete a File
5. Exit
Please select an option (1-5): 5
Exiting the program. Goodbye!
(kali㉿kali)-[~]
└─$
```