Department of Electrical & Computer Engineering Second Semester, 2022/2023 ENCS3130 Linux Laboratory

Shell Scripting Project

Statistics of Running Processes on Linux Machine

Names and ID's :

- **Raghad Afaghani 1192423**
- **Ebaa Taleeb 1203073**

Section : 2

Instructor : Dr. Aziz Qaroush

Date : 13.6.2023

Introduction

This project aimed to develop a shell script that provides statistics on the processes running on a Linux computer. The top command was used to obtain real-time information on the current processes, similar to the Windows' task manager. The shell script includes functions that display the top processes based on CPU or memory usage and prompts the user to input the number of processes they want to view. The script presents the information in a user-friendly format, providing valuable insights into the system's performance and enabling the user to identify and address any issues promptly. The report highlights the features and functionality of the shell script and explores the benefits of using shell scripts for process monitoring and management on Linux systems.

1) Printing the main menu and ask the user to select an option on the screen

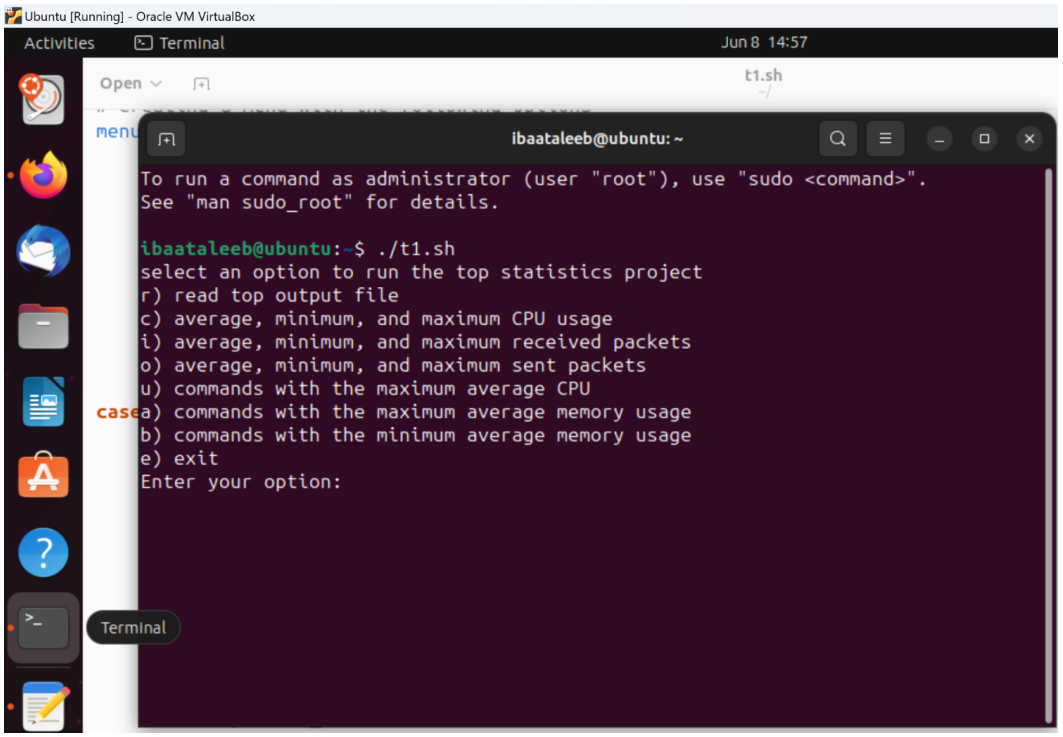
Code

```
#!/bin/bash

# creating a menu with the following options
menu() {
    echo "select an option to run the top statistics project"
    echo "r) read top output file"
    echo "c) average, minimum, and maximum CPU usage"
    echo "i) average, minimum, and maximum received packets"
    echo "o) average, minimum, and maximum sent packets"
    echo "u) commands with the maximum average CPU"
    echo "a) commands with the maximum average memory usage"
    echo "b) commands with the minimum average memory usage"
    echo "e) exit"
    read -p "Enter your option: " option
case $option in
        r)
            verifing_file
            ;;

        e)
            exiting
            ;;
        c)
            cpu_usage
            ;;
        i)
            packets_in
            ;;
        o)
            packets_out
            ;;
        u)
           prompt_for_integer
           ;;
        a)
           prompt_for_integer
           ;;
        b)
           prompt_for_integer
           ;;
        *)
            echo "Invalid option. Please try again."
esac
}
```

In the menu function,

- echo is used to display the list of options to the user. For example, echo "select an option to run the top statistics project" prints the message "select an option to run the top statistics project" to the console.
- The read command is also used to read input from the user. In the menu function, read -p "Enter your option: " option prompts the user to enter their choice of option and reads the input from the user into the option variable.
- The case statement is used to determine which option the user has selected and call the corresponding function.

Output



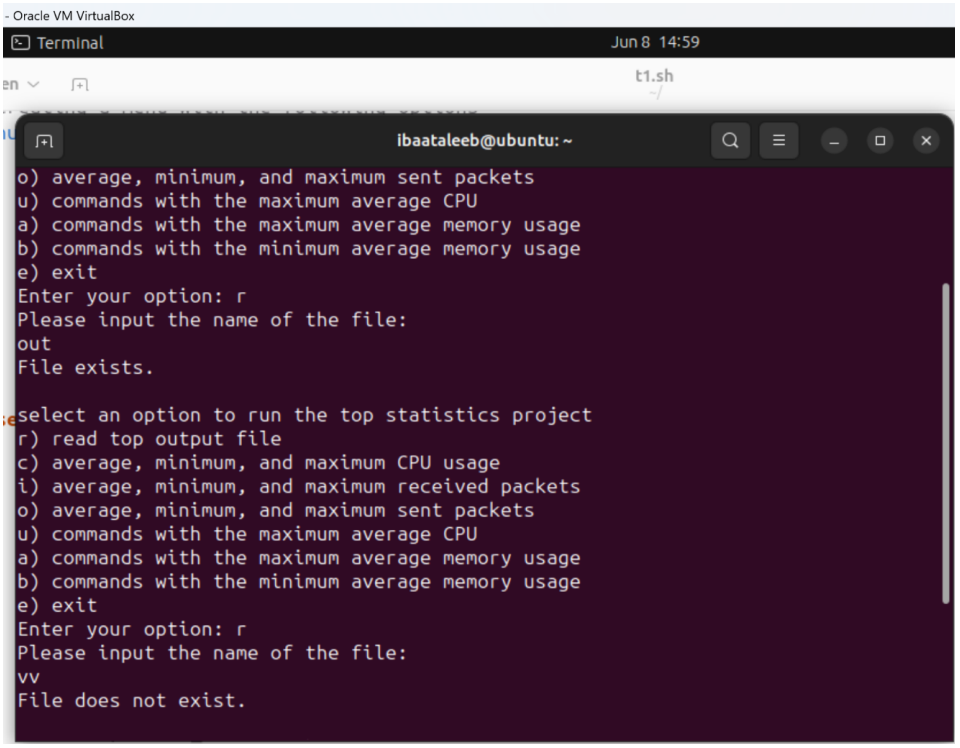
2) **If the user inputs the character 'r', the program will display the message "Please enter the file name" on the screen. The program will then check if the file exists. If the file does not exist, a message will be displayed on the screen saying "File does not exist" and the program will return to the main menu.**

Code

```
verifing_file() {
    echo "Please input the name of the file:"
    read -r filename

    if [[ -f $filename ]]; then
        echo "File exists."
    else
        echo "File does not exist."
    fi

    echo
    menu
}
```

The if statement is used to check if the file exists. The if statement is a conditional statement in Bash that is used to test a condition. In this case, the -f option of the test command is used to test if the file exists. If the file exists, the echo command is used to print the message "File exists." to the console. If the file does not exist, the echo command is used to print the message "File does not exist." to the console.

Output


- Oracle VM VirtualBox
Terminal
Jun 8 14:59
t1.sh
ibaataleeb@ubuntu: ~
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: r
Please input the name of the file:
out
File exists.
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: r
Please input the name of the file:
vv
File does not exist.

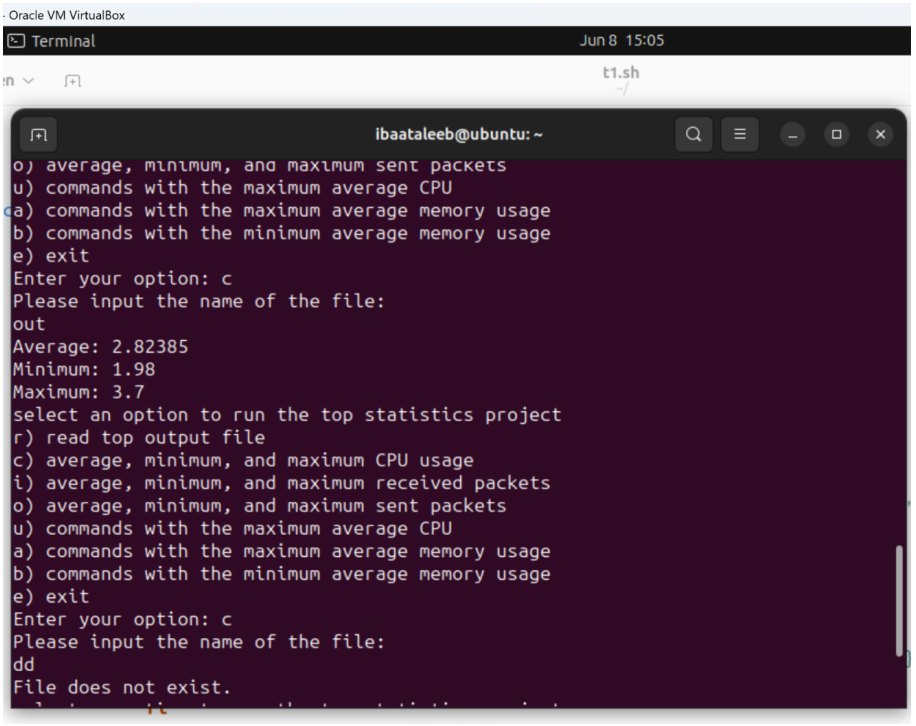
3) **When the user inputs 'c', the program will extract the CPU usage from the file for each time instance. Subsequently, the program will display the average, minimum, and maximum CPU usage on the screen. After printing this information, the program will display the main menu again.**

Code

```
cpu_usage() {
 echo "Please input the name of the file:"
 read -r filename
 if [[ -f $filename ]]; then
     num=0
     cpu_sum=""
     cpu_min=""
     cpu_max=3.98

     # Read the file line by line
     while IFS= read -r line; do
         # Extract CPU usage
         cpu_usage=$(grep -oE 'CPU usage: [0-9.]+' <<< "$line" | grep -oE '[0-9.]+')

         # If CPU usage is found, perform calculations
         if [[ -n $cpu_usage ]]; then
             cpu_sum=$(awk "BEGIN{ print $cpu_sum + $cpu_usage }")

             if [[ -z $cpu_max ]] || (( $(awk "BEGIN{ print $cpu_usage > $cpu_max }") )); then
                 cpu_max=$cpu_usage
             fi
             if [[ -z $cpu_min ]] || (( $(awk "BEGIN{ print $cpu_usage < $cpu_min }") )); then
                 cpu_min=$cpu_usage
             fi
             ((num++))
         fi
     done < "$filename"

     if ((num > 0)); then
         cpu_average=$(awk "BEGIN{ print $cpu_sum / $num }")
         printf "Average: $cpu_average\n"
         printf "Minimum: $cpu_min\n"
         printf "Maximum: $cpu_max\n"
     else
         printf "No CPU usage information found in the file.\n"
     fi
 else
     printf "File does not exist.\n"
 fi

 menu
}
```

- The echo command is used to prompt the user to enter the filename, and the read command is used to read the user's input into the filename variable.
- The -f option of the test command is used to check if the file exists. If the file exists, the function reads the contents of the file line by line using a while loop and performs calculations on the CPU usage data.
- The grep command is used to extract the CPU usage data from each line of the file.
- The awk command is then used to perform calculations on the CPU usage data, such as calculating the sum, average, minimum, and maximum CPU usage.
- The printf command is used to display the results to the console.
- If no CPU usage information is found in the file, the function prints the message "No CPU usage information found in the file." to the console. If the file does not exist, the function prints the message "File does not exist." to the console.
- Display the menu again and prompt the user to enter another option.

Output


Oracle VM VirtualBox
Terminal
Jun 8 15:05
t1.sh
ibaataleeb@ubuntu: ~
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: c
Please input the name of the file:
out
Average: 2.82385
Minimum: 1.98
Maximum: 3.7
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: c
Please input the name of the file:
dd
File does not exist.

4) **When the user inputs 'i', the program will extract the number of packets received over the network for each time instance. Then, the program will display the average, minimum, and maximum received packets on the screen. After printing this information, the program will display the main menu again.**

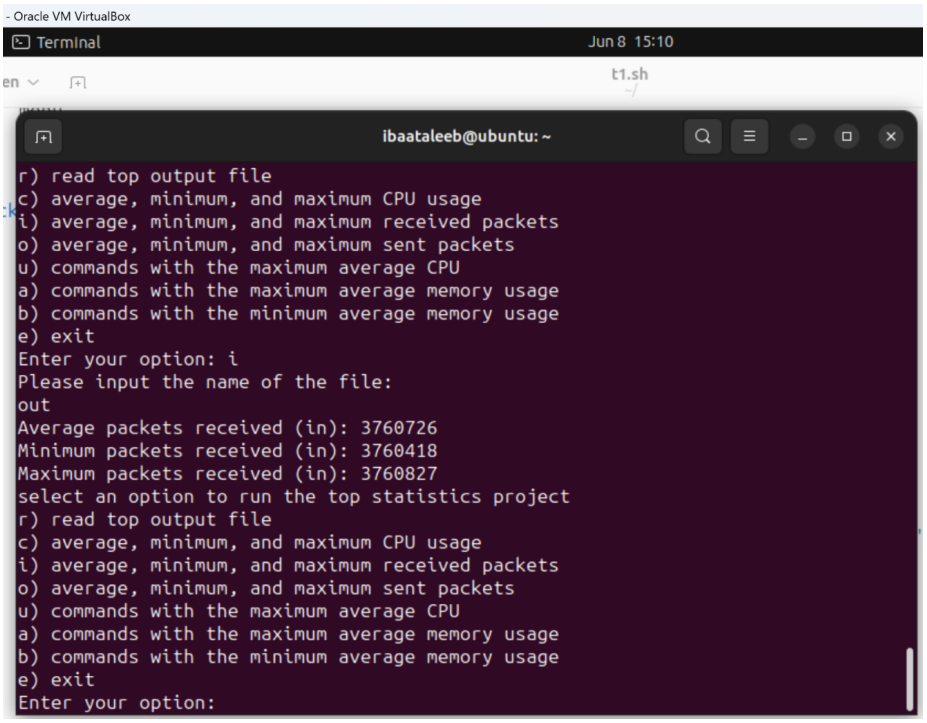
Code

```
packets_in() {
    echo "Please input the name of the file:"
    read -r filename
    if [[ -f $filename ]]; then

    packets=()
    total_packets=0
    min_packets=
    max_packets=

    while IFS= read -r line; do
        packets_in=$(grep -oE 'packets: [0-9]+/' <<< "$line" | grep -oE '[0-9]+' | head -1)
        if [ -n "$packets_in" ]; then
            packets+=("$packets_in")
            total_packets=$((total_packets + packets_in))

            if [ -z "$min_packets" ] || [ "$packets_in" -lt "$min_packets" ]; then
                min_packets=$packets_in
            fi

            if [ -z "$max_packets" ] || [ "$packets_in" -gt "$max_packets" ]; then
                max_packets=$packets_in
            fi
        fi
    done < "$filename"

    num_packets=${#packets[@]}
    if [ "$num_packets" -gt 0 ]; then
        average_packets=$((total_packets / num_packets))
        echo "Average packets received (in): $average_packets"
        echo "Minimum packets received (in): $min_packets"
        echo "Maximum packets received (in): $max_packets"
    else
        echo "No packets received (in) found in the file."
    fi
  else
        printf "File does not exist.\n"
    fi

    menu

}
```

- The echo command is used to prompt the user to enter the filename, and the read command is used to read the user's input into the filename variable.
- The -f option of the test command is used to check if the file exists. If the file exists, the function reads the contents of the file line by line using a while loop and extracts information about received packets from each line.
- The grep command is used to extract the received packet information from each line of the file. The received packet data is then stored in an array and calculations are performed on the data to calculate the average, minimum, and maximum received packets. The echo command is used to display the results to the console.

- If no received packet information is found in the file, the function prints the message "No packets received (in) found in the file." to the console. If the file does not exist, the function prints the message "File does not exist." to the console.
- Display the menu again and prompt the user to enter another option.

Output



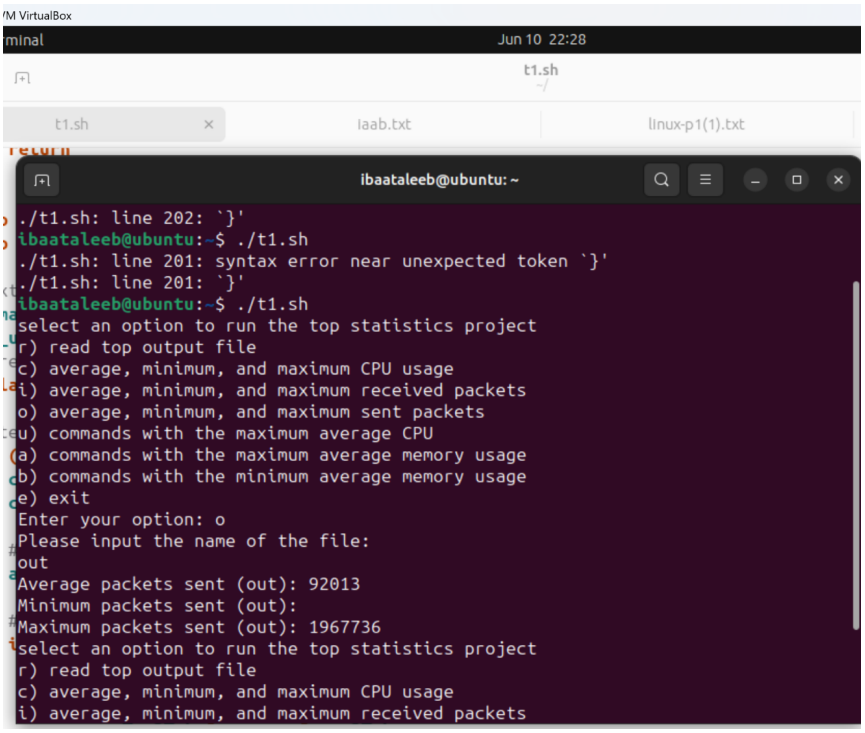
5) **When the user inputs 'o', the program will extract the number of packets sent over the network for each time instance. Then, the program will display the average, minimum, and maximum sent packets on the screen. After printing this information, the program will display the main menu again.**

Code

```
}
packets_out() {
    echo "Please input the name of the file:"
    read -r filename
    if [[ -f $filename ]]; then
        packets=()
        num=
        packet_sum=
        packet_min=1976584
        packet_max=

        # Read the file line by line
        while IFS= read -r line; do
            # Extract the number before "out"
            packet=$(grep -oE '([0-9]+)\/[0-9]+M out' <<< "$line" | grep -oE '^[0-9]+')

            # Update sum, minimum, and maximum values
            packet_sum=$((packet_sum + packet))
            if (( packet < packet_min )); then
                packet_min=$packet
            fi

            if (( packet > packet_max )); then
                packet_max=$packet
            fi

            ((num++))
        done < "$filename"

        # Calculate average
        packet_average=$((packet_sum / num))

        # Print the statistics
        echo "Average packets sent (out): $packet_average"
        echo "Minimum packets sent (out): $packet_min"
        echo "Maximum packets sent (out): $packet_max"
    else
        echo "File not found."
    fi

    menu


}
```

We define a function called "packets_out" that prompts the user to input a filename, reads the file line by line, extracts a number before the string "out" in each line, and calculates statistics based on the extracted numbers. The statistics calculated are the average, minimum, and maximum values of the extracted numbers. If the file doesn't exist, it prints "File not found."

Output


/M VirtualBox
minal
Jun 10 22:28
t1.sh
~/
t1.sh
laab.txt
linux-p1(1).txt
ibaataleeb@ubuntu: ~
./t1.sh: line 202: `}'
ibaataleeb@ubuntu:~$ ./t1.sh
./t1.sh: line 201: syntax error near unexpected token `}'
./t1.sh: line 201: `}'
ibaataleeb@ubuntu:~$ ./t1.sh
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: o
Please input the name of the file:
out
Average packets sent (out): 92013
Minimum packets sent (out):
Maximum packets sent (out): 1967736
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets

6) **In the event that the user inputs 'u':**

a. The software must request the user to input an integer value, m. Should the inputted value not be an integer, the software must display an error message.

b. The software must display on the screen m commands that possess the greatest maximum average CPU usage. Additionally, the software must exhibit the average CPU usage for each of these commands on the screen. Finally, the software must reprint the main menu.

Code

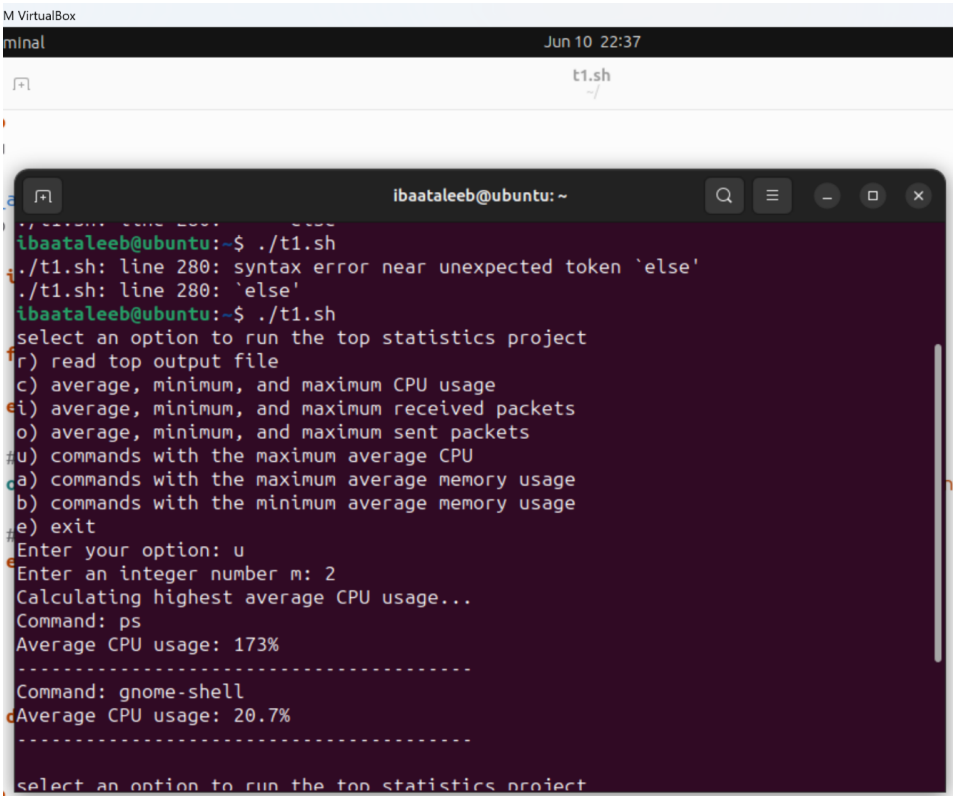
```
highest_average_cpu_commands() {
 read -p "Enter an integer number m: " m

        if [[  ! $m =~ ^[0-9]+$ ]]; then
            echo "Error Invalid input. Please enter an integer."
            continue
        fi

        echo "Calculating highest average CPU usage..."

        # Get the top m commands with the highest maximum average CPU usage
        output=$(ps -e -o comm,%cpu --sort=-%cpu --no-headers | awk '!seen[$1]++' | head -n "$m")

        # Display the commands with their average CPU usage
        echo "$output"| while read -r line; do
            command= $(echo "$line" | awk '{print $1}')
            cpu= $(echo "$line" | awk '{print $2}')
            printf "Command: $command"
            printf "Average CPU usage: $cpu%"
            printf "----------------------------------------"
        done


    echo
    menu
}
```

prompts the user to input an integer value m. The function then checks whether the input is a valid integer using a regular expression. If the input is not a valid integer, the function prints an error message and continues prompting the user to input a valid integer.

Once a valid integer is entered, the function proceeds to calculate the highest average CPU usage for the top m commands. It does this by using the ps command to list all running processes, sorting them by CPU usage in descending order, and using awk to remove duplicate command names. It then selects the top m processes and displays them along with their CPU usage.

The function uses a while loop to read each line of the output of the ps command, extracts the command and CPU usage values using awk, and prints them to the terminal along with a separator. Finally, the function calls another function named "menu", which is assumed to be defined elsewhere in the script or in another script.

Output


M VirtualBox
minal
Jun 10 22:37
t1.sh
~/
ibaataleeb@ubuntu: ~
ibaataleeb@ubuntu:~$ ./t1.sh
./t1.sh: line 280: syntax error near unexpected token `else'
./t1.sh: line 280: `else'
ibaataleeb@ubuntu:~$ ./t1.sh
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: u
Enter an integer number m: 2
Calculating highest average CPU usage...
Command: ps
Average CPU usage: 173%

Command: gnome-shell
Average CPU usage: 20.7%
---
select an option to run the top statistics project

7) **If the user enters 'a':**

a. The program should prompt you for an integer number m. If the entered value is not an integer, the program should print an error message.

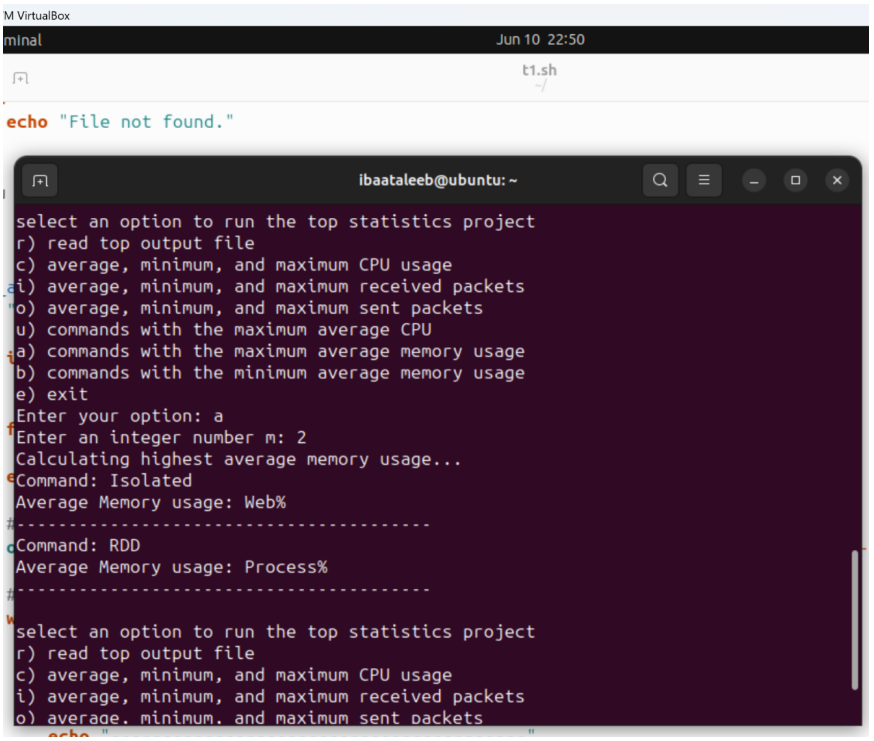
b. The program should print m commands that have the highest average memory usage on the screen. The program should also print the average memory usage for each of these commands on the screen. Also, print the main menu again.

Code

```
maximum_average_memory_commands() {
read -p "Enter an integer number m: " m

        if [[  ! $m = ~ ^[0-9]+$ ]]; then
            echo "Error Invalid input. Please enter an integer."
            continue
        fi

        echo "Calculating highest average memory usage..."

        # Get the top m commands with the highest average memory usage
        output=$(ps -e -o comm,%mem --no-headers | awk '!seen[$1]++' | sort -k2 -r | head -n "$m")

        # Display the commands with their average memory usage
        while read -r line; do
            command =$(echo "$line" | awk '{print $1}')
            mem= $(echo "$line" | awk '{print $2}')
            printf "Command: $command"
            printf "Average Memory usage: $mem%"
            printf "----------------------------------"
        done <<< "$output"

echo
menu
}
```

This code defines a function that prompts the user to input an integer value m. It then calculates and displays the top m processes with the highest average memory usage. The function uses the ps command to list all running processes, uses awk to remove duplicate command names, sorts the output based on memory usage in descending order, selects the top m processes, and displays them along with their memory usage.

Output


M VirtualBox
minal
Jun 10 22:50
t1.sh
~/
echo "File not found."
ibaataleeb@ubuntu: ~
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: a
Enter an integer number m: 2
Calculating highest average memory usage...
Command: Isolated
Average Memory usage: Web%

Command: RDD
Average Memory usage: Process%
---
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets

8) **In the case where the user inputs 'b':**

a. The program must ask the user for an integer value, m. In the event that the entered value is not an integer, the program should exhibit an error message.

b. The program must display m commands on the screen that have the lowest average memory usage. The program must also present the average memory usage for each of these commands on the screen. Finally, the program should reprint the main menu.

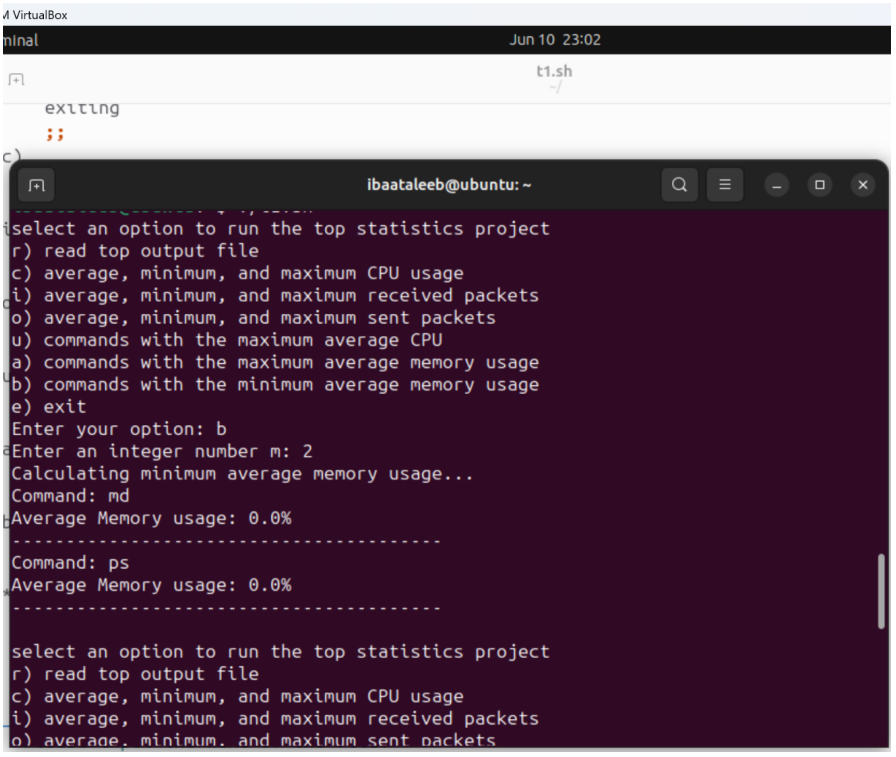
Code

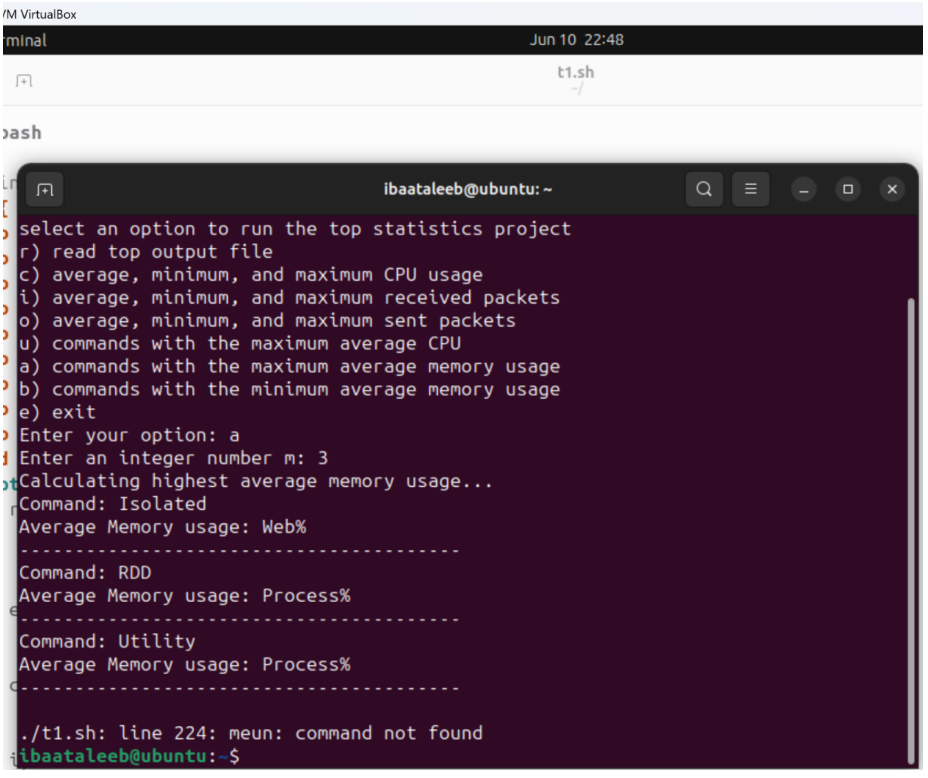
```
minimum_average_memory_commands() {
read -p "Enter an integer number m: " m

        if [[  ! $m =~  ^[0-9]+$ ]]; then
            echo "Error Invalid input. Please enter an integer."
            continue
        fi

        if [[ $choice == "a" ]]; then
            sorting_order="-r"  # Sort in descending order for highest average memory usage
            sorting_label="highest"
        else
            sorting_order=""  # Sort in ascending order for minimum average memory usage
            sorting_label="minimum"
        fi

        echo "Calculating $sorting_label average memory usage..."

        # Get the top m commands with the highest/minimum average memory usage
        output=$(ps -e -o comm,%mem --no-headers | awk '!seen[$1]++' | sort -k2 $sorting_order | head -n "$m")

        # Display the commands with their average memory usage
        while read -r line; do
            command =$(echo "$line" | awk '{print $1}')
            mem= $(echo "$line" | awk '{print $2}')
            printf "Command: $command"
            prinf "Average Memory usage: $mem%"
            printf "----------------------------------------"
        done <<< "$output"
        echo
        menu
        }
menu
```

We define a function that prompts the user to input an integer value m. It then calculates and displays the top m processes with either the highest or minimum average memory usage based on the value of the choice variable. The function uses the ps command to list all running processes, uses awk to remove duplicate command names, sorts the output based on memory usage in either ascending or descending order, selects the top m processes, and displays them along with their memory usage.

Output

.


VirtualBox
Jun 10 23:02
t1.sh
exiting
;;
ibaataleeb@ubuntu: ~
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: b
Enter an integer number m: 2
Calculating minimum average memory usage...
Command: md
Average Memory usage: 0.0%
Command: ps
Average Memory usage: 0.0%
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets


VM VirtualBox
Jun 10 22:48
t1.sh
ibaataleeb@ubuntu: ~
select an option to run the top statistics project
r) read top output file
c) average, minimum, and maximum CPU usage
i) average, minimum, and maximum received packets
o) average, minimum, and maximum sent packets
u) commands with the maximum average CPU
a) commands with the maximum average memory usage
b) commands with the minimum average memory usage
e) exit
Enter your option: a
Enter an integer number m: 3
Calculating highest average memory usage...
Command: Isolated
Average Memory usage: Web%
Command: RDD
Average Memory usage: Process%
Command: Utility
Average Memory usage: Process%
./t1.sh: line 224: meun: command not found
ibaataleeb@ubuntu:~$

9) **If the user enters ‘e’: the program should print on the screen “Are you sure you want to exist”. If the person inputs “yes”, the program ends. Otherwise, the program should return to the main menu.**

Code

```
exiting() {
    echo "Are you sure you want to exit? (yes/no)"
    read -r choice

    if [[ $choice == "yes" ]]; then
        echo "Exiting the program."
        exit 0
    elif [[ $choice == "no" ]]; then
        menu
    else
        echo "Invalid choice. Please try again."
        exit
    fi
}
```

- It echoes a prompt asking the user if they want to exit, with yes/no options.
- It reads the user's input using the read -r choice command and stores it in the choice variable.
- It then has an if/elif conditional:

-If the user enters yes, it echoes a message saying it's exiting and then calls exit 0 to actually exit the program
-If the user enters no, it calls the menu function, presumably to go back to the main menu.
-For any other input, it echoes an error message and calls exit to exit the program.

Output